

Contribución Académica

Los apuntes de electrónica digital representan un recurso valioso en la formación académica de los estudiantes, ya que consolidan los conocimientos teóricos y prácticos necesarios para el dominio de los sistemas digitales. Su estructura organizada facilita el aprendizaje progresivo, permitiendo a los alumnos comprender desde los fundamentos del álgebra booleana hasta el diseño y análisis de circuitos lógicos complejos.

Además, estos apuntes fomentan el desarrollo del pensamiento lógico y la capacidad de resolución de problemas, habilidades esenciales en la ingeniería y áreas afines. Su contenido, complementado con ejemplos, ejercicios y aplicaciones prácticas, no solo refuerza la teoría impartida en clase, sino que también sirve como material de consulta para el desarrollo de proyectos académicos y profesionales. En este sentido, los apuntes de electrónica digital contribuyen significativamente a la formación de futuros ingenieros y técnicos, brindándoles herramientas clave para enfrentar los desafíos del mundo tecnológico.

INDICE

UNIDAD 1	6
Fundamentos de sistemas digitales.....	6
1.1 Introducción a los sistemas digitales.....	6
1.2 Señales analógicas y digitales.	6
1.3 Relación entre los sistemas análogos y los sistemas digitales.	10
1.4 Sistemas numéricos.	14
1.4.1 Binario	15
1.4.2 Octal.....	17
1.4.3 Hexadecimal.....	18
1.5 Conversión entre sistemas numéricos.....	19
1.5.1 Conversión de decimal a binario.	19
1.5.2 Conversión de decimal al octal.	20
1.5.3 Conversión de decimal a hexadecimal.	21
1.5.4 Conversión de hexadecimal a binario.	22
1.5.6 Conversión de binario a octal.....	23
1.5.7 Conversión de binario a hexadecimal.....	24
1.6 Operaciones básicas con diferentes sistemas numéricos.	26
1.7 Códigos binarios y alfanuméricos.....	42
1.7.1 Código Gray	43
1.7.2 Código BCD (decimal codificado en binario).....	46
1.7.3 Código ASCII	48
1.7.4 Unicode	50
1.7.5 Código Hamming	51
Unidad 2.....	58
Compuertas lógicas y algebra booleana	58
2.1 Compuertas básicas.....	58
2.2 Familias lógicas	77
2.3 Teoremas y postulados del algebra booleana.	93
2.4 Simplificación de funciones booleanas.	100
2.4.2 Mapas de Karnaugh.	106
2.4.3 Método de Quine-McClusKey.....	114
UNIDAD 3	124

Circuitos combinacionales.	124
3.1 Procedimiento de diseño.....	125
3.2 Circuitos básicos combinacionales.	125
3.3 Simulación de los circuitos combinacionales.	134
3.3.1 Multiplexores.	137
3.3.2 Demultiplexores.	139
3.3.3 Decodificadores.....	142
3.3.4 Codificadores.	146
3.3.5 Indicadores numéricos (Display's)	149
3.4 Dispositivos lógicos programables.	155
3.5 Lenguajes de descripción de hardware (HDL).....	169
UNIDAD 4	213
Circuitos secuenciales.....	213
4.1 Máquinas de Mealy y Máquinas de Moore.	213
4.2 temporizadores.	215
4.2.1 LM555 modo monoestable.	217
4.2.2 LM555 Modo astable.	218
4.3 FLIP FLOPS.....	222
4.3.1 Flip Flop S - R	222
4.3.2 El Flip Flop JK	230
4.3.3 Flip Flop tipo D	234
4.3.4 Flip Flop tipo T.....	236
4.3.5 Flip Flop Maestro – Esclavo.	239
4.4 Diagramas y ecuaciones de estado.....	240
4.5 Circuitos síncrono y asíncronos.....	250
4.6 Circuitos secuenciales básicos.	262
4.6.1 Registros.....	262
4.6.2 Contadores.....	268
4.6.3 Memorias.....	283
4.7 Diseño de circuitos mediante lenguajes de descripción de hardware.	296
Unidad 5.....	359
Convertidores.....	359
5.1 Funcionamiento de DAC.	360

5.2 Diseño con lenguaje de descripción de hardware.....	369
5.3 Funcionamiento del ADC.	373
5.4 Aplicaciones.....	382
Referencias	392

UNIDAD 1

Fundamentos de sistemas digitales

1.1 Introducción a los sistemas digitales.

En la actualidad la palabra digital es de uso común debido a que la mayoría de las tecnologías utilizadas en el diseño de equipos que se manejan todos los días como son: las computadoras, celulares, equipos de entretenimiento, automóviles etcétera. Están diseñados con circuitos electrónicos digitales que procesan la información y los datos con señales discretas con dos niveles el uno(1) lógico para un nivel alto y un cero(0) lógicos para un nivel bajo.

Cuando se habla de un sistema digital se incluyen diferentes tópicos como son: el álgebra booleana, sistemas numéricos, compuertas lógicas, circuitos combinacionales, circuitos secuenciales, también los microcontroladores y los dispositivos lógicos programables como los FPGAs entre otros.

1.2 Señales analógicas y digitales.

Representaciones numéricas

En las actividades de cada día, en el trabajo en la ciencia en los negocios, trabajamos con cantidades es decir una representación numérica de algo cuyas unidades pueden ser: kilogramos, pesos, centímetros etc. Existen básicamente dos formas de representar estas cantidades y son: representación analógica y representación digital.

Representación analógica.

En este tipo de representación se muestra la cantidad en un indicador que va variando de forma continua por medio de otra cantidad que es proporcional a esta.

Los aparatos que muestran las cantidades analógicas han ido desapareciendo con el tiempo. Ejemplos de estos son:

Velocímetro de aguja en un automóvil antiguo.

Un velocímetro analógico se compone de una bobina, la cual se vincula con un cable. Esta velocidad de rotación se relaciona directamente con el movimiento de las ruedas. En el extremo opuesto del cable se encuentra un imán que genera una corriente eléctrica que provoca el movimiento de la aguja del velocímetro sobre una escala graduada.



Figura 1.1 Velocímetro analógico [1]

Termómetro de mercurio.

El termómetro de mercurio es un dispositivo utilizado para la medición de la temperatura, que se emplea comúnmente para cuantificar la temperatura de un objeto específico. Se le denomina termómetro de mercurio debido a que el mismo contiene un líquido de mercurio, el cual se dilata o contrae con la temperatura, la caratula se gradúa de tal manera que muestre la temperatura en función del movimiento del mercurio.



Figura 1.2 Termómetro de mercurio. [2]

Báscula de gancho.

Las básculas de gancho mecánicas operan a través de un sistema de muelle elástico. Estas básculas se benefician de la capacidad del muelle para deformarse frente a la carga aplicada, lo que facilita la determinación del peso del objeto suspendido en el gancho, la lectura del peso se hace en una carátula graduada en kilogramos o libras la cual depende del desplazamiento del gancho.



Figura 1.3 Báscula de gancho. [3]

Las cantidades analógicas aquí representadas tienen la principal característica que el intervalo de valores que representa es continuo, es decir cualquier valor que esté en este intervalo puede ser medido con la tabulación adecuada en la caratula del instrumento.

Representaciones digitales.

En este tipo de representación los valores de las cantidades no son proporcionales a otras, simplemente se representan por dígitos como en los siguientes ejemplos:

Velocímetro digital

Los velocímetros digitales son una evolución más contemporánea y emplean pantallas digitales para representar la velocidad en términos numéricos. En estos dispositivos se puede hacer una lectura más exacta, también es posible entregar información adicional como el desplazamiento total y el tiempo transcurrido.



Figura 1.4 Velocímetro digital. [4]

Termómetro digital.

Estos aparatos utilizan dispositivos transductores que generan un pequeño voltaje o corriente en función de la temperatura a la que se somete y un conjunto de circuitos electrónicos para cambiar el voltaje en números equivalentes a la temperatura actual y se muestran en la caratula del termómetro.



Figura 1.5 Termómetro digital. [5]

Báscula digital.

Una báscula digital se define como un aparato de medición que emplea sensores electrónicos para identificar y exhibir con exactitud el peso de un objeto o sustancia en una pantalla digital. las básculas digitales transforman la fuerza del peso en una señal eléctrica y en conjunto con otros circuitos electrónicos muestran el peso en forma de dígitos por lo que hay una mayor precisión de la lectura.



Figura 1.6 Báscula digital. [6]

Las cantidades representadas en forma digital en los ejemplos anteriores son discretas, es decir no se pueden representar todos los valores existentes en el intervalo que marca el aparato por lo que las cantidades digitales se consideran discontinuas.

Con lo que visto anteriormente podemos decir que las cantidades analógicas, aunque son continuas en el intervalo de valores que marca el equipo su medición puede ser no precisa, eso depende de la persona que hace la medición y la posición que guarda el equipo con respecto quien hace la medición y si maneja bien el equipo o no entre otros factores, por otro lado, las cantidades digitales son discretas es decir hay valores que no se pueden leer en el intervalo que marca el equipo, pero no hay error en la lectura que se hace. Por ejemplo, en la figura 1.1 el velocímetro analógico tiene un intervalo de lectura de 0 a 200 Km/H cualquier valor de velocidad que esté en ese rango puede ser leída, por ejemplo, 1.23 Km/H basta que la caratula este bien rotulada, en cambio en la figura 1.4 velocímetro digital su

intervalo de valores es de 0 a 199 Km/H sin embargo no podemos leer una velocidad 1.236 Km/H.

Entonces las señales que maneja un sistema pueden ser del tipo analógica o digital y la principal diferencia es que la señal analógica es continua en el tiempo y la señal digital es discreta es decir paso por paso.

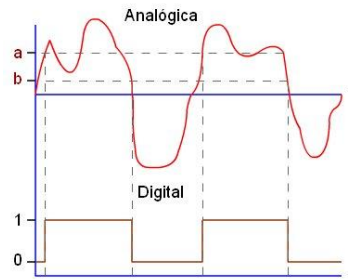


Figura 1.7 Señal analógica y señal digital. [7]

En las cantidades que se muestran a continuación ponga una A para las cantidades analógica y una D a las digitales:

1. Década de resistencias. ()
2. Flotador de un tinaco. ()
3. Temperatura de un vaso con agua. ()
4. Apagador de una habitación. ()
5. Granos de arena en una playa. ()
6. potenciómetro d 1K Ω . ()

1.3 Relación entre los sistemas análogos y los sistemas digitales.

Sistema electrónico.

Aparato diseñado para la extracción, almacenamiento, transporte y procesamiento de la información proveniente de una señal.

Se compone de tres fases:

1) Fase inicial (sensor o transductor).

Se encarga de la transformación de las señales provenientes del mundo físico, tales como presión, temperatura, sonido, entre otros, en señales de voltaje o corriente.

2) Fase de procesamiento de información.

Conjunto de circuitos electrónicos que tienen como objetivo la transformación, almacenamiento y procesamiento de las señales de voltaje y corriente que emanan de los transductores.

Procedimiento de salida (actuador).

Responsable de transformar las señales de corriente o voltaje en señales de utilidad física. Por ejemplo: un dispositivo de visualización (display) que registra la temperatura.

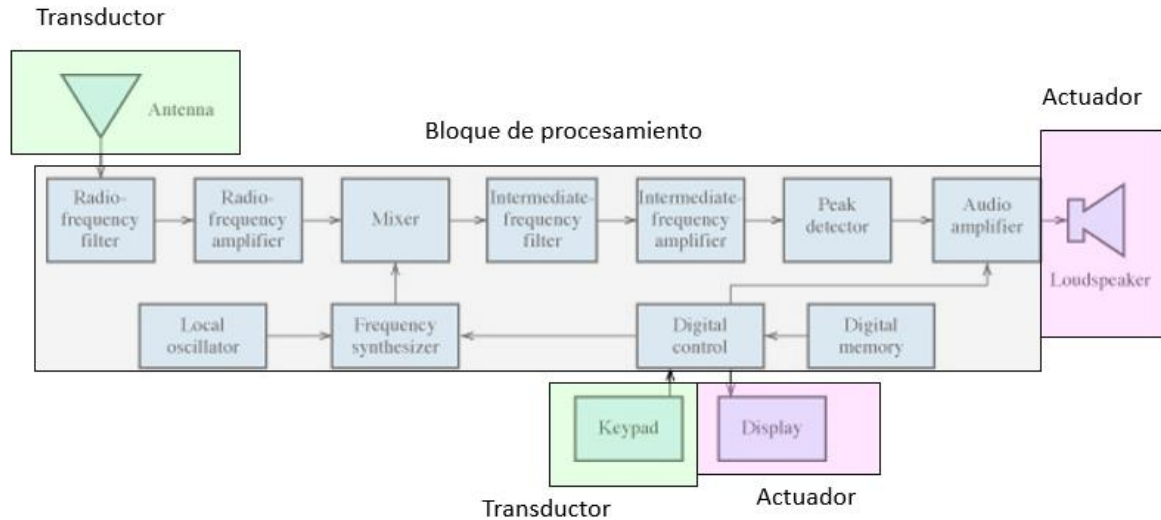


Figura 1.8 Diagrama a bloques de un sistema electrónico.

Sistema digital

Un sistema digital consta de un conjunto dispositivos electrónicos y en algunos casos software que procesan, almacenan y transmiten información en formato digital, es decir, utilizando valores discretos, normalmente unos y ceros.

Este tipo de sistemas se diseñan para efectuar operaciones lógicas, aritméticas y de control a una gran velocidad y con gran precisión.

También existen sistemas digitales neumáticos y mecánicos.

Algunos ejemplos de estos sistemas son:

Computadoras, teléfonos celulares, pantalla, sistemas de audio y video, equipos de control (PLC'S) etc.

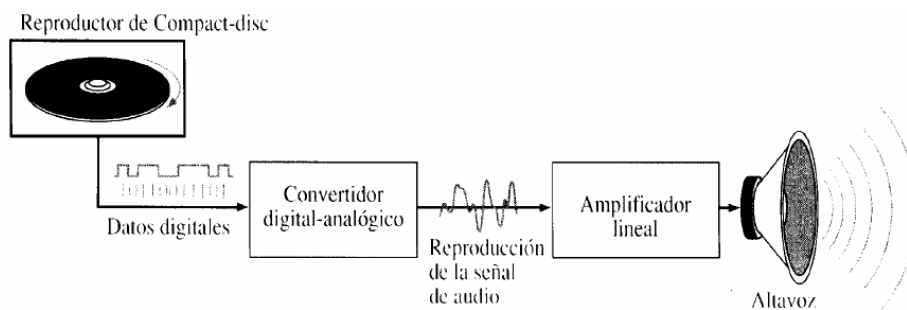


Figura 1.9 Sistema digital reproductor de discos compactos. [8]

Sistema analógico

Es aquel que está diseñado con un conjunto de circuitos electrónicos para el manejo y procesamiento de señales continuas en el tiempo es decir señales analógicas.

Ejemplo de estos sistemas son tocadiscos de acetato, donde la aguja sigue las ondas grabadas en el vinilo sin saltos, reproduciendo el sonido de forma fluida, grabadora y reproductora de cintas magnéticas, control de iluminación de una habitación etc.

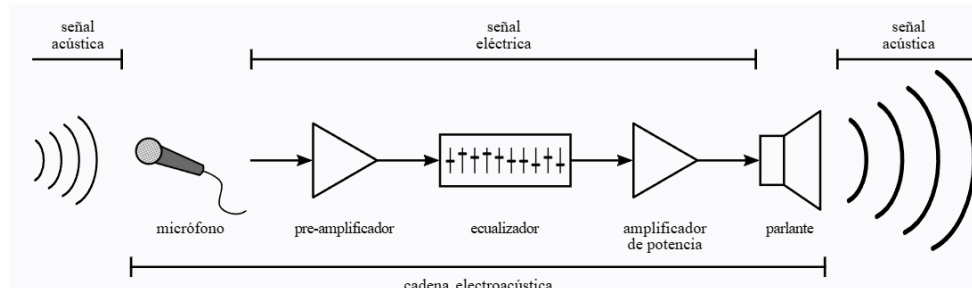


Figura 1.10 Sistema analógico, amplificador de audio. [9]

Ventajas de los sistemas digitales.

- Se diseñan de una manera más fácil ya que se cuenta con una gran cantidad de circuitos electrónicos de conmutación para estas aplicaciones incluyendo microprocesadores, microcontroladores y sistemas embebidos.
- Facilidad de almacenamiento y recuperación de información. Existen muchos medios para almacenar la información digital como lo son las cintas magnéticas, discos compactos, tarjetas perforadas, circuitos semiconductores de memorias etc.
- Mayor precisión y exactitud ya que estos sistemas trabajan con sólo dos valores (0 y 1) lo que permite minimizar los errores y hacer cálculos más rápidos y precisos.
- Se puede reprogramar la operación del sistema sin cambiar el circuito.
- Les afecta de manera mínima el ruido eléctrico ya que los circuitos digitales tienen un margen de ruido bastante amplio por lo que no se pierde sus valores de alto o bajo, es decir 1 o 0.
- El espacio que ocupa un circuito digital es menor que el de un circuito analógico por lo que se pueden fabricar más circuitos digitales en un solo circuito integrado.

Limitaciones de los sistemas digitales.

Las desventajas de los sistemas digitales se pueden decir que son dos:

- Las variables físicas del mundo son básicamente analógicas(presión, velocidad, temperatura etc.) por lo tanto, se tienen que discretizar es decir convertirlas a señales digitales lo que implica la pérdida de pequeñas porciones de la señal original.
- Las técnicas de conversión para procesar las variables físicas requieren de un determinado tiempo para representar su valor analógico a su equivalente digital.

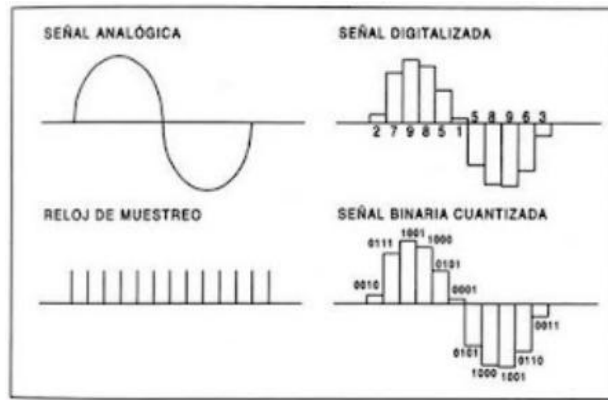


Figura 1.11 Conversión de una señal analógica a digital. [10]

Para obtener el mejor resultado en los sistemas digitales se deben tener en cuenta los siguientes pasos:

- Utilizar el mejor transductor para de la variable física que se requiere controlar, este entregará una señal analógica eléctrica(corriente o voltaje).
- Usar el mejor convertidor analógico a digital (ADC) que cumpla con los requisitos de velocidad para la variable que se desea controlar.
- Hacer uso de microprocesadores, microcontroladores o equipo de cómputo con el software que nos permita controlar la variable.
- Utilizar el convertidor de señal digital a analógica (DAC) para aplicarla a los actuadore si es necesario en nuestro sistema.

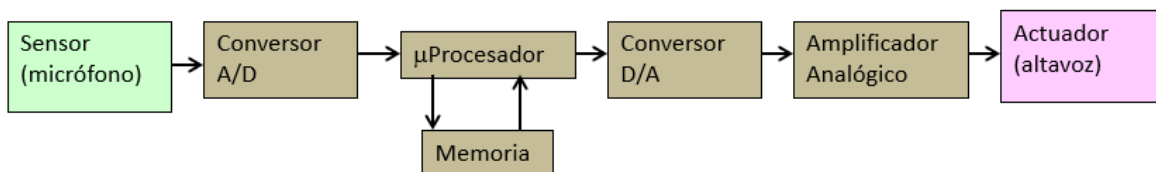


Figura 1.12 Etapas para el manejo digital de una variable analógica.

1.4 Sistemas numéricos.

Un sistema numérico está formado por un conjunto de símbolos y reglas para representar cantidades, existen diferentes sistemas que son utilizados en los sistemas digitales, los más utilizados son:

Decimal
Binario
Octal
Hexadecimal

Decimal

Este sistema está compuesto por diez símbolos o dígitos(dígito se deriva de la palabra dedo) los símbolos son: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9; por lo que se le conoce como sistema de valor posicional base 10, ya que cada símbolo o dígito toma su valor en función de la posición que guarda en la cantidad que se está representando(unidades, decenas, centena etc.).

En el sistema decimal en el número **945.32**₁₀ el dígito menos significativo es el de la derecha (5) y se representa como LSD (dígito menos significativo) y el de la izquierda (9) es el dígito más significativo (MSD), el (10) representa la base del sistema, es importante que se indique el sistema numérico al que pertenece la cantidad.

El valor de cada dígito se obtiene multiplicando el dígito por la base(10) elevada a la potencia del lugar que guarda el dígito en la cantidad, para la parte entera del número las unidades es la posición cero el siguiente dígito la uno el siguiente la dos y así sucesivamente, para la fracción las décimas es la posición menos 1 la siguiente menos dos y así sucesivamente.

En el sistema decimal el número **945.32**₁₀, significa:

9 centenas + **4** decenas + **5** unidades + 3 décimas + 2 centésimas, es decir:

$9 \cdot 10^2 + 4 \cdot 10^1 + 5 \cdot 10^0 + 3 \cdot 10^{-1} + 2 \cdot 10^{-2}$ o, lo que es lo mismo:

$900 + 40 + 5 + 0.3 + 0.02 = 945.32_{10}$

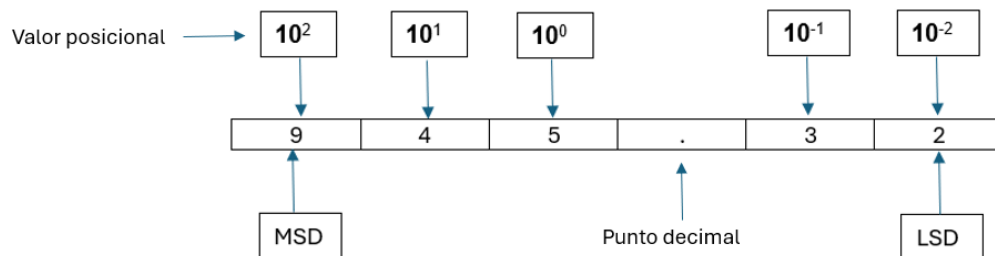


Figura 1.13 Representación del número decimal 945.32

Se puede obtener el valor de cualquier número decimal multiplicando cada dígito por su valor posicional que le corresponda y sumando los valores de todos los dígitos.

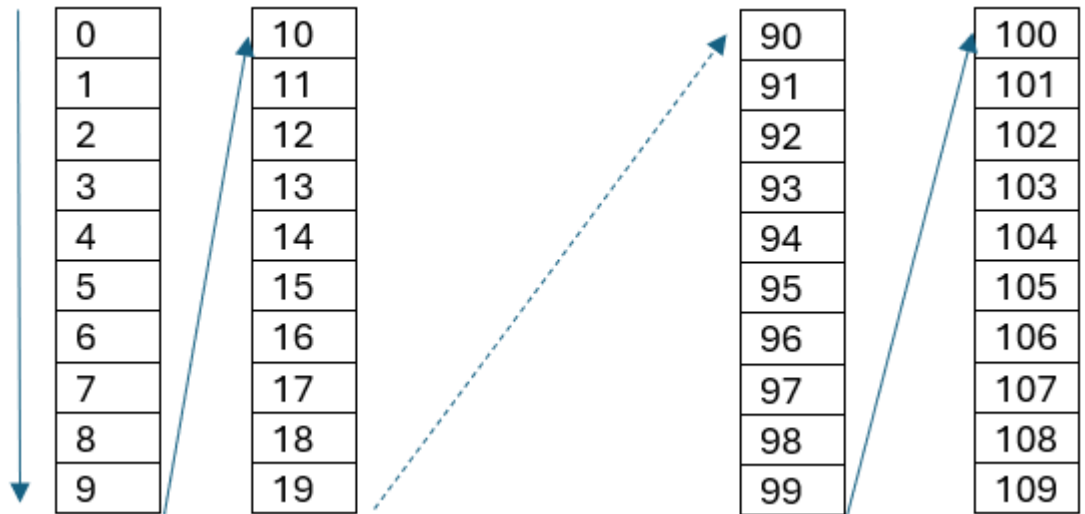


Figura 1.14 conteo en el sistema numérico decimal.

Para el conteo en el sistema decimal, las unidades se incrementan de uno en uno, las decenas se incrementan en uno cada 10 conteos de las unidades, las centenas cada diez conteo de las decenas y así sucesivamente. Se puede hacer un conteo máximo de $10^n - 1$ iniciando con el conteo en cero, n es el número de dígitos, si se tienen 3 dígitos el máximo valor de conteo será 999, es decir de 000 a 999.

1.4.1 Binario

El sistema binario es de valor posicional base 2 ya que sólo cuenta con dos símbolos o dígitos binarios (bits) que son el 0 y el 1, es el más utilizado en los sistemas digital ya que con sólo dos bits podemos representar cualquier cantidad.

Este sistema sigue las mismas reglas que el sistema decimal lo que cambia es la base (2), el valor de cada bit o dígito binario también depende de la posición que guarde en la cantidad que está representando, se multiplica el dígito binario (bit) por 2 elevado a la potencia positiva para la parte entera iniciando en cero de derecha a izquierda y potencia negativa para la fracción iniciando en menos uno de izquierda a derecha.

En el sistema binario en el número $1\ 1\ 0\ 1\ .\ 1\ 0\ 1_2$ el dígito binario (bit) menos significativo es el de la derecha LSB y el dígito más significativo es el de la izquierda MSB.

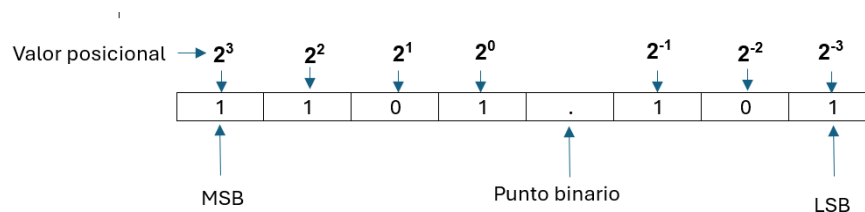


Figura 1.15 Representación del número binario 1101.101

El numero binario $1\ 1\ 0\ 1\ .1\ 0\ 1_2$ representa el valor decimal siguiente:

$1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3}$, es decir:

$$1 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 + 1 \cdot 0.5 + 0 \cdot 0.25 + 1 \cdot 0.125$$

$$8 + 4 + 0 + 1 + 0.5 + 0 + 0.125 = 13.625$$

$$1\ 1\ 0\ 1\ .1\ 0\ 1_2 = 13.625_{10}$$

Cuando sumamos el valor posicional del de cada dígito binario lo que obtenemos es su equivalente en decimal.

Conteo en binario

Para hacer un conteo binario de n bits partiendo de la cuenta cero, es decir todos los bits estarán inicialmente en cero, el bit menos significativo alterna su valor por cada cuenta, el siguiente bit alternara su valor cada vez que el bit anterior cambie de 1 a 0, el siguiente bit alternada su valor cada que los dos bits anteriores bits cambien de 1 a cero simultáneamente, el cuarto bit cambiara de estado cada que los tres bits anteriores cambien simultáneamente de unos a ceros simultáneamente, el proceso se repite par los siguientes bits. Se puede observar que el primer bit cambia por cada conteo, el segundo cada dos conteos el tercero cada cuatro conteos el cuarto cada 8 conteos etc.

Número de cuentas que se requieren en cada bit para alternar su valor.	2^3 8	2^2 4	2^1 2	2^0 1		Conteo en decimal
	0	0	0	0	→	0
	0	0	0	1	→	1
	0	0	1	0	→	2
	0	0	1	1	→	3
	0	1	0	0		4
	0	1	0	1		5
	0	1	1	0		6
	0	1	1	1		7
	1	0	0	0		8
	1	0	0	1		9
	1	0	1	0		10
	1	0	1	1		11
	1	1	0	0		12
	1	1	0	1	→	13
	1	1	1	0	→	14
	1	1	1	1	→	15

Figura 1.16 Conteo en el sistema numérico binario.

1.4.2 Octal

El sistema numérico octal es de valor posicional base 8 ya que está compuesto por ocho símbolos o dígitos que son: 1, 2, 3, 4, 5, 6, 7

Para saber qué número decimal representa un número octal se sigue la misma regla que en el sistema decimal, es decir se multiplica cada dígito por la base, en este caso 8 elevada a la potencia que le pertenezca de acuerdo a su posición en el número, para la parte entera el dígito menos significativo será la potencia cero el siguiente potencia uno y así sucesivamente, para la parte de la fracción, la potencia será negativa, para el primer dígito la potencia será menos uno el siguiente dígito menos dos y así sucesivamente.

Para determinar el valor decimal del número octal 265.3_8 se hace lo siguiente:

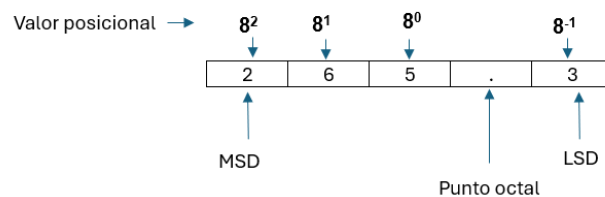


Figura 1.17 Representación del número octal 265.3_8

El número octal 265.3_8 representa el valor decimal siguiente:

$$265.3_8 = 2 \cdot 8^2 + 6 \cdot 8^1 + 5 \cdot 8^0 + 3 \cdot 8^{-1}$$

$$2 \cdot 64 + 6 \cdot 8 + 5 \cdot 1 + 3 \cdot 0.125 = 181.375_{10}$$

$$265.3_8 = 181.375_{10}$$

Al sumar el valor posicional de cada dígito octal lo que se obtiene es su equivalente en decimal.

Conteo en octal

Para hacer el conteo en octal de n dígitos, el primer dígito LSD se incrementa por cada conteo hasta llegar al 7 y se regresa a 0 repitiendo el conteo de este dígito, el segundo dígito se incrementará en uno cuando el primer dígito pase de 7 a 0, el tercer dígito se incrementará en uno cuando los dígitos uno y dos pasen simultáneamente de 77 a 00, el tercer dígito se incrementará en uno cuando los tres dígitos anteriores pasen de 777 a 000, el proceso se repite para los siguientes dígitos.

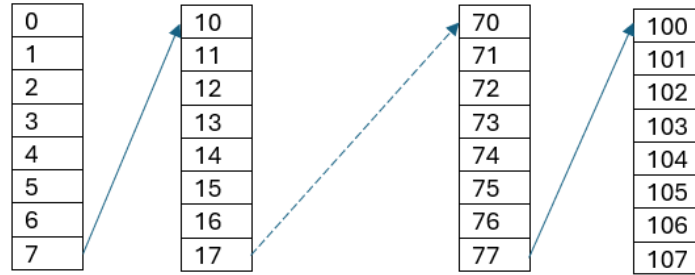


Figura 1.18 Conteo en el sistema numérico octal.

1.4.3 Hexadecimal

El sistema numérico hexadecimal es de valor posicional base 16 ya que está compuesto por dieciséis símbolos, entre dígitos y letras, estos son: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y la F. El valor que toma cada símbolo depende la posición que tenga en la cantidad que está representando, se multiplica el dígito o letra por la base 16 elevada a la potencia 0 si es el primer símbolo de la parte entera de la derecha antes del punto hexadecimal, a la potencia 1 si es el segundo a la potencia 2 si es el tercero y así sucesivamente, para el primer símbolo después del punto hexadecimal las potencias son negativas el primer símbolo se multiplica por la base 16 elevada a la potencia menos 1 el segundo a la menos 2 y así sucesivamente. Para encontrar el valor decimal de la cantidad hexadecimal las letras toman los siguientes valores: A = 10, B = 11, C = 12, D = 13, E = 14 y F = 15.

El número hexadecimal 2B3D_H también que puede representar 2B3D₁₆.

Y representa el siguiente número decimal:

$$2B3D_H = 2 \cdot 16^3 + B \cdot 16^2 + 3 \cdot 16^1 + D \cdot 16^0$$

Los valores de las letras son: B = 11 y D = 13.

$$2B3D_H = 2 \cdot 4096 + 11 \cdot 256 + 3 \cdot 16 + 13 \cdot 1 = 11069_{10}$$

$$2B3D_H = 11069_{10}$$

De esta forma se puede obtener el equivalente de cualquier número hexadecimal a decimal.

Conteo en hexadecimal.

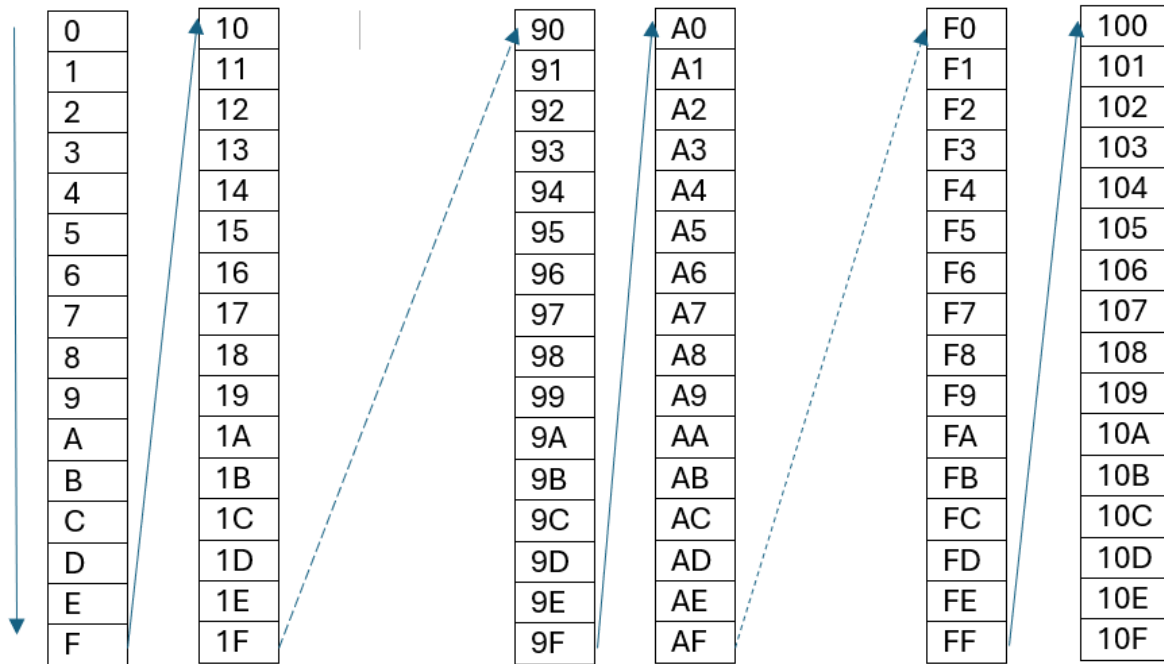


Figura 1.19 Conteo en hexadecimal.

1.5 Conversión entre sistemas numéricos.

Para hacer la conversión entre diferentes sistemas numéricos deben seguirse una serie de reglas sencillas, es importante saber pasar de un sistema numérico a otro de una manera rápida por lo que se debe conocer a cuál sistema conviene pasar primero, ya que se puede pasar a todos los sistemas antes de llegar al sistema numérico al que se quiere cambiar.

1.5.1 Conversión de decimal a binario.

Para hacer la conversión de decimal a binario se toma la parte entera y se divide entre 2 (base dos) las veces que sea necesario hasta que el residuo sea 0 o 1, en el cociente el ultimo dígito binario obtenido será el bit más significativo (MSB) y el primer dígito binario del cociente será el bit menos significativo (LSB). Para la fracción del número decimal, se multiplica por separado por la base 2, se toma la parte entera del producto la cual puede ser 0 o 1 y primer bit obtenido será el bit más significativo de la fracción, esto se repite hasta que el producto sea 0 o al número de bits que se desee aproximar si la fracción del número decimal no es potencia de 2, el ultimo bit obtenido del producto será el bit menos significativo de la fracción.

Proceso para convertir el número 78.62510 a binario.

Se toma la parte entera y se divide entre la base del sistema numérico binario que es el 2.

$$\begin{array}{l} 78 \div 2 = 39 \text{ Resto: } 0 \quad \uparrow \text{ LSB} \\ 39 \div 2 = 19 \text{ Resto: } 1 \\ 19 \div 2 = 9 \text{ Resto: } 1 \\ 9 \div 2 = 4 \text{ Resto: } 1 \\ 4 \div 2 = 2 \text{ Resto: } 0 \\ 2 \div 2 = 1 \text{ Resto: } 0 \\ 1 \div 2 = 0 \text{ Resto: } 1 \quad \text{MSB} \end{array}$$

Ya se tiene la partera entera en binario del número decimal, ahora se trabaja con la fracción:

$$\begin{array}{l} 0.625 \times 2 = 1.25 \quad \text{parte entera } 1 \quad \text{MSB de la fracción} \\ 0.25 \times 2 = 0.5 \quad \text{parte entera } 0 \\ 0.5 \times 2 = 1 \quad \text{parte entera } 1 \quad \text{LSB de la fracción} \end{array}$$

Por último, se pone la igualdad de los dos números el decimal y el binario.

$$78.62510 = 1001110.1012$$

Es importante poner la base a la que pertenece cada número ya que cualquiera de ellos se puede interpretar como decimal ya que el decimal contiene el 0 y el 1 del sistema binario.

1.5.2 Conversión de decimal al octal.

El procedimiento para esta conversión es el mismo que para el binario, sólo que para la división se divide entre 8 que es la base del sistema al que queremos convertir el número decimal, de igual manera se hace para la fracción se multiplica por 8, los dígitos obtenidos tanto en la parte entera como en la fracción sigue la misma regla, en la división el primer dígito obtenido será el menos significativo y el ultimo será el más significativo, en la fracción el primer dígito obtenido será el más significativo y el ultimo el menos significativo.

Para convertir el número 385.87_{10} a octal se hace lo siguiente:

Se trabaja primero la parte entera.

$$\begin{array}{l} 385 \div 8 = 48 \text{ Resto: } 1 \quad \uparrow \text{ LSD} \\ 48 \div 8 = 6 \text{ Resto: } 0 \\ 6 \div 8 = 0 \text{ Resto: } 6 \quad \text{MSD} \end{array}$$

Ahora se trabaja la fracción.

$$0.87 \times 8 = 6.96 \text{ parte entera } 6 \text{ MSD}$$

$$0.96 \times 8 = 7.68 \text{ parte entera } 7$$

$$0.68 \times 8 = 5.44 \text{ parte entera } 5 \text{ LSD}$$

Como la fracción no es potencia de 8 se aproxima a la cantidad de dígitos que más convenga.

Entonces:

$$385.87_{10} \cong 601.675_8$$

1.5.3 Conversión de decimal a hexadecimal.

El procedimiento de conversión de decimal a hexadecimal es el mismo que los anteriores sistemas, pero la base para la división es 16 que es la del sistema a la que se quiere convertir el número decimal, de la misma forma se hace para la fracción, se multiplica por 16, los dígitos obtenidos tanto en la parte entera como en la fracción sigue la misma regla, en la división el primer dígito obtenido será el dígito menos significativo y el último será el más significativo, en la fracción el primer dígito obtenido será el más significativo y el último el menos significativo.

Para convertir el número 924.45_{10} a hexadecimal se procede de la siguiente forma:

Primero se trabaja la parte entera.

$$924 \div 16 = 57 \text{ Resto: } 12 \rightarrow C \text{ LSD}$$

$$57 \div 16 = 3 \text{ Resto: } 9 \rightarrow 9$$

$$3 \div 16 = 0 \text{ Resto: } 3 \rightarrow 3 \text{ MSD}$$

Recuerde que cuando el cociente sea mayor a 9, se cambia por la letra a la que corresponda: 10 = A, 11 = B, 12 = C, 13 = D, 14 = E y 15 = F.

Ahora la parte de la fracción.

$$0.45 \times 16 = 7.2 \text{ parte entera } 7 \text{ MSD}$$

$$0.2 \times 16 = 3.2 \text{ parte entera } 3$$

$$0.2 \times 16 = 3.2 \text{ parte entera } 3 \text{ LSD}$$

Como la fracción no es potencia de 16, se ajusta al número dígitos que más convenga.

$$924.45_{10} \cong 39C.733_H$$

$$924.45_{10} \cong 39C.733_{16}$$

1.5.4 Conversión de octal a binario.

Para realizar esta conversión se cambia cada dígito del número octal por su equivalente en binario con grupos de 3 bits ya que con ellos se logra las combinaciones posibles para los 8 dígitos del sistema octal, se debe respetarse la posición del punto binario.

Ejemplo convertir el 471.103_8 a binario.

Ya que cada dígito del sistema octal se representará por su equivalente en tres bits, tenga presente la siguiente tabla para la conversión.

Decimal	Binario	Octal
0	000	0
1	001	1
2	010	2
3	011	3
4	100	4
5	101	5
6	110	6
7	111	7

Figura 1.20 Equivalencia entre los números binarios, decimal y octal.

Se cambia cada dígito del número decimal por su equivalente en binario con grupo de 3 bits.

4	7	1	.	1	0	3_8
↓	↓	↓	↓	↓	↓	↓
100	111	001	.	001	000	011

El resultado de la conversión es el siguiente:

$$471.103_8 = 100111001.001000\ 011_2$$

No se debe olvidar la base del número binario y se debe respetar la posición del punto binario.

1.5.4 Conversión de hexadecimal a binario.

La conversión de hexadecimal a binario se hace cambiando cada símbolo del número hexadecimal por un grupo de 4 bits ya que con estos bits se logra obtener las 16 combinaciones o valores posibles que se requiere para representar todos los símbolos del sistema numérico hexadecimal, recuerde que a las letras del sistema hexadecimal le corresponde un valor numérico. La posición del punto hexadecimal se debe respetar.

En la siguiente figura se muestra el valor tanto decimal como binario que le corresponde a cada uno de los símbolos del sistema numérico hexadecimal.

DECIMAL	BINARIO	HEXADECIMAL
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Figura 1.21 Equivalencia entre los sistemas numéricos decimal, binario y hexadecimal.

Ejemplo convertir el número hexadecimal A7B2.AD3_H a binario.

Se cambia cada símbolo del número hexadecimal por el grupo de 4 bits que se muestra en la tabla, cada grupo equivale al valor decimal de los símbolos hexadecimales,

A	7	B	2	.	A	D	3
↓	↓	↓	↓	↓	↓	↓	↓
1010	0111	1011	0010	.	1010	1101	0011

$$A7B2.AD3_H = 1010011110110010.11010011_2$$

1.5.6 Conversión de binario a octal.

Para hacer la conversión de binario a octal, se hacen grupos de 3 bits del punto binario a la izquierda, si faltan bits para el último grupo se completa con ceros a la izquierda, para la fracción se hacen grupos de 3 bits de izquierda a derecha y si faltan bits para el último grupo se completa con ceros a la derecha.

Ejemplo convertir el siguiente número binario a octal.

1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 . 1 0 0 1 1 0 1 1₂

Se inicia haciendo grupos de 3 bits en la parte entera.

1 0 1 1 1 0 1 1 1 0 1 1 1 . 1 0 0 1 1 0 1 1

A la parte entera le falta un bit se le puede agregar un cero a la izquierda para no afectar su valor y completar el grupo de 3 bits.

0 1 0 1 1 1 0 1 1 1 0 1 1 1 . 1 0 0 1 1 0 1 1

Ahora se hacen los grupos de 3 bits de la parte de la fracción.

0 1 0 1 1 1 0 1 1 1 0 1 1 1 . 1 0 0 1 1 0 1 1

Le falta un bit al último grupo de bits, se agrega un 0 a la derecha.

0 1 0 1 1 1 0 1 1 1 0 1 1 1 . 1 0 0 1 1 0 1 1 0

Por último, se pone el valor decimal que representa cada grupo de 3 bits, se sabe que el máximo valor será el 7, no olvidar poner la base del número octal.

0 1 0 1 1 1 0 1 1 1 0 1 1 1 . 1 0 0 1 1 0 1 1 0
2 7 3 5 6 7 . 4 6 6₈

El resultado de la conversión es:

1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 . 1 0 0 1 1 0 1 1₂ = 273567.466₈

1.5.7 Conversión de binario a hexadecimal.

La conversión de binario a hexadecimal es igual que la de binario a octal sólo que se hacen grupo de 4 bitas para lograr las 16 combinaciones o valores de los símbolos del sistema numérico hexadecimal.

Ejemplo convertir el siguiente número binario a hexadecimal.

1 0 1 1 1 0 1 0 1 0 1 1 1 1 0 1 1 0 1 1 1 . 1 0 0 1 1 0 1 0 1 1₂

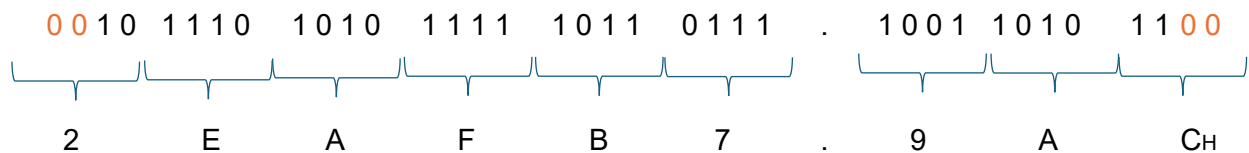
Lo primero es hacer los grupos de cuatro bits.

1 0 1 1 1 0 1 0 1 0 1 1 1 1 0 1 1 0 1 1 1 . 1 0 0 1 1 0 1 0 1 1

Le faltan dos bits para completar el grupo de 4 bits a la parte entera, se agregan ceros a la izquierdo, a la fracción le faltan dos bits para completar el grupo de 4 bits se le agrega dos ceros a la derecha.



Por último, poner el valor hexadecimal que le corresponde a cada grupo de 4 bits usando la tabla de la figura 1.21 y el punto hexadecimal respetando su posición.



$$101110101011110110111.1001101011_2 = 2EAFB7.9ACH$$

Convertir a los restantes sistemas numéricos los siguientes números.

- A) 12345.75₁₀
- B) 64324.625₈
- C) 10110011001111011.11011101₂
- D) 7ADB2C.DF34_H

La conversión será lenta o más rápida dependiendo la forma en la que se realice, lo conveniente es pasar primero al sistema que requiera el menor número de operaciones, por ejemplo, es conveniente en el inciso A, convertir primero a hexadecimal que requiere menor número de divisiones que si lo convirtiéramos a binario.

Sólo se resolverá el inciso A, los demás quedan de ejercicios.

- A) 12345.75₁₀

Primero se convierte a hexadecimal.

Parte entera:

$$12345 \div 16 = 771 \text{ residuo } 9$$

$$771 \div 16 = 48 \text{ residuo } 3$$

$$48 \div 16 = 3 \text{ residuo } 0$$

$$3 \div 16 = 0 \quad 3$$

Ahora la fracción

$$0.75 \times 16 \text{ producto } 12 \text{ esto es equivalente a la C.}$$

Entonces

$$12345.75_{10} = 3039.CH$$

Para convertir a binario se cambia cada símbolo del hexadecimal por su equivalente en 4 bits y le ponemos la base.

$$3 \quad 0 \quad 3 \quad 9 \quad . \quad CH$$

$$0011 \ 0000 \ 0011 \ 1001 \ . \ 1100_2$$

Para obtener el octal se agrupan los bits en grupo de 3.

$$0 \ 011 \ 000 \ 000 \ 111 \ 001 \ . \ 110 \ 0_2$$

$$0 \ 3 \ 0 \ 0 \ 7 \ 1 \ . \ 6 \ 0_8 = 30071.6_8$$

Por lo tanto:

$$12345.75_{10} = 3039.CH = 30071.6_8 = 001100000011001.110_2$$

1.6 Operaciones básicas con diferentes sistemas numéricos.

En los equipos electrónicos digitales como las computadoras, PLC'S, sistemas embebidos, calculadoras entre muchos más, se hacen operaciones aritméticas con los datos binarios para tomar decisiones en un programa de control y mostrar los resultados al usuario.

Por lo tanto, conocer las bases de cómo se realizan estas operaciones aritméticas con los diferentes sistemas numéricos es importante, sobre todo cuando se ejecutan los programas que harán el control en el sistema digital que se esté utilizando.

1.7.1 Suma binaria

La suma con el sistema numérico binario se hace de la misma forma que se hace en el sistema numérico decimal.

Las reglas para realizar la suma binaria son las siguientes:

$$a) \ 0 + 0 = 0$$

$$b) \ 0 + 1 = 1$$

$$c) \ 1 + 0 = 1$$

$$d) \ 1 + 1 = 10$$

En este caso la suma es 10₂ se pone el 0 y el uno es el acarreo para el siguiente bit más significativo.

e) $1 + 1 + 1 = 11$ En este caso la suma es 11_2 se pone el 1 y el uno es el acarreo para el siguiente bit más significativo.

Ejemplos de la suma binaria:

1)

$$\begin{array}{r} 110110 \\ + 011011 \\ \hline \end{array}$$

Al sumar el primer bit (LSB) la suma es 1_2 regla b.

$$\begin{array}{r} 110110 \\ + 011011 \\ \hline 1 \end{array}$$

$$\begin{array}{r} 1 \\ + 110110 \\ + 011011 \\ \hline 01 \end{array}$$

Al sumar el segundo bit la suma es 10_2 , aplicamos la regla d) por lo tanto, se pone 0 y se agrega el acarreo al siguiente bit.

Ahora se suma los dos bits de la suma más el acarreo, la suma es 10_2 se pone 0 y el acarreo se suma al siguiente bit.

$$\begin{array}{r} 11 \\ + 110110 \\ + 011011 \\ \hline 001 \end{array}$$

La suma de los tres bits es 10_2 , se pone 0 y se agrega el acarreo al siguiente bit.

$$\begin{array}{r} 111 \\ + 110110 \\ + 011011 \\ \hline 0001 \end{array}$$

Al sumar los 3 bits da 11_2 aplicando la regla e) se pone 1 y se agrega el acarreo al siguiente bit.

$$\begin{array}{r} 111 \\ 110110 \\ + 011011 \\ \hline 10001 \end{array}$$

Sumando los bits de la suma más el acarreo, da 112

$$\begin{array}{r} 1111 \\ + 110110 \\ 011011 \\ \hline 1001 \end{array}$$

La suma da 10_2 , como es el último bit se pone el 10_2 y con esto se concluye la suma.

$$\begin{array}{r} 1\,1\,1\,1 \\ + 1\,1\,0\,1\,1\,0 \\ 0\,1\,1\,0\,1\,1 \\ \hline 1\,0\,1\,0\,0\,0\,1 \end{array}$$

Se puede comprobar la suma con los valores equivalentes en decimal.

$$\begin{array}{r} + \quad 110110_2 \\ \quad 011011_2 \\ \hline 101000_2 \end{array} \quad \begin{array}{r} + \quad 54_{10} \\ \quad 27_{10} \\ \hline 81_{10} \end{array}$$

Ejemplo de números binario con fracción.

$$\begin{array}{r} 1\ 1\ 0\ 1\ 1\ 0\ 0\ 1.\ 1\ 0\ 1_2 \\ + \\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1.\ 1\ 1\ 1_2 \\ \hline 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1.\ 1\ 0\ 0_2 \end{array}$$

Para efectuar la suma de los números binarios con fracción se deben aplicar las mismas reglas que en el caso anterior, se debe respetar la posición del punto binario.

$$\begin{array}{r}
 1011001.101_2 \\
 + 1001001.111_2 \\
 \hline
 10110101.100_2
 \end{array}$$

Se puede hacer la comprobación sumando en decimal el equivalente de los números binarios.

$$\begin{array}{r} 11011001.101_2 \\ + 10010001.111_2 \\ \hline 10110101.100_2 \end{array} \qquad \begin{array}{r} 217.625_{10} \\ + 145.875_{10} \\ \hline 363.5_{10} \end{array}$$

Números binarios con signo.

Los números con signo son muy importantes debido a que las computadoras o sistemas digitales que procesan la información los utiliza para hacer sus cálculos matemáticos.

Existen dos formas de representar los números binarios con signo, estos son:

En la primera forma el bit más significativo (MSB) es el signo, y el resto de los bits del número representan la magnitud del número. Si el bit de signo es un 0 la cantidad es positiva (+) y si es un 1 la cantidad es negativa (-).

Ejemplo

En el número con signo 0 1 0 1 1 0 1 1 0

El bit más significativo es un 0 por lo tanto, es positivo, la magnitud se obtiene al convertir a decimal el resto de los bits.

$$\begin{array}{ccccccccc} \textcolor{red}{0} & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ \underbrace{}_{+} & \underbrace{}_{182} & & & & & & & \end{array}$$

El número es el +182.

Si se cambia el bit más significativo por 1, entonces el número será negativo, la magnitud es la misma.

$$\begin{array}{cccccccc} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ \hline & & & & & & & & \\ - & & & & & & & & 182 \end{array}$$

El número es -182.

Para la segunda forma de representar números con signo el bit más significativo sigue siendo el bit de signo si es 0 el número es positivo si es 1 el número es negativo, en los números positivos la magnitud la representa el resto de los bits del número sólo se convierte a decimal y esa será la magnitud, si el bit más significativo es 1 el número es negativo y la magnitud se obtiene al sacar el complemento a 2 del resto de los bits, una vez obtenido el complemento a 2 lo convertimos a decimal y el valor que arroje es la magnitud.

Complemento a 1 de un número binario.

El complemento a 1 del número binario se obtiene al cambiar los ceros por unos y los unos por ceros.

Ejemplo

Para el número $1\ 0\ 1\ 1\ 0\ 1\ 1\ 1_2$ el complemento a uno se obtiene cambiando de estado todos los bits.

$$\begin{array}{cccccccc} 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1_2 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0_2 \end{array}$$

El número $0\ 1\ 0\ 0\ 1\ 0\ 0\ 0_2$ es el complemento a 1 del número $1\ 0\ 1\ 1\ 0\ 1\ 1\ 1_2$

Complemento a 2 de un número binario.

Para obtener el complemento a 2 de cualquier número binario, primero se obtiene el complemento a 1 y al resultado se le suma un 1, el valor obtenido es el complemento a dos.

Ejemplo

Si se tiene el número binario: $1\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 1_2$

El complemento a 1 será: $0\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 0_2$

Al complemento a 1 se le suma 1: $0\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 0_2$

$$\begin{array}{cccccccc} & & & & & & & 1 \\ + & & & & & & & \\ \hline \text{Este será el complemento a 2} & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1_2 \end{array}$$

Números con signo usando complemento a 2

Los números positivos en complemento a 2 se obtienen igual que en la representación anterior, el bit más significativo es el bit de signo, si es 0 el número es positivo, si es 1 el número es negativo y la magnitud está en complemento a 2, por lo tanto, para encontrar la magnitud se vuelve a complementar a 2.

Ejemplo

Escribir el número +17 binario con 7 bits incluido el bit de signo en complemento a 2.

Primero se pone el bit más significativo que es el signo en este caso 0, luego se escribe el número 17 en binario con el resto de los bits, para este caso 6 bits:

$$\begin{array}{ccccccc} 0 & 0 & 1 & 0 & 0 & 0 & 1_2 \\ \underbrace{\hspace{1.5cm}}_{+} & \underbrace{\hspace{1.5cm}}_{17} & & & & & \end{array}$$

Ejemplo

Escribir el número -13 en binario con 7 bits incluido el bit de signo en complemento a 2.

Se escribe el bit de signo en este caso 1, después se trabaja con la magnitud 13, se escribe en 6 bits y se le saca el complemento a 2 y esa será la magnitud para el número negativo.

0 0 1 1 0 ₁₂ Magnitud 13 en 6 bits.

1 1 0 0 1 ₀₂ Complemento a uno de la magnitud.

1 1 0 0 1 ₁₂ Complemento a 2 de la magnitud.

El número -13 ya con el signo es: **1** 1 1 0 0 1 ₁₂

En caso de ser necesarios más bits debido a que la mayoría de los circuitos que realizan estas operaciones son múltiplos de 4 bits, para el número positivo se le agregan ceros a la izquierda y si el número es negativo se le agregan unos.

Negación de un número con signo en complemento a 2.

La negación de un número positivo se obtiene sacándole el complemento a 2, lo mismo sucede al número negativo.

Ejemplo

El número -13 en complemento a 2 es: **1** 1 1 0 0 1 ₁₂ si se saca el complemento a 2 dará el número +13, esto es: **0** 0 0 1 1 0 ₁₂

Resta binaria en complemento a 2 con un circuito sumador.

Para efectuar una resta con circuito sumador, se complementa a 2 el sustraendo de la resta con igual número de bits del minuendo y el resultado se suma al minuendo, el resultado es la resta de los dos números. En caso de que se genere un acarreo se debe ignorar.

Ejemplo

Efectuar la resta en forma de suma:

$$\begin{array}{r} 11010012 \longrightarrow 105_{10} \\ - 1001102 \longrightarrow 38_{10} \\ \hline \end{array}$$

Lo primero que se debe hacer es completar el sustraendo con igual número de bits del minuendo agregándole ceros a la izquierda.

$$\begin{array}{r} 11010012 \\ - 01001102 \\ \hline \end{array}$$

Se pone el sustraendo en complemento a 2 y se hace la suma.

$$\begin{array}{r} 11010012 \\ + 10110102 \text{ Sustraendo en complemento a 2} \\ \hline 11000011 \end{array}$$

El acarreo no se toma en cuenta por lo tanto la resta será: 1000011_2 esto es 67_{10}

Sumas en complemento a 2

a)

$$\begin{array}{rcl} +7 & \longrightarrow & 001112 \text{ sumando} \\ +3 & \longrightarrow & 000112 \text{ sumando} \\ \hline & & 010102 \text{ suma} = +10 \end{array}$$

b)

$$\begin{array}{rcl}
 +7 & \longrightarrow & 0\ 0\ 1\ 1\ 1_2 \text{ sumando} \\
 -3 & \longrightarrow & 1\ 1\ 1\ 0\ 1_2 \text{ sumando} \\
 \hline
 & & \textcolor{red}{1}\ 0\ 0\ 1\ 0\ 0_2 \text{ suma} = +4 \text{ recuerda que el acarreo se elimina.}
 \end{array}$$

c)

$$\begin{array}{rcl}
 -7 & \longrightarrow & 1\ 1\ 0\ 0\ 1_2 \text{ sumando} \\
 -3 & \longrightarrow & 1\ 1\ 1\ 0\ 1_2 \text{ sumando} \\
 \hline
 & & \textcolor{red}{1}\ \textcolor{blue}{1}\ 0\ 1\ 1\ 0_2 \text{ suma} = -10 \text{ el acarreo se descarta el signo es negativo.}
 \end{array}$$

d)

$$\begin{array}{rcl}
 -7 & \longrightarrow & 1\ 1\ 0\ 0\ 1_2 \text{ sumando} \\
 +3 & \longrightarrow & 0\ 0\ 0\ 1\ 1_2 \text{ sumando} \\
 \hline
 & & \textcolor{blue}{1}\ 1\ 1\ 0\ 0_2 \text{ suman} = -4
 \end{array}$$

Multiplicación con números binarios

Para realizar una multiplicación con números binarios se siguen los mismos pasos que se hacen en la multiplicación decimal sólo que aquí se tiene dos símbolos (bits) únicamente el 0 y el 1 por lo que resulta más sencillo.

Ejemplo

$$\begin{array}{r}
 1\ 0\ 1\ 0_2 \\
 \times 0\ 1\ 1\ 1_2 \\
 \hline
 \end{array}$$

Para desarrollar el producto se toma el primer bit del multiplicador y se multiplica por todo el multiplicando, en seguida se toma el segundo bit del multiplicador y se multiplica por el multiplicando este producto se coloca en la segunda posición es decir con un corrimiento de un bit a la derecha y se continua así hasta terminar con todos los bits del multiplicador y en seguida se hace la suma de estos productos parciales y el resultado será la multiplicación de los dos números binarios.

$$1\ 0\ 1\ 0_2 \quad 10_{10}$$

$$\begin{array}{r}
 X \quad 1 \ 0 \ 0 \ 1_2 \quad 9_{10} \\
 \hline
 1 \ 0 \ 1 \ 0 \text{ producto del primer bit del multiplicador} \\
 0 \ 0 \ 0 \ 0 \text{ producto del segundo bit del multiplicador} \\
 0 \ 0 \ 0 \ 0 \text{ producto del tercer bit del multiplicador} \\
 1 \ 0 \ 1 \ 0 \text{ producto del cuarto bit del multiplicador} \\
 \hline
 1 \ 0 \ 1 \ 1 \ 0 \ 1 \ 0_2 = 90_{10} \text{ producto total, suma de los productos parciales.}
 \end{array}$$

Ejemplo

$$\begin{array}{r}
 1 \ 1 \ 1 \ 0 \ 1_2 \quad 29_{10} \\
 X \quad 1 \ 1 \ 1 \ 1_2 \quad 15_{10} \\
 \hline
 \end{array}$$

La mayoría de los sistemas digitales efectúan las multiplicaciones binarias con dos números a la vez por lo que se puede multiplicar los dos primeros bits del multiplicador por el multiplicando y se hace la primera suma parcial de estos, en seguida se multiplica el tercer bit del multiplicador por el multiplicando y se suma a la primera suma parcial y da la segunda suma parcial y esto se repite hasta terminar con todos los bits del multiplicados, siempre respetando la posición que le corresponde a cada bit del multiplicador en la suma parcial, el resultado de la última suma es el producto de los dos números binarios.

Ejemplo

1 1 1 0 1 ₂	29 ₁₀
X 1 1 1 1 ₂	15 ₁₀
$ \begin{array}{r} \hline 1 \ 1 \ 1 \ 0 \ 1 \\ \hline 1 \ 1 \ 1 \ 0 \ 1 \\ \hline 1 \ 0 \ 1 \ 0 \ 1 \ 1 \ 1 \\ 1 \ 1 \ 1 \ 0 \ 1 \\ \hline 1 \ 1 \ 0 \ 0 \ 1 \ 0 \ 1 \ 1 \\ 1 \ 1 \ 1 \ 0 \ 1 \\ \hline 1 \ 1 \ 0 \ 1 \ 1 \ 0 \ 0 \ 1 \ 1_2 \end{array} $	producto del primer bit producto del segundo bit suma parcial de los 2 primeros bits producto del tercer bit suma parcial del producto de los tres primeros bits producto del cuarto bit suma parcial el cuarto bit = al producto total = 435 ₁₀

División binaria

El procedimiento de la división binaria sigue las mismas reglas que la división en decimal sólo que, en binario tenemos nada más dos símbolos el 0 y el 1 (bits).

Para hacer la división se toman igual números de bits del dividendo que los que tiene el divisor si son mayor o igual a la magnitud que representa el divisor, se pone 1 en el cociente en seguida se multiplica este por el divisor el resultado se le resta a los bits que se está dividiendo en ese momento del dividendo, una vez que se tiene la resta se baja el siguiente bit del dividendo si la magnitud es menor que el divisor, se pone 0 en el cociente y se baja el siguiente bit hasta que el dividendo sea mayor o igual el divisor, esto se repite hasta que en el dividendo ya no haya bits para dividir, los bits que no alcanzan el valor del divisor será el residuo.

Ejemplo

101 $\overline{) 1\ 1\ 0\ 1\ 1\ 0}$ esto es en decimal $54_{10} \div 5_{10}$

1

101 $\overline{) 1\ 1\ 0\ 1\ 1\ 0}$ tomamos los tres primeros bits del dividendo, es mayor que el divisor.

Los tres bits que tomamos del dividendo representan el número 6 decimal y el divisor el 5 por lo tanto le toca a 1.

1

101 $\overline{) 1\ 1\ 0\ 1\ 1\ 0}$
 $\underline{1\ 0\ 1}$
 0 0 1

Al efectuar el producto del cociente (1) por el divisor y restándolo al dividendo nos quedó 001

Al bajar el siguiente bit del dividendo nos queda 0011 no alcanza el valor del divisor, ponemos 0 en el cociente y bajamos el siguiente bit del dividendo.

1 0

101 $\overline{) 1\ 1\ 0\ 1\ 1\ 0}$
 $\underline{1\ 0\ 1}$
 0 0 1 1

1 0

$$\begin{array}{r}
 101 \overline{) 110110} \\
 \underline{101} \\
 00111
 \end{array}$$

El dividendo que quedó al bajar el siguiente bit es 00111 este es mayor que el divisor, se pone 1 en el cociente y se repite la multiplicación y resta para tener los nuevos bits del divisor.

$$\begin{array}{r}
 101 \overline{) 110110} \\
 \underline{101} \\
 00111 \\
 \underline{101} \\
 010
 \end{array}$$

Se baja el último bit del dividendo y lo que queda es 0100 esto es menor que el divisor se pone 0 en el cociente, por lo tanto, se concluyó con la división estos bits son el residuo.

$$\begin{array}{r}
 101 \overline{) 110110} \\
 \underline{101} \\
 00111 \\
 \underline{101} \\
 0100 = 4_{10}
 \end{array}$$

Si se multiplica el cociente por divisor y le sumamos el residuo debe dar el dividendo, lo que indicaría que la división fue correcta, $(5 \cdot 10) + 4 = 54_{10}$

Para la división binaria con punto decimal se trabaja de la misma forma que en la división en decimal, sólo se sube el punto binario al cociente respetando su posición.

Ejemplo

$$\begin{array}{r}
 111.111 \\
 111 \overline{) 110111.101} \\
 \underline{111} \\
 1101 \\
 \underline{111} \\
 1101 \\
 \underline{111} \\
 1101 \\
 \underline{111} \\
 1100 \\
 \underline{111} \\
 1011 \\
 \underline{111} \\
 100
 \end{array}$$

Suma BCD

Los sistemas digitales hacen cálculos con el código BCD, este código se utiliza para representar los dígitos del sistema decimal, por lo tanto, sólo se utilizan 10 combinaciones de las 16 posibles que se pueden hacer con 4 bits, estas son del 0 al 9, las restantes combinaciones las del 10 al 15 no son válidas.

Ejemplo

Sumar en BCD.

3 decimal	0 0 1 1 BCD
+ 5 decimal	+ 0 1 0 1 BCD
<u>8</u>	<u>1 0 0 0</u>

El resultado es correcto ya que los 4 bits si representan un código BCD.

Ejemplo sumar en BCD.

36 decimal	0 0 1 1	0 1 1 0	BCD
+ 42 decimal	+ 0 1 0 0	0 0 1 0	BCD
78	0 1 1 1	1 0 0 0	BCD

El resultado es correcto los dos grupos de 4 bits que arroja la suma son BCD.

Ejemplo

Sumar en BCD.

74 decimal	0 1 1 1	0 1 0 0	BCD
+ 17	0 0 0 1	0 1 1 1	BCD
91	1 0 0 0	1 0 1 1	→ el primer grupo de 4 bits no es BCD

Cuando la suma de dos dígitos es mayor que 9, se le deben sumar 6 para regresarlo a BCD, esto provocará un acarreo que se le suma al siguiente dígito.

74 decimal	0 1 1 1	0 1 0 0	BCD
+ 17	0 0 0 1	0 1 1 1	BCD
91	1 0 0 0	1 0 1 1	
	1	0 1 1 0	
	1 0 0 1	0 0 0 1	BCD

Ejemplo

Sumar en BCD.

278 decimal	0 0 1 0	0 1 1 1	1 0 0 0	BCD
+ 119 decimal	+ 0 0 0 1	0 0 0 1	1 0 0 1	BCD
397	0 0 1 1	1 0 0 0	0 0 0 1	
		1	0 1 1 0	
	0 0 1 1	1 0 0 1	0 1 1 1	BCD

En la suma del primer dígito rebasó el 15 binario lo que provoca un acarreo al siguiente dígito, por lo tanto, a este dígito se le suman 6 no importa que arroje un valor válido en BCD. Esto será lo mismo para cualquier dígito.

Resumiendo, cuando en la suma de dos números en BCD sea mayor 9 se le deben sumar 6, si la suma provoca un acarreo también se le suman 6 y el acarreo se suma al siguiente dígito.

Suma hexadecimal

La suma en hexadecimal también sigue las reglas de la suma decimal, sólo se debe tener en cuenta que del número hexadecimal 9 sigue la A y así hasta la F, después de la F sigue el 10h, es decir se pone 0 en la suma y hay un acarreo para el siguiente dígito.

Es importante recordar cómo se hace el conteo en hexadecimal para no cometer errores en la suma, por ejemplo, si se suma 5 + 9 esto da 14 decimal, pero como se está trabajando en hexadecimal se cambia el 14 por la letra que le corresponda, en este caso es la letra E.

Ejemplo

Sumar los siguientes números.

$$\begin{array}{r} 79h \\ + \quad 24h \\ \hline 9Dh \end{array} \quad \text{La suma de } 9 + 4 \text{ es trece en decimal en hexadecimal es Dh}$$

Ejemplo

Sumar los siguientes números.

$$\begin{array}{r} \text{1 acarreo} \\ A67Bh \\ + 28B1h \\ \hline CF2Ch \end{array}$$

Se puede cambiar el dígito hexadecimal por grupo de 4 bits y realizar la suma binaria y al final volverlo a cambiar a los símbolos del sistema numérico hexadecimal.

1 0 1 0 0 1 1 0 0 1 1 1 1 0 1 1

$$\begin{array}{cccc}
 + & 0010 & 1000 & 1011 & 0000 \\
 \hline
 & 1100 & 1111 & 0010 & 1100 \\
 & C & F & 2 & Ch
 \end{array}$$

Resta hexadecimal

La resta se hace igual que en decimal, sólo hay que tener en cuenta que en decimal el máximo número es el 9 y en hexadecimal la F.

Ejemplo

Restar los siguientes números.

$$\begin{array}{r}
 A34Fh \\
 - 7B31h \\
 \hline
 281Eh
 \end{array}$$

También se puede hacer en binario en grupo de 4 bits y al final regresar a hexadecimal.

$$\begin{array}{cccc}
 1010 & 0011 & 0100 & 1111 \\
 - 0111 & 1011 & 0011 & 0001 \\
 \hline
 0011 & 1000 & 0001 & 1110 = 381Eh
 \end{array}$$

También se puede hacer la resta en forma de suma sacando el complemento a 2 al sustraendo y sumando al minuendo, en caso de que haya un acarreo se descarta.

Sacando el complemento a dos al sustraendo queda de la siguiente manera.

7B31h es igual en complemento a 2 de 84CFh

Sumamos este valor al minuendo.

$$\begin{array}{r}
 A34Fh \\
 + 84CFh \\
 \hline
 1281Eh
 \end{array}$$

se elimina el acarreo, el resultado de la resta es 281Eh.

Unidad Lógica Aritmética.

La Unidad Lógica Aritmética (ALU, por sus siglas en inglés) es un circuito digital fundamental dentro de un procesador o microcontrolador. Su función principal es ejecutar operaciones aritméticas (suma, resta, multiplicación, división) y lógicas (AND, OR, XOR, NOT) sobre datos binarios.

Características de una ALU en circuito integrado

1. Componentes principales:

- Sumadores y restadores: Circuitos que permiten realizar operaciones aritméticas básicas.
- Puertas lógicas: Implementan operaciones lógicas fundamentales.
- Multiplexores: Seleccionan la operación que debe ejecutarse.
- Registros temporales: Almacenan datos intermedios durante la ejecución de cálculos.
- Flags o indicadores: Señalan condiciones especiales, como "overflow" o "carry".

2. Operaciones soportadas:

- Aritméticas: Suma, resta, incremento, decremento, desplazamientos.
- Lógicas: AND, OR, XOR, NOT, comparación de valores.
- Desplazamientos y rotaciones: Operaciones de corrimiento de bits.

3. Arquitectura en un circuito integrado:

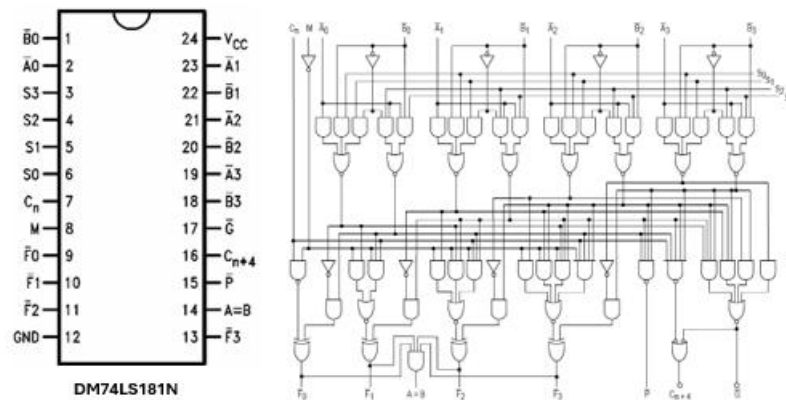
- Diseñada con transistores CMOS o TTL en chips específicos.
- Puede formar parte de microprocesadores como los Intel x86, ARM o controladores embebidos.
- En procesadores avanzados, una ALU puede ser parte de una Unidad de Punto Flotante (FPU) para cálculos más complejos.

4. Ejemplo de ALU en circuito integrado:

- **74LS181**: Un circuito integrado de la serie **TTL** que implementa una ALU de 4 bits capaz de realizar múltiples operaciones lógicas y aritméticas.

Importancia en la computación.

La ALU es un componente esencial en cualquier CPU o microcontrolador, ya que realiza las operaciones matemáticas y lógicas necesarias para la ejecución de instrucciones de software.



Selection	Active-low data			Active-high data		
	Logic M = H	Arithmetic M = L		Logic M = H	Arithmetic M = L	
S3 S2 S1 S0		Cn = L (no carry)	Cn = H (carry)		Cn = H (no carry)	Cn = L (carry)
L L L L	$\bar{A}$	A minus 1	A	$\bar{A}$	A	A plus 1
L L L H	$\bar{A}\bar{B}$	AB minus 1	AB	$A + B$	$A + B$	$(A + B)$ plus 1
L L H L	$\bar{A} + B$	$\bar{A}\bar{B}$ minus 1	$\bar{A}\bar{B}$	$\bar{A}\bar{B}$	$A + \bar{B}$	$(A + \bar{B})$ plus 1
L L H H	Logical 1	-1 (two's complement)	0 (zero)	Logical 0	-1 (two's complement)	0 (zero)
L H L L	$\bar{A} + \bar{B}$	A plus $(A + \bar{B})$ plus 1	$\bar{A}$ plus $(A + \bar{B})$ plus 1	$\bar{A}\bar{B}$	A plus $\bar{A}\bar{B}$	A plus $(\bar{A}\bar{B})$ plus 1
L H L H	$\bar{B}$	$\bar{A}\bar{B}$ plus $(A + \bar{B})$ plus 1	$\bar{A}\bar{B}$ plus $(A + \bar{B})$ plus 1	$\bar{B}$	$(A + B)$ plus $\bar{A}\bar{B}$	$(A + B)$ plus $\bar{A}\bar{B}$ plus 1
L H H L	$\bar{A} \oplus B$	A minus B minus 1	A minus B	$A \oplus B$	A minus B minus 1	A minus B
L H H H	$\bar{A} + \bar{B}$	$A + \bar{B}$	$A + \bar{B}$ plus 1	$\bar{A}\bar{B}$	$\bar{A}\bar{B}$ minus 1	$\bar{A}\bar{B}$
H L L L	$\bar{A}\bar{B}$	A plus $(A + \bar{B})$ plus 1	A plus $(A + \bar{B})$ plus 1	$\bar{A} + B$	A plus $\bar{A}\bar{B}$	A plus $\bar{A}\bar{B}$ plus 1
H L L H	$\bar{A} \oplus \bar{B}$	A plus B	A plus B plus 1	$\bar{A} \oplus \bar{B}$	A plus B	A plus B plus 1
H L H L	B	$\bar{A}\bar{B}$ plus $(A + \bar{B})$ plus 1	$\bar{A}\bar{B}$ plus $(A + \bar{B})$ plus 1	B	$(A + \bar{B})$ plus $\bar{A}\bar{B}$	$(A + \bar{B})$ plus $\bar{A}\bar{B}$ plus 1
H L H H	$\bar{A} + B$	$A + B$	$(A + B)$ plus 1	$\bar{A}\bar{B}$	$\bar{A}\bar{B}$ minus 1	$\bar{A}\bar{B}$
H H L L	Logical 0	$\bar{A}$ plus $\bar{A}$	$\bar{A}$ plus $\bar{A}$ plus 1	Logical 1	A plus A	A plus A plus 1
H H L H	$\bar{A}\bar{B}$	$\bar{A}\bar{B}$ plus A	$\bar{A}\bar{B}$ plus A plus 1	$A + \bar{B}$	$(A + \bar{B})$ plus A	$(A + \bar{B})$ plus A plus 1
H H H L	$\bar{A}\bar{B}$	$\bar{A}\bar{B}$ plus A	$\bar{A}\bar{B}$ plus A plus 1	$A + B$	$(A + B)$ plus A	$(A + B)$ plus A plus 1
H H H H	A	A	A plus 1	A	A minus 1	A

Figura 1.22 unidad Lógica aritmética 74LS181. [11]

1.7 Códigos binarios y alfanuméricos.

Códigos binarios.

Es un sistema de codificación que se utiliza en los sistemas computacionales y los sistemas digitales que tiene como objetivo la conversión de un sistema de datos de origen a otro sistema de datos de destino para el manejo de programas, datos, instrucciones, control etc. Y sólo maneja dos símbolos el 0 y el 1, algunos de estos códigos binarios son: BCD, Gray y el exceso 3 entre otros.

1.7.1 Código Gray

El Código Gray es un código no ponderado cuya característica principal es que, entre dos números consecutivos, sólo cambia de estado un bit por lo que también es llamado código de cambio único. Es utilizado para detectar posiciones angulares debido a su característica y por ello el error que se puede cometer en la posición sólo depende de la cantidad de bits con la que se lee la posición, a mayor número de bits menor será el error. También es utilizado para enumerar los renglones y columnas de los mapas de Karnaugh que son una herramienta para simplificar las expresiones lógicas en el diseño de un circuito combinacional o secuencial.

Dígito Decimal	Código Gray	Dígito Decimal	Código Gray
0	0 0 0 0	8	1 1 0 0
1	0 0 0 1	9	1 1 0 1
2	0 0 1 1	10	1 1 1 1
3	0 0 1 0	11	1 1 1 0
4	0 1 1 0	12	1 0 1 0
5	0 1 1 1	13	1 0 1 1
6	0 1 0 1	14	1 0 0 1
7	0 1 0 0	15	1 0 0 0

Figura 1.24 Código Gray para los números del 0 al 15.

Observe que entre dos cuentas sucesivas sólo cambia de estado un solo bit, entre la cuenta 11 y 12 tenemos: 1 1 1 0 → 1 0 1 0, entre la cuenta 3 y 4 tenemos:

0 0 1 0 → 0 1 1 0

Conversión de binario a Gray

Para escribir un número en código gray de n bits, primero escribimos el número en binario, en seguida se copia el bit más significativo del binario como el bit más significativo del código gray, en seguida se hace una operación XOR lógica de este bit con el siguiente del binario el resultado se pone como siguiente bit del código gray, se toma el segundo bit y se hace la XOR con el tercer bit del binario el resultado es el siguiente bit del código Gray y se repite el procedimiento hasta terminar con todos los bits.

Ejemplo

Escribir en código gray el número 54_{10}

Escribimos al 54_{10} en binario

Se copia el bit más significativo como primer bit del Gray, en seguida se hace en forma horizontal la operación lógica XOR de este bit con el que sigue del binario para generar el siguiente bit del Gray y se repite hasta terminar con todos los bits, note que cuando los dos bits de la XOR son iguales generan un 0 y cuando son diferente generan un 1.

$$\begin{array}{ccccccccc} 1 & \oplus & 1 & \oplus & 0 & \oplus & 1 & \oplus & 1 & \oplus & 0_2 & \text{binario } 54 \\ \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & \\ 1 & & 0 & & 1 & & 1 & & 0 & & 1 & \text{Gray } 45 \end{array}$$

Conversión de Gray a binario.

Si se tiene un código Gray y se desea saber qué número representa, se copia el bit más significativo de este código como bit más significativo del binario, en seguida se hace una operación lógica XOR de este bit con el siguiente del Gray para generar el nuevo bit del binario, esto se repite hasta terminar con todos los bits.

Ejemplo

Diga que número Gray es el siguiente:

$$1 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0 \quad \text{Gray}$$

$$\begin{array}{ccccccc} 1 & 1 & 0 & 1 & 1 & 0 & \text{Gray} \\ \downarrow \oplus & \downarrow \oplus & \downarrow \oplus & \downarrow \oplus & \downarrow \oplus & & \\ 1 & 0 & 0 & 1 & 0 & 0 & \text{binario } 36_2 \end{array}$$

Por lo tanto, el código Gray $1 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0$ es el 36

Podemos comprobarlo cambiando nuevamente este código binario a Gray como en el ejemplo anterior.

$$\begin{array}{ccccccccc} 1 & \oplus & 0 & \oplus & 0 & \oplus & 1 & \oplus & 0 & \oplus & 0 & \text{binario } 36 \\ \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & \\ 1 & & 1 & & 0 & & 1 & & 1 & & 0 & \text{Gray } 36 \end{array}$$

Uso del código gray y sensores ópticos para detectar la posición angular.

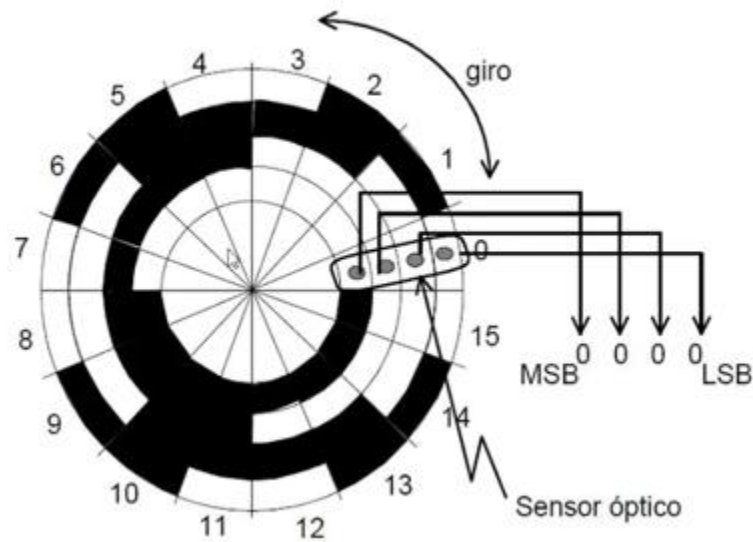


Figura 1.25 Detección de posición angular con el código Gray.

La precisión de la posición depende del número de bits (sensores) que se usen, si sólo son 4 bits se pueden hacer 16 combinaciones diferentes, por lo tanto, se divide 360 grados entre 16 y nos da la precisión, en este caso es de 22.5 grados. Si se utilizan 8 bits la precisión sería de 1.4 grados.

Uso del código Gray en mapas de Karnaugh.

		abc							
def	abc	000	001	011	010	110	111	101	100
	def	000	001	011	010	110	111	101	100
000	000	0	8	24	16	48	56	40	32
001	001	1	9	25	17	49	57	41	33
011	011	3	11	27	19	51	59	43	35
010	010	2	10	26	18	50	58	42	34
110	110	6	14	30	22	54	62	46	38
111	111	7	15	31	23	55	63	47	39
101	101	5	13	29	21	53	61	45	37
100	100	4	12	28	20	52	60	44	36

Figura 1.26 Mapa de Karnaugh de 6 variables.

Los renglones y las columnas están enumeradas con el código Gray. Debido a que entre dos cuentas consecutivas de este código sólo cambia de estado un bit, nos ayuda a simplificar la función lógica.

Códigos alfanuméricos

Este tipo de codificación utiliza una combinación de números (1,2..9), letras (a, A, b, B ...z, Z) y símbolos especiales (#, %, &....@) y es muy utilizada en sistemas de información y de control.

Ejemplo de estos es el código ASCII (Código Estándar Americano Para el Intercambio de Información) y el UNICODE (Estándar Universal de Codificación de Caracteres).

1.7.2 Código BCD (decimal codificado en binario)

Este código utiliza un grupo de 4 bits para representar cada uno de los dígitos del sistema numérico decimal. Es importante entender que el conjunto de bits que representa un código BCD no es equivalente al número binario que representa este conjunto.

Digito Decimal	BCD8421
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Figura 1.23 Dígitos decimales y su código en BCD.

Ejemplo

Escriba en código BCD el número 78610

cada dígito del número decimal se cambia por su grupo de 4 bits que lo representa.

7 = 0111, 8 = 1000 y el 6 = 0110

78610 = 0 1 1 1 1 0 0 0 0 1 1 0 en BCD

El puro conjunto de bits del código BCD es 011110000110 su equivalente binario sería 191610 que es diferente al código que representa.

Ejemplo

Diga si la siguiente cadena de bits es un código BCD.

0 1 1 0 1 0 0 1 0 0 1 1 0 1 1 1 0 1 1 0 0 0 0 1

Lo primero que debe cumplir es que sean grupos de cuatro bits exactamente.

0 1 1 0 1 0 0 1 0 0 1 1 0 1 1 1 0 1 1 0 0 0 0 1

La primera regla si la cumple, no importa si se agrupan los bits de izquierda a derecha o al revés.

La segunda regla es que el grupo de cuatro bits representen el número binario entre 0 y 9 ya que estos son los códigos del BCD, con un solo grupo que no cumpla la condición se descarta la cadena como un código BCD.

0 1 1 0 1 0 0 1 0 0 1 1 0 1 1 1 0 1 1 0 0 0 0 1

La cadena de bits si cumple con las dos reglas, por lo tanto, es el código 693761 BCD

Ejemplo

Diga si la siguiente cadena de bits es un código BCD.

1 0 0 0 1 1 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0 1 0 1

Si cumple con la primera regla.

1 0 0 0 1 1 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0 1 0 1

No cumple con la segunda regla, el grupo de bits 1 1 0 0 representa el número 12 binario, este no forma parte del código BCD, por lo tanto, no es una cadena de código BCD.

Ejemplo

Diga si la siguiente cadena de bits es un código BCD.

1 0 0 0 1 0 0 0 0 1 1 1 0 1 1 0 0 1 0

Se hacen los grupos de 4 bits.

1 0 0 0 1 0 0 0 0 1 1 1 0 1 1 0 0 1 0

No se cumple los grupos de 4 bits, no importa si se agrupan de derecha a izquierda o, al contrario, por lo tanto, no es código BCD.

1.7.3 Código ASCII

Código ASCII estándar, los sistemas que intercambian información usan 8 bits, el código sólo requiere 7 de los 8 bits para poder representar hasta 128 simbol entre letras, números y caracteres de control, el octavo bit se usa como bit de control de paridad de unos o ceros que se envían o reciben de un equipo a otro.

La siguiente figura muestra el código ASCII con su número que le corresponde en decimal y hexadecimal, por ejemplo, el ASCII "a" le corresponde el valor decimal 97₁₀ o el hexadecimal 61h.

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	\$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-~	63	3F	?	95	5F	_	127	7F	DEL

Figura 1.27 Código ASCII estándar. [12]

Detección de errores con la paridad con paridad en ASCII

Cuando se desea transmitir información en código ASCII estándar se puede utilizar el método de paridad con el octavo bit para detectar los errores que se producen al enviar o recibir información, se puede usar la paridad de unos par o impar, tanto el transmisor como el receptor deben checar que el dato enviado o recibido coincide con la paridad que se está manejando, si no se cumpla la paridad se cometió un error.

Para trabajar la paridad debe ponerse en estado de 0 o 1 el octavo bit para que cumpla la paridad de unos con la que se desea trabajar.

Ejemplo de quiere transmitir la palabra “BUENO” con paridad par.

Primero se ve el valor hexadecimal que le corresponde a cada símbolo ASCII “BUENO”:

B = 0 1 0 0 0 0 1 0 = 42h

U = 0 1 0 1 0 1 0 1 = 55h

E = 0 1 0 0 0 1 0 1 = 45h

N = 0 1 0 0 1 1 1 0 = 4Eh

O = 0 1 0 0 1 1 1 1 = 4Fh

Ahora se revisa la cantidad de unos que tiene cada ASCII si es par se queda igual en caso contrario se pone en 1 el octavo bit para tener número de unos pares’

B = 0 1 0 0 0 0 1 0 = 42h tiene número de unos pares, se queda igual

U = 0 1 0 1 0 1 0 1 = 45h tiene número de unos pares, se queda igual

E = 1 1 0 0 0 1 0 1 = C5h se cambió el octavo bit para tener unos pares

N = 0 1 0 0 1 1 1 0 = 4Eh tiene número de unos pares, se queda igual

O = 1 1 0 0 1 1 1 1 = 4Fh se cambió el octavo bit para tener unos pares

Entonces los datos que se debe transmitir en paridad par para la palabra “BUENO” deben ser: 42h, 45h, C5h, 4Eh, 4Fh.

En caso de que se requiera mandar la palabra “BUENO” con paridad impar queda de la siguiente manera:

B = 1 1 0 0 0 0 1 0 = C2h se cambió el octavo bit para tener unos impar

U = 1 1 0 1 0 1 0 1 = D5h se cambió el octavo bit para tener unos pares

E = 0 1 0 0 0 1 0 1 = 45h tiene número de unos impar, se queda igual

N = 1 1 0 0 1 1 1 0 = DEh se cambió el octavo bit para tener unos impar

O = 0 1 0 0 1 1 1 1 = 4Fh tiene número de unos impar, se queda igual

Código ASCII extendido, este código utiliza los 8 bit por lo que se pueden representar hasta 256 símbolos diferentes, en el caso de que se requiera manejar la paridad para la detección de errores, el sistema agrega un noveno bit con este fin.

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	Ł	224	E0	α
129	81	ü	161	A1	í	193	C1	ł	225	E1	β
130	82	é	162	A2	ó	194	C2	Ł	226	E2	Γ
131	83	â	163	A3	ú	195	C3	ł	227	E3	π
132	84	à	164	A4	ñ	196	C4	Ł	228	E4	Σ
133	85	ã	165	A5	Ñ	197	C5	ł	229	E5	σ
134	86	ä	166	A6	ª	198	C6	Ł	230	E6	μ
135	87	ç	167	A7	º	199	C7	ł	231	E7	ι
136	88	ê	168	A8	Ł	200	C8	Ł	232	E8	Φ
137	89	ë	169	A9	ł	201	C9	Ł	233	E9	Θ
138	8A	è	170	AA	ł	202	CA	Ł	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	Ł	235	EB	δ
140	8C	î	172	AC	¼	204	CC	Ł	236	EC	∞
141	8D	ì	173	AD		205	CD	Ł	237	ED	φ
142	8E	Ā	174	AE	<	206	CE	Ł	238	EE	ε
143	8F	Ā	175	AF	>	207	CF	Ł	239	EF	∩
144	90	Ē	176	B0	⋮	208	D0	Ł	240	FO	≡
145	91	æ	177	B1	⋮	209	D1	Ł	241	F1	±
146	92	Æ	178	B2	⋮	210	D2	Ł	242	F2	±
147	93	ô	179	B3		211	D3	Ł	243	F3	≤
148	94	ö	180	B4		212	D4	Ł	244	F4	
149	95	ò	181	B5		213	D5	Ł	245	F5	
150	96	û	182	B6		214	D6	Ł	246	F6	→
151	97	ù	183	B7		215	D7	Ł	247	F7	≈
152	98	ÿ	184	B8		216	D8	Ł	248	F8	≈
153	99	Ō	185	B9		217	D9	Ł	249	F9	·
154	9A	Ū	186	BA		218	DA	Ł	250	FA	·
155	9B	ϕ	187	BB		219	DB	Ł	251	FB	√
156	9C	£	188	BC		220	DC	Ł	252	FC	n
157	9D	¥	189	BD		221	DD	Ł	253	FD	²
158	9E	₤	190	BE		222	DE	Ł	254	FE	■
159	9F	f	191	BF		223	DF	Ł	255	FF	■

Figura 1.27 Código ASCII extendido. [12]

1.7.4 Unicode

UNICODE constituye un estándar para la codificación de caracteres, diseñado para facilitar la representación, manipulación y almacenamiento de texto en diversos sistemas y plataformas. La finalidad de este código es consolidar los diversos sistemas de codificación existentes, asignando un número único a cada carácter, sin importar el idioma o el software empleado.

Este estándar abarca una amplia variedad de escrituras del mundo, símbolos, emojis y caracteres especiales, permitiendo la compatibilidad entre distintos dispositivos y aplicaciones. Unicode se actualiza constantemente para incluir nuevos caracteres y mejorar su compatibilidad con diferentes lenguajes y sistemas de escritura.

	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
B	U+0080 PAD	U+0081 HOP	U+0082 BPH	U+0083 NBH	U+0084 IND	U+0085 NEL	U+0086 SSA	U+0087 ESA	U+0088 HTS	U+0089 HTJ	U+008A LTS	U+008B PLD	U+008C PLU	U+008D RI	U+008E SS2	U+008F SS3
9	U+0090 DCS	U+0091 PU1	U+0092 PU2	U+0093 STS	U+0094 CCH	U+0095 MW	U+0096 SPA	U+0097 EPA	U+0098 SOS	U+0099 SGCI	U+009A SCI	U+009B CSI	U+009C ST	U+009D OSC	U+009E PM	U+009F APC
A	U+00A0 NBSP	U+00A1 '	U+00A2 '	U+00A3 £	U+00A4 €	U+00A5 ƒ	U+00A6 §	U+00A7 ¨	U+00A8 ©	U+00A9 «	U+00AA »	U+00AB ™	U+00AC ®	U+00AD —	U+00AE —	U+00AF —
B	U+00B0 °	U+00B1 ±	U+00B2 ²	U+00B3 ³	U+00B4 ´	U+00B5 µ	U+00B6 ¶	U+00B7 ·	U+00B8 ¸	U+00B9 ¹	U+00BA º	U+00BB »	U+00BC ¼	U+00BD ½	U+00BE ¾	U+00BF ¿
C	U+00C0 À	U+00C1 Á	U+00C2 Â	U+00C3 Ã	U+00C4 Ä	U+00C5 Å	U+00C6 Æ	U+00C7 Ç	U+00C8 È	U+00C9 É	U+00CA Ê	U+00CB Ë	U+00CC Ì	U+00CD Í	U+00CE Î	U+00CF Ï
D	U+00D0 Ð	U+00D1 Ñ	U+00D2 Ò	U+00D3 Ó	U+00D4 Ô	U+00D5 Õ	U+00D6 Ö	U+00D7 ×	U+00D8 Ø	U+00D9 Ù	U+00DA Ú	U+00DB Û	U+00DC Ü	U+00DD Ý	U+00DE Þ	U+00DF ß
E	U+00E0 à	U+00E1 á	U+00E2 â	U+00E3 ã	U+00E4 ä	U+00E5 å	U+00E6 æ	U+00E7 ç	U+00E8 è	U+00E9 é	U+00EA ê	U+00EB ë	U+00EC ì	U+00ED í	U+00EE î	U+00EF ï
F	U+00F0 π	U+00F1 ρ	U+00F2 ς	U+00F3 σ	U+00F4 τ	U+00F5 υ	U+00F6 φ	U+00F7 χ	U+00F8 ψ	U+00F9 ω	U+00FA ϊ	U+00FB ϋ	U+00FC ό	U+00FD ύ	U+00FE ώ	U+00FF ■

A 00000041				Ω 000003A9				語 00008A9E				≡ 00010384				UTF-32																																																																																																																																											
A 0041				Ω 03A9				語 8A9E				≡ D800 DF84				UTF-16																																																																																																																																											
A 41				Ω CE A9				語 E8 AA 9E				≡ F0 90 8E 84				UTF-8																																																																																																																																											
<table><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>1F926</td><td>1F936</td><td>1F946</td><td>1F956</td><td>1F966</td><td>1F976</td><td>1F986</td><td>1F996</td><td>1F9A6</td><td>1F9B6</td><td>1F9C6</td><td>1F9D6</td><td>1F9E6</td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>1F927</td><td>1F937</td><td>1F947</td><td>1F957</td><td>1F967</td><td></td><td>1F987</td><td>1F997</td><td>1F9A7</td><td>1F9B7</td><td>1F9C7</td><td>1F9D7</td><td>1F9E7</td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>1F928</td><td>1F938</td><td>1F948</td><td>1F958</td><td>1F968</td><td></td><td>1F988</td><td>1F998</td><td>1F9A8</td><td>1F9B8</td><td>1F9C8</td><td>1F9D8</td><td>1F9E8</td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>1F929</td><td>1F939</td><td>1F949</td><td>1F959</td><td>1F969</td><td></td><td>1F989</td><td>1F999</td><td>1F9A9</td><td>1F9B9</td><td>1F9C9</td><td>1F9D9</td><td>1F9E9</td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>1F92A</td><td>1F93A</td><td>1F94A</td><td>1F95A</td><td>1F96A</td><td>1F97A</td><td>1F98A</td><td>1F99A</td><td>1F9AA</td><td>1F9BA</td><td>1F9CA</td><td>1F9DA</td><td>1F9EA</td><td></td></tr></table>																														1F926	1F936	1F946	1F956	1F966	1F976	1F986	1F996	1F9A6	1F9B6	1F9C6	1F9D6	1F9E6																1F927	1F937	1F947	1F957	1F967		1F987	1F997	1F9A7	1F9B7	1F9C7	1F9D7	1F9E7																1F928	1F938	1F948	1F958	1F968		1F988	1F998	1F9A8	1F9B8	1F9C8	1F9D8	1F9E8																1F929	1F939	1F949	1F959	1F969		1F989	1F999	1F9A9	1F9B9	1F9C9	1F9D9	1F9E9																1F92A	1F93A	1F94A	1F95A	1F96A	1F97A	1F98A	1F99A	1F9AA	1F9BA	1F9CA	1F9DA	1F9EA	
1F926	1F936	1F946	1F956	1F966	1F976	1F986	1F996	1F9A6	1F9B6	1F9C6	1F9D6	1F9E6																																																																																																																																															
1F927	1F937	1F947	1F957	1F967		1F987	1F997	1F9A7	1F9B7	1F9C7	1F9D7	1F9E7																																																																																																																																															
1F928	1F938	1F948	1F958	1F968		1F988	1F998	1F9A8	1F9B8	1F9C8	1F9D8	1F9E8																																																																																																																																															
1F929	1F939	1F949	1F959	1F969		1F989	1F999	1F9A9	1F9B9	1F9C9	1F9D9	1F9E9																																																																																																																																															
1F92A	1F93A	1F94A	1F95A	1F96A	1F97A	1F98A	1F99A	1F9AA	1F9BA	1F9CA	1F9DA	1F9EA																																																																																																																																															

Figura 1.28 Tabla de símbolos UNICODE. [13]

1.7.5 Código Hamming

Este código fue desarrollado por Richard Hamming en el año 1950, es utilizado para la detección y corrección de errores en la transmisión de datos. Este código agrega bits de paridad al dato que se transmitirá, los bits se enumeran iniciando con el número uno de derecha a izquierda, los bits que ocupen la posición de la potencia de 2 (1, 2, 4, 8 etc.) se llaman bits de paridad, se agregan los que sean necesario de acuerdo con el tamaño del dato que se transmitirá. Para determinar el valor de estos bits se toma el primer bit de paridad y se salta el siguiente se toma el siguiente y se salta el que sigue, así hasta terminar con todos, si la cantidad de bits que se tomaron es par, el bit de paridad es 0 para paridad par y si es impar el bit será 1 para tener número de unos pares. Se toma el siguiente bit de paridad el 2 con el que le sigue (grupo de dos bits) y se saltan dos y se toman los dos que siguen y así hasta terminar con todos, si el número de unos en los bits que se tomaron es

para el bit 2 de paridad será 0 y 1 en caso contrario, para el bit de paridad 4 se toma este con 3 bits más (grupo de cuatro bits) se brincan 4 y se toman los siguientes 4 y así hasta terminar con todos los bits, si el número de unos en los bits tomados es par, este bit de paridad es 0 y 1 en caso contrario, esto se repite para todos los bits de paridad.

Quien recibe el dato deberá aplicar la misma técnica para ver que los datos llegaron todos con paridad par.

Al revisar el bit de paridad potencia 0, deberá llegar con paridad par si es así se pone un 0 como bit menos significativo del dato que nos dirá en que bit está el error si es que lo hay en caso contrario se pone un 1, se prueba el siguiente bit de paridad, 2 (potencia dos a la uno) si es par se pone 0 como siguiente bit del dato de error en caso contrario ponemos 1, se prueba el siguiente bit de paridad 4 (potencia de dos a la dos) y se repite el procedimiento hasta terminar con todos los bits. Se revisa el dato de error si es cero no hubo errores en caso contrario el valor binario de este dato es el bit que tiene el error por lo que se puede corregir cambiándolo de estado.

Ejemplo

Usando paridad par, construir el código de Hamming para el carácter "b" de 7 bits

Código ASCII para la letra "b" 1 1 0 0 0 0 0

Número de bit iniciando en 1

1	2	3	4	5	6	7	8	9	10	11
P	P		P				P			

Se marcan los bits de paridad, es decir los bits que son potencia de 2

En seguida se pone el dato que se desea transmitir en los espacios libres iniciando de izquierda a derecha tanto del dato como de la enumeración de los bits.

1	2	3	4	5	6	7	8	9	10	11
P	P	1	P	1	0	0	P	0	0	0

Se toma el primer bit de paridad para saber qué valor deberá tener para paridad par.

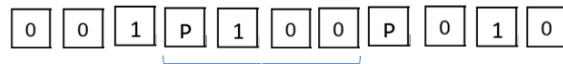
P	P	1	P	1	0	0	P	0	1	0
---	---	---	---	---	---	---	---	---	---	---

Como el número de unos tomados son par el bit de paridad debe ser 0.

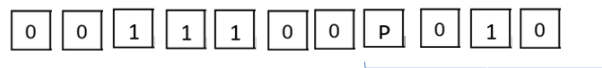
0	P	1	P	1	0	0	P	0	1	0
---	---	---	---	---	---	---	---	---	---	---

Ahora se prueba el siguiente bit de paridad en grupos de dos y saltando dos.

El número de unos son par, por lo tanto, ponemos este 0 y probamos el siguiente bit de paridad

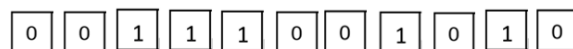


La paridad de unos de esta potencia es impar por lo tanto se pone un 1 para hacerla par y probamos el siguiente bit.



Solo quedan cuatro bits agrupados con paridad impar, por lo tanto, este bit de paridad es 1, ya se terminó, ya se tiene el código Hamming para este dato "b".

Código Hamming para "b" en 7 bits:



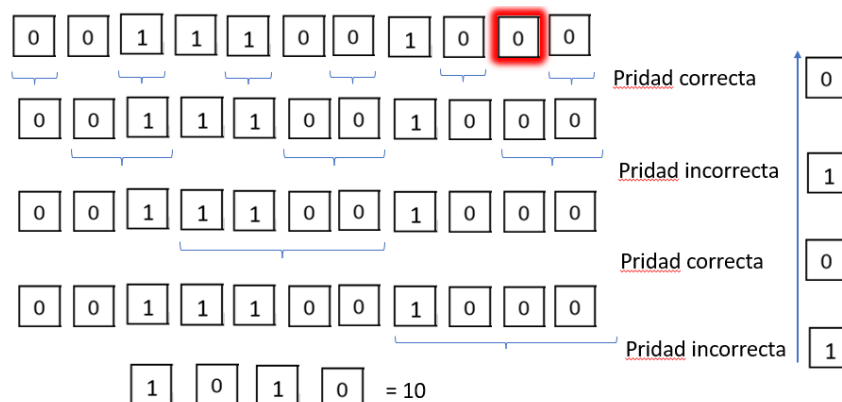
El equipo receptor deberá hacer la prueba de Hamming para asegurar que el dato que llegó es correcto.

Suponiendo que el dato que recibió es:



Hay un error en un bit, al hacer la prueba, esta nos dirá dónde está el error.

Cuando la paridad del bit es correcta se pone un cero, y un 1 cuando es incorrecta, el primer bit obtenido es el bit menos significativo del dato de posición de error y el ultimo el más significativo, si el dato de error es cero entonces el dato llegó correctamente en caso contrario cambiamos de estado el bit que nos diga el dato de error.



El dato de error indica que el bit 10 es incorrecto, por lo tanto, se cambia de estado.

0	0	1	1	1	0	0	1	0	1	0
---	---	---	---	---	---	---	---	---	---	---

Dato recibido y corregido por medio del método Hamming.

Tarea Unidad 1

1. Explique cuáles es la principal característica de las señales analógicas y las digitales.
2. Cuáles son las ventajas de los sistemas digitales frente a los sistemas analógicos.
3. Realice la conversión de dos números decimales con fracción a los restantes sistemas numéricos (entero de 5 cifras y fracción de 3).
4. Escriba dos números decimales de 6 cifras en los códigos BCD y Exceso 3 (investigar código Exceso 3)
5. Escriba el código Gray para los números 42610 y 72710.
6. Qué código Gray son los siguientes: 1001011 y 0111011.
7. Qué número hexadecimal se tiene que transmitir para el código ASCII "Examen" paridad impar.
8. Diga cuales son las ventajas y desventajas de la comunicación serie y paralela.
9. Ponga en paridad par por el método de Hamming el dato 1 1 0 1 0 0 1 1 0 1 1

Examen Unida 1

Lea cuidadosamente la pregunta y seleccione la respuesta correcta.

1 Es la característica principal de las señales analógicas.

- a) Es muy fácil de manipular.
- b) Es muy precisa .
- c) Es discontinua en el tiempo.
- d) Es continua en el tiempo.

2 ¿Cuál de las siguientes cantidades es digital?

- a) Temperatura
- b) Ruido eléctrico.
- c) Interruptor.
- d) Potenciómetro de $1K\Omega$

3 Un sistema analógico procesa señales en forma de unos y ceros.

- a) Verdadero
- b) Falso

4 Para procesar una señal digital es necesario el uso de un convertidor analógico a digital ADC.

- a) Verdadero
- b) Falso

5 Son las dos formas que existen para representar las cantidades.

- a) Binario y Decimal
- b) Analógico y Hexadecimal
- c) Analógica y Digital
- d) Binario y Digital

6 Es una desventaja de los sistemas digitales.

- a) Se requiere mucho espacio para implementar un circuito digital.
- b) Hay pocas formas de almacenar la información.
- c) Se tiene que digitalizar la señal física (analógica) para procesarla.
- d) Tienen poca precisión y exactitud.

7 Convierta a los restantes sistemas numéricos la cantidad 7865.625d y seleccione la opción correcta.

- a) 1EB9.Ah 17271.5oct 1111010111001.101bin
- b) 1E19,Bh 16261.6oct 1111110000110.101bin
- c) 1EB9.10h 17271.5oct 1001010111001.101bin
- d) 1DA9.10h 17271.5oct 1110010111001.101bin

8 Diga si la cadena de unos y cero 1001001101111011 01000111 es un código BCD.

- a) Verdadero

b) Falso

9 Escriba en código exceso 3 la cantidad 1864910

- a) 01001011100101111100
- b) 00011000011001001001
- c) 01001000011001001100
- d) 01001011100001111011

10 Escriba el código Gray 231.

- a) 11101011
- b) 11101011
- c) 10010100
- d) 10010011

11 ¿Qué código Gray es 11001101?

- a) 137
- b) 246
- c) 150
- d) 171

12 El dato, 0 0 1 1 1 0 1 1 0 1 0 1 es recibido en código Hamming en paridad par, diga si se recibió correcto o en qué bit está el error.

- a) El dato es Correcto.
- b) Error en el bit 1
- c) Error en el bit 2
- d) Error en el bit 3
- e) Error en el bit 4
- f) Error en el bit 5
- g) Error en el bit 6
- h) Error en el bit 7
- i) Error en el bit 8
- j) Error en el bit 9
- k) Error en el bit 10
- l) Error en el bit 11

13 ¿Si se quiere transmitir la palabra CLASE en código ASCII paridad par, ¿cuál de los siguientes datos son los correctos?

- a) 43 52 41 51 45
- b) 43 52 C1 D9 45
- c) Ninguna de las anteriores.

Unidad 2

Compuertas lógicas y algebra booleana

2.1 Compuertas básicas.

Las Compuertas lógicas son circuitos electrónicos que se utiliza en los sistemas digitales para llevar a cabo operaciones lógicas, estas pueden tener una o varias entradas y sólo una salida, en la que se mostrara el resultado de la operación realizada con las entradas aplicando las reglas del algebra booleana.

Existen básicamente tres compuertas lógicas que son:

AND

OR

NOT

Compuertas compuestas.

Las compuertas compuestas es una combinación de las compuertas básicas y son:

NAND

NOR

XOR

XNOR

Las tecnologías más comunes con la que se fabrican estas compuertas son:

Tecnología TTL (lógica de transistor transistor)

Está basada en transistores bipolares BJT que conmutan a grandes velocidades pero que consumen más energía que otras tecnologías, la serie para esta familia es la 74XX

Tecnología CMOS (Semiconductor de Oxido Metálico Complementario)

Se basa en transistores MOSFET canal N y canal P (NMOS y PMOS) que son de muy bajo consumo de energía y de alta integración, es decir se pueden fabricar más circuitos digitales CMOS en la misma área que circuitos digitales TTL.

Tecnología BiCMOS (Bipolar – CMOS)

Es una combinación de las tecnologías BJT y CMOS que aprovecha la respuesta de alta velocidad de BJT y el bajo consumo de energía de los transistores CMOS y es utilizada comúnmente que requieren un alto rendimiento como los microprocesadores y circuitos utilizados en redes de telecomunicaciones entre otros.

Variables booleanas

A las entradas y salidas de las compuertas lógicas se les da un nombre normalmente el de una de las letra del abecedario pueden ser mayúscula o minúsculas y son conocidas como variables booleanas ya que sólo pueden tomar dos valores que son el 0 y el 1, el 0

representa el valor de falso (no existe la señal) y el 1 el valor de verdadero (existe la señal), en el circuito electrónico el 0 representa 0 volts, aunque en realidad este valor puede ser de 0 a 0.8 volts o más dependiendo la tecnología con la que es fabricada la compuerta, y el 1 representa 5 volts pero también puede ser desde 3.4 volt o menos dependiendo la tecnología del circuito. Entonces el 0 y el 1 son niveles lógicos que tienen las entradas y las salidas de una compuerta y se les conoce como nivel bajo cuando es 0 y nivel alto cuando es 1 o falso cuando es 0 y verdadero cuando es 1.

Resumiendo, podemos decir que una variable booleana es aquella que se utiliza para representar las entradas y salidas de una compuerta o circuito digital y sólo puede tomar dos valores el 0 y el 1.

Tabla de Verdad

Es una tabla en la que por un lado se representa todas las entradas(variables booleanas) de un circuito digital con todas las combinaciones posibles de ellas(2^n donde n es el número de variables) y por el otro la salida(variable booleana) con el valor que arroja cada combinación de entrada. Son una herramienta fundamental tanto para el diseño de circuitos lógicos como para mostrar el funcionamiento de este, es decir el valor que se mostrará en la salida para cada combinación de las variables de entrada.

Las combinaciones posibles de las entradas son de 2^n donde n es el número de variables.

Tabla de verdad para circuitos con 1, 2, 3 y 4 entradas.

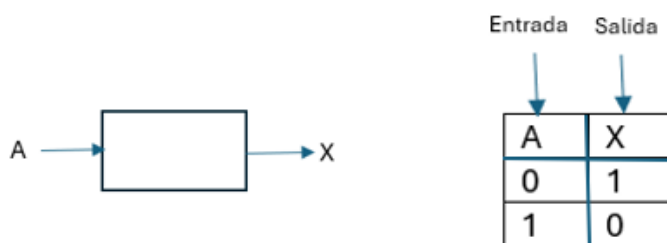


Figura 2.1 Circuito de una entrada y una salida y su tabla de verdad.

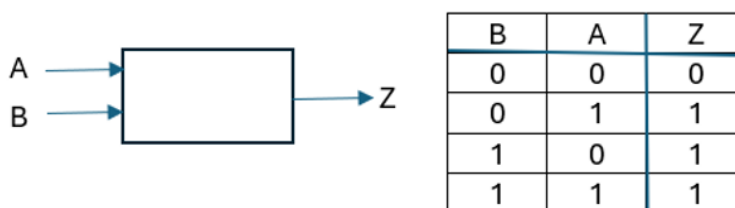


Figura 2.2 Circuito de dos entradas y una salida y su tabla de verdad.

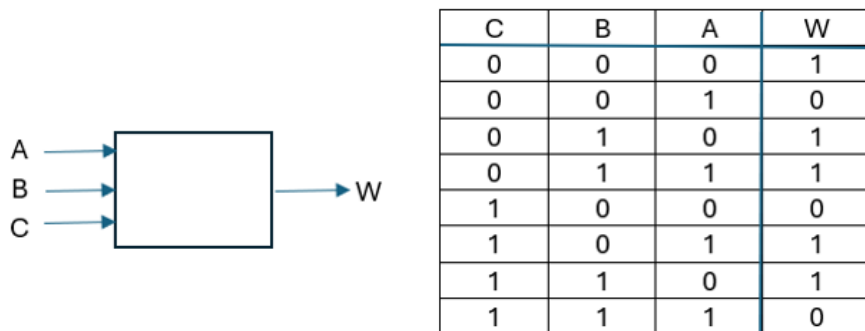


Figura 2.3 Circuito de tres entradas y una salida y su tabla de verdad.

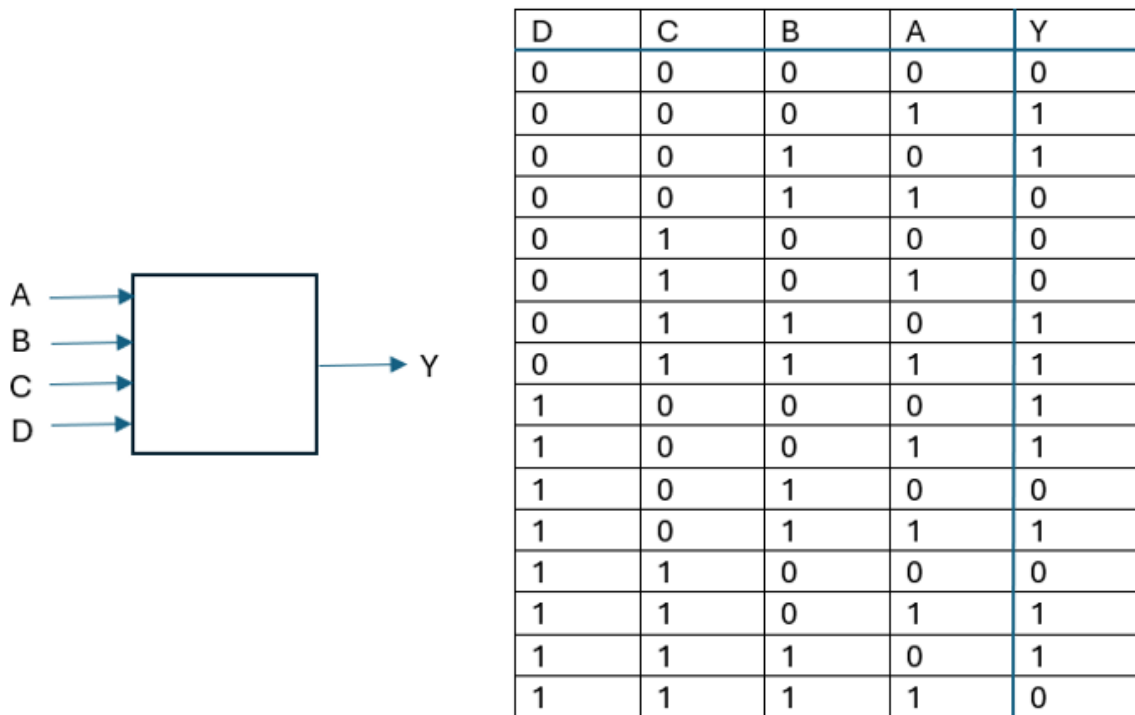


Figura 2.4 Circuito de cuatro entradas y una salida y su tabla de verdad.

Compuerta lógica OR (O)

Esta compuerta lógica produce un 1 de salida cuando cualquiera de sus entradas está en 1 y 0 cuando todas las entradas están en 0, el símbolo lógico de la operación OR es el signo +.

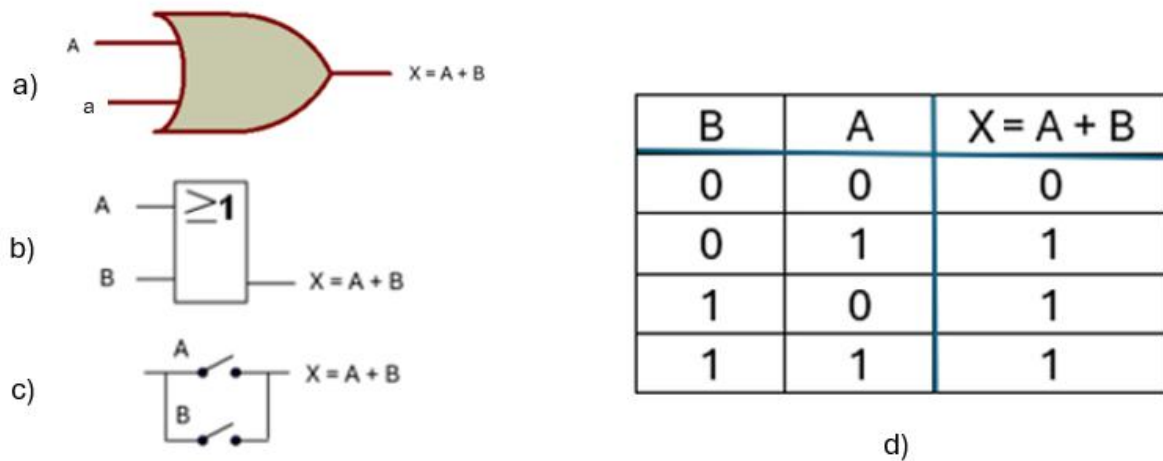


Figura 2.5 Compuerta OR a) símbolo americano o ANSI, b) símbolo europeo o IEC, c) circuito eléctrico y d) tabla de verdad.

La función lógica $X = A + B$ se lee A or B, el signo + no es la suma, si se tiene que $A = 1$ y $B = 1$ entonces la función lógica será $Z = 1 + 1$ es decir la salida toma el valor de 1, $Z = 1$ no es $z = 2$.

Un ejemplo de una compuerta OR de 2 entradas es el circuito integrado 74132 que trae 4 compuertas de tecnología TTL.

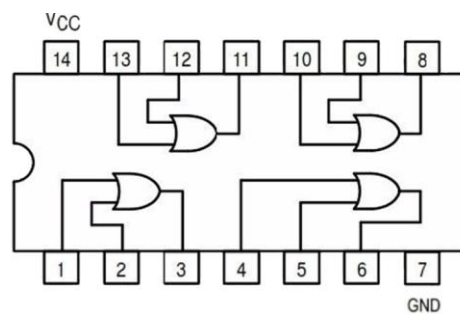


Figura 2.6 Compuerta OR 74132.

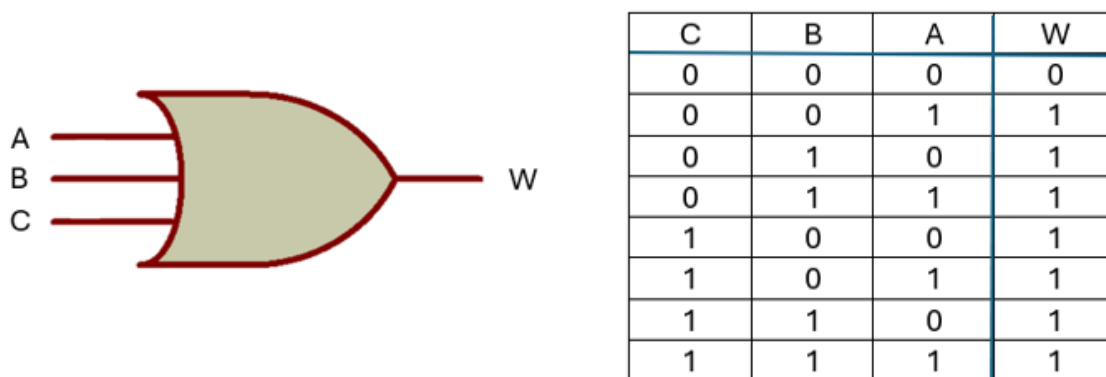


Figura 2.7 Compuerta OR de 3 entradas y su tabla de verdad.

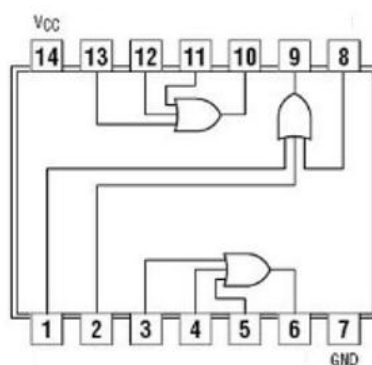


Figura 2.8 Compuerta OR de 3 entradas 744075.

De la compuerta OR podemos resumir:

- 1 una compuerta OR da una salida a nivel alto 1 si alguna de sus entradas está a nivel alto 1 no importando las demás entradas.
- 2 El operativo lógico de la compuerta OR es +.
- 3 La función lógica $W = A + B$ se debe leer "W es igual a A OR B"

Ejemplo de la aplicación de la compuerta OR.

Se tiene una alarma para que se accione cuando una de las dos variables que se están controlando se salga de su rango de operación.

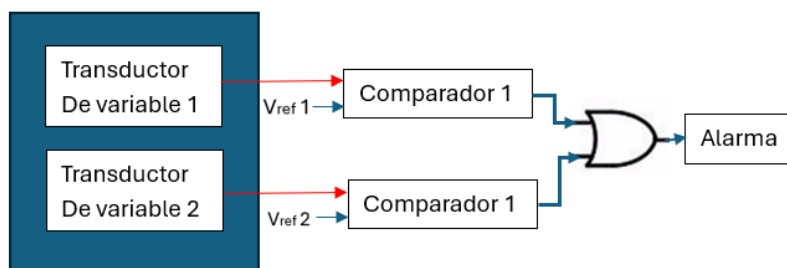


Figura 2.9 Control de alarma para dos variables.

Diagramas de Tiempo

Las cartas o diagramas de tiempo nos muestran el comportamiento de las señales de entrada y salida de un circuito donde estas están cambiando en el tiempo.

Ejemplo

La compuerta OR tiene las entradas A y B que está cambiando en el tiempo y se muestra la salida que produce estas entradas.

En los diagramas de tiempo es conveniente dibujar cursores verticales por cada cambio que exista en las entradas (líneas punteadas), el primer cursor estará marcado por el tiempo t_0 el siguiente cursor por t_1 y así sucesivamente hasta que ya no haya cambio en las señales de entrada, enseguida se revisa el valor de las entradas en t_0 y se dibuja el nivel alto o bajo que se produce de acuerdo a la tabla de verdad de la compuerta, en este caso $A = 0$ y $B = 0$ la salida es 0 y nos detenemos en t_1 , se revisa hasta el inicio de t_2 en este tiempo $A = 1$ y $B = 0$ la salida es 1, se dibuja la salida, esto se repite hasta llegar al último tiempo de nuestro diagrama.

B	A	$X = A + B$
0	0	0
0	1	1
1	0	1
1	1	1

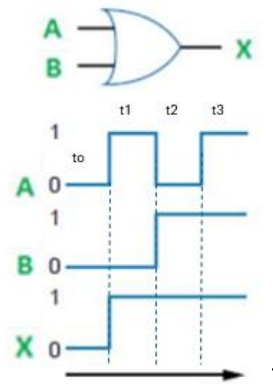


Figura 2.10 Diagrama de tiempo con la compuerta OR.

Compuerta AND (Y)

El operador de la compuerta AND es un punto .

Esta compuerta da una salida a nivel alto 1 sólo cuando todas las entradas están a nivel alto 1, cualquier entrada que se ponga en nivel bajo producirá una salida a nivel bajo 0 sin importar el estado de las demás entradas.

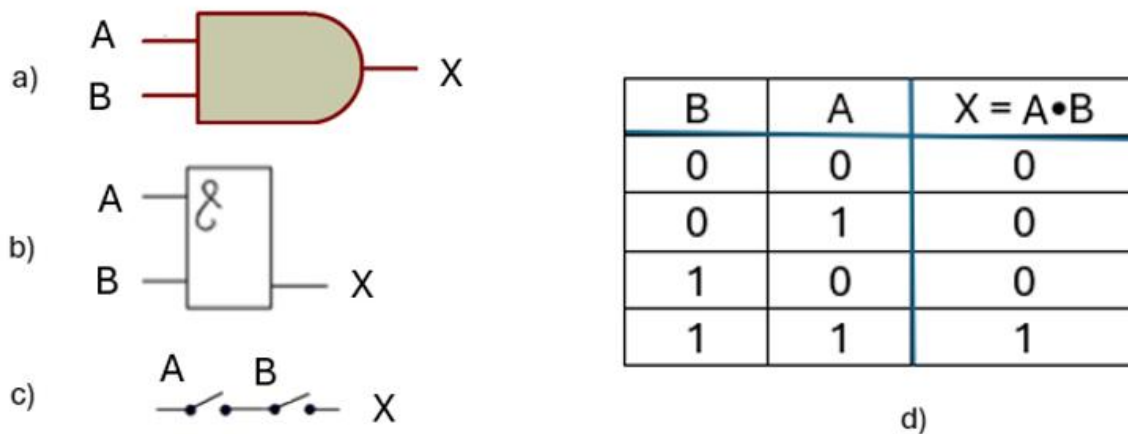


Figura 2.11 Compuerta AND a) símbolo americano o ANSI, b) símbolo europeo o IEC, c) circuito eléctrico y d) tabla de verdad.

La función lógica $X = A \cdot B$ se lee “X igual a A and B” el punto, símbolo del producto es el operador lógico, pero se comporta igual que un producto aritmético según su tabla de verdad, este operador lógico suele omitirse, por lo tanto, se puede escribir la función lógica de la siguiente manera: “ $X = AB$ ” y de esta manera se trabajará en adelante.

Un ejemplo de compuerta and de dos entradas es el circuito integrado 7408, trae 4 compuertas de tecnología TTL.

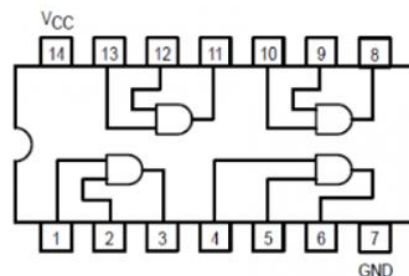


Figura 2.12 Compuerta and de dos entradas 7408 de tecnología TTL.

Existen circuitos integrados de compuerta and de más de dos entradas.

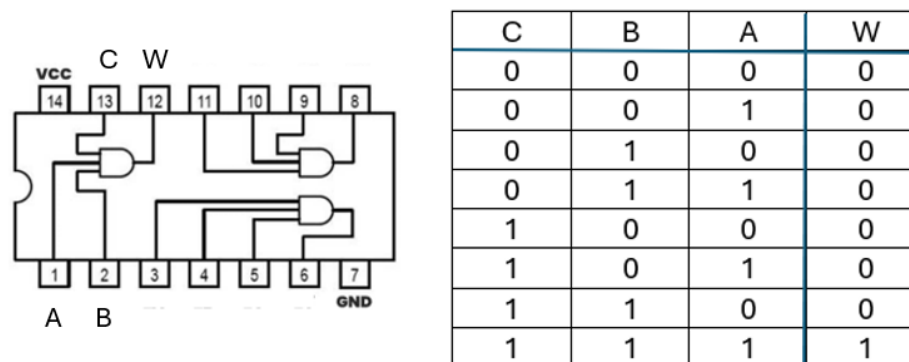


Figura 2.12 Compuerta and de 3 entradas 7411de tecnología TTL y su tabla de verdad.

Ejemplo del uso de la compuerta lógica AND

Se tiene una lavadora de ropa la cual requiere estar cargada de agua y que la puerta de ésta esté cerrada para iniciar su ciclo de lavado. Los sensores son ON/OFF es decir dan un nivel alto cuando se cumple la condición de lavadora llena de agua y puerta cerrada.

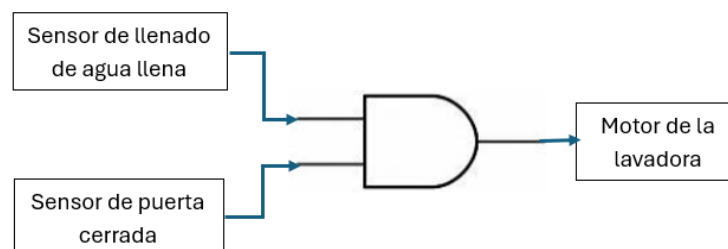


Figura 2.13 control de encendido de una lavadora.

Diagrama de tiempo para la compuerta AND.

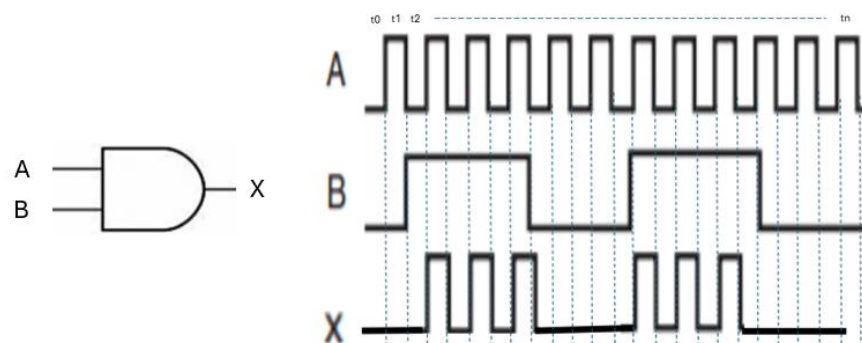


Figura 2.14 Diagrama de tiempo con la compuerta OR.

De la operación de la compuerta AND se puede resumir:

La AND lógica se comporta como una multiplicación normal de unos y ceros.

La compuerta AND sólo da un 1 de salida cuando todas sus entradas son 1, cualquier 0 de entrada produce un 0 de salida.

La función lógica $Z = A B$ se lee “Z igual a A and B”

Compuerta NOT (inversor)

Esta compuerta sólo tiene una entrada y una salida, el valor que presenta en la salida será el complemento del valor de la variable de entrada, si la entrada tiene un 1 la salida será un 0 y si la entrada tiene un 0 la salida será un 1. También se le suele nombrar compuerta inversor o simplemente inversor. El operador lógico es una barra arriba de la variable de salida $\bar{A}$ o también se puede escribir A' .

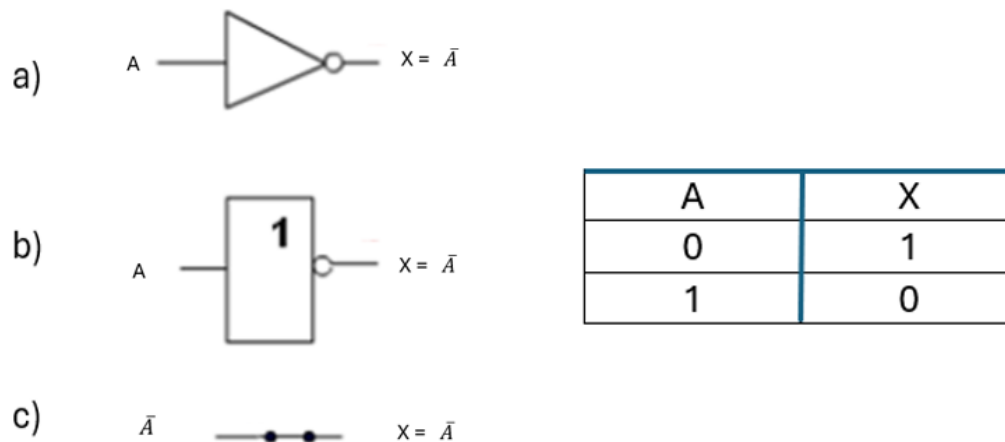


Figura 2.16 Compuerta NOT a) símbolo americano o ANSI, b) símbolo europeo o IEC, c) circuito eléctrico y d) tabla de verdad.

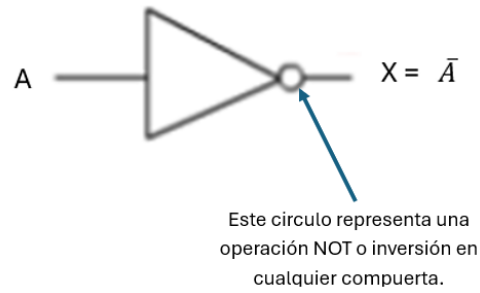


Figura 2.17 El pequeño círculo en las compuertas representa la operación NOT o inversor.

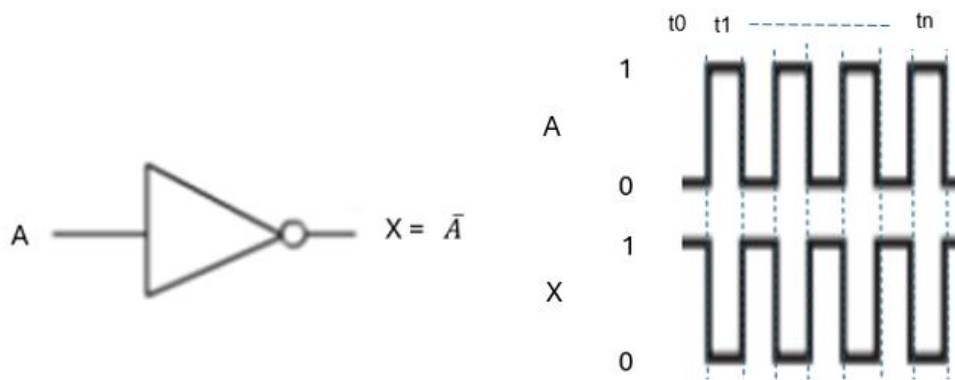


Figura 2.17 Diagramas de tiempo para la compuerta NOT.

Ejemplo

Se tiene un sensor de presencia que da como salida un nivel bajo (0), cuando detecta algo en caso contrario se mantiene en nivel de salida alto (1). Se requiere que se encienda una luz cada que el sensor detecta una presencia.

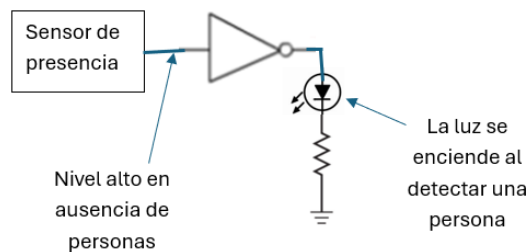


Figura 2.18 Circuito sensor de presencia con la compuerta NOT.

Un ejemplo de circuito integrado para la compuerta NOT es la 7404 que tiene seis compuertas NOT.

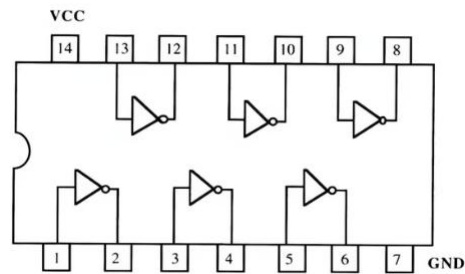


Figura 2.19 Circuito integrado 7404 compuerta NOT.

Compuertas compuestas.

Cualquier compuerta que está formada por más de una compuerta básica es una compuerta compuesta.

Compuerta NAND (NO AND)

Esta compuerta está formada por una compuerta básica AND y una compuerta básica NOT, la salida será nivel bajo (0) sólo cuando todas las entradas estén a nivel alto (1) y será nivel alto (1) cuando cualquier entrada esté a nivel bajo (0).

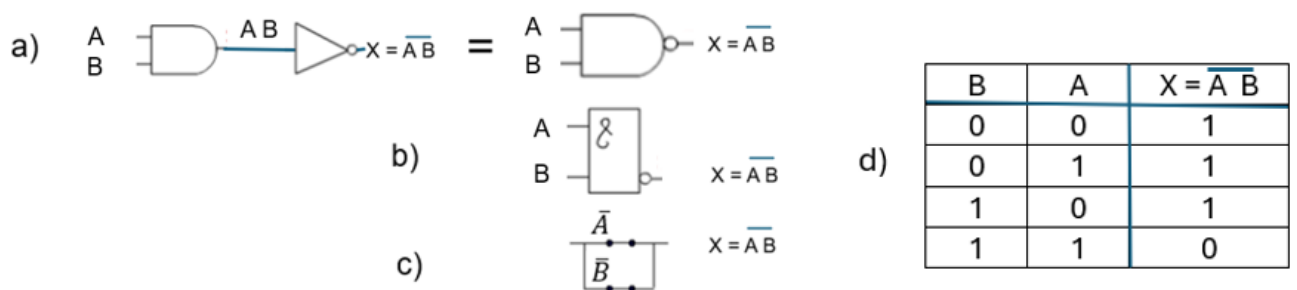


Figura 2.20 Compuerta NAND a) símbolo americano o ANSI, b) símbolo europeo o IEC, c) circuito eléctrico y d) tabla de verdad.

Compuerta NOR (NO OR)

La compuerta NOR está formada por dos compuertas básicas, la compuerta OR y la compuerta NOT, esta compuerta genera un nivel bajo de salida (0) cuando cualquiera de sus entradas es un nivel alto (1) y genera un nivel alto (1) cuando todas sus entradas están a nivel alto (1).

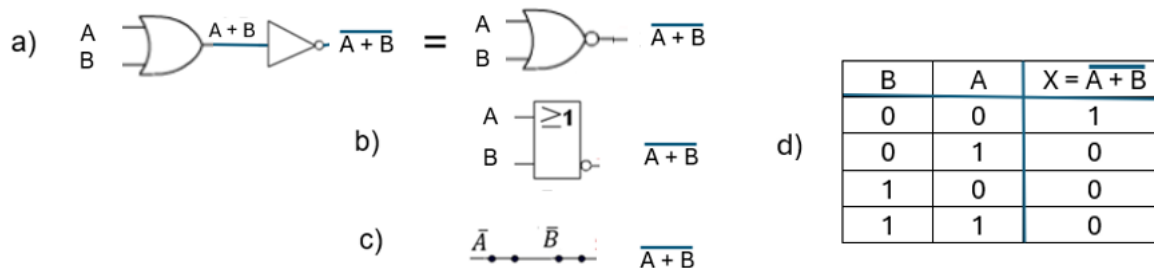


Figura 2.21 Compuerta NOR a) símbolo americano ANSI, b) símbolo europeo o IEC, c) circuito eléctrico y d) tabla de verdad.

Compuerta XOR (OR EXCLUCIVA)

La compuerta XOR cuyo operativo es $\oplus$, es un arreglo de compuertas básicas, dos compuertas AND dos compuertas básicas NOT y una compuerta básica OR, Al evaluar la salida con las cuatro combinaciones de las variables de entrada nos queda el siguiente circuito.

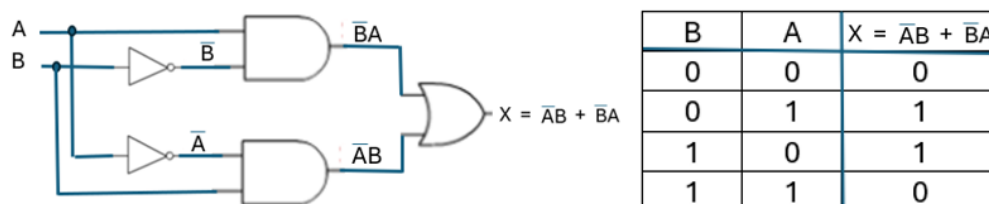


Figura 2.22 Arreglo de compuertas que forman una compuerta XOR.

El arreglo anterior de compuertas es lo que forman como un total la compuerta compuesta XOR.

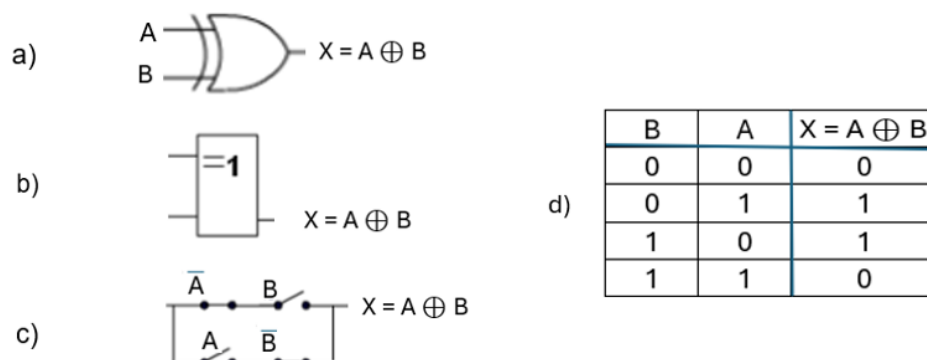


Figura 2.23 Compuerta XOR a) símbolo americano o ANSI, b) símbolo europeo o IEC, c) circuito eléctrico y d) tabla de verdad.

Un ejemplo de un circuito integrado de compuertas XOR es el 7486, tiene cuatro compuertas de dos entradas.

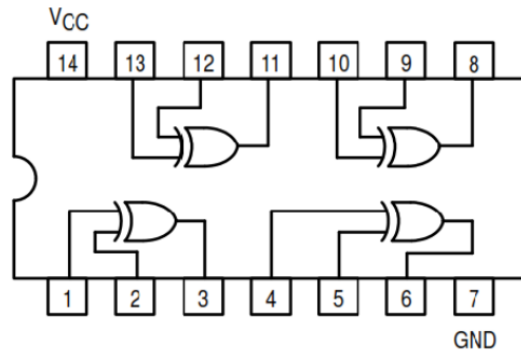


Figura 2.24 Circuito integrado 7486 compuerta XOR.

Los circuitos anteriormente vistos también los hay en tecnología CMOS y suele ser los adecuados para circuitos que trabajarán en ambientes donde se generan señales con mucho ruido eléctrico, la tecnología CMOS soporta alimentaciones desde 3 volts hasta los 18 volts por lo que el margen de ruido que soporta es mucho mayor que la tecnología TTL, algunos números de integrados son los siguientes:

Compuerta NOT se tiene el circuito CMOS y CD4069

Compuerta OR se tiene CD4071, CD4075

Compuerta AND CD4081 y CD4073

Compuerta NAND CD4011 y CD4023

Compuerta NOR ce 4001 y CD4025

Compuerta XOR CD 4070

Expresiones booleanas de un circuito lógico.

Se puede obtener la función lógica de cualquier circuito por muy complejo que este sea, sólo se tiene que ir poniendo a la salida de cada compuerta el valor booleano que le corresponde a las variables que hay en su entrada, esto se hace hasta llegar a la salida de todo el circuito.

Ejemplo

Obtener la función booleana del siguiente circuito:

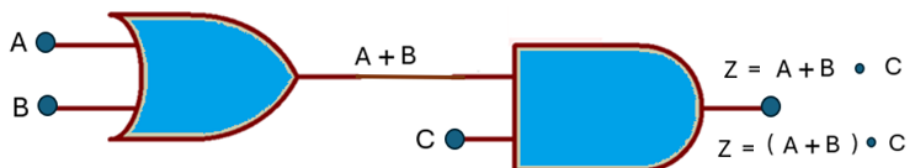


Figura 2.25 Circuito lógico y su función booleana de salida.

Como se puede observar se representa la salida Z de dos formas, la primera no tiene paréntesis en este caso se debe tener en cuenta que en los operadores lógicos primero se realiza la operación AND y luego la OR. En este caso se podría interpretar la salida de dos maneras, uno $Z = A + (BC)$ y el otro $(A+B)C$ por lo que es recomendado usar los paréntesis para evitar confusión.

Ejemplo

Encuentre la función booleana del siguiente circuito.

El procedimiento siempre es el mismo iniciamos con la o las compuertas de entrada y vamos escribiendo en la salida de estas lo que le corresponda de acuerdo con las entradas y el operando de la compuerta, lo repetimos hasta llegar a la salida, si hay un inversor todas las variables que entren en este deben ser negadas con una barra que abarque a todas las variables incluyendo los operadores lógicos.

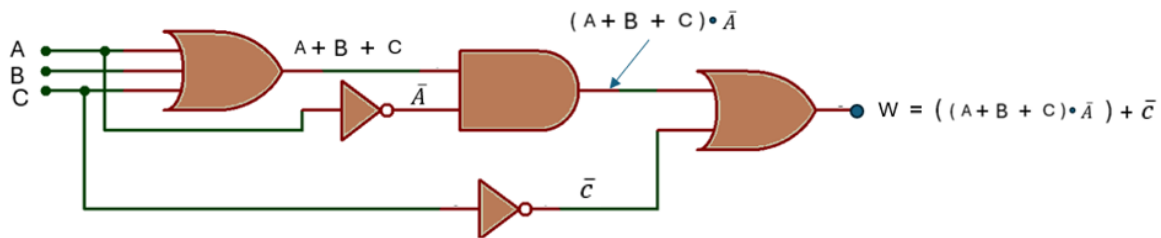


Figura 2.26 Circuito lógico y su función booleana de salida.

Ejemplo

Encuentre la función booleana del siguiente circuito.

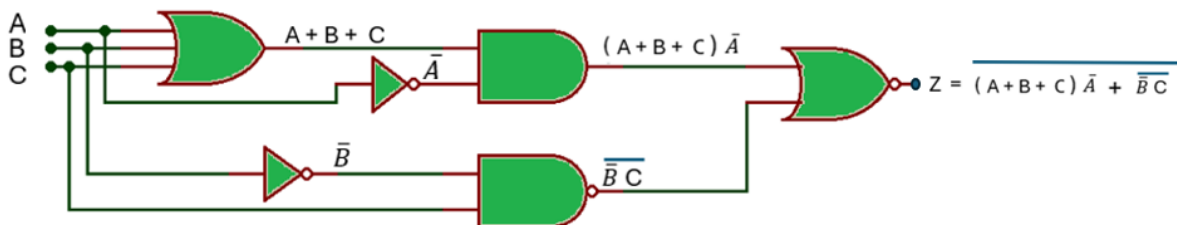


Figura 2.27 Circuito lógico y su función booleana de salida.

Es importante ir anotando a la salida de cada compuerta lo que produce de acuerdo con las entradas y tener mucho cuidado de utilizar los operadores lógicos que le corresponde a cada compuerta.

Tablas de verdad de los circuitos lógicos.

Para encontrar la tabla de verdad de un circuito lógico en particular, se debe evaluar poniendo las diferentes combinaciones posibles de las variables de entrada y anotar a la salida que es lo que nos produce por cada una de ellas, esto puede ser un 0 o un 1 de salida.

Ejemplo

Evaluar el siguiente circuito.

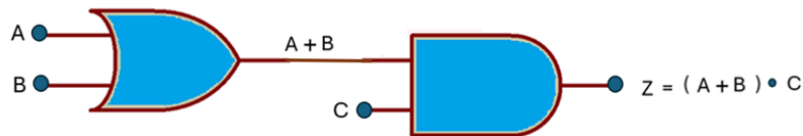


Figura 2.28 Circuito lógico a evaluar.

Este circuito tiene tres variables de entrada, por lo tanto, tiene ocho combinaciones posibles en la que se tiene que evaluar para ver qué es lo que produce a la salida.

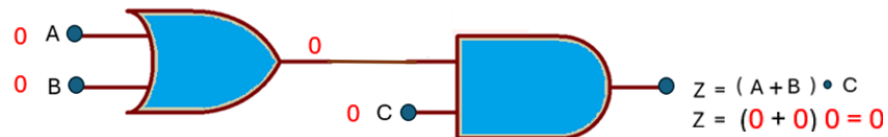


Figura 2.29 Circuito lógico evaluado con la combinación de entrada A=0, B=0, C=0.

En la figura 2,29 se muestra la evaluación con la combinación de las variables todas en 0, al ir poniendo las diferentes combinaciones de las variables de entrada el circuito nos arroja los siguientes resultados, y decimos que esta es su tabla de verdad.

A	B	C	Z
0	0	0	0
0	0	1	0
0	1	1	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Figura 2.30 Tabla de verdad del circuito de la figura 2.29.

Evaluación de un circuito por pasos con tablas de verdad.

Para obtener la tabla de verdad de un circuito por partes, se escribe la tabla de verdad con todas las combinaciones posibles de las variables de entrada y enseguida se escribe una columna por cada salida de las compuertas que forman el circuito y se procede a evaluarlo con todas las combinaciones, las compuertas que tengan inversor a la salida primero se evalúa la compuerta sin inversor y luego se invierte el resultado.

Ejemplo obtenga la tabla de verdad para el siguiente circuito:

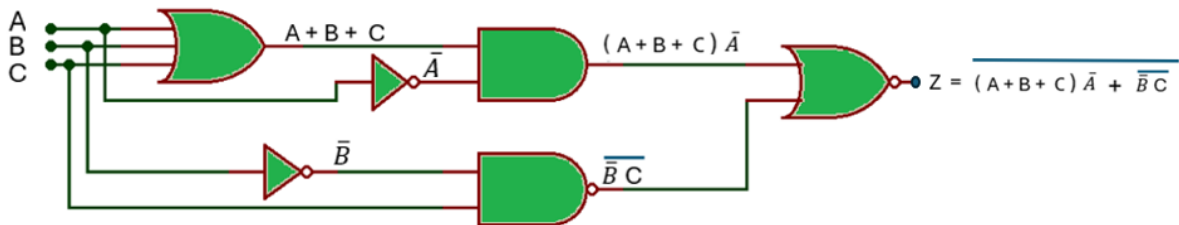


Figura 2.31 Circuito para evaluar por tablas de verdad.

A	B	C	$A+B+C$	$\bar{A}$	$\bar{B}$	$(A+B+C)\bar{A}$	$\bar{B}C$	$\overline{\bar{B}C}$	$Z = (A+B+C)\bar{A} + \bar{B}C$	$Z = \overline{(A+B+C)\bar{A} + \bar{B}C}$
0	0	0	0	1	1	0	0	1	1	0
0	0	1	1	1	1	1	1	0	1	0
0	1	0	1	1	0	1	0	1	1	0
0	1	1	1	1	0	1	0	1	1	0
1	0	0	1	0	1	0	0	1	1	0
1	0	1	1	0	1	0	1	0	0	1
1	1	0	1	0	0	0	0	1	1	0
1	1	1	1	0	0	0	0	1	1	0

Figura 2.32 Tabla de verdad para el circuito de la figura 2.31.

Evaluación de funciones booleanas.

Evaluar la siguiente función booleana para los valores de entrada que se muestran:

A=1, B=0, C=1

$$Z = \overline{(A+B+C)\bar{A} + \bar{B}C}$$

Para evaluar la función lógica se llevan a cabo los siguientes pasos:

Primero cambiamos las variables con los valores que se desean evaluar.

$$Z = \overline{(1+0+1)\bar{1} + \bar{0}1}$$

Segundo niegue las variables individuales que tengan una barra y elimine la barra.

$$Z = (1 + 0 + 1) 0 + \overline{11}$$

Tercero efectúe las operaciones que tengan paréntesis.

$$Z = (1) 0 + \overline{11}$$

Cuarto si un término de más de una variable tiene una negación, primero realice la operación lógica.

$$Z = (1) 0 + \overline{1}$$

Quinto niegue el resultado de esta operación.

$$Z = (1) 0 + 0$$

Sexto efectúe las operaciones lógicas que falta.

$$Z = \overline{0}$$

Séptimo si la variable de salida esta negada invierta su valor y quite la negación.

$$Z = 1$$

Implementación de un circuito lógico a partir de su función booleana.

Para dibujar el circuito lógico de una expresión booleana primero se identifican en la expresión todas las compuertas que la componen, las variables individuales que tengan negación se cablean al final.

Ejemplo

Dibuje el circuito de la siguiente expresión:

$$\overline{(\overline{A}\overline{B} + \overline{A}C) + \overline{A}B\overline{C}} = x$$

Primero se identifican las compuertas que forman la expresión booleana.

$$\underbrace{\underbrace{\overline{A}\overline{B}}_{\text{NAND}} + \underbrace{\overline{A}C}_{\text{AND}}}_{\text{OR}} + \underbrace{\overline{A}B\overline{C}}_{\text{NAND}} = x$$

NOR

Luego se implementa el circuito teniendo cuidado de no olvidar negar las variables individuales que así lo estén en la expresión booleana.

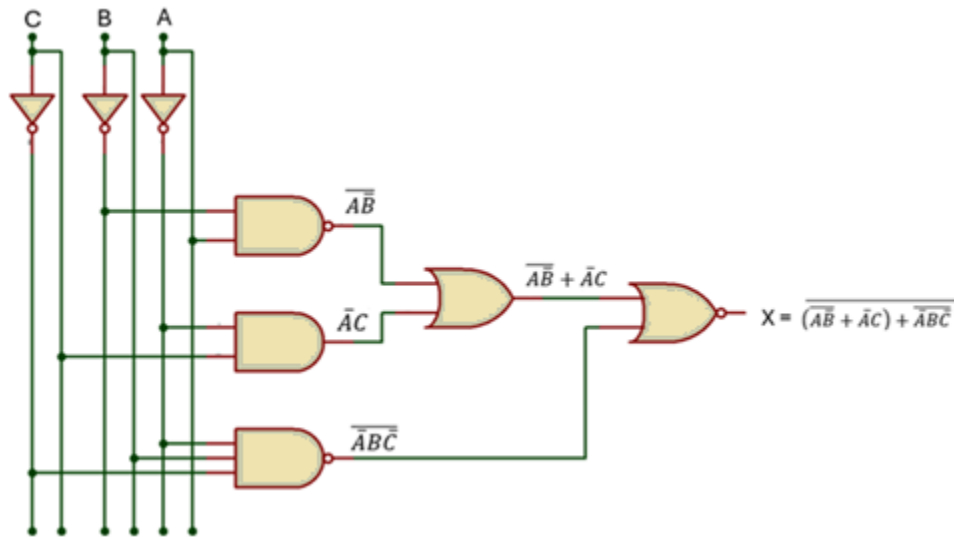


Figura 2.33 Circuito final obtenido de la expresión booleana.

Equivalencia entre compuertas

Para saber si el arreglo de compuertas es igual a una en particular, se obtiene la tabla de verdad de ambos circuitos y si las tablas son iguales, entonces los circuitos son equivalentes. Se hará la demostración de la primera equivalencia, las demás quedan para su demostración.

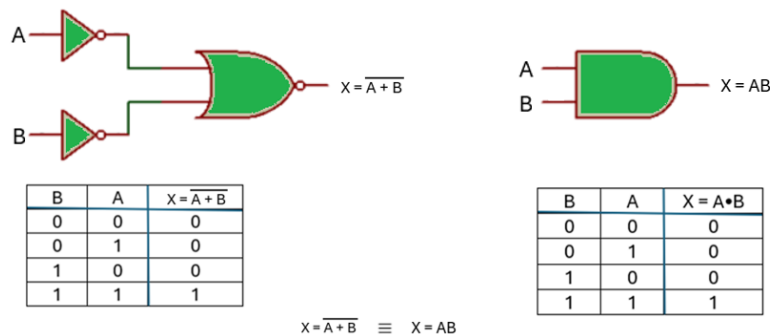


Figura 2.34 circuitos equivalentes.

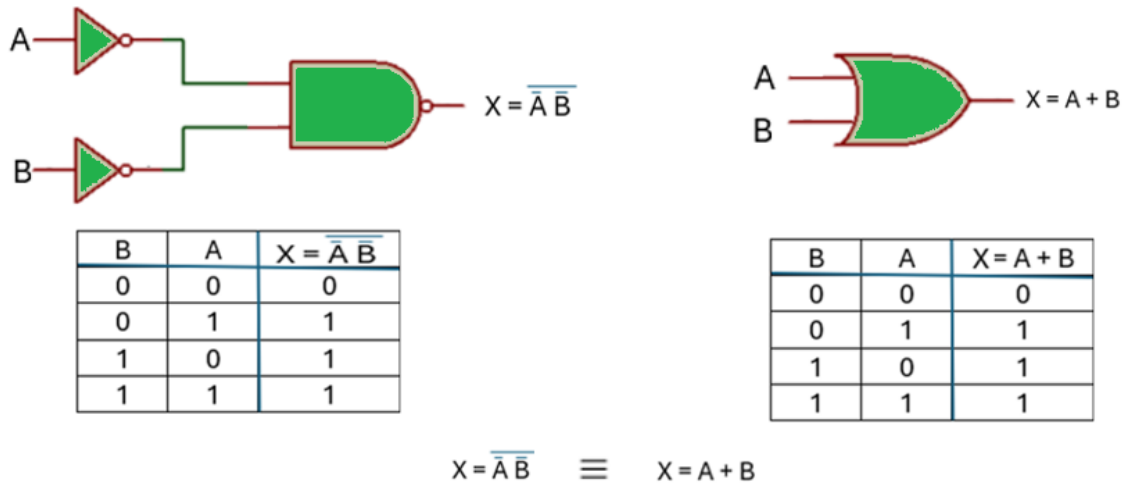


Figura 2.35 circuitos equivalentes.

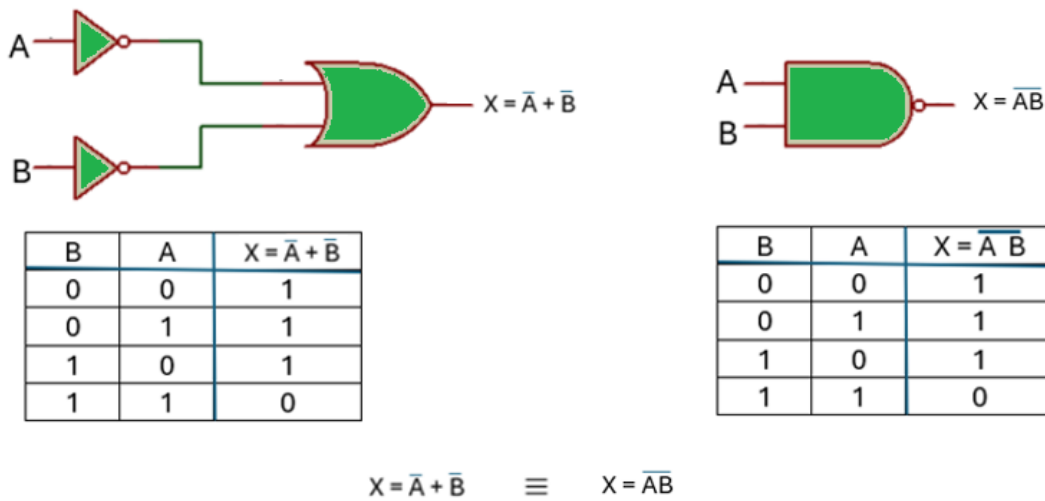


Figura 2.36 circuitos equivalentes.

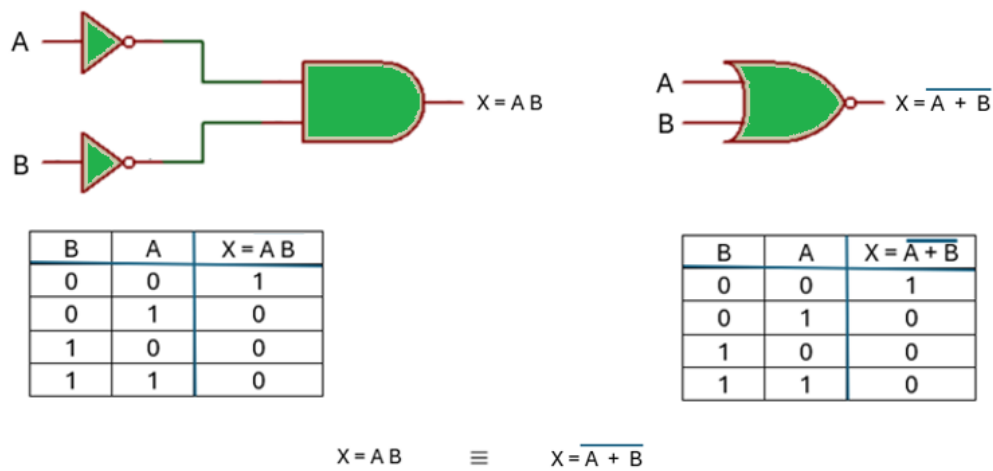


Figura 2.37 circuitos equivalentes.

2.2 Familias lógicas

Introducción a las Familias Lógicas

En el área de la electrónica digital, las familias lógicas son circuitos electrónicos que implementan compuertas lógicas que manejan principios de diseño, niveles de voltaje y características de conmutación similares.

Las principales propiedades de los circuitos digitales suelen ser:

Inmunidad al ruido.

El consumo de energía.

La velocidad de operación.

Los niveles de voltaje con los que opera.

Características principales de las familias lógicas.

La clasificación de las familias lógicas se determina por el comportamiento eléctrico y la tecnología de los semiconductores con la que se construyen, las siguientes propiedades son muy importantes:

Retardo de Propagación.

La duración de tiempo requerida para que una señal atraviese una compuerta.

Consumo de Energía.

La cantidad de energía que consume una compuerta durante su operación.

Inmunidad al Ruido.

Capacidad para soportar variaciones de voltaje sin afectar su funcionamiento.

Niveles de Voltaje.

Especifica los voltajes lógicos para ser interpretados como ALTO y BAJO.

Entre las principales familias lógicas se encuentran TTL (Transistor-Transistor Logic), CMOS (Complementary Metal-Oxide-Semiconductor), ECL (Emitter-Coupled Logic) y otras variantes híbridas.

1. Familia TTL (Transistor-Transistor Logic)

La tecnología TTL fue una de las primeras en popularizarse en la electrónica digital. Se basa en transistores bipolares para implementar las funciones lógicas y se caracteriza por su velocidad moderada y su robustez. Algunas subfamilias de TTL incluyen:

- TTL Estándar: La primera versión, con un consumo de energía relativamente alto.
- TTL de Baja Potencia (L, LS, ALS): Variantes que reducen el consumo eléctrico.
- TTL de Alta Velocidad (H, S, AS): Diseñadas para mejorar los tiempos de conmutación.

A pesar de su amplio uso, la TTL ha sido reemplazada en muchas aplicaciones por tecnologías más eficientes como la CMOS.

2. Familia CMOS (Complementary Metal-Oxide-Semiconductor)

La familia CMOS se caracteriza por su bajo consumo de energía y alta inmunidad al ruido, gracias al uso de transistores MOS complementarios (PMOS y NMOS). Sus principales características son:

- Bajo consumo de energía: Solo consume corriente en el cambio de estado lógico.

- Alta impedancia de entrada: Reduce la carga en las interconexiones.
- Compatibilidad con distintos niveles de voltaje: Permite la integración con diversas tecnologías.

Las subfamilias de CMOS incluyen:

- 4000B: Primeros circuitos CMOS, de baja velocidad, pero alta flexibilidad.
- 74HC y 74HCT: Versiones compatibles con TTL, mejorando la velocidad y reduciendo el consumo.

Gracias a sus ventajas, la familia CMOS es la más utilizada en la actualidad para el diseño de circuitos integrados.

3. Familia ECL (Emitter-Coupled Logic)

La tecnología ECL utiliza transistores bipolares en configuraciones de acoplamiento por emisor, lo que permite una operación extremadamente rápida. Sus principales características son:

- Altas velocidades de conmutación: Opera en el rango de nanosegundos.
- Bajo retardo de propagación: Adecuado para aplicaciones de alta frecuencia.
- Alto consumo de energía: Debido a que los transistores nunca están completamente apagados.

ECL se utiliza principalmente en sistemas de telecomunicaciones, procesamiento de datos y supercomputadoras.

4. Familias Híbridas y Otras Variantes

Además de las familias principales, existen combinaciones híbridas y tecnologías especializadas, tales como:

- BiCMOS: Integra la alta velocidad de los bipolares con el bajo consumo de CMOS.
- GTL (Gunning Transceiver Logic): Diseñado para minimizar el ruido y mejorar la integridad de la señal en buses de datos.
- LVDS (Low Voltage Differential Signaling): Utilizado en transmisiones de alta velocidad con bajo voltaje.

Nomenclatura de los circuitos digitales

A pesar de la existencia de numerosos fabricantes de dispositivos electrónicos digitales, la mayoría de la nomenclatura utilizada es estándar a continuación, se muestra los términos más utilizados.

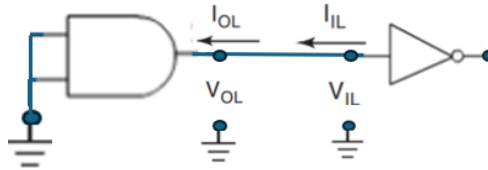


Figura 2.37 Parámetros de salida y entrada para un nivel bajo.

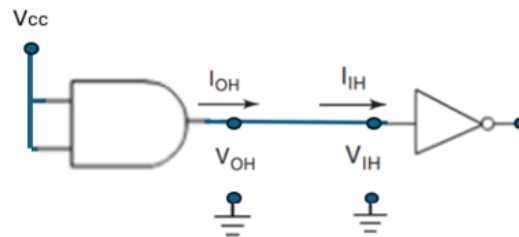


Figura 2.38 Parámetros de salida y entrada para un nivel alto.

Los siguientes parámetros los da el fabricante de los circuitos integrados y normalmente se presenta los valores mínimos, típicos y los máximos, estos deben tenerse en cuenta para el correcto funcionamiento de estos.

$V_{OL}(\text{máx})$ Es el voltaje máximo que entrega un circuito cuando su salida está en nivel bajo.

I_{OL} Corriente de salida para un nivel bajo, esta corriente entra a la compuerta.

I_{IL} Corriente de entrada para un nivel bajo.

$V_{IL}(\text{máx})$ Voltaje máximo de entrada para que sea reconocido como nivel bajo.

I_{OH} Corriente de salida a nivel alto.

$V_{OH}(\text{mín})$ Voltaje de salida a nivel alto.

I_{IH} Corriente de entrada a nivel alto.

V_{IH} Voltaje de entrada a nivel alto.

Las corrientes que se muestran en las figuras 2.37 y 2.38 pueden tener dirección diferente, esto depende de las distintas familias lógicas.

Tiempo de retardo en la propagación.

Cuando se pone una señal de entrada en un circuito lógico, este tarda un tiempo en responder a la salida, a estos tiempos se les conoce como tiempos de propagación, el fabricante presenta estos tiempos en las hojas de datos y los define de la siguiente manera:

t_{PLH} Es el tiempo que tarda en pasar la señal de salida de un circuito lógico de un nivel alto a un nivel bajo.

t_{PLH} Es el tiempo que tarda en pasar la señal de salida de un circuito lógico de un nivel bajo a un nivel alto.

Normalmente estos tiempos se toman a partir del cincuenta por ciento de la amplitud de la señal de entrada y el cincuenta por ciento en la señal de salida, como se muestra en la siguiente figura. Estos tiempos **t_{PHL}** y **t_{PLH}** pueden ser diferentes.

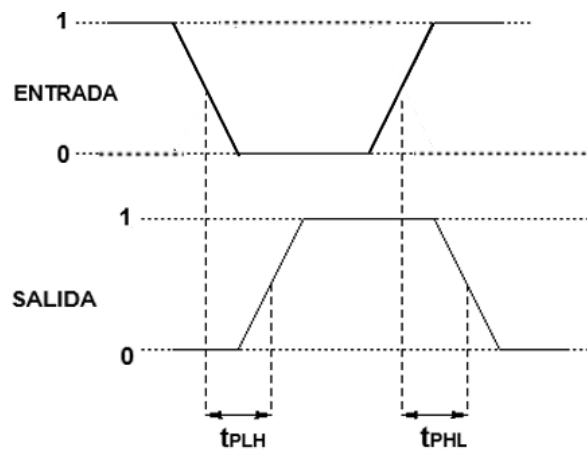


Figura 2.39 Tiempos de propagación.

Margen de ruido.

Normalmente los circuitos lógicos se trabajan en lugares donde hay campos eléctricos y magnéticos lo que ocasionan voltajes no deseados, a esto se le conoce como ruido, este se suma a las señales de entrada y pueden hacer que cambie por debajo o por encima de los valores $V_{IH\text{mín}}$ y $V_{IL\text{máx}}$ lo que provocaría un funcionamiento no deseado.

La inmunidad a este ruido se le conoce como margen de ruido y se denota como V_{NL} para el margen de ruido a nivel bajo y V_{NH} para el nivel alto. Estos valores se pueden calcular con los parámetros que da el fabricante de los circuitos y es diferente para los distintos circuitos integrados, las fórmulas para calcular estos tiempos son las siguientes:

Margen de ruido para el estado alto.

$$V_{NH} = V_{OH}(\text{mín}) - V_{IH}(\text{mín})$$

Margen de ruido para el estado bajo

$$V_{NL} = V_{IL}(\text{máx}) - V_{OL}(\text{máx})$$

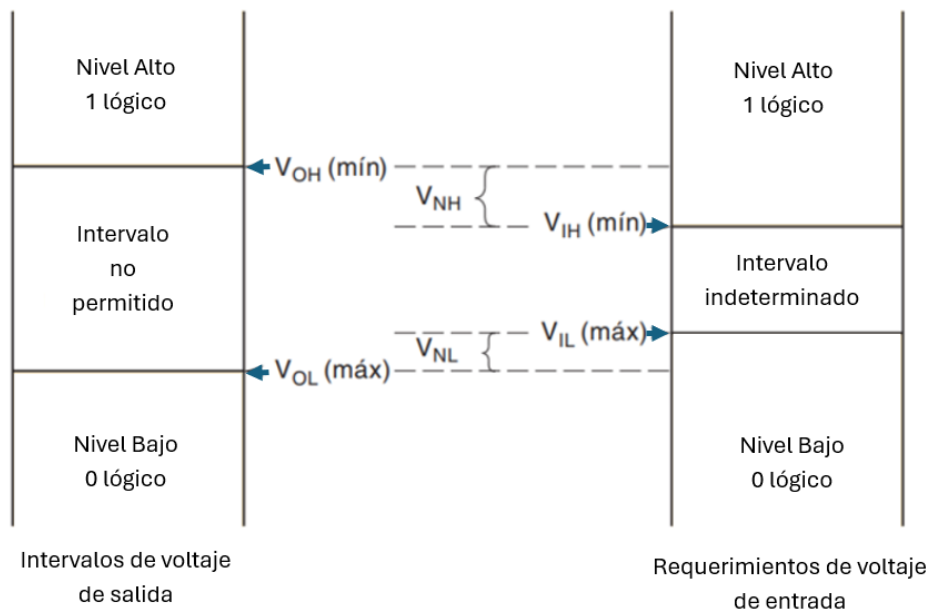


Figura 2.40 Márgenes de ruido de voltaje.

La figura anterior nos indica que por ningún motivo debemos caer en la zona de Intervalo no permitido ni en la zona de intervalo indeterminado ya que la respuesta del circuito es impredecible.

Ejemplo

Encontrar los márgenes de ruido para el nivel alto y bajo de la familia TTL estándar cuyos valores proporcionados por el fabricante son:

Parámetro	Voltage mínimo	Voltage típico	Voltage máximo
V_{OH}	2.4	3.4	
V_{OL}		0.2	0.4
V_{IH}	2.0		
V_{IL}			0.8

Tabla 2.1 Valores de voltaje de entrada y salida de la familia TTL estándar.

$$V_{NH} = V_{OH}(\text{mín}) - V_{IH}(\text{mín})$$

$$V_{NH} = 2.4V - 2.0V$$

$V_{NH} = 0.4$ volts.

$$V_{NL} = V_{IL(\text{máx})} - V_{OL\text{máx}}$$

$$V_{NL} = 0.8 \text{ v} - 0.4 \text{ v}$$

$$V_{NL} = 0.4 \text{ volts}$$

Lógica de suministro y drenado de corriente.

Las diferentes familias lógicas pueden ser descritas de acuerdo con la trayectoria de la corriente entre la salida de un circuito y la entrada de otro.

Suministro de corriente

Cuando una compuerta tiene como salida un nivel alto V_{OH} , la trayectoria de la corriente es de salida, es decir esta compuerta suministra corriente para la entrada de la compuerta de carga I_{IH} la cual se comporta como una resistencia a tierra.

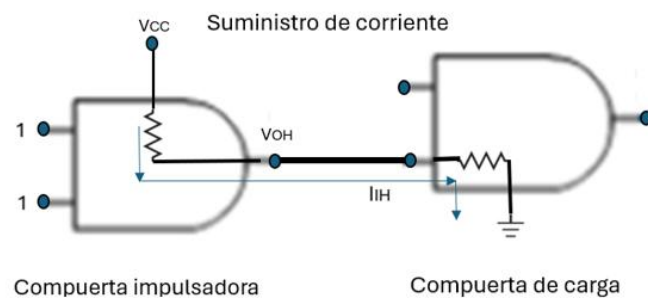


Figura 2.41 Suministro de corriente.

Drenado de corriente.

Cuando una compuerta tiene la salida a nivel bajo V_{OL} , se comporta como una resistencia a tierra y la trayectoria de la corriente es hacia dentro de la compuerta, es decir drena la corriente.

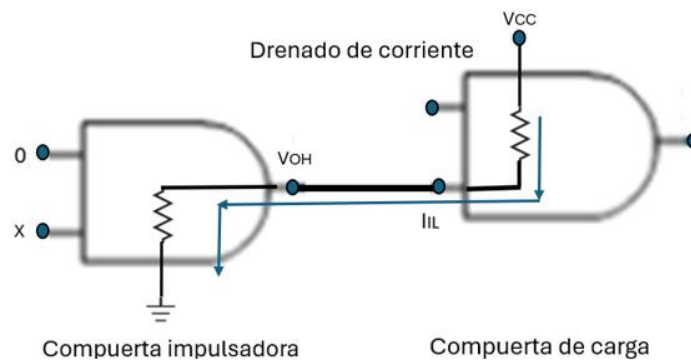


Figura 2.42 Drenado de corriente.

Potencia disipada.

Todos los circuitos integrados requieren de una cierta cantidad de energía para su correcto funcionamiento, la potencia promedio que disipa un circuito se puede calcular multiplicando la corriente promedio por el voltaje de alimentación de este. $I_{CC} * V_{CC}$ o $I_{DD} * V_{DD}$.

La corriente promedio $I_{CC}(\text{prom})$ se calcula sumando la corriente que consume el circuito cuando todas sus salidas son nivel bajo más la corriente que consume cuando todas las salidas son nivel alto dividida entre dos.

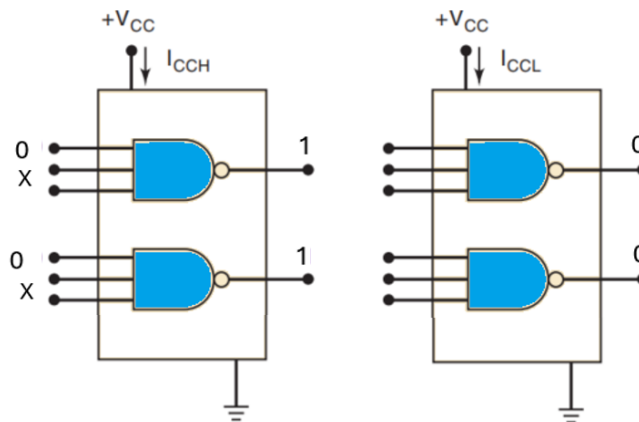


Figura 2.43 Consumo de corriente a nivel bajo y a nivel alto.

$$I_{CC}(\text{prom}) = \frac{I_{CCH} + I_{CCL}}{2}$$

La potencia promedio que disipa el circuito será:

$$PD(\text{prom}) = I_{CC}(\text{prom}) \times V_{CC}$$

Factor de carga (fan-out)

El fan-out indica la cantidad de entradas que puede impulsar una compuerta con su salida ambas de la misma familia y viene determinada por las siguientes formulas:

Factor de carga a nivel bajo = $I_{OL}(\text{máx}) / I_{IL}(\text{máx})$

Factor de carga a nivel alto = $I_{OH}(\text{máx}) / I_{IH}(\text{máx})$

Ejemplo

Una compuerta tiene una corriente de salida máxima a nivel bajo $I_{OL(m\acute{a}x)}$ de 7mA, y una corriente máxima de salida a nivel alto $I_{OH(m\acute{a}x)}$ de 0.3 mA, decir cuantas entradas a nivel bajo y a nivel alto puede impulsar si la corriente de entrada máxima a nivel bajo $I_{IL(m\acute{a}x)}$ es de 0.2 mA y la corriente de entrada a nivel alto máxima $I_{IH(m\acute{a}x)}$ es de 18 μ A.

Factor de carga a nivel bajo = $7\text{mA} / 0.2 \text{ mA} = 35$, se pueden impulsar 35 entradas de la misma familia.

Factor de carga a nivel alto = $0.3 \text{ mA} / 18 \mu\text{A} = 16.6$, puede impulsar 16 entradas a nivel alto de la misma familia.

Entradas a un circuito lógico con resistencia a tierra.

Cuando se polariza una entrada a un circuito lógico con una resistencia para poder cambiar su estado, es importante dar un valor adecuado a la resistencia para evitar que el voltaje que se produce en ella por la corriente que devuelve la entrada sea mayor al $V_{IL(m\acute{a}x)}$ que permite el circuito. Este valor se calcula de la siguiente manera:

$$R_{m\acute{a}x} = \frac{V_{IL(m\acute{a}x)}}{I_{IL}}$$

Ejemplo

Calcular el valor de la resistencia del siguiente circuito para evitar rebasar el valor $V_{IL(max)}$ de 0.6v si la corriente de entrada a nivel bajo I_{IL} de la compuerta es de 0.4 mA.

$R_{m\acute{a}x} = 0.6\text{v} / 0.4\text{mA} = 1500 \Omega$, este valor puede ser redondeado al valor comercial más próximo hacia abajo.

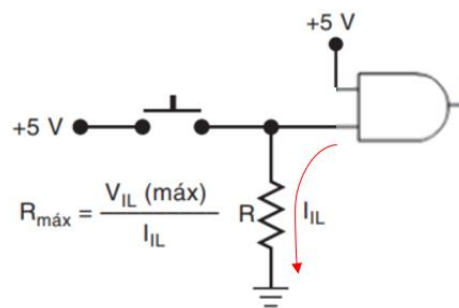


Figura 2.44 Resistencia de polarización a tierra.

Encapsulado de los circuitos integrados.

En la actualidad existen una variedad de encapsulados de los circuitos integrados, este depende de la aplicación del circuito, disipación de calor, tamaño de la placa donde se montará entre otros aspectos. Los encapsulados más comunes son:

Montaje de inserción

Los que atraviesan la placa para su montaje.

DIP (Dual In-line Package) Encapsulado dual en líneas. Los pines se extienden a lo largo del encapsulado en ambos lados y tienen una marca como todos los demás para indicar la terminal 1.

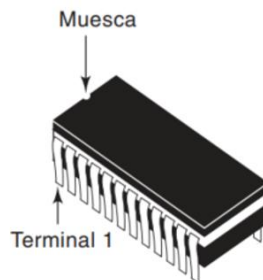


Figura 2.43 Circuito integrado con encapsulado DIP.

SIP (Single In-line Package) Paquete en una sola línea.

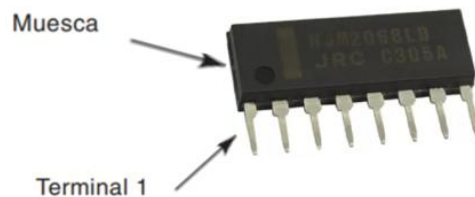


Figura 2.44 Circuito integrado con encapsulado SIP.

PGA (Pin Grid Array) matriz de pines, encapsulado para circuitos integrados (ICs) que se caracteriza por tener una matriz de pines en la parte inferior, organizados en una cuadrícula. Estos pines se insertan en un zócalo especial en la PCB o, en algunos casos, se sueldan directamente.



Figura 2.45 Circuito integrado con encapsulado PGA.

Montaje superficial.

Los que quedan montados sobre la placa.

SOP (Small Outline Package) Encapsulado de contorno pequeño, más pequeño que los encapsulados DIP.

Se monta directamente sobre la superficie de una PCB (placa de circuito impreso).

Mejora la disipación térmica y reduce el espacio ocupado en la PCB.

Se usa ampliamente en aplicaciones donde el tamaño y la eficiencia son factores clave.



Figura 2.46 Circuito integrado con encapsulado SOP.

TSOP (Thin Small Outline Package) encapsulado pequeño y delgado, más delgado que el SOP estándar.

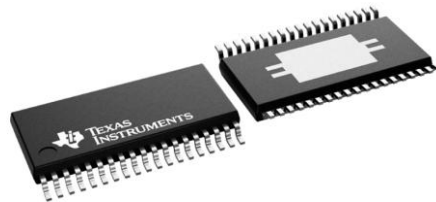


Figura 2.47 Circuito integrado con encapsulado TSOP.

QFP(Quad Flat Package) Paquete cuadrado plano.

Es la versión mejorada del encapsulado SOP donde los pines de conexión se extienden a lo largo de los cuatro bordes el circuito.

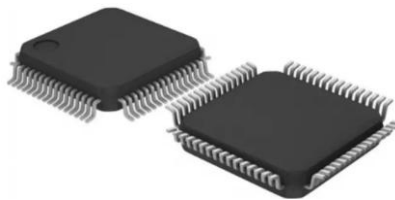


Figura 2.48 Circuito integrado con encapsulado QFP

El TSSOP (Thin Shrink Small Outline Package) es un tipo de encapsulado de montaje superficial (SMD, Surface-Mount Device) que se caracteriza por ser más delgado y compacto que un SSOP (Shrink Small Outline Package) estándar.



Figura 2.49 Circuito integrado con encapsulado TSSOP.

PLCC (Plastic Leaded Chip Carrier) es un tipo de encapsulado de montaje superficial SMD (Surface-Mount Device) y también puede usarse con zócalos especiales. Se caracteriza por tener pines en forma de "J" que se doblan hacia abajo y hacia adentro en los cuatro lados del encapsulado.

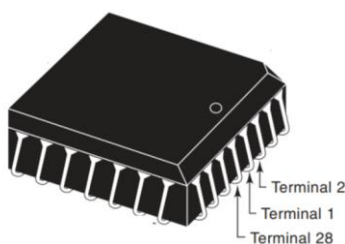


Figura 2.50 Circuito integrado con encapsulado PLCC.

La siguiente tabla presenta un resumen de los diferentes encapsulados de los circuitos integrado con sus dimensiones.

Abreviatura	Nombre del encapsulado	Altura	Distancia entre terminales
DIP	Encapsulado dual en línea	200 mils (5.1 mm)	100 mils (2.54 mm)
SOIC	Circuito integrado de pequeño contorno	2.65 mm	50 mils (1.27 mm)
SSOP	Encapsulado reducido de contorno pequeño	2.0 mm	0.65 mm
TSSOP	Encapsulado reducido delgado de contorno pequeño	1.1 mm	0.65 mm
TVSOP	Encapsulado delgado de contorno muy pequeño	1.2 mm	0.4 mm
PLCC	Portador de chip de plástico con terminales	4.5 mm	1.27 mm
QFP	Encapsulado plano de cuatro lados	4.5 mm	0.635 mm
TQFP	Encapsulado plano delgado de cuatro lados	1.6 mm	0.5 mm
LFBGA	Matriz de rejillas de bolas finas de bajo perfil	1.5 mm	0.8 mm

Tabla 2.1 Encapsulados de los circuitos integrados. [14]

Resumen de la familia TTL

TTL S, Schottky, series 74S, 54S.

Esta familia conecta un diodo Schottky entre la base y el colector (transistor Schottky) del transistor para reducir el tiempo de almacenamiento t_s , y conmutar más rápido de encendido a apagado lo que aumenta su velocidad de funcionamiento, la potencia que disipa sigue siendo alta.

TTL LS Schottky series 74LS 54LS.

Esta familia es de menor velocidad que la serie S pero disipa menor potencia por estar fabricada con transistores Schottky- clamped y resistencias de valor más grandes para disminuir la potencia.

TTL AS Schottky Avanzado series 74AS 54AS.

Esta familia es una versión mejorada de la LS, maneja mayor velocidad, menor disipación de potencia, mayor integración y maneja mayores márgenes de ruido.

TTL ALS Schottky avanzado de baja potencia, serie 74ALS 54ALS.

Es la versión mejorada de toda la serie TTL, mejora las características de la familia ALS.

TTL F rápida serie 74F 54F

Usa una serie de muy alta velocidad que se logra al disminuir el valor de las resistencias por lo que aumenta el consumo de potencia, útiles en aplicaciones donde se requiere altas velocidades y donde la disipación de potencia no es crítica.

Características típicas de las familias TTL						
Parámetros	74	74S	74LS	74AS	74ALS	74F
Tiempo de propagación (ns)	9	3	9.5	1.7	4	3
Disipación de potencia (mW)	10	20	2	8	1.2	6
Capacidad de carga (misma serie)	10	20	20	40	20	33
Frecuencia máxima de reloj (MHz)	35	125	45	200	70	100
$V_{OH}(\text{mín})$	2.4	2.7	2.7	2.5	2.5	2.5
$V_{OL}(\text{máx})$	0.4	0.5	0.5	0.5	0.5	0.5
$V_{IH}(\text{mín})$	2.0	2.0	2.0	2.0	2.0	2.0
$V_{IL}(\text{máx})$	0.8	0.8	0.8	0.8	0.8	0.8

Tabla 2.3 Características típicas de las familias TTL

Resumen de la familia CMOS

La tecnología CMOS (Complementary Metal-Oxide-Semiconductor) serie 74HC/HCT es ampliamente utilizada en electrónica digital, la C indica que trabaja con MOSFET canal N y MOSFET canal P, normalmente en la trayectoria de la corriente del circuito que se esté manejando hay un MOSFET encendido con una resistencia muy baja y un MOSFET con una resistencia de apagado de valor muy alto por lo que la corriente que circula en ellos es mucho muy pequeña.

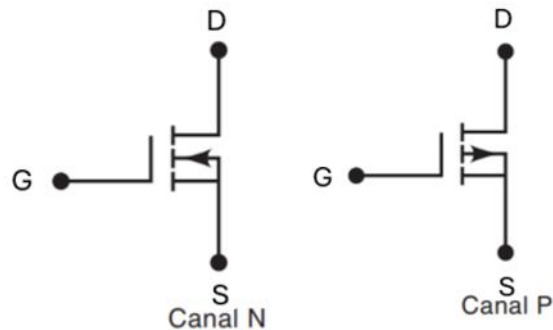


Figura 2.51 MOSFET canal N y P.

Ventajas de la familia CMOS

Muy bajo consumo de energía: Solo consume corriente durante la conmutación, ya que en la trayectoria de la alimentación siempre se encuentra un MOSFET encendido y otro apagado lo que implica la existencia de una resistencia de un elevado valor por lo que la corriente que fluye a través de ella es muy pequeña.

Con la tecnología CMOS se puede tener una alta densidad de integración: Lo que permite fabricar circuitos con millones de transistores en un solo chip.

Los circuitos CMOS cuentan con una Alta inmunidad al ruido, sus márgenes de ruido son mucho mayor que los circuitos TTL lo que los hace mejores para trabajar en ambientes altamente ruidosos.

La tecnología CMOS tiene un Amplio rango de voltajes de operación, puede funcionar con niveles de voltaje más bajos que la tecnología TTL, reduciendo el consumo.

Serie 4000/14000

4000 es la serie más antigua desarrollada por RCA, su equivalente al 14000 es desarrollada por motorola, ambas son muy lentas al compararlas con la familia TTL y otras familias CMOS, son de muy bajo consumo de energía y tienen un rango de operación de voltaje de 3 a 15 volts, normalmente ya no son utilizadas.

Serie 74HC/HCT (CMOS de alta velocidad)

Esta familia de circuitos integrados digitales CMOS son de alta velocidad, diseñados para ser compatibles con la tecnología TTL, sus principales características son: alta velocidad (74HC), la serie 74HCT es compatibles con la serie TTL y puede trabajar en forma combinada con la serie TTL tradicional, opera con voltajes que van de 2 a 6 volts, la serie 74HCT trabaja de 4.5v a 5.5v por lo que se puede acoplar a los niveles TTL, su consumo de energía es mucho menor que la serie TTL tradicional, su velocidad de operación va de 7 a 25 nanosegundos, tienen un margen de ruido grande, disipan muy poca potencia y cuenta con una gran variedad de compuertas, contadores multiplexores etc.

Serie 74AC/ACT (CMOS avanzado)

Esta familia pertenece a la tecnología CMOS avanzada de alta velocidad diseñada para mejorar el rendimiento de la serie 74HC/HCT, algunas de sus características son: La serie 74AC ofrece velocidades comparables a la lógica TTL avanzada, La serie 74ACT (Advanced CMOS compatible con TTL) está diseñada para ser compatible con los niveles de voltaje TTL estándar, niveles de entrada compatibles con TTL, por lo que permite su uso con dispositivos lógicos TTL sin necesidad de interfaces adicionales, la serie 74AC opera con voltajes de 2 a 6 volts, la serie 74ACT opera de 4.5 a 5.5 volts lo que asegura su compatibilidad con los circuitos lógicos TTL tradicionales sin necesidad de interfaces, la velocidad de operación va de 2 a 8 nanosegundos, disipa baja potencia, mayor inmunidad al ruido en comparación con la familia HC/HCT, lo que los hace adecuados para aplicaciones de alta velocidad, maneja mayores corrientes de entrada/salida en comparación con la serie HC, tiene una gran variedad de compuertas lógicas (AND, OR, NOT, NAND, NOR, XOR, XNOR), flip-flops, contadores, multiplexores, buffers, etc.

La enumeración de estos circuitos es diferente a la serie TTL, algunos ejemplos son los siguientes:

74AC11004 = 74HC04

74ACT11293 = 74HCT293

Serie AHC/AHCT (CMOS avanzado de alta velocidad)

La familia 74HC/AHCT de circuitos integrados son de la serie 74HC (High-Speed CMOS) y 74AHCT (Advanced High-Speed CMOS con niveles de entrada TTL). Son muy utilizados en electrónica digital en la implementación de funciones lógicas, registros, contadores, multiplexores etc. Las principales características que la familia HC tiene son: Operan con un rango de voltaje de alimentación de 2 a 6 volts, son compatible con niveles de entrada de CMOS estándar, disipan baja potencia, tienen alta inmunidad al ruido. Las características de la familia AHCT son: Son una variante avanzada de la familia HC, son compatible con niveles de entrada TTL, opera con un voltaje de alimentación de 4.5 a 5.5volts, trabajan a velocidades más altas que la serie 74HC, tienen una mayor capacidad de corriente de salida. La elección entre 74HC y 74AHCT depende del entorno en el que se usen; si se requiere compatibilidad con señales TTL, 74AHCT es la mejor opción, mientras que 74HC es ideal en sistemas CMOS puros.

Entradas no utilizadas en los circuitos CMOS

En los circuitos CMOS las entradas no conectadas se le conoce como entradas flotantes y deben evitarse ya que pueden causar un mal funcionamiento, algunos de los problemas que se pueden presentar son: Captar señales de ruido debido a su alta impedancia de entrada, puede quedar en un estado intermedio ente el 0 y el 1 lo que provocará un mal funcionamiento y los transistores internos pueden quedar en conducción simultáneamente aumentando el consumo de energía.

Para evitar lo anterior es importante conectar resistencias de pull -up o pull-down, mandar las entradas a VDD o VSS o interconectarlas con otras entradas que no afecten el funcionamiento.

Sensibilidad a la electroestática

Los circuitos integrados CMOS (Complementary Metal-Oxide-Semiconductor) tienen una impedancia extremadamente alta, las compuertas de los transistores MOSFET de los circuitos integrados están aisladas por una delgada capa de óxido de silicio (SiO_2), Este óxido es extremadamente frágil y puede perforarse con voltajes estáticos relativamente bajos. Sin protección adecuada, un simple toque de un dedo cargado con estática puede dañar el circuito, una persona puede no sentir la descarga hasta que supere los 3,000V, pero valores menores aún pueden dañar los componentes CMOS.

Muchos circuitos CMOS actuales incluyen diodos de protección en las entradas para disipar cargas estáticas, aun así, estos diodos tienen límites y no pueden soportar descargas muy grandes.

Una forma de protegerlos es tocando una superficie metálica conectada a tierra antes de manipular el circuito, usar una pulsera antiestática conectada a tierra al trabajar con componentes sensibles y evitar usar ropa sintética que pueda generar estática.

Las siguientes tablas muestran los parámetros de bajo voltaje, alta velocidad y entradas salidas de las distintas familias de circuitos lógicos.

Parámetro	LV	ALVC	AVC	ALVT	ALB
VCC	2.7–3.6	2.3–3.6	1.65–3.6	2.3–2.7	3–3.6
TPD (ns)	18	3	1.9	3.5	2
VIH (V)	De 2 a VCC 1 0.5	De 2 a 4.6	De 1.2 a 4.6	De 2 a 7	De 2.2 a 4.6
VIL (V)	0.8	0.8	0,7	0.8	0.6
IOH (mA)	6	12	8	32	25
IOL (mA)	6	12	8	32	25

Tabla 2.4 Características de la serie de bajo voltaje. [14]

Familia lógica	t_{pd} (ns)	$PD(mW)$ 100 kHz	Margen de ruido para el peor caso (mV)	Velocidad máxima de reloj (MHz)
74AS	1.7	8	300	200
74F	3.8	6	300	100
74AHC	3.7	0.006	550	130
74AVC	2	0.006	250	-
74ALVT	2.4	0.33	400	-
74ALB	2.2	1	400	-
ECL	0.3	25	150	1400

Tabla 2.5 Comparación de familias lógicas de alta velocidad. [14]

Parámetro	CMOS				TTL				
	4000B	74HC/H CT	74AC/ACT	74AHC/AHCT	74	74LS	74AS	74ALS	74F
$I_H(\text{máx})$	1 μA	1 μA	1 μA	1 μA	40 mA	20 μA	20 μA	20 μA	20 μA
$I_L(\text{máx})$	1 μA	1 μA	1 μA	1 μA	1.6 mA	0.4 mA	0.5 mA	100 mA	0.6 mA
$I_{OH}(\text{máx})$	0.4 mA	4 mA	24 mA	8 mA	0.4 mA	0.4 mA	2 mA	400 mA	1.0 mA
$I_{OL}(\text{máx})$	0.4 mA	4 mA	24 mA	8 mA	16 mA	8 mA	20 mA	8 mA	20 mA

Tabla 2.6 Corrientes de entrada/salida de las diferentes familias lógicas. [14]

2.3 Teoremas y postulados del algebra booleana.

El Álgebra de Booleana es un sistema matemático que fue desarrollado por George Boole en el siglo XIX, es utilizado para analizar y simplificar expresiones lógicas. Es una herramienta fundamental de la lógica digital y se utiliza en el diseño de circuitos electrónicos en la programación y en las bases de datos.

Postulados del algebra.

Básicamente se obtienen del comportamiento de las compuertas OR, AND y NOT, suele dársele un nombre, que en realidad lo importante es entender de donde viene cada uno de ellos.

Postulados de identidad.

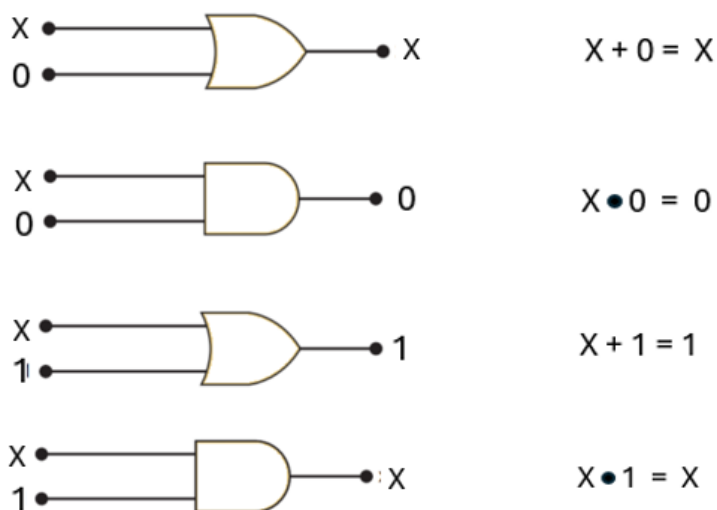


Figura 2.52 Postulados de identidad.

Postulados de idempotencia.

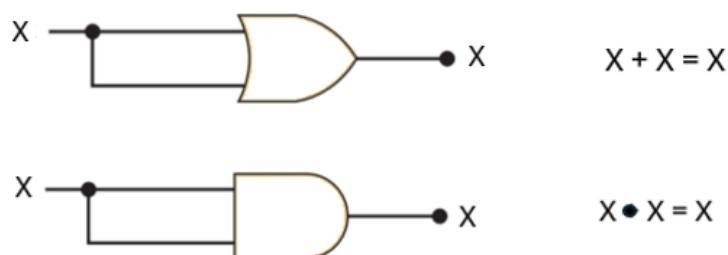


Figura 2.53 Postulados de idempotencia.

Postulados complementarios.

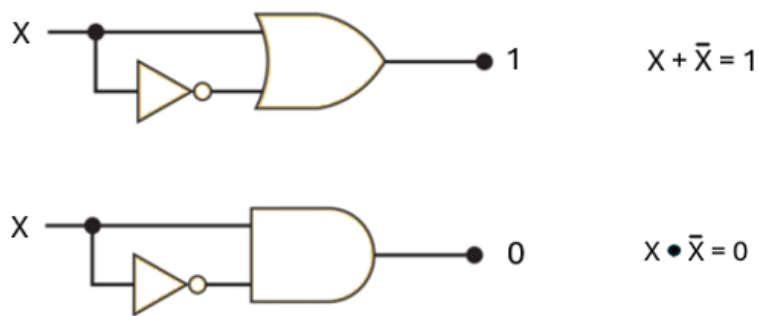


Figura 2.54 Postulados complementarios.

Postulados conmutativos.

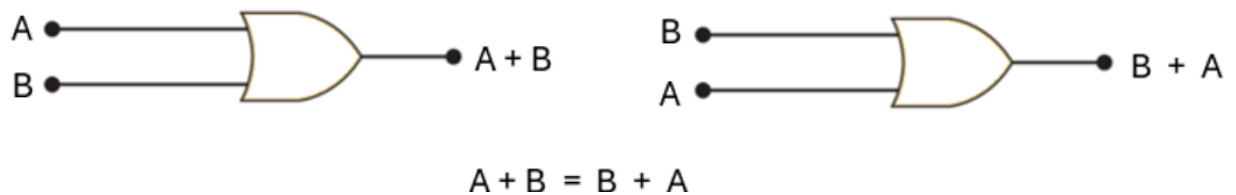


Figura 2.55 Postulados conmutativos.

Postulados asociativos.

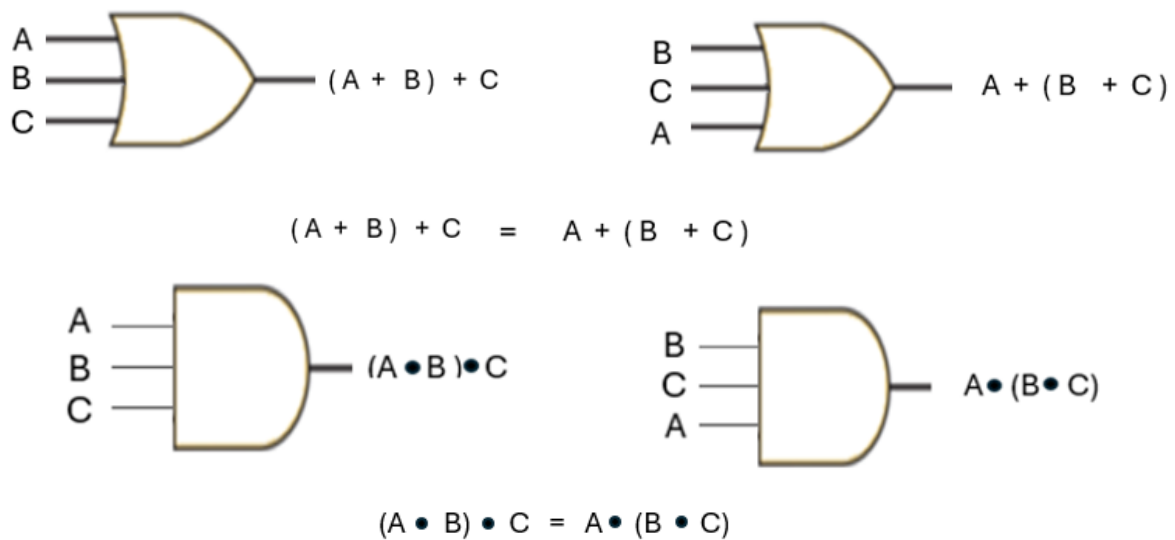
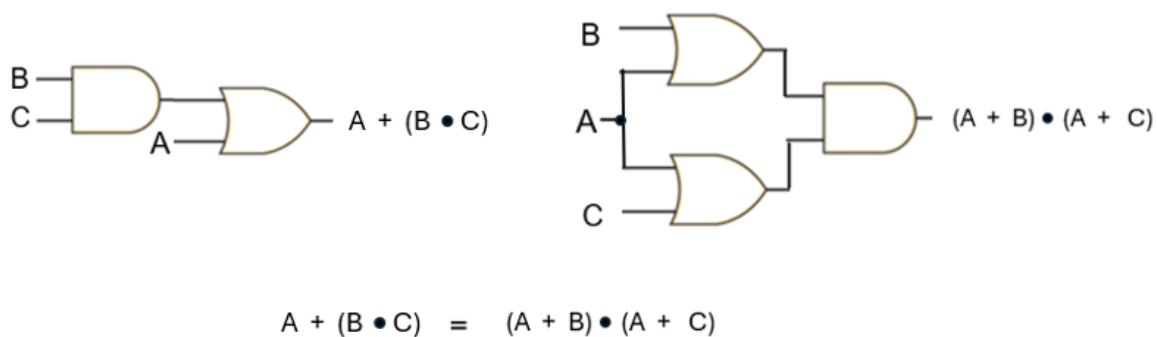
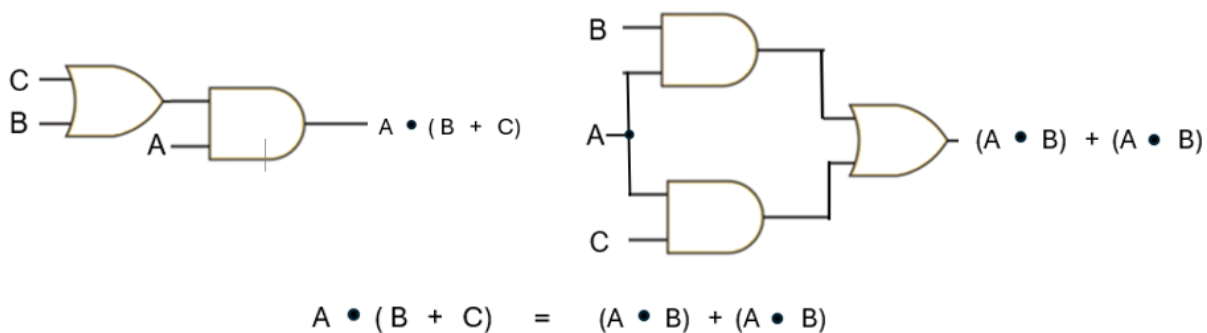


Figura 2.56 Postulados asociativos.

Postulados distributivos.



A	B	C	BC	A + (BC)
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

A	B	C	A + B	A + C	(A + B) (A + C)
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	1	1	1
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

Figura 2.57 Postulado distributivo y demostración por tablas de verdad .

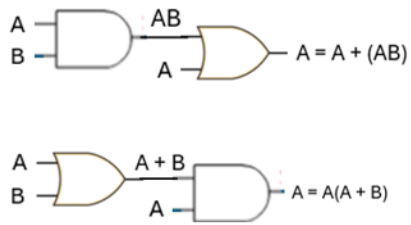
Teoremas del algebra.

A partir de los postulados se obtienen los teoremas del algebra y son los siguientes:

Ley de Absorción.

$$A + (A \bullet B) = A$$

$$A \bullet (A + B) = A$$

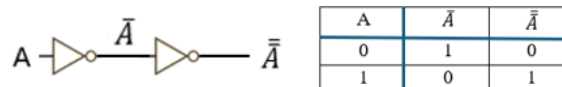


B	A	AB	A+B	A + (AB)	A(A+B)
0	0	0	0	0	0
0	1	0	1	1	1
1	0	0	1	0	0
1	1	1	1	1	1

Figura 2.58 Ley de Absorción circuitos u tabla de verdad.

Ley de Involución.

$$\bar{\bar{A}} = A$$



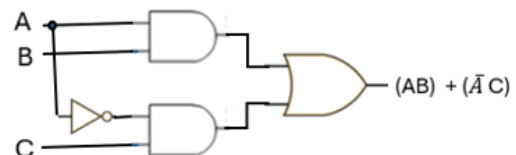
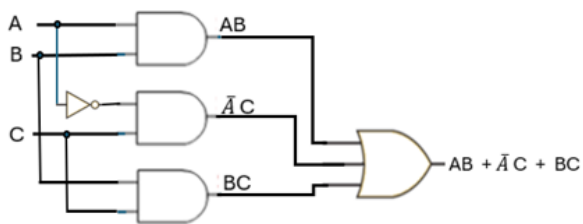
A	$\bar{A}$	$\bar{\bar{A}}$
0	1	0
1	0	1

$$A = \bar{\bar{A}}$$

Figura 2.59 Ley de la Involución circuito y tabla de verdad.

Teoremas de Consenso.

$$(AB) + (\bar{A}C) + (BC) = (AB) + (\bar{A}C)$$



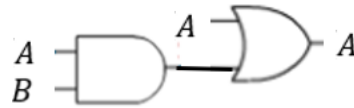
$$AB + \bar{A}C + BC = (AB) + (\bar{A}C)$$

A	B	C	$\bar{A}$	AB	$\bar{A}C$	BC	$(AB) + (\bar{A}C) + (BC)$	$(AB) + (\bar{A}C)$
0	0	0	1	0	0	0	0	0
0	0	1	1	0	1	0	1	1
0	1	0	1	0	0	0	0	0
0	1	1	1	0	1	1	1	1
1	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	1	0	0	1	0	0	1	1
1	1	1	0	1	0	1	1	1

Figura 2.60 Teorema de Consenso circuitos y tabla de verdad.

Teoremas para demostrar con tablas de verdad.

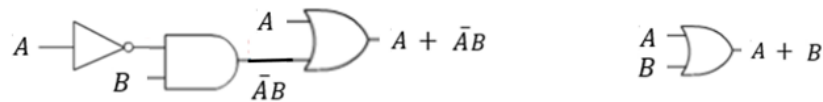
a) $A + AB = A$



A	B	AB	A + AB
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

Figura 2.61 Circuito y tabla de verdad del teorema $A + AB = A$.

b) $A + \bar{A}B = A + B$

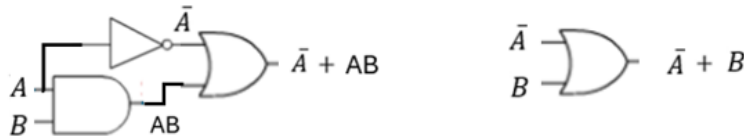


$$A + \bar{A}B = A + B$$

A	B	$\bar{A}$	$\bar{A}B$	$A + \bar{A}B$	$A + B$
0	0	1	0	0	0
0	1	1	1	1	1
1	0	0	0	1	1
1	1	0	0	1	1

c) Figura 2.62 Circuito y tabla de verdad del teorema $A + \bar{A}B = A + B$.

d) $\bar{A} + AB = \bar{A} + B$



$$\bar{A} + AB = \bar{A} + B$$

A	B	$\bar{A}$	AB	$\bar{A} + AB$	$\bar{A} + B$
0	0	1	0	1	1
0	1	1	0	1	1
1	0	0	0	0	0
1	1	0	1	1	1

a) Figura 2.63 Circuito y tabla de verdad del teorema $\bar{A} + AB = \bar{A} + B$.

Teoremas de Morgan.

Son dos teoremas muy importantes aportados por un matemático de apellido Morgan, estos teoremas nos permiten eliminar negaciones que abarcan más de una variable y poder aplicar las reglas de simplificación.

$$b) \overline{(x + y)} = \bar{x}\bar{y}$$

$$c) \overline{(xy)} = \bar{x} + \bar{y}$$

La representación de estos teoremas con compuertas queda de la siguiente manera:

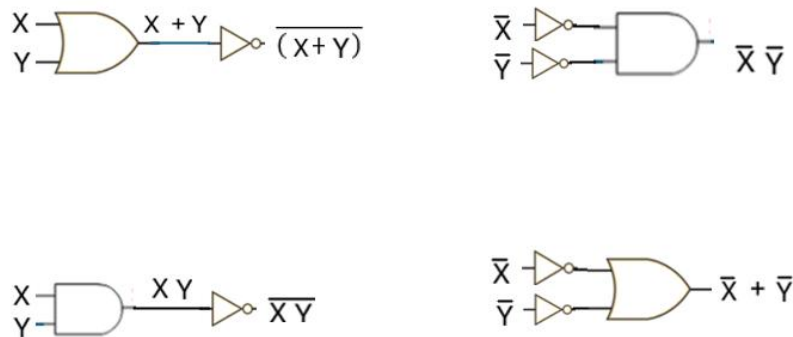


Figura 2.64 Circuitos para el teorema de Morgan.

Se puede aplicar el teorema de Morgan a cualquier término que tenga una negación y abarque más de una variable y se puede proceder de la siguiente manera:

- Negar las variables individuales que no estén negadas y quitar la negación a las variables individuales negadas (complementar las variables individuales).
- Cambiar las operaciones lógicas entre las variables, las OR por AND y las AND por OR.
- Si todo el término está negado, quitar la negación y si no está negado, negar todo el término (complementar todo el término).

Ejemplo

Aplice el teorema de Morgan al siguiente término:

$$\overline{(A\bar{B}\bar{C} + \bar{A})} = z$$

$$a) \overline{(\bar{A}BC + A)} = z \quad \text{Complementando las variables individuales}$$

$$b) \overline{(\bar{A} + B + C)A} = z \quad \text{Cambiando las operaciones OR por AND y viceversa.}$$

$$c) (\bar{A} + B + C)A = z \quad \text{Complementando todo el término.}$$

$$\overline{(A\bar{B}\bar{C} + \bar{A})} = z \quad = \quad (\bar{A} + B + C)A = z$$

Puede ser que una función lógica sólo tenga algún término con una negación que abarque más de una sólo variable, se le puede aplicar el teorema sólo a ese término.

Ejemplo

Aplique el teorema de Morgan al termino que tiene negación en más de una variable.

$$W = \overline{\bar{A}B + \bar{C}D} + A\bar{B}C + AB\bar{C}$$

Aplicando el teorema sólo al término.

$$\frac{\overline{\bar{A}B + \bar{C}D}}{\bar{A}B + \bar{C}D} = (A + \bar{B})(C + \bar{D})$$

Escribiendo la función lógica completa queda de la siguiente manera:

$$W = (A + \bar{B})(C + \bar{D}) + A\bar{B}C + AB\bar{C}$$

Ejemplo

Aplique el teorema de Morgan a la siguiente función booleana:

$$x = \overline{(A + \bar{B}C)} + \bar{B}\bar{C}$$

Aplicando el teorema al primer término queda:

$$x = \bar{A}(B + \bar{C}) + \bar{B}\bar{C}$$

Note cómo se cambió las compuertas AND por OR y la OR por AND, se complementó las variables individuales y luego el término completo.

Ahora aplicando al segundo término y queda:

$$x = \bar{A}(B + \bar{C}) + \bar{B} + C$$

Lo mismo que en el primer término se complementó las variables individuales, se cambió la operación AND por OR y se complementó el término.

$$x = \overline{(A + \bar{B}C)} + \bar{B}\bar{C} \quad = \quad x = \bar{A}(B + \bar{C}) + \bar{B} + C$$

2.4 Simplificación de funciones booleanas.

2.4.1.1 Simplificación por minitérminos.

Los minitérminos son una suma de productos(SOP) de las variables que se manejan en una fusión lógica, es decir compuertas AND de dos o más entradas según sea necesario y la salida de ella son conectadas a una compuerta OR de las entradas que se requieran.

Cuando en una función lógica todos los términos (compuertas AND) tienen todas las variables de entrada sin importar si están negadas o no, se dice que la función de minitérmino está en forma estándar o canónica SOPs y su notación formal es:

$$X(A,B,C) = \sum m(0,1,4,5,..)$$

Ejemplo de una función lógica en forma de minitérminos estándar:

$$F = \bar{A}\bar{B}CD + \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}CD + \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D$$

Ejemplo de una función lógica que no está en forma estándar o canónica.

$$W = \bar{P}\bar{Q} + \bar{P}R + PQR$$

En la función anterior a los dos primeros términos le faltan una variable, al primero R y segundo Q.

Para poner la función lógica en forma estándar, los términos a los que le haga falta una o más variables, se debe multiplicar por la OR de la variable negada y no negada que le hagan falta una variable a la vez.

Ejemplo

Poner en forma estándar la siguiente función lógica.

$$W = \bar{P}\bar{Q} + \bar{P}R + PQR$$

Al primer término le falta R se multiplica por $R + \bar{R}$ para no afectar la función ya que es equivalente a multiplicar por 1 el término, al segundo término le falta Q, se multiplica por $Q + \bar{Q}$.

$$W = \bar{P}\bar{Q}(R + \bar{R}) + \bar{P}R(Q + \bar{Q}) + PQR$$

Al desarrollar el producto queda de la siguiente manera:

$$W = \bar{P}\bar{Q}R + \bar{P}\bar{Q}\bar{R} + \bar{P}RQ + \bar{P}R\bar{Q} + PQR$$

Si algún término está repetido sólo se deja uno de ellos.

Nota

los minitérminos son los unos de salida de la tabla de verdad del circuito lógico.

Para simplificar una función lógica por minitérminos se aplica los postulado y teoremas del algebra booleana incluyendo los teoremas de Morgan.

Ejemplo

Simplificar la siguiente función lógica.

$$W = ABC\bar{D} + \bar{A}B\bar{C}D + \bar{A}\bar{B}C\bar{D} + A\bar{B}C\bar{D}$$

En los dos primeros términos se tiene en común $B\bar{C}D$ y en los dos últimos tenemos en común $\bar{B}C\bar{D}$ se pueden factorizar.

$$W = B\bar{C}D (A + \bar{A}) + \bar{B}C\bar{D}(\bar{A} + A)$$

Aplicando el teorema $A + \bar{A} = 1$

$$W = B\bar{C}D(1) + \bar{B}C\bar{D}(1)$$

Aplicando el teorema $A \bullet 1 = A$

$$W = B\bar{C}D + \bar{B}C\bar{D}$$

A este término ya no se le puede aplicar ningún teorema, por lo tanto, es el resultado.

Como se puede apreciar la función original tenía cuatro compuertas AND de cuatro entradas y una OR de cuatro entradas además los inversores de las variables individuales, en la función booleana simplificada solo tiene dos compuertas AND de tres entradas y un OR de dos entradas además los inversores de las variables individuales.

Ejemplo

Simplifique la siguiente función booleana.

$$x = (\bar{A} + B)(A + B + D)\bar{D}$$

Primero se desarrolla el producto de las variables que están entre paréntesis como lo hacemos en el algebra tradicional.

$$x = \bar{A}A\bar{D} + \bar{A}B\bar{D} + \bar{A}D\bar{D} + BA\bar{D} + BB\bar{D} + BD\bar{D}$$

Se tiene los siguientes postulados que podemos aplicar en los términos de la función:

$$X \bullet \bar{X} = 0$$

$$X \bullet X = X$$

$$x = 0\bar{D} + \bar{A}B\bar{D} + \bar{A}0 + B\bar{A}\bar{D} + \bar{B}\bar{D} + B0$$

$$x = 0 + \bar{A}B\bar{D} + 0 + B\bar{A}\bar{D} + \bar{B}\bar{D} + 0$$

Los términos que tienen 0 se eliminan de acuerdo con la regla:

$$X + 0 = X$$

$$x = \bar{A}B\bar{D} + B\bar{A}\bar{D} + \bar{B}\bar{D}$$

Todos los términos tienen las variables $\bar{B}\bar{D}$ se puede factorizar.

$$x = \bar{B}\bar{D}(\bar{A} + A + 1)$$

Aplicando los siguientes postulados:

$$X + 1 = 1$$

$$X \bullet 1 = X$$

$$x = (\bar{A} + B)(A + B + D)\bar{D} = \bar{B}\bar{D}$$

$$x = \bar{B}\bar{D}$$

Ejemplo

Simplificar la siguiente función lógica:

$$F = \bar{A}B\bar{C}D + AB\bar{C}D + ABCD + A\bar{B}CD$$

Agrupar los términos que tengan factores en común.

$$F = B\bar{C}D(\bar{A} + A) + ACD(B + \bar{B})$$

Se aplica el postulado:

$$X + \bar{X} = 1$$

$$F = B\bar{C}D + ACD$$

También se puede factorizar la variable D aunque ya no se reduce la función.

$$F = D(B\bar{C} + AC)$$

Ejemplo.

Simplifique la siguiente función lógica.

$$z = \overline{AC}(\overline{ABD}) + \overline{ABC} \overline{D} + \overline{ABC}$$

La función tiene un término a la que se debe aplicar el teorema de Morgan para poder simplificar, si se trabaja por separado queda de la siguiente manera:

$$(\overline{ABD}) = (A + \overline{B} + \overline{D})$$

Lo incluimos en la función.

$$z = \overline{AC}(A + \overline{B} + \overline{D}) + \overline{ABC} \overline{D} + \overline{ABC}$$

Ahora se desarrolla el producto para luego poder simplificar.

$$z = \overline{ACA} + \overline{ACB} + \overline{ACD} + \overline{ABC} \overline{D} + \overline{ABC}$$

Aplicando el teorema $X \bullet \overline{X} = 0$ al primer término.

$$z = \overline{A} \overline{B} C + \overline{ACD} + \overline{ABC} \overline{D} + \overline{ABC}$$

Factorizando los términos que tienen variables en común queda:

$$z = \overline{B} C (\overline{A} + A) + \overline{A} \overline{D} (C + \overline{B} C)$$

Al primer término se aplica los teoremas, $X + \overline{X} = 1$ y $X \bullet 1 = X$ y al segundo término el teorema, $A + \overline{A} B = A + B$.

$$z = \overline{B} C + \overline{A} \overline{D} (B + C)$$

Nota

Los minitérminos son los unos de salida en la tabla de verdad, y cada uno de ellos generan un término que es la AND de las variables de entradas, la función lógica será la OR de todos los minitérminos (suma de productos).

A	B	C	X	
0	0	0	0	
0	0	1	1	$\rightarrow \overline{A} \overline{B} C$
0	1	0	1	$\rightarrow \overline{A} B \overline{C}$
0	1	1	0	
1	0	0	0	
1	0	1	0	
1	1	0	1	$\rightarrow A B \overline{C}$
1	1	1	0	

$X = \sum m[\overline{A} \overline{B} C + \overline{A} B \overline{C} + A B \overline{C}]$
 $X(A,B,C) = \sum m[1,2,6]$

Figura 2.65 Tabla de verdad y sus minitérminos.

2.4.1.2 Simplificación por maxitérminos

Los maxitérminos también llamados producto de sumas(POS) es una manera de representar una función lógica y consiste en tener compuertas OR con las variables de entrada y la salida de estas conectadas a una compuerta AND de las entradas que sea necesario.

Cuando en una función lógica todos los términos(compuertas OR) tienen todas las variables de entrada sin importar si están negadas o no se dice que la función está en forma de productos de sumas estándar o canónica(POS) y su notación formal es:

$$X(A,B,C) = \prod M(0,1,3\dots)$$

Ejemplo

$$F = (A + B + \bar{C}) \cdot (A + \bar{B} + \bar{C}) \cdot (\bar{A} + B + \bar{C}) \cdot (\bar{A} + \bar{B} + C)$$

Si a una función lógica, uno o más términos (OR) no tiene todas las variables de entrada del circuito sin importar si están negadas o no, no está en forma estándar.

Ejemplo de una función lógica por maxitérminos no estándar:

$$F = (A + B) \cdot (A + C)$$

Para poner una función de maxitérminos en forma estándar, a cada término que le falte una variable, se cambia por dos términos con las variables que tenía más la variable que le falte, un término con la variable no negada y el otro con la variable negada.

Trabajando con el primer término queda de la siguiente manera:

$$F = (A + B + C)(A + B + \bar{C})(A + C)$$

Trabajando el segundo término queda de la siguiente manera:

$$F = (A + B + C)(A + B + \bar{C})(A + B + C)(A + \bar{B} + C)$$

En este caso dos términos son iguales sólo se deja uno de ellos y finalmente queda la siguiente función en forma de producto de sumas estándar(POS) que son los ceros de salida en la tabla de verdad, las variables de cada término están negadas con respecto a las variables de entrada.

$$F = (A + B + C)(A + B + \bar{C})(A + \bar{B} + C)$$

Para simplificar una función lógica presentada en maxitérmino, primero se desarrollan los productos y luego se aplica las reglas del algebra booleana, esto lo hace muy laborioso por lo que siempre se simplifica por minitérminos.

Ejemplo

Simplifique la siguiente función lógica:

$$F = (A + B + \bar{C}) \cdot (A + \bar{B} + C)$$

Desarrollando los productos de la función queda de la siguiente forma:

$$F = AA + A\bar{B} + AC + BA + B\bar{B} + BC + \bar{C}A + \bar{C}\bar{B} + \bar{C}C$$

Aplicando los siguientes postulados:

$$X \bullet \bar{X} = 0$$

$$X \bullet X = X$$

$$F = A + A\bar{B} + AC + BA + 0 + BC + \bar{C}A + \bar{C}\bar{B} + 0$$

Factorizamos los términos comunes:

$$F = A(1 + \bar{B} + C + B + \bar{C}) + BC + \bar{C}\bar{B}$$

Aplicando el siguiente postulado:

$$X \bullet 1 = X$$

$$F = A + BC + \bar{C}\bar{B}$$

Nota

Los maxitérminos son los ceros de salida en la tabla de verdad, y cada uno de ellos generan un término que es la OR de las variables de entradas, pero negadas, la función lógica será la AND de todos los términos(producto de sumas).

A	B	C	x	
0	0	0	0	← $A + B + C$
0	0	1	1	
0	1	0	1	
0	1	1	1	
1	0	0	0	← $\bar{A} + B + C$
1	0	1	1	
1	1	0	1	
1	1	1	1	

$X(A,B,C) = \prod M[(A+B+C)(\bar{A}+B+C)]$
 $X(A,B,C) = \prod M[\pi(0,4)]$

Figura 2.66 Tabla de verdad y sus maxitérminos.

2.4.2 Mapas de Karnaugh.

Los mapas de Karnaugh Fueron desarrollados por **Maurice Karnaugh** en 1953 y son una herramienta que permite simplificar una función lógica, por lo tanto, reducir el número de compuertas del circuito. Los mapas tienen 2^n celdas (todas las combinaciones de las variables de entrada), donde n representa el número de variables que maneja la función lógica. Estas celdas están ordenadas en renglones y columnas.

En estos mapas se enumeran tanto los renglones como las columnas en por medio del código Gray, es decir entre dos renglones o dos columnas consecutivos sólo cambia de estado una variable lo que permite la eliminación de esta.

En el mapa de Karnaugh se vacía la información de la tabla de verdad del circuito que se esté diseñando y se puede trabajar con los unos de la tabla (minitérminos) o con los ceros (maxitérminos).

Los mapas pueden ser 2 o más variables, aunque suele trabajarse mapas de hasta 6 variable, para un mayor número de variables es conveniente trabajar algún software para simplificar las funciones lógicas.

Celdas adyacentes

Dos celdas son adyacentes si entre ella cambia de estado sólo una variable, normalmente son en forma horizontal o vertical nunca en diagonal, las celdas adyacentes se pueden agrupar en número de potencias de 2, es decir una sola celda, grupo de dos celdas, grupo de cuatro celdas, grupo de ocho celdas etc.

Mapas de Karnaugh de 2 variables.

Como son dos variables se tienen cuatro combinaciones posibles, por lo tanto, el mapa debe ser de cuatro celdas.

A	B	X
0	0	1
0	1	1
1	0	0
1	1	0

A \ B	0	1
0	1	1
1	0	0

Figura 2.67 Tabla de verdad y su mapa de Karnaugh. Se puede ver que en el mapa de Karnaugh fue vaciada la información de la tabla de verdad, ahora se revisa las celdas que tengan un 1 y si son adyacentes se elimina la variable que cambia de estado, la celda 00 tiene 1 que pertenece a $\bar{A}\bar{B}$ y también la celda 01 que pertenece a $\bar{A}B$, se comparan estos dos términos y la variable que cambia de estado es A, por lo tanto, se elimina y X será iguala a la OR de todos los términos que arroje el mapa, en este caso sólo hay uno.

A \ B	0	1
0	1	1
1	0	0

$\bar{B}$

$X = \bar{B}$

Figura 2.68 Mapa de Karnaugh con un enlace de dos celdas y su resultado.

Reducir la función lógica de X por mapas de Karnaugh para la siguiente tabla de verdad:

B	A	X
0	0	1
0	1	0
1	0	1
1	1	0

B \ A	0	1
0	1	0
1	1	0

Figura 2.69 Mapa de Karnaugh con un enlace de dos celdas.

Las celdas 00 correspondientes a $\bar{B}\bar{A}$ y la 10 correspondiente a $B\bar{A}$ son adyacentes ya que sólo cambia de estado la variable B por lo que se elimina esta.

B \ A	0	1
0	1	0
1	1	0

$\bar{A}$
 $X = \bar{A}$

Figura 2.70 Mapa de Karnaugh con la función X simplificada.

Reducir la función lógica de X por mapas de Karnaugh para la siguiente tabla de verdad:

B	A	X
0	0	1
0	1	0
1	0	0
1	1	1

B \ A	0	1
0	1	0
1	0	1

Figura 2.71 Mapa de Karnaugh sin celdas adyacentes.

La celda correspondiente a 00 ($\bar{B}\bar{A}$) y la celda correspondiente a 11 (AB) que tienen 1 no son adyacentes debido a que cambian de estado las dos variables, por lo tanto, no se puede reducir la función lógica, la función lógica X es la OR de todos los términos que arroja la tabla.

B \ A	0	1
0	1	0
1	0	1

$\bar{A}\bar{B}$
 AB
 $X = \bar{B}\bar{A} + AB$

Figura 2.72 Mapa de Karnaugh sin celdas adyacentes.

Mapas de Karnaugh de 3 variables.

Estos mapas tienen 8 celdas ya que con tres variables se tienen ese número de combinaciones. El mapa de Karnaugh se puede dibujar de dos maneras diferentes:

$\begin{array}{c} \text{C} \\ \text{BA} \end{array}$	00	01	11	10
0				
1				

$\begin{array}{c} \text{CB} \\ \text{A} \end{array}$	0	1
00		
01		
11		
10		

Figura 2.73 Mapa de Karnaugh para 3 variables.

Simplificar por mapas de Karnaugh la función lógica de X de la siguiente tabla de verdad y demostrar que se puede usar cualquiera de los mapas anteriores.

C	B	A	X
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Figura 2.74 Tabla de verdad de 3 variables.

Vaciando la información de la tabla de verdad en los dos mapas teniendo cuidado de no cometer errores al poner los ceros y unos exactamente en la celda que les corresponde, se tiene lo siguiente:

C	B	A	X
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

$\begin{array}{c} \text{C} \\ \text{BA} \end{array}$	00	01	11	10
0	0	1	1	0
1	1	0	0	1

$\begin{array}{c} \text{CB} \\ \text{A} \end{array}$	0	1
00	0	1
01	0	1
11	1	0
10	1	0

Figura 2.75 Tabla de verdad y su vaciado en los mapas de Karnaugh.

Las celdas que tienen unos, 001 ($\bar{C}\bar{B}A$) y 011 ($\bar{C}BA$) son adyacentes ya que sólo cambia de estado la variable B y las celdas 100 ($C\bar{B}\bar{A}$) y la 110 ($CB\bar{A}$) son adyacentes ya que sólo cambia de estado la variable B. (los extremos de un mapa de Karnaugh son adyacentes).

C \ BA		00	01	11	10
0	0	0	1	1	0
1	1	1	0	0	1

CB \ A		0	1
00	0	0	1
01	0	0	1
11	1	1	0
10	1	1	0

Figura 2.76 Mapas con los enlaces de las celdas adyacentes.

Se elimina la variable de entrada que cambie de estado en cada término y al final se hace una operación OR de todos los términos y se iguala a la variable de salida.

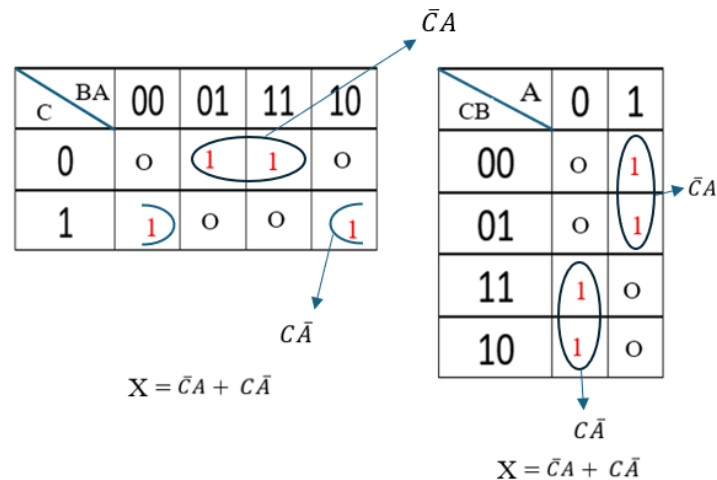


Figura 2.77 Mapas de Karnaugh con la variable de salida ya simplificada.

En un mapa de Karnaugh donde cuatro celdas son adyacentes se eliminan dos variables las que cambian de estado entre renglón y renglón, y de columna a columna, los que abarcan cuatro columnas consecutivas o las que abarcan cuatro renglones consecutivos. Siempre que una variable cambie de estado a lo largo del enlace ya sea en renglones o columnas se elimina independientemente del tamaño del enlace.

Para el primer enlace la celda 001 y 011 se elimina B y de las celdas 001 y 101 se elimina C y sólo queda el término con la variable A, en el segundo mapa las celdas 000 y 010 eliminan B y las celdas 000 y 100 eliminan C y sólo queda A.

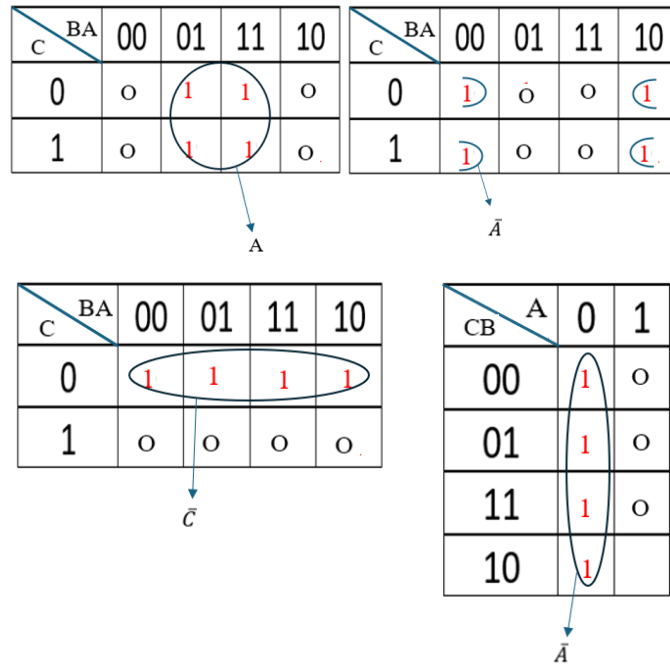


Figura 2.78 Mapas de Karnaugh con cuatro celdas adyacentes.

Mapas de Karnaugh de cuatro variables.

Las funciones lógicas de cuatro variables tienen 16 combinaciones diferentes posibles, por lo tanto, los mapas de Karnaugh serán de 16 celdas, recuerde que los renglones y las columnas se enumeran con el código Gray.

D	C	B	A	Z
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

DC \ BA	00	01	11	10
00				
01				
11				
10				

Figura 2.79 Tabla de verdad de cuatro variables y su mapa de Karnaugh.

Al vaciar la información de la tabla de verdad y formar los enlaces queda de la siguiente manera:

D	C	B	A	Z
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

DC \ BA	00	01	11	10
00	1	0	0	1
01	1	0	0	0
11	1	0	0	0
10	1	0	1	1

Diagram showing Karnaugh map groupings:

- Group 1 (Red oval): Cells (00,00), (01,00), (11,00), (10,00) → $\bar{C}\bar{A}$
- Group 2 (Red oval): Cells (00,00), (01,01), (01,11), (01,10) → $\bar{B}\bar{A}$
- Group 3 (Red oval): Cells (10,11), (10,10) → $D\bar{C}B$

$$Z = \bar{B}\bar{A} + \bar{C}\bar{A} + D\bar{C}B$$

Figura 2.80 Tabla y mapa con los enlaces y la función lógica para Z.

En la figura anterior se puede ver que tres de las esquinas forman parte de dos enlaces al mismo tiempo a estas celdas se les conoce como celdas pivote ya que nos permiten hacer dos o más enlaces. Siempre que se pueda formar un enlace con el mayor número de celdas debe hacerse para asegurar que la función lógica quedo simplificada al grado máximo, independientemente de que alguna celda ya haya sido usada para otro enlace. Si al resultado arrojado por un mapa de Karnaugh se le puede reducir por algebra booleana quiere decir que algún enlace pudo abarcar más celdas, el circuito seguirá funcionando, pero quedó con más compuertas.

Mapas de Karnaugh de 5 variables.

Los mapas de 5 variables tienen 32 celdas y se puede trazar un eje imaginario que divide el mapa en dos partes iguales, las celdas equidistantes a este eje son adyacentes.

En la siguiente figura el enlace de dos celdas la única variable que cambia de estado es C, en el enlace de cuatro celdas las variables que cambian de estado son C y D.

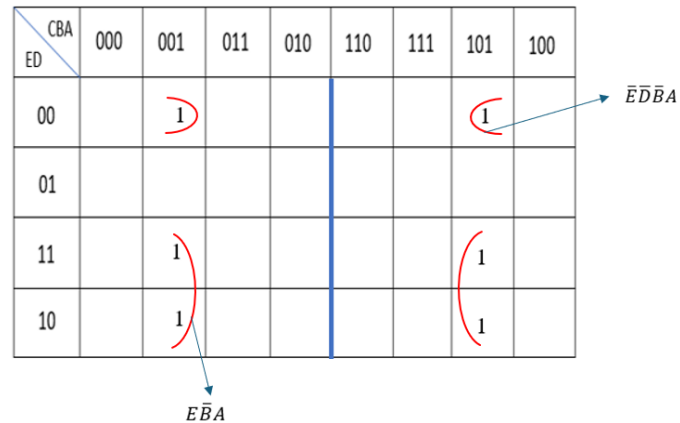


Figura 2.81 Mapa de Karnaugh de 5 variable con dos enlaces en celdas equidistantes.

Mapas de Karnaugh de 6 variables.

Los mapas de 6 variables tienen 64 celdas y se pueden trazar dos ejes imaginarios que divide el mapa en cuatro partes iguales, las celdas equidistantes a estos ejes son adyacentes. En el enlace de cuatro variables se elimina C y F por cambiar de estado, en el enlace de ocho celdas las variables que cambian de estado son C, D y F.

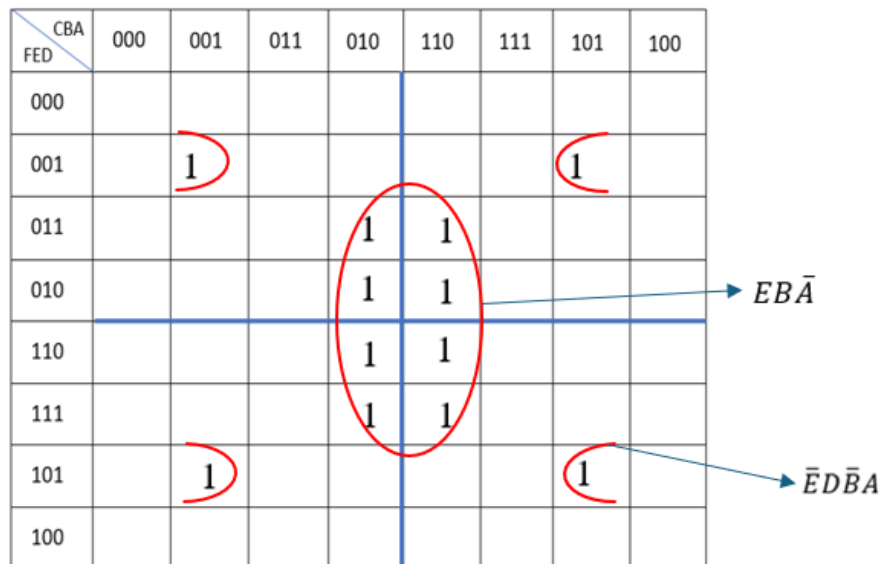


Figura 2.82 Mapa de Karnaugh de 6 variables con enlaces de 4 y 8 celdas adyacentes.

En un enlace de ocho celdas se eliminan tres variables, en un enlace de diez siséis celdas se eliminan cuatro variables, si un mapa tiene todas las celdas con unos, la función lógica siempre daría un uno de salida independientemente de las combinaciones de entrada. Matemáticamente, esto significa que la función es una **función tautológica**, es decir:

$$F(F,E,D...) = 1$$

Si un mapa tiene todas sus celdas con ceros, la salida siempre será cero independientemente de las combinaciones de las variables de entrada. Matemáticamente, esto significa que la función es una **función contradictoria** o **función nula**, es decir:

$$F(F,E,D...) = 0$$

Los términos DONT CARE (X , no importa) en las tablas de verdad indica que el resultado que arroje el circuito a la salida con una combinación de entrada específica no importa si da un 1 o si da un 0.

Estos términos se trabajan con el valor que más convenga a la hora de simplificar una función lógica.

Ejemplo

Simplifique la función lógica a de la siguiente tabla de verdad utilizando mapas de Karnaugh.

D	C	B	A	a
0	0	0	0	X
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	X
0	1	1	0	0
0	1	1	1	1
1	0	0	0	X
1	0	0	1	X
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	X

BA \ DC	00	01	11	10
00	X	0	0	1
01	1	X	1	0
11	0	1	X	1
10	X	X	0	1

Figura 2.83 Tabla de verdad y mapa de Karnaugh con términos no importa (X).

Se hacen los enlaces de las celdas adyacentes teniendo en cuenta que los términos no importan (X) se pueden tomar como 1 si conviene, es decir nos ayuda a simplificar más la función lógica de otra manera los tomamos como 0.

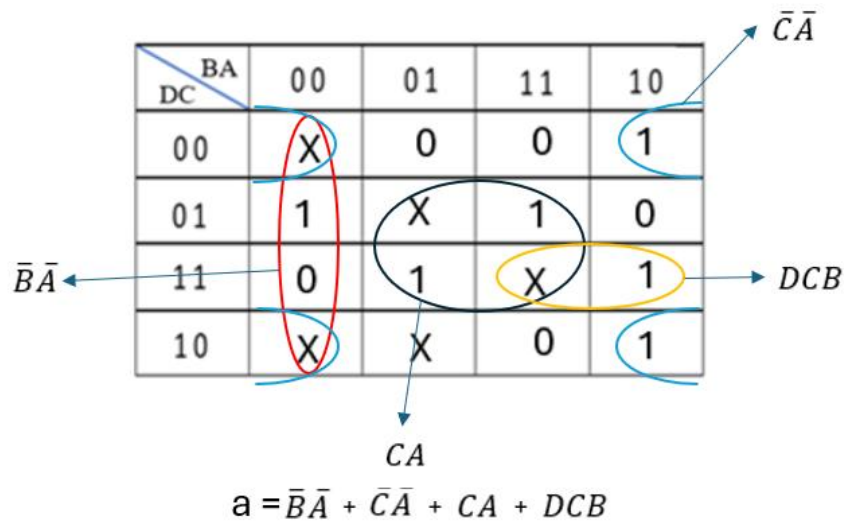


Figura 2.84 Mapa de Karnaugh utilizando los términos no importa (dont care) X.

Se puede ver que los términos X en las celdas 0000 y 1000 se tomaron como 1 para hacer un enlace de cuatro celdas lo mismo con los términos X de las celdas 0101 y 1111, el término X de la celda 1001 se tomó como 0, ya que no ayuda a simplificar la función.

2.4.3 Método de Quine-McClusKey

Este método de simplificación de funciones lógicas es un procedimiento sistemático parecido al método de mapas de Karnaugh que nos lleva a la máxima simplificación de la función lógica, pero es utilizado para funciones lógicas de más de 4 variables y funciona de la siguiente manera:

- Poner la función lógica a su manera estándar o canónica, (SOP) es decir suma de productos estándar (minitérminos).
- Agrupar los minitérminos de acuerdo con la cantidad de bits que estén en 1.
- Se comparan y se combinan los minitérminos que tengan de diferencia un solo bit, se reemplaza este bit por un guion.
- Se repite el procedimiento anterior hasta que ya no se puedan hacer más combinaciones.
- Se tiene que identificar los términos que no se cubren con otros a los que se les llama "términos primos esenciales"
- Hacer una tabla de términos primos aplicantes para seleccionar la combinación de términos primos esenciales que cubran toda la función lógica.

Ejemplo

Simplificar la función lógica siguiente:

$$F(A,B,C,D) = \sum m(4,8,10,11,12,15)$$

Se Ponen todos los minitérminos en forma binaria.

$$F = 0100 + 1000 + 1001 + 1010 + 1001 + 1100 + 1110 + 1111$$

Se hace una tabla agrupando los minitérminos que tengan un solo uno, los que tengan dos unos los que tengan tres unos y el de cuatro unos.

m4	0100	Término de un bit.
m8	1000	
m9	1001	Término de dos bits.
m10	1010	
m12	1100	
m11	1011	Términos de 3 bits.
m14	1110	
m15	111	

Tabla 2.85 grupos de minitérminos de 1, 2 y tres bits.

Se hace una tabla con los términos que sólo cambie un bit de estado entre ellos, quitamos el bit, lo sustituimos por un guion.

m(4,12)	_100
m(8,9)	100_
m (8,10)	10_0
m(8,12)	1_00
m(9,11)	10_1
m(10,11)	101_
m(10,14)	1_10
m (12,14)	11_0
m(11,15)	1_11
m (14,15)	111_

Tabla 2.86 Grupos de minitérminos que sólo cambian un bit.

Analizar los minitérminos que tengan en la misma posición el guion y si entre ellos sólo cambia un bit se combinan y se pone otro guion en el bit que cambie de estado entre ellos.

m(8,9,10,11)	10__
m (8,10,12,14)	1__0
m(10,11,14,15)	1_1_

Tabla 2.87 Grupos de minitérminos que tenían un guion en la misma posición y solo cambia de estado una variable.

Se hace una tabla con los minitérminos implicantes primos encontrados seguidos de los números de minitérminos de la función a simplificar, tachando la columna de los números

que se puedan formar al cambiar los guiones por cero y unos, los minitérminos con un guion tienen dos combinaciones, los que tienen dos guiones cuatro combinaciones.

Implicantes primos	4	8	10	11	12	15
$_100$	X				X	
$10_$		X	X	X		
$1--0$		X	X		X	
$1_1_$			X	X		X

Tabla 2.88 Implicantes primos encontrados y sus combinaciones en los minitérminos.

Los minitérminos que sólo tenga un X por columna son los implicantes primos esenciales, para este ejemplo son los minitérminos 4 y 15, el resultado es:

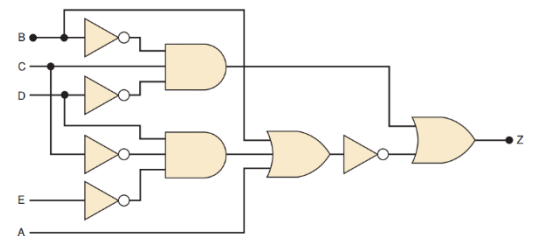
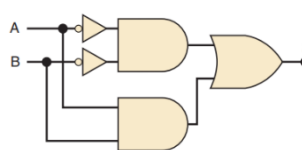
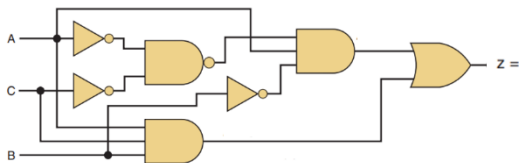
$$F = 0100 + 1111$$

$$F = \bar{A}\bar{B}\bar{C}\bar{D} + ABCD$$

Trabajo

Contestar las siguientes preguntas: [14]

1. Encontrar la función lógica de los siguientes circuitos:



2. Dibuje el circuito Lógico de las siguientes funciones:

$$x = \bar{A}BC + ABC + \bar{A}\bar{B}C + ABC + A\bar{B}\bar{C} + ABC$$

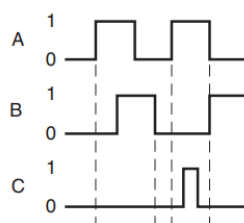
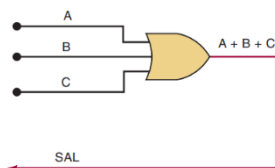
$$z = (\bar{A} + C) + (\bar{B} + \bar{D})$$

3. Simplificar las siguientes funciones lógicas por algebra Booleana.

$$S = \bar{P}\bar{Q}\bar{R} + \bar{P}\bar{Q}R + \bar{P}QR + \bar{P}QR + PQR$$

$$X = \bar{A}\bar{B}C + \bar{A}BC + ABC + A\bar{B}C$$

4. Encuentre la forma de onda de salida para el siguiente circuito.



5. Aplicar el teorema de DEMORGAN a las siguientes funciones:

$$z = \overline{A + \overline{B} \cdot C}$$

$$\omega = \overline{(A + BC) \cdot (D + EF)}$$

$$x = \overline{\overline{AB} \cdot \overline{CD} \cdot \overline{EF}}$$

6. Encuentre la tabla de verdad de **x** y **f** si el orden de las variables de entrada es ABCD.

$$x = \overline{A} + B + C + \overline{AD}$$

$$f = (C+D)(B'+D)(B'+C)(A'+C)$$

7. Encuentre la función lógica para f y X por minitérminos y por maxi términos, simplifíquelas con algebra booleana y compruebe los resultados con los que obtenga utilizando Mapas de Karnaugh para simplificarlos.

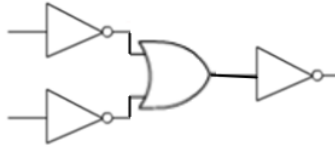
A	B	C	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

A	B	C	D	X
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

Examen

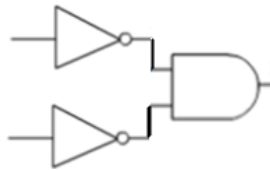
Lea cuidadosamente las instrucciones y seleccione la respuesta correcta.

1. El arreglo de compuertas básicas que se muestra en la figura es equivalente a una compuerta:



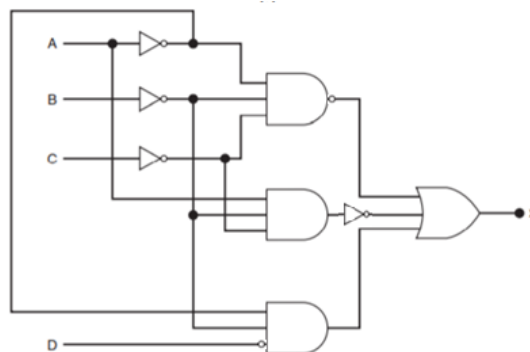
- a) NOR
- b) NAND
- c) AND
- d) OR

2. El arreglo que se muestra en la figura de compuertas básicas es equivalente a una compuerta:



- a) NAND
- b) OR
- c) AND
- d) NOR

3. La expresión booleana para X que describe el siguiente circuito es :

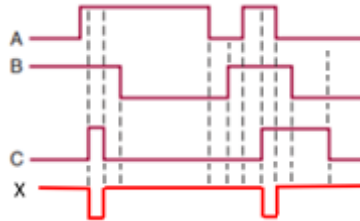


- a) $(A'B'C')' + (ABC') + A'B'D'$

b) $(A'B'C')' + A'B'D' + (AB'C')'$

c) $(A'B'D')' + A'B'C' + (A'BC')'$

4. Si A, B, y C son las entradas de una compuerta y X es la salida, ¿a qué compuerta corresponde la siguiente carta de tiempo?



a) NAND

b) OR

c) AND

d) NOR

5. Aplique el teorema De Morgan a la función lógica Z para eliminar todas las negaciones que abarquen más de una variable y seleccione la respuesta correcta.

$$Z = \overline{A(B + C)D}$$

a) $Z = A' + B' + CD'$

b) $Z = A' + BC + D'$

c) $Z = A' + B + C + D'$

6. Encuentre la expresión Booleana simplificada por álgebra booleana para S de la tabla de verdad de la figura y seleccione la respuesta correcta.

P	Q	R	S
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

a) $S = Q'R + QR + PQ'$

b) $S = P'Q + R$

c) $S = QR' + PQ' + Q'R'$

d) $S = PQ' + R'$

7. Ponga en forma estándar o canónica la expresión booleana, $X = AB' + C'$ y diga cuáles son los minitérminos de la tabla de verdad.

a) $X = m_1 + m_2 + m_3 + m_5 + m_7$

b) $X = m_0 + m_1 + m_4 + m_5 + m_7$

c) $X = m_1 + m_3 + m_4 + m_5 + m_7$

d) $X = m_0 + m_4 + m_5 + m_6 + m_7$

8. Ponga en forma estándar la expresión booleana, $X = C'(A'+B)$ y diga cuáles son los maxitérminos de la tabla de verdad.

a) $X = M_0, M_1, M_2, M_3, M_7$

b) $X = M_1, M_3, M_4, M_5, M_7$

c) $X = M_1, M_2, M_5, M_6, M_7$

d) $X = M_0, M_3, M_5, M_6, M_7$

9. Simplifique con mapas de Karnaugh usando los minitérminos de la siguiente tabla de verdad asegurándose de obtener la máxima simplificación y seleccione la respuesta correcta.

A	B	C	D	X
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

a) $X = A'D + BD + A'BD$

b) $X = A'C'D' + A'BD + ACD' + ABD$

c) $X = A'BD + AD' + B'D'$

d) $X = A'BD + AD' + BD' + AC'D'$

10. Simplifique con mapas de Karnaugh usando los maxitérminos de la siguiente tabla de verdad asegurándose de obtener la máxima simplificación y seleccione la respuesta correcta.

A	B	C	D	X
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

a) $X = (A + C)(B' + C')(B' + D')$

b) $X = (A' + C')(B + C)(B + D)$

c) $X = (A' + B' + D')(A + B')(A + B + C)$

11. Simplifique con mapas de Karnaugh usando los minitérminos de la tabla de verdad asegurándose de obtener la máxima simplificación utilizando correctamente los términos no importa (X) y seleccione la respuesta correcta.

A	B	C	D	Z
0	0	0	0	X
0	0	0	1	X
0	0	1	0	1
0	0	1	1	X
0	1	0	0	0
0	1	0	1	X
0	1	1	0	0
0	1	1	1	X
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	X
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	X

a) $Z = B'D + DC'B + AB'D'$

b) $Z = B'D' + BD$

c) $Z = BD + ABD + B'CD'$

d) $Z = B'D + BD'$

Práctica Compuertas Lógicas.

Objetivo

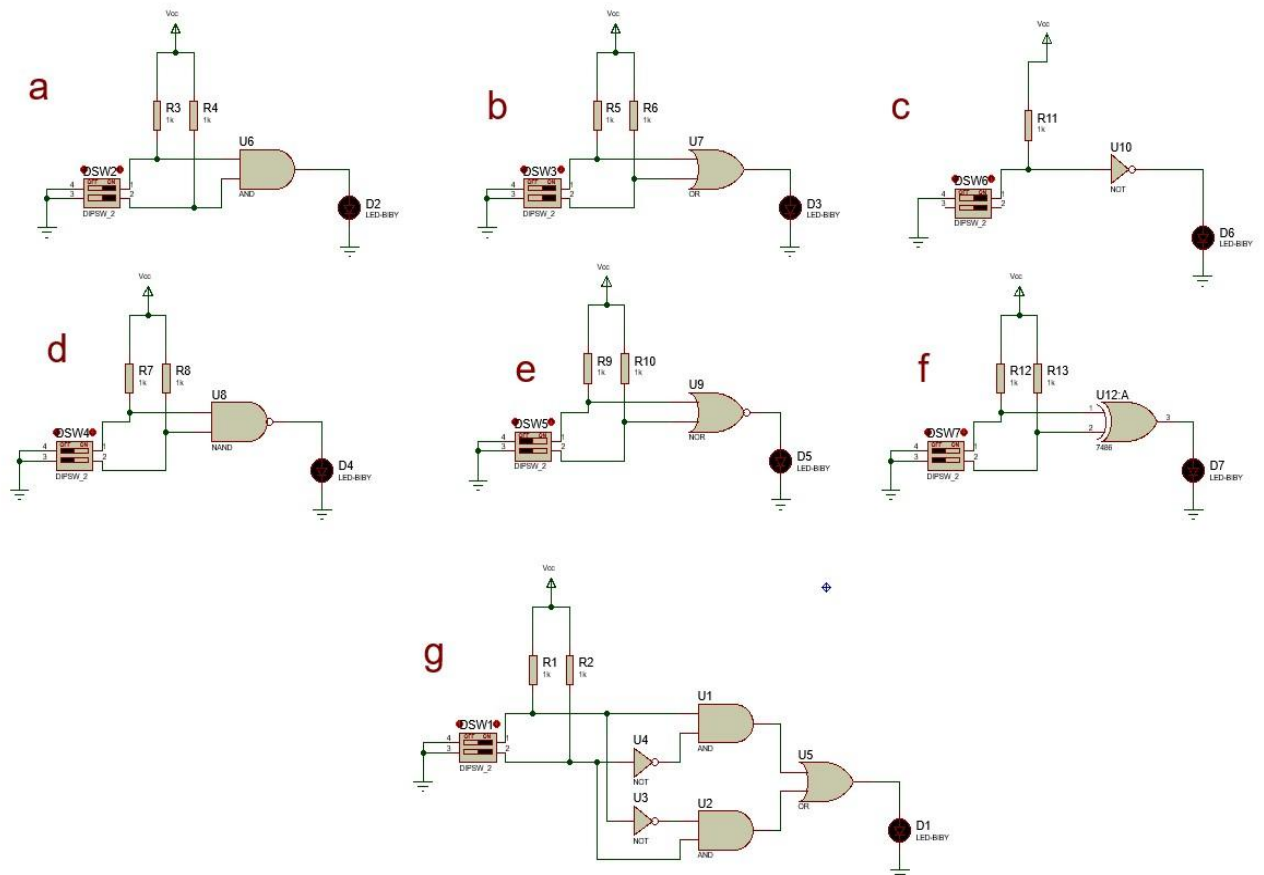
Comprobar la tabla de verdad de las compuertas básicas y compuestas.

Material

- 1 compuerta 74LS04
- 1 compuerta 74LS08
- 1 compuerta 74LS00
- 1 compuerta 74LS32
- 1 compuerta 74LS86

Desarrollo

Arme los siguientes circuitos, llene la tabla de verdad con los valores observados en las distintas combinaciones de entrada y compare los resultados con la tabla que da el fabricante. Simule los circuitos con algún software.

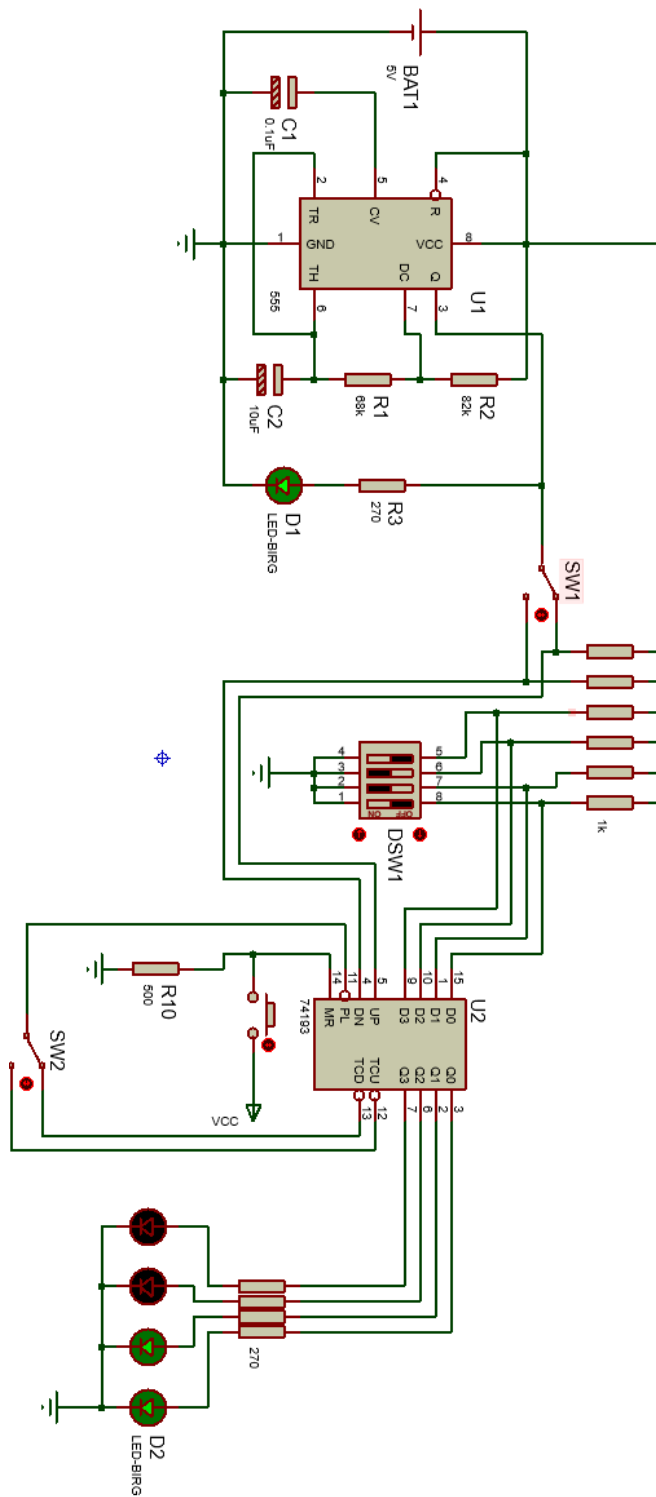


Práctica Contador Binario

Objetivo

Comprender el conteo binario y el manejo del LM555 como multivibrador Astable para señal de Reloj del contador binario 74LS193.

Arme el siguiente circuito y compruebe su funcionamiento como contador ascendente y descendente preinicializable, realice los cálculos para el LM555 de tal manera que tengamos una frecuencia de 1.2 Hertz. Simule el circuito con algún software



UNIDAD 3

Circuitos combinacionales.

Los circuitos lógicos combinacionales son un aparte de la electrónica digita y son aquellos que tienen variables de entrada y una o más variables de salida cuyo valor sólo dependen de la combinación de las entradas presentes sin importar los estados anteriores, no tienen memoria y sus salidas no forman parte de las variables de entrada, la respuesta en las salidas es inmediata al cambio de las variables de entrada, sólo depende del retardo inherente a las compuertas, estos circuitos combinacionales se diseñan con las compuertas AND, OR, NOT, XOR, NAND, NOR, XNOR.



Figura 3.1 Representación de un circuito lógico combinacional.

Algunos ejemplos de estos se muestran en la siguiente figura:

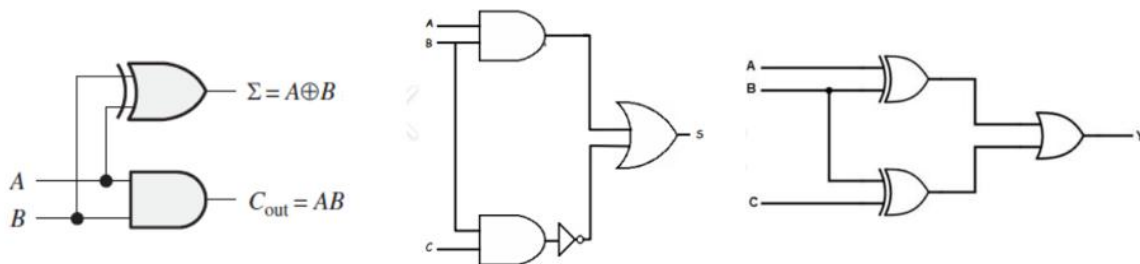


Figura 3.2 Circuitos lógicos combinacionales.

Muchos de los circuitos electrónicos integrados digitales son circuitos combinacionales, ejemplo de algunos de ellos son:

- Decodificadores
- Sumadores
- Comparadores
- Multiplexores
- Demultiplexores
- Codificadores

3.1 Procedimiento de diseño.

Para diseñar un circuito lógico se recomienda abarcar los siguientes pasos:

- a) Tener muy clara la idea de lo que requiere que haga el circuito, las variables de entrada y salida que manejará y tener en cuenta cuales serían las restricciones de este.
- b) Hacer la tabla de verdad con todas las combinaciones posibles de las variables de entrada y poner en la o las salidas cómo quiere que responda el circuito a estas combinaciones.
- c) Obtener la función lógica de la tabla de verdad, puede ser por minitérminos(POS) o maxitérminos(SOP) en ambos casos queda en forma canónica, utilizar alguna herramienta para simplificarla, puede ser por algebra booleana, mapas de Karnaugh o algún software para ello.
- d) Dibujar el circuito resultante de la función lógica simplificada, es recomendable evaluar la función para asegurar que cumple con la tabla de verdad.
- e) Simular el circuito con algún software como puede ser Proteus, Multisim, Logisim, SimulIDE etc.

3.2 Circuitos básicos combinacionales.

Ejemplo 1

Diseñar un circuito lógico combinacional de tres variables de entrada C,B y A con una variable de salida Z, el circuito deberá dar un 1 de salida cada que la combinación de entrada tenga número de unos impar y cero en cualquier otro caso.

Primer paso, se escribe la tabla de verdad para que cumpla el requisito pedido queda de la siguiente manera:

C	B	A	Z
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Figura 3.3 tabla de verdad del ejemplo 1.

Para este diseño se trabajará con los unos de la tabla de verdad es decir los minitérminos, por lo que cada 1 de la tabla de verdad genera un término equivalente al producto de las variables de entrada.

C	B	A	Z	
0	0	0	0	
0	0	1	1	$\rightarrow \bar{C}\bar{B}A$
0	1	0	1	$\rightarrow \bar{C}B\bar{A}$
0	1	1	0	
1	0	0	1	$\rightarrow C\bar{B}\bar{A}$
1	0	1	0	
1	1	0	0	
1	1	1	1	$\rightarrow CBA$

3.4 Tabla de verdad con los minitérminos del diseño.

La variable de salida Z será igual a la OR de todos los minitérminos (AND) de la tabla, quedará en forma de suma de productos estándar (SOP), la notación formal es la siguiente:

$$Z(C,B,A) = \sum m(\bar{C}\bar{B}A + \bar{C}B\bar{A} + C\bar{B}\bar{A} + CBA)$$

$$Z(C,B,A) = \sum m(1,2,4,7)$$

El siguiente paso es simplificar la función lógica Z, puede ser por algebra booleana o mapas de Karnaugh.

$\begin{array}{c} \text{BA} \\ \diagdown \\ C \end{array}$	00	01	11	10	
0		1	1		
1	1		1		$\rightarrow BA$

$\downarrow$ $C\bar{B}\bar{A}$ $\downarrow$ $C\bar{A}$

Figura 3.5 Mapa de Karnaugh con los enlaces resultantes.

La función lógica simplificada para Z es:

$$Z = BA + C\bar{A} + C\bar{B}\bar{A}$$

Al implementar el circuito queda de la siguiente manera:

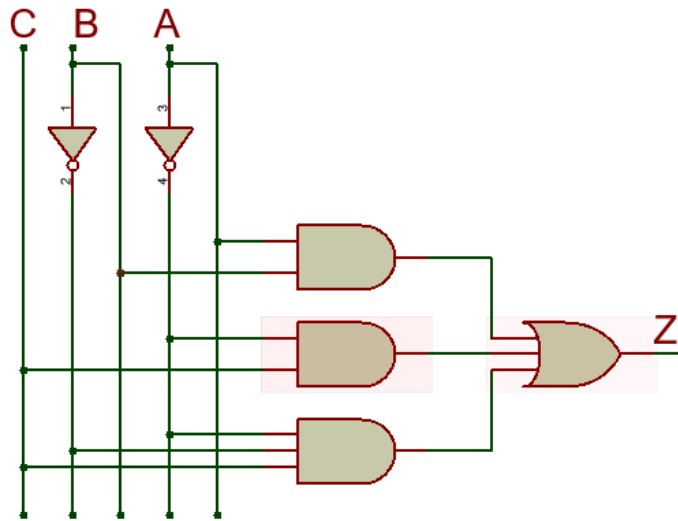


Figura 3.6 Circuito de la función lógica Z.

Ejemplo 2

Diseñar un circuito lógico combinacional para hacer el control de giro de un motor a pasos unipolar, las variables de entrada para seleccionar la bobina que se activará son Q1 y Q0 y la variable de entrada que controlará el sentido del giro del motor es G.

Si la variable G es un 0 las bobinas se activarán en la secuencia A, B, C y D con las combinaciones de entrada(Q1 y Q0), 00, 01, 10 y 11 respectivamente, si G es 1 la secuencia de la activación de las bobinas será D, C, B y A con las combinaciones de entrada(Q1 y Q0), 11, 10, 01, y 00 respectivamente.

Se escribe la tabla de verdad que cumpla con los requisitos pedidos.

G	Q1	Q0	A	B	C	D
0	0	0	1	0	0	0
0	0	1	0	1	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	1
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

Figura 3.7 Tabla de verdad para el control de giro del motor a pasos.

Para el diseño de este control se requieren cuatro salidas, cada una de ellas controlará una de las bobinas del motor, por lo que se obtendrá la función lógica de cada una.

Para la bobina A, la función que se genera en la tabla es la siguiente:

G	Q1	Q0	A
0	0	0	1
0	0	1	0
0	1	0	0
0	0	0	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

→ $\bar{G}\bar{Q}_1\bar{Q}_0$

→ GQ_1Q_0

$$A = \bar{G}\bar{Q}_1\bar{Q}_0 + GQ_1Q_0$$

Figura 3.8 Tabla de verdad y función lógica para la bobina A.

Se puede observar que la función lógica A no puede ser simplificada ni por algebra booleana ni por mapas de Karnaugh, esto mismo parará con las funciones lógicas de B, y C.

Para la bobina B la función se genera con la siguiente tabla.

G	Q1	Q0	B
0	0	0	0
0	0	1	1
0	1	0	0
0	0	0	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

→ $\bar{G}\bar{Q}_1Q_0$

→ $GQ_1\bar{Q}_0$

$$B = \bar{G}\bar{Q}_1Q_0 + GQ_1\bar{Q}_0$$

Figura 3.9 Tabla de verdad y función lógica para la bobina B.

Para la bobina C la función se genera con la siguiente tabla.

G	Q1	Q0	C
0	0	0	0
0	0	1	0
0	1	0	1
0	0	0	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

$\rightarrow \bar{G}Q_1\bar{Q}_0$
 $\rightarrow G\bar{Q}_1Q_0$

$$C = \bar{G}Q_1\bar{Q}_0 + G\bar{Q}_1Q_0$$

Figura 3.10 Tabla de verdad y función lógica para la bobina C.

Para la bobina D la función se genera con la siguiente tabla.

G	Q1	Q0	D
0	0	0	0
0	0	1	0
0	1	0	0
0	0	0	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

$\rightarrow \bar{G}\bar{Q}_1\bar{Q}_0$
 $\rightarrow G\bar{Q}_1\bar{Q}_0$

$$D = \bar{G}\bar{Q}_1\bar{Q}_0 + G\bar{Q}_1\bar{Q}_0$$

$$D = \bar{Q}_1\bar{Q}_0(G + \bar{G})$$

$$D = \bar{Q}_1\bar{Q}_0$$

Figura 3.11 Tabla de verdad y función lógica para la bobina D.

A la función lógica para D se puede aplicar algebra booleana a los dos minitérminos y queda simplificada a un solo término como se muestra en la figura 3.11.

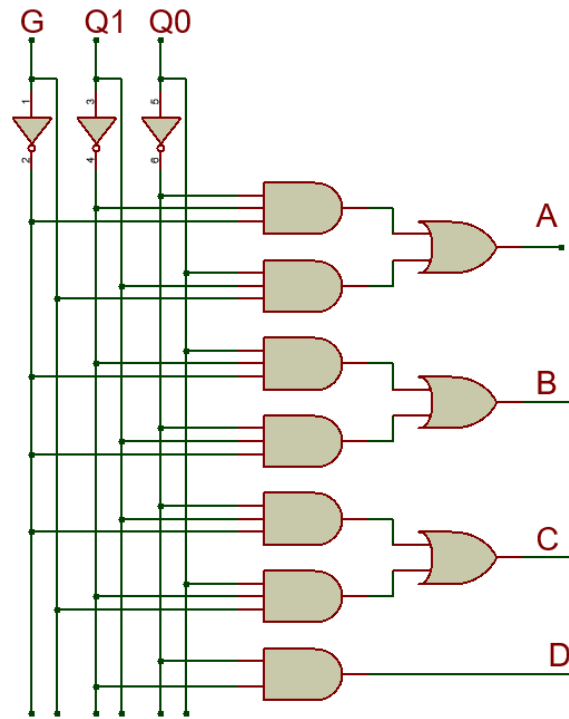


Figura 3.12 Circuito para el control del motor a pasos.

Ejemplo 3

Diseñar un control para un semáforo de dos fases de tal manera que los tiempos de cada una de ellas queden iguales. Las variables de entrada son cuatro Q3, Q2, Q1 y Q0, las variables de salida de la fase 1 son V1, R1, A1 y para la fase dos son V2, R2 y A2, la tabla de verdad que debe cumplir es la siguiente:

A	B	C	D	V1	R1	A1	V2	R2	A2
0	0	0	0	1	0	0	0	1	0
0	0	0	1	1	0	0	0	1	0
0	0	1	0	1	0	0	0	1	0
0	0	1	1	1	0	0	0	1	0
0	1	0	0	1	0	0	0	1	0
0	1	0	1	1	0	0	0	1	0
0	1	1	0	0	0	1	0	1	0
0	1	1	1	0	0	1	0	1	0
1	0	0	0	0	1	0	1	0	0
1	0	0	1	0	1	0	1	0	0
1	0	1	0	0	1	0	1	0	0
1	0	1	1	0	1	0	1	0	0
1	1	0	0	0	1	0	1	0	0
1	1	0	1	0	1	0	1	0	0
1	1	1	0	0	1	0	0	0	1
1	1	1	1	0	1	0	0	0	1

Figura 3.13 Tabla de verdad para el control del semáforo.

Para la salida del V1 la tabla de verdad y la simplificación por mapas de Karnaugh queda de la siguiente manera:

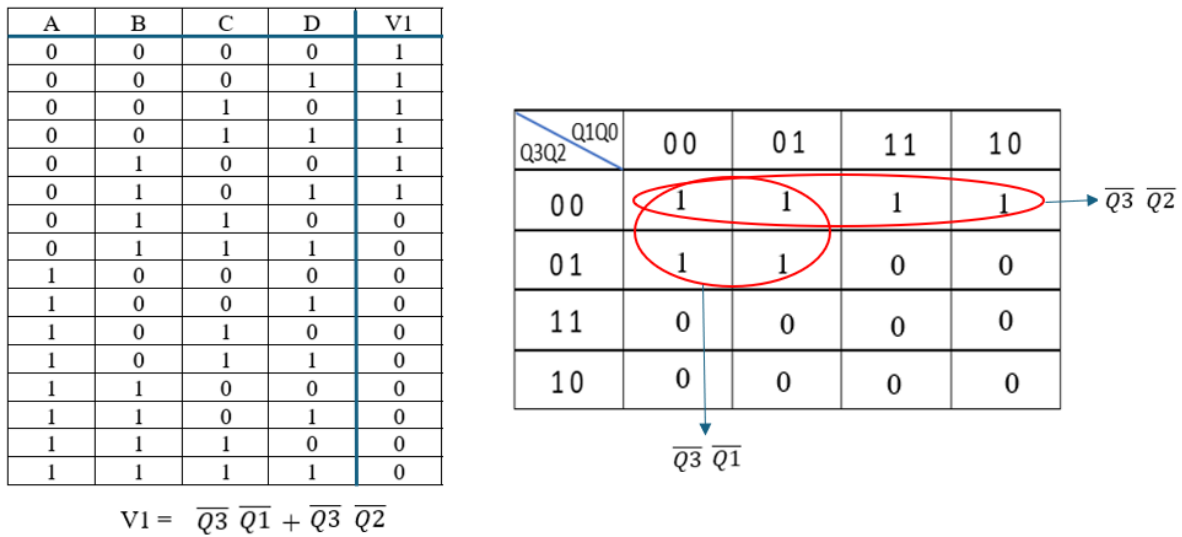


Figura 3.14 Tabla de verdad y mapa de Karnaugh para V1.

La variable de salida para R1 ya simplificada queda de la siguiente manera:

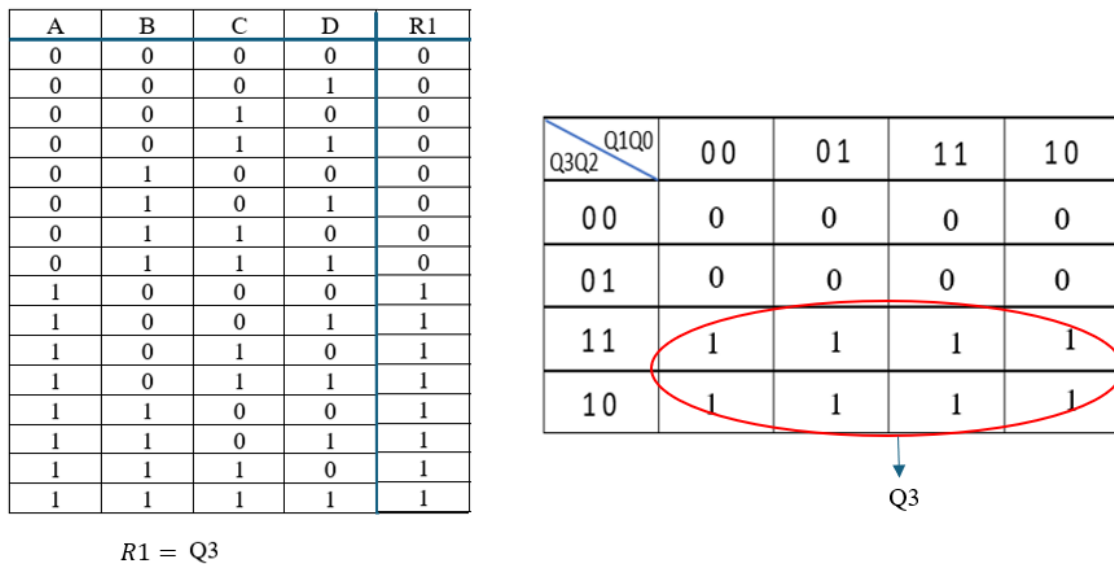


Figura 3.15 Tabla de verdad y mapa de Karnaugh para R1.

Para la salida del A1:

A	B	C	D	A1
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

$$A1 = \overline{Q3} Q2 Q1$$

$Q3Q2 \backslash Q1Q0$	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	0	0	0	0
10	0	0	0	0

$\overline{Q3} Q2 Q1$

Figura 3.16 Tabla de verdad y mapa de Karnaugh para A1.

Para la variable V2:

A	B	C	D	V2
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

$$V2 = Q3 \overline{Q1} + Q3 \overline{Q2}$$

$Q3Q2 \backslash Q1Q0$	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	0	0
10	1	1	1	1

$Q3 \overline{Q1}$ $Q3 \overline{Q2}$

Figura 3.17 Tabla de verdad y mapa de Karnaugh para V2.

Para la variable R2:

A	B	C	D	R2
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

$$R2 = \overline{Q3}$$

Q3Q2 \ Q1Q0	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	0	0	0	0
10	0	0	0	0

$\overline{Q3}$

Figura 3.18 Tabla de verdad y mapa de Karnaugh para R2.

Para la variable A2:

A	B	C	D	A2
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

$$A2 = Q3Q2Q1$$

Q3Q2 \ Q1Q0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	1	1
10	0	0	0	0

$Q3Q2Q1$

Figura 3.19 Tabla de verdad y mapa de Karnaugh para A2.

Una vez obtenidas las funciones lógicas de salida se procede a armar el circuito lógico para cada una de ellas.

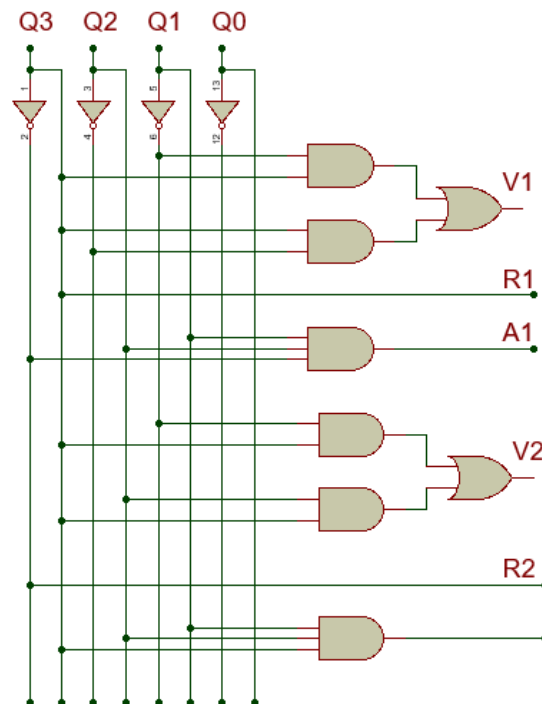


Figura 3.20 Circuito lógico para el semáforo.

3.3 Simulación de los circuitos combinacionales.

La simulación de los circuitos combinacionales se puede llevar a cabo con diferentes herramientas de software, algunas libres en línea como: Logisim Evolution, Logicly, CircuitVerse, Tinkercad Circuits etc., software profesional como: Proteus y Multisim o software libre como: Logisim y SimulIDE.

La simulación se hará con el software SimulIDE por ser una herramienta de código libre.

Lo primero que se debe hacer es descargar el software de acuerdo con el sistema operativo del equipo en el que se instalará de la página oficial [Downloads – SimulIDE](#), una vez descargado se descomprime y se ejecuta el programa, no requiere de una instalación formal.

Interfaz del usuario.

La interfaz para el usuario consta de tres ventanas que son:

Ventana izquierda.

En ella se encuentran los componentes, equipos de medición, fuentes etc. que maneja el simulador, sólo se tiene que seleccionar el que se requiera y arrastrarlo a la ventana de

trabajo, también se encuentra el explorador de archivo para guardar los proyectos realizados.

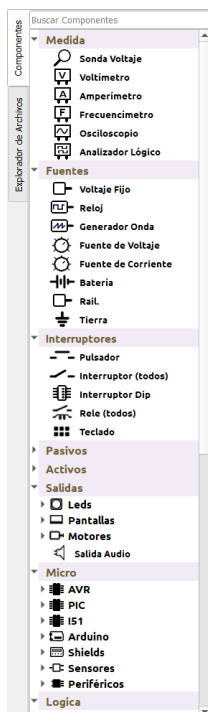


Figura 3.21 Ventana izquierda del SimulIDE.

Ventana central.

Esta ventana es el área de trabajo, en ella se construyen los circuitos y también se simulan, en la parte superior cuenta con las herramientas para ello y en la parte inferior con un área para mostrar los errores y mensaje del circuito.

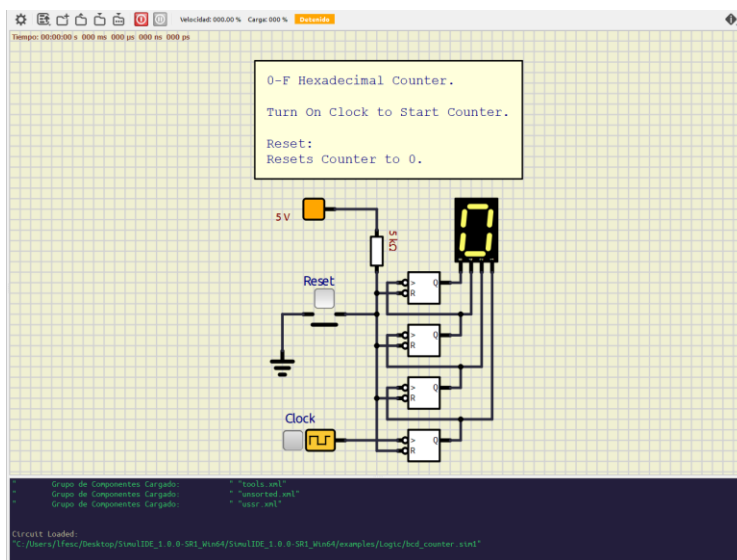


Figura 3.22 Ventana central del SimulIDE.

Ventana derecha.

Es un editor de texto básico de edición y depuración, útil para escribir y cargar código para algunos componentes que puede soportar SimulIDE.

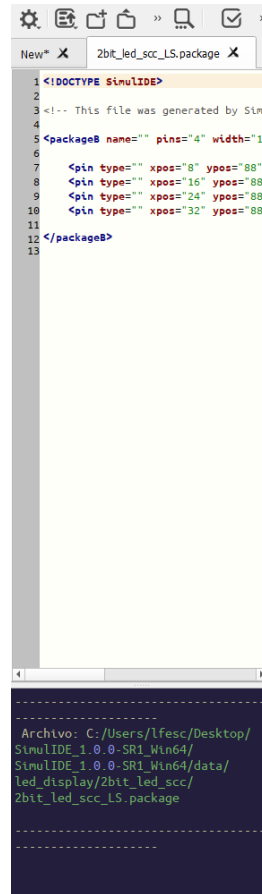


Figura 3.23 Ventana izquierda del SimulIDE.

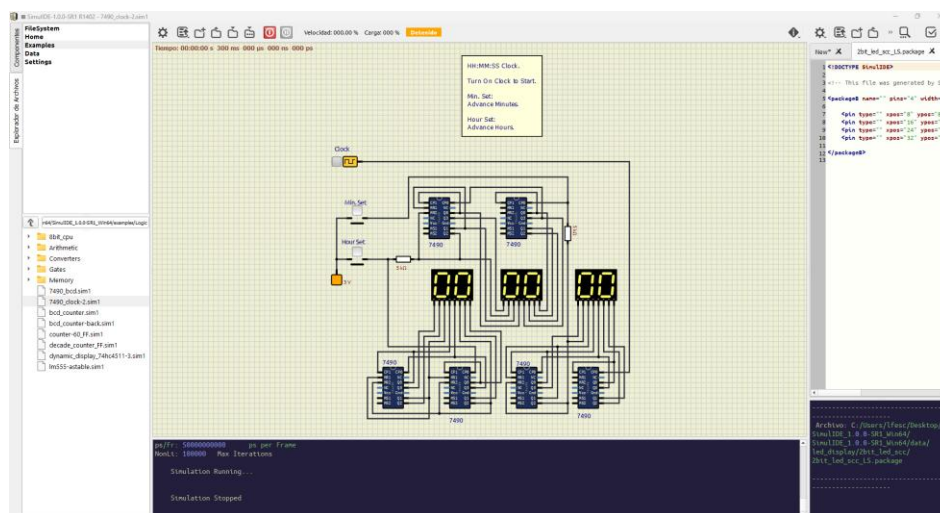


Figura 3.24 Ventana completa del SimulIDE.

Crear y simular un circuito.

Para crear un circuito y simularlo se pueden llevar a cabo los pasos siguientes:

- a) Seleccionar de la ventana izquierda los elementos necesarios que formarán el circuito, arrastrarlos a la ventana central, también se pueden elegir los equipos que se ocupen como osciloscopio, frecuencímetro, fuentes etc.
- b) Realizar las conexiones de los diferentes componentes del circuito haciendo clic en el inicio de la conexión y volviendo a dar clic al final de la conexión.
- c) Configurar las propiedades de los elementos del circuito dando un doble clic en cada uno de ellos, aparecerá una ventana con los parámetros que se pueden cambiar.
- d) Iniciar la simulación, ya terminado el circuito y configurado sus elementos se da clic en el botón de encendido que está en la herramienta de simulación en la ventana superior, se puede poner en pausa en cualquier momento o hacer ajustes al circuito hasta que el funcionamiento sea el deseado.

Para auxiliarse en el manejo más avanzado del simulador se puede acceder a los siguientes links:

<https://www.polimetro.com/tutorial-como-usar-simulide-simular-circuitos-logica-digital/>

<https://www.polimetro.com/tutorial-simulide-simulacion-circuitos-sensores-actuadores/>

<https://www.polimetro.com/simulacion-circuitos-potencia-simulide-tutorial-detallado/>

3.3.1 Multiplexores.

El multiplexor es un circuito lógico combinacional que tiene dos o más líneas de entrada, pero sólo tiene una salida, cuenta con líneas de selección para elegir cuál de las entradas se enviará a la salida, también puede tener una o más líneas de control que habilitan la salida. Un multiplexor tiene 2^n líneas de entrada donde n es el número de líneas de selección de entrada, si se tiene 3 líneas de selección entonces el multiplexor tiene 8 líneas entradas.

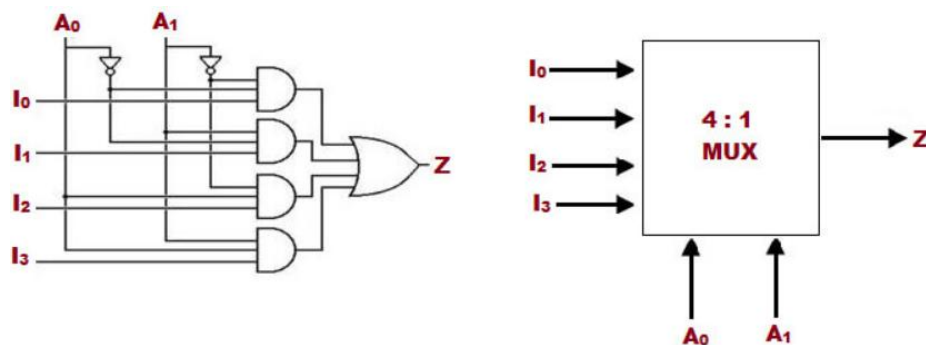


Figura 3.25 Multiplexor de cuatro entradas y una salida. [15]

Este tipo de circuitos son muy utilizados donde se requiere mandar varias señales a través de sólo dos líneas el común del circuito (tierra) y la señal.

Un ejemplo de este tipo de circuito es el multiplexor es el circuito 74LS151 tiene 8 líneas de entrada, una salida con su complemento y una línea de habilitación.

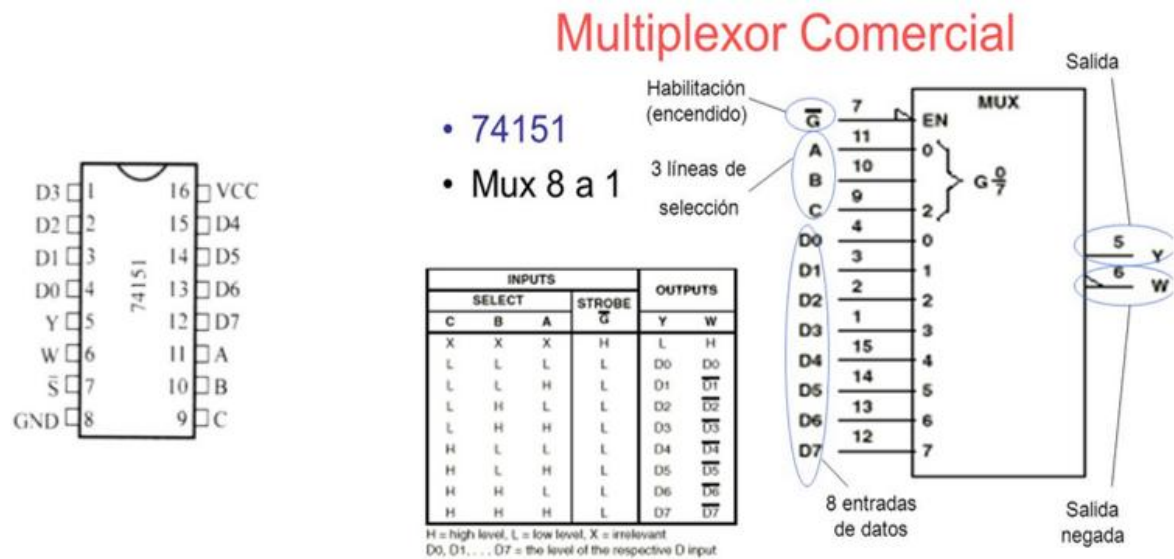


Figura 3.26 Multiplexor 74LS151 8 entradas y una salida.

Ejemplo

Simule el circuito multiplexor 74151 de tal manera que en un LED se refleje el valor de la entrada seleccionada cuando esté habilitado el circuito.

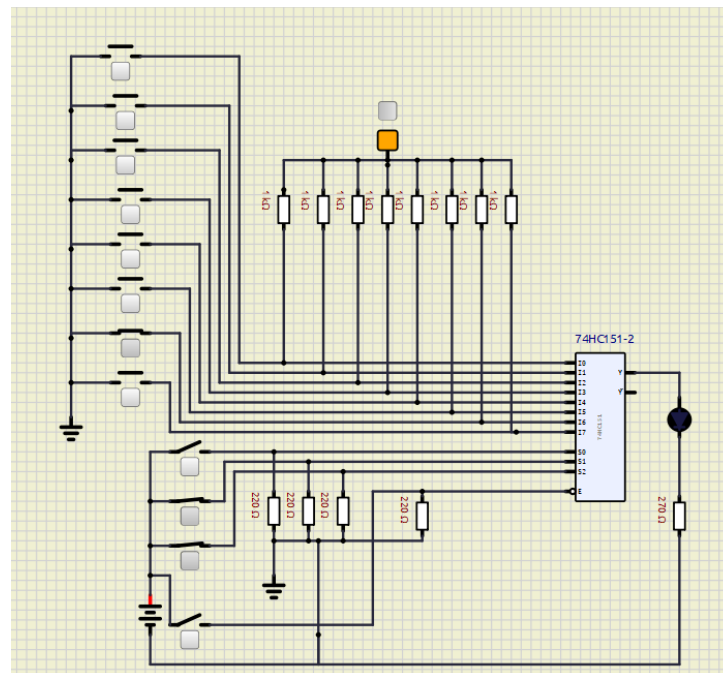


Figura 3.27 Simulación del multiplexor 74151 en SimulIDE.

Otro demultiplexor en circuito integrado es el 74LS150 de 16 entradas una salida negada y un habilitador.

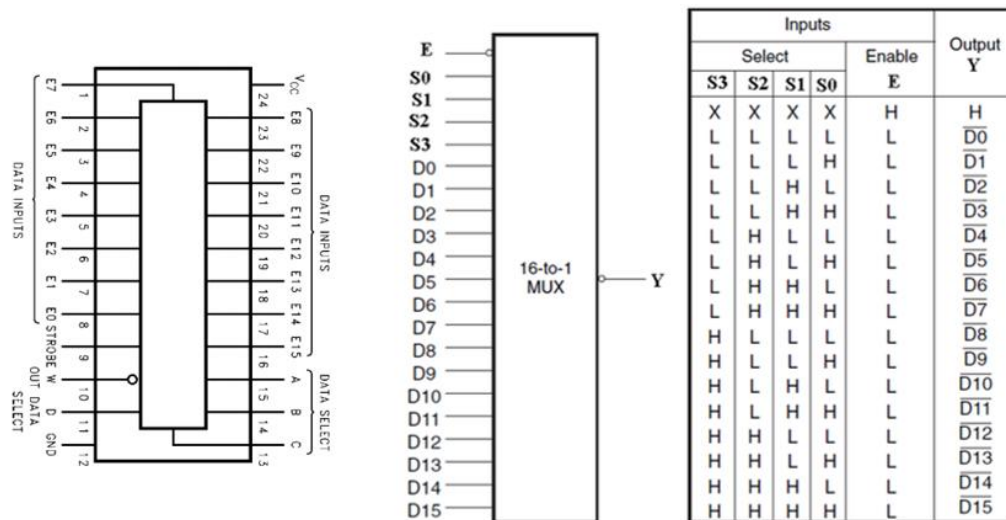


Figura 3.28 Multiplexor 74LS150 16 entradas y una salida.

3.3.2 Demultiplexores.

Un demultiplexor (decodificadores) es un circuito lógico combinacional que trabaja de manera inversa al multiplexor, es decir tiene una sola entrada y 2^n salidas donde n es el número de líneas de selección, si tiene 3 líneas de selección entonces tendrá 8 líneas de salida, puede tener uno o más habilitadores. Este tipo de circuitos son muy utilizados cuando una señal de entrada debe ser enviada a diferentes dispositivos conectados a la salida. La única diferencia entre el decodificador y el multiplexor es que el decodificador no tiene la línea de entrada.

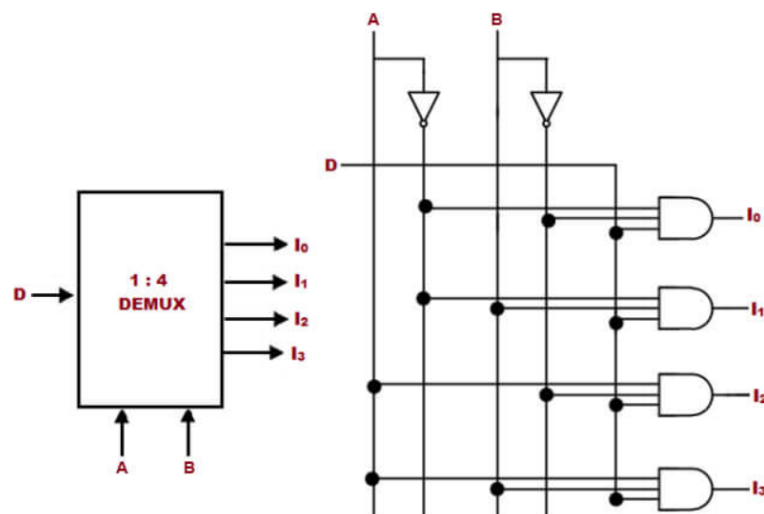


Figura 3.29 Demultiplexor de una entrada y cuatro salidas. [15]

Se puede manejar como demultiplexor el circuito 74LS138 que en realidad es un decodificador, se utiliza una de las líneas de control como dato de entrada.

Como se puede ver en la figura 3.29 la tabla de verdad nos indica que se tienen que cumplir las condiciones de los tres habilitadores ($\overline{E}_1$, $\overline{E}_2$ y E_3) para que la salida seleccionada por A_0 , A_1 y A_2 esté activa de otra forma todas las salidas estarán a nivel alto.

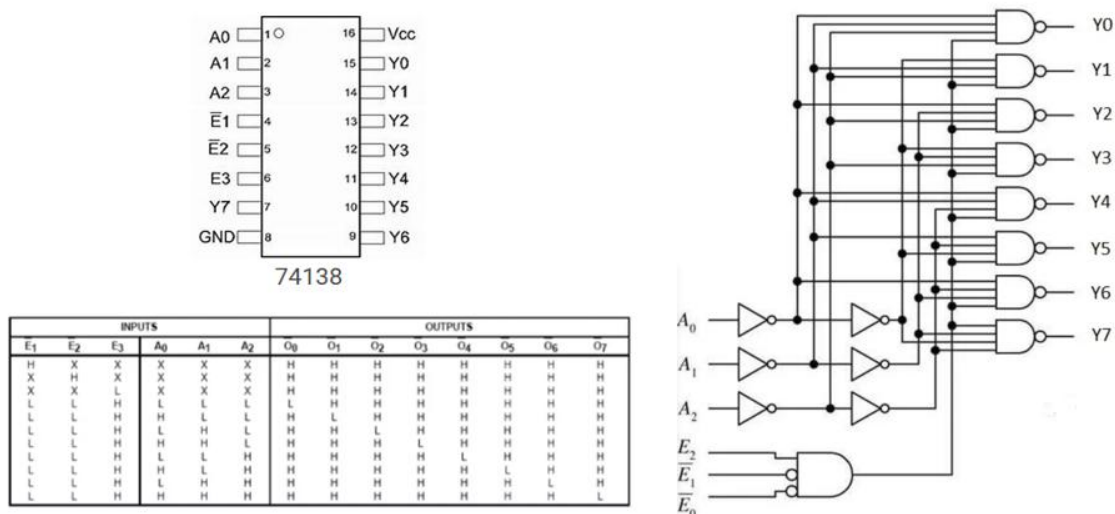


Figura 3.30 Demultiplexor/Decodificador 74138, tabla de verdad y su circuito lógico.

Para trabajar el decodificador como un demultiplexor se utilizar uno de los habilitadores como dato de entrada, se pueden dar diferentes combinaciones dependiendo de cómo se quiera manejar el dato.

Ejemplo

Se requiere mandar una señal de 500 Hz a la salida seleccionada del demultiplexor.

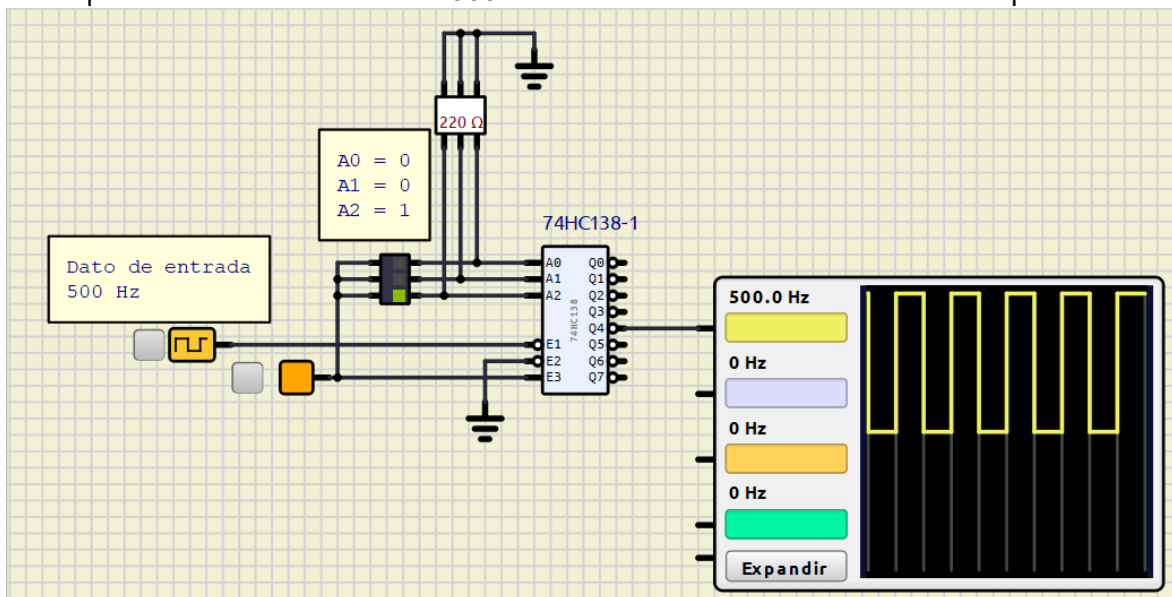
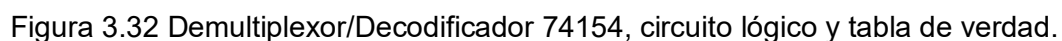


Figura 3.31 Decodificador 74138 trabajando como demultiplexor en SimuliIDE.

El decodificador/demultiplexor 74154 es de 16 líneas de salida y se puede manejar el habilitador como dato de entrada para que cumpla la función de demultiplexor.



Ejemplo

Diseñe un circuito para manejar el 74154 como demultiplexor y poder mandar una señal de 400Hz a la salida seleccionada por interruptores conectados al selector de entrada.

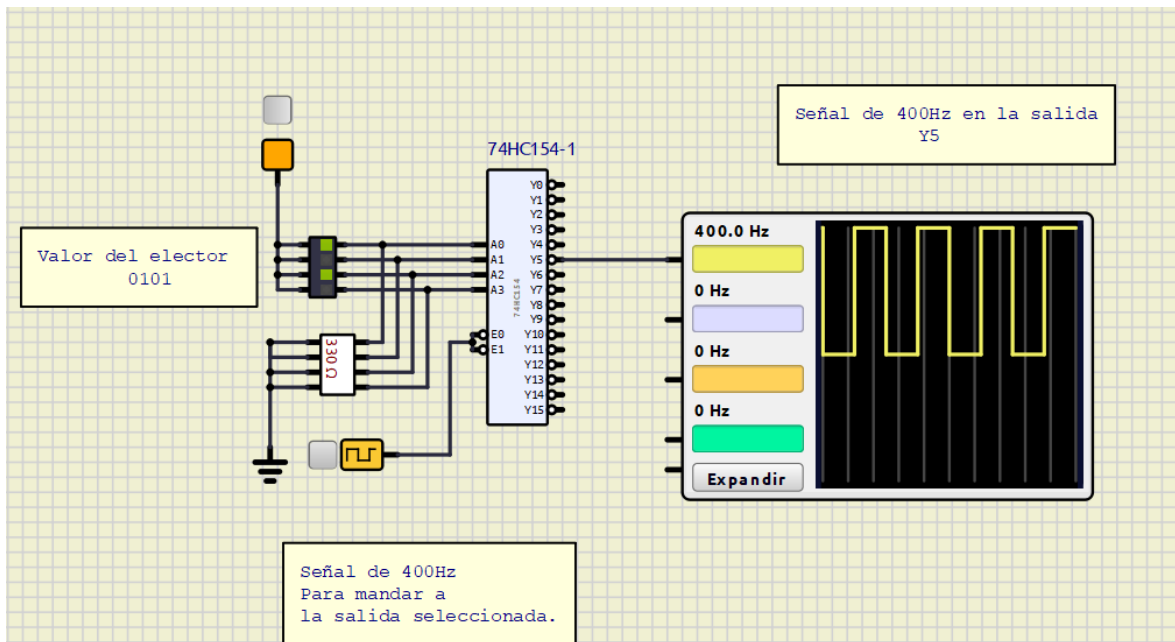


Figura 3.33 Circuito demultiplexor 74154 para mandar una señal a cualquier salida.

3.3.3 Decodificadores

Existen circuitos decodificadores que funcionan de una manera distinta a los demultiplexores, tienen líneas de entrada y de salida e incluso líneas de habilitación, pero tienen una función en especial, también son circuitos lógicos combinacionales, algunos ejemplos son los siguientes:

Decodificador de BCD a decimal.

También conocido como decodificador uno de diez o decodificador de cuatro a diez, Este circuito integrado tiene cuatro líneas de entrada en la que se les da la combinación binaria que puede ser de 0 a 9 (BDC decimal codificado en binario) y se activará a nivel bajo la línea que corresponde a esa combinación, son diez las líneas de salida, si recibe alguna combinación binaria de entrada mayor a nueve no se habilitará ninguna salida, este decodificador no tiene líneas de habilitación. El circuito que hace esta función es el 7442.

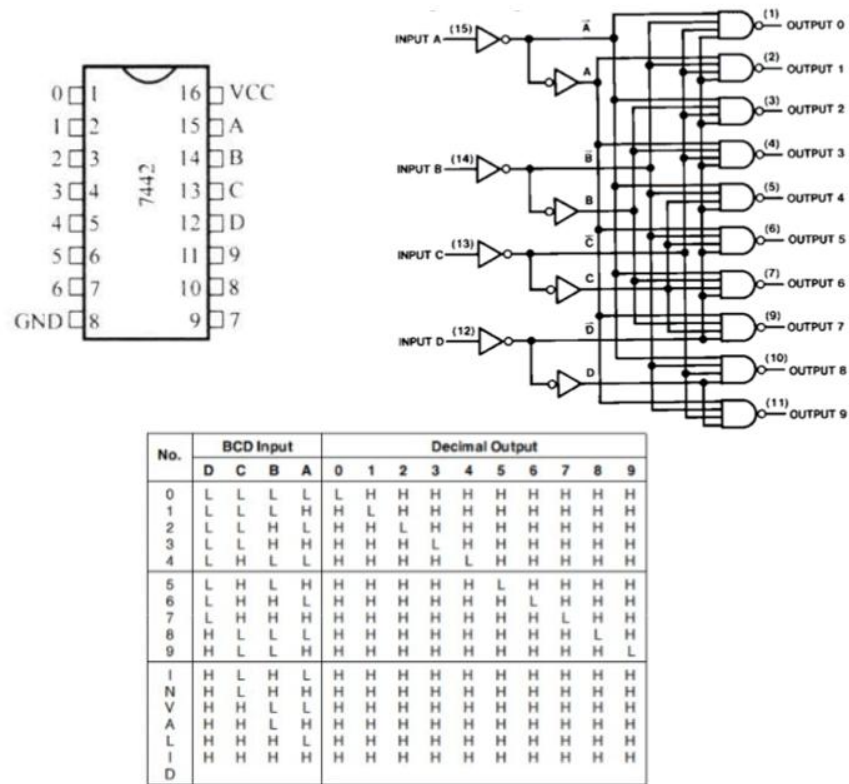


Figura 3.34 Decodificador 7442, circuito lógico y tabla de verdad.

El decodificador 7445 también es un decodificador de BCD a decimal, sólo que tiene las salidas en colector abierto, lo que permite drenar hasta 80 mA cuando la salida está a nivel bajo y puede manejar hasta 30 V a nivel alto, lo que lo hace ideal para el manejo de relevadores, de corriente directa o lámparas indicadoras.

Ejemplo

Diseñe un circuito para que muestre el funcionamiento del decodificador 7442 de tal manera que en la salida a través de un LED se muestre cual es la combinación binaria de entrada.

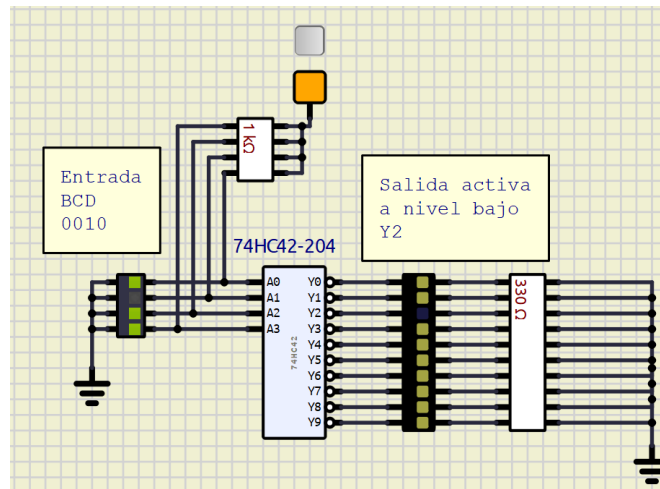


Figura 3.35 Funcionamiento del decodificador 7442 en SimulIDE.

Decodificador BCD a 7 segmentos.

Este tipo de decodificadores son circuitos lógicos combinacionales diseñados para el manejo de los displays de 7 segmentos, estos decodificadores tienen cuatro líneas de entrada que se usan para dar una combinación binaria de 0 a 9(BCD) y tiene 7 línea de salida que se conectan a los leds(segmentos) del display, este es un arreglo de leds en una caja rectangular en forma de un ocho, unidos por el cátodo o el ánodo del led y cada led(segmento) ya tienen un nombre específico, dependiendo de los leds que se enciendan es el número que se mostrará en el display.

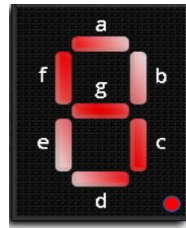
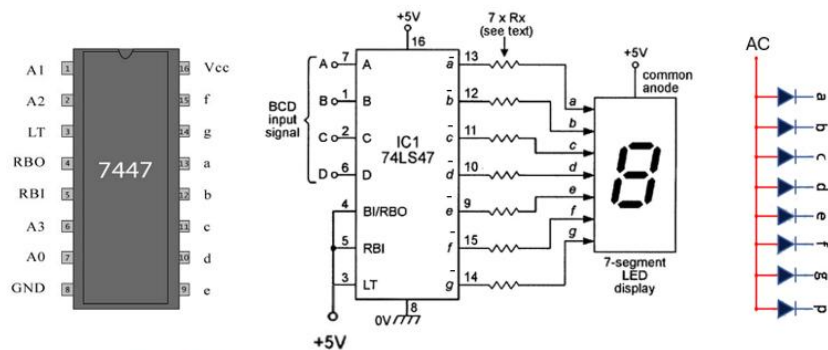


Figura 3.36 Display de 7 segmento.

Los circuitos decodificadores de BCD a 7 Segmentos son el 7447 para el display de ánodo común y el 7448 para el display de cátodo común.



Function Table

Decimal or Function	LT	RBI	D	C	B	A	BI/RBO (Note 1)	a	b	c	d	e	f	g	Note
0	H	H	L	L	L	L	H	L	L	L	L	L	L	H	(Note 2)
1	H	X	L	L	L	H	H	H	L	L	H	H	H	H	
2	H	X	L	L	H	L	H	L	L	H	L	L	H	L	
3	H	X	L	L	H	H	H	L	L	L	L	H	H	L	
4	H	X	L	H	L	L	H	H	L	L	H	H	L	L	
5	H	X	L	H	L	H	H	L	H	L	L	H	L	L	
6	H	X	L	H	H	L	H	H	H	L	L	L	L	L	
7	H	X	L	H	H	H	H	L	L	L	H	H	H	L	
8	H	X	H	L	L	L	H	L	L	L	L	L	L	L	
9	H	X	H	L	L	H	H	L	L	L	H	H	L	L	
10	H	X	H	L	H	L	H	H	H	H	L	L	H	L	
11	H	X	H	L	H	H	H	H	H	L	L	H	H	L	
12	H	X	H	H	L	L	H	H	L	H	H	H	L	L	
13	H	X	H	H	L	H	H	L	H	H	L	H	L	L	
14	H	X	H	H	H	L	H	H	H	H	L	L	L	L	
15	H	X	H	H	H	H	H	H	H	H	H	H	H	H	
BI	X	X	X	X	X	X	L	H	H	H	H	H	H	H	(Note 3)
RBI	H	L	L	L	L	L	L	H	H	H	H	H	H	H	(Note 4)
LT	L	X	X	X	X	X	H	L	L	L	L	L	L	L	(Note 5)

H = HIGH level, L = LOW level, X = Don't Care

Figura 3.37 Decodificador 7447, conexiones con el display, arreglo del display y tabla de verdad.

Este decodificador tiene 3 líneas de control B1/RB0, RB1 y LT que se utilizan para probar los segmentos o para apagarlos cuando no se requieran y ahorrar energía, su manejo lo podemos ver en la tabla de verdad.

Decodificador 7448 de BCD a 7 segmentos.

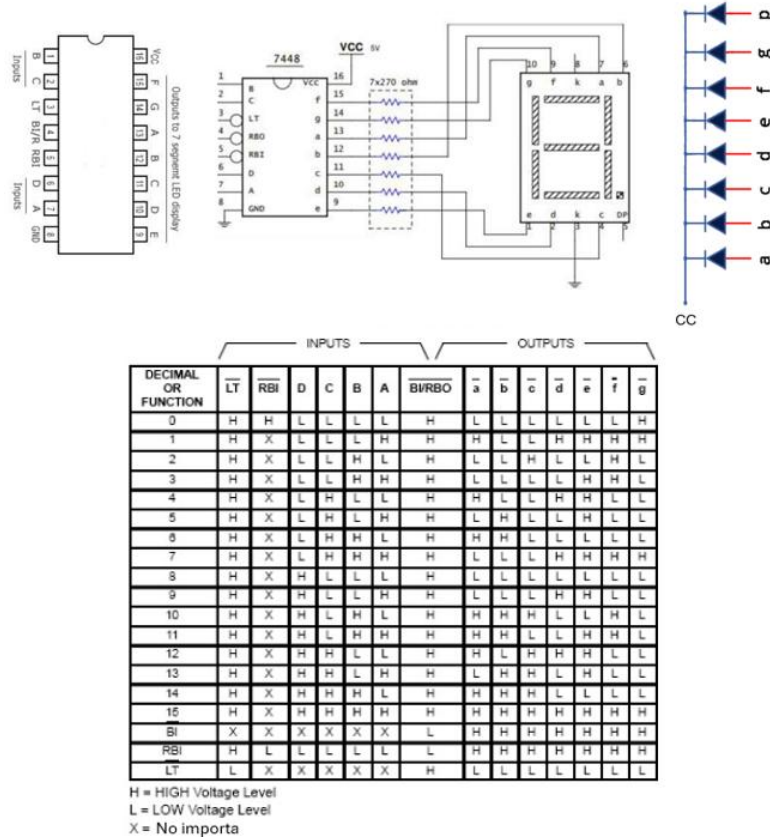


Figura 3.38 Decodificador 7448, conexión al display, arreglo del display y tabla de verdad.

Ejemplo

Diseñe un circuito para el manejo del decodificador 7448 de tal manera que en un display de 7 segmentos se vea el conteo del 0 al 9, utilice el contador BCD 74192.

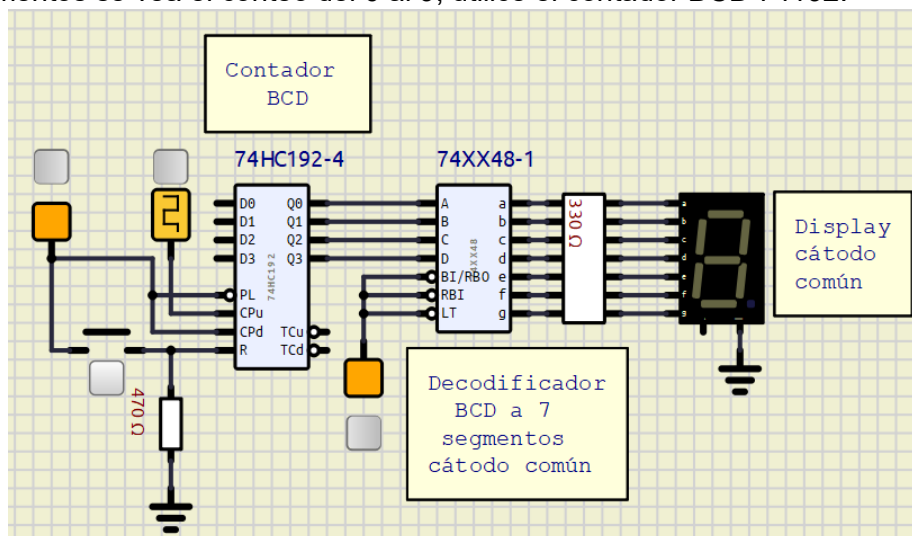


Figura 3.39 Contador de 0 a 9 utilizando el decodificador 7448 y un contador BCD 74192 en SimulIDE.

3.3.4 Codificadores.

Un codificador(enconder) es un circuito lógico combinacional que trabaja de forma contraria de un decodificador, es decir tiene 2^n entradas y n salidas, dependiendo de la entrada que se active la salida mostrará la combinación binaria al número de entrada.

Este tipo de circuitos son muy útiles en el manejo de teclados lineales, controles remotos y muchas aplicaciones más.

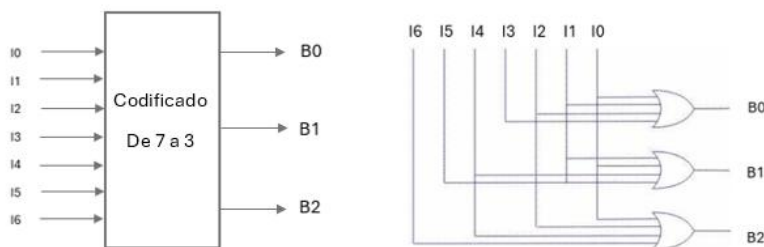
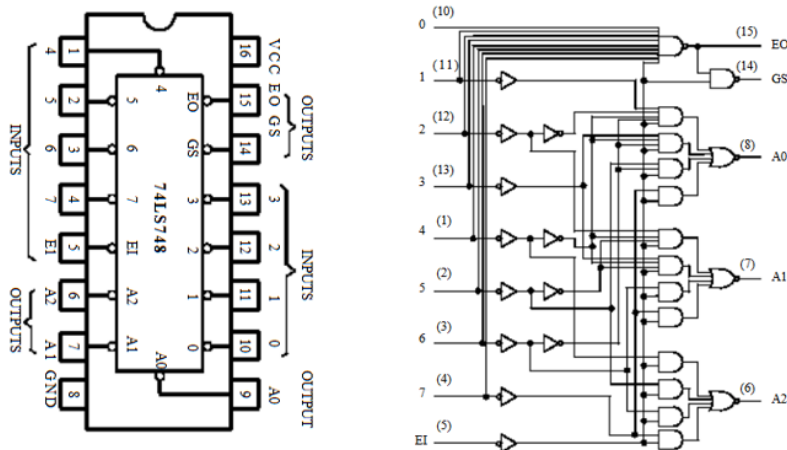


Figura 3.40 Codificador de 7 líneas de entrada a tres de salida y el circuito lógico.

Codificador de 8 a 3 líneas.

El circuito integrado 74148 es un codificador de 8 a 3 líneas de alta prioridad, esto quiere decir que tiene 8 líneas de entrada activas a nivel bajo y 3 de salida donde se mostrará el numero binario de la línea que está siendo activada, como el circuito es de prioridad, si se activan dos o más líneas de entrada la salida mostrará la combinación binaria de la línea de más peso, siendo la de menor peso la línea 0 y la más alta la línea 7.



INPUTS									OUTPUTS				
EI	0	1	2	3	4	5	6	7	A2	A1	A0	GS	EO
H	X	X	X	X	X	X	X	X	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	L
L	X	X	X	X	X	X	X	L	L	L	L	L	H
L	X	X	X	X	X	X	L	H	L	L	H	L	H
L	X	X	X	X	X	L	H	H	L	H	L	L	H
L	X	X	X	X	L	H	H	H	L	H	L	L	H
L	X	X	X	L	H	H	H	H	H	L	H	L	H
L	X	L	H	H	H	H	H	H	H	H	L	L	H
L	L	H	H	H	H	H	H	H	H	H	H	L	H

Figura 3.41 Circuito codificador 74148, diagrama lógico y tabla de verdad.

De acuerdo con la tabla de verdad para que el circuito responda a una entrada, debe estar habilitado es decir E1 a nivel bajo, cuando las salidas GS está a nivel alto y E0 están a nivel bajo indica que todas las entradas están a nivel alto, es decir ninguna línea de entrada activa, si GS está a nivel bajo y E0 a nivel alto, las salidas A0, A1 y A2 representa el valor de la entrada más alta que esta activa (nivel bajo) sin importar si hay más de una línea de entrada activa.

Codificador de decimal a BCD con prioridad.

El circuito integrado 74147 es un codificador de decimal a BCD con prioridad, tiene 9 líneas de entrada activas a nivel bajo para indicar los números decimales del 1 al 9, las salidas representan en binario invertido el número de línea mayor que fue activada sin importar si hay más de una línea activa.

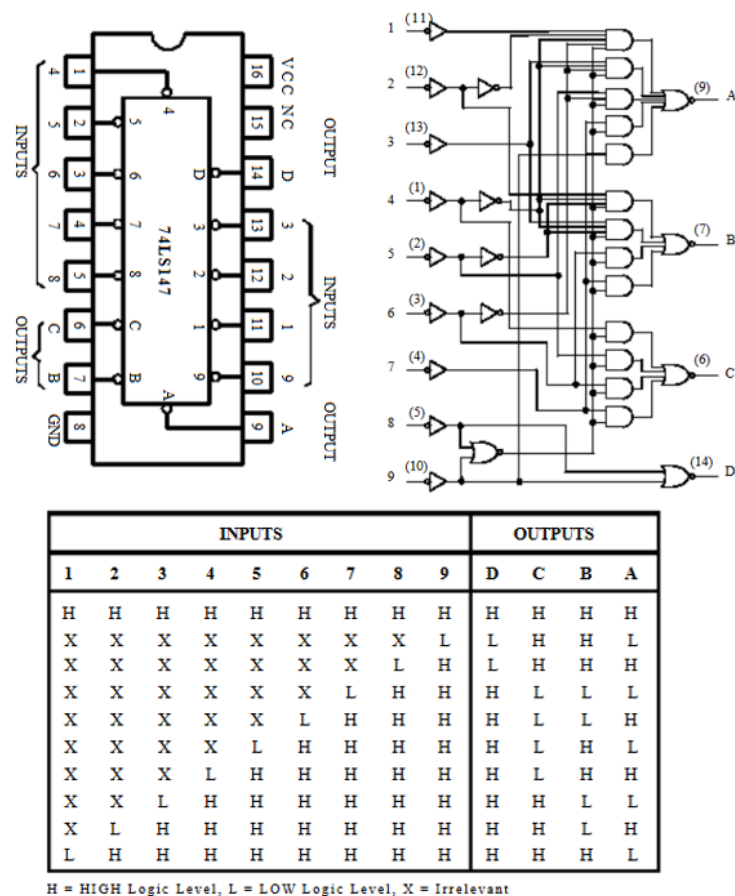


Figura 3.42 Circuito codificador 74147, diagrama lógico y tabla de verdad.

La tabla de verdad indica que cuando no hay ninguna entrada activa las salidas serán todas a nivel alto lo que indicaría la línea 0 que no tiene, recuerde que las salidas están en complemento, si se requiere tener el número de línea activa sin complemento se pueden poner inversores a las salidas(D, C, B y A).

Ejemplo

Diseñe un circuito con el codificador 74147 para mostrar a la salida el valor de la tecla pulsada a la entrada, si se pulsan dos o más teclas deberá dar como salida el número de tecla más alto, las salidas no deberán estar en complemento.

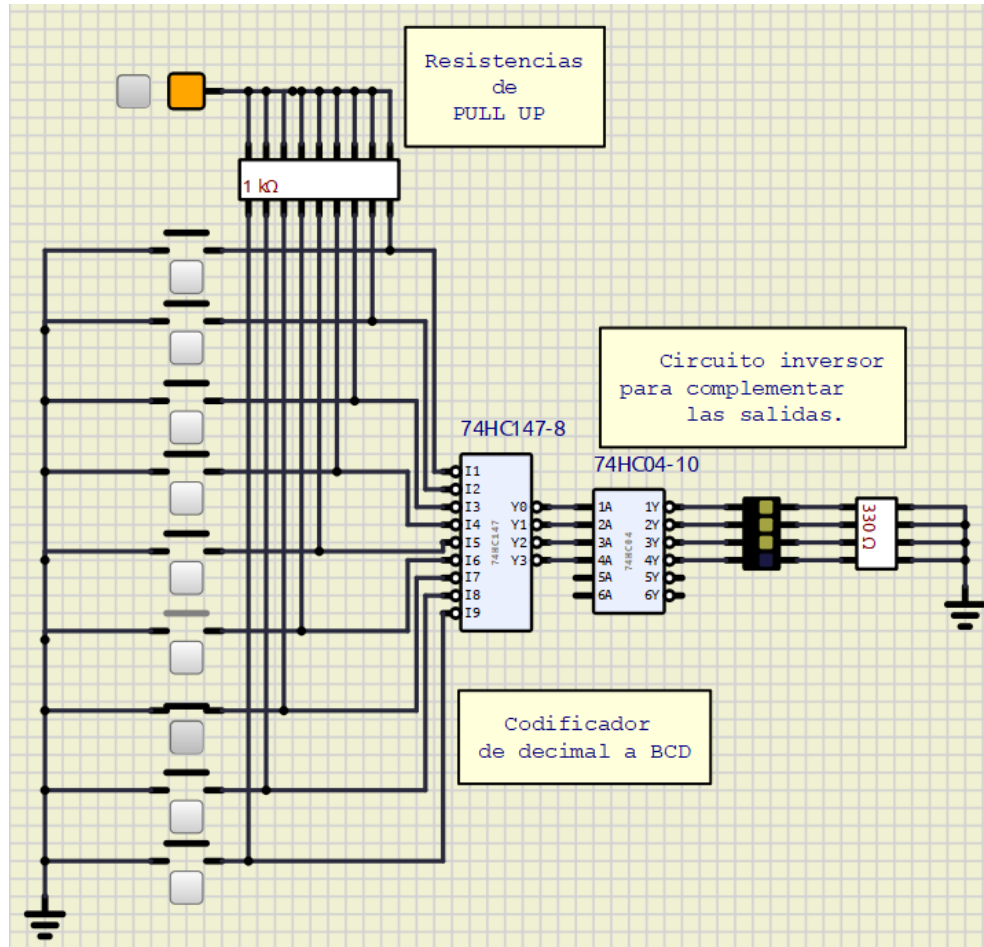


Figura 3.43 Circuito detector de tecla pulsada con el codificador 74147 en SimulIDE.

En el circuito de la figura 3.43 se puede cambiar el teclado lineal por sensores de tal manera que el sensor que más interese este en la entrada más alta y tomar decisiones en función de ellos.

3.3.5 Indicadores numéricos (Display's)

Display de 7 segmentos.

Un display de 7 segmentos es un dispositivo electrónico que se utiliza para mostrar dígitos numéricos y, ocasionalmente, algunas letras. Está compuesto por siete segmentos individuales (LEDs Diodos emisores de luz) que pueden iluminarse de manera independiente. Estos display's se emplean frecuentemente en relojes digitales, calculadoras y otros dispositivos que requieren un método sencillo para mostrar números.

Cada uno de los Leds (segmentos del display) tienen un nombre específico que van de la letra "a" a la "g" la mayoría de los display's también tienen un led para mostrar un punto que puede ser utilizado para aplicaciones donde se requiera el punto decimal, además los display tienen un punto común de conexión para los leds, si todos los leds del display están conectados por el ánodo se le conoce como display de ánodo común y si están conectados al cátodo se le conoce como cátodo común.

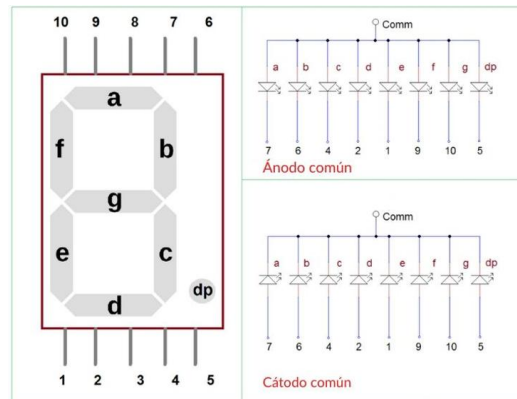


Figura 3.44 display de 7 segmentos.

Es importante saber que cada led del display maneja una corriente máxima que da el fabricante y que se tiene que respetar si no se quiere dañar el dispositivo, por lo tanto, cada segmento (led) debe llevar su resistencia limitadora.

Existen circuitos integrados llamados decodificadores de BCD (decimal codificado en binario) a 7 segmentos para el manejo de estos display's, ejemplo de estos son el circuito son el 74LS47 para el display de ánodo común y el 74LS48 para el display de cátodo común.

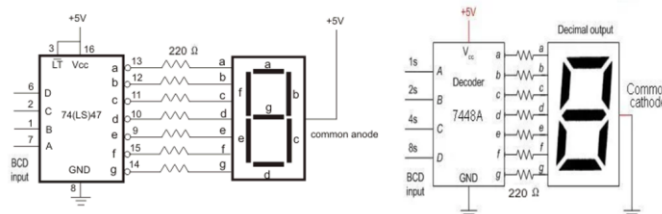


Figura 3.45 decodificadores de cátodo y ánodo común.

Existen paquetes de display donde todos los segmentos de cada display están conectados en paralelo, sólo el común de cada uno de ellos está libre para su control. Este tipo de display's son muy útiles cuando se usan dispositivos de control con pocas líneas disponibles para el manejo de estos como lo son los microcontroladores.

Display LCD

Las pantallas de cristal líquido en formato 16x2 y 20x4 son dos tipos comunes utilizadas en proyectos de electrónica y sistemas embebidos.

La **pantalla LCD 16x2** presenta una disposición de 16 columnas y 2 filas, permitiendo mostrar hasta 32 caracteres a la vez. Cada uno de estos caracteres se construye en una matriz de 5x8 píxeles, ideal para representar letras, números y algunos símbolos básicos. Este tipo de pantalla es ideal para proyectos pequeños donde se requiere información simple, como el estado de un dispositivo o lecturas básicas.

Por otro lado, la **pantalla LCD 20x4** dispone de 20 columnas y 4 filas, permitiendo mostrar un total de 80 caracteres. Su espacio adicional resulta útil para proyectos que necesitan visualizar información detallada en varias líneas o múltiples datos, sin necesidad de cambiar la pantalla.

El **display HD44780** es una referencia a módulos LCD que emplean el controlador **Hitachi HD44780**. Este controlador es muy utilizado en pantallas alfanuméricas y es compatible con configuraciones comunes como 16x2 y 20x4.

Características principales de la pantalla HD44780:

- Formato de caracteres: Generalmente, usa una cuadrícula de 5x8 píxeles para cada carácter, para texto y símbolos básicos.
- Interfaz paralela: Utiliza una interfaz de 8 o 4 bits, que facilita la conexión con microcontroladores y microprocesadores.
- Caracteres personalizados: Permite crear hasta ocho símbolos personalizados, que se pueden diseñar asignando patrones de píxeles y guardarlos en la memoria del LCD.
- Control de retroiluminación: Muchos de estos displays incluyen una función de retroiluminación, mejorando la visibilidad y permitiendo ahorrar energía cuando no es necesario iluminar la pantalla.

Funciones y comandos del HD44780:

El controlador HD44780 permite funciones como:

- Ubicación del cursor: Posiciona el cursor en cualquier celda de la pantalla.
- Desplazamiento del texto: Permite mover el texto a la izquierda o derecha.
- Limpieza de pantalla: Ofrece la opción de borrar el contenido y reposicionar el cursor.
- Almacenamiento de caracteres personalizados: Facilita la creación y almacenamiento de símbolos personalizados en ubicaciones de memoria específicas.

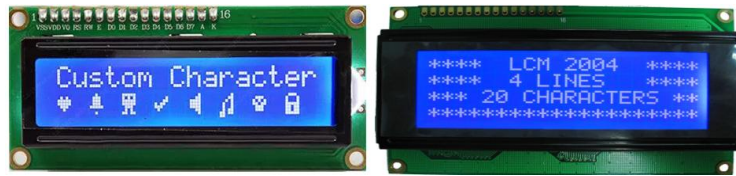


Figura 3.46 Displays de 2 x16 y 4 x20.

Los dos displays tienen la misma configuración tanto en las terminales como en los comandos por lo que se programan de la misma manera.

Para manejar el display con el microcontrolador es necesario conocer las características eléctricas, las terminales, los comandos y cómo hacer los caracteres propios.

Características eléctricas del display LCD 4 x 20.

ELECTRICAL CHARACTERISTICS						
ITEM	SYMBOL	CONDITION	STANDARD VALUE			UNIT
			MIN.	TYP.	MAX.	
Input Voltage	V_{DD}	$V_{DD} = +5\text{ V}$	4.7	5.0	5.3	V
		$V_{DD} = +3\text{ V}$	2.7	3.0	5.3	
Supply Current	I_{DD}	$V_{DD} = +5\text{ V}$	-	8.0	10.0	mA
Recommended LC Driving Voltage for Normal Temperature Version Module	V_{DD} to V_0	-20 °C	5.0	5.1	5.7	V
		0 °C	4.6	4.8	5.2	
		25 °C	4.1	4.5	4.7	
		50 °C	3.9	4.2	4.5	
		70 °C	3.7	3.9	4.3	
LED Forward Voltage	V_F	25 °C	-	4.2	4.6	V
LED Forward Current	I_F	25 °C	-	540	1080	mA
EL Power Supply Current	I_{EL}	$V_{EL} = 110\text{ V}_{AC}, 400\text{ Hz}$	-	-	5.0	mA

Figura 3.47 características eléctricas del LCD.

Terminales del display LCD 4 x20.

INTERFACE PIN FUNCTION		
PIN NO.	SYMBOL	FUNCTION
1	V_{SS}	Ground
2	V_{DD}	+3 V or +5 V
3	V_0	Contrast adjustment
4	RS	"H" Data Input "L" Command Input
5	R/W	H/L read/write signal
6	E	H → L enable signal
7	DB0	H/L data bus line
8	DB1	H/L data bus line
9	DB2	H/L data bus line
10	DB3	H/L data bus line
11	DB4	H/L data bus line
12	DB5	H/L data bus line
13	DB6	H/L data bus line
14	DB7	H/L data bus line
15	A	Power supply for LED (4.2 V)
16	K	Power supply for B/L (0 V)

Figura 3.48 terminales del display LCD 4x20.

Comandos (instrucciones) para controlar el display LCD de 16 x 2 y 4 x20, los comandos son los mismo para los dos displays, sólo se debe tener en cuenta dónde inicia cada renglón para poder posicionar el cursor y los mensajes que se quieran exhibir en el LCD.

Instruction	Instruction Code										Description	Execution time (fosc=270Khz)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Write "20H" to DDRAM, and set DDRAM address to "00H" from AC	1.52ms
Return Home	0	0	0	0	0	0	0	0	0	1	Set DDRAM address to "00H" from AC and return cursor to its original position if shifted. The contents of DDRAM are not changed.	1.52ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	SH	Sets cursor move direction and specifies display shift. These operations are performed during data write and read.	37 μ s
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	D=1:entire display on C=1:cursor on B=1:cursor position on	37 μ s
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	—	—	Set cursor moving and display shift control bit, and the direction, without changing DDRAM data.	37 μ s
Function Set	0	0	0	0	1	DL	N	F	—	—	DL:interface data is 8/4 bits N:number of line is 2/1 F:font size is 5x11/5x8	37 μ s
Set CGRAM Address	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address counter	37 μ s
Set DDRAM Address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address counter	37 μ s
Read Busy Flag and Address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Whether during internal operation or not can be known by reading BF. The contents of address counter can also be read.	0 μ s
Write Data to RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data into internal RAM (DDRAM/CGRAM)	37 μ s
Read Data from RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM)	37 μ s

* "—" : don't care

Figura 3.49 comandos del display LCD.

Memoria ROM interna de LCD con los códigos ASCII que se pueden mostrar en la pantalla.

Upper 4bit Lower 4bit	LLLL	LLLH	LLHH	LLHH	LHLH	LHLH	LHHH	HLLL	HLLH	HLLH	HLLH	HLLH	HLLH	HLLH	HLLH
CG RAM (1)	±	00P	'	P	S	E	á	'	(	R	P	t			
CG RAM (2)	≡	!	1	A	Q	a	u	a	i	"	J	t	y	v	
CG RAM (3)	7	"	2	E	R	b	r	e	f	E	6	°	°	S	8
CG RAM (4)	¿	#	3	C	S	c	e	á	é	ú	'	P	ñ	e	v
CG RAM (5)	[	\$	4	D	T	d	t	á	é	t	'	e	r	z	o
CG RAM (6)	[	%	5	E	U	e	u	á	é	ú	'	t	z	q	W
CG RAM (7)	)	&	6	F	U	t	v	á	é	ú	'	u	j	0	0
CG RAM (8)	)	'	7	B	U	e	u	á	é	ú	'	R	x	°	Á
CG RAM (1)	(	C	B	X	h	x	e	u	á	é	ú	'	E	k	W
CG RAM (2)	{	>	9	I	Y	i	y	á	é	ú	'	L	T	W	W
CG RAM (3)	*	*	:	J	Z	j	z	e	u	á	é	ú	7	z	μ
CG RAM (4)	[	+	:	K	L	k	L	C	i	P	3	*	L	T	W
CG RAM (5)	≡	,	<	L	\	1	1	i	P	8	*	J	8	z	0
CG RAM (6)	ω	—	=	M	I	n	>	i	3	8	+	°	U	π	—
CG RAM (7)	E	.	.	>	N	^	n	^	á	é	ú	'	0	0	p
CG RAM (8)	3	/	?	0	_	o	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ

Figura 3.50 memoria ROM de códigos ASCII del LCD.

El display LCD cuenta con una memoria RAM de 64 localidades llamada CGRAM par que el usuario diseñe sus propios caracteres y los guarde en ella, estos permanecerán mientras no se quite la alimentación al LCD, el usuario puede guardar hasta 8 caracteres ya que se requiere de 8 datos por cada carácter.

La matriz para formar un carácter es la siguiente:

Instruction	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0	Description
Set CGRAM Addr	0	0	0	1	Character Generator RAM						Sets CGRAM Address

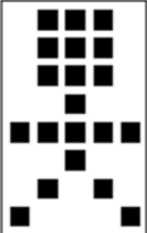
Carácter a definir por el usuario	Valores a almacenar En la CG RAM																		
	<table> <tr><th>BINARIO</th><th>HEXADECIMAL</th></tr> <tr><td>01110</td><td>\$0E</td></tr> <tr><td>01110</td><td>\$0E</td></tr> <tr><td>01110</td><td>\$0E</td></tr> <tr><td>00100</td><td>\$04</td></tr> <tr><td>11111</td><td>\$0F</td></tr> <tr><td>00100</td><td>\$04</td></tr> <tr><td>01010</td><td>\$0A</td></tr> <tr><td>10001</td><td>\$11</td></tr> </table>	BINARIO	HEXADECIMAL	01110	\$0E	01110	\$0E	01110	\$0E	00100	\$04	11111	\$0F	00100	\$04	01010	\$0A	10001	\$11
BINARIO	HEXADECIMAL																		
01110	\$0E																		
01110	\$0E																		
01110	\$0E																		
00100	\$04																		
11111	\$0F																		
00100	\$04																		
01010	\$0A																		
10001	\$11																		

Figura 3.51 formato de un carácter del usuario.

- Para guardar un carácter debe seguirse los siguientes pasos:
- Poner el LCD en forma de comando RS = 0
- Poner la terminal R/W en forma de escritura =0
- Poner en el bus de datos la primera dirección de la CGRAM, 0x40.
- Activar la terminal E enable a 1 y regresarla a 0.
- Regresar RS = 1

De aquí en adelante se deben enviar los datos de cada símbolo, ocho por cada uno, los primeros ochos datos corresponden al carácter 0, los siguientes al 1 y así hasta el carácter 8.

Para exhibir los caracteres se lee el primero como 0, el siguiente como 1 y así hasta el 7. Recuerde que antes de mandar caracteres se tiene que poner el cursor a la posición del display en la que se quiere exhibir, ya que el apuntador se quedó en la CGRAM.

Resumen de comandos o instrucciones del display LCD.

La terminal RS debe estar en 0 para que sea un comando y 1 para mandar un dato a la pantalla.

0x01 Limpieza del display, borra toda la información y regresa el curso a home.

0x02 Regresa el cursor a y mensaje a home sin borrar la información.

0x04 al 0x07 Configura el movimiento del cursor y/o el display en lectura y escritura.

0x08 al 0x0F Prende o apaga el curso y display y dar formato al cursor

0x10 Mueve el cursor una posición a la izquierda.

0x14 Mueve el curso una posición a la derecha.

0x18 Mueve el display una posición a la izquierda.

0x1C Mueve el display una posición a la derecha.

0x20 Configura el display en bus de datos de 4 bits y un renglón.

0x28 Configura el display en bus de datos de 4 bits y dos renglones.

0x30 Configura el display en bus de datos de 8 bits y un renglón.

0x38 Configura el display en bus de datos de 8 bits y dos renglones.
0x80 Ubica el cursor al inicio del primer renglón.
0xC0 Ubica el cursor al inicio del segundo renglón.
0xD4 Ubica el cursor al inicio del cuarto renglón.

Cuando la terminal RS es 1 el dato que se pone en el display es el código ASCII que debe exhibirse en la posición actual del cursor, recuerde que los caracteres del usuario son los datos 0 para el primero 1 para el segundo y así hasta el 7.

Si R/W está a 1 se hace una lectura (bus de datos de salida) lo que manda el display al bus en el bit 7 es la bandera de ocupado BF, si es 1 no ha terminado de ejecutar el comando anterior si es 0 ya está listo para otro comando.

Normalmente esta línea se manda a 0, y externamente se hace un retardo entre un comando y otro, de lo contrario se debe tener en cuenta que el bus de datos del display en lectura es de salida y hacer los cambios en el circuito para evitar dañarlo.

Display en bus de 4 bits.

Para trabajar el display en bus de datos de 4 bits, se trabajan los bits D7 a D4.

Se pone RS en 0 y se manda un 2 en el bus, 0010 (D7 a D4) se pone E = 1 y se regresa a 0, el cursor aparecerá en home, de aquí en adelante se mandan los comandos que se requieran en dos partes, primero la parte alta del comando y se habilita y deshabilita, en seguida se manda la parte baja del comando, se habilita y deshabilita.

La otra forma es mandar en dos partes la siguiente lista de comandos: 0x23, 0x22 y 0x28 y listo aparece el cursor en home.

Ejemplo

Mandar el mensaje “HOLA” al display LCD en formato de bus de 8 bits.

Los comandos son: 38H, 0EH y 01H. Para las letras usar el código ASCII.

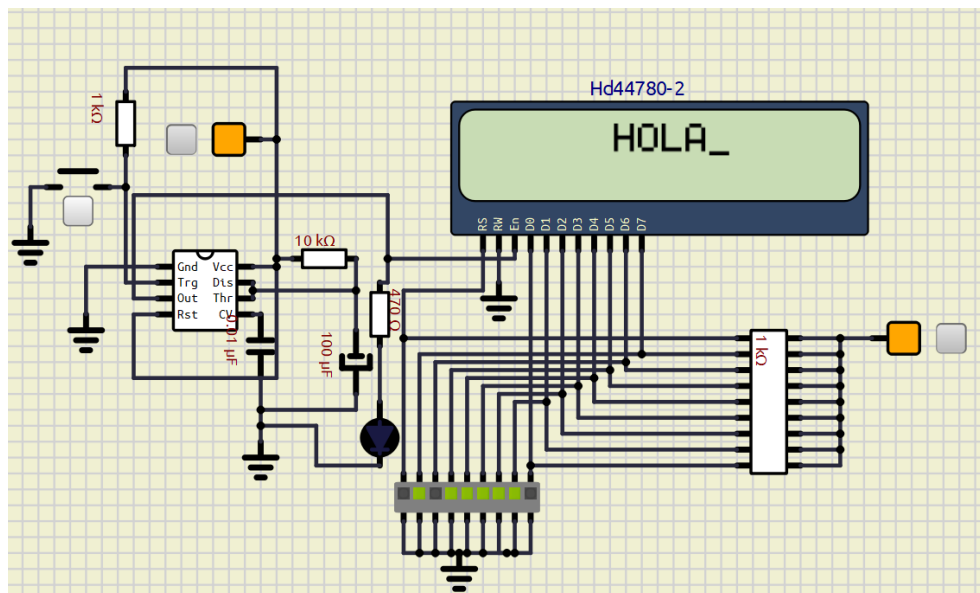


Figura 3.52 Display LCD de 2 renglones por 16 columnas simulado en SimulIDE.

3.4 Dispositivos lógicos programables.

Dispositivos Lógicos Programables (PLD)

Los PLD's son circuitos integrados digitales con arreglo de matrices de compuertas (AND y OR) que pueden ser programadas mediante Software(VHDL) por el usuario para realizar una función lógica específica de acuerdo con las necesidades del diseño que se desea implementar, estos dispositivos se pueden encontrar disponibles con n entradas y m salidas y se clasifican de acuerdo al arreglo interno de las matrices de las compuertas, algunas matrices pueden ser fijas es decir no se puede cambiar su conexión y otras si pueden ser programables por el usuario.

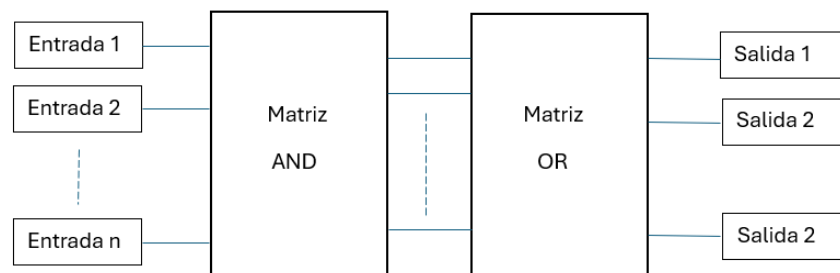


Figura 3.53 Esquema general de un Dispositivo Lógico Programable PLD.

Una clasificación de estos dispositivos es:

PAL (Arreglo Lógico Programable) Arreglo de compuertas AND programable y OR fijas.

PLA (Arreglo Lógico Programable) Arreglo AND y OR programables.

GAL (Matriz Lógica Genérica).

CPLD (Dispositivos Lógicos Programables Complejos).

PAL (Arreglo Lógico Programable)

Es un arreglo de compuertas AND que se pueden programar y una matriz de compuertas OR fija. Es una estructura simple con la que se pueden hacer diseños de circuitos combinacionales.

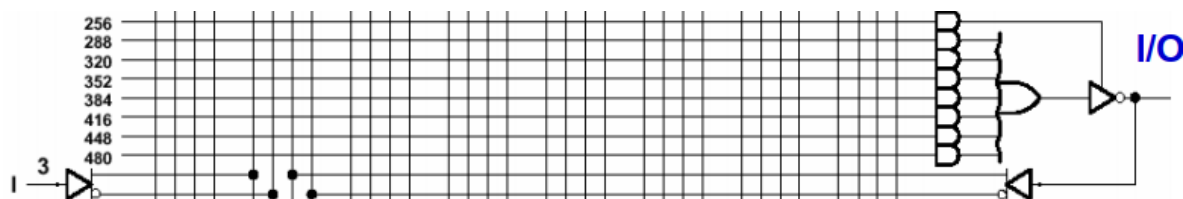


Figura 3.54 PAL arreglo de compuertas AND programables y Matriz de compuertas OR fija.

PAL16L8

PLD16L8 este dispositivo puede tener hasta 16 entradas y de 2 hasta 8 salidas, dependiendo la configuración que se haga del circuito, tiene una matriz de compuertas AND programable y compuertas OR fijas, no tiene registros de salida por lo que sólo se hace lógica combinacional, se puede programar una sola vez (OTP).

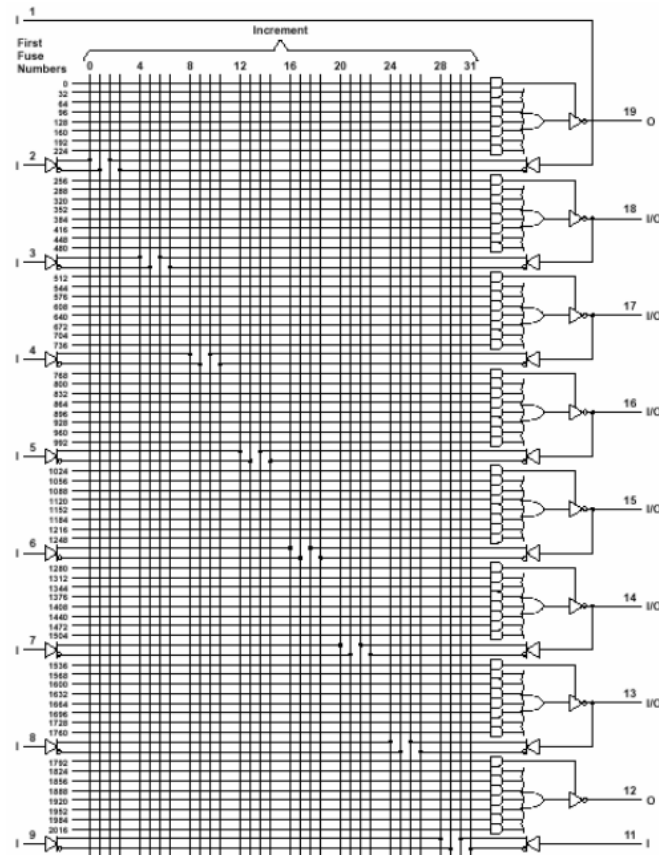


Figura 3.55 Arreglo interno del PAL 16L8, 16 entradas 8 salidas.

PAL tipo R

Este tipo de PLD's tienen un registro en la salida por lo que también se pueden diseñar circuitos secuenciales ya que puede almacenar sus estados y trabajar con señal de reloj.

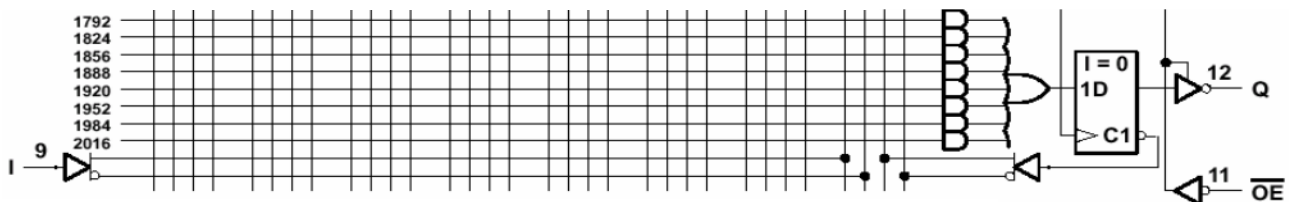


Figura 3.56 PAL16R4 Matriz AND programable OR fija y registros de salida tipo D.

PAL16R4

El PAL16R4 puede tener de 10 a 12 entradas dependiendo la configuración que se haga, cuatro salidas con registro flip flop's tipo D y hasta 4 salidas sin registro dependiendo de la configuración, tiene una matriz AND programable y una OR fija, sólo se puede programar una sola vez (OTP).

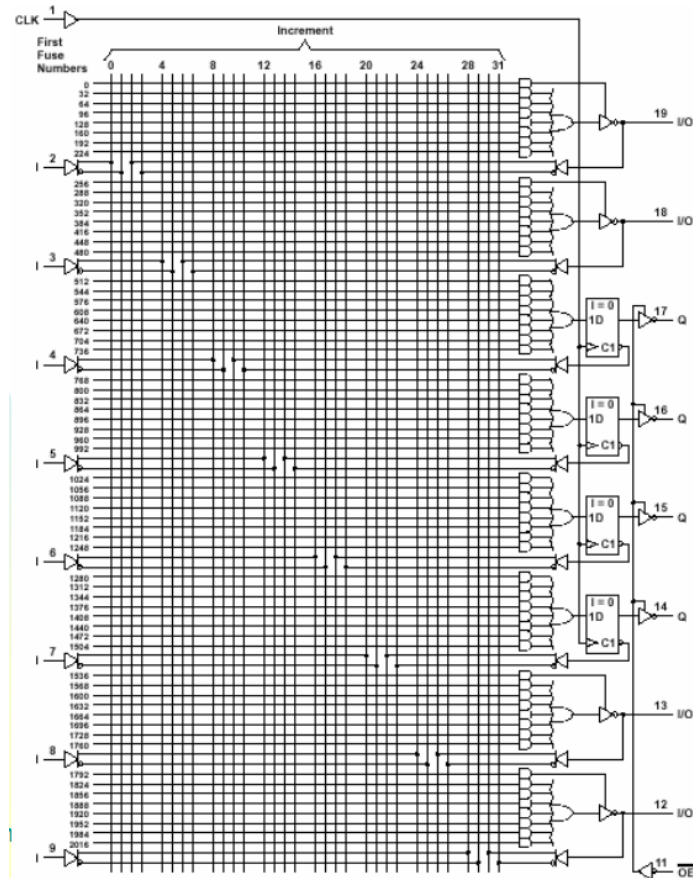


Figura 3.57 Arreglo interno del PAL16R4.

PAL tipo V (versátil)

Esta familia de PLD's es una versión mejorada y más versátil que los primeros dispositivos PAL como los PAL16R4 o los PAL16L8. Estos tipos de dispositivos PAL se diseñaron para tener una mayor flexibilidad y funcionalidad, combinando lo mejor de los dispositivos anteriores, permitiendo mejores configuraciones tanto para circuitos combinacionales como para secuenciales, cuentan con una macrocelda de salida formada por un registro, un multiplexor de salida y un multiplexor para retroalimentar las entradas.

se pueden programar una sola vez(OTP).

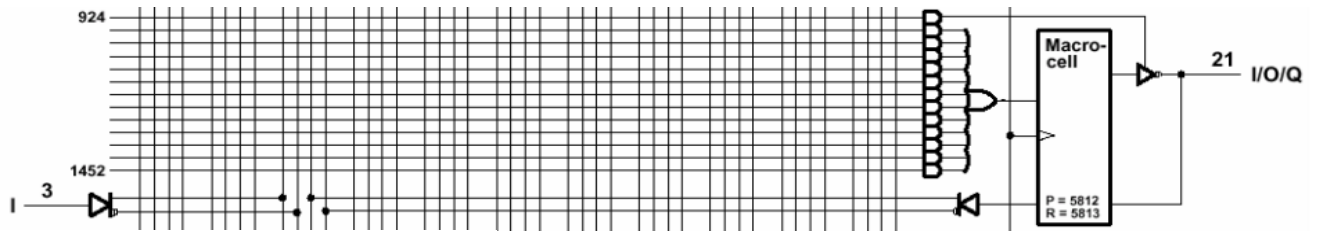


Figura 3.58 PAL tipo V Matriz AND programable OR fija con macrocelda.

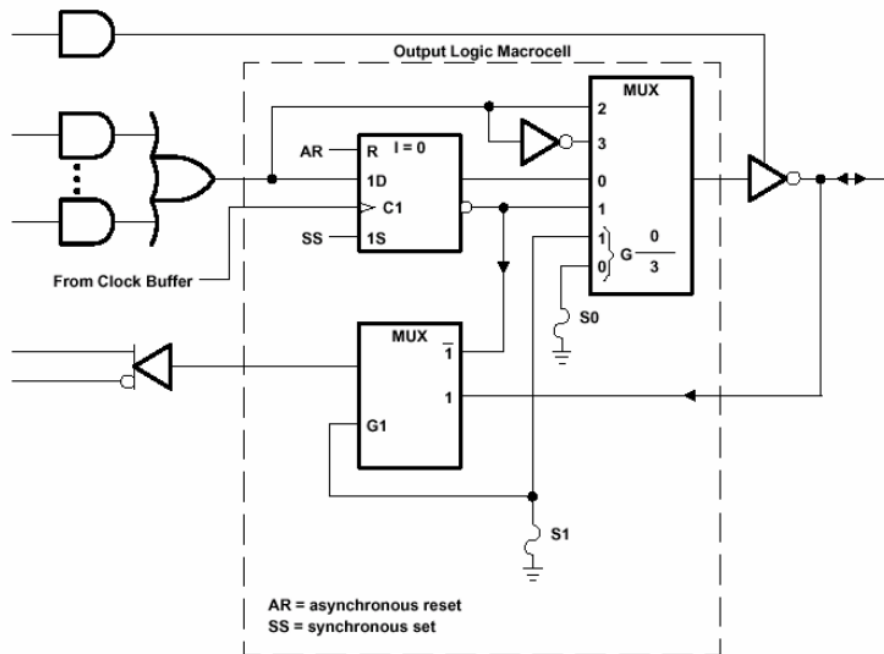


Figura 3.59 Macrocella de un PAL tipo V.

PAL22V10

Este dispositivo puede tener hasta 12 entradas y 10 salidas que se pueden configurar como combinacionales o con registro, tiene una matriz de compuertas AND programable y una OR fija, se programa una sola vez(OTP).

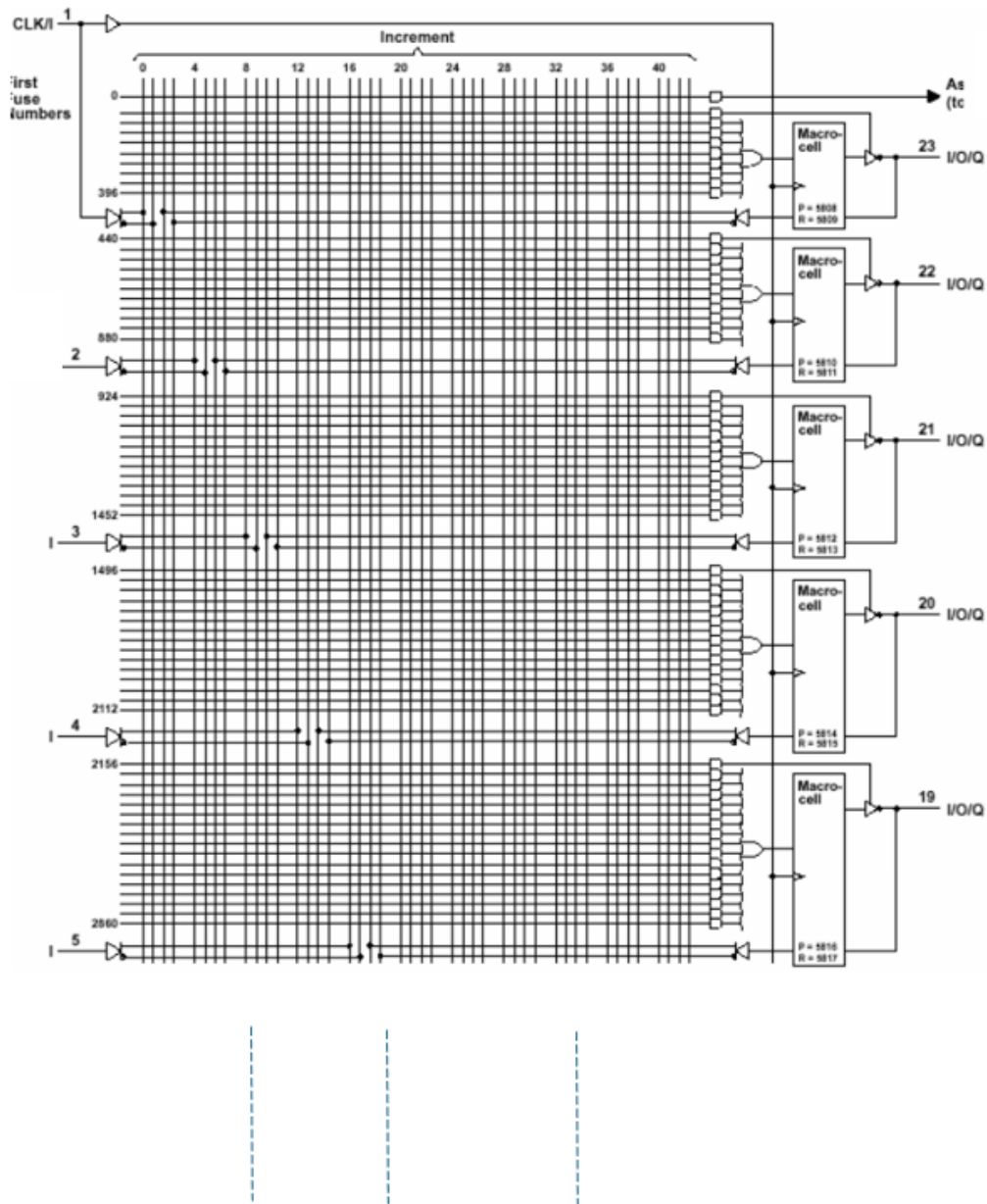


Figura 3.60 PAL22V10.

PLA (Matriz de Lógica Programable)

El PLA es un circuito integrado que tiene una estructura totalmente programable, tiene matrices de compuertas AND y OR programables, a diferencia del PAL que tiene la matriz de compuertas OR fija, por lo que se pueden programar funciones lógicas más personalizadas. También son conocidos como FPLA (Matriz de Compuertas Lógicas Programables en Campo)

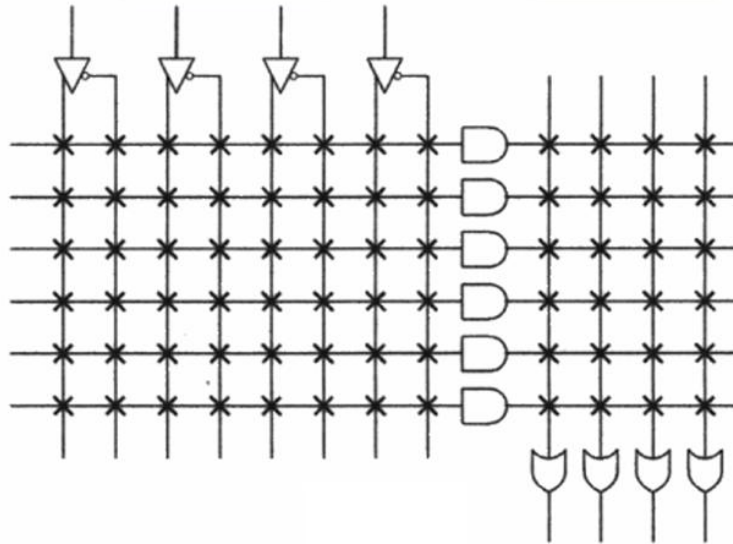


Figura 3.61 Estructura de un PLA con matriz de compuertas AND y OR programables.

GAL (Matriz Lógica Genérica).

Una GAL es un dispositivo de lógica programable reprogramable, diseñado como una evolución de los PALs. Estos dispositivos se pueden reprogramar múltiples veces ya que están contruidos generalmente con la tecnología EEPROM, cuentan con una matriz de compuertas AND programable y un de compuertas OR fija, tienen una macrocelda de salida por lo que se puede diseñar funciones lógicas combinacionales y también secuenciales. Pueden reemplazar múltiples tipos de dispositivos PAL con un solo circuito integrado, ahorrando espacio y costo. Son ideales para hacer prototipos y producción.

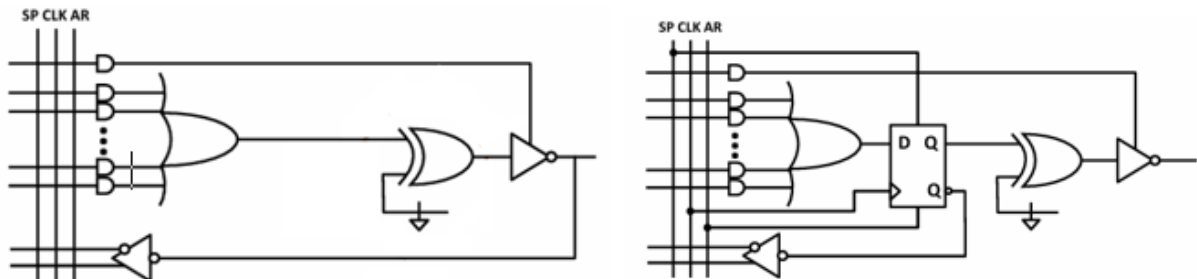


Figura 3.62 Bloques de salida combinacional y salida registrada, polaridad de salida programable.

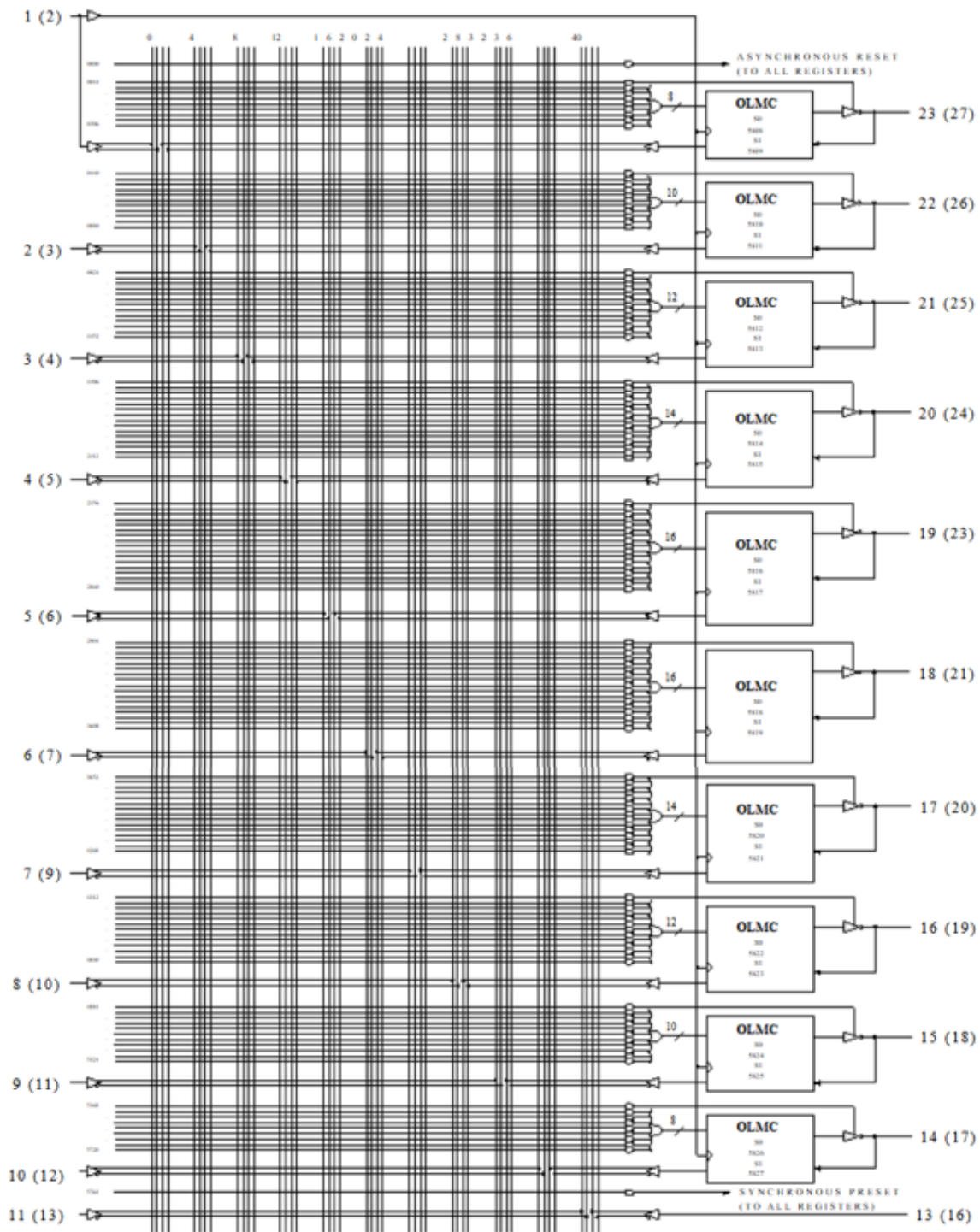


Figura 3.63 GAL22V10 Lattice Semiconductor Corporation. [16]

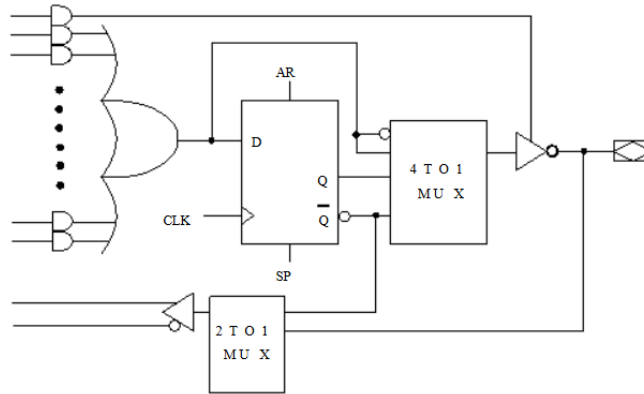


Figura 3.64 Macrocelda de salida de la GAL22V10.

CPLD

Los CPLD (*Dispositivo Lógico Programable Complejo*) son circuitos lógicos integrados programables que permite implementar muchas funciones lógicas que pueden ser complejas y personalizadas. Es una evolución de los PALs y GALs, con más capacidad, flexibilidad y rendimiento. Compuesto por varios bloques lógicos llamados macroceldas conectadas entre sí. Usa tecnología EEPROM o Flash, permitiendo que se pueda reprogramar una gran cantidad de veces. Son de muy alta capacidad, soporta miles de compuertas lógicas, en este tipo de circuitos se pueden implementar funciones lógicas complejas, máquinas de estados, registros, contadores, etc.

los CPLDs tienen tiempos de propagación más predecibles.

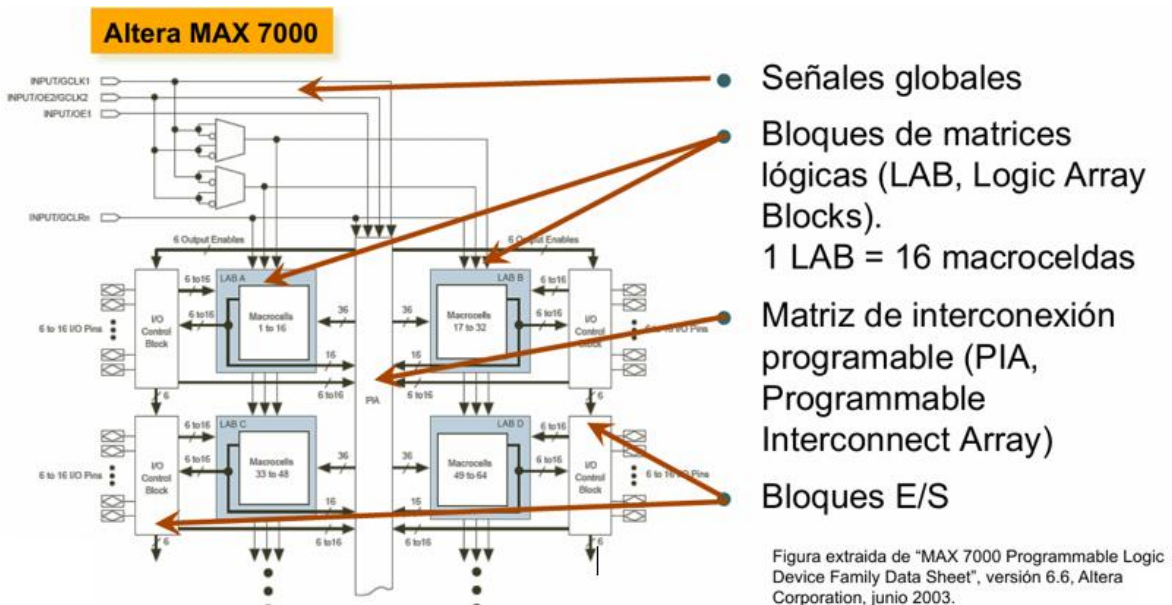


Figura 3.65 Diagrama a bloques del CPLD Max 7000 de Altera. [17]

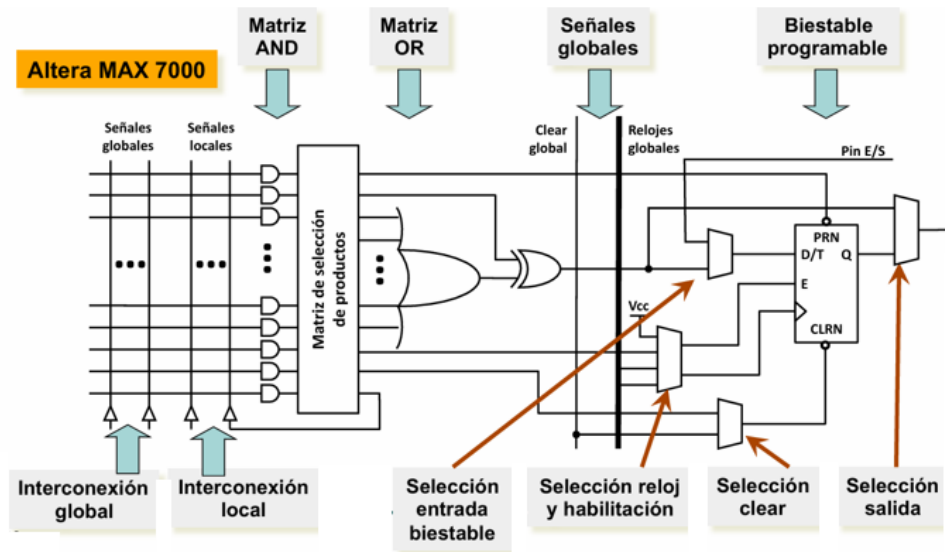


Figura 3.66 CPLD, Macrocelda de MAX 7000 Altera. [17]

Matriz de interconexión global (PIA)

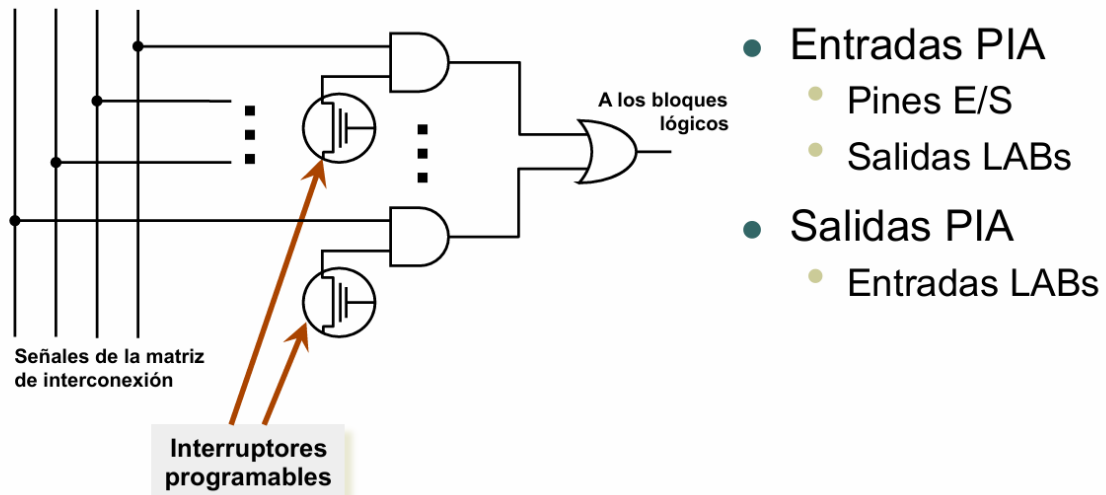


Figura 3.67 Matriz de interconexión global CPLD. [17]

Resumen de las características del CLPD

Estructura de PAL con registros y lógica de interconexión.

Capacidad media (hasta 25000 puertas).

La matriz de interconexión global limita el tamaño.

ISP (In-System Programming). JTAG.

Velocidad media/alta.

Consumo de energía alto.

Tecnología EPROM (reprogramable, no volátil)

Precio bajo

FPGA (Matrices de Compuertas Programables en Campo)

Estos dispositivos superan las limitaciones de tamaño de los CPLDs mediante una arquitectura avanzada.

Ofrecen mayor variedad de recursos lógicos.

- Lógica combinacional
- Lógica secuencial
- Memoria RAM
- Conformadores de reloj
- Señales globales
- Multiplicadores

Fabricantes:

Xilinx

Altera

Actel

Atmel

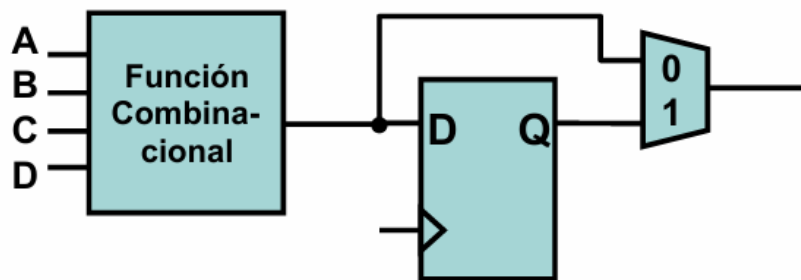


Figura 3.68 FPGA celda lógica básica. [17]

Función combinacional:

- LUT (Look-Up Table): SRAM, volátil.

Función combinacional + Biestable.

Otras variaciones:

- 2 FC + 1 biestable.
- 2 FC + 2 biestables.

Funcionalidad adicional:

- Lógica de acarreo.
- FC de 6 u 8 entradas.
- Varias señales de reloj y reset.
- Diferentes configuraciones del biestable: nivel, flanco de subida, flanco de bajada.

FPGA interconexiones.

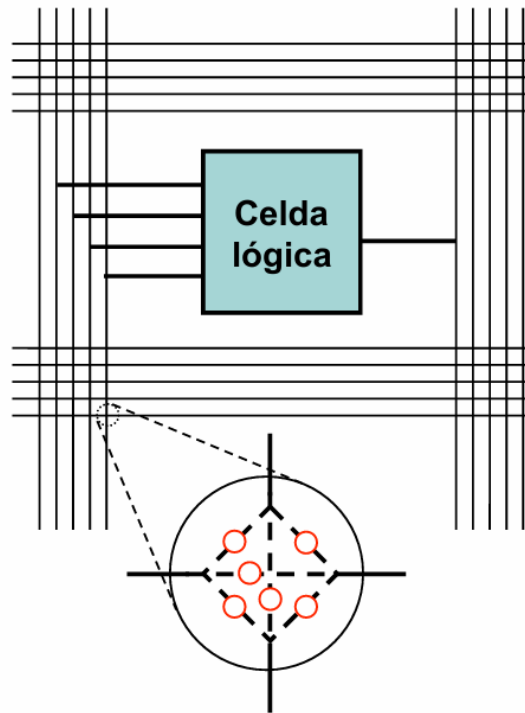


Figura 3.69 FPGA interconexiones.

Interconexiones programables

- Locales:
- Abundantes y rápidas
- Para conectar celdas cercanas
- Globales
- Para conectar zonas lejanas

Lectura del número del circuito integrado.

El número que tiene el circuito integrado PAL se debe leer de la siguiente manera:

PAL (CE) XX AB YY C ZZ DEF

CE : En CMOS borrable eléctricamente.

XX : número de entradas.

AB : Estructura de salida

 L : Combinacional nivel activo bajo.

 H: Combinacional nivel activo alto.

 R: Registro

 V: Versátil.

YY: número de salidas.

C: Consumo(Normal, bajo o muy bajo)

ZZ: Tiempo de programación en ns.

DEF: Tipo de encapsulado, rango de temperatura.

Herramientas para programar los dispositivos:

USB Blaster / USB Blaster, II Intel / Altera (Quartus) para CPLD, FPGA

JTAG-HS3 / Platform Cable USB II, Xilinx / AMD (Vivado, ISE) para CPLD, FPGA

Lattice USB Programmer Lattice (Diamond, Radiant) para CPLD, FPGA

Atmel-ICE / Microchip SNAP Atmel / Microchip (WinCUPL, MPLAB X) para GAL, CPLD

TL866II Plus Multi-marca (universal) GAL, PAL, EEPROM, otros

Herramientas de Software:

Quartus Prime Intel / Altera para CPLD, FPGA.

Vivado Design Suite Xilinx (ahora parte de AMD) para FPGA.

ISE WebPACK Xilinx CPLD, FPGA (más antiguos).

Lattice Diamond / Radiant Lattice Semiconductor para CPLD, FPGA.

Microchip MPLAB X / Libero SoC Microchip (antes Atmel) para CPLD, FPGA.

WinCUPL Atmel / Microchip para PAL, GAL.

Synplify Pro Synopsys (independiente) FPGA (varios fabricantes).

VHDL/Verilog IDEs (como ModelSim) Simulación y diseño lógico para Todos los PLD.

Algunos fabricantes de los distintos PLDs se muestran en las siguientes figuras:

Fabricante	Dispositivo lógico Programable simple
Altera	Clásicos, Flash Logic
Atmel	PAL
Cypres	PAL
Lattice	GAL
Philips	PAL
Vantis	PAL

Figura 3.70 Fabricantes de Dispositivos Lógicos Programables Simples SPLDs.

Fabricante	Dispositivos Lógicos Programables Complejos
Altera	MAX 5000, 7000 y 9000
Atmel	ATF y ATV
Cypress	FLASH370 y ULTRA3700
Lattice	IspLSI1000 a 8000
Philips	XPLA
Vantis	MACH 1 a 5
Xilinx	XC9500 CoolRunner

Figura 3.71 Fabricantes de Dispositivos Lógicos Programables Complejos CPLDs.

Fabricante	Arreglo de Compuertas Programables en Campo
Actel	ACT1 a 3, MX y SX
Altera	Flex, Cyclone y Stratix
Atmel	AT6000 y AT40K
Lucent	ORCA 1 a 3
QuickLogic	pASIC1 a 3
Vantis	VFI
Xilinx	Spartan, Artix, Kintex y Virtex

Figura 3.72 Fabricantes de Arreglo de Compuertas Programables en Campo FPGAs.

La figura 3.73 muestra las tecnologías que son utilizadas en los dispositivos lógicos programables(PLDs).

Dispositivo (PLD)	Tecnología del elemento programable	Elemento programable	Símbolo	Método de programación	Reprogramable	Método de borrado	Volátil	Fabricantes
SPLD	Bipolar	Fusible en serie con diodo		Eléctrica en corriente	No	-	No	ATMEL, RHOM semiconductor
CPLD	EPROM	Transistor FAMOS		Eléctrica en tensión	Si	Luz ultravioleta	No	ATMEL, Dinies
CPLD	EEPROM	Transistor FLOTOX		Eléctrica en tensión	Si	Eléctrico	No	Microchip, ATMEL, RHOM semiconductor
FPGA	Antifusible	Antifusible		Eléctrica en tensión	No	-	No	Actel, Quicklogic
FPGA	SRAM	Celda SRAM		Eléctrica en tensión	Si	Eléctrico	Si	Altera, Xilinx

Figura 3.73 Resumen de las características de las diferentes tecnologías de programación. [18]

La figura 3.74 muestra los principales fabricantes de PLDs, la familia su arquitectura las macroceldas con las que cuenta y sus características.

	Familia	Arquitectura	Macrocelda	Características
BPLD	GAL	-Tipo PAL (una matriz de conexión)	- Un biestable - Una realimentación	-DIP20 y DIP24 -Borrables eléctric.
	GAL 6001/2	-Matriz AND -Matriz OR	- Dos biestables - Doble realimentación	-DIP24
APLD	MACH1 y 2	-Segmentada -Dos matrices con.	- Un biestable - Doble realimentación	-De 44 a 100 pines -De 32 a 128 Macroceldas
	MACH4	-Segmentada -Tres matrices con.	- Hasta 3 biestables - Hasta 3 realimentac.	-De 44 a 352 pines -De 32 a 512 Macroceldas
	ispLSI5000	-Segmentada -Dos matrices con.	- Un biestable - Doble realimentación	-256 a 512 Macroceldas -12000 a 24000 puertas -Programable en el circuito
	MACH5	-Segmentada (3 matr.) -Dos matrices con.	- Un biestable - Doble realimentación	-128 a 512 Macroceldas -De 68 a 256 pines
	ispLSI1000 ispLSI2000 ispLSI3000	-Dos matrices de con. -Bloques log.complejos	- Un biestable - Una realimentación	-Hasta 448 Macroceldas -Hasta 20000 puertas equiv. -Programables en circuito
CPLD	ispLSI6000	-Dos matrices de con. -Bloques log.complejos -Recursos adicionales: memoria y contadores	- Un biestable - Una realimentación	-8000 puertas PLD -Programable en circuito -Hasta 208 pines
	ispLSI8000	-Tres matrices de con. -Bloques log.=APLDs	- Un biestable - Doble realimentación	-32000 a 60000 puertas -Hasta 1080 Macroceldas

Figura 3.74 Principales fabricantes de PLDs. [19]

3.5 Lenguajes de descripción de hardware (HDL)

Los lenguajes que describen el Hardware son una herramienta muy importante en el diseño, modelado y simulación de circuitos digitales, son diferentes a los lenguajes de programación tradicionales, los lenguajes de descripción de hardware (HDL) describen el comportamiento y la estructura física electrónica (Hardware) de los sistemas digitales, los lenguajes más utilizados son VHDL (Very High Speed Integrated Circuit) y Verilog, los dos permiten simular y sintetizar circuitos con FPGAs o ASICs.

VHDL de origen europeo, maneja una sintaxis detallada y robusta ideal para diseños muy grandes y complejos.

Verilog maneja una sintaxis más simple muy parecida a la del lenguaje C, lo que hace más fácil su manejo y escritura.

Los dos lenguajes permiten trabajar a diferentes niveles de abstracción:

Comportamental, estructural y a nivel de transferencia de registros (RTL)

Estos lenguajes son muy importantes en la ingeniería electrónica actual por permitir realizar pruebas antes de fabricar el circuito, obtener el máximo rendimiento y reducir los errores.

En la actualidad, también se utilizan herramientas como SystemVerilog y Chisel, que aumentan las capacidades de los lenguajes HDL tradicionales para afrontar desafíos modernos en el diseño digital.

Netlist Los primeros lenguajes manejaba un conjunto de instrucciones que describían la interconexión entre los diferentes componentes del diseño del circuito, y generaba una lista de conexiones Netlist.

HDL estos lenguajes solucionan el problema de describir un circuito por sus conexiones ya que lo hacen por su funcionamiento.

Un diseño en VHDL está compuesto por dos partes fundamentales que son:

Entidad (ENTITY)

Arquitectura (ARCHITECTURE)

Entidad (ENTITY).

La entidad define el símbolo del circuito, es decir sus entradas y sus salidas.

La arquitectura define el funcionamiento de la entidad.

Declaración de la entidad ENTITY.

Se requiere diseñar un sumador completo de un bit que tiene tres líneas entradas que son Cin para el acarreo de entrada, A y B como los datos a sumar y tiene dos líneas salidas que son la SUMA y el acarreo de salida Cout como se muestra en la siguiente figura:

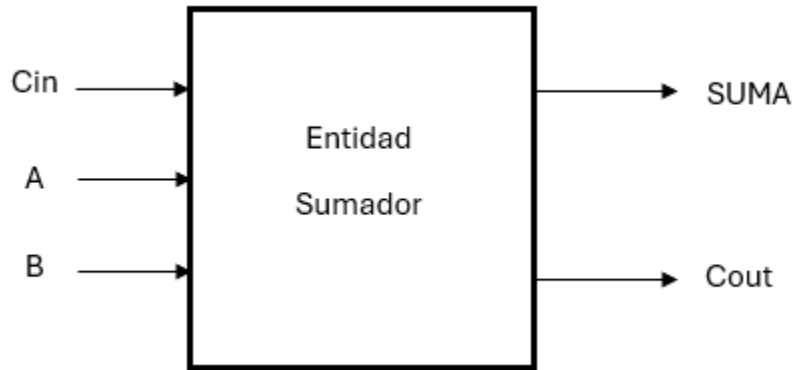


Figura 3.75 Sumador completo de un bit.

```
entity sumador is
    Port (A:    in    std_logic;
          B:    in    std_logic;
          Cin:  in    std_logic;
          Cout: out    std_logic;
          SUMA: out    std_logic );
end sumador;
```

La declaración de una entidad inicia en el primer renglón con la palabra entity seguida del nombre que se le dará a esta terminando la sentencia is.

En el segundo renglón va la palabra Port seguida de un paréntesis, dentro de este se declara el nombre de las entradas y salidas seguidas de dos puntos y el tipo de estas, se cierra el paréntesis seguida de punto y coma y en el último renglón la palabra end seguida del nombre de la entidad y punto y coma.

Las palabras que están en azul son palabras reservadas del lenguaje que utiliza para la declaración de entidades, arquitecturas y tipo de señales de entrada o salida, las que están en color magenta indican el tipo de dato de las entradas y salidas.

Normalmente los lenguajes tienen una plantilla ya establecida para las entidades y arquitectura por lo que es muy fácil tanto el manejo como la comprensión de ellas.

Formato de las entradas y salidas:

Las entradas y salidas tienen tres partes fundamentales que son:

Nombre
Modo
Tipo

A $:$ in STD_LOGIC
 ↑ ↑ ↑
 nombre modo tipo

Nombre

Todos los nombres de entradas y salidas y todos los identificadores que se usen dentro de una descripción de hardware deben cumplir con las siguientes reglas:

- Letras, números y guion bajo.
- El primer carácter debe ser una letra.
- El último carácter no puede ser guion bajo.
- No puede haber dos guiones bajos seguidos.
- No es sensible a minúsculas y mayúsculas.
(Entrada1, ENTRAD1, eNTRADA1 etc.)

Modo

Existen diferentes tipos de modos los más utilizados son:

In:

Señal de entrada, no se le puede asignar ningún valor dentro del programa, no deben aparecer al lado izquierdo de una asignación.

Ejemplo:

```
clk : in std_logic;
```

out:

Señal de salida, no puede leerse es decir no pueden ser usadas a la derecha de una asignación.

Ejemplo:

```
led : out std_logic;
```

Inout:

Señal de entrada/salida bidireccional, puede ser leída y escrita normalmente se utiliza en el manejo de buses, se debe tener cuidado para no genera conflicto con las señales.

Ejemplo:

```
data_bus : inout std_logic_vector(7 downto 0);
```

Buffer:

Señal de salida que puede ser leída internamente, similar al out pero con lectura permitida, actualmente ya casi no es usada.

Ejemplo:

```
count : buffer integer;
```

Tipo

Los tipos más utilizados son:

bit: Toma los valores 0 y 1 lógicos.

bit_vector: Representa un vector de bits.

std_logic: Standar Logic.

dtd_logic_vector: Standar Logic.

boolean: Valor lógico verdadero o falso.

integer: Números enteros(dentro de un rango definido).

character: Un solo carácter.

real: Números reales (no se sintetiza).

Se recomienda que en los diseños se maneje **std_logic** y **std_logic_vector**.

Los valores que puede tomar el Standar Logic son los siguientes:

'0', (Forcing 0): Cero lógico.

'1', (Forcing 1): Uno lógico.

'Z', (High Impedance) Alta impedancia.

'-', (Don'care) No importa.

'X', (Forcing Unknown) Conflicto o valor indefinido.

'U', (Uninitialized) No inicializado, valor desconocido.

'W', (Weak Unknown) Valor débil indefinido.

'L', (Weak 0) Valor bajo débil.

'H', (Weak 1) Valor alto débil.

Los cuatro primeros son los más utilizados para sintetizar.

Declaración de entidades.

Para declarar la entidad para un sumador de dos datos de cuatro bits, con acarreo de entrada y la salida de cuatro bits y el acarreo de salida que se muestra en la figura 3.73.



Figura 3.76 Sumador de cuatro bits.


```

entity sumador_cuatro_b is
    Port (      A:      in  std_logic_vector (3 downto 0);
              B:      in  std_logic_vector (3 downto 0);
              SUMA:    out std_logic_vector (3 downto 0);
              Cin:     in  std_logic ;
              Cout :   out std_logic);
end sumador_cuatro_b;

```

En los vectores (3 **downto** 0), significa que el bit de la izquierda es el más significativo, se puede cambiar el orden.

Librerías y paquetes.

En el lenguaje VHDL, las librerías y paquetes son muy importantes para reutilizar código, organizar proyectos y acceder a los diferentes tipos de datos, funciones, operadores y componentes que fueron previamente definidos, compilados y personalizados.

Las librerías son una colección de recursos que pueden contener paquetes, entidades y arquitecturas. Las librerías permiten organizar el código y acceder a funciones ya definidas con el fin de agilizar el diseño.

Los paquetes son un conjunto de tipos de datos, constantes, funciones, procedimientos y componentes que pueden ser reutilizados en diferentes partes de un diseño.

Ejemplo de librerías más utilizadas.

```

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
use IEEE.numeric_std.all;

```

Para el diseño en VHDL se recomienda iniciar en la parte superior con los comentarios que nos indiquen de qué se trata el circuito a desarrollar, su nombre, la fecha, el autor etc.

Para ello se ponen dos guiones seguidos para que se interpreten como comentarios, en cada renglón de comentario se deben poner los guiones.

En seguida se agregan las librerías que serán utilizadas en el diseño con el formato previamente descrito.

Se agrega la entidad y por último la arquitectura.

Arquitectura (ARCHITECTURE).

La arquitectura es la que describe el comportamiento o estructura interna de una entidad y se hace a través de las sentencias propias del lenguaje VHDL, las arquitecturas más comunes son:

Behavioral: Describe el comportamiento con procesos, condicionales, etc.

Structural: Interconecta múltiples componentes (como un diagrama de bloques).

Dataflow: Usa asignaciones concurrentes para representar el flujo de datos.

Circuito combinacional en VHDL

La salida de un circuito combinacional es sólo función de la o las entradas.

Sentencias básicas de asignación:

(<=) Sentencia de asignación simple.

(with select) Sentencia de selección de señal.

(when else) Sentencia de asignación condicional.

(generate) Generación condicional.

Ejemplo

Compuerta AND.



--Diseño de una compuerta en VHDL

--Compuerta AND

--Librerías

library IEEE;

use IEEE.std_logic_1164.all;

use IEEE.std_logic_arith.all;

use IEEE.std_logic_unsigned.all;

-- Entidad

entity compuerta_and is

Port (entrada_1 : in std_logic ;

entrada_2 : in std_logic ;

salida : out std_logic) ;

end compuerta_and;

--Arquitectura

architecture Behavioral of compuerta_and is

begin

salida <= entrada_1 and entrada_2 ;

end Behavioral;

Señal (signal) en VHDL.

Una señal (signal) en VHDL representa una conexión física entre componentes del hardware. Es una conexión interna que transporta información (niveles lógicos o datos) dentro de un diseño digital.

Las señales sólo se declaran en la arquitectura del diseño de un circuito nunca en la entidad, sirven para expresar la salida de una parte del circuito e introducir su valor como entrada a otra sección del circuito.

Ejemplo

Desarrollar el programa en VHDL para el siguiente circuito:

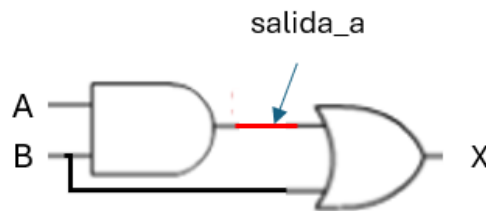


Figura 3.77 Arreglo de compuertas para manejar una señal (signals).

--Arreglo de compuertas para el manejo de una señal.

--Librerías

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
```

-- Entidad

```
entity and_or is
    Port ( A : in std_logic;
          B : in std_logic;
          X : out std_logic );
end and_or;
```

--Arquitectura

```
architecture Behavioral of and_or is
    signal salida_a : std_logic;
begin
    salida_a <= A and B;
    X <= salida_a or B;
end Behavioral;
```

← Aquí se declara la señal.

Sentencia de selección de señal (with select)

Esta sentencia ayuda a trabajar con la tabla de verdad del circuito que se desea diseñar.

Sintaxis:

With variable_seleccion select

```
Salida <= valor_1 when valor_variable1,  
          valor_2 when valor_variable2,  
          valor_3 when others ;
```

Ejemplo

Desarrollar el programa en VHDL para la siguiente tabla de verdad de un multiplexor:

A1	A0	X
0	0	1
0	1	1
1	0	0
1	1	1

Figura 3.78 Tabla de verdad para el multiplexor.

--Programa en VHDL partiendo de una tabla de verdad.

--Librerías

```
library IEEE;  
use IEEE.std_logic_1164.all ;  
use IEEE.std_logic_arith.all ;  
use IEEE.std_logic_unsigned.all ;
```

-- Entidad

```
entity tabla_verdad is  
  Port ( A : in std_logic_vector (1 downto 0);  
        X : out std_logic );  
end tabla_verdad ;
```

--Arquitectura

```
architecture Behavioral of tabla_verdad is  
begin  
  with A select  
    X <= '1' when "00",  
        '1' when "01",  
        '0' when "10",  
        '1' when others;  
end Behavioral;
```

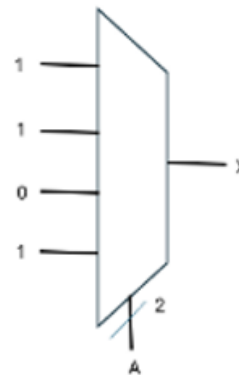


Figura 3.79 Multiplexor de 2 a cuatro.

Ejemplo

Desarrollar una unidad lógica aritmética (ALU) que pueda realizar la suma y resta aritmética y las operaciones AND y OR lógicas con datos de entrada de 8 bits y una salida de 8 bits mostrada en la figura 3.80.

--Programa en VHDL para la ALU.

--Librerías

library IEEE;

use IEEE.std_logic_1164.all;

use IEEE.std_logic_arith.all;

use IEEE.std_logic_unsigned.all;

-- Entidad

entity ALU is

Port (selector : in std_logic_vector (1 downto 0);

dato_1 : in std_logic_vector (7 downto 0);

dato_2 : in std_logic_vector (7 downto 0);

dato_salida : out std_logic_vector (7 downto 0);

end ALU;

--Arquitectura

architecture Behavioral of ALU is

begin

with selector select

dato_salida <= dato_1 + dato_2 when "00",

dato_1 - dato_2 when "01",

dato_1 or dato_2 when "10",

dato_1 and dato_2 when others;

end Behavioral;

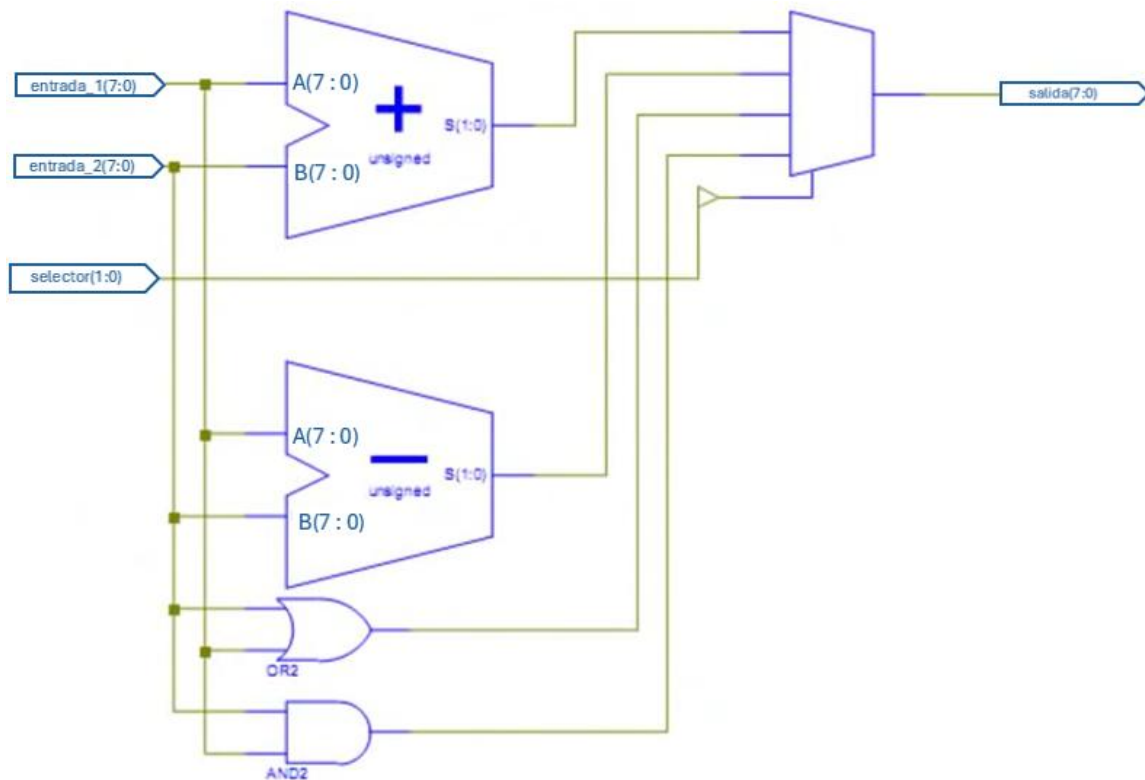


Figura 3.80 síntesis de la unidad lógica aritmética ALU.

Sentencia de asignación condicional (when – else)

Esta sentencia permite realizar las operaciones que la sentencia with – select, pero su implementación es diferente.

La manera que se implementa la sentencia when – else da la oportunidad de diseñar circuitos en la que se requiera manejar la prioridad en la asignación.

Sintaxis

```
Salida <= expresion_1 when condicion_1 else
      expresion_2 when condicion_2 else
      expresion_3 when condicion_3 else
      . . . .
      expresion_n ;
```

los acentos no se usan en VHDL

Funcionamiento:

Se evalúa la condición_1, si es verdadera se asigna:

Salida <= expresión_1

Si condición_1 es falsa entonces se procede a evaluar condición_2.

Si condición_2 es verdadera se asigna:

Salida <= expresión_2:

Si condición_2 es falsa entonces se procede a evaluar condición_3.

Así se continua hasta asignarle a la salida la expresión_n si todas las condiciones son falsas.

Se implementa de la siguiente manera:

salida <= expresion_1 when condicion_1 else expresion_2

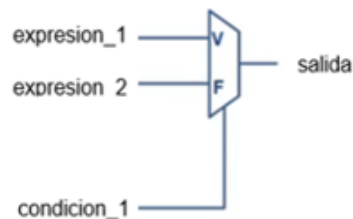


Figura 3.81 when else de una condición.

salida <= expresion_1 when condicion_1 else
expresion_2 when condicion_2 else
expresion_3 ;

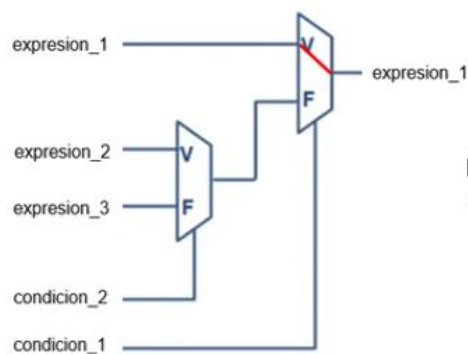


Figura 382 when else de dos condiciones primera verdadera.

Si la condicion_1 es verdadera se asigna a la salida expresión_1 ya no importa si la expresion_2 es verdadera.

Si la condición_1 es falsa y la condición_2 es verdadera se asigna a la salida expresión_2.

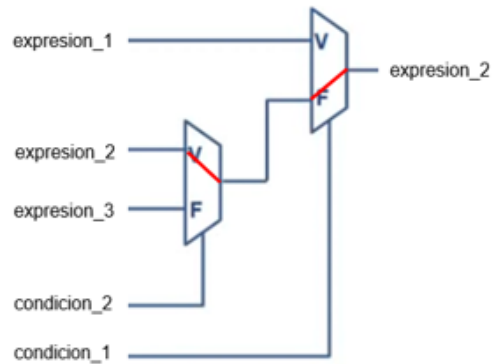


Figura 3.83 when else de dos condiciones, primera falsa y segunda verdadera.

Si la condición_1 es falsa y la condición_2 es falsa, se asigna a la salida expresión_3.

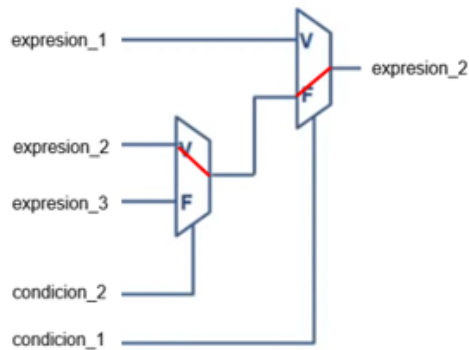


Figura 3.84 when else de dos condiciones, primera falsa y segunda verdadera.

Ejemplo

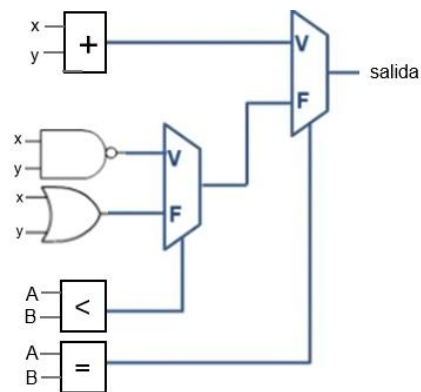


Figura 3.85 Circuito para el manejo de la sentencia when else.

--Programa en VHDL para el mux_w. Con when else.

--Librerías

library IEEE;

use IEEE.std_logic_1164.all;

use IEEE.std_logic_arith.all;

use IEEE.std_logic_unsigned.all;

-- Entidad

entity mux_w is

Port (x : in std_logic ;

y : in std_logic ;

salida : out std_logic) ;

end mux_w;

--Arquitectura

architecture Behavioral of mux_w is

begin

salida <= x + y when A = B else

x nand y when A < B else

A or B ;

end Behavioral;

process

Un proceso (process) en VHDL es un bloque secuencial de código que se ejecuta cuando alguna de las señales en su lista de sensibilidad cambia. Es utilizado para modelar lógica secuencial (como flip-flops, contadores...), escribir condicionales (if, case), usar variables, controlar temporización, como relojes (clk), resets, etc.

Sintaxis de process:

process (lista_de_sensibilidad)

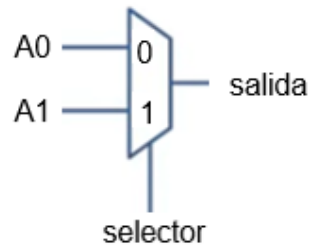
begin

-- Código secuencial

end process;

Ejemplo

Escribir el programa en VHDL con la sentencia process para el siguiente multiplexor:



--Programa en VHDL para el multiplexor usando la sentencia process.

-- Entidad

entity mux is

Port (A0 : in std_logic ;

A1 : in std_logic ;

selector : in std_logic ;

salida : out std_logic) ;

end mux;

--Arquitectura

architecture Behavioral of mux is

p_mux : process (selector, A0,A1)

begin

if selector = '0' then

salida <= A0 ;

else

salida = A1 ;

end if ;

end process ;

end Behavioral;

Sentencia (with-select)

Esta sentencia es utilizada para implementar asignaciones condicionales concurrentes de señales, comúnmente usada para describir multiplexores o decodificadores de manera compacta.

Sintaxis:

```
with <expresión_selección> select  
señal_destino <= valor_1 when condición_1,  
                valor_2 when condición_2,  
                ...  
                valor_default when others;
```

Ejemplo

Desarrollar el programa para el siguiente multiplexor con la sentencia with – select.

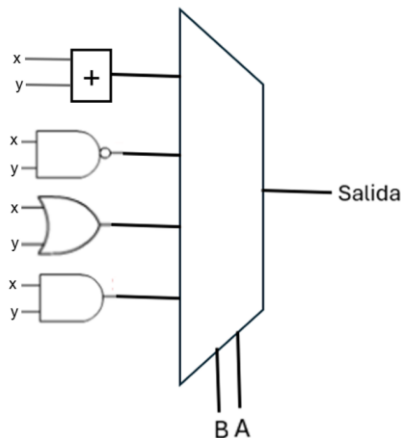


Figura 3.86 Multiplexor para el manejo de la sentencia when else.

--Programa en VHDL para el mux. Con with select.

--Librerías

library IEEE;

use IEEE.std_logic_1164.all;

use IEEE.std_logic_arith.all;

use IEEE.std_logic_unsigned.all;

-- Entidad

```

entity mux is
    Port ( selector : in  std_logic_vector (1 downto 0);
          x : in  std_logic ;
          y : in  std_logic ;
          salida : out std_logic );
    end mux;

--Arquitectura
architecture Behavioral of mux is
begin
    selector <= A & B ;
    with selector select
    salida  <=  A + B    when "00" ,
              A nand B  when "01" ,
              A or  B   when "10" ,
              A  and B  when others ;

```

Software WinCUPL

Este software es un entorno de desarrollo gratuito de la empresa Microchip Technology Inc. (antes ATMEL), que permite escribir y compilar código en el lenguaje CUPL (Compiler for Universal Programmable Logic).

Es utilizado para desarrollar diseños de lógica digital que puede ser implementa en circuitos integrados programables como:

- PLD (Programmable Logic Device)
- CPLD (Complex PLD)
- GAL (Generic Array Logic)
- Dispositivos de la familia ATF de Microchip (como ATF16V8, ATF22V10, etc.)

Las características del software son:

Se puede escribir el diseño en lenguaje CUPL (similar a VHDL o Verilog pero más simple). Con este software se puede simular el comportamiento de la lógica digital diseñada. Proporciona un archivo con extensión .JED (JEDEC) necesarios para programar los circuitos integrados.

Incluye compilador, simulador y herramientas para pruebas lógicas.

Algunos diseños que se pueden hacer con este software son:

Compuertas lógicas personalizadas.

Sistemas digitales simples como contadores, decodificadores, sumadores, circuitos de control etc.

Simular circuitos digitales antes de programarlos en hardware.

El software se puede descargar gratuitamente desde el sitio de Microchip:

<https://www.microchip.com/en-us/tools-resources/archives/atmel-wincupl>

Es importante entender que WinCUPL es un software un poco retro, pero muy útil para iniciar en la programación los dispositivos lógicos programables, se recomienda que una vez que se comprenda la programación CUPL y el manejo de los PLDs básicos se trabaje con software más especializado como Xilinx Vivado (Para FPGAs Xilinx), Intel Quartus Prime (Para FPGAs Intel/Altera) etc.

Instalación del WinCUPL

Ir al sitio oficial <https://www.microchip.com/en-us/tools-resources/archives/atmel-wincupl>

Buscar el botón “Download WinCUPL(Windows)” y darle clic.

Guardar el archivo .exe en la PC.

Ejecutar el archivo como administrador y seguir los pasos del instalador, se recomienda la configuración predeterminada. El instalador copia los archivos en una carpeta que puede ser C:\MCHP\WinCUPL\ o en la ruta que se desee.

WinCUPL puede no funcionar correctamente si se instala en rutas con espacios o caracteres raros. Lo mejor es dejar la ruta por defecto.

Revisar que funcione correctamente, en la carpeta WinCUPL ejecutar el archivo WinCUPL, se deberá abrir una ventana con un menú y opciones del proyecto.

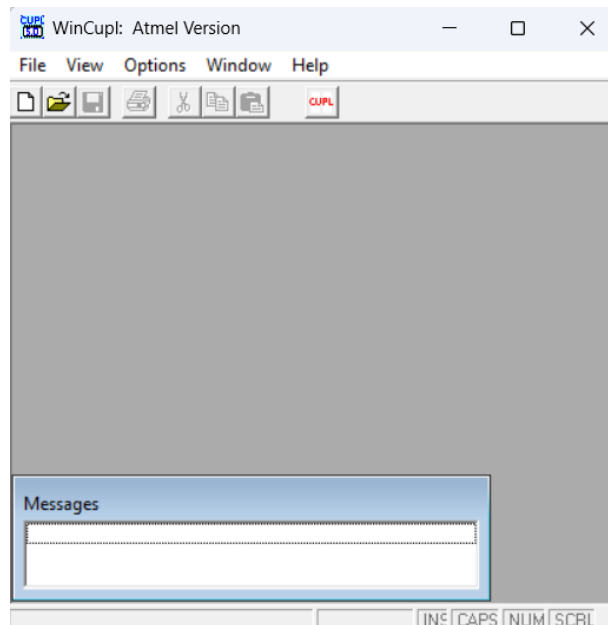


Figura 3.87 Ventana principal de WinCUPL.

Se puede probar un ejemplo incluido en la ruta donde se instaló WinCUPL en la carpeta Examples. C:\MCHP\WinCUPL\Examples\
Abrir un archivo .pld para probar la compilación y simulación.

Ejemplo 1

Diseño con la GAL22V10 el control para el motor de pasos unipolar del ejemplo 2 de la sección procedimiento de diseño cuya tabla de verdad es:

G	Q1	Q0	A	B	C	D
0	0	0	1	0	0	0
0	0	1	0	1	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	1
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

Figura 3.88 Tabla de verdad para el motor a pasos unipolar.

Se ejecuta WinCUPL, se selecciona la ceja File, New, Project.

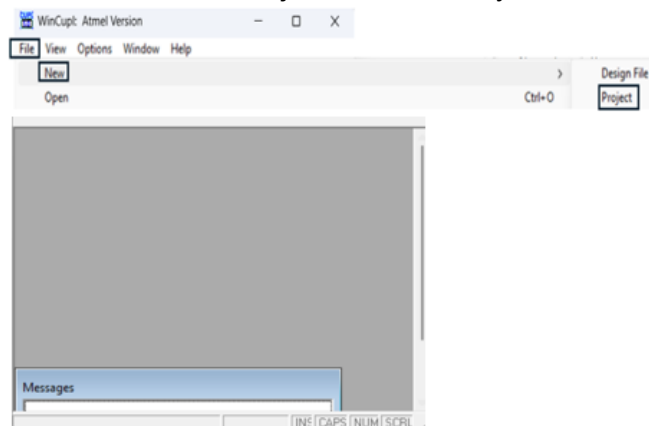
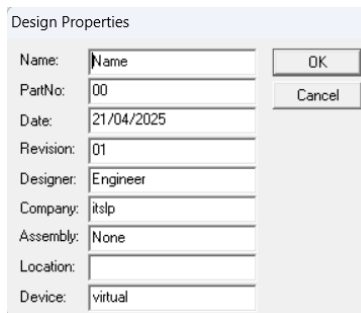


Figura 3.89 Ventana para iniciar un nuevo proyecto.

La nueva ventana permite dar el nombre del proyecto, seleccionar el dispositivo en este caso g22v10 y algunos datos que pueden ser importantes como la fecha del diseño, la revisión, la compañía etc.

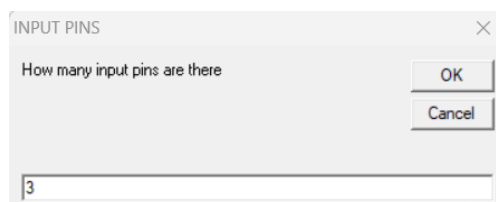


Design Properties

Name:	Name	OK
PartNo:	00	Cancel
Date:	21/04/2025	
Revision:	01	
Designer:	Engineer	
Company:	itslp	
Assembly:	None	
Location:		
Device:	virtual	

Figura 3.90 Ventana para dar nombre al proyecto, fecha, dispositivo a manejar compañía, versión etc.

Después de poner los datos y darle Ok pide el número de entradas que tendrá el diseño. En este caso 3.



INPUT PINS

How many input pins are there

OK

Cancel

3

Figura 3.91 Ventana para dar el número de entradas al proyecto.

Al darle Ok pedirá el número de salidas que tendrá el diseño. Para el motor son 4 una por cada bobina.



OUTPUT PINS

How many output pins are there

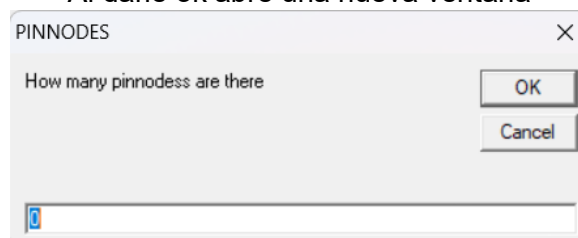
OK

Cancel

4

Figura 3.92 Ventana para dar el número de salidas al proyecto.

Al darle ok abre una nueva ventana



PINNODES

How many pinnodes are there

OK

Cancel

7

Figura 3.93 Ventana para darle el umero total entre entradas y salidas.

Esta venta se deja como esta, solo se le da OK o se le indica cuantos pines fueron definidos como entradas y salidas en este caso serían 7.

Se presentará la ventana donde se definirán los pines de entrada, los de salida y se puede iniciar con el diseño.

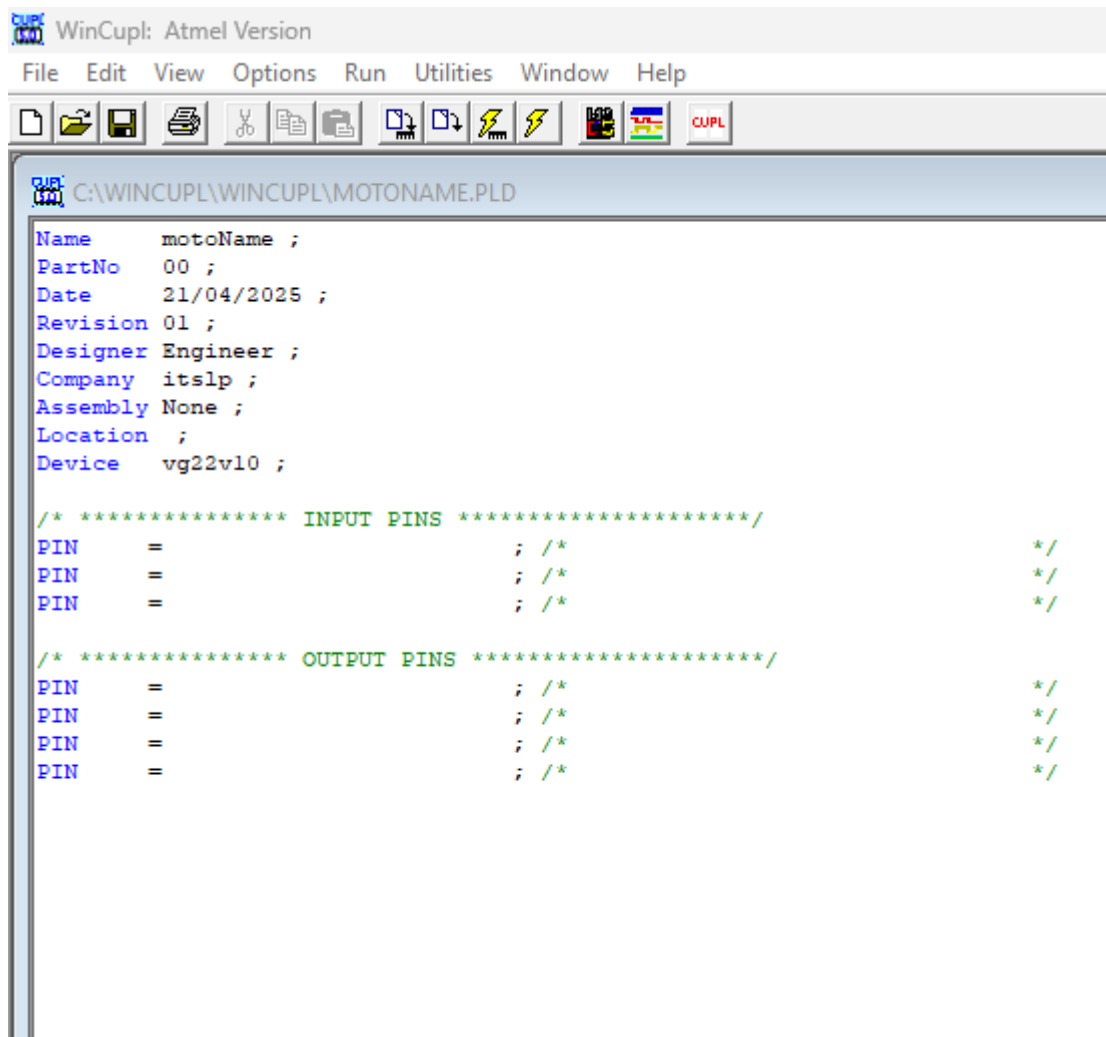


Figura 3.94 Ventana para asignar el número de pin a cada entrada y salida e iniciar el proyecto.

Es importante tener el pdf de la GAL22v10 para poder asignarle el número de pin tanto a las entradas como a las salidas y que correspondan al circuito integrado utilizado.

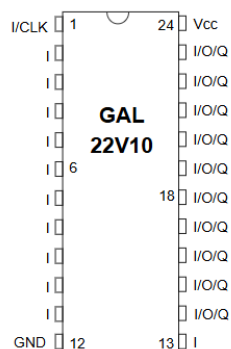


Figura 3.95 Gal 22V10 y sus pines.

La ventana completa es la siguiente

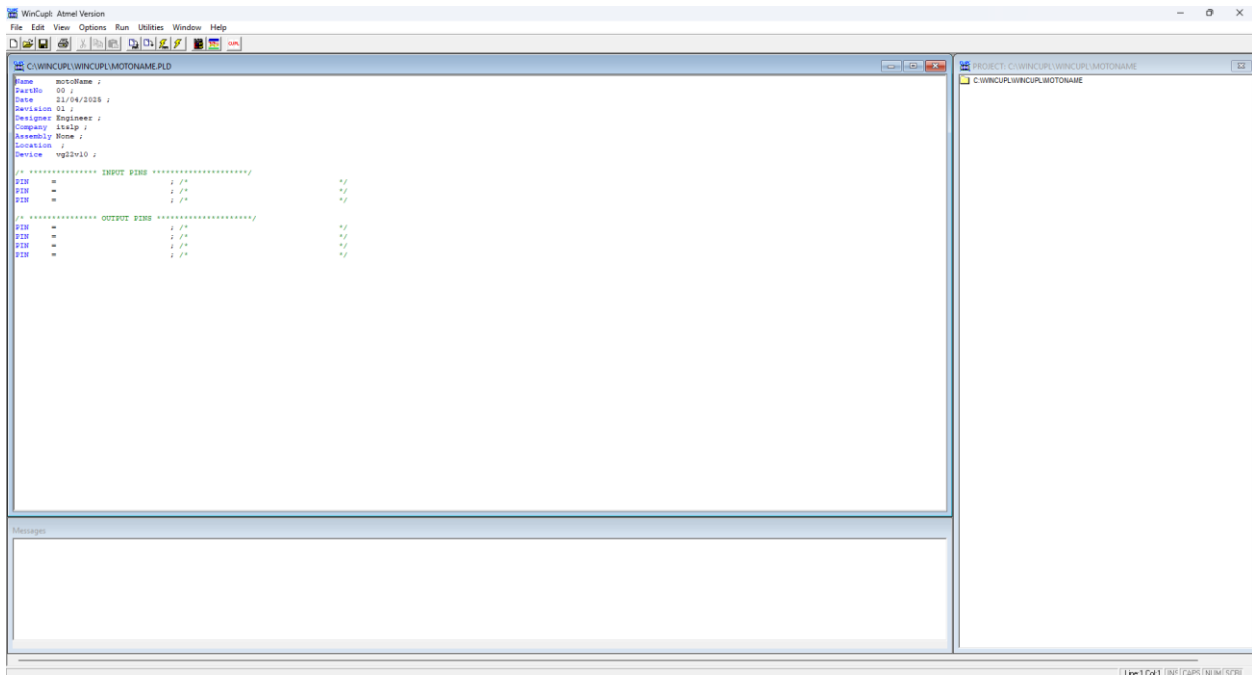


Figura 3.96 Pantalla completa de WinCUPL para trabajar el diseño.

Para asegurar que todo quedó bien configurado y que el proyecto nos generará el archivo .JED con el que se programa la GAL22V10 y algunas otras herramientas se hace lo siguientes:

Abir la ceja Options en ella permite seleccionar el dispositivo Devices con el que se trabajará en este caso la GAL22V10 se le da clic de esta manera no marcará errores al compilar.

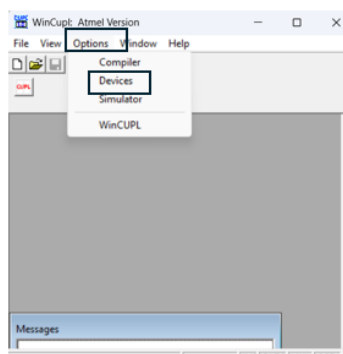


Figura 3.97 Ventana para seleccionar el dispositivo que se desea programar. Se abre la ventana para seleccionar el dispositivo. Una vez seleccionado se le da OK.

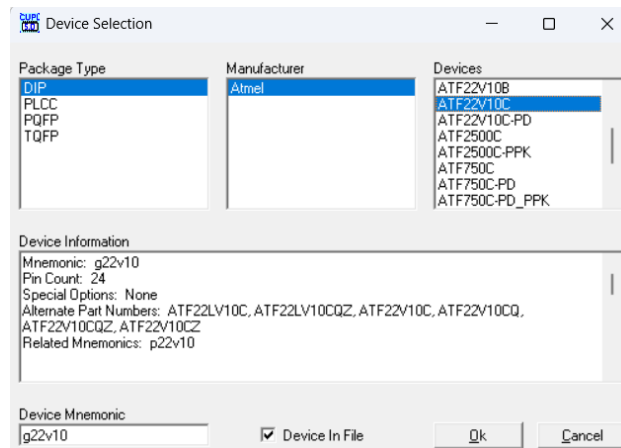


Figura 3.98 Ventana de selección del dispositivo.

Se vuelve a seleccionar la caja Options dentro de ella se selecciona la caja Compiler, para asegurar que este seleccionada la opción JEDEC para que nos genere el archivo que se cargará en la GAL22V10. Las demás opciones que aparecen seleccionadas las dejamos así. Se puede pedir que genere el documento con las ecuaciones, los fusibles utilizados y el dispositivo con el nombre asignado a los pines.(Fuse Plot, Equations, PLA) seleccionando estas opciones.

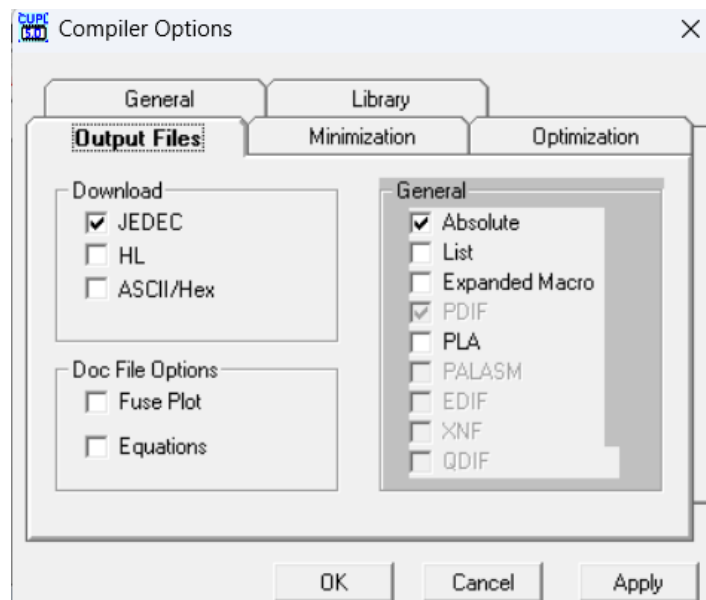


Figura 3.99 Ventana para seleccionar los archivos que se quiere que genere y herramientas de simplificación.

Si se desea simular el circuito se selecciona la caja general y se activa la opción Simulate.

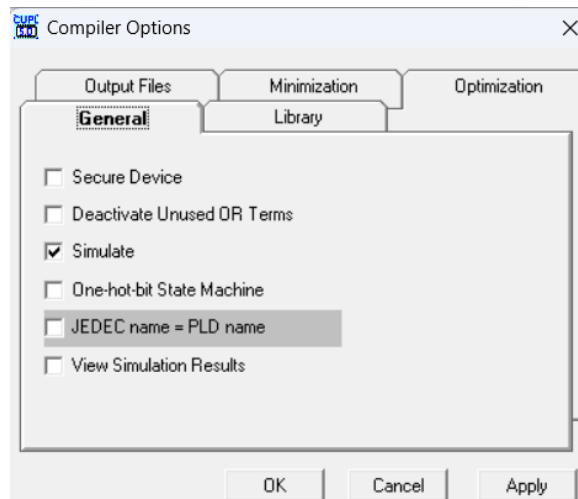


Figura 3.100 Ventana para seleccionar la simulación.

La demás ceja de dejan con las opciones que vienen por default.

Una vez configurado el proyecto, se inicia el diseño, para este caso se usará el método de la tabla de verdad, WinCUPL se encargará de generar las funciones lógicas para cada salida y las cableará internamente.

```

C:\USERS\LFESC\DESKTOP\GAL\EJEMPLOS WINCUPL ENE JUN 2022\MOTOR_PASOS\MOTOR_PASOS.PLD

Name      motor_pasos ;
PartNo    00 ;
Date      21/09/2022 ;
Revision  01 ;
Designer  Engineer ;
Company   ITSLP ;
Assembly  None ;
Location  ;
Device    g22v10 ;

/* ***** INPUT PINS ***** */
PIN 2 = co      ; /* */
PIN 3 = q1      ; /* */
PIN 4 = q0      ; /* */

/* ***** OUTPUT PINS ***** */
PIN 23 = a      ; /* */
PIN 22 = b      ; /* */
PIN 21 = c      ; /* */
PIN 20 = d      ; /* */

TABLE co,q1,q0 => a,b,c,d {
    'b'000 => 'b'1000;
    'b'001 => 'b'0100;
    'b'010 => 'b'0010;
    'b'011 => 'b'0001;
    'b'100 => 'b'0001;
    'b'101 => 'b'0010;
    'b'110 => 'b'0100;
    'b'111 => 'b'1000;
}
```

Figura 3.101 Ventana con los pines asignados y el programa con tabla de verdad.

Formato que debe cumplirse cuando se trabaja con tablas de verdad.

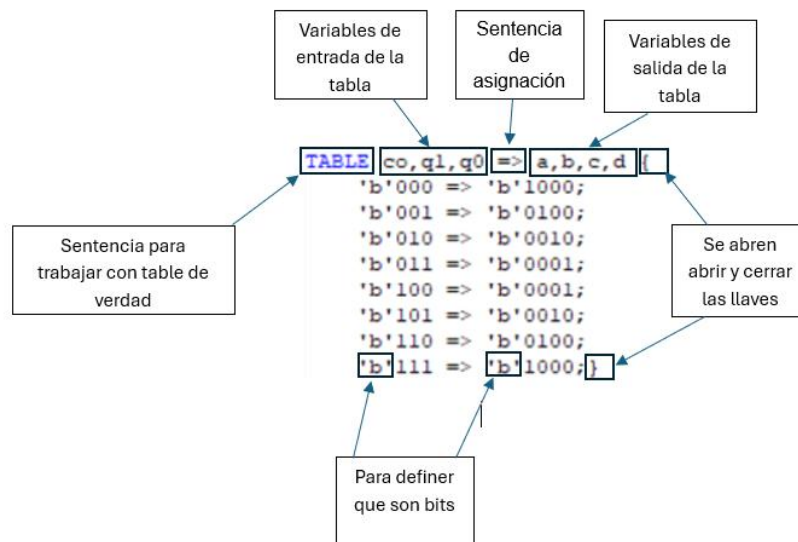


Figura 3.102 Formato para diseñar con tablas de verdad.

Para compilar el programa hay dos opciones:

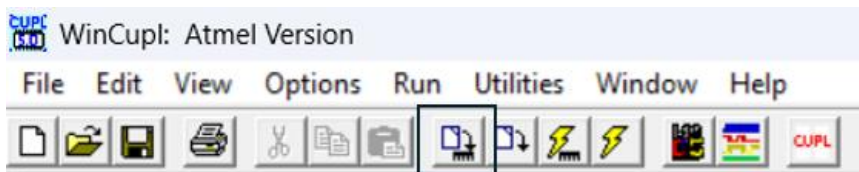


Figura 3.103 Ventana para compilar.

Darle clic al icono donde está el circuito.

La otra forma es dar clic en la ceja Run y después en Device Dependent Compile (F9)

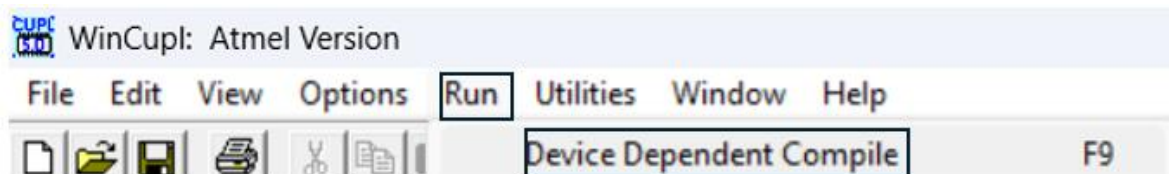


Figura 3.104 Ventana para compilar de otra forma.

Si no hubo problema alguno, se mostrará la siguiente ventana:

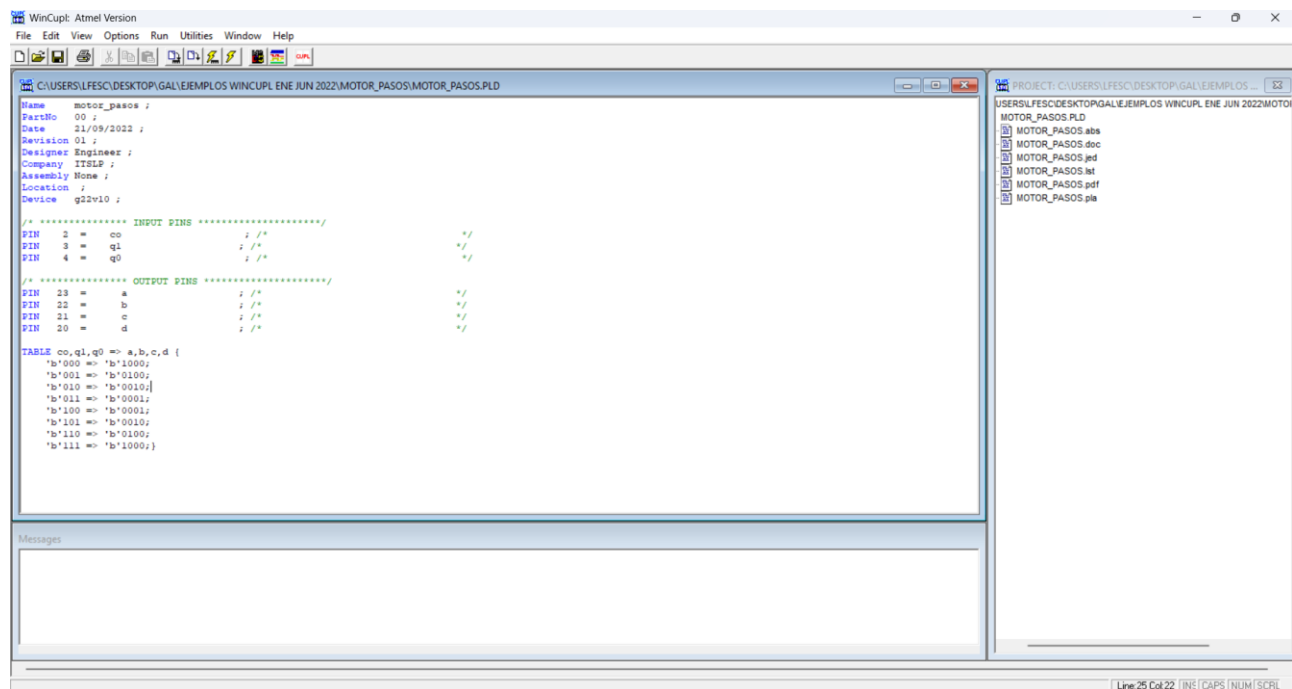


Figura 3.105 Ventana con el programa compilado y los archivos generados.

En la parte izquierda de la pantalla se muestra los archivos generados, debe estar el archivo .JED que es el que se cargará en la GAL22V10, el archivo .PLD es el programa.

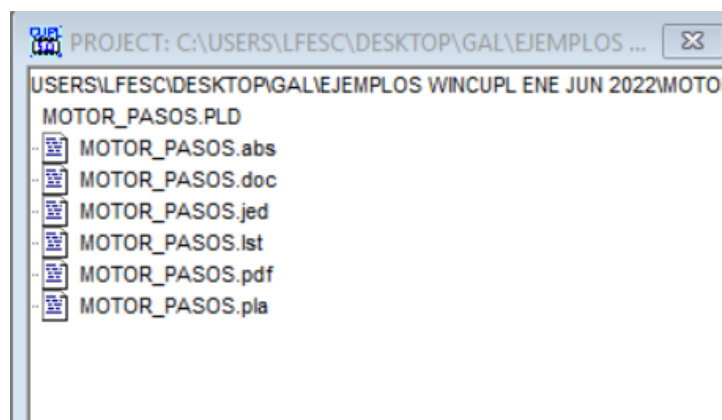


Figura 3.106 Archivos generados al compilar.

El archivo .doc se muestra el encabezado del programa, las funciones lógicas que genero para cada bobina, el circuito GAL22v10 con los pines que fueron utilizados y sus nombres y el Fuse Plot.

Para simularlo en Proteus se arma el circuito se da doble clic en la GAL22V10, se busca el archivo .jed que nos generó WinCUPL. Y se le da ok.

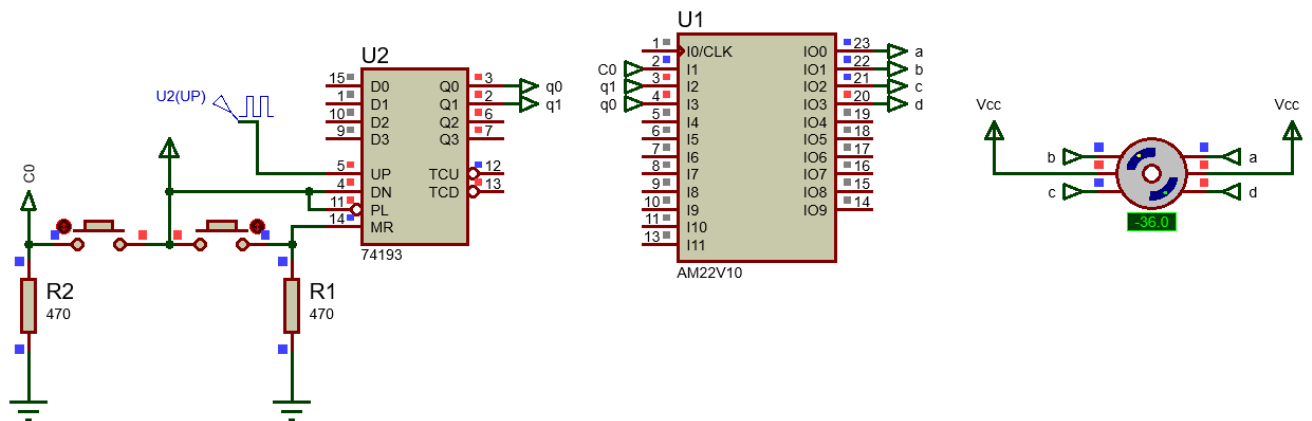


Figura 3.107 Circuito simulado en Proteus.

Figura Simulación en Proteus del control de un motor de pasos unipolar usando la GAL22V10.

Ejemplo 2

Utilizando el software WinCUPL Implemente en la GAL22V10 el control del semáforo diseñado en el apartado “3.2 Circuitos básicos combinacionales” ejemplo 3 de mapas de Karnaugh en la que se obtuvieron las siguientes funciones lógicas:

$$A2 = Q3Q2Q1$$

$$R2 = \overline{Q3}$$

$$V2 = Q3\overline{Q1} + Q3\overline{Q2}$$

$$A1 = \overline{Q3}Q2Q1$$

$$R1 = Q3$$

$$V1 = \overline{Q3}\overline{Q1} + \overline{Q3}\overline{Q2}$$

Agregar al control del semáforo el parpadeo de las fases verdes un instante antes de pasar a preventivo (amarillo).

Para agregar el parpadeo en los colores verdes se revisa la tabla de verdad para ver que combinación de entrada tienen en ese momento y agregar el arreglo necesario para que haga esta función. Se debe notar que se requiere de otra fuente de reloj de más frecuencia para que permita ver el parpadeo en el último instante del verde para ambas fases.

A	B	C	D	V1	R1	A1	V2	R2	A2
0	0	0	0	1	0	0	0	1	0
0	0	0	1	1	0	0	0	1	0
0	0	1	0	1	0	0	0	1	0
0	0	1	1	1	0	0	0	1	0
0	1	0	0	1	0	0	0	1	0
0	1	0	1	1	0	0	0	1	0
0	1	1	0	0	0	1	0	1	0
0	1	1	1	0	0	1	0	1	0
1	0	0	0	0	1	0	1	0	0
1	0	0	1	0	1	0	1	0	0
1	0	1	0	0	1	0	1	0	0
1	0	1	1	0	1	0	1	0	0
1	1	0	0	0	1	0	1	0	0
1	1	0	1	0	1	0	1	0	0
1	1	1	0	0	1	0	0	0	1
1	1	1	1	0	1	0	0	0	1

Figura 3.108 tabla de verdad para el semáforo con las combinaciones de entrada para el parpadeo.

De acuerdo con la tabla de verdad el verde 1 deberá parpadear en la combinación de las variables de entrada 0101 y el verde 2 en la combinación 1101.

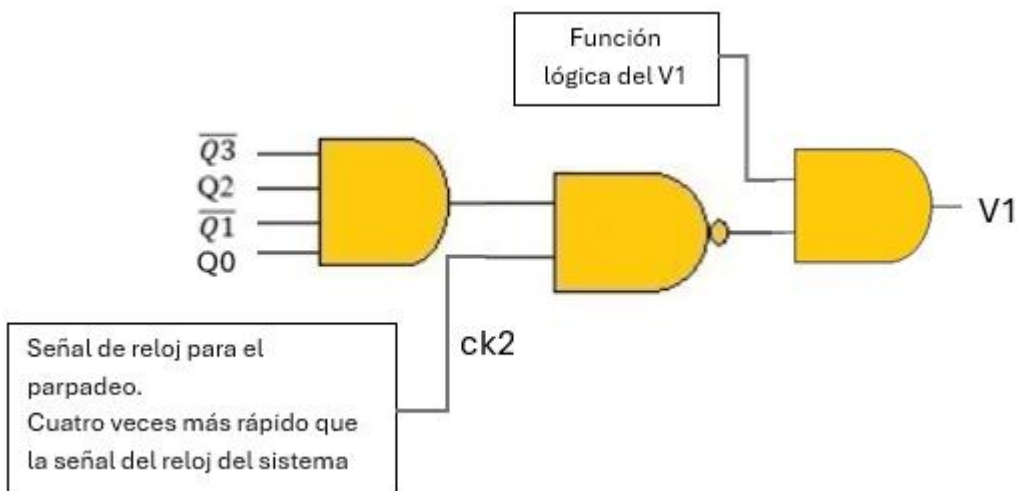


Figura 3.109 Circuito de control de parpadeo para el V1.

La compuerta and de 4 entadas sólo da un 1 en la cuenta número 5 (minitérmino 5)cualquier otra combinación entrega un 0, la compuerta nand sólo dejara pasar la señal de reloj del parpadeo en la cuenta número 5 en cualquier otra cuenta dará un 1, permitiendo que el control del verde 1 funcione correctamente, cuando el control del verde 1 termine su función el verde 1 se apagará. Por lo tanto, el parpadeo sólo se verá en la cuenta 5.

Para el verde 2 sólo cambia la compuerta de 4 entradas para la cuenta número 13 (minitérmino 13), el circuito queda de la siguiente manera:

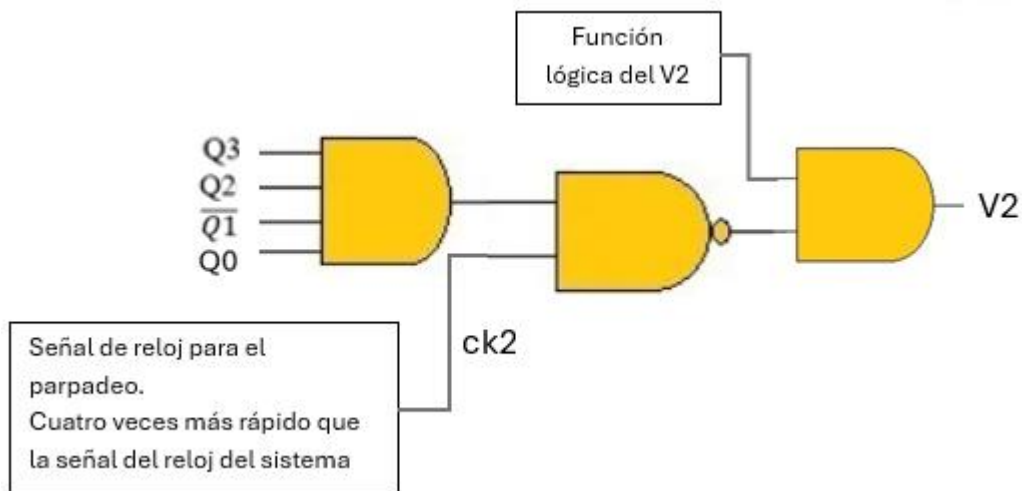


Figura 3.110 Circuito de control de parpadeo para el V2.

Ya teniendo la parte de control que faltaba, se inicia el proyecto en WinCUPL siguiendo el procedimiento descrito anteriormente.

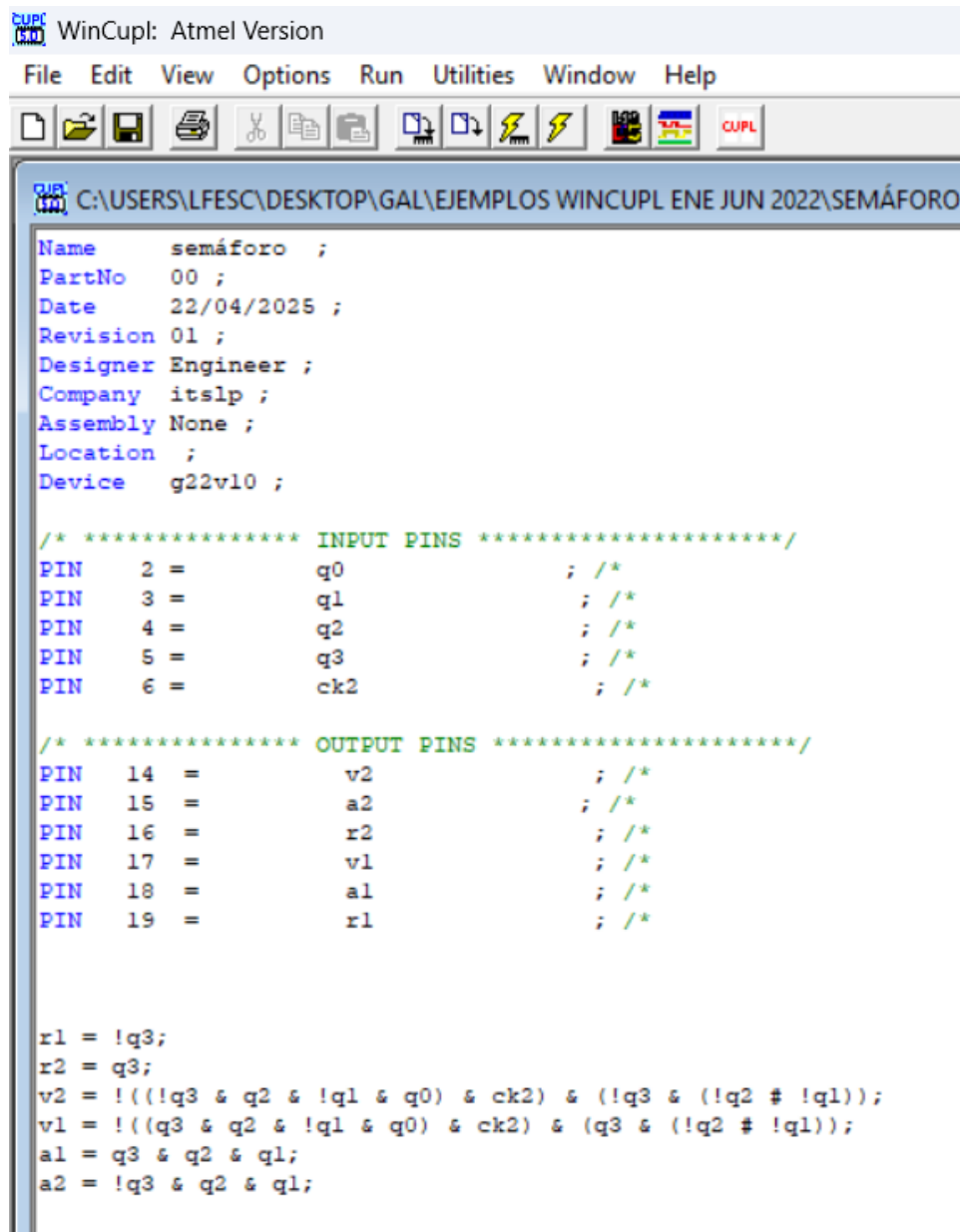
Para este diseño se trabajará con las funciones lógicas para describir la arquitectura del control del semáforo, al asignarle los pines a las señales de entrada y salida se estaría definiendo la entidad del diseño.

WinCUPL utiliza los siguientes símbolos para las operaciones lógicas:

! Operación lógica NOT

& Operación lógica AND

Operación lógica OR



```

WinCupl: Atmel Version
File Edit View Options Run Utilities Window Help

C:\USERS\LFESC\DESKTOP\GAL\EJEMPLOS WINCUPLE ENE JUN 2022\SEMÁFORO

Name      semáforo ;
PartNo    00 ;
Date      22/04/2025 ;
Revision  01 ;
Designer  Engineer ;
Company   itslp ;
Assembly  None ;
Location  ;
Device    g22v10 ;

/* ***** INPUT PINS ***** */
PIN 2 =      q0      ; /*
PIN 3 =      q1      ; /*
PIN 4 =      q2      ; /*
PIN 5 =      q3      ; /*
PIN 6 =      ck2     ; /*

/* ***** OUTPUT PINS ***** */
PIN 14 =      v2      ; /*
PIN 15 =      a2      ; /*
PIN 16 =      r2      ; /*
PIN 17 =      v1      ; /*
PIN 18 =      a1      ; /*
PIN 19 =      r1      ; /*

r1 = !q3;
r2 = q3;
v2 = (!((q3 & q2 & !q1 & q0) & ck2) & (!q3 & (!q2 # !q1)));
v1 = (!((q3 & q2 & !q1 & q0) & ck2) & (q3 & (!q2 # !q1)));
a1 = q3 & q2 & q1;
a2 = !q3 & q2 & q1;

```

Figura 3.111 Ventana de WinCUPL con el programa para el semáforo.

```

r1 = !q3;
r2 = q3;
v2 = !((!q3 & q2 & !q1 & q0) & ck2) & (!q3 & (!q2 # !q1));
v1 = !((q3 & q2 & !q1 & q0) & ck2) & (q3 & (!q2 # !q1));
a1 = q3 & q2 & q1;
a2 = !q3 & q2 & q1;

```

Es muy importante el manejo de los paréntesis para evitar un error en la edición de las funciones lógicas.

Una vez definida las entradas y salidas(entidad) y editado las funciones lógicas(arquitectura) se procede a la compilación.

Si no hay errores se generarán los archivos que se le pidió al definir el proyecto incluyendo el archivo .jed.

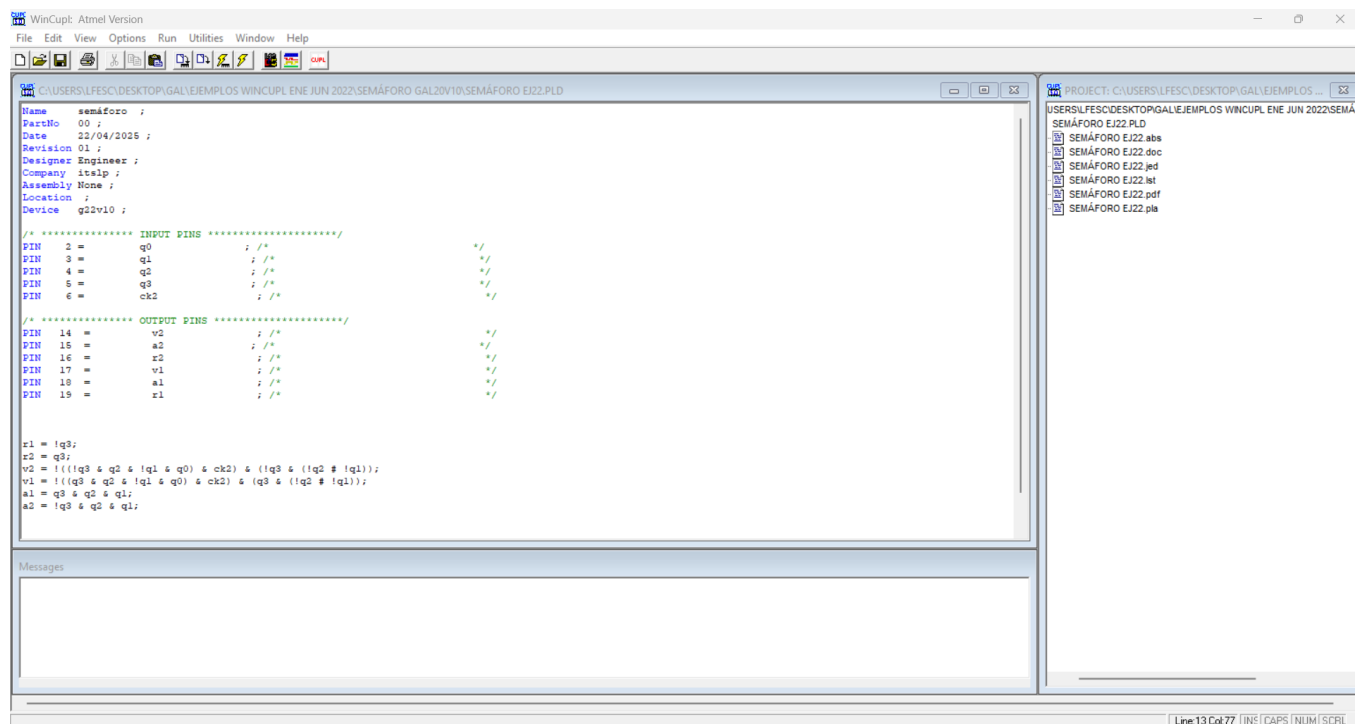


Figura 3.112 Programa compilado sin errores para el control del semáforo.

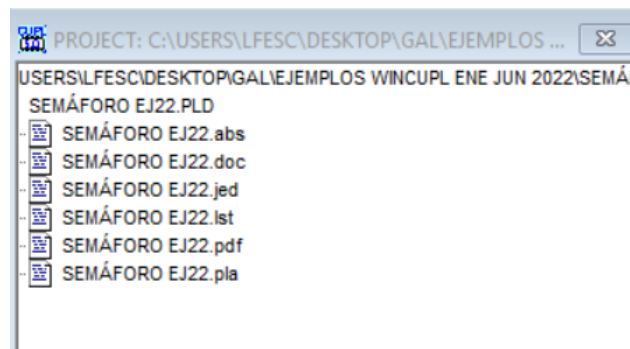


Figura 3.113 Ventana izquierda con los archivos generados.

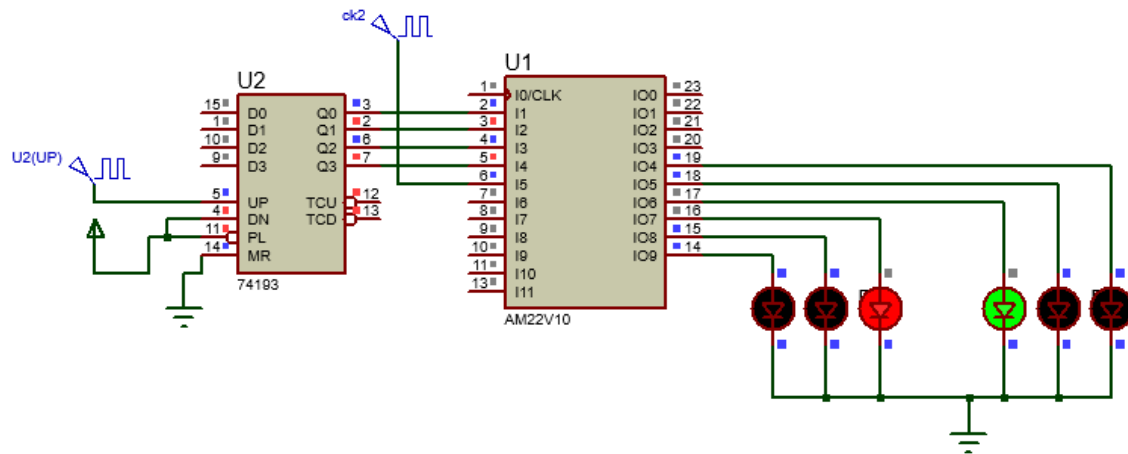


Figura 3.114 Simulación del control del semáforo en Proteus.

Simulación en WinCUPL

Para simular un proyecto que ya se tenga, se selecciona la caja de simulación.

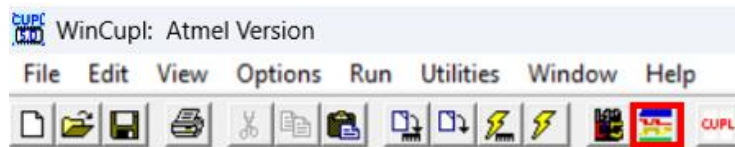


Figura 3.115 Ventana para seleccionar la caja del simulador WinSim.

Nos mostrará la siguiente ventana:

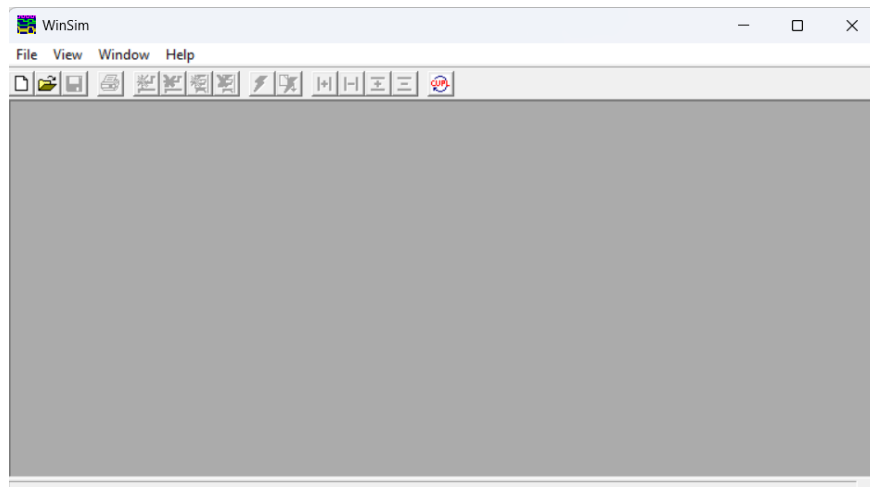


Figura 3.116 Ventana para la simulación de WinCUPL.

Se selecciona la opción File y enseguida la opción New y nos mostrará la ventana para seleccionar el archivo previamente guardado para el semáforo.

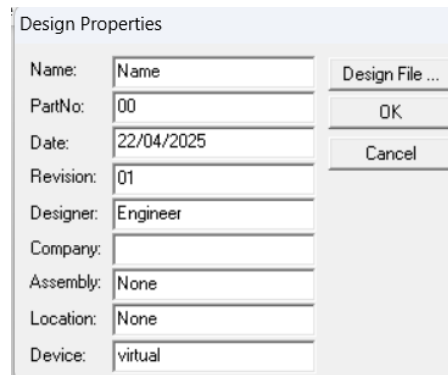


Figura 3.117 Ventana para seleccionar el archivo a simular.

Se selecciona la ceja Design File y pedirá la ruta donde esta guardado el archivo del semáforo .PLD, se selecciona y mostrará la siguiente ventana:

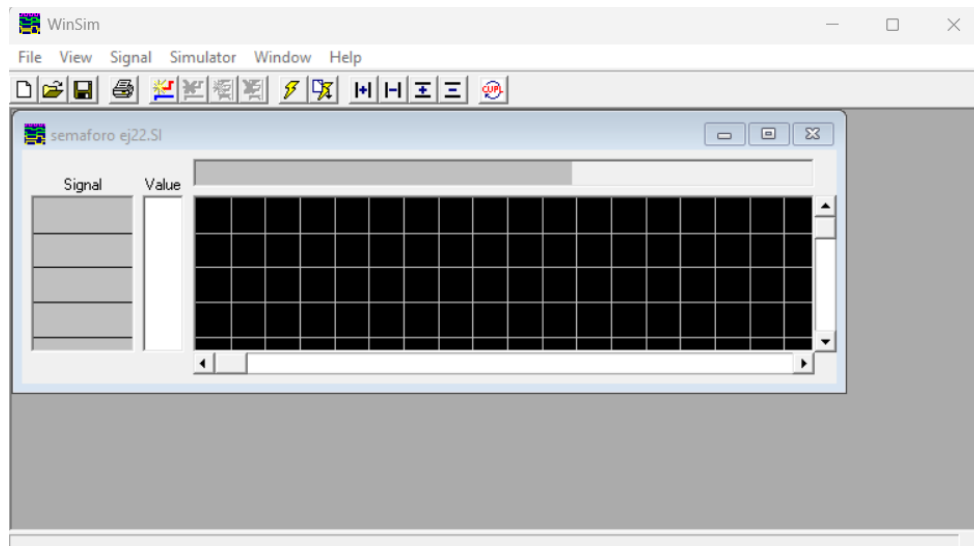


Figura 3.118 Ventana para la simulación del proyecto.

Se selecciona la ceja Signal para agregar las señales de entrada y salida del proyecto, en la ventana que aparecerá se le da ok a cada variable que nos muestre, también se puede elegir si son activas a nivel bajo o alto.

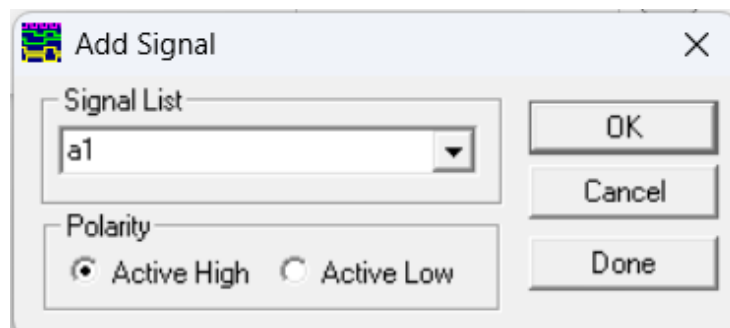


Figura 3.119 Ventana para agregar las entradas y salidas que se van a simular.

La nueva ventana que mostrara es donde se le da los niveles a cada variable para que lo simule.

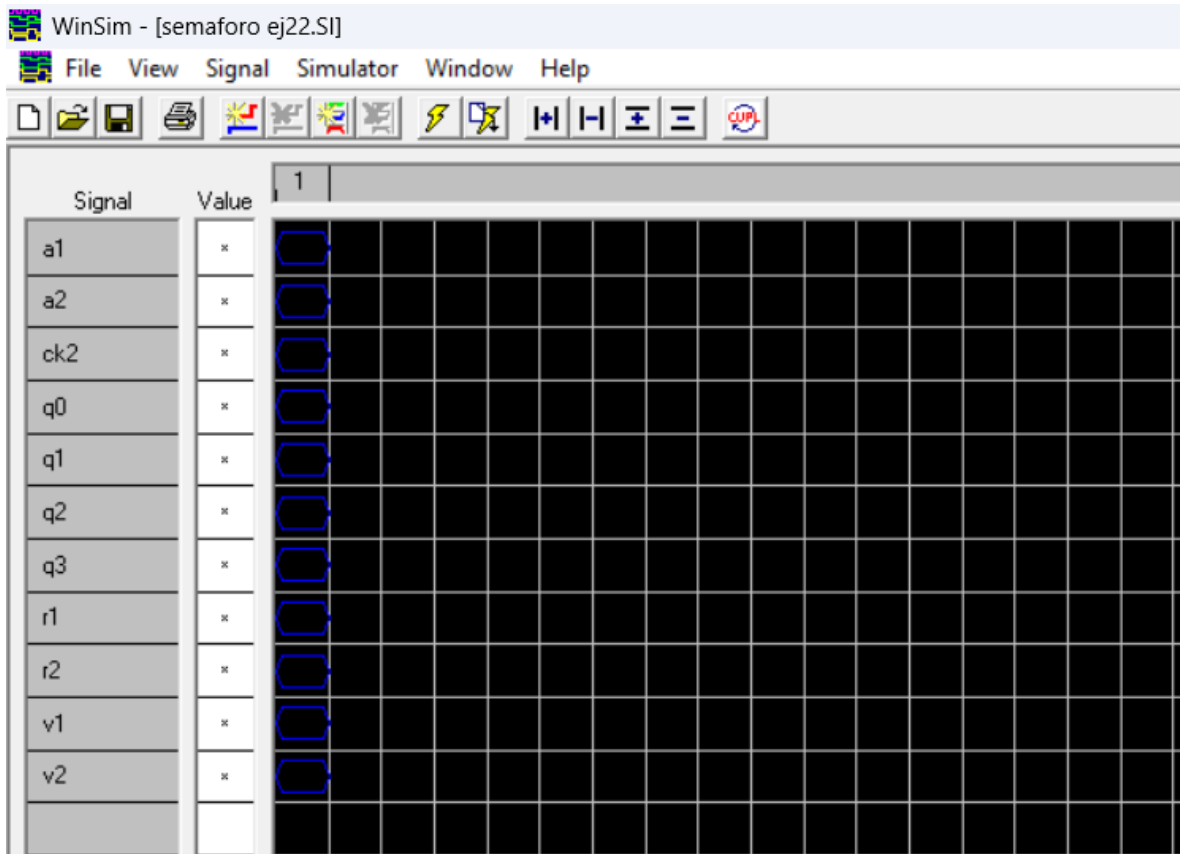


Figura 3.120 Ventana para la simulación con las variables ya seleccionadas.

Cada columna representa un vector si se requiere aumentar el número de estos se selecciona la parte de la ventana donde está el 1 se da clic izquierdo al ratón, luego a la opción Add Vector, y se elige el número de vectores que se requiera, en estos vectores se le da los pulsos de reloj, dando clic en la parte superior de cada cuadro pone la señal en alto, dando clic en la parte baja del cuadro lo pone en bajo.

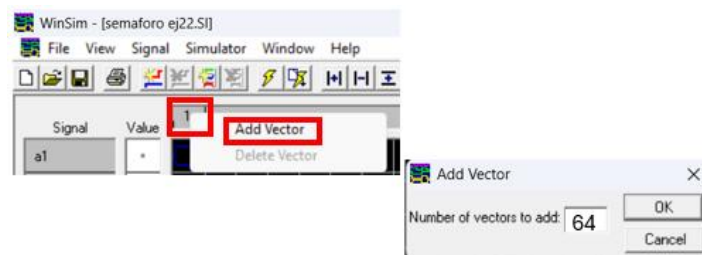


Figura 3.121 Ventana para agregar vectores al simulador.

Se eligieron 64 vectores para cubrir los 16 pulsos de reloj que ocupa el contador y que estos puedan absorber los pulsos del parpadeo de los colores V1 y V2, que son 4 pulsos por uno del reloj del contador.

La ventana que mostrará es la siguiente:

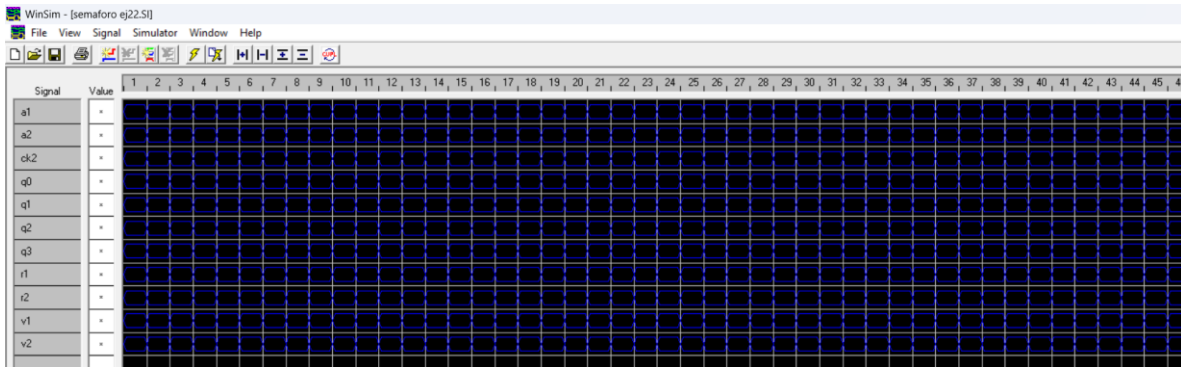


Figura 3.122 Ventana para dar el nivel lógico a cada variable en cada vector.

Para este proyecto se tiene que dar los niveles a las señales de reloj para la simulación, una vez dados estos guardar el archivo y ejecutar la simulación.

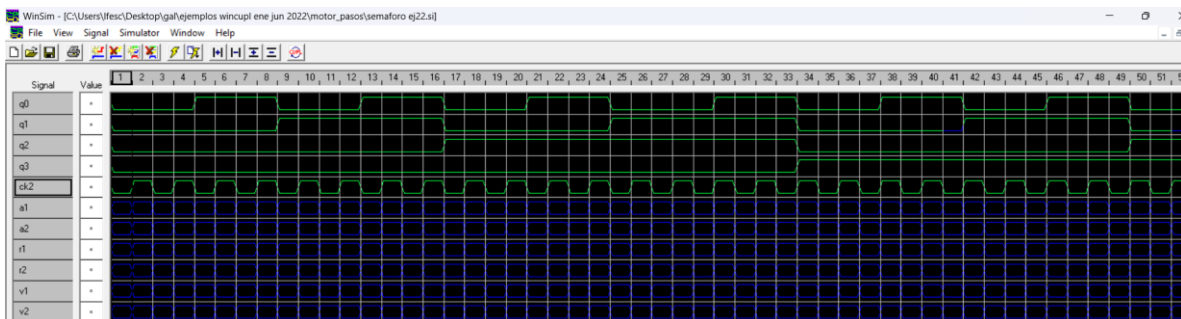


Figura 3.123 Ventana mostrando activadas las señales de entrada y la del ck2.

Trabajo

Realice un resumen de 10 cuartillas que abarque todo el capítulo 8 Familias lógicas de circuitos integrados del libro Sistemas digitales principios y aplicaciones del autor Ronal J. Tocci. [14]

Examen

Selecciona la opción correcta para cada pregunta. Solo hay una respuesta válida por pregunta.

1. ¿Cuál es el primer paso en el procedimiento de diseño de un circuito combinacional?
 - a) Simplificar las funciones booleanas.
 - b) Construir el diagrama lógico.
 - c) Enunciar el problema.
 - d) Seleccionar los circuitos integrados.
2. ¿Qué herramienta se utiliza para simplificar funciones booleanas de hasta 6 variables?
 - a) Álgebra booleana.
 - b) Mapas de Karnaugh.
 - c) Tablas de verdad.
 - d) Diagramas de flujo.
3. ¿Cuál es la función principal de un multiplexor en un sistema digital?
 - a) Convertir señales analógicas en digitales.
 - b) Seleccionar una de varias entradas y dirigirla a una única salida.
 - c) Dividir una señal en múltiples salidas.
 - d) Almacenar datos temporalmente.
4. ¿Qué componente se utiliza para convertir un código BCD en señales para un display de 7 segmentos?
 - a) Multiplexor de 4 a 1.
 - b) Decodificador BCD a 7 segmentos.
 - c) Codificador de prioridad.
 - d) Demultiplexor de 1 a 8.
5. ¿Cuál de los siguientes es un dispositivo lógico programable?
 - a) Transistor.

- b) Resistor.
- c) FPGA.
- d) Diodo.

6. ¿Qué lenguaje se utiliza comúnmente para describir el comportamiento de circuitos digitales?

- a) Python.
- b) VHDL.
- c) Java.
- d) HTML.

7. ¿Cuál es la principal diferencia entre un multiplexor y un demultiplexor?

- a) El multiplexor tiene múltiples salidas y el demultiplexor múltiples entradas.
- b) El multiplexor selecciona una entrada para múltiples salidas, mientras que el demultiplexor selecciona una salida para múltiples entradas.
- c) El multiplexor selecciona una de varias entradas para una única salida, mientras que el demultiplexor dirige una única entrada a una de varias salidas.
- d) No hay diferencia; ambos realizan la misma función.

8. ¿Qué tipo de display se utiliza comúnmente para representar dígitos del 0 al 9?

- a) Display de matriz de puntos.
- b) Display LCD.
- c) Display de 7 segmentos.
- d) Display OLED.

9. ¿Qué característica distingue a los dispositivos lógicos programables de otros componentes electrónicos?

- a) Su capacidad para amplificar señales.
- b) Su capacidad para almacenar datos permanentemente.
- c) Su capacidad para ser configurados por el usuario para realizar funciones lógicas específicas.

d) Su capacidad para convertir señales analógicas en digitales.

10. ¿Cuál de los siguientes lenguajes es utilizado para describir hardware digital?

a) C++.

b) Verilog.

c) JavaScript.

d) Ruby.

11. ¿Qué representa cada celda en un mapa de Karnaugh?

a) Una combinación de entradas y su correspondiente salida.

b) Una puerta lógica específica.

c) Un componente físico del circuito.

d) Una dirección de memoria.

12. ¿Cuál es la ventaja principal de utilizar mapas de Karnaugh en el diseño de circuitos?

a) Reducen el número de variables necesarias.

b) Simplifican visualmente las funciones booleanas.

c) Eliminan la necesidad de compuertas lógicas.

d) Aumentan la complejidad del circuito.

13. ¿Qué componente se utiliza para seleccionar entre múltiples señales de entrada y dirigir una a la salida?

a) Decodificador.

b) Multiplexor.

c) Demultiplexor.

d) Codificador.

14. ¿Cuál es la función principal de un decodificador en un sistema digital?

a) Codificar datos para transmisión.

b) Convertir señales analógicas en digitales.

- c) Convertir códigos binarios en señales únicas de salida.
- d) Multiplexar varias señales en una sola línea.

15. ¿Qué tipo de dispositivo lógico programable permite una configuración más flexible y compleja?

- a) PLA.
- b) PAL.
- c) GAL.
- d) FPGA.

16. ¿Qué significa la sigla HDL en el contexto de diseño de hardware?

- a) High Definition Logic.
- b) Hardware Description Language.
- c) Hybrid Digital Logic.
- d) High-Level Design.

17. ¿Cuál de los siguientes es un ejemplo de lenguaje HDL?

- a) C#.
- b) VHDL.
- c) Python.
- d) SQL.

18. ¿Qué componente se utiliza para convertir una entrada binaria en una señal única de salida?

- a) Multiplexor.
- b) Decodificador.
- c) Codificador.
- d) Demultiplexor.

19. ¿Cuál es la principal ventaja de utilizar dispositivos lógicos programables en el diseño de circuitos?

- a) Reducción de costos de producción.
- b) Mayor consumo de energía.
- c) Flexibilidad para modificar el diseño sin cambiar el hardware físico.
- d) Limitación en la complejidad del circuito.

20. ¿Qué componente se utiliza para mostrar números decimales en un display de 7 segmentos?

- a) Codificador de prioridad.
- b) Decodificador BCD a 7 segmentos.
- c) Multiplexor de 4 a 1.
- d) Demultiplexor de 1 a 8.

21. ¿Cuál es la estructura básica de un diseño en VHDL?

- a) Librería → Arquitectura → Entidad
- b) Librería → Entidad → Arquitectura
- c) Entidad → Arquitectura → Librería
- d) Arquitectura → Entidad → Librería

22. ¿Qué palabra clave se utiliza para declarar una entidad en VHDL?

- a) entity
- b) module
- c) component
- d) architecture

23. ¿Cuál de los siguientes es un tipo de dato escalar en VHDL?

- a) bit_vector
- b) integer
- c) array
- d) record

24. ¿Qué tipo de modelado en VHDL describe el comportamiento del sistema utilizando procesos secuenciales?

- a) Estructural
- b) Conductual
- c) Dataflow
- d) Concurrente

25. ¿Cuál es la diferencia principal entre una señal (signal) y una variable (variable) en VHDL?

- a) Las señales se actualizan inmediatamente; las variables al final del proceso.
- b) Las variables se actualizan inmediatamente; las señales al final del proceso.
- c) No hay diferencia; ambas se comportan igual.
- d) Las señales solo se usan en procesos; las variables fuera de ellos.

26. ¿Qué sentencia se utiliza para seleccionar entre múltiples opciones en VHDL?

- a) if
- b) case
- c) select
- d) choose

27. ¿Cuál de las siguientes opciones representa una asignación condicional en VHDL?

- a) `Y <= A when S='1' else B;`
- b) `Y = A if S='1' else B;`
- c) `if S='1' then Y <= A; else Y <= B; end if;`
- d) `Y := A when S='1' else B;`

28. ¿Qué instrucción se utiliza para crear bucles en VHDL?

- a) loop
- b) for

- c) while
- d) Todas las anteriores

29. ¿Qué palabra clave se utiliza para incluir bibliotecas estándar en VHDL?

- a) include
- b) library
- c) use
- d) import

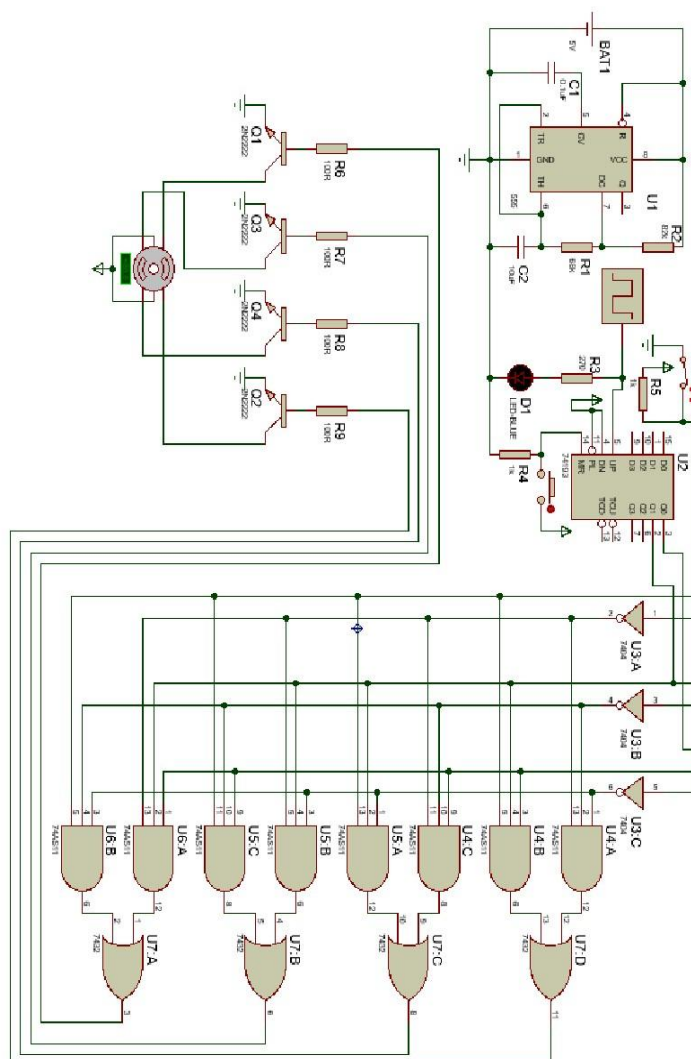
30. ¿Cuál de los siguientes es un tipo de modelado en VHDL que describe cómo los datos fluyen a través del sistema?

- a) Conductual
- b) Estructural
- c) Dataflow
- d) Secuencial

Diseñar por minitérminos el control de giro de un motor a pasos unipolar.

Objetivo

Comprender el diseño de controles lógicos por medio de la técnica de suma de productos(minitérminos).



Práctica

Diseño de un control para un semáforo de dos fases.

Objetivo

Implementar el diseño con compuertas el control del semáforo usando la técnica de minterminos y utilizar algebra booleana para la simplificación de las funciones lógicas.

Material

1 Fuente de 5 Volts

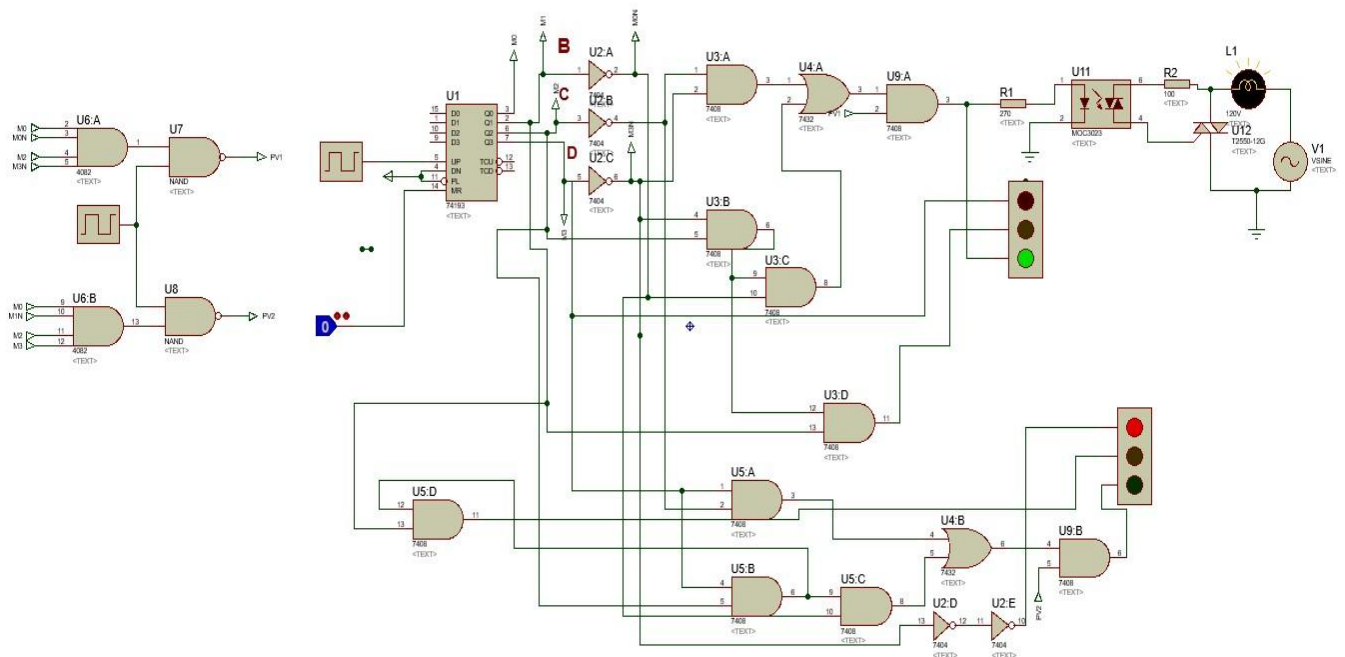
Conjunto de compuertas lógica según el diseño.

1 Optoacoplador MOC 3010.

1 Triac TIC216.

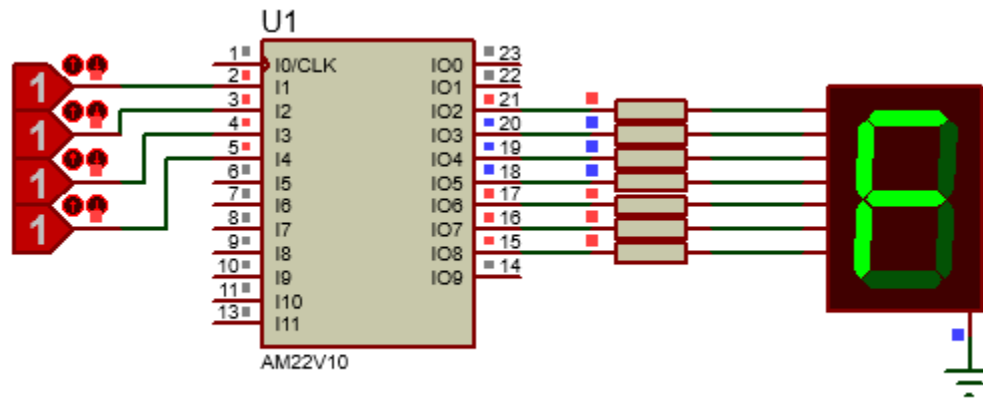
1 Foco de 100 W.

1 Cable con clavija.



Práctica

Diseñar un circuito combinacional que realice la decodificación de binario a hexadecimal y que muestre el resultado en un display de 7 segmentos (puede ser ánodo o cátodo común).
Programe la Gal22V10 con su diseño y compruebe su funcionamiento.



UNIDAD 4

Circuitos secuenciales.

Los circuitos lógicos secuenciales son circuitos lógicos combinacionales que están implementados de tal forma que la respuesta en la salida depende de la combinación de las variables de entrada actuales y de los valores previos de salida, es decir tiene elementos de memoria para registrar la salida y regresarla (retroalimentación) como parte de las entradas, estos elementos de memoria normalmente son flip flops.

Los circuitos secuenciales pueden ser síncronos si tienen una señal de reloj para sincronizar las entradas y las salidas previas y que sólo respondan a la salida cuando llegue esta señal o asíncronos si no tienen señal de reloj, estos responden a la salida inmediatamente con cualquier cambio en las variables de entrada.



Figura 4.1 Circuitos secuenciales síncrono y asíncrono. [20]

Este tipo de circuitos son muy importantes ya que permiten el diseño de dispositivos como son: registros, contadores, máquinas de estados y algunos sistemas más complejos como son los microprocesadores.

Para el análisis y diseño de los circuitos secuenciales se requieren de herramientas tales como: diagramas de estados, tablas de transición y ecuaciones lógicas que determinan su comportamiento dinámico.

4.1 Máquinas de Mealy y Máquinas de Moore.

Máquina de Mealy

La máquina de Mealy es una máquina de estados finitos que se utiliza en circuitos lógicos secuenciales, su comportamiento depende tanto del estado actual como de las entradas presentes.

En la máquina de Mealy las salidas pueden cambiar inmediatamente al cambiar una entrada, lo que permite una respuesta más rápida del sistema.

Esta máquina se representa mediante un conjunto de estados, transiciones entre ellos, una función de transición de estados (que determina a qué estado se va en función del estado actual y la entrada), y una función de salida (que también depende del estado y la entrada).

Por lo anterior dicho, las máquinas de Mealy son más compactas en términos de cantidad de estados requeridos, aunque también más sensibles a las variaciones de las entradas. Las máquinas de Mealy son muy utilizadas para el diseño de sistemas digitales que requieren respuestas inmediatas, como controladores, protocolos de comunicación, y sistemas embebidos.

Una máquina de Mealy se puede diseñar con circuitos secuenciales utilizando flip-flops, compuertas lógicas y, en la mayoría de los casos, una señal de reloj que sincroniza los cambios de estado.

En resumen, esta máquina es una herramienta poderosa para modelar comportamientos secuenciales donde la salida debe reaccionar rápidamente a los cambios en las entradas, lo que la hace ideal para aplicaciones de control en tiempo real.

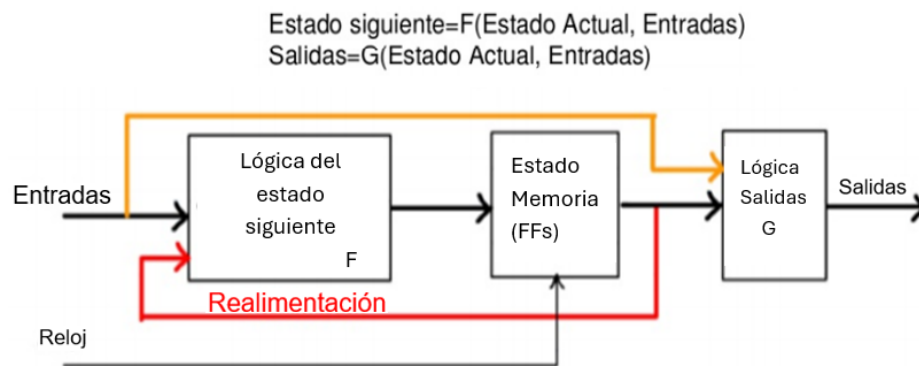


Figura 4.2 Máquina de Mealy.

Máquina de Moore

Una máquina de Moore es una máquina de estados finitos que se usa en el diseño de circuitos lógicos secuenciales. La característica principal de esta máquina es que las salidas dependen únicamente del estado actual del sistema, y no directamente de las entradas. Esto significa que la salida sólo cambia cuando ocurre una transición de estado, y generalmente está sincronizado por una señal de reloj. Esta característica proporciona estabilidad y predictibilidad en las salidas, ya que no responden de forma inmediata a los cambios en las entradas, sino que lo hacen cuando se cambia de estado.

El diseño de una máquina de Moore involucra tres elementos principales: un conjunto finito de estados, una función de transición de estados que determina el siguiente estado en función del estado actual y las entradas, y una función de salida que asigna una salida específica a cada estado. En este tipo de máquina, cada estado tiene asociada una salida fija.

Las máquinas de Moore se utilizan en múltiples aplicaciones, como controladores de procesos, protocolos de comunicación, autómatas digitales y sistemas embebidos. Aunque pueden requerir más estados que las máquinas de Mealy para cumplir una misma función, su diseño suele ser más simple y confiable en entornos donde se busca evitar cambios rápidos o inesperados en las salidas.

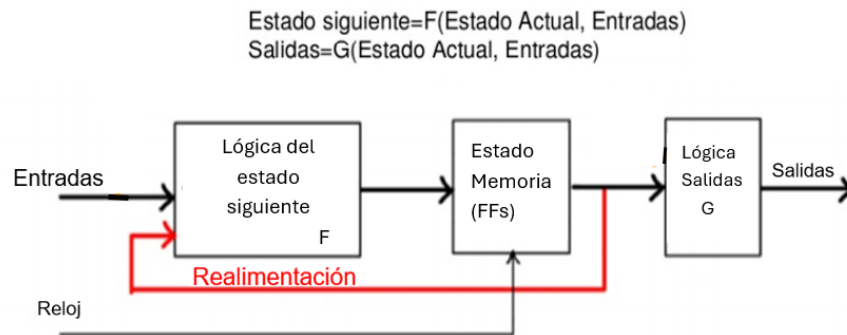


Figura 4.3 Máquina de Moore.

4.2 temporizadores.

Los temporizadores (también conocidos como timers) son dispositivos electrónicos que se utilizan para medir intervalos de tiempo, controlar la duración de eventos, generar retrasos temporales en sistemas digitales y electrónicos.

Un ejemplo de estos temporizadores es el circuito integrado LM555, es compatible con los circuitos TTL ya que tiene un rango de alimentación que va de 4.5v a 15v para el LM555 estándar y de 2v a 18V para el circuito integrado CMOS ICM7555.

Sus principales características son:

Temporización desde Microsegundos hasta Horas.

Opera en modos astable y monoestable.

Ciclo de trabajo ajustable.

La salida puede suministrar o absorber 200 mA.

Salida y Suministro Compatible con TTL.

Estabilidad de Temperatura Mejor que 0.005% por °C.

Salida Normalmente Encendida y Normalmente Apagada.

Aplicaciones

Generación de Pulsos.

Temporización Secuencial.

Generación de Retardo de Tiempo.

Modulación por Ancho de Pulso.

Modulación por Posición de Pulso.

Generador de Rampas Lineales.

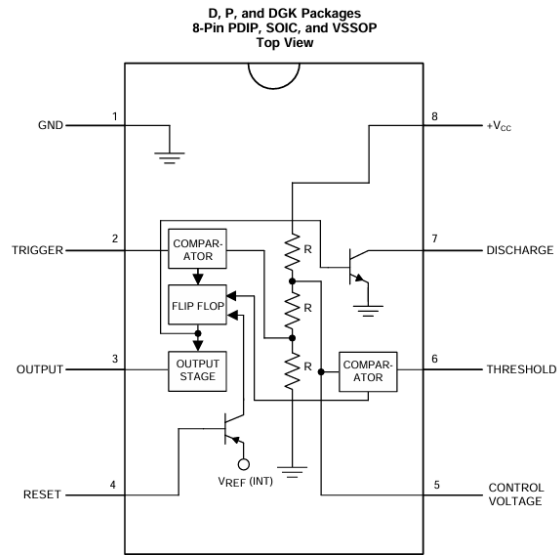


Figura 4.4 Circuito integrado LM555. [21]

PIN NO.	NAME	I/O	DESCRIPTION
5	Control Voltage	I	Controls the threshold and trigger levels. It determines the pulse width of the output waveform. An external voltage applied to this pin can also be used to modulate the output waveform
7	Discharge	I	Open collector output which discharges a capacitor between intervals (in phase with output). It toggles the output from high to low when voltage reaches 2/3 of the supply voltage
1	GND	O	Ground reference voltage
3	Output	O	Output driven waveform
4	Reset	I	Negative pulse applied to this pin to disable or reset the timer. When not used for reset purposes, it should be connected to VCC to avoid false triggering
6	Threshold	I	Compares the voltage applied to the terminal with a reference voltage of 2/3 V _{CC} . The amplitude of voltage applied to this terminal is responsible for the set state of the flip-flop
2	Trigger	I	Responsible for transition of the flip-flop from set to reset. The output of the timer depends on the amplitude of the external trigger pulse applied to this pin
8	V ⁺	I	Supply voltage with respect to GND

Figura 4.5 Descripción de terminales del LM555. [21]

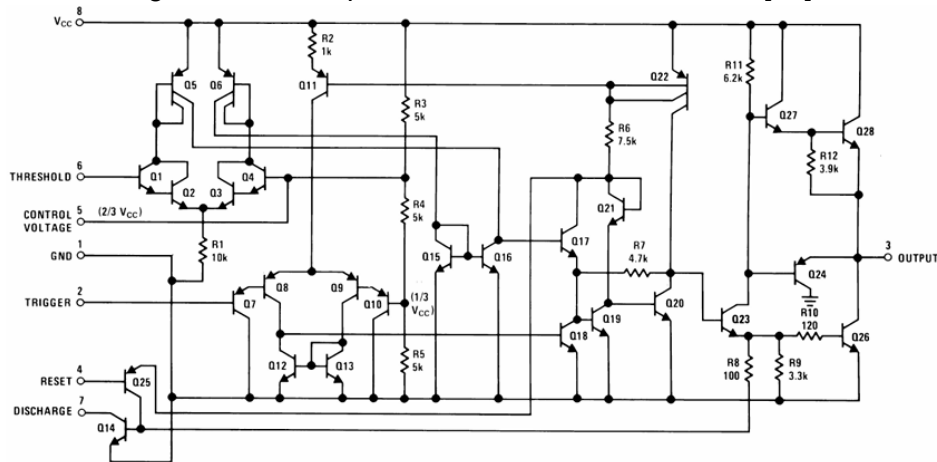


Figura 4.6 Diagrama esquemático del LM555. [21]

4.2.1 LM555 modo monoestable.

En esta configuración del 555 la salida se mantiene a nivel bajo, hasta que se le da un pulso de disparo (trigger) lo que hace que la salida (terminal 3) pase a nivel alto aproximadamente al valor de voltaje V_{cc} y se mantendrá en este estado por el tiempo calculado por el usuario con la ecuación $T = 1.1RC$. Es importante entender que esta configuración se considera redispensible, es decir, si antes de que termine el tiempo de nivel alto que se calculó se le da otro disparo a la entrada, el tiempo que había transcurrido ya no cuenta, inicia en ese momento sin que la señal de salida se regrese a nivel bajo.

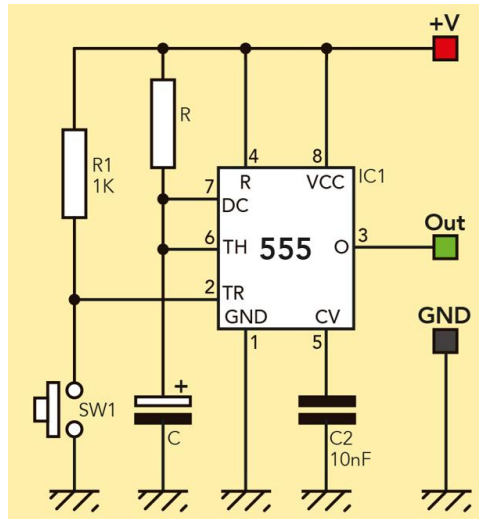


Figura 4.7 Circuito para la configuración monoestable.

Se debe asegurar que el valor de la resistencia R_1 conectada al switch de disparo tenga un valor entre $1K\Omega$ y $3.3M\Omega$ y el capacitor C_2 mayor a 500 pF , lo ideal es de 10 nF

Ejemplo

Se requiere que la salida 3 se active al presionar sw_1 y se mantenga en ese estado por 1 segundo.

Como $T = 1.1RC$

$1\text{ seg} = 1.1RC$

Si se propone un capacitor de $10\mu\text{F}$

Entonces $R = 1/(1.1 \times 10\mu\text{F})$

$R = 90.909\text{ K}\Omega$

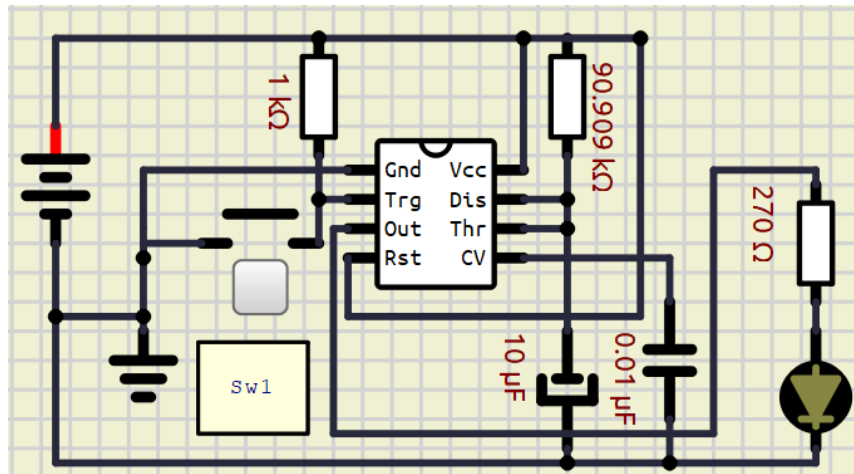


Figura 4.8 Simulación del 555 como monoestable en SimulIDE.

4.2.2 LM555 Modo astable.

A este modo de operación del 555 también se le conoce como temporizador de carrera libre ya que una vez configurado como se muestra en la figura 4.4 sólo requiere la alimentación para generar una onda rectangular de salida con dos niveles de voltaje, para el nivel alto V_H es aproximadamente el valor de alimentación del circuito V_{cc} y para el nivel bajo V_L es de 0V.

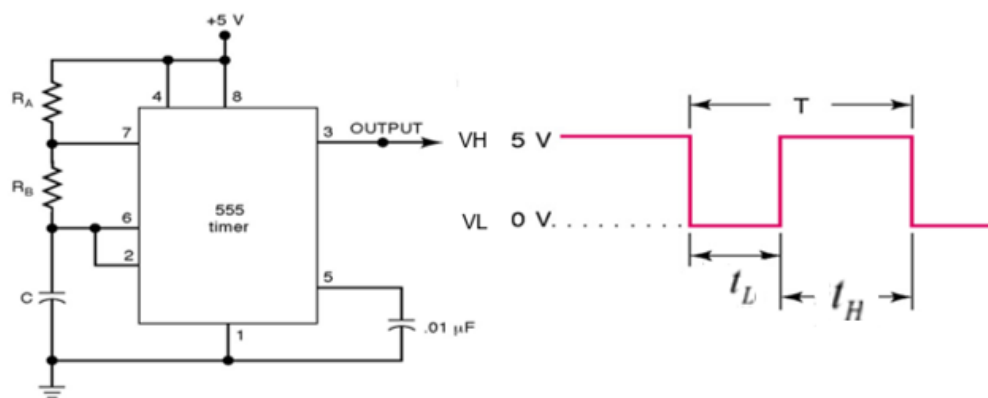


Figura 4.9 LM555 configurado como astable y su señal de salida.

El tiempo que dura la señal de salida a nivel alto viene dada por la ecuación:

$$T_H = 0.693(R_A + R_B)C$$

El tiempo para la señal de salida a nivel bajo viene dado por la ecuación:

$$T_L = 0.693 R_B C$$

El periodo de la señal es la suma de estos dos tiempos:

$$T = T_H + T_L$$

La frecuencia de la señal es:

$$F = \frac{1}{T}$$

El ciclo de trabajo se define como el tiempo de la señal de salida a nivel alto entre el periodo de la señal y viene dado por la siguiente ecuación:

$$\text{Ciclo de trabajo, } D = \frac{T_H}{T}$$

De acuerdo con las ecuaciones anteriores se puede observar que con esta configuración del 555 no se logra obtener un ciclo de trabajo de 50% ya que el T_H siempre es mayor que T_L , podría aproximarse si se hace $R_B \gg R_A$ (manteniendo R_A mayor a $1K\Omega$).

Ejemplo

Calcular los valores de resistencia para generar una señal de 1Hz que tenga un nivel alto del 80% del período y 20% de a nivel bajo como se muestra en la siguiente figura:

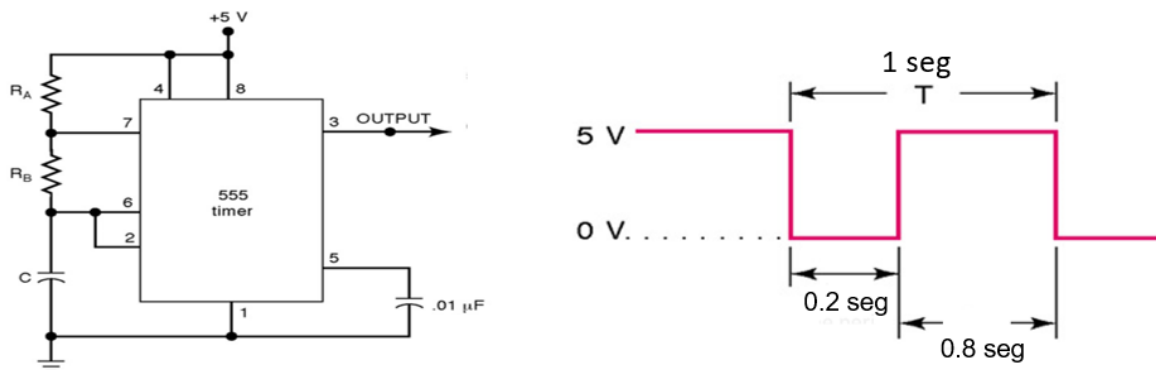


Figura 4.10 Temporizador astable.

Se inicia con la ecuación para T_L ya que en ella hay sólo dos de los elementos que utiliza esta configuración R_B y C .

Se propone el valor del condensador con un valor comercial, para este caso será de $10\mu F$.

$$T_L = 0.693 R_B C$$

$$0.2 \text{ seg} = 0.693 * R_B * 10 \mu F$$

Despejando R_B

$$R_B = 28.86 K\Omega$$

Se puede ajustar el valor de esta resistencia a un valor comercial más próximo, puede ser de $27K\Omega$, el tiempo se modifica de la siguiente manera:

$$T_L = 0.693 * 27K\Omega * 10 \mu F$$

$$T_L = 0.187\text{seg}$$

Ahora se trabaja con la ecuación para el tiempo alto T_H donde se despeja el valor de la resistencia R_A .

$$T_H = 0.693 * (R_A + R_B) * C$$

$$0.8 = 0.693 * (R_A + 27K\Omega) * 10 \mu F$$

Despejando R_A

$$R_A = 88.44 K\Omega$$

Ajustando al valor comercial más cercano que es $82K\Omega$

$$T_H = 0.693 * (82K\Omega + 27K\Omega) * 10 \mu F$$

$$T_H = 0.755\text{seg.}$$

El periodo que se obtiene con estos valores comerciales ajustados es:

$$T = 0.187\text{seg} + 0.755\text{seg}$$

$$T = 0.942 \text{ seg.}$$

$$F = 1.061 \text{ Hz. Valor muy cercano al pedido.}$$

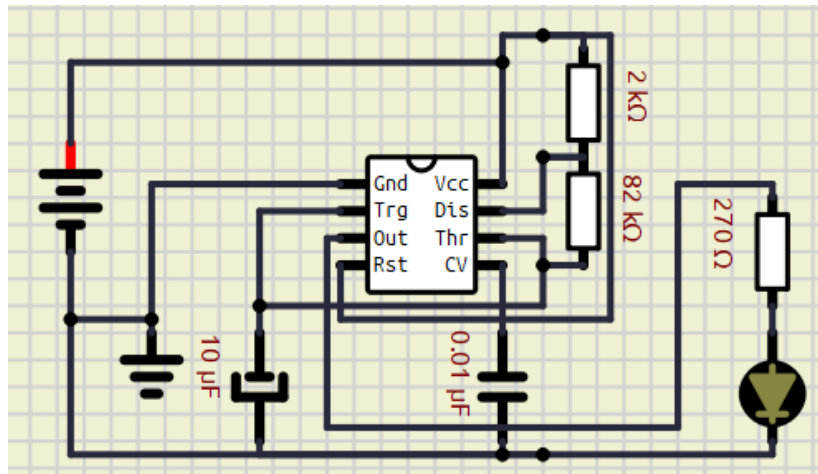


Figura 4.11 Simulación del 555 como astable en SimulIDE.

Para poder obtener señales con ciclo de trabajo menor al 50% se agregan dos diodos con el fin de permitir que el capacitor se cargue a través de R_A y que se descargue sólo a través de R_B , esto se muestra en la siguiente figura:

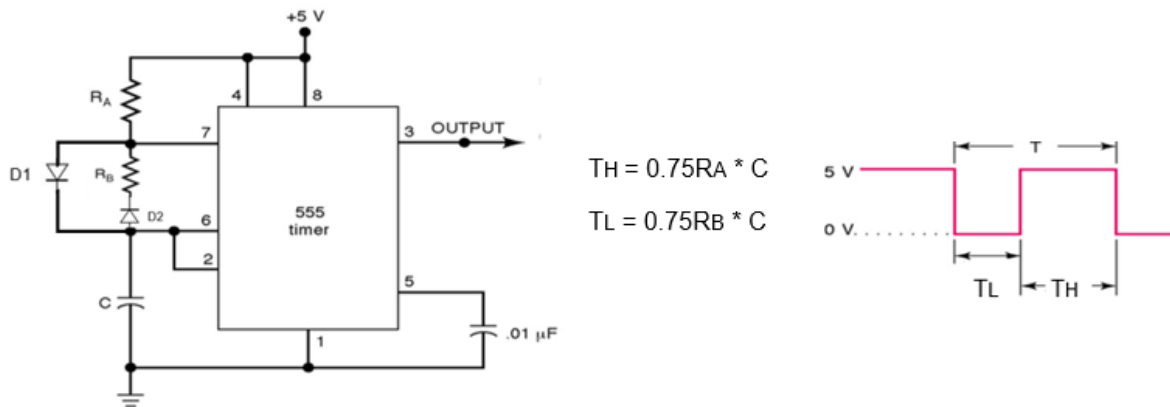


Figura 4.12 LM555 configurado para ciclos de trabajo menores del 50%.

La constante 0.75 es únicamente para la alimentación de 5V.

Ejemplo

Calcular los valores de los elementos R_A , R_B y C para generar una señal de 2 KHz con un ciclo de trabajo de 30%.

El período es el inverso de la frecuencia, entonces tenemos que $T = 0.5 \text{ mseg}$.

Por lo tanto, $T_H = 0.30(0.5\text{mseg}) = 0.15 \text{ mseg}$.

Y $T_L = 0.70(0.5\text{mseg}) = 0.35 \text{ mseg}$.

$T_H = 0.15\text{mseg} = 0.75 * R_A * C$

Si se propone un capacitor de $0.1\mu\text{F}$

$R_A = 2000 \Omega$

Para T_L

$T_L = 0.35\text{mseg} = 0.75 * R_B * 0.1\mu\text{F}$

$R_B = 4666.6 \Omega$

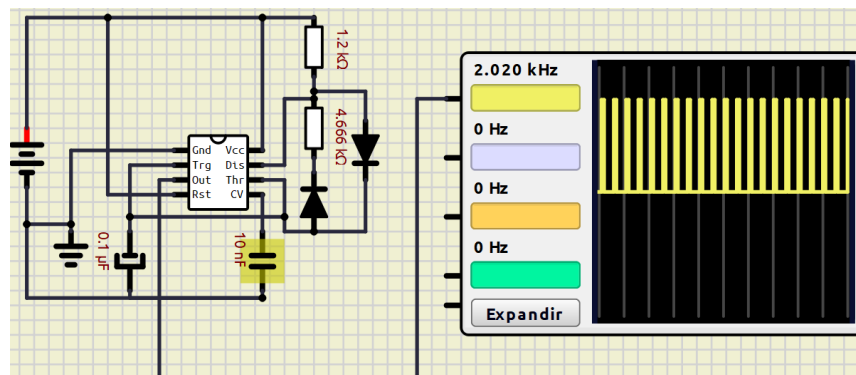


Figura 4.13 Simulación para ciclo de trabajo menor de 50% en SimulIDE.

4.3 FLIP FLOPS

Un flip flop (FF) es la unidad básica de memoria capaz de almacenar un 1 o un 0 lógico, está construido por un conjunto de compuertas lógicas, es también el bloque básico de la electrónica digital, en la construcción de memorias, registros, y sistemas secuenciales.

Un flip flop tiene dos o más entradas dependiendo del tipo de este y tiene dos salidas que son Q y $\bar{Q}$, donde Q es la salida activa a nivel alto y $\bar{Q}$ el complemento.

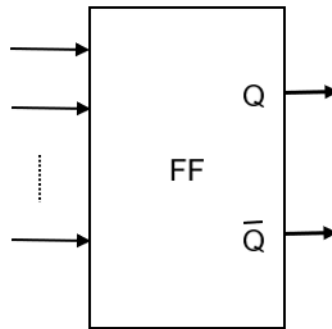


Figura 4.14 Símbolo general de un flip flop.

Cuando la salida Q está en 1 se dice que está a nivel alto, también se le conoce como estado SET o establecido y cuando la salida Q está en 0 se dice que está a nivel bajo y se le conoce como estado de CLEAR o RESET que es equivalente a borrar o reestablecer.

Los flip flops pueden tener o no señal de reloj, por lo que se les conocen como síncronos a los que manejan una señal de reloj y asíncronos los que no la tienen, el comportamiento se verá más adelante.

4.3.1 Flip Flop S - R

Es el flip flop más básico construido con compuertas NAND o compuertas NOR conocido también como LATCH. Inicialmente después de cableadas las compuertas, al ponerle la energía no es predecible cual será el valor de Q , esto depende de la velocidad de respuesta de cada compuerta, como se ve en la figura si las entradas S y R están a nivel alto, cualquiera de las respuestas a la alimentación es correcta de acuerdo con el funcionamiento de las compuertas NAND.

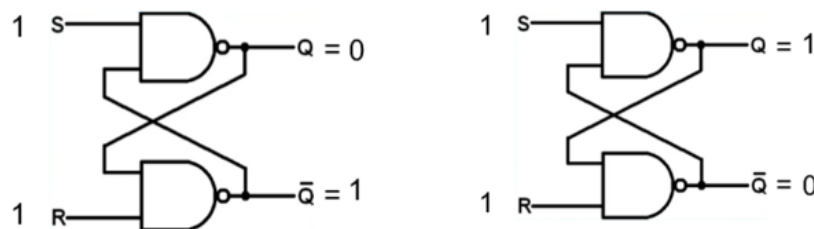


Figura 4.15 Posibles valores de salida para el flip flop S-R con entradas en S y R en 1.

Si ahora se le aplica un nivel bajo de t_0 a t_1 a la terminal de entrada S ocasionará que la salida Q pase a un nivel alto estableciéndola y la salida $\bar{Q}$ a nivel bajo. Si la terminal S se regresa a 1 ya no afecta la salida Q, esta se mantendrá a nivel alto, de allí que a la entrada S se le llama SET (poner en 1 a Q).

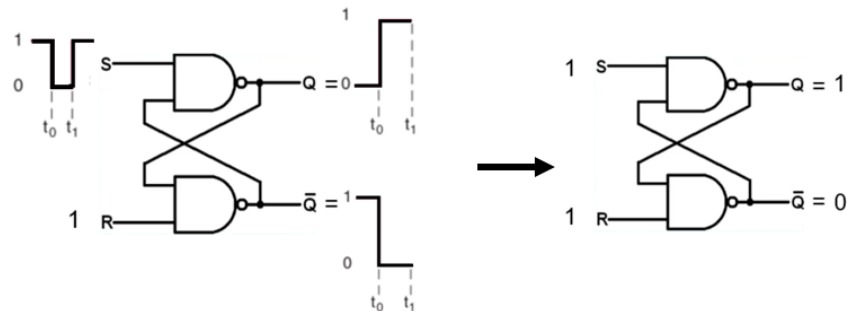


Figura 4.16 Establecimiento a 1 de la salida Q con la terminal S (SET) activa a nivel bajo.

Si nuevamente se aplica un nivel bajo de t_0 a t_1 a la entrada SET la salida Q ya no cambia de estado se mantiene en 1, memoriza su valor en 1.

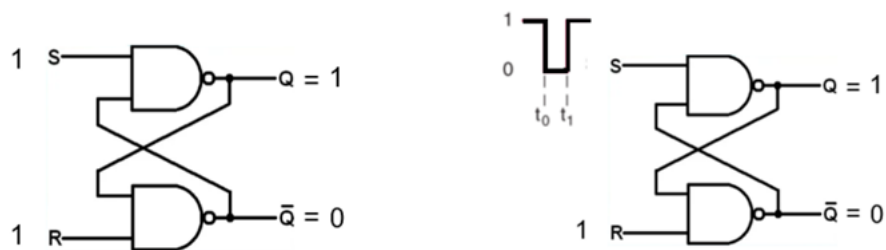


Figura 4.17 Cuando Q está en 1 ya no importa si se activa SET, Q se mantiene en 1.

Si ahora se aplica un nivel bajo de t_0 a t_1 a la entrada R ocasionará que la salida Q se vaya a cero se reestablezca o se borre, por eso a la terminal R se le conoce como RESET.

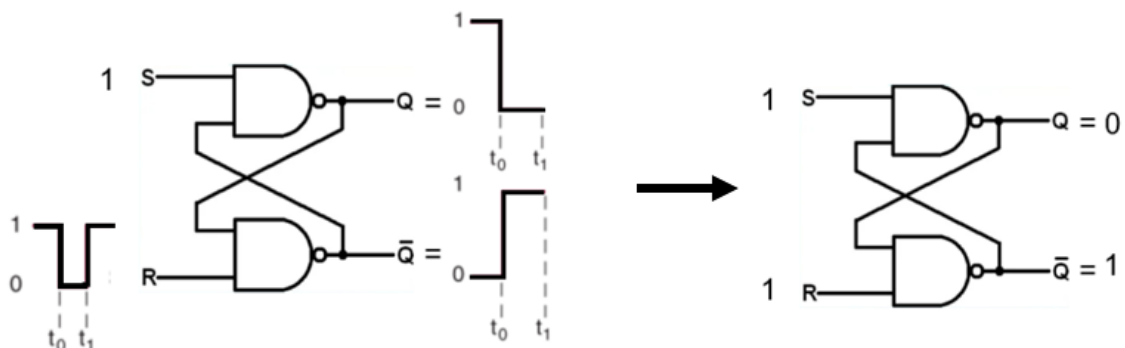


Figura 4.18 Un nivel bajo en la entrada R ocasiona que Q se vaya a cero se reestablezca de aquí en nombre de RESET.

Si nuevamente se aplica un nivel bajo de t_0 a t_1 a la entrada RESET la salida Q ya no cambia de estado se mantiene en 0, memoriza su valor en 0.

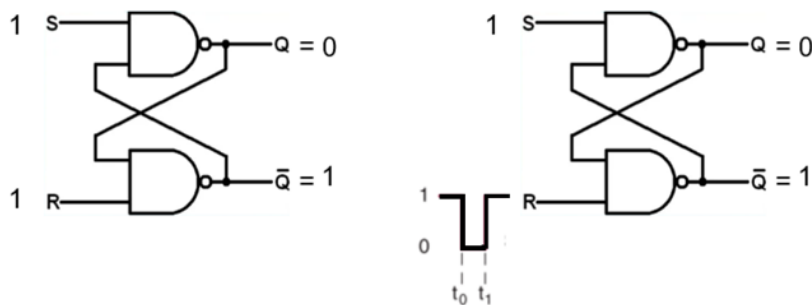


Figura 4.19 Si Q está en 0 no importa si se da un RESET Q permanece en su estado.

Si se aplica al mismo tiempo a las entradas SET y RESET un nivel bajo de t_0 a t_1 , producida un nivel alto tanto en la salida Q como en la salida $\bar{Q}$, un estado no permitido.

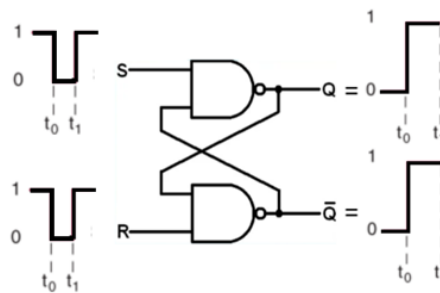


Figura 4.20 Activar SET y RESET al mismo tiempo es un estado no permitido.

Conclusión las entradas para un flip flop con compuertas NAND son activas a nivel bajo y nunca deben activarse al mismo tiempo ya que producirá un estado no permitido, dando las dos salidas a nivel alto.

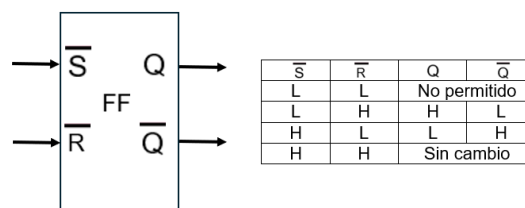


Figura 4.21 Símbolo para el flip flop SR con compuertas NAND y su tabla de verdad.

Flip Flop SR con compuertas NOR

Inicialmente cableadas las compuertas pueden tomar cualquier valor en su salida, esto depende de qué compuerta responda primero una vez que se alimenta. Las entradas para este flip flop son activas a nivel alto por lo que las entradas se inician con niveles bajos para evitar un estado no permitido.

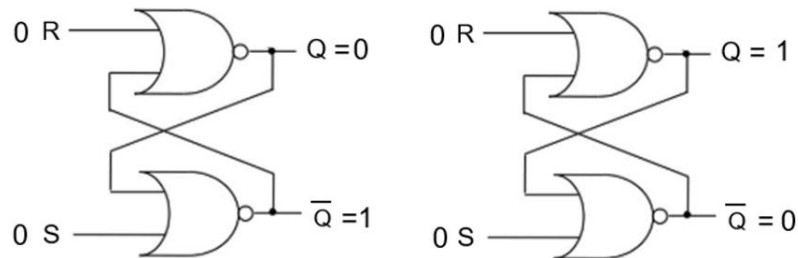


Figura 4.22 Flip Flop con compuertas NOR en su estado inicial.

Si aplicamos un nivel alto de t_0 a t_1 a la terminal S (SET), la salida Q tomará el valor alto 1.

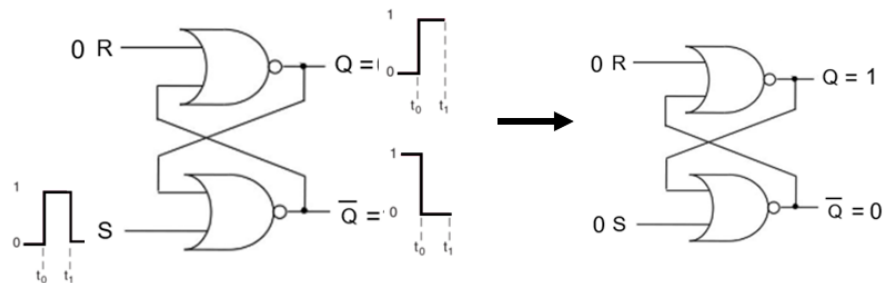


Figura 4.23 Establecimiento a 1 de la salida Q con la terminal S (SET) activa a nivel alto.

Si se le vuelve a dar un SET la salida Q ya no cambia de estado alto ya que almacena este valor hasta que se le dé una señal de RESET.

Al aplicar un RESET a nivel alto, la salida Q se irá a nivel bajo (se restablecerá) y permanecerá en ese estado hasta que se le vuelva a dar un SET.

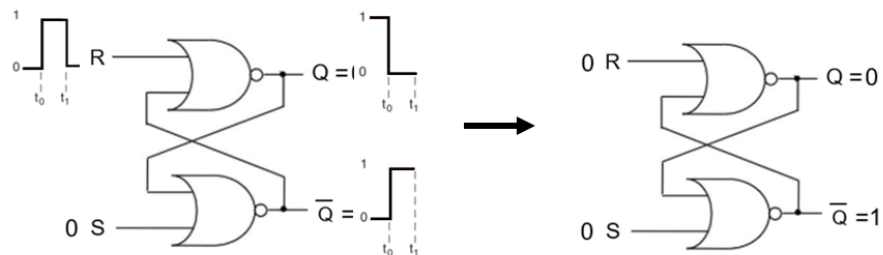


Figura 4.24 Puesta a 0 de la salida Q con la terminal RESET activa a nivel alto.

Si se llegan activar las dos entradas SET y RESET al mismo tiempo producirá un estado no permitido, es decir las salidas Q y $\bar{Q}$ tarán a nivel bajo.

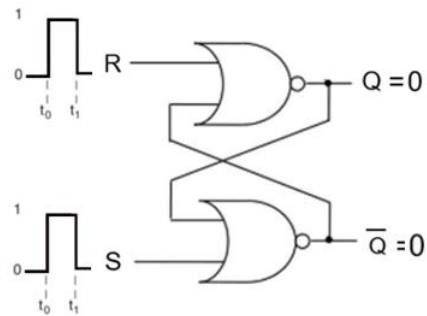


Figura 4.25 Flip Flop con las dos entradas SET y RESET activas ocasionando un estado no permitido.

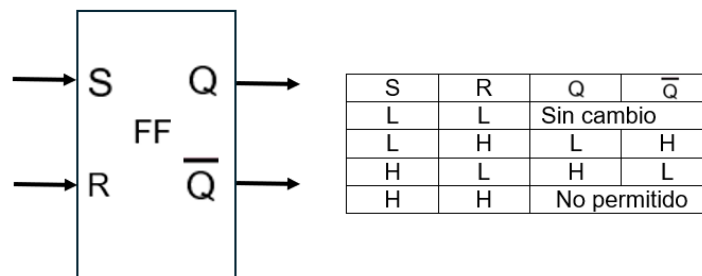


Figura 4.26 Símbolo para el flip flop SR con compuertas NOR y su tabla de verdad.

Ejemplos del uso de los Flip Flops:

La respuesta normal de un interruptor mecánico es presentar rebotes tanto al cerrar el interruptor como al abrirlo como se muestra en la siguiente figura:



Figura 4.27 Rebote mecánico de un interruptor.

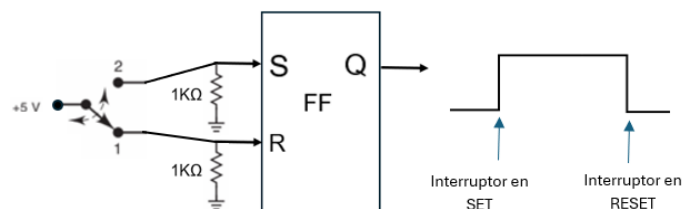


Figura 4.28 Eliminación del rebote mecánico con el flip flop.

Señal de reloj

Una señal de reloj es un tren de pulsos rectangulares utilizados para sincronizar algunos dispositivos como los flip flops, contadores, máquinas secuenciales, microprocesadores etc.

Estas señales tienen cuatro componentes a una de las cuales responden ciertos circuitos.

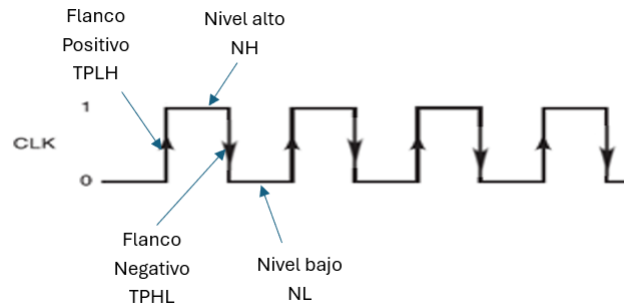


Figura 4.29 Señal de reloj.

Flip Flop Síncronos

Cuando a un flip flop se le agrega una señal de reloj se convierte en un FF síncrono, es decir sólo responde a las entradas SET y RESET cuando se habilita la señal de reloj, si la señal de reloj es por nivel bajo o alto y se deja activa se dice que el flip flop es transparente ya que responde inmediatamente a las entradas.

Flip Flop síncrono con señal de reloj por nivel.

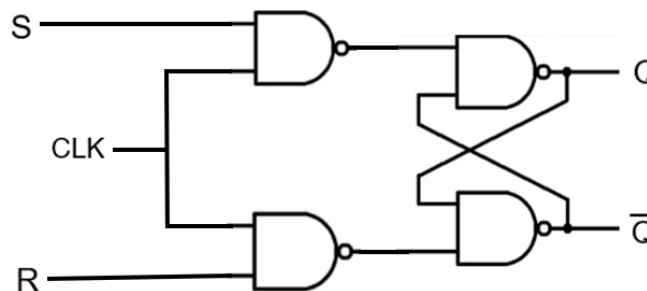


Figura 4.30 Flip Flop síncrono con señal de reloj por nivel.

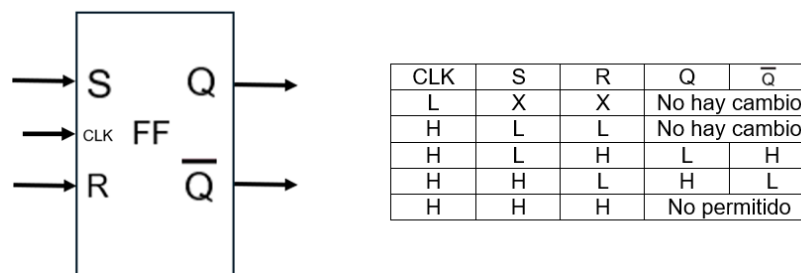


Figura 4.31 Flip Flop síncrona por nivel alto de reloj.

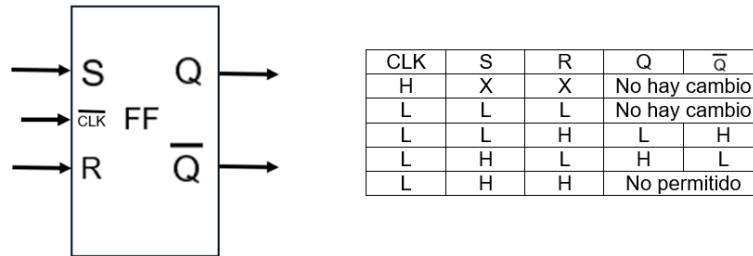


Figura 4.32 Flip Flop síncrono por nivel bajo de reloj.

Cuando un flip flop tiene una señal de reloj es importante tomar en cuenta la sincronización que debe haber entre las entradas antes de la llegada de la señal activa del reloj y después de esta. Estos tiempos requeridos para el correcto funcionamiento de los FF los da el fabricante y son:

Tiempo de establecimiento (t_s)

Es el tiempo mínimo antes de la señal activa del reloj a la que responde el flip flop en la que los datos de entrada deben estar estables. Si los datos cambian durante este tiempo el flip flop no responderá correctamente.

Tiempo de retención (t_H)

Es el tiempo mínimo después de la señal activa de reloj a la que responde el flip flop en la que los datos de entrada deben permanecer estable. Si el dato cambia demasiado pronto después de la señal activa del reloj, también podría producir errores.

Estos tiempos los da el fabricante para los distintos flip flops, el tiempo de establecimiento t_s es del orden de los nanosegundos normalmente en el rango de 5 a 50 ns y el tiempo de retención t_H en el rango de 1 a 10 ns, esto depende de la familia de los dispositivos.

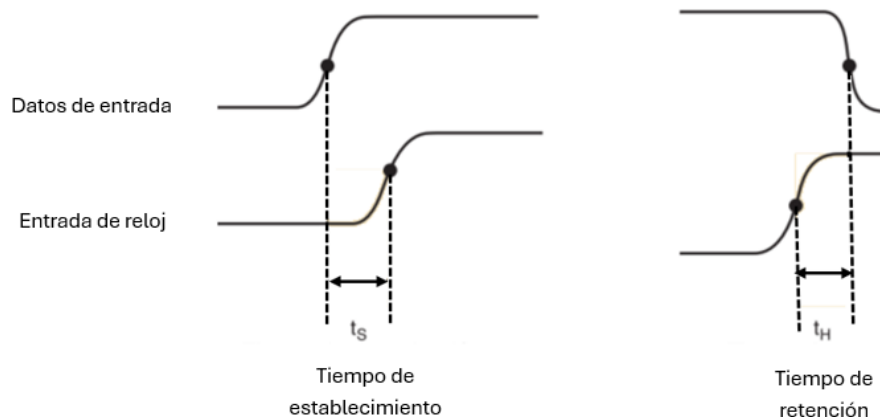


Figura 4.33 Tiempos requeridos para los datos de entrada antes y después de la señal activa del reloj.

Detectores de flanco de la señal de reloj.

Hay flip flops que responden sólo a los flancos de la señal de reloj, puede ser en el flanco positivo o en el negativo.

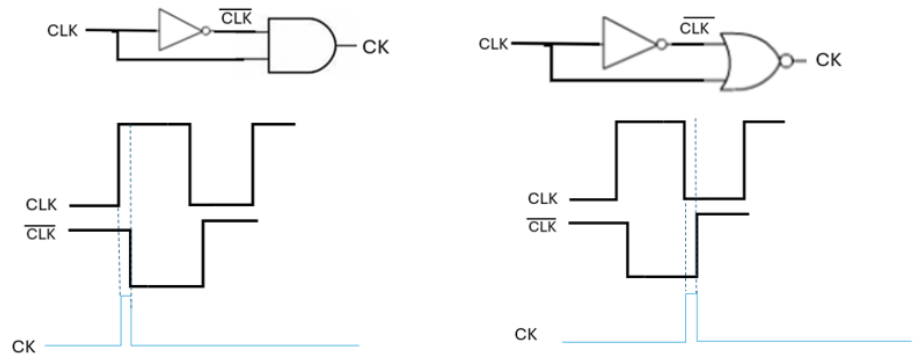


Figura 4.34 detectores de flanco positivo y negativo de una señal de reloj.

Estos detectores de flanco son posibles gracias al retraso en la propagación de la señal de salida, en el detector de flanco positivo este se da cuando la señal de reloj y la negada coincide a nivel alto, por lo tanto, ese pequeño pulso que se genera es del orden de los nanosegundos. En el detector de flanco negativo las dos entradas deben coincidir en nivel bajo y el pulso que se genera es también de unos nanosegundos.

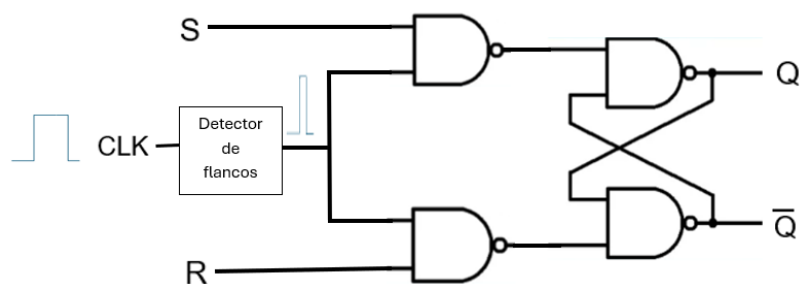


Figura 4.35 Flip Flop con señal de reloj por flanco.

La representación de los Flip Flops que manejan señal de reloj por flanco se muestran en las siguientes figuras:

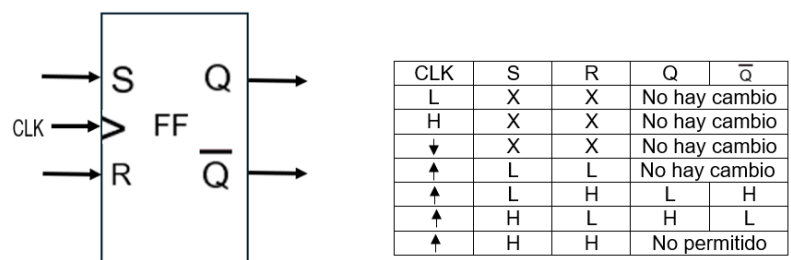


Figura 4.36 Flip Flop con señal de reloj por flanco positivo y su tabla de verdad.

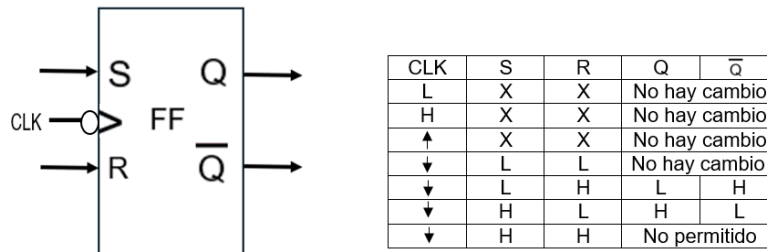


Figura 4.37 Flip Flop con señal de reloj por flanco negativo y su tabla de verdad.

Ejemplo

Encontrar la respuesta del FF con la carta de tiempos que se muestra en la figura.

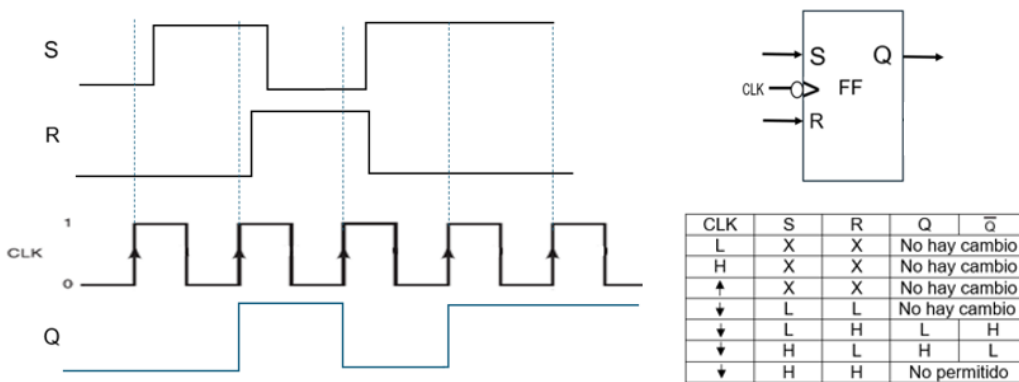


Figura 4.38 Respuesta del FF con flanco de reloj positivo.

Recuerde que los flip flops por flanco sólo responden a las entradas cuando llega el flanco de reloj al que es activo, en este caso sólo marcamos los flancos positivos y dependiendo la entrada activa es como responderá el FF.

4.3.2 El Flip Flop JK

Es una mejora del flip-flop SR, diseñada para eliminar el problema de la condición no permitida cuando ambas entradas están activadas simultáneamente. Si el Flip Flop JK tiene sus dos entradas activas y llega el pulso de reloj, las salidas conmutan, es decir, Q toma el valor de $\bar{Q}$ y $\bar{Q}$ toma el valor de Q.

El Flip Flop JK está formado básicamente por tres etapas que son:

Flip Flop básico con compuertas NAND

Circuito conductor del pulso.

Circuito detector de flancos.

La entrada J tiene la misma función que la entrada SET, es decir Q pasa a nivel alto cuando J está activa y llega el flanco activo del reloj, y la entrada K tiene la función del RESET, es decir si K está activa la salida Q se irá a nivel bajo cuando llegue el flanco activo del reloj.

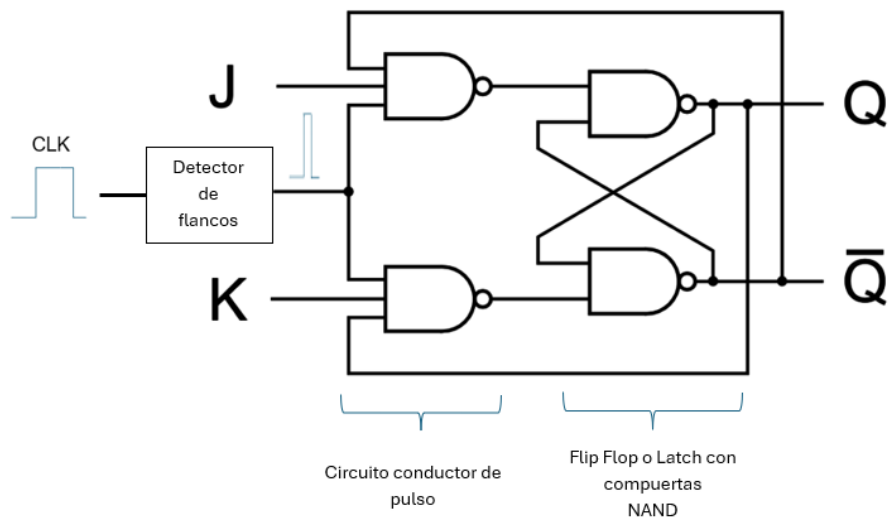
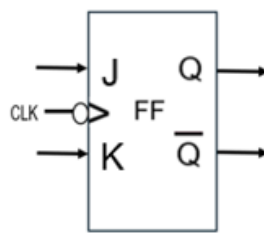
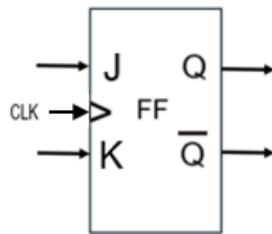


Figura 4.39 Flip Flop JK disparado por flanco.



CLK	J	K	Q	$\bar{Q}$
H	X	X	No hay cambio	
L	X	X	No hay cambio	
↑	X	X	No hay cambio	
↓	L	L	No hay cambio	
↓	L	H	L	H
↓	H	L	H	L
↓	H	H	$Q = \bar{Q}, \bar{Q} = Q$	

Figura 4.40 Símbolo y tabla de verdad de Flip Flop JK por flanco negativo de reloj.



CLK	J	K	Q	$\bar{Q}$
H	X	X	No hay cambio	
L	X	X	No hay cambio	
↓	X	X	No hay cambio	
↑	L	L	No hay cambio	
↑	L	H	L	H
↑	H	L	H	L
↑	H	H	$Q = \bar{Q}, \bar{Q} = Q$	

Figura 4.41 Símbolo y tabla de verdad de Flip Flop JK por flanco positivo de reloj.

Divisor de frecuencia.

Un flip Flop JK puede ser utilizado para dividir la frecuencia de la señal de reloj si se mantiene $j = 1$ y $K = 1$

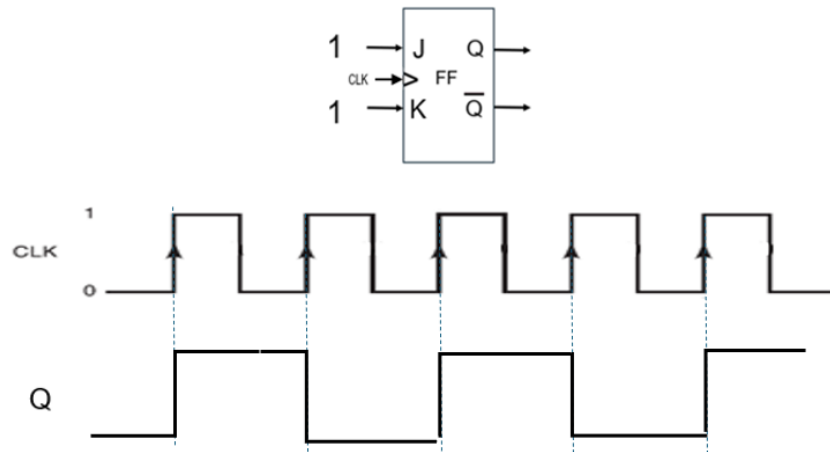


Figura 4.42 Divisor de frecuencia con el flip flop JK.

Por cada flip flop que se utilice la frecuencia se va dividiendo por dos, el primer FF tendría la mitad de la frecuencia del reloj en su terminal Q, el segundo FF tendría un cuarto de la frecuencia del reloj el tercer FF un octavo y así sucesivamente.

Cartas de tiempo con el FF JK.

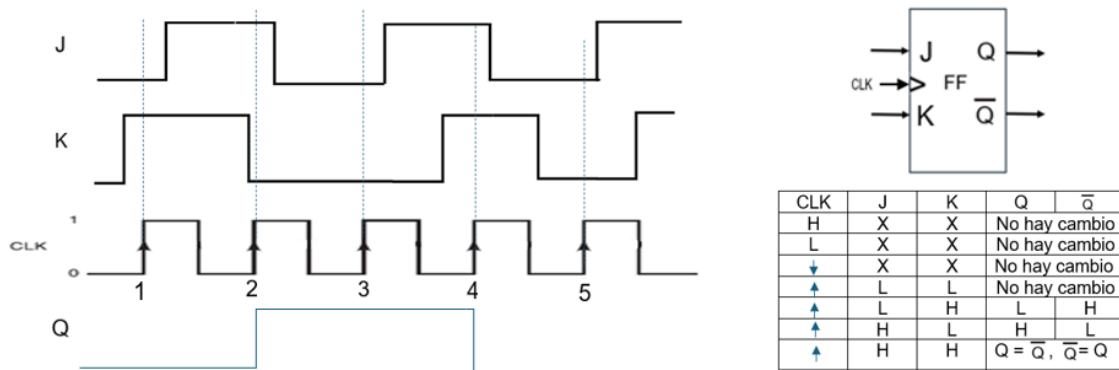


Figura 4.43 Carta tiempo para el FF JK con flancos positivos de reloj.

En el flanco 1 positivo $K = 1$ y $J = 0$, como la salida $Q = 0$ esta se mantiene ya que K hace que Q se vaya a 0, en el flanco positivo 2, $J = 1$ y $K = 0$ lo que hace que Q se vaya a 1, en el flanco positivo número 3 $J = 0$ y $K = 0$ por lo que Q no cambia se mantiene en 1, en el flanco positivo 4 $J = 1$ y $K = 1$ lo que ocasiona que Q alterne su valor, se irá a 0, en el flanco positivo 5 de reloj $J = 0$ y $K = 0$ ya no hay cambios.

Algunas de las Aplicaciones más comunes del FF JK son:

Contadores binarios.

Registros de desplazamiento.

Sistemas de memoria simples.

Circuitos secuenciales.

Flip Flops con señales asíncronas de alta prioridad.

Los FF también cuentan con señales asíncronas que operan independientemente de la demás señal incluyendo el reloj, estas señales permiten que el flip flop tome un valor

deseado en la salida Q en cualquier momento y son muy utilizadas para asegura un valor de salida en el flip flop una vez que se energiza, estas señales pueden poner a 1 la salida Q o ponerla a 0 en cualquier momento sin importar las demás entradas.

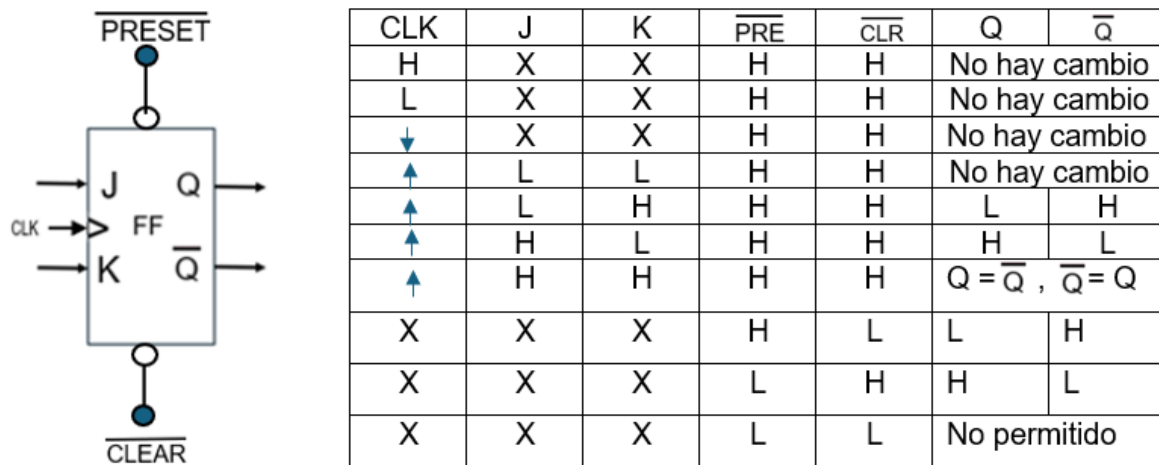


Figura 4.44 Flip Flop JK con señales asíncronas de alta prioridad.

En la tabla de verdad del flip flop JK se puede observar que cuando se presenta la señal PRESET a nivel bajo y CLEAR a nivel alto la salida Q se pone en 1 sin importar las demás entradas, si se presenta CLEAR a nivel bajo y PRESET a nivel alto la salida Q se pone en 0 sin importar las demás entradas, cuando PRESET y CLEAR se ponen a nivel bajo se presenta un estado no permitido ya que se pretende poner un 0 y un 1 al mismo tiempo en la salida Q.

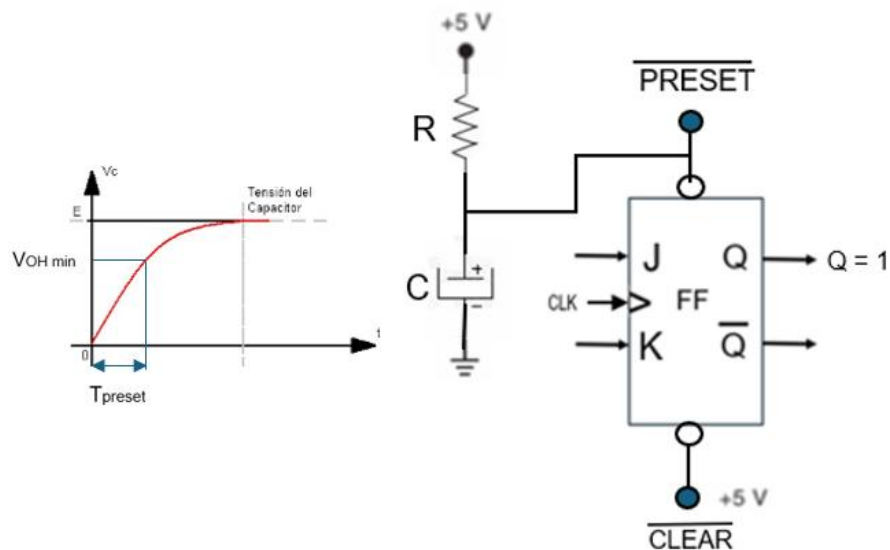


Figura 4.45 Circuito para asegurar $Q = 1$ al energizar el circuito (preset automático).

Al energizar el circuito el condensador se encuentra descargado ocasionando que la terminal PRESET lo tome como un 0 lógico y hace que la salida Q se ponga en 1, el condensador requiere un tiempo para que el valor del voltaje llegue al valor mínimo para que la entrada PRESET lo tome como un 1 lógico, al llegar y pasar este valor la terminal PRESET queda deshabilitada, el valor de la resistencia y el capacitor deben ser calculados de acuerdo a los parámetros que pida el FF con el que se está trabajando.

Ejemplo de divisor de frecuencia con el FF JK.

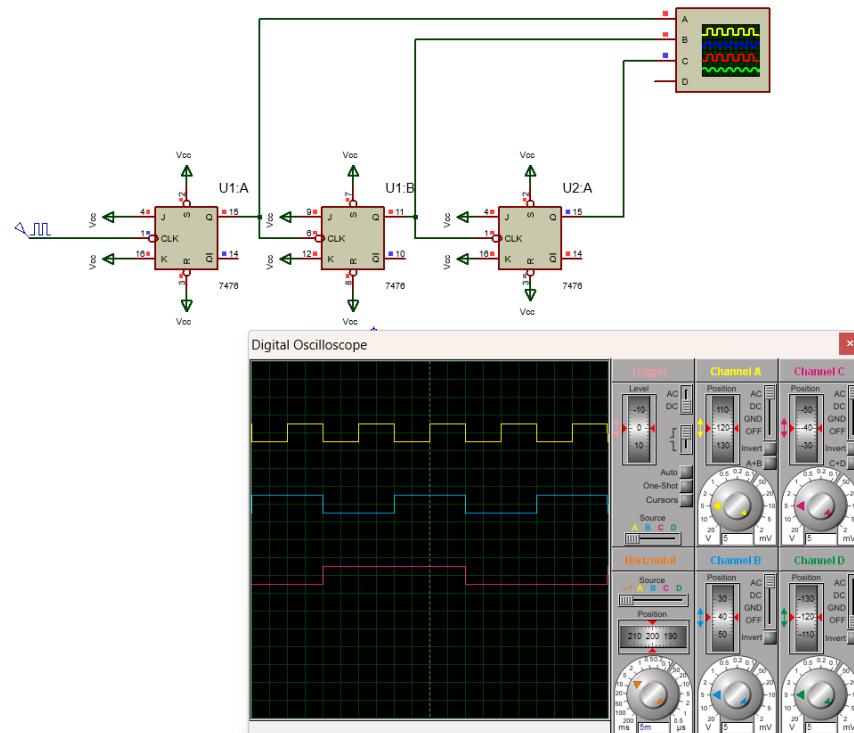


Figura 4.46 Divisor de frecuencia con el FF JK simulación en Proteus.

4.3.3 Flip Flop tipo D

El flip-flop tipo D (también llamado latch D o registro tipo D) es un circuito que se utiliza para almacenar un bit de información. Es uno de los flip-flops más comunes en electrónica digital y tiene como función captura el valor de la entrada D y ponerlo en la salida Q en el momento en que llega un pulso de reloj (CLK).

Básicamente este FF es un JK sólo que unidas las entradas a través de un inversor dejando de esta manera una sola entrada a la que se llama D.

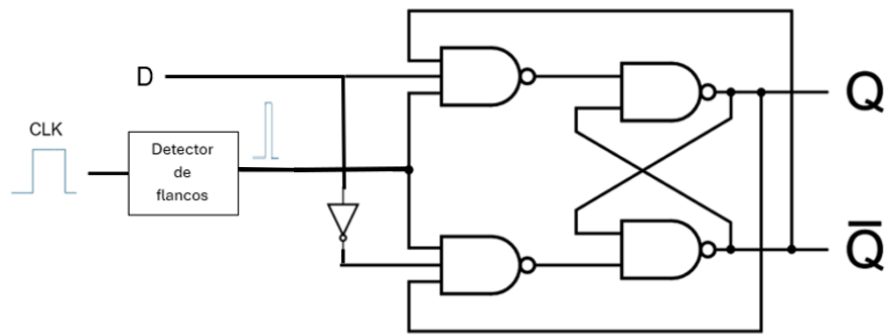


Figura 4.47 Flip Flop tipo D.

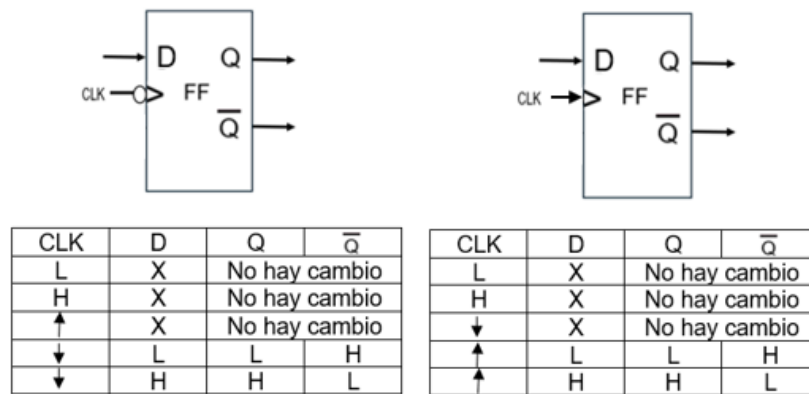


Figura 4.48 FF tipo D disparados por flanco y sus tablas de verdad.

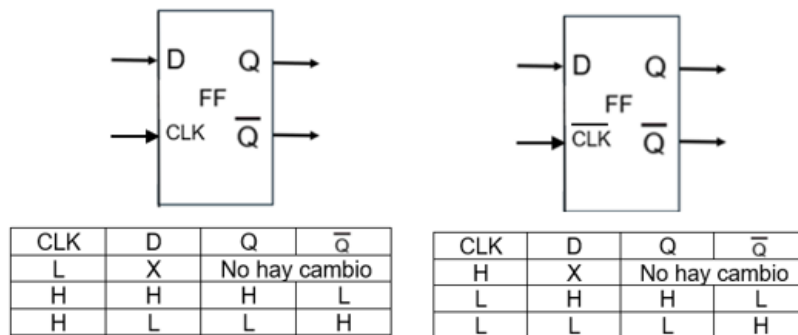


Figura 4.49 FF tipo D con reloj a nivel (latch transparentes).

En este tipo de flip flops a la señal de reloj suele llamársele habilitador y se representa por EN.

Alguna de las aplicaciones de los flip flops tipo D son:

Registro de datos

Se utilizan para guardar datos de manera temporal. Por ejemplo, 8 flip-flops tipo D pueden almacenar 1 byte (8 bits) de información lo que es muy común en microprocesadores y microcontroladores.

Contadores.

En combinación con compuertas lógicas, los flip-flops D se usan en el diseño de contadores binarios y divisores de frecuencia.

Máquinas de estados.

Son la parte fundamental para guardar el "estado actual" en autómatas finitos.

Cada flip-flop representa un bit del estado.

Memorias estáticas.

Se pueden diseñar memorias de no muy alta capacidad de datos por el espacio que ocupan, los flip-flops D forman la base de celdas de memoria estática.

Registros de corrimiento

Muy utilizados en la comunicación tipo serie y paralelo.

Encontrar la salida Q para el siguiente circuito:

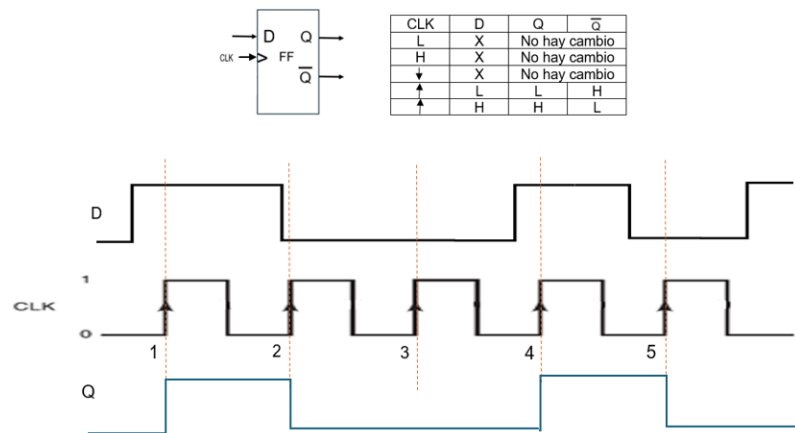


Figura 4.50 respuesta del FF tipo D por flanco positivo de reloj.

El flip flop responde únicamente a los flancos positivos de reloj, como se inicia con $Q = 0$, al llegar el primer flanco positivo la entrada D está a nivel alto por lo que la salida Q pasará a nivel alto y se mantiene hasta el flanco número 2 donde la entrada D está a nivel bajo por lo que Q pasará a nivel bajo hasta el flanco positivo 3 donde la entrada D está en bajo por lo que Q se mantiene en bajo hasta el flanco positivo 4 encontrando la entrada D en estado alto lo que ocasiona que Q pase a nivel alto hasta el flanco positivo 5 donde D está en nivel bajo y hace que Q pasa a nivel bajo.

4.3.4 Flip Flop tipo T

El flip-flop T es conocido como Flip-Flop de Conmutación ya que es capaz de alternar el valor de su salida dependiendo de lo que tenga la entrada.

El término Toggle lo que significa es que el bit de salida será invertido, es decir, de alto a bajo o de bajo a alto. Para la operación de este flip flop se requiere de una señal de reloj, por lo tanto, es un flip-flop con dos entradas.

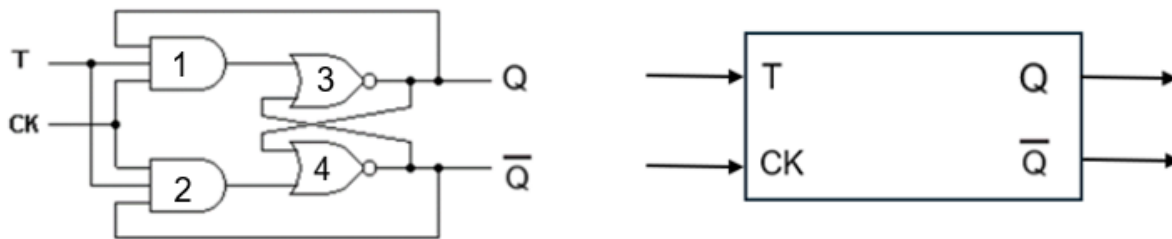


Figura 4.51 Flip Flop tipo T con compuertas y su representación simbólica.

T	Q	Q(t+1)
0	0	0
1	0	1
0	1	1
1	1	0

Figura 4.52 Tabla de verdad para el FF tipo T.

Aquí, T es la entrada de Toggle, Q es la entrada del estado actual, Q(t+1) es la salida del siguiente estado.

De aquí podemos ver que, siempre que Toggle (T) sea 0, la salida del siguiente estado (Qt+1) es la misma que la entrada del estado actual (Q).

Siempre que Toggle (T) sea 1, la salida del siguiente estado (Qt+1) será el complemento de la entrada del estado actual (Q), lo que significa que se alterna.

Ecuación Característica

La ecuación característica nos indica cuál será el siguiente estado del flip flop en términos del estado presente.

Aquí, siempre que T sea 0, Q(t+1) es igual a la entrada Q.

Y, siempre que T sea 1, Q(t+1) es el complemento de la entrada Q.

Para obtener la ecuación característica, se construye el Mapa de Karnaugh, que se muestra a continuación:

T \ Qn	$\overline{Qn}$	Qn
$\overline{T}$	0	1
T	1	0

Figura 4.53 Mapa de Karnaugh par el FF tipo T.

Si resolvemos el mapa de Karnaugh, entonces la ecuación característica será:

$$Q(n+1) = TQn' + T'Qn = T \oplus Qn$$

Funcionamiento del T Flip Flop

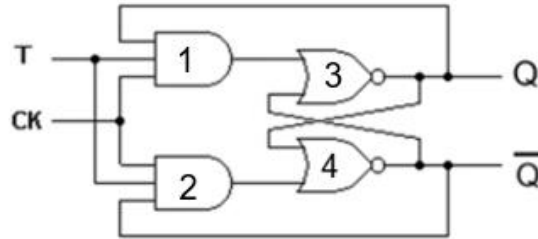


Figura 4.54 Flip FLOP tipo T.

Suponiendo que $T = 0$ y el pulso del reloj está en nivel alto, entonces la salida de las dos compuertas AND 1 y 2 será 0, la salida de la compuerta NOR 3 será Q y de manera similar la salida de la compuerta NOR 4 será Q', por lo que ambos valores de Q y Q' son los mismos que su valor anterior, lo que significa estado de retención.

Suponiendo que $T=1$, entonces la salida de ambas compuertas AND será $(T * \text{reloj} * Q)$, y dado que T y reloj son ambos 1, la salida de la compuerta AND 1 será Q, y de manera similar la salida de la compuerta AND 2 será $(T * \text{reloj} * Q')$ es decir, Q'. Ahora, la salida de la compuerta 3 será $(Q' + Q)'$ y supongamos que Q' es cero, entonces la salida de la compuerta 3 será $(0 + Q)'$ lo que significa Q' y de manera similar la salida de la compuerta 4 será $(Q + Q')$ y dado que Q' es cero, la salida de la compuerta 4 será Q' lo que significa 0 ya que Q' es cero. Por lo tanto, en este caso podemos decir que la salida cambia, porque $T=1$.

Ejemplo con cartas de tiempos.

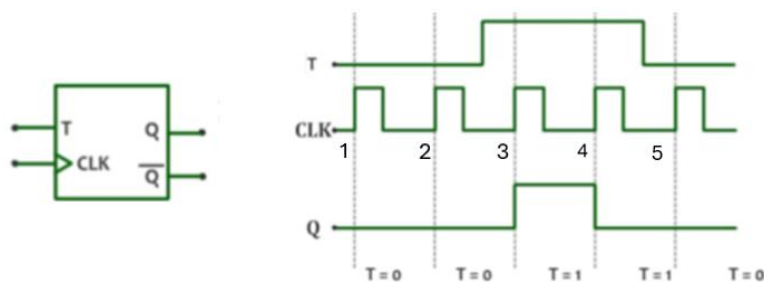


Figura 4.55 Respuesta en el tiempo del FF tipo T.

Recordando que la salida Q depende de la operación lógica XOR ($Q(n+1) = T \oplus Qn$), en el flanco positivo 1 $T = 0$ y $Q = 0$ entonces la salida sigue siendo 0, en el flanco positivo 2, $T = 0$ y $Q = 0$ la salida sigue siendo 0, en el flanco positivo 3 $T = 1$ y $Q = 0$ la salida se va a 1, en el flanco positivo 4 $T = 1$ y $Q = 1$ la salida es 0 y en el flanco positivo 5 $T = 0$ y $Q = 0$ la salida es 0.

4.3.5 Flip Flop Maestro – Esclavo.

El flip-flop maestro-esclavo es una configuración especial de flip-flop diseñada para evitar problemas como los cambios indeseados de estado causados por pulsos largos o inestables en la señal de reloj (clock).

El flip flop Maestro Esclavo está formado por dos flip flops JK (podrían ser FF SR, pero tiene el inconveniente del estado no permitido) las entradas J y K serán las entradas del maestro primer FF y las salidas Q y $\bar{Q}$ serán las entradas a J y K del segundo flip flop el cual será el esclavo, la señal de reloj entra directamente al maestro y pasara a la entrada de reloj del esclavo a través de un inversor como se muestra en la siguiente figura.

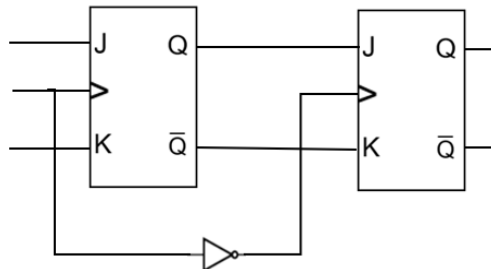


Figura 4.56 Flip Flop Maestro Esclavo.

Funcionamiento

Cuando el reloj está en 1(flanco positivo):

El maestro responde a las entradas J y K.

El esclavo permanece inactivo sin cambiar su estado, debido a que ve en su entrada de reloj un flanco negativo.

El maestro revisa el nuevo estado, pero no lo transfiere aún a la salida general del sistema.

Cuando la señal de reloj cambia a 0(flanco negativo) activo para el esclavo:

El maestro ya no responde a J y K.

El esclavo toma el estado de salida del maestro y lo refleja en su salida.

Así, el cambio de estado sólo ocurre en el flanco de bajada del reloj.

Ventajas del Flip Flop Maestro-Esclavo

Esta configuración permite que el flip-flop JK trabaje de manera sincrónica, ya que evita que la salida cambie varias veces durante un mismo pulso de reloj, eliminando el efecto de “rebote” o condición de carrera que puede ocurrir en flip-flops JK simples. Además, conserva todas las ventajas del flip-flop JK, como la capacidad de trabajar con las condiciones de set ($J=1, K=0$), reset ($J=0, K=1$), y toggle ($J=1, K=1$), con mayor control y fiabilidad.

Uso del FF maestro esclavo

El flip-flop maestro-esclavo es ampliamente utilizado en sistemas digitales como registros, contadores y memorias, donde se requiere sincronización precisa con una señal de reloj para evitar errores en el almacenamiento y procesamiento de datos.

En la actualidad ya casi no se maneja esta configuración.

Ejemplo

Arme el flip flop Maestro Esclavo en Proteus, mantenga las entradas del Flip Flop Maestro J y K en 1 y observe la forma de onda que se genera a la salida del Maestro y la del esclavo.

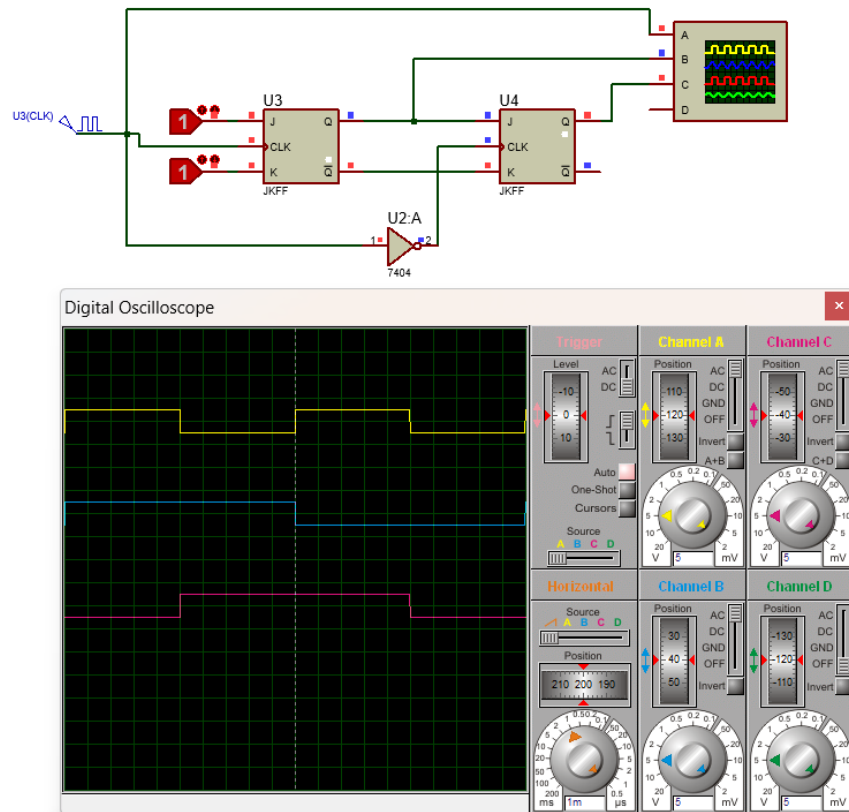


Figura 4.57 Simulación del FF Maestro Esclavo en Proteus.

Funcionamiento:

Los flip flops son activos al flanco positivo del reloj y como las dos entradas del maestro J y K están en uno, cuando llega el primer flanco positivo sólo el maestro hace el toggle de la salida, en ese momento al esclavo le llega un flanco negativo por lo que no hace ningún cambio, al llegar el flanco negativo de reloj al maestro este no hace nada sólo mantiene sus salidas que son las que le llegan como entradas al esclavo, en este momento al esclavo le llega un flanco positivo por lo que responde a su salida de acuerdo a los valores que le entrega el maestro.

En la figura se puede ver que el maestro responde a los flancos positivos del reloj de entrada mientras que el esclavo a los flancos negativos debido a que esta señal pasa por el inversor.

4.4 Diagramas y ecuaciones de estado.

Diagramas de estado

Los diagramas de estado son representaciones gráficas que muestran cómo un sistema secuencial cambia de un estado a otro en respuesta a determinadas entradas. Son una herramienta fundamental en el análisis y diseño máquinas de estado finito(FSM).

Estos diagramas están compuestos por círculo y flechas, los círculos nos muestran los diferentes estados por los que pasa la máquina o circuito secuencial, la cola de la flecha

indica el estado actual del sistema y la punta de la flecha indica el estado siguiente, normalmente en el cuerpo de la flecha se representa la acción o evento que hace que la máquina pase al siguiente estado(evento de transición), salidas (opcionalmente): Pueden indicarse dentro del estado o junto a la transición, dependiendo del tipo de máquina (Moore o Mealy).

Moore: Las salidas dependen solo del estado actual suelen colocarse dentro del estado.

Mealy: Las salidas dependen del estado actual y de las entradas, las salidas suelen colocarse en la flecha de transición.

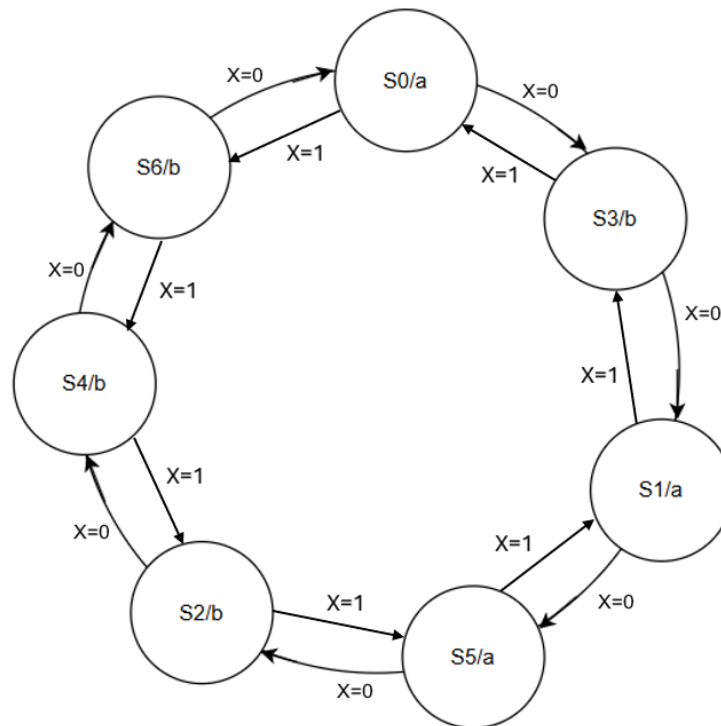


Figura 4.58 Diagrama de estado para una máquina de Moore.

En el diagrama de estados, si estamos en el estado S0/a significa que la salida es a y para pasar al estado siguiente S3/b la entrada X debe ser 0, si X es 1 entonces parará al estado S6/b donde la salida es b, en cualquier estado que se encuentre la flecha indica cuál es el estado siguiente dependiendo del evento que se dé para la transición que es la que está en el cuerpo de la flecha.

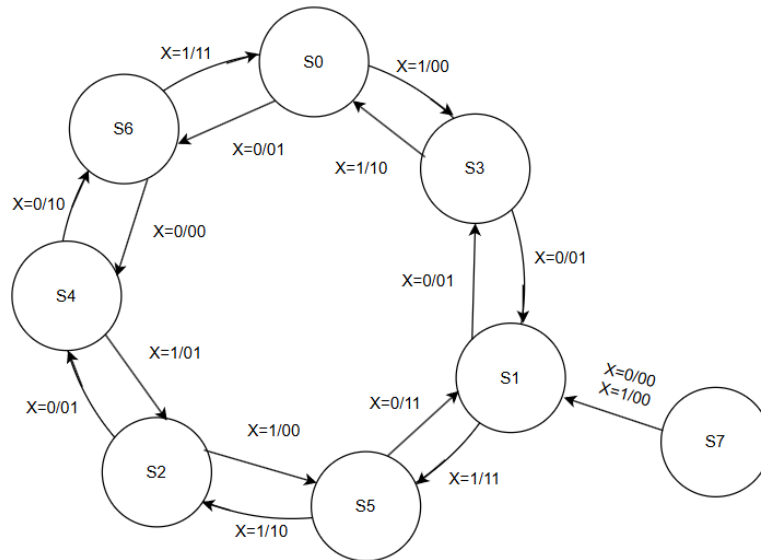


Figura 4.59 Diagrama de estados para una máquina de Mealy.

Cada círculo representa un estado diferente de la máquina, si se está por ejemplo en el estado S6 con la salida 11, se requiere que se presente el evento de transición $X=1$ para pasar al estado siguiente que es S0 y se producirá una salida 00.

Los diagramas de estado también pueden representar sólo un circuito secuencial que sólo requieren de la señal de reloj para pasar de un estado a otro sin entradas ni salidas.

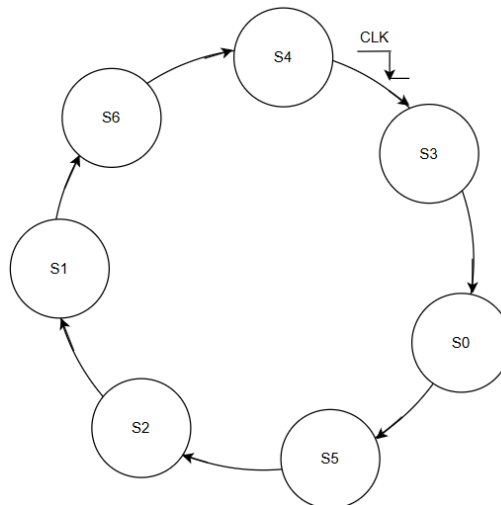


Figura 4.60 Diagrama de estados de un circuito secuencial que cambia de estado por flanco negativo de reloj.

Tablas de transición de estado.

Las tablas de transición de estados son representaciones por medio de tablas que describen cómo se cambia de un estado a otro en un circuito secuencial en función de sus entradas y las salidas que produce. Son una herramienta fundamental para el diseño y

análisis de máquinas de estados finitos (FSM), y se implementan a partir de a los diagramas de estado.

Las tablas de transición de estados en general pueden incluir las siguientes columnas:

Estado actual, que es el que tiene la máquina al iniciar su funcionamiento, antes de cualquier entrada.

Entradas, que hacen que la maquina cambie de estado si es que las tiene.

Estado siguiente, es el estado al que se irá al darse el evento de transición.

Salidas que producirá la máquina si está diseñada como máquina de Moore o Mealy.

Para construir las tablas de transición de estado es necesario saber cuántos estados diferentes tiene la máquina o circuito secuencial para determinar con cuantos bits se logran hacer todas las combinaciones posibles y generar la tabla en la columna de estado presente, a partir del estado presente y con el diagrama de estados se obtiene la columna del estado siguiente.

Ejemplo

Obtener la tabla de transición de estados del siguiente diagrama de estados.

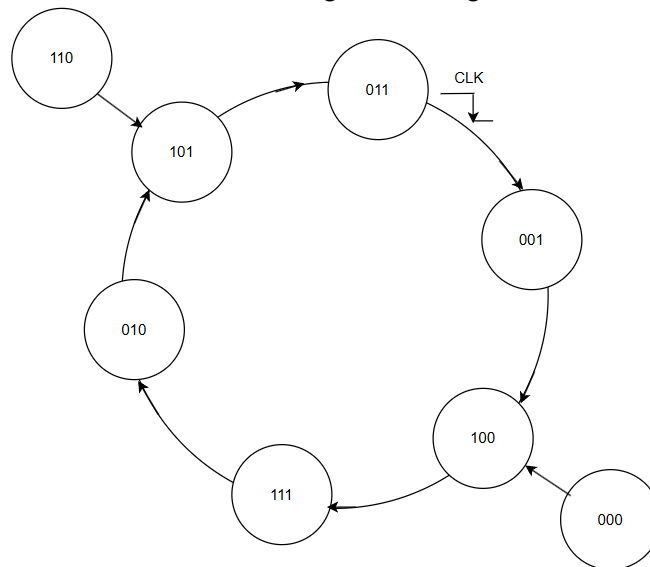


Figura 4.61 Diagrama de estados de un circuito secuencial.

En la secuencia del diagrama tiene 6 estados diferentes que forman parte del funcionamiento del circuito y dos que no son parte de ella pero que se deben tomar en cuenta y asignarle a donde se tiene que ir en caso de que al energizar el circuito aparezca alguno de estos dos estados y así evitar eventos aleatorios.

Estado presente de la tabla de transición:

Lo formamos con todas las combinaciones posibles de los tres bits en forma ascendente.

Normalmente se diseñan los circuitos secuenciales con Flip Flops por lo que llamaremos a las entradas Qc, Qb y Qa.

Estado presente		
Qc	Qb	Qa
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Figura 4.62 Tabla del estado presente para el diagrama de estados.

Para obtener la tabla de transición del estado siguiente nos ubicamos en el estado 000 del diagrama de estados y la flecha nos dirá cuál es el estado siguiente, en seguida nos ubicamos en el estado 001 del diagrama de estados y la flecha nos dirá cuál es el estado siguiente, esto se repite hasta terminar con todas las combinaciones del estado presente.

Estado presente			Estado siguiente		
0	0	0	1	0	0
0	0	1	1	0	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	0	1	0

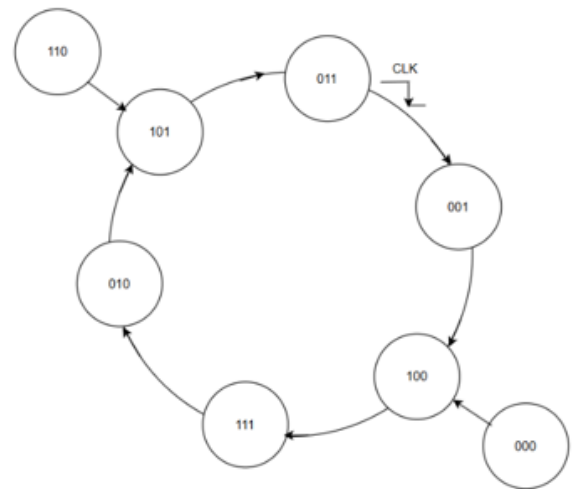


Figura 4.63 Tabla de transición de estados para el diagrama de estados mostrado.

Al ubicarse en el estado 000 el diagrama de estados muestra que el siguiente estado es 100, al ubicarse en el estado presente 001 el diagrama muestra que el siguiente estado es 100 esto se repite hasta llegar al estado presente 111 donde el diagrama indica que el siguiente estado es 010. Con esto se completa la tabla de transición de estados para el estado presente y el estado siguiente.

Obtener la tabla de transición de estados del siguiente diagrama de estados con entradas y salidas.

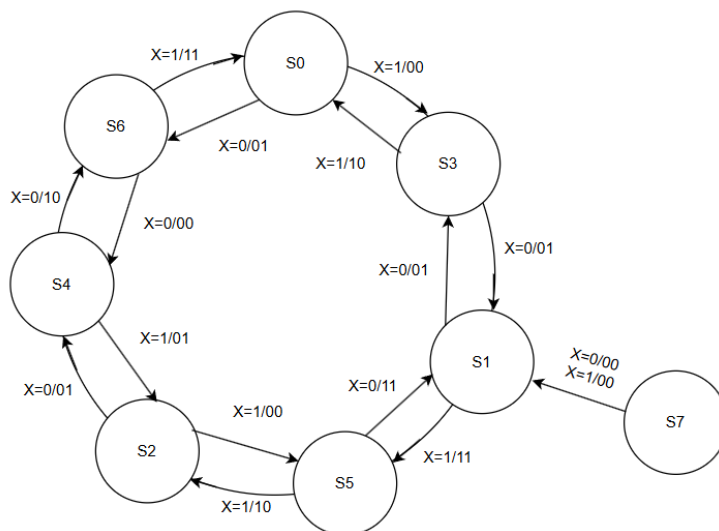


Figura 4.64 diagrama de estados de una máquina de Mealy.

Se define el valor de cada estado con los bits que se requieran, las salidas y la entrada:

Valor de los estados S			
S0	0	0	0
S1	0	0	1
S2	0	1	0
S3	0	1	1
S4	1	0	0
S5	1	0	1
S6	1	1	0
S7	1	1	1

Salidas	
b	a
0	0
0	1
1	0
1	1

Entada
X
0
1

Figura 4.65 Valores de los estados, entadas y salida.

Una vez que se tenga la columna de la variable de entrada y los estados presentes, se va al primer estado presente que normalmente es el estado de inicio y se revisa cual es el estado siguiente cuando la variable de entrada es cero y se escribe en la columna de estado siguiente, se revisa que salidas produce para anotarlas en la tabla en la columna de salidas, se va al siguiente estado presente de la tabla y se revisa a qué estado se va con la variable de entrada en cero se anota en la tabla en el estado siguiente y también las salidas que se produce, esto se repite hasta llegar al final de la tabla, note que en la tabla de transición la variable de entrada primero toma el valor de 0 para todos los estados y luego el valor de 1 también para todos los estados, esto es con el fin de cubrir todas las combinaciones posibles que se marcan en el diagrama de estados.

La tabla final de transición de estados queda de la siguiente manera:

Entrada X	Estado presente			Estado siguiente			Salidas	
	Qc	Qb	Qc	Qc'	QB'	Qa'	b	a
0	0	0	0	1	1	0	1	0
0	0	0	1	0	1	1	0	1
0	0	1	0	1	0	0	0	1
0	0	1	1	0	0	1	0	1
0	1	0	0	1	1	0	1	0
0	1	0	1	0	0	1	1	1
0	1	1	0	1	0	0	0	0
0	1	1	1	0	0	1	0	0
1	0	0	0	0	1	1	0	0
1	0	0	1	1	0	1	1	1
1	0	1	0	1	0	1	0	0
1	0	1	1	0	0	0	1	0
1	1	0	0	0	1	0	0	1
1	1	0	1	0	1	0	1	0
1	1	1	0	0	0	0	1	1
1	1	1	1	0	0	1	0	0

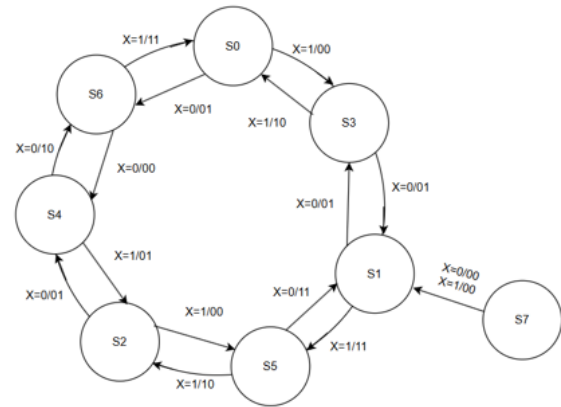


Figura 4.66 Tabla de transición de estados obtenida del diagrama de estados.

Ecuaciones de estado.

Las ecuaciones de estado en un diagrama de estado (o diagrama de máquinas de estados) son expresiones lógicas o algebraicas que describen cómo cambia el estado de un sistema en función de sus entradas y su estado actual.

Estas ecuaciones dependen del tipo de flip flop que se esté utilizando para la máquina secuencial, por lo tanto, es importante conocer la tabla de excitación del flip flop que se maneje, es decir que debe tener a la entrada el flip flop para que al llegar la señal de reloj arroje el valor que se desea.

Tabla de excitación del flip flop JK

Se le llama estado presente al valor que tiene el FF antes de que llegue la señal de reloj y estado siguiente al valor que entrega al llegar la señal de reloj, los valores que deben tener las entradas J y K para pasar de un estado presente a un estado siguiente requerido se muestra en la siguiente tabla.

Estado presente	Estado siguiente	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Figura 4.67 Tabla de excitación del flip flop JK.

En la tabla se ve que para pasar del estado presente 0 al siguiente en 0, J debe tomar el valor de 0 y K no importa (X) es decir puede ser 0 o 1 y así podemos revisar para las cuatro combinaciones posibles.

Ejemplo

Encontrar las ecuaciones de estado para el siguiente circuito secuencial usando flip flops JK.

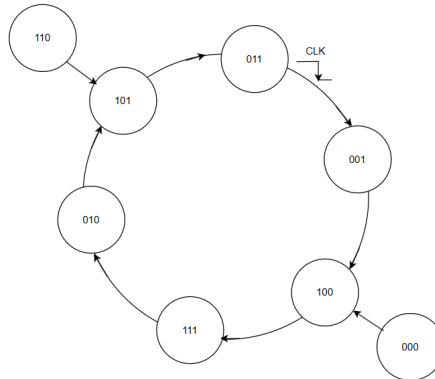


Figura 4.68 diagrama de estados para el circuito secuencial'

Lo primero es obtener la tabla de transición con los estados presentes y los estados siguientes a partir del diagrama de estados, note que se requieren de 3 bits para abarcar todos los estados posibles, se les llama Qa, Qb, y Qc a los estados presentes y Qa', Qb' y Qc' a los estados siguientes por ser la salida de los flip flops.

Estado presente			Estado siguiente		
Qc	Qb	Qa	Qc'	Qb'	Qa'
0	0	0	1	0	0
0	0	1	1	0	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	0	1	0

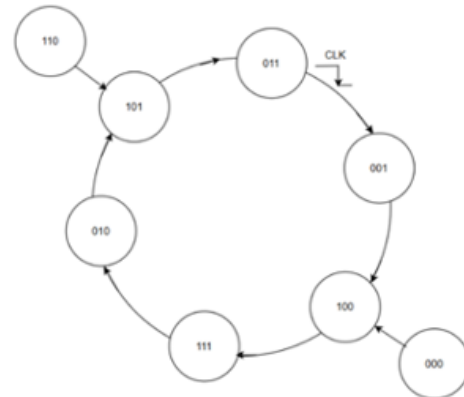


Figura 4.69 Tabla de transición con el estado presente y el siguiente del diagrama de estados.

El siguiente paso es agregar dos columnas una para J y otra para K por cada bit de entrada, en este caso son tres bits, se agregarán 6 columnas.

Estado presente			Estado siguiente			Entradas de los flip flops JK					
Qc	Qb	Qa	Qc'	Qb'	Qa'	Jc	Kc	Jb	Kb	Ja	Ka
0	0	0	1	0	0	1	X	0	X	0	X
0	0	1	1	0	0	1	X	0	X	X	1
0	1	0	1	0	1	1	X	X	1	1	X
0	1	1	0	0	1	0	X	X	1	X	0
1	0	0	1	1	1	X	0	1	X	1	X
1	0	1	0	1	1	X	1	1	X	X	0
1	1	0	1	0	1	X	0	X	1	1	X
1	1	1	0	1	0	X	1	X	0	X	1

Figura 4.70 Tabla de transición de estados completa con los FF.

En la tabla anterior se llenaron las columnas J y K usando la tabla de excitación del FF JK, se inicia bit por bit en el primer estado presente (000), Qc vale 0 en el estado presente y en el estado siguiente Qc' vale 1, de acuerdo a la tabla de excitación del FF, J debe ser 1 y K X, por lo tanto, Jc = 1, K = X, se revisa el siguiente bit, Qb = 0 y Qb' = 0, de acuerdo a la tabla de excitación J = 0 y K = X, por lo tanto Jb = 0 y Kb = X, el tercer bit Qa vale 0 en el estado presente Qa' vale 0 en el estado siguiente, de acuerdo a la tabla de excitación Ja debe ser 0 y Ka debe ser X, por lo tanto, Ja = 0 y Ka = X. Se revisa el siguiente estado presente (001) y se repite el procedimiento bit por bit para generar los valores de J y K de cada bit, esto se debe repetir hasta llenar la tabla.

Para obtener las ecuaciones de estado se trabaja con mapas de Karnaugh por cada columna de J y K, en este caso son 6 mapas.

Para Jc y Kc tenemos:

$$Jc = Qa' + Qb'$$

QbQa Qc	0 0	0 1	1 1	1 0
0	1	1	0	1
1	X	X	X	X

$$Kc = Qa$$

QbQa Qc	0 0	0 1	1 1	1 0
0	X	X	X	X
1	0	1	1	0

Figura 4.71 Mapas de Karnaugh para Jc y Kc.

Para Jb y Kb tenemos:

$$Jb = Qc$$

QbQa Qc	0 0	0 1	1 1	1 0
0	0	0	X	X
1	1	1	X	X

$$Kb = Qc' + Qa'$$

QbQa Qc	0 0	0 1	1 1	1 0
0	X	X	1	1
1	X	X	0	1

Figura 4.72 Mapas de Karnaugh para Jb y Kb.

Para Ja y Ka tenemos:

$$J_a = Q_c + Q_b$$

$$K_a = Q_c Q_b + Q_c' Q_b'$$

$Q_c \backslash Q_b Q_a$	0 0	0 1	1 1	1 0
0	0	X	X	1
1	1	X	X	1

$Q_c \backslash Q_b Q_a$	0 0	0 1	1 1	1 0
0	X	1	0	X
1	X	0	1	X

Figura 4.73 Mapas de Karnaugh para Ja y Ka.

Las ecuaciones obtenidas son las entradas a cada uno de los flip flops que conformarán el circuito secuencial.

Implementación y simulación del circuito secuencial.

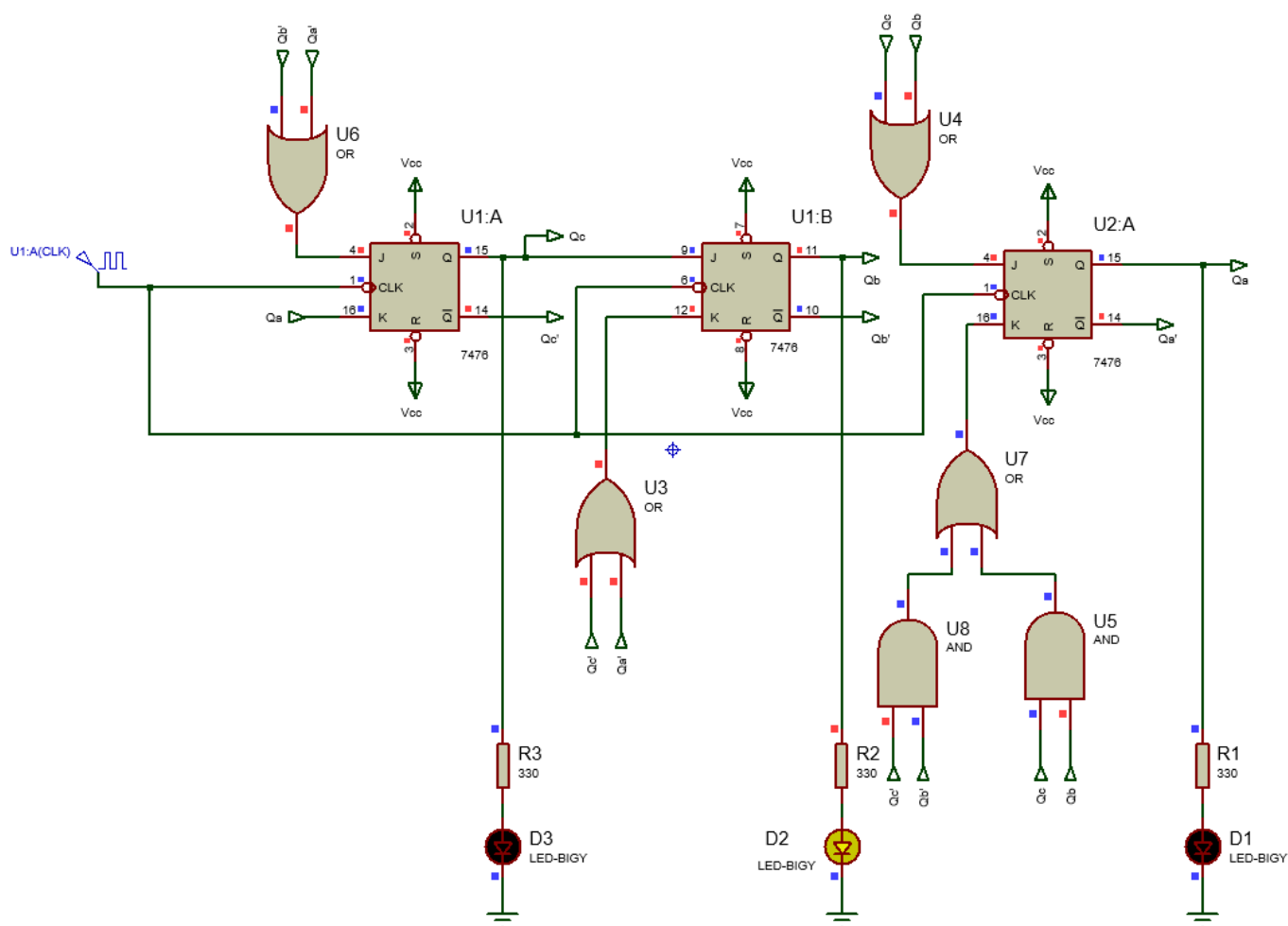


Figura 4.74 Implementación y simulación del circuito secuencial en Proteus.

4.5 Circuitos síncrono y asíncronos.

Circuito secuencial síncrono

Un circuito secuencial síncrono es un circuito lógico donde el cambio de las salidas depende del estado actual del circuito y de las entradas, pero el cambio en las salidas sólo se produce a la llegada de la señal de reloj.

También se puede decir que un circuito secuencial es donde todos los flip-flops (elementos de memoria) cambian de estado simultáneamente, en los flancos (borde positivo o negativo) de una señal de reloj que tienen en común.

Este tipo de circuitos son más fáciles de diseñar partiendo del diagrama de estados, el único inconveniente que tienen es que son más lentos debido a que se tiene que esperar a que llegue la señal de reloj para hacer el cambio en la salida.

Ejemplo

Diseñar con flip flops tipo D el circuito síncrono que respete el siguiente diagrama de estados.

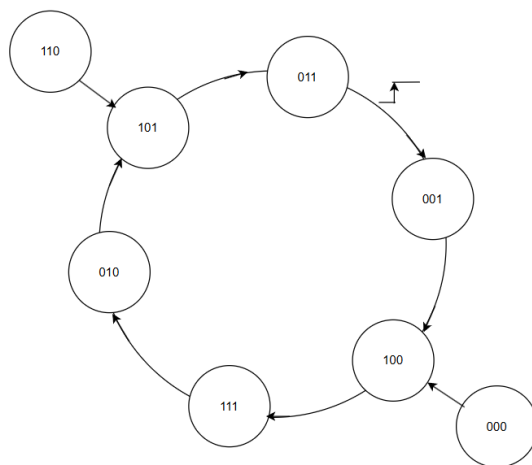


Figura 4.75 Diagrama de estados para el circuito secuencial síncrono.

Para este diseño se usará la tabla de excitación del flip flop tipo D, es la siguiente:

Estado presente	Estado siguiente	D
0	0	0
1	1	1

Figura 4.76 Tabla de excitación de FF tipo D.

Se hace la tabla de transición del estado presente y el siguiente siguiendo el diagrama de estados:

Estado presente			Estado siguiente		
Qc	Qb	Qa	Qc'	Qb'	Qa'
0	0	0	1	0	0
0	0	1	1	0	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	0	1	0

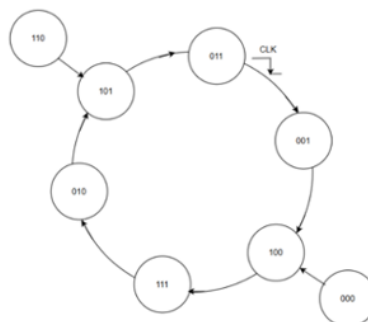


Figura 4.77 Tabla de transición del estado presente y el siguiente obtenida del diagrama de estados.

Se agrega una columna más por cada bit de entrada que se tenga, debido a que el flip flop tipo D sólo tiene una entrada, note que las columnas que se agregaran tienen la misma información que el estado siguiente por lo que se puede decir que cuando se usa FF tipo D los mapas de Karnaugh se pueden trabajar con el estado siguiente.

Estado presente			Estado siguiente			Entadas a los FF D		
Qc	Qb	Qa	Qc'	Qb'	Qa'	Dc	Db	Da
0	0	0	1	0	0	1	0	0
0	0	1	1	0	0	1	0	0
0	1	0	1	0	1	1	0	1
0	1	1	0	0	1	0	0	1
1	0	0	1	1	1	1	1	1
1	0	1	0	1	1	0	1	1
1	1	0	1	0	1	1	0	1
1	1	1	0	1	0	0	1	0

Figura 4.78 tabla de transición completa.

Ahora trabajamos con los mapas de Karnaugh para obtener las ecuaciones de estado.

$$Dc = Qa' + Qc'Qb'$$

Qc \ QbQa	0 0		0 1		1 1		1 0	
	0	1	0	1	0	1	0	1
0	1	1	0	0	1	1	0	0
1	1	0	0	0	1	1	0	0

Figura 4.79 Mapa de Karnaugh para la entada Dc.

$$Db = Qb'Qc + QaQc$$

Qc \ QbQa	0 0		0 1		1 1		1 0	
	0	1	0	1	0	1	0	1
0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0

Figura 4.80 Mapa de Karnaugh para la entada Db.

$$Da = QcQb' + Qc'Qb + QbQa'$$

Qc \ QbQa	0 0		0 1		1 1		1 0	
	0	1	0	1	0	1	0	1
0	0	0	0	0	1	1	0	0
1	1	1	1	1	0	0	1	1

Figura 4.81 Mapa de Karnaugh para la entada Da.

Ahora se implementa el circuito trabajando con las ecuaciones de estado obtenidas de los mapas de Karnaugh.

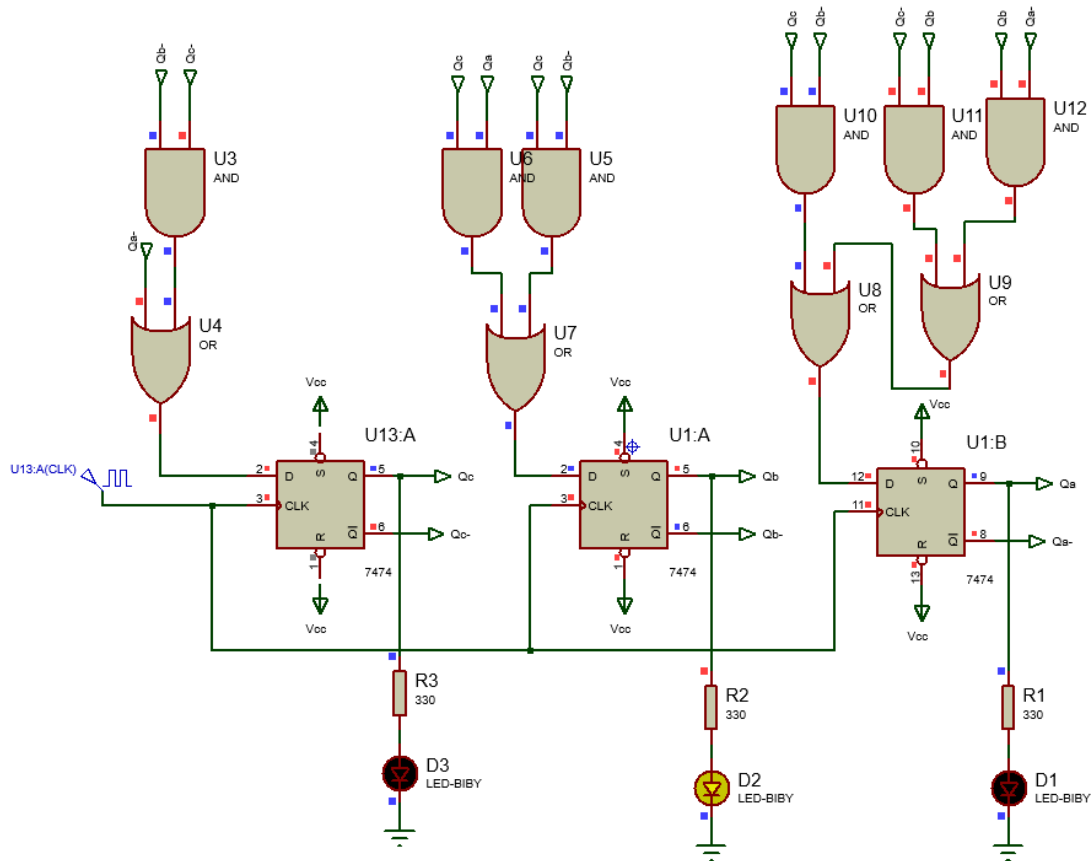


Figura 4.82 Circuito secuencial síncrono armado y simulado en Proteus.

Circuitos secuenciales asíncronos.

Los circuitos secuenciales asíncronos no tienen una señal de reloj para sincronizar su funcionamiento, son circuitos de media a baja complejidad y el número de compuertas requerido para su implementación no es muy elevado.

En los circuitos asíncronos las salidas cambian de estado no sólo a través de las entradas recibidas en ese instante, sino también de la respuesta anterior del sistema. Es decir, de la secuencia de entradas anteriores hasta ese momento. Esto quiere decir que un circuito secuencial puede tomar valores de salida diferente para una misma combinación de variables de entrada.

Para poder diferenciar entre los distintos estados para las mismas entradas en los que puede estar el sistema secuencial, es necesario el manejo de variables adicionales a las que se les llamarán variables internas las cuales se representan con las letras Q_t , algunas veces estas variables coinciden con las salidas del sistema.

En los sistemas secuenciales asíncronos, la operación es casi instantánea ya que no tienen que esperar un pulso de reloj. Están limitados solamente por el retraso de propagación de las compuertas lógica.

Los circuitos asíncronos pueden pasar a un estado incorrecto cuando se activan 2 entradas al mismo tiempo. Esto se llama condición de carrera. Los circuitos asíncronos son más difíciles de diseñar para una operación confiable. Estos se utilizan principalmente en sistemas de alta velocidad.

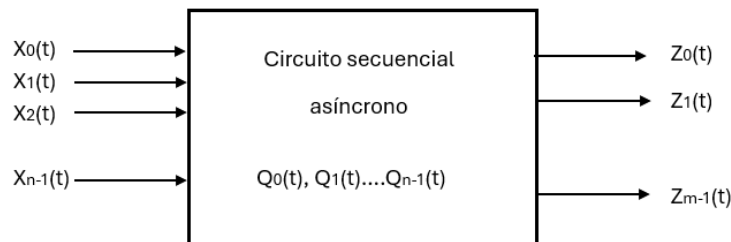


Figura 4.83 Esquema de un circuito secuencial asíncrono con las variables internas $Q(t)$.

Este tipo de circuitos se caracterizan por las dos siguientes ecuaciones:

$$Z_i(t) = f(X_j(t), Q_h(t))$$

Para una misma combinación de variables de entrada y de las variables internas $Q_h(t)$ el circuito presentará el mismo valor en las variables de salida.

$$Q_h(t + \Delta t) = J_h(X_j(t), Q_n(t))$$

Para realizar una secuencia determinada el circuito evoluciona de un estado interno a otro lo que determina una actualización de las variables internas $Q_h(t + \Delta t)$.

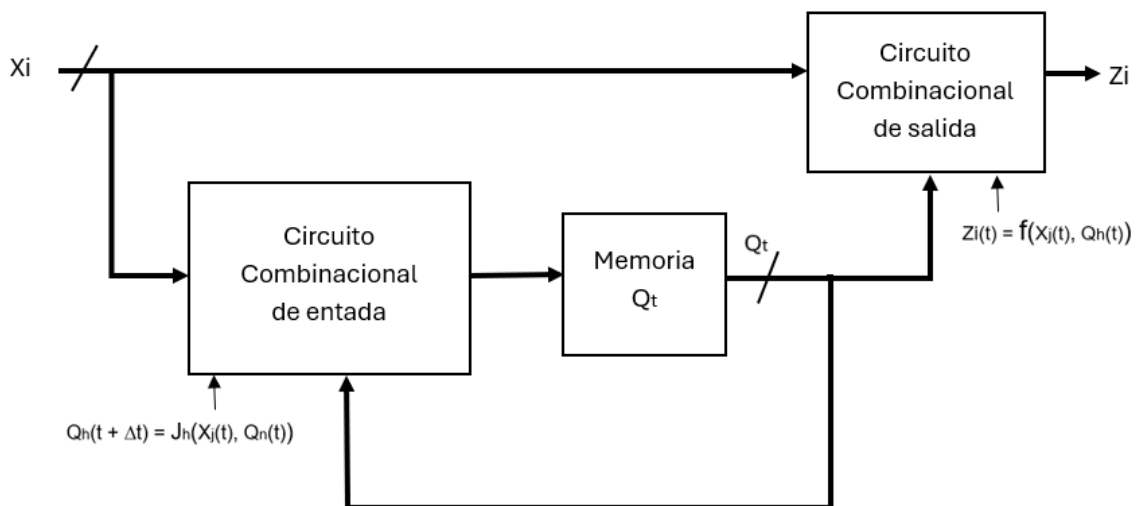


Figura 4.84 Esquema general de un circuito secuencial asíncrono.

Pasos para el diseño o síntesis de un circuito secuencial asíncrono:

- 1 Identificar todas las variables del sistema tanto entradas como salidas.
- 2 diseñar la tabla de las fases del circuito
- 3 Reducir al grado máximo posible la tabla de fases.
- 4 Asignar las variables internas necesarias del circuito.
- 5 Hacer la tabla de verdad a partir de la tabla de fases reducida.
- 6 Obtener las funciones algebraicas.
- 7 implementar el circuito.

Ejemplo

Diseñar un control con un circuito secuencial asíncrono para una puerta de un super que cumpla las siguientes condiciones.

- a) Cuando una persona sea detectada la puerta iniciará la apertura.
- b) Si la persona permanece en el área de detección la puerta deberá permanecer abierta.
- d) Cuando la persona sale del área de detección cuando la puerta está abriendo, esta deberá abrir en su totalidad y luego cerrarse.
- e) Si la puerta está cerrando y detecta una persona esta deberá abrir nuevamente en su totalidad.

Diseñar el circuito con las tablas de fases.

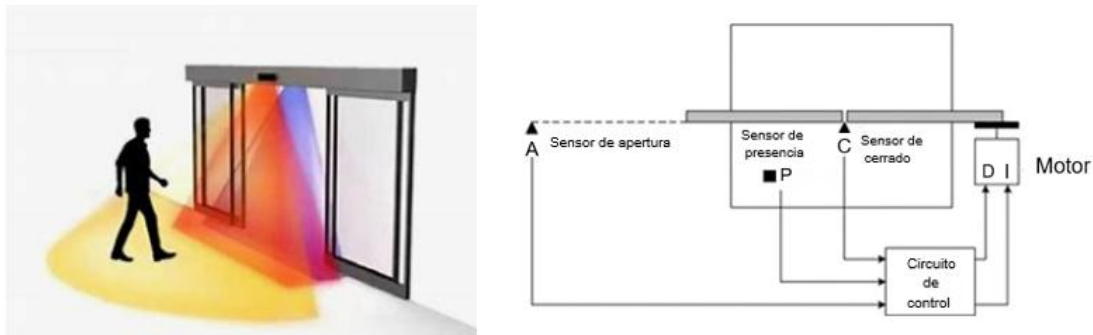


Figura 4.85 Puerta automática y sus sensores.

- a) Identificación de variables.

A sensor de puerta abierta

P sensor de presencia

C sensor de puerta cerrada

Estos tres sensores producen un 1 de salida cuando están activados.

Las variables D son las del motor y se comportan de la siguiente manera:

D	I	Acción del motor
0	0	Parado
0	1	Cerrando
1	0	Abriendo
1	1	No se debe dar

Figura 4.86 tabla de comportamiento del motor de la puerta.

Con las 3 variables de entrada se pueden hacer 8 combinaciones, no todas son posibles que se den, pero las tomaremos en cuenta para hacer la tabla de estados completa.

PCA	000	001	010	011	100	101	110	111	D	I

Figura 4.87 Tabla con las combinaciones de las variables para los generar los estados.

Como primer paso se revisa cuál combinación de variables se puede dar. Suponiendo que no hay personas y la puerta está cerrada, entonces, $P=0$, $C=1$ y $A=0$, el motor debe estar parado, esto nos da el estado estable 1 en esa combinación. En seguida se revisa que evento se puede dar partiendo de este, recuerde que no pueden cambiar dos variables de entrada al mismo tiempo, además las combinaciones $P=0$, $C=1$ y $A=1$ lo mismo que $P=1$, $C=1$ y $A=1$ no se pueden dar ya que no puede estar abierta y cerrada la puerta al mismo tiempo. Recordando que los estados estables se representan con un número encerrado en un círculo o entre paréntesis, las transiciones sólo con el número y los eventos que no se pueden dar con un guion, queda de la siguiente manera:

PCA	000	001	010	011	100	101	110	111	D	I
	-	-	(1)	-	-	-	2	-	0	0
							(2)		1	0

La combinación 000 no se puede dar ya que el motor está parado, la combinación 110 si se puede dar que indica que una persona está presente por lo tanto ponemos un estado transitorio 2 al estado estable (2) y el motor debe abrir la puerta, las demás combinaciones no son posibles dado que cambian 2 variables al mismo tiempo.

Revisando el segundo renglón, se observa que la combinación 010 se puede dar, es decir, la persona puede retirarse una vez que inicio la apertura de la puerta, se pone un estado transitorio 3 a un estado estable (3), el motor debe seguir abriendo hasta su totalidad y enseguida inicia el cierre si no hay persona, también se puede dar la combinación 100 al iniciar la apertura de la puerta, se pone un transitorio 4 a un estado estable (4) el motor sigue abriendo, las demás combinaciones del renglón 2 cambian dos entradas por lo que se marca con guion, esto queda de la siguiente manera:

PCA	000	001	010	011	100	101	110	111	D	I
	-	-	(1)	-	-	-	2	-	0	0
	-	-	3	-	4	4	(2)	4	1	0
			(3)						1	0
					(4)				1	0

Para el renglón 3 se puede observar que la combinación 110 que indica que hay una persona y la puerta se sigue abriendo por lo tanto ponemos un estado transitorio al estado estable (2) para que la puerta se siga abriendo, la combinación 000 si se puede dar ya que la puerta se sigue abriendo por lo que se pone un estado transitorio 5 el estado estable (5) para que la puerta siga abriendo, en el resto de las combinaciones cambia 2 entradas por lo tanto, se pone un guion, la tabla con el renglón 3 queda de la siguiente manera:

PCA	000	001	010	011	100	101	110	111	D	I
	-	-	(1)	-	-	-	2	-	0	0
	-	-	3	-	4	4	(2)	4	1	0
	5	-	(3)	-	-	-	2	-	1	0
					(4)				1	0
	(5)								1	0

Revisando el renglón cuatro donde se tiene una persona presente, puede suceder que la persona se retire cuando la puerta se está abriendo, entonces se puede dar la combinación 000 en la que se tendrá que mandar al estado estable (5) para que la puerta se abra totalmente, también puede ser que la persona se quede hasta que la puerta este totalmente abierta que es la combinación 101, se pone un estado transitorio 6 al estado estable (6) donde se tiene que parar el motor hasta que la persona se quite, en el resto de las combinaciones cambian dos variables de entrada por lo tanto, se pone un guion, la tabla queda de la siguiente manera:

PCA	000	001	010	011	100	101	110	111	D	I
	-	-	(1)	-	-	-	2	-	0	0
	-	-	3	-	4	4	(2)	4	1	0
	5	-	(3)	-	-	-	2	-	1	0
	5	-	-	-	(4)	6	-	-	1	0
	(5)								1	0
						(6)			0	0

Las posibilidades para el renglón 5 donde no hay persona y la puerta se está abriendo, puede ser que se presente una persona mientras esto sucede y quedaría la combinación 100, por lo que la puerta debe seguir abriendo se pone un transitorio 4 al estado estable (4) que es donde está abriendo la puerta, que cambie a 010 no se puede dar ya que la puerta está abriendo, la combinación 001 si se puede dar ya no hay persona pero la puerta está totalmente abierta por lo tanto se pone un transitorio 7 al estado estable (7) donde inicia el cierre de la puerta, las demás combinaciones no se pueden dar, se pone un guion, la tabla queda de la siguiente manera:

PCA	000	001	010	011	100	101	110	111	D	I
	-	-	(1)	-	-	-	2	-	0	0
	-	-	3	-	4	4	(2)	4	1	0
	5	-	(3)	-	-	-	2	-	1	0
	5	-	-	-	(4)	6	-	-	1	0
	(5)	7	-	-	4	-	-	-	1	0
						(6)			0	0
		(7)							0	1

En el renglón 6 esta una persona y la puerta está totalmente abierta, está en reposo, lo que puede pasar es que la persona se retire y entonces la puerta debe empezar a cerrar que es la combinación 001, como ya se tiene un estado que está cerrando la puerta se pone un transitorio 7 a ese estado, ya no hay más cambios posibles, se pone un guion y queda de la siguiente manera:

PCA	000	001	010	011	100	101	110	111	D	I
	-	-	(1)	-	-	-	2	-	0	0
	-	-	3	-	4	4	(2)	4	1	0
	5	-	(3)	-	-	-	2	-	1	0
	5	-	-	-	(4)	6	-	-	1	0
	(5)	7	-	-	4	-	-	-	1	0
	-	7	-	-	-	(6)	-	-	0	0
		(7)							0	1

En el renglón no hay persona y la puerta está abierta pero comienza a cerrarse si se presenta una persona antes del inicio del cierre que es el caso 101 debe parar el motor que es el estado estable (6) se pone un transitorio 6, otra posibilidad es que se presente la combinación 000, ya se está cerrando la puerta y no hay persona, entonces tiene que cerrar completamente, como en el estado (5) la puerta se está abriendo no se puede ir a él, se pone un transitorio 8 para el estado estable (8) que mantenga cerrando la puerta, los demás estados no se pueden dar se pone un guion, la tabla queda de la siguiente manera:

PCA	000	001	010	011	100	101	110	111	D	I
	-	-	(1)	-	-	-	2	-	0	0
	-	-	3	-	4	4	(2)	4	1	0
	5	-	(3)	-	-	-	2	-	1	0
	5	-	-	-	(4)	6	-	-	1	0
	(5)	7	-	-	4	-	-	-	1	0
	-	7	-	-	-	(6)	-	-	0	0
	8	(7)	-	-	-	6	-	-	0	1
	(8)								0	1

Por último, en el renglón 8 puede que una persona se acerque lo que daría la combinación 100 se tendría que abrir la puerta que es el estado estable (4) se pone un transitorio a ese estado, también puede aparecer el estado 010 donde no hay persona y la puerta ya está cerrada, se pone un transitorio al estado estable (1) no hay persona y el motor está parado,

el caso 001 no se puede dar porque la puerta está cerrando, en los demás casos cambian dos variables se pone un guion, la tabla final queda de la siguiente manera:

PCA	000	001	010	011	100	101	110	111	D	I
	-	-	(1)	-	-	-	2	-	0	0
	-	-	3	-	4	4	(2)	4	1	0
	5	-	(3)	-	-	-	2	-	1	0
	5	-	-	-	(4)	6	-	-	1	0
	(5)	7	-	-	4	-	-	-	1	0
	-	7	-	-	-	(6)	-	-	0	0
	8	(7)	-	-	-	6	-	-	0	1
	(8)	-	1	-	4	-	-	-	0	1

Reducción de la tabla.

Para reducir la tabla se revisan los renglones y los que tengan los mismos estados estables o estado estable y mismo transitorio o un guion se pueden fusionar, es conveniente poner nombre a los renglones para poder asociarlos y no cometer errores.

PCA	000	001	010	011	100	101	110	111	D	I
a	-	-	(1)	-	-	-	2	-	0	0
b	-	-	3	-	4	4	(2)	4	1	0
c	5	-	(3)	-	-	-	2	-	1	0
d	5	-	-	-	(4)	6	-	-	1	0
e	(5)	7	-	-	4	-	-	-	1	0
f	-	7	-	-	-	(6)	-	-	0	0
g	8	(7)	-	-	-	6	-	-	0	1
h	(8)	-	1	-	4	-	-	-	0	1

Se puede ver que los renglones b, c, d, e y f se pueden fusionar ya que en cada columna se tiene el mismo estado estable o el mismo transitorio o un guion por ejemplo en la combinación 000 se tiene:

b	-
c	5
d	5
e	(5)
f	-

La columna tiene – 5 5 (5) – por lo tanto, se pueden fusionar, lo mismo sucede con el resto de las columnas, para estos renglones.

Las columnas a, g y h también se pueden fusionar ya que todas sus columnas tienen el mismo estado o transitorio o guion, los renglones que se pueden fusionar están marcados con el mismo color en la siguiente tabla:

PCA	000	001	010	011	100	101	110	111	D	I
a	-	-	(1)	-	-	-	2	-	0	0
b	-	-	3	-	4	4	(2)	4	1	0
c	5	-	(3)	-	-	-	2	-	1	0
d	5	-	-	-	(4)	6	-	-	1	0
e	(5)	7	-	-	4	-	-	-	1	0
f	-	7	-	-	-	(6)	-	-	0	0
g	8	(7)	-	-	-	6	-	-	0	1
h	(8)	-	1	-	4	-	-	-	0	1

Se genera una nueva tabla reducida en donde la columna tomará primero el valor del estado estable si no hay entonces el estado transitorio y si no el guion, además se pone el valor de la salida, en los transitorios se pone como salida XX ya que esto no afecta pues el transitorio sólo dura unos nanosegundos. Se agrega una columna para poner la o las variables internas Q_t .

Por ejemplo, en los renglones agh se tiene -, 8 y (8), se toma el estado estable (8) y su salida que en este caso es 00, en la columna 100 se tiene -, - y 4 se toma el estado transitorio 4 y la salida XX, esto se repite para todas las columnas, en este caso sólo quedan dos renglones, por lo tanto, sólo se agrega una variable interna Q_t , la tabla queda de la siguiente manera:

PCA	000	001	010	011	100	101	110	111	Q_t
agh	(8) 01	(7) 01	(1) 00	-	4 XX	6 XX	2 XX	-	0
bcdef	(5) 10	7 XX	(3) 10	-	(4) 10	(6) 00	(2) 10	-	1

Ahora se genera la tabla de verdad a partir de la tabla fusionada y queda de la siguiente forma:

	PCA Q_t	$Q(t + \Delta t)$	D	I
(8)	0000	0	0	1
(5)	0001	1	1	0
(7)	0010	0	0	1
7	0011	0	X	X
(1)	0100	0	0	0
(3)	0101	1	1	0
-	0110	X	X	X
-	0111	X	X	X
4	1000	1	X	X
(4)	1001	1	1	0
6	1010	1	X	X
(6)	1011	1	0	0
2	1100	1	X	X
(2)	1101	1	1	0
-	1110	X	X	X
-	1111	X	X	X

La columna $Q(t + \Delta t)$ se genera a partir de la columna Q_t y de la primera columna que indica en ese renglón si es estado estable, transición o guion, si Q_t tiene un estado estable en ese renglón entonces $Q(t + \Delta t)$ permanece con el valor de Q_t , si tiene un estado transitorio entonces $Q(t + \Delta t)$ cambia al valor contrario que tenga Q_t y si tiene un guion entonces $Q(t + \Delta t)$ toma el valor X.

Ahora se obtienen las funciones lógicas usando mapas de Karnaugh.

$$Q(t + \Delta t) = F(P, C, A, Q_t)$$

PC \ AQ _t	00	01	11	10
00	0	1	0	0
01	0	1	X	X
11	1	1	X	X
10	1	1	1	1

$$Q(t + \Delta t) = A'Q_t + P$$

$$D = G(P, C, A, Q_t)$$

PC \ AQ _t	00	01	11	10
00	0	1	X	0
01	0	1	X	X
11	X	1	X	X
10	X	1	0	X

$$D = A'Q_t$$

PC \ AQ _t	00	01	11	10
00	1	0	X	1
01	0	0	X	X
11	X	0	X	X
10	X	0	0	X

$$I = C'Q_t'$$

Ya se puede implementar el circuito con las funciones lógicas obtenidas.

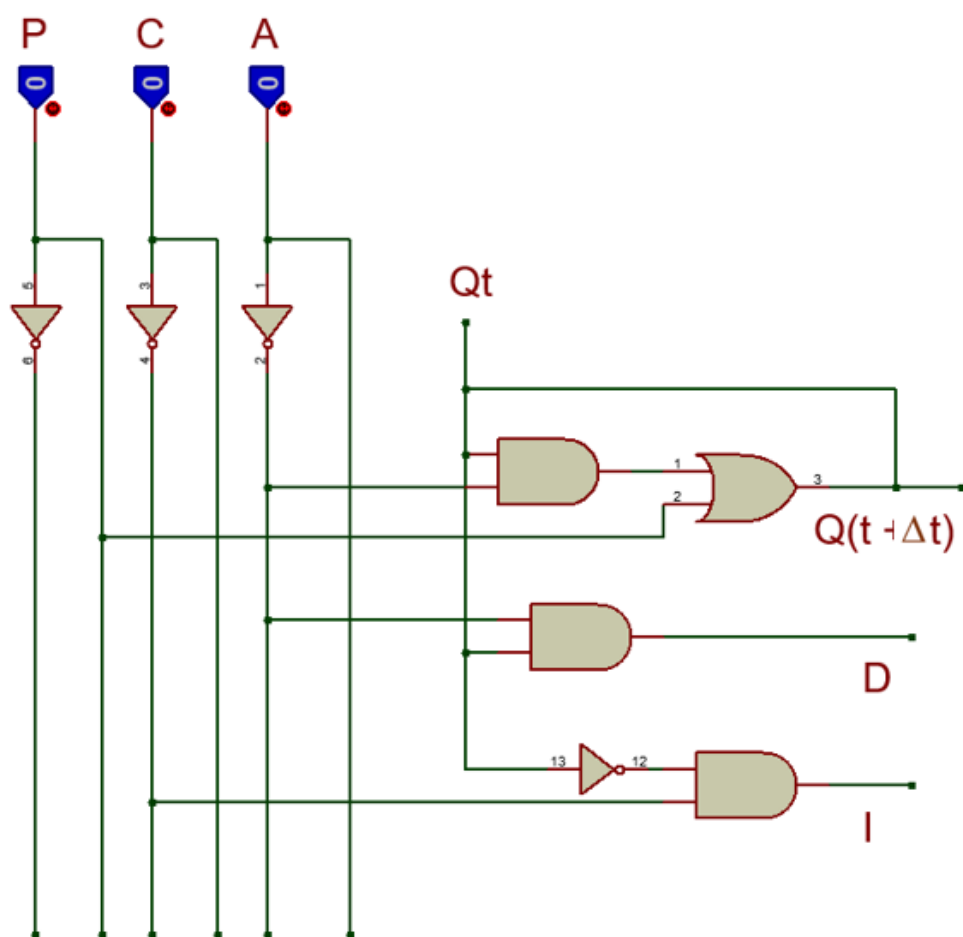


Figura 4.88 Circuito armado y simulado en Proteus.

4.6 Circuitos secuenciales básicos.

4.6.1 Registros.

Un registro es un circuito secuencial compuesto por flip-flops que almacena bits de datos de forma simultánea. Cada flip-flop almacena un bit, por lo que un registro de n bits tiene n flip-flops. Los registros se utilizan ampliamente en sistemas digitales para guardar, transferir y manipular datos, se les llama también registros de corrimiento o desplazamiento debido a que se pueden mover los datos en los registros de derecha a izquierda o de izquierda a derecha y también sirven para convertir datos de serie a paralelo y de paralelo a serie.

Los registros se clasifican en 4 grupos de acuerdo con la forma en la que cargan y sacan los datos, estos grupos son:

- a) Entrada en paralelo y salida paralelo (PIPO).
- b) Entrada serie y salida serie (SISO).
- c) Entrada en paralelo y salida serie (PISO).
- d) Entrada en serie y salida paralelo (SIPO).

Registro de entrada paralelo y salida en paralelo (PIPO).

Este tipo de registros requieren de sólo un pulso de reloj para transferir todos los datos de entrada a la salida, son muy utilizados en sistemas con microprocesadores y en computadoras.

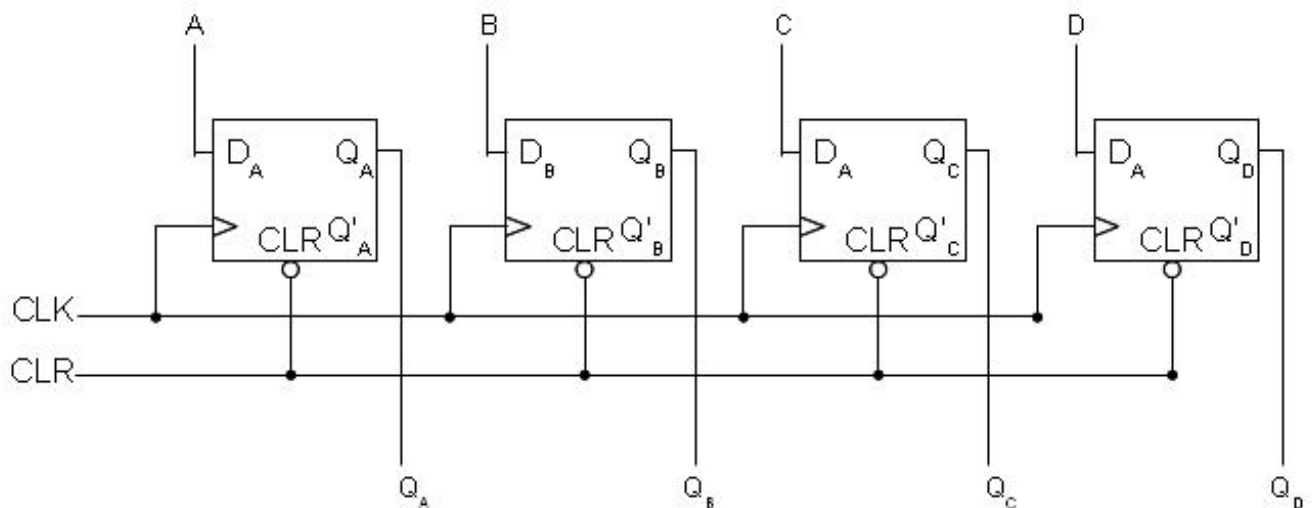


Figura 4.89 Registro de 4 bits carga paralelo salida paralela.

Un ejemplo de este tipo de registros es el 74374 es de 8 bits con inversor a la salida de los flip flops y su habilitador.

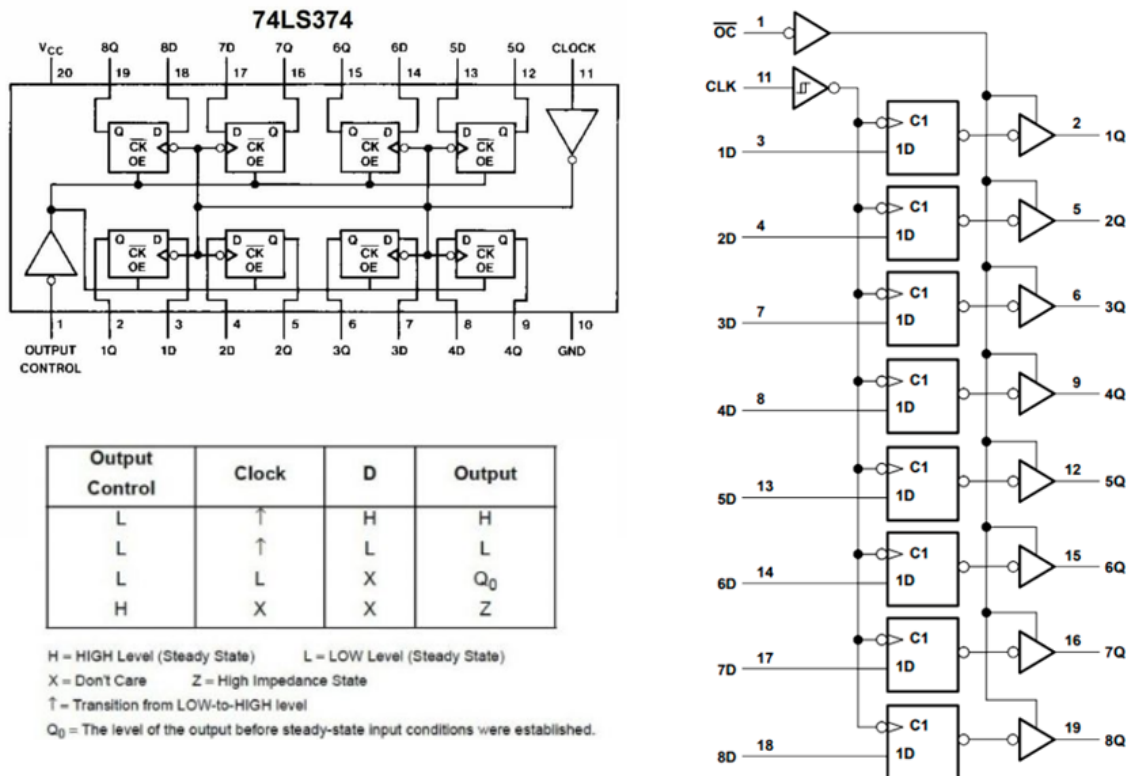


Figura 4.90 Registro octal de carga en paralelo salida en paralelo 74374.

Registros de entrada serie salida serie (SISO).

Este tipo de registro de desplazamiento permite ingresar y extraer datos bit a bit, se requiere de una señal de reloj para impulsar los bits, un pulso de reloj por cada bit. Está compuesto por una cadena de flip-flops conectados en serie, son muy utilizados en:

Comunicación serie:

Transmitir datos bit a bit por una sola línea (por ejemplo, UART, SPI).

Conversión serie-paralelo y viceversa:

Parte de sistemas que convierten flujos de datos entre formatos.

Ahorro de pines en hardware:

Se puede enviar un dato de varios bits usando solo una línea de datos.

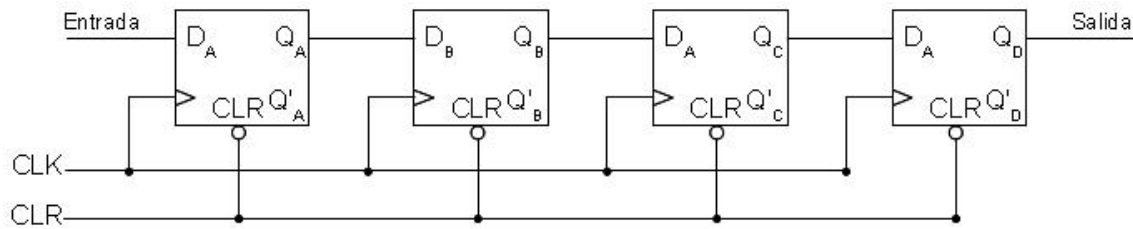


Figura 4.91 Registro de entrada serie salida serie.

Un ejemplo de este tipo de registros es el 74LS91 registro de 8 bits, requiere de un flanco positivo de reloj por cada bit, recuerde que los flip flops siempre tienen de un valor inicial cuando se energizan por lo que los primeros 8 bits que salen del registro son desconocidos.

Inputs AT _n		Outputs AT _n +8	
A	B	Q _H	Q _L
H	H	H	L
L	X	L	H
X	L	L	H

T_n = Tiempo de referencia de reloj en bajo

T_n+8 = Después de 8 bits con transiciones de bajo a lato del reloj.

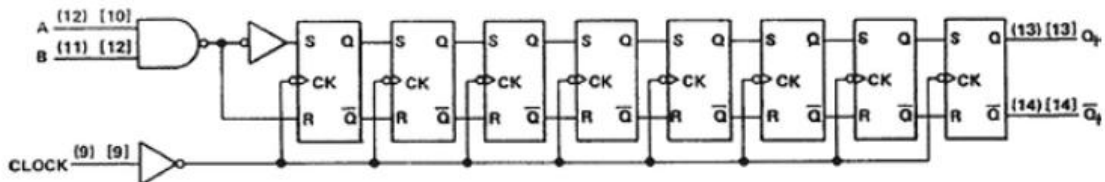
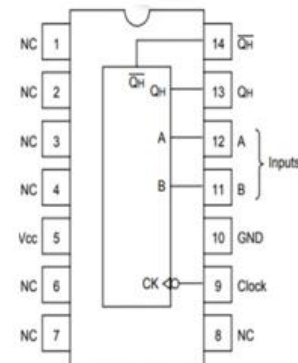


Figura 4.92 Registro octal 74LS91 entada serie salida serie y su tabla de verdad.

Registro de entada paralelo salida serie (PISO).

En estos registros el dato de n bits se carga a los flip flops con una terminal que tiene esta función, en seguida se inicia la salida de los datos en la terminal de dato de salida serie, un bit por cada pulso de reloj ya sea flanco positivo o flanco negativo, el primer flip flop del arreglo tiene su entrada D libre por lo que también podría ser un registro de entrada serie y salida serie si se deshabilita la línea de carga paralela.

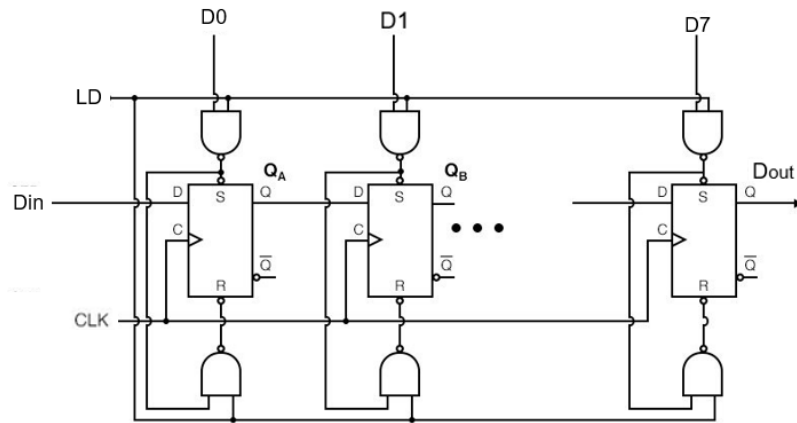


Figura 4.93 Registro con entrada paralelo y salida serie.

Un ejemplo de este Registro es el circuito integrado 74LS165, es de 8 bits y cumple con lo mencionado anteriormente.

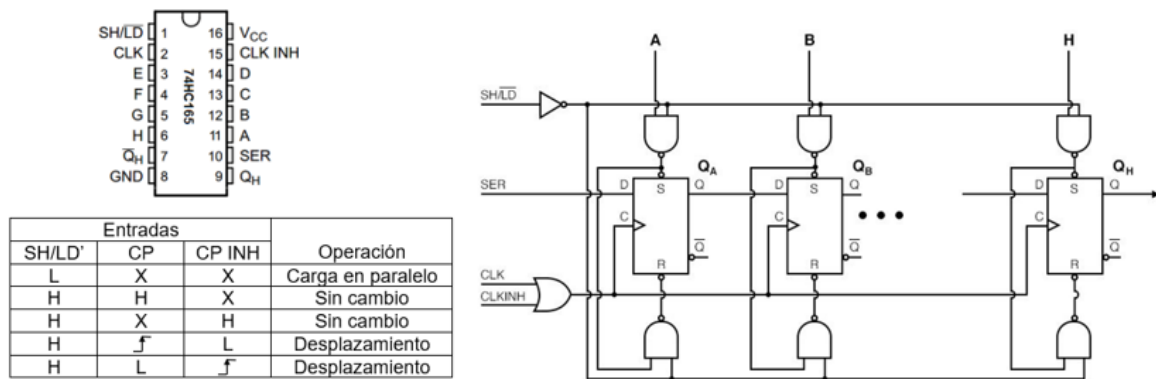


Figura 4.94 Registro de 8 bits carga en paralelo salida en serie.

Ejemplo

Dibujar la carta de tiempo para los datos paralelos 10011010 y 01010011 manteniendo DS(SER) en cero y CP INH en cero.

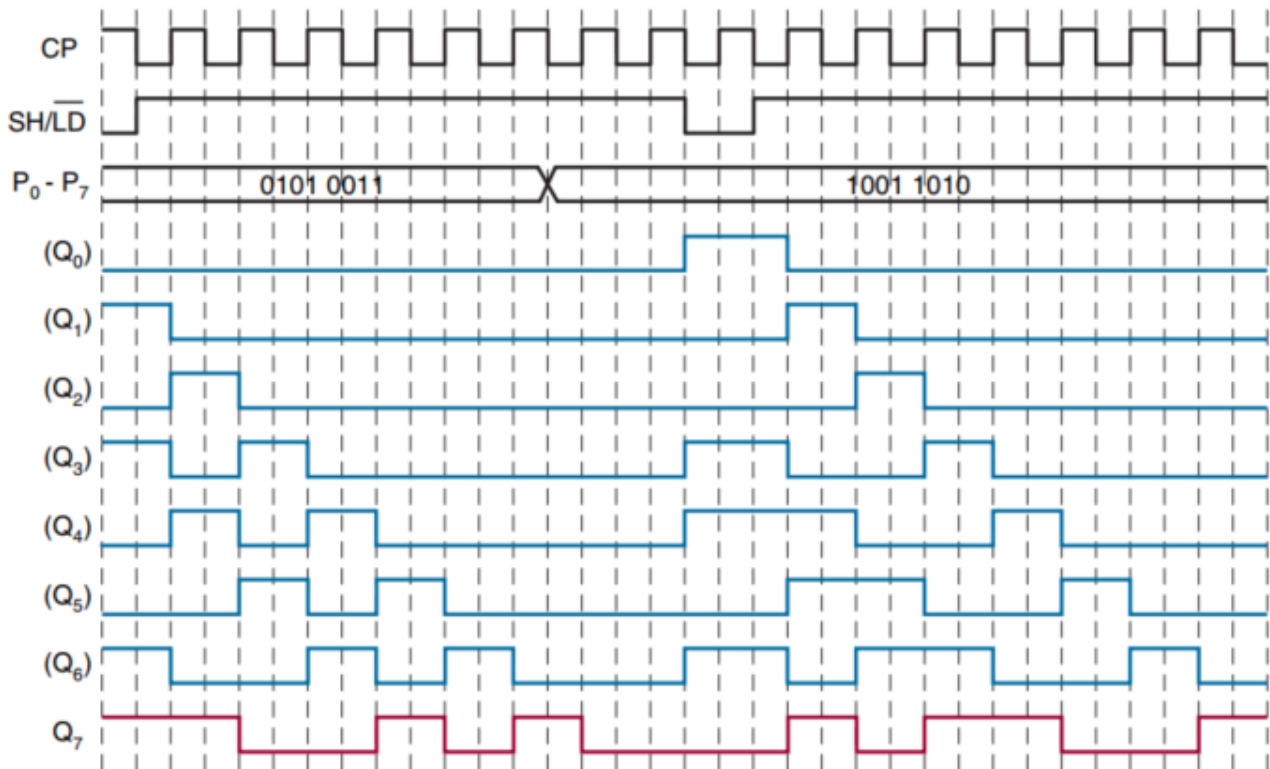


Figura 4.95 Carta de tiempo para los datos 10011010 y 01010011. [14]

Registro de entrada en serie salida en paralelo (SIPO).

En este tipo de registros se carga en dato en los flip flops bit por bit, un bit por cada pulso de reloj, el dato de 8 bits se puede ver en cualquier momento, sin embargo, para leer el dato correcto se debe leer al término del octavo pulso de reloj.

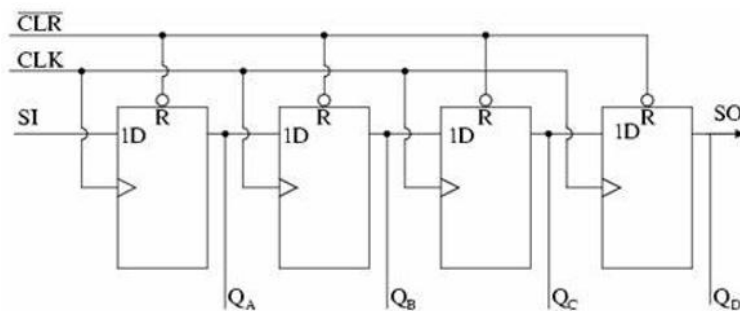


Figura 4.97 Registro de entrada serie salida paralelo.

Se puede observar en la figura que en este tipo de registros también podemos tener la salida serie leyendo el dato en el último bit.

Un ejemplo de este tipo de registros es el circuito integrado 74LS164, es un registro de 8 bits en el que siempre se tiene presente las salidas de los 8 flip flops, se puede tomar el último bit Q7 para leer el dato de salida en forma serie.

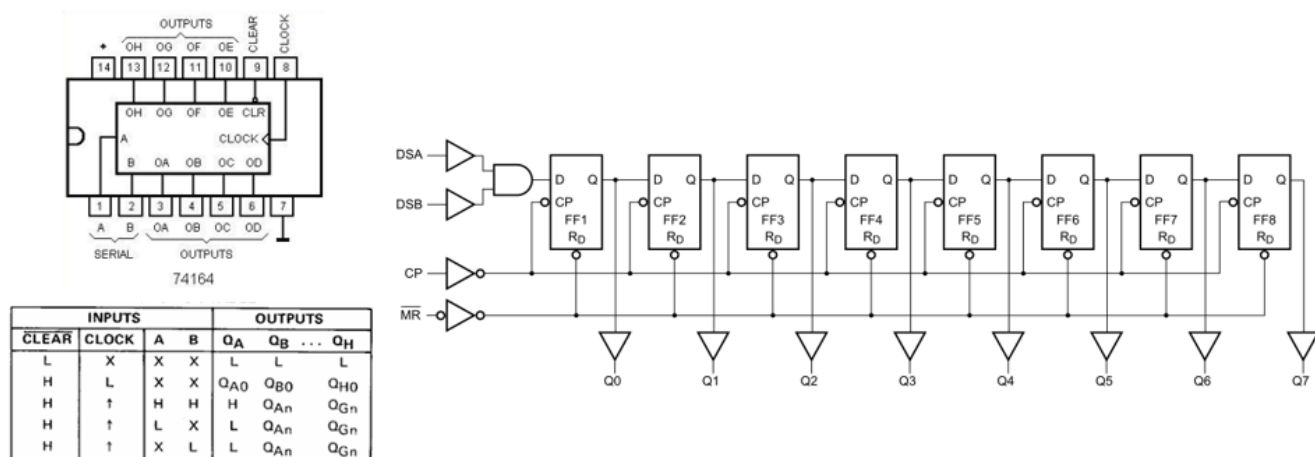


Figura 4.97 Registro de 8 bits 74164 entrada en serie salida en paralelo.

Ejemplo

Utilizando el registro 74164 diseñe un circuito para que al presionar un botón inicie el encendido de ocho cargas una por cada pulso de reloj y se detenga al estar las ocho prendidas, al presionar nuevamente el botón se irán apagando una por cada pulso de reloj, al apagarse la última presionar el botón para detener el proceso.

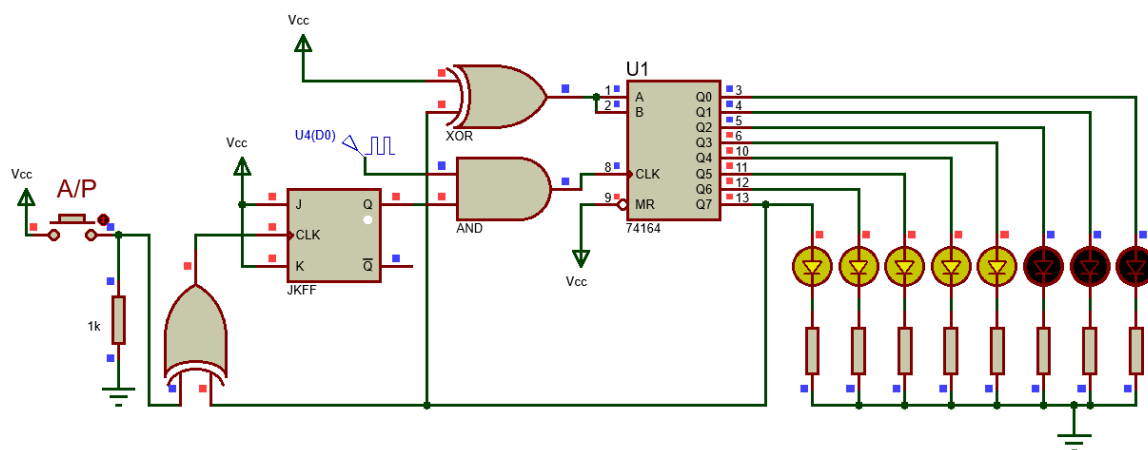


Figura 4.98 Circuito para activar prender y apagar cargas en forma secuencial.

En el circuito al inicio el FF JK está bloqueando la señal de reloj del sistema, se requiere presionar el botón para iniciar el proceso, la compuerta XOR que está conectada a las entradas A y B es la que permite en el arranque que se carguen puros unos hasta prender todas las cargas, al llegar a la última carga la XOR cambia de estado y ahora pone ceros a los datos A y B pero al mismo tiempo el FF JK cambia de estado y bloquea los pulsos de reloj del sistema para impedir que se empiecen a apagar las cargas, se requiere de presionar nuevamente el botón para permitir los pulsos de reloj y comience a apagarse las cargas, al apagarse la última se presiona el botón para parar el proceso, la XOR conectada al reloj del FF genera un flanco positivo cuando se presiona el botón o cuando se prende la última carga haciendo que el FF JK este alternando el valor de Q para deja pasar los pulsos del reloj del sistema o bloquearlos.

4.6.2 Contadores

Un contador binario es un circuito secuencial digital que cuenta pulsos de reloj representando el conteo en forma binaria. Está compuesto por una serie de flip-flops conectados entre sí, y con cada pulso de reloj el contador cambia su estado para representar el siguiente número binario en la secuencia.

Los contadores binarios se pueden clasificar de la siguiente manera:

Contadores asíncronos.

Contadores síncronos.

Contador de anillo.

Contador Johnson.

Contador asíncrono.

Este tipo de contadores también llamados de rizo tienen la característica que la señal de reloj sólo le llega al primer flip flop y la señal de reloj de los demás flip flops es la salida del flip flop anterior al que está conectado, ya sea la Q o la Q'.

Para este tipo de contador se usa el flip flop JK con sus entradas a 1 para que cada que llegue un flanco de reloj en este caso negativo, invierta el valor de la salida.

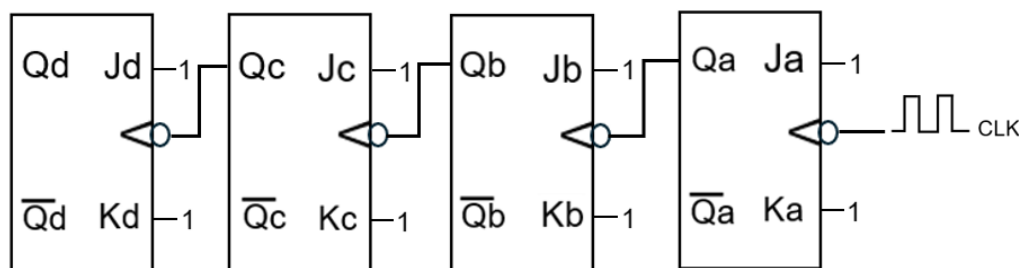


Figura 4.99 Contador asíncrono o de rizo ascendente de 4 bits.

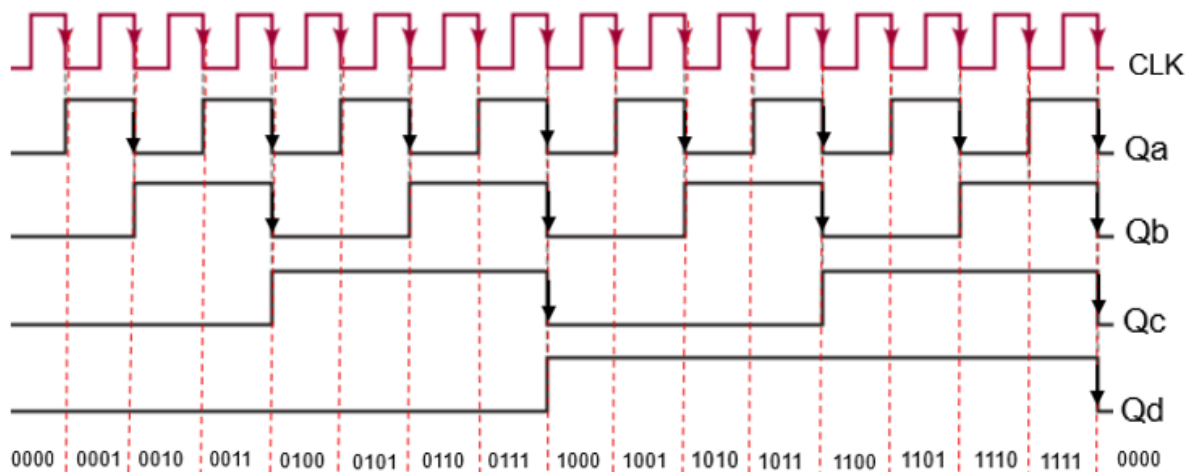


Figura 4.100 carta de tiempo de respuesta de los flip flops a la señal de reloj.

En la carta de tiempo se puede observar que por cada flanco negativo de reloj del primer flip flop la salida Qa invierte su valor debido a que todas las entradas de los flip flops están a 1, esta salida se toma como entrada del reloj para el segundo flip flop por lo que cada que cambie de alto a bajo hará que la salida del segundo flip flop Qb invierta su valor, esta salida es tomada como entrada de reloj del tercer flip flop y cada que cambie de alto a bajo hará que el tercer flip flop alterne su salida Qc, esta salida es la señal de reloj del cuarto flip flop y cada que cambie de alto a bajo alternara el valor de su salida Qd.

Se observa que la salida se comporta como un contador ascendente de 4 bits el cual tiene 16 estados diferentes ya que después se repite la secuencia.

A estos contadores se les conoce como contadores de modulo 2^n (mod n) donde n es el número de bits (flip flops) que tenga el contador, en este caso es de 4 bits por lo que tiene 16 estados diferentes al llegar al último estado se repite la secuencia.

Básicamente el contador es un circuito secuencial por lo que se puede representar su comportamiento en un diagrama de estados de la siguiente manera:

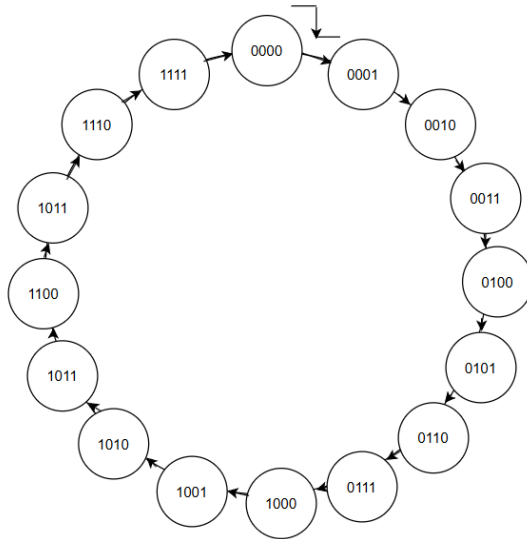


Figura 4.101 Diagrama de estados para el contador mod 16.

Retardo de los contadores asíncronos o de rizo.

Cada flip flop que compone el contador tiene un retardo de propagación t_{pd} en responder a la señal de reloj, como la señal de reloj sólo le llega al primer flip flop se da este retardo su tiempo lo da el fabricante, la señal de reloj del segundo flip flop es la salida Q_a del primer flip flop por lo que se vuelve a dar este retardo en el segundo flip flop y esto se repite para cada flip flop por lo que los tiempos de retardo se van sumando, lo que es un inconveniente para este tipo de contadores ya que para un número grande de contadores la frecuencia que pueden manejar es menor, el tiempo total de retardo en la propagación se da por $N \times t_{pd}$ donde N es el número de flip flops del que está compuesto el contador.

El tiempo de la señal de reloj debe ser mayor que $N \times t_{pd}$, por lo tanto, la frecuencia máxima a la que responde este tipo de contadores viene dada por:

$$f_{\text{máx}} = \frac{1}{N \times t_{pd}}$$

Por ejemplo, si se tiene un contador asíncrono de 4 bits y cada flip flop tiene un tiempo de retardo de propagación de 30 ns, la frecuencia máxima que puede manejar el contador es:

$$f_{\text{ax}} = (1/(4 \times 30 \text{ ns})) = 8.33 \text{ MHz}$$

Si el contador es de 8 bits, la frecuencia máxima sería:

$$f_{\text{ax}} = (1/(8 \times 30 \text{ ns})) = 4.16 \text{ MHz.}$$

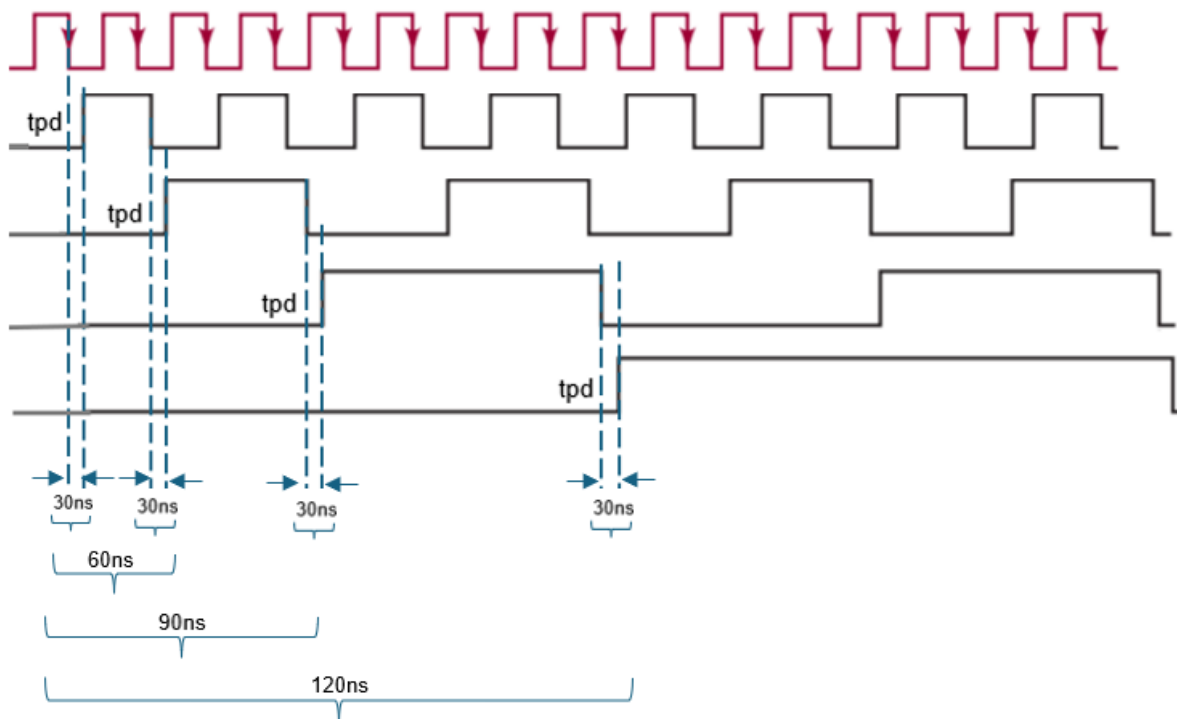


Figura 4.102 Retraso de propagación de los flip flops.

Contador asíncrono descendente.

Para hacer un contador descendente sólo hay que conectar las salidas negadas de los flip flops a la entrada de reloj a partir del segundo flip flop en adelante ya que el primer flip flop recibe la señal de reloj.

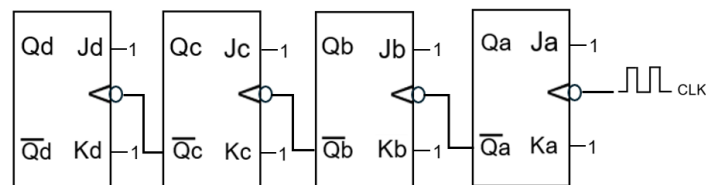


Figura 4.103 Contador asíncrono descendente de 4 bits.

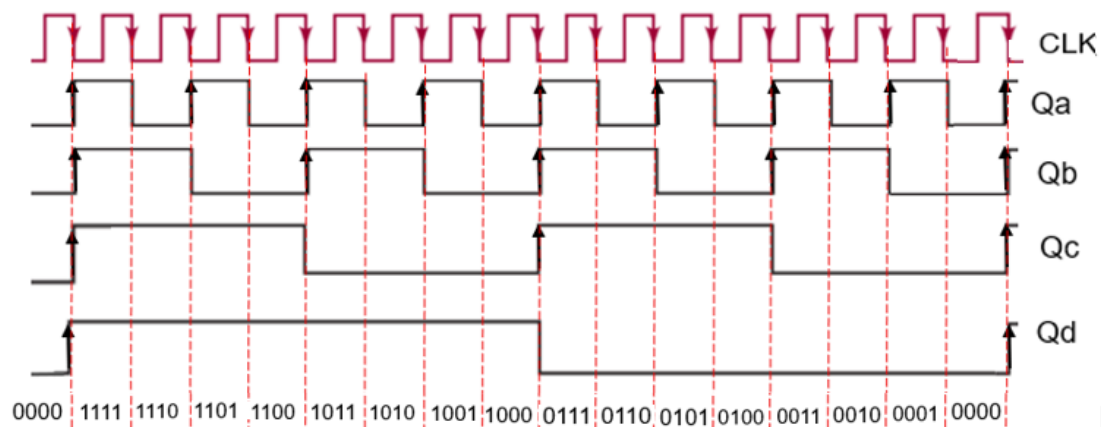
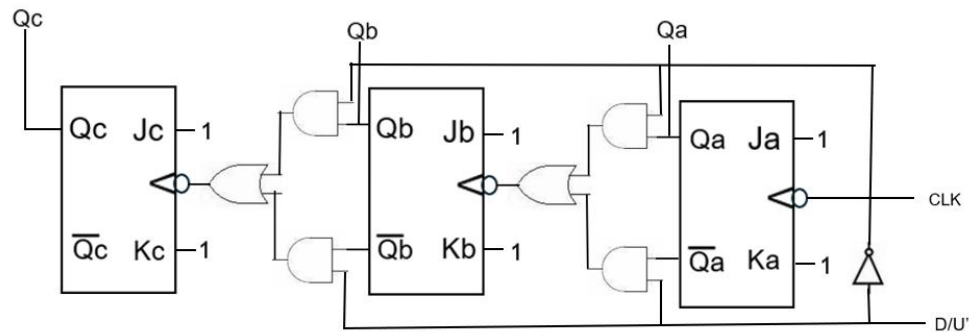


Figura 4.104 Carta de tiempo para el contador descendente asíncrono de 4 bits.

Contador ascendente descendente asíncrono mod n.



Como se revisó en los ejemplos anteriores, si la señal de reloj a partir del segundo flip flop es la salida no negada Q, entonces será un contador ascendente, pero si en Q' entonces será un contador descendente. Hacerlo ascendente descendente se logra eligiendo que salida se conectara al reloj a partir del segundo flip flop y se hace a través de las compuertas AND y las compuertas OR, si la entrada D/U' es uno se habilitan las compuertas que dejan pasar la Q' y contará en forma descendente, si la línea D/U' es 0 entonces habilitara las compuertas AND que deja pasar las salidas Q y contará en forma ascendente.

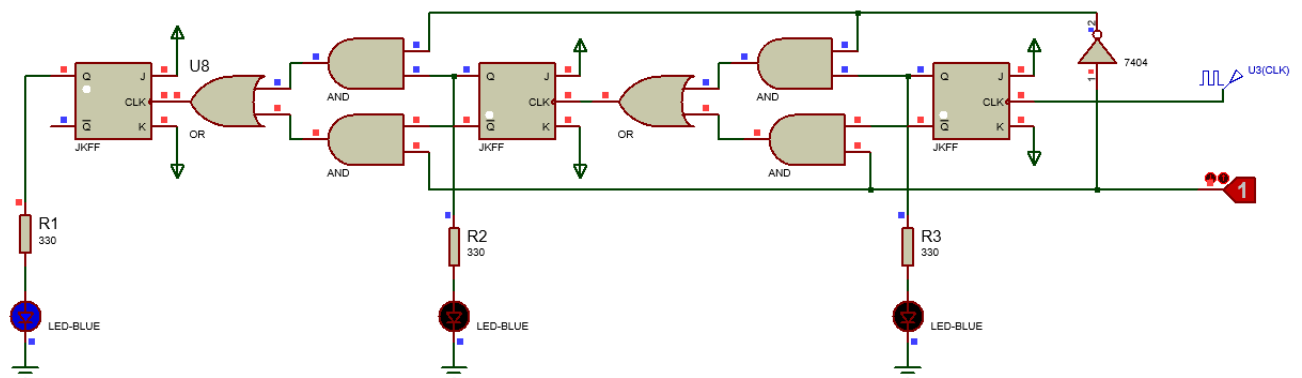


Figura 4.106 Contador ascendente descendente Simulado en Proteus.

Contadores con modulo diferente de 2^n .

Para diseñar contadores con módulo diferente de 2^n se trabajan con flop flops que tengan las entradas asíncronas clear para poder restablecer el contador en la cuenta que se requiera.

Por ejemplo, si se requiere diseñar un contado ascendente que solo haga la cuenta de 0 a 5 (se requieren 3 bits y sería 8 estados) se toma la cuenta inmediata superior como estado transitorio para reiniciar el contador en este caso sería la cuenta 6, si el contador es descendente se toma como estado transitorio el inmediato inferior para reiniciar el contador, si el contador fuera de 7 a 4, se tomaría la cuenta 3 como el estado transitorio.

Ejemplo

Se requiere un contador de 0 a 5. (módulo 6 ya que tiene 6 estados diferentes)

Para este contador se requiere de 3 bit (3 flip flops), lo que daría 8 estados diferentes, lo que se tiene que hacer es tomar el estado transitorio que en este caso es el 6 110 y se busca que combinación de esos bits no se repiten en los conteos anteriores y los usamos para regresar el contador a cero, si es necesario se ocupan los 3 bit para el reinicio.

Cuando aparece el conteo del transitorio no lo logramos observar debido que al momento de aparecer reinicia todos lo flip flops, es decir sólo aparece en lo que tarda en reiniciar el contador de allí que se llame estado transitorio.

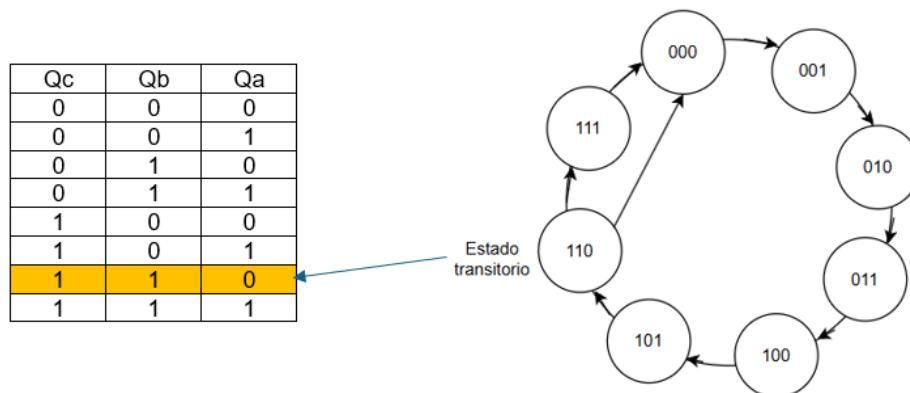


Figura 4.107 diagrama de estados para el contador modulo 6.

Al observar el estado transitorio en la tabla del conteo vemos que los bits Qb =1 y Qa =0 se repiten en la cuenta anterior 010, revisamos los bits Qc =1 y Qb =1 y vemos que no se repiten en ninguna cuenta anterior, por lo tanto, usamos esta combinación para regresar el contador a 0, si los bits se repitieran entonces tendríamos que usar los tres bits para regresar el contador a 0.

Figura 4.108 Circuito para el conteo de 0 a 5.

Figura 4.109 Carta de tiempo para el contador de 0 a 5.

Figura 4.110 simulación del circuito en Proteus.

Contadores síncronos.

Este tipo de contadores se distinguen por que la señal de reloj la reciben todos los flip flops al mismo tiempo por lo que el tiempo de retardo en la propagación sólo será la de un flip flop, por lo tanto, la frecuencia que pueden manejar es muy alta sin importar el número de bits del contador, prácticamente están limitados por los datos que da el fabricante. Como la señal de reloj la reciben todos los flip flops al mismo tiempo, también todos cambian al mismo tiempo de estado por lo que es necesario controlar las entradas J y K para que sólo cambien los flip flops que sean necesarios para logra hacer la cuenta binaria correcta, si J y K están a 0 cuando llegue la señal de reloj el flip flop no cambiará de estado, solo cuando tengan J =1 y K =1 en sus entradas, se aprovecha esta condición para lograr hacer el conteo adecuado, si observamos la tabla de conteo para un contador de 4 bits vemos que el primer bit siempre cambia de estado con cada pulso de reloj, para que el segundo bit cambie de estado debe haber un 1 en el primer bit antes de que llegue la señal de reloj, para que cambie el tercer bit, los dos primeros bits deben estar a 1 antes de que llegue la señal de reloj y para que cambie el cuarto bit se requiere que antes de que llegue el pulso de reloj los tres bits anteriores deben estar en 1.

Qd	Qc	Qb	Qa
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

Figura 4.111 Tabla de conteo binario para 4 bits.

De acuerdo con la tabla anterior debemos asegurar que a partir del segundo flip flop la entrada J y K sean 1 si el bit anterior es 1, j y K del tercer flip flop deben tener 1 si los dos bits anteriores son 1 y las entradas J y K del cuarto flip flop deben ser 1 si los tres bits anteriores son 1, esto se logra manejando compuertas AND a partir del segundo flip flop como se muestra en la siguiente figura:

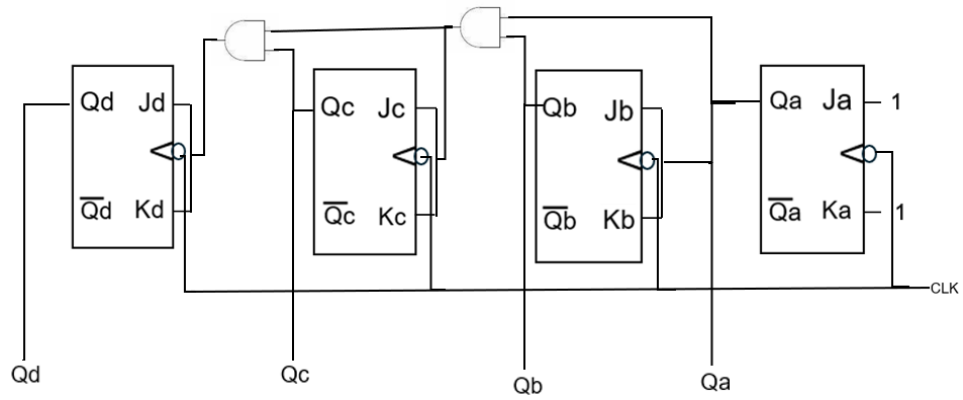


Figura 4.112 Contador síncrono de 4 bits.

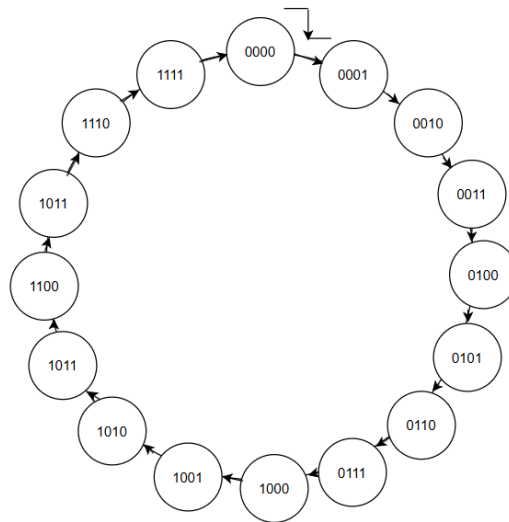


Figura 4.113 Diagrama de estados para el contador síncrono de 4 bits.

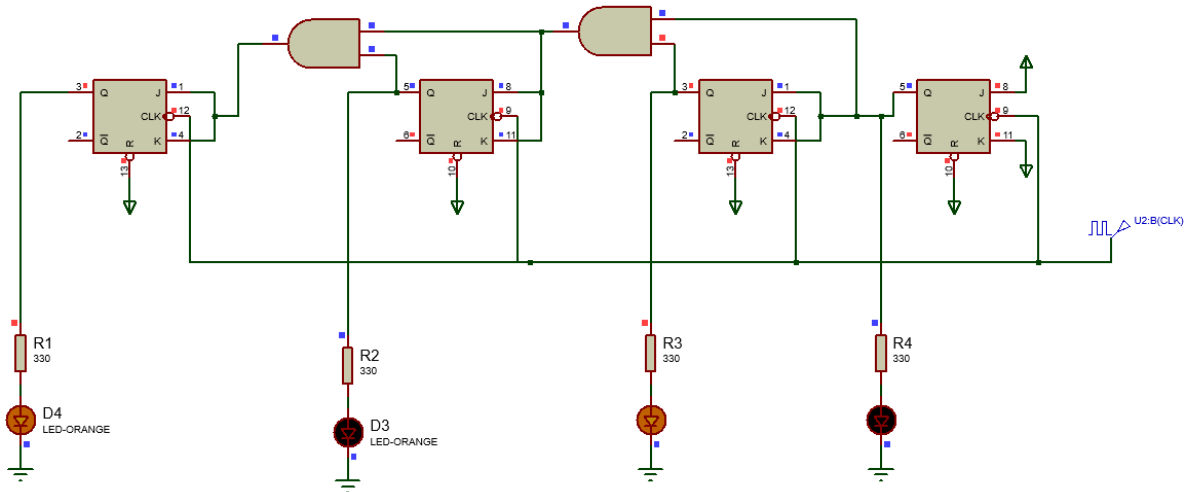


Figura 4.114 Contador síncrono de 4 bits simulado en Proteus.

Contador síncrono descendente.

Es igual que el ascendente, sólo que las salidas que controlarán las entradas J y K de los flip flops son las salidas negadas, Q'a, Q'b, Q'c y Q'd.

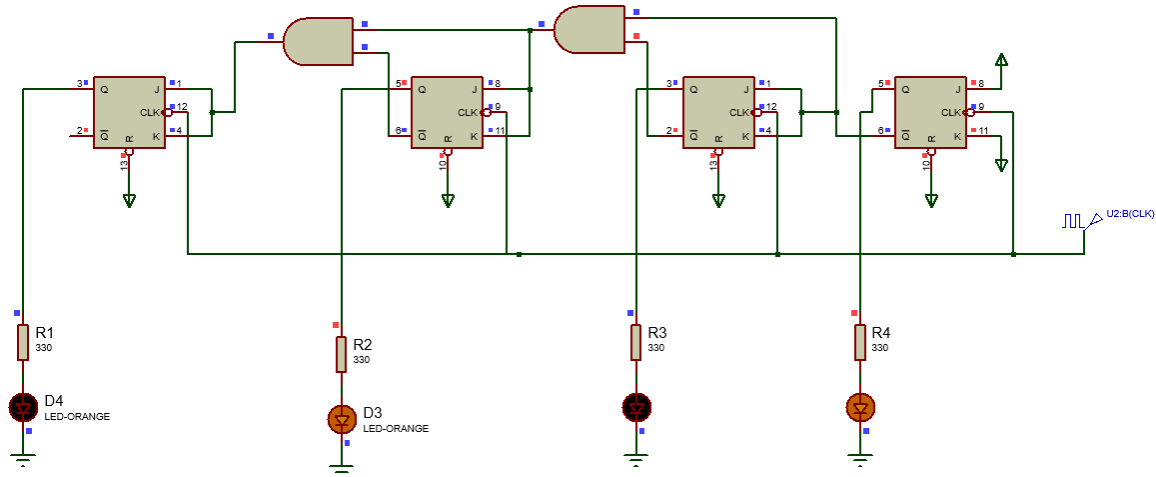


Figura 4.115 contador descendente de 4 bits simulado en Proteus.

Contadores con módulo diferente de 2^n .

Para diseñar un contador con un módulo diferente de 2^n se hace lo mismo que con los contadores asíncronos, se toma el estado transitorio para reiniciar el contador.

Ejemplo diseñar un contador síncrono que cuente del 0 al 10, módulo 11 ya que son once los estados diferentes por los que pasa el contador.

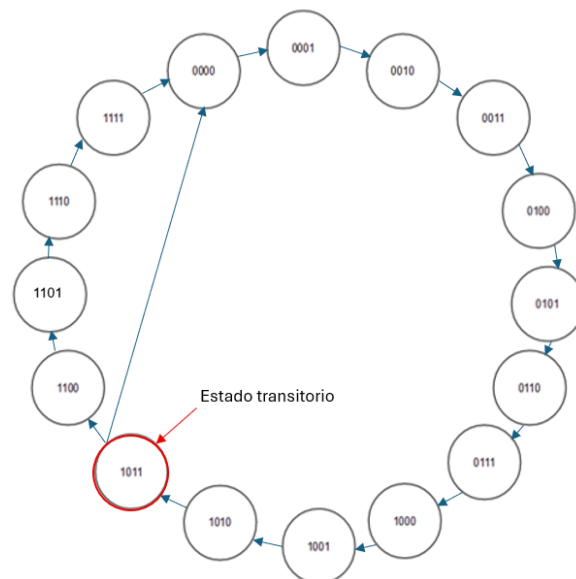


Figura 4.116 Diagrama de estados para el contador síncrono de 0 al 10.

Qd	Qc	Qb	Qa
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

Figura 4.117 Tabla del conteo para 4 bits con el estado transitorio marcado en azul.

En la tabla se puede observar que los bits Qa, Qb y Qd que están en 1, no se repiten en estados anteriores por lo que se pueden utilizar para reiniciar el contador.

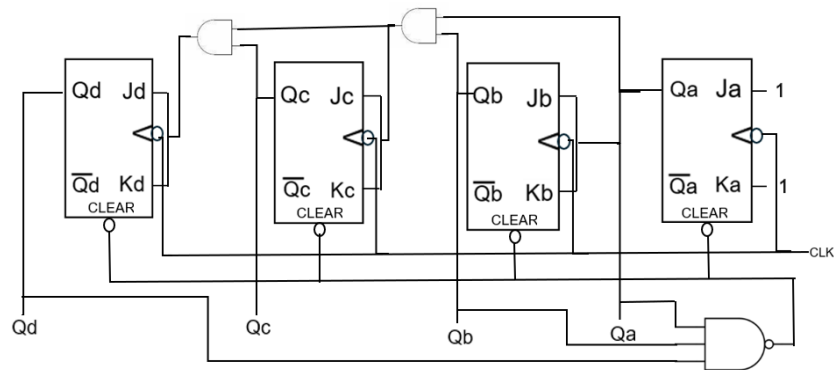


Figura 4.118 Contador de 4 bits síncrono de 0 al 10, módulo 11.

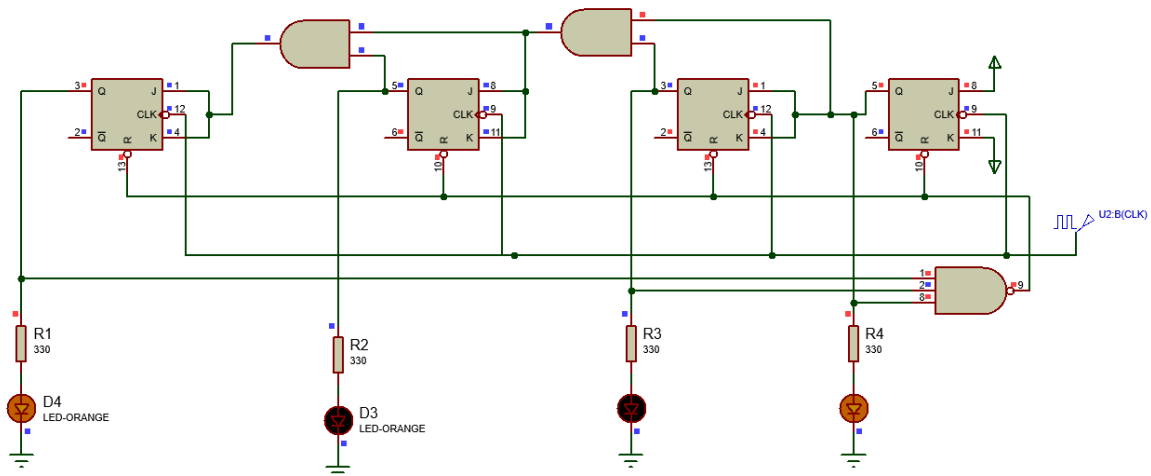


Figura 4.119 Contador de 4 bits módulo 11 simulado en Proteus.

Contador de anillo.

Un contador de anillo es un tipo de contador secuencial utilizado en sistemas digitales. Está formado por un conjunto de flip flops normalmente del tipo D conectados en serie, donde la salida de uno se convierte en la entrada del siguiente, y el último flip-flop se conecta de vuelta al primero, formando un circuito cerrado o anillo, el primer flip flop deberá tener un 1 al inicio de su funcionamiento y los demás en 0, por este motivo un contador de anillo tiene n estados donde n es el número de flip flops que lo conforman.

El funcionamiento es el siguiente:

Al inicio se pone en 1 el primer flip flop y los demás en 0, la señal de reloj que es común a todos desplaza por cada pulso los bits, el primer bit pasa al segundo flip flop, el segundo al tercero y así hasta el último flip flop, este regresa su bit al primer flip flop repitiéndose el ciclo.

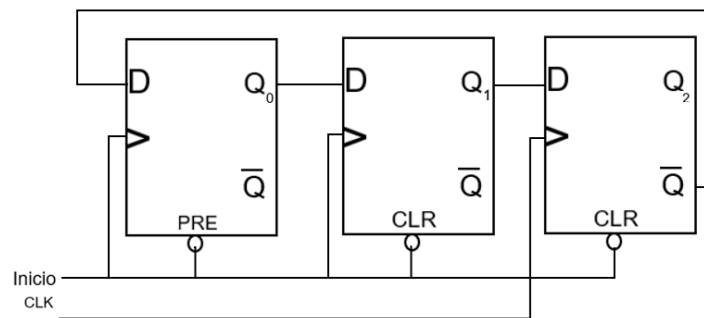


Figura 4.120 Contador de anillo de 3 bits.

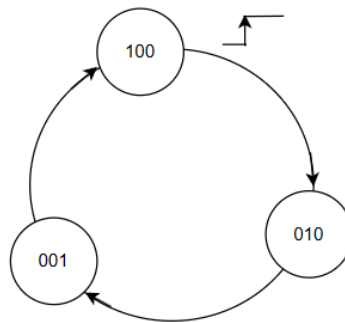


Figura 4.116 Diagrama de estados para el contador de anillo de 3 bits.

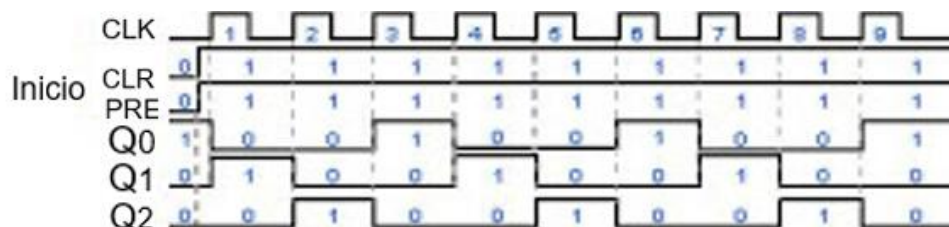


Figura 4.121 Carta de tiempo para el contador de anillo de 3 bits.

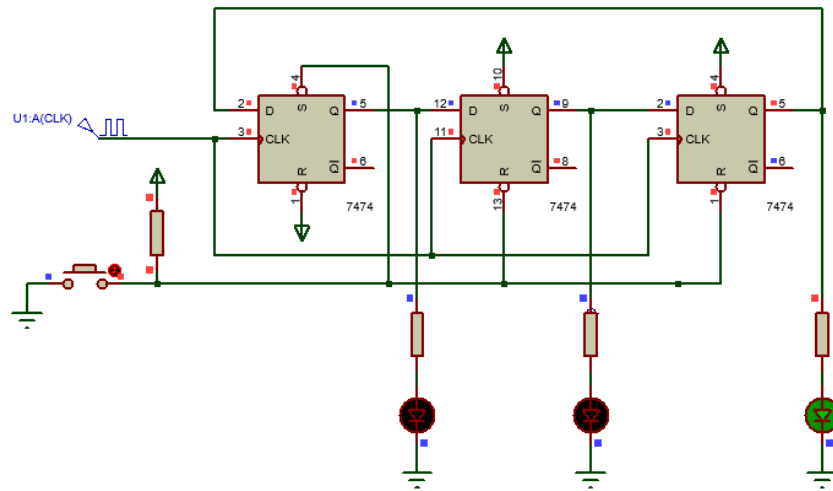


Figura 4.122 Contador de anillo simulado en Proteus.

Contador Johnson.

Este contador es una variante del contador de anillo, la diferencia es que el último flip flop regresa al primer flip flop la salida negada, este contador tiene $2n$ estados diferentes, si el contador es de 3 bits entonces tendrá 6 estados diferentes.

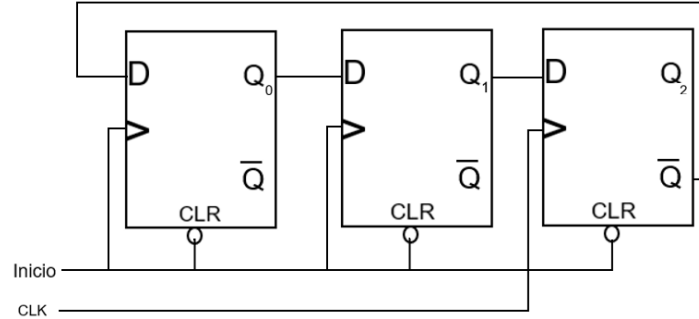


Figura 4.123 Contador Johnson de 3 bits.

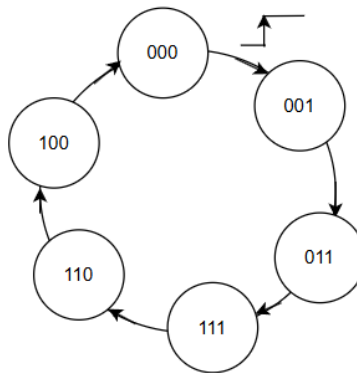


Figura 4.124 Diagrama de estados para el contador Johnson.

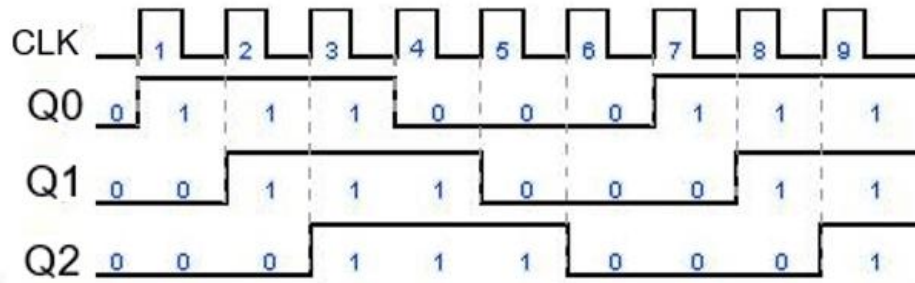


Figura 4.125 Carta de tiempo para el contador Johnson.

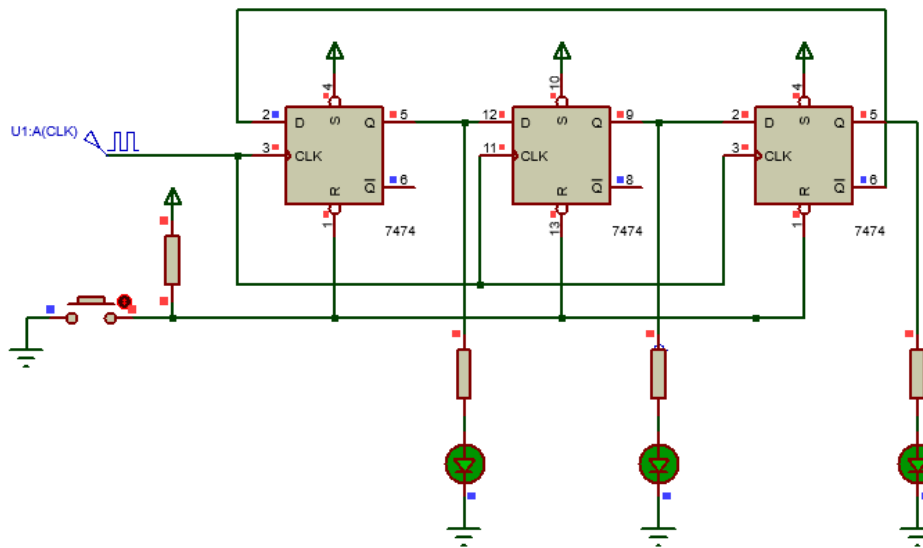


Figura 4.126 Contador Johnson simulado en Proteus.

Como ejemplo del manejo de contadores en circuitos integrados como el contador 74192, decodificadores BCD a 7 segmentos y displays de 7 segmentos se muestra el diseño de un reloj de tiempo real en horas minutos y segundos en un formato de 24 horas.

El primer contador (unidades de segundo) recibe el pulso de un segundo y contará se 0 a 9, en el siguiente pulso se regresa a 0 pero incrementa el segundo contador (decenas de segundos) este sólo contará hasta 5 en el siguiente impulso que le llegue se regresa a 0 e impulsa al tercer contador(unidades de minutos) que contará de 0 a 9 y al regresar a 0 impulsará al cuarto contador(decenas de minutos) que contará de 0 a 5 en el siguiente impulso recibido se regresa a 0 e impulsa al quinto contador(unidades de horas) este contará de 0 a 9 si decenas de horas es 0 o 1 pero si es 2 sólo contará hasta 3 en el siguiente impulso se regresa a 0 y pone decenas de horas(sexto contador) en 0 que indicará la hora 00:00:00.

La primera compuerta and conectada al contador de decenas de segundos es la encargada de regresar el contador a 0 (módulo 6) lo mismo que la compuerta and conectada a decenas de minutos y la tercera compuerta and se encarga de regresar unidades de hora y decenas de hora a cero cuando se presente el impulso en la hora 23:59:59.

4.6.3 Memorias

En los sistemas digitales se tiene una gran ventaja en el almacenamiento y manejo de grandes cantidades de datos a diferencia de los sistemas analógicos.

En los sistemas digitales, las memorias son componentes esenciales encargados de almacenar información, ya sea temporal o permanentemente. Las memorias se implementan en circuitos integrados con diferentes tecnologías, lo que permite tener una gran cantidad de espacio para el almacenamiento de datos, bajo consumo de energía y velocidad muy altas de operación lo que las hace adecuadas para distintas aplicaciones, desde microcontroladores hasta procesadores avanzados.

Existen dos grandes grupos de memorias en circuitos integrados de acuerdo con su forma de operar estos son:

memorias volátiles.

memorias no volátiles.

Las memorias volátiles, pierden su información cuando se interrumpe la alimentación, por lo que se utilizan en los sistemas que requieren manejar y actualizar información durante su funcionamiento sin importar que suceda al apagar el sistema, una de ellas es la RAM (Memoria de Acceso Aleatorio),

Las memorias no volátiles, como la ROM, PROM, EEPROM y Flash, conservan la información incluso sin energía, lo cual es fundamental para almacenar programas y configuraciones permanentes.

Terminología utilizada en el manejo de memorias.

Bit.

Digito binario que puede ser un 0 o un uno, normalmente implementado en una celda de memoria como el flip flop.

Nibble.

Conjunto de cuatro bits que puede representar 16 valores diferentes normalmente usados en el manejo de operaciones en BCD.

Byte.

Conjunto de 8 bits, es uno de los términos más utilizados en el manejo de la información en los microprocesadores y microcontroladores, un Byte es igual a dos nibbles llamado nibble alto (NH) a los 4 bits más significativos del byte y nibble bajo (NL) a los cuatro bits menos significativos del byte.

Word.

Es una palabra formada por 16 bits, por lo tanto, es igual a dos bytes (byte alto BH a los 8 bits más significativo y byte bajo BL a los 8 bits menos significativos) o cuatro nibbles.

Bus.

Línea o conjunto de líneas a través de las cuales se manda o recibe información, estos buses pueden ser de entrada, salida o bidireccionales y cualquiera de ellos pueden ser manejado en tercer estado si el sistema lo requiere.

Tiempo de acceso.

Es el tiempo que se requiere para hacer una lectura de la memoria, es decir lo que tarda la memoria en poner el dato en la salida cuando se le da una dirección y se le pide esta operación, este tiempo lo da el fabricante y es muy importante tomarlo en cuenta cuando la memoria trabaja con dispositivos de alta velocidad como los microprocesadores.

Operación de lectura.

Es la acción de sacar un dato almacenado en la memoria en una dirección específica.

Operación de escritura.

Es la acción de introducir un dato en la memoria en una dirección específica.

Buses de una memoria.

La memoria tiene 3 buses que son:

Bus de dirección.

Es un bus de entrada que permite decirle a la memoria en que dirección se quiere leer o escribir un dato, el número de líneas de este bus depende de la capacidad de almacenamiento de la memoria y normalmente se nombran las líneas con la letra A y un subíndice ($A_0, A_1 \dots A_n$).

Bus de datos.

Este bus es bidireccional para permitir meter o sacar un dato en la memoria, el número de líneas puede ser 1, 4, 8 o 16 normalmente, la dirección de bus depende la operación que se realice, si es una lectura el bus será de salida y si es una escritura el bus será entrada. Estos buses son en la mayoría de memoria de tercer estado es decir se pone en alta impedancia si no se utiliza.

Bus de control.

Son líneas de entrada que le indican a la memoria que operación debe realizar.

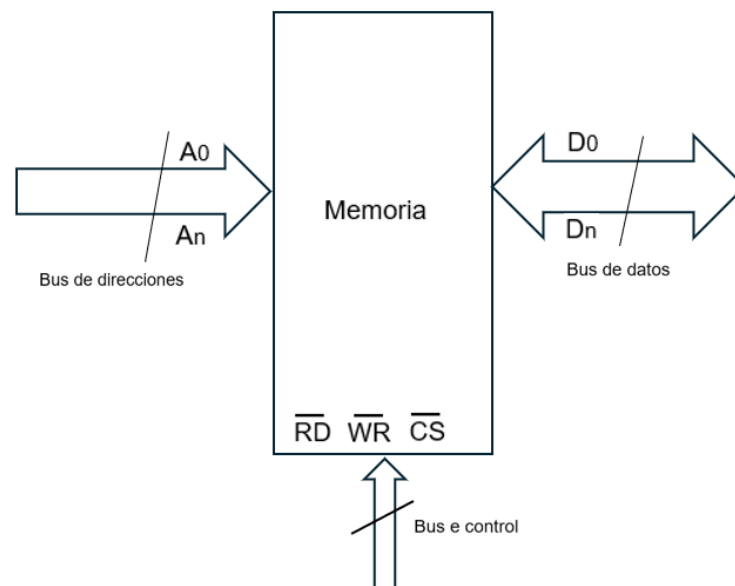


Figura 4.28 Buses de una memoria.

Capacidad de almacenamiento de una memoria.

La cantidad de información que puede almacenar una memoria se determina por el número de líneas de dirección que maneja y el ancho del bus de datos. Se determina cuantas combinaciones se logran con el bus de direcciones y se multiplica por la cantidad de bits del bus de datos, por ejemplo, si una memoria tiene las líneas de dirección de A0 a A5 y líneas de datos de D0 a D3 se dice que tiene una capacidad de 2^6 por 4 es decir, 64 por 4, se pueden almacenar 64 datos de 4 bits ya que el bus de direcciones es de 6 líneas, algunos fabricantes dan la capacidad de almacenamiento por bits sin importar el ancho del bus de direcciones, en el ejemplo anterior se puede manejar como una capacidad de 256 bits pero no se puede direccionar bit por bit.

En las memorias se le llama 1K de memoria a 1024 localidades que equivalen a tener 10 líneas de dirección(A0, A1..A9) y de allí en adelante por cada línea más de dirección se duplica la capacidad, si tiene 11 líneas de dirección(A0, A1..A10) tendrá una capacidad de 2K por el ancho del bus de datos, para una memoria que tenga un bus de direcciones de 16 líneas(A0 hasta A15) y un bus de datos de 8 bits(d0 a D7) su capacidad es de 64K por 8.

Tipo de memorias:

ROM
PROM
EPROM
EEPROM
FLASH
SRAM
DRAM

Memorias ROM (Read-Only Memory) memoria de sólo lectura, este tipo de memorias ya vienen programadas por el fabricante y es del tipo no volátil, no pierde la información cuando se le quita la energía, normalmente son fabricadas en grandes cantidades para su uso en equipos electrodomésticos, juegos etc.

Memoria PROM (Programmable Read-Only Memory) este tipo de memoria es del tipo no volátil viene lista para programarse una sola vez por lo que se debe estar seguro de los datos que se deben programar, si hay algún error la memoria que es inservible, esta memoria requiere de un voltaje de programación mayor al de operación por eso se programan fuera del sistema donde trabajara y con un programador especial.

Memoria EPROM(Erasable Programmable Read-Only Memory) memoria de sólo lectura programable y borrrable, son memorias del tipo no volátil que pueden ser borradas para volverse a programar, tienen una ventana para poder ser borradas con luz ultravioleta por lo que se requiere del borrador y el programador.

Memoria EEPROM (Electrically Erasable Programmable Read-Only Memory), o memoria de solo lectura programable y borrrable eléctricamente es un tipo de memoria no volátil que permite leer, escribir y borrar datos eléctricamente, sin necesidad de retirarla del circuito ni usar luz ultravioleta (como en la EPROM), se requiere del programador.

Memoria Flash es un tipo de memoria no volátil que permite leer, escribir y borrar bloques de datos eléctricamente, y que combina las ventajas de la EEPROM (borrado eléctrico) con una mayor densidad de almacenamiento y mayor velocidad. Se puede reprogramar miles o millones de veces.

Memoria SRAM (Static Random Access Memory), memoria estática de acceso aleatorio es una memoria volátil que almacena datos mientras tenga alimentación eléctrica, y que se caracteriza por su alta velocidad y simplicidad de acceso. Cada bit es un flip flop por lo que la hace costosa y ocupa más espacio por bit.

Memoria DRAM (Dynamic Random-Access Memory), memoria dinámica de acceso aleatorio, es una memoria volátil ampliamente utilizada como memoria principal (RAM) en computadoras, laptops, consolas de videojuegos y otros dispositivos electrónicos. Cada bit de información se almacena en una célula de memoria compuesta por un transistor y un condensador. El condensador almacena una carga eléctrica que representa un 1 o un 0. Esta carga se disipa rápidamente, por lo que la memoria DRAM debe ser refrescada miles de veces por segundo para mantener los datos. De ahí el nombre "dinámica".

Memoria SRAM 6116

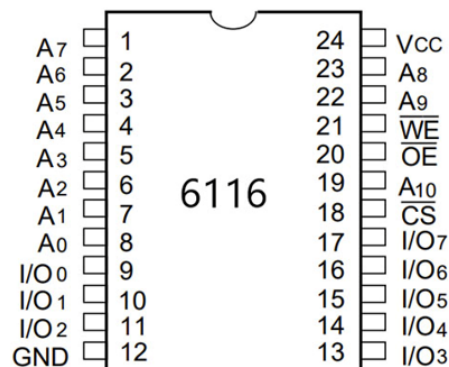


Figura 4.129 Memoria SRAM 6116.

En el circuito se puede ver que el bus de direcciones abarca las direcciones de A0 hasta A10, por lo tanto, tienen 2^{11} localidades de memoria que son 2048 y como 1024 localidades forman un kilo entonces es de 2K, el bus de datos es de 8 líneas, entonces se dice que es una memoria de 2k por 8, tiene las terminales de entrada WE que se utiliza para escribir en la memoria es activa a nivel bajo, tiene la entrada OE que se utiliza para una lectura de la memoria también es activa a nivel bajo y la terminal CS que habilita la memoria para cualquiera de sus funciones, es activa a nivel bajo.

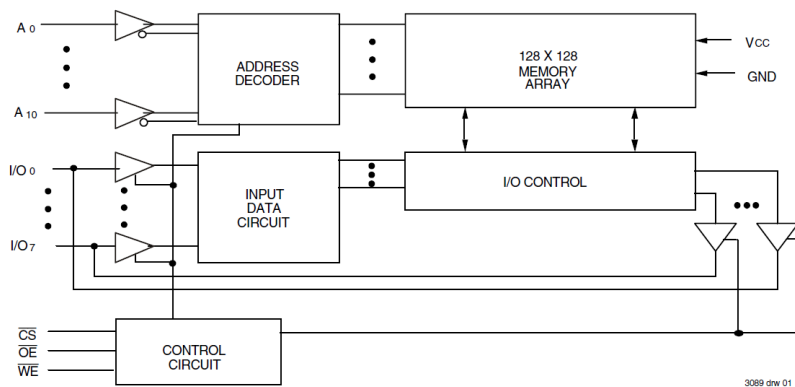


Figura 4.130 arquitectura interna de la memoria SRAM 6116.

$\overline{CS}$	$\overline{WE}$	$\overline{OE}$	Operación	Bus de datos
H	X	X	HZ	HZ
L	L	H	Escritura	Entrada
L	H	L	Lectura	Salida
L	L	L	Escritura	Entrada

Figura 4.127 Tabla de verdad de la memoria 6116.

Cuando en esta memoria se habilitan las entradas de lectura(OE) y escritura(WE) al mismo tiempo le da prioridad a la escritura para proteger al circuito ya que si se tiene algo conectado al bus de datos se podría dañar.

Memoria EEPROM 2816

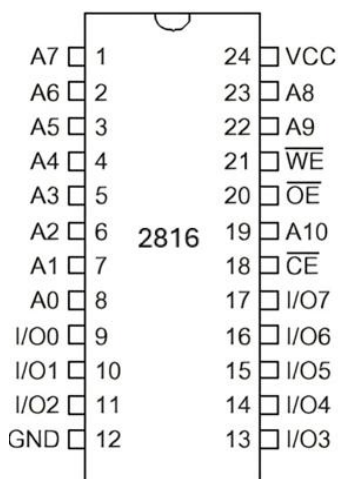


Figura 4.131 Memoria EEPROM 2816.

Esta memoria también es de 2K por 8, es una memoria no volátil también conocida como memoria de programa ya que retiene la información aun cuando se le quite la energía, tiene la misma distribución de las terminales que la memoria SRAM6116.

La memoria 2816 se puede programar de dos modos: una de 5 volts positivo y otro de alto voltaje(para el borrado total de la memoria). El primer modo se inicia con un pulso de

escritura con un flanco de bajada es decir de alto a bajo de duración de 200 ns. La memoria 2816 borra automáticamente el byte seleccionado antes de escribir el nuevo dato, el tiempo total de borrado y escritura es de 10 ms. Para leer la memoria su tiempo de acceso es de 250 ns.

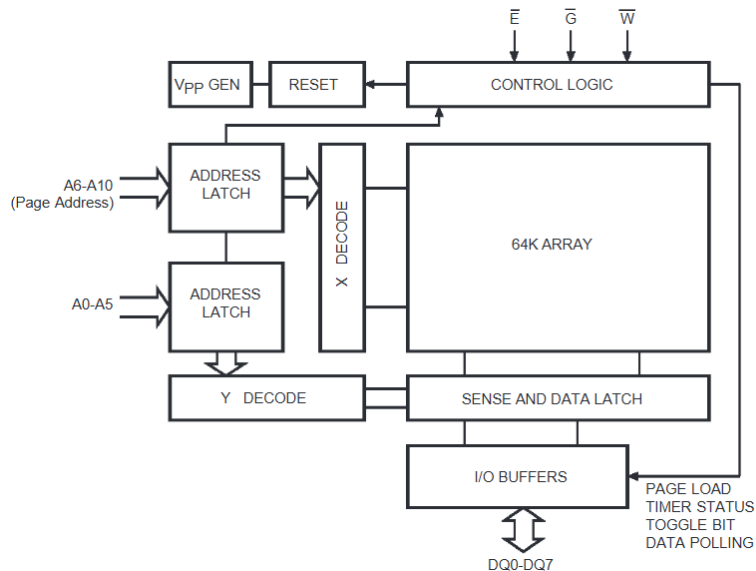


Figura 4.32 Diagrama a bloques de la memoria EEPROM 2816.

$\overline{CS}$	$\overline{WE}$	$\overline{OE}$	Operación	Bus de datos
H	X	X	HZ	HZ
L	L	H	Escritura	Entrada
L	H	L	Lectura	Salida
L	L	L	Escritura	Entrada

Figura 4.133 Tabla de verdad de la memoria EEPROM 2816.

Proceso de escritura en una memoria de programa(ROM, PROM, EEPROM)

Este tipo de memoria requiere de un voltaje mayor de 5V para poder almacenar el dato, normalmente es en la terminal Vpp(voltaje de programación) este debe ser elevado desde 5v hasta el valor que marque el fabricante que va de 12.5 hasta 21V con una duración que también indica el fabricante, algunas requieren el pulso de 40 ms, estos dos parámetros deben ser respetados para evitar dañar la memoria.

Se pone la dirección en la que se desea guardar el dato, se pone el dato a ser guardado se habilita la memoria en forma de escritura y se le da el voltaje de programación, recuerde que en forma de escritura el bus de datos está en forma de entrada. Normalmente este proceso lo hacen todos los programadores comerciales que permiten elegir el tipo de

memoria, el número y el fabricante, permite cargar el archivo en formato hexadecimal a ser guardado o introducir manualmente en una tabla los datos que se quieran grabar.

Si se requiere programar la memoria manualmente se debe dar la dirección manualmente poner el dato a grabar, habilitar la memoria y darle el pulso de grabación que marca el fabricante, recuerde que el bus de datos en escritura está de entrada.

Proceso de lectura en una memoria(ROM, EPROM y EEPROM).

En un programador, se elige la memoria, el número y el fabricante, se selecciona la opción de lectura y se mostrará en una tabla el contenido de la memoria localidad por localidad(dirección), permite descargar el programa en un formato hexadecimal.

Si se requiere leerla manualmente se debe elegir la dirección que se desea leer, poner la memoria en forma lectura y habilitarla, si se requiere monitorear el dato con leds debe poner un buffer en el bus de datos para no demandar demasiada corriente a la memoria, recuerde que el bus de datos está en forma de salida.

Proceso de escritura en una memoria SRAM.

Si se está utilizando un programador comercial, sólo se elige el número de la memoria y el fabricante, en seguida se selecciona leer y mostrará una tabla con los datos leídos, recuerde que una memoria SRAM pierde su información al quitarle la energía por lo que debemos interpretar los datos leídos como basura. Esto sólo sirve para probar que la memoria está bien.

Proceso de escritura en una memoria SRAM.

Si se utiliza un programador comercial, después de elegir el tipo, número y fabricante de la memoria, elegir la opción de escritura y mostrara la opción de cargar el archivo en formato hexadecimal o mostrará una tabla para introducir los datos que se requieran guardar. No es muy común leer o escribir memorias SRAM con un programador ya que la información se pierde al quitarle la energía. Esto sólo sirve para probar que la memoria está bien.

Para Escribir manualmente en una memoria SRAM se Hace lo siguiente:

Poner la dirección en al que se requiera guardar el dato.

Poner el dato a guardar, normalmente lo introducimos a través de un buffer bidireccional asegurando de que este en forma de entrada hacia la memoria.

Habilitar la memoria.

Habilitar la terminal de escritura WR.

Para leer manualmente la memoria SRAM se hace lo siguiente:

Poner la dirección que se desea leer.

Poner el buffer al bus de datos en forma de salida para poder poner leds y leer el dato.

Habilitar la memoria asegurándose que está en forma de lectura para evitar dañar la memoria ya que estamos sacando datos almacenados.

Práctica Memoria SRAM

Objetivo

Comprender el funcionamiento de la memoria SRAM 6116 y los circuitos que se deben conectar para trabajar adecuadamente en la memoria.

Se diseñará un circuito en Proteus que permita leer y escribir en la memoria RAM 6116 y ver la información en display conectados en paralelo (paquete de cuatro displays).

Desarrollo.

Memoria RAM 6116

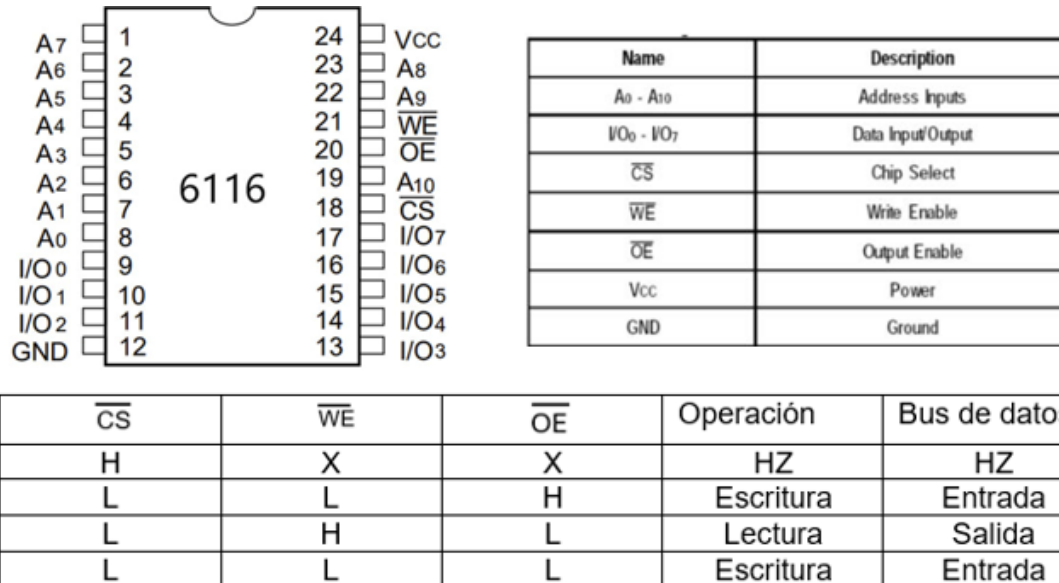


Figura 4.134 Memoria SRAM 6116 y su tabla de verdad.

De acuerdo con lo visto anteriormente se sabe que la capacidad de la memoria es de 2K x 8, es decir se puede guardar 2048 datos de 8 bits y permanecerán en la memoria mientras este energizada. Según la tabla de verdad se entiende que el bus de datos es bidireccional y, por lo tanto, si se quiere manejar la memoria sin un microprocesador se tiene que manejar un circuito bidireccional de 8 bits para poder introducir los datos a la memoria (escribir) o poderlos sacar (leer). El circuito que se usara es el 74245 buffer octal bidireccional el cual se muestra en la figura.

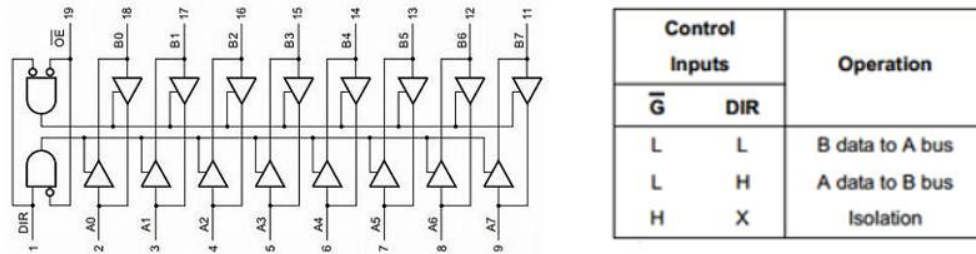


Figura 4.135 Circuito 74245 Buffer bidireccional y su tabla de verdad.

Para observar lo que se escribe o lee en la memoria, se usarán los displays conectados en paralelo. Estos sólo tienen los 7 segmentos para todos los displays del paquete, pero manejan aparte el ánodo o cátodo de cada display, de tal manera que, si hay información en los segmentos, sólo se verán en los displays que estén activados a través de su cátodo o ánodo, normalmente se usa un transistor para controlar los comunes del display como se muestra en la figura.

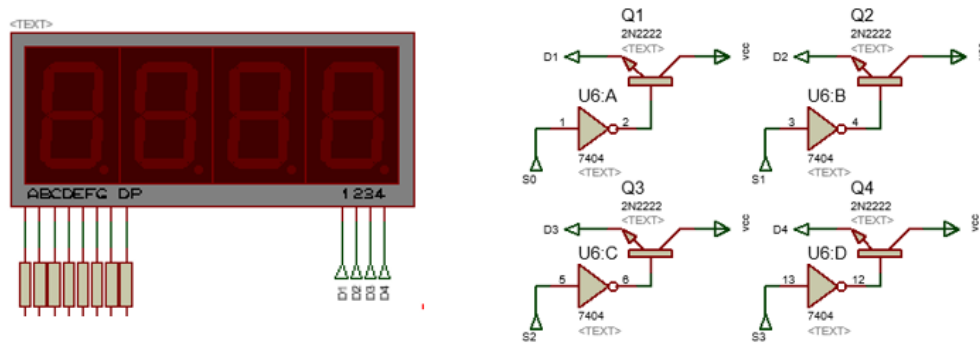


Figura 4.136 Paquete de Display's y transistores para controlar los ánodos.

Se necesita un nivel bajo en los inversores para que dé un nivel alto en la base de los transistores y prendan los displays, si hay información en los segmentos se verán en los displays cuyos transistores estén en on (interruptores).

En la figura siguiente se pusieron en on todos los transistores:

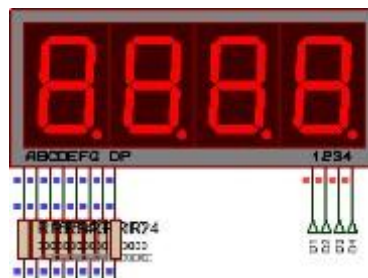
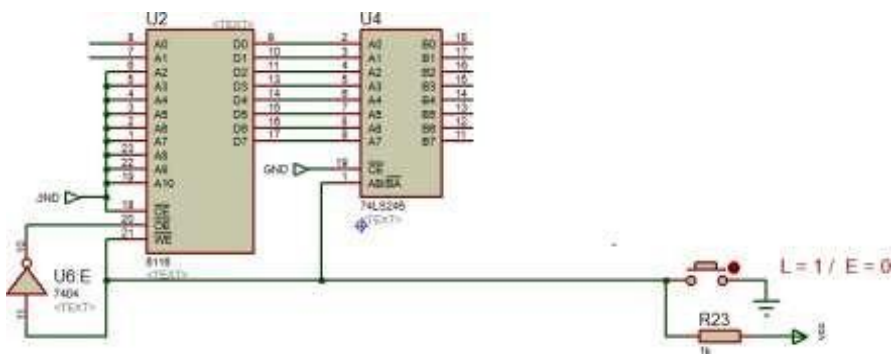


Figura 4.137 Paquete de display's encendidos mostrando la misma información.

Conexión del buffer 74245 a la memoria para controlar la dirección del dato, es decir para poder escribir o leer en la memoria. El circuito debe quedar como en la siguiente figura:



De acuerdo con la tabla de verdad de la memoria 6116 y la del buffer 74245, cuando el botón está abierto el 74245 recibe un uno en la terminal AB/BA y como OE está permanentemente en cero el dato va de A a B, por otro lado, a la memoria le llega un uno a la terminal WE y un cero en la terminal OE, lo que pone la memoria en forma de lectura es decir el dato sale de la memoria y pasa por el 74245. El dato de la memoria será el de la dirección que se elija con las terminales A0 hasta A10.

292

Con lo anterior se cumple el requisito para poder leer o escribir en la memoria y, además, se amplifica la corriente que da la memoria para poder conectar un led.

¿Cómo se puede escribir en la memoria o leerla y poder ver lo que se escribió o leerla? Para ello se usa otro 74245 que permita conectar interruptores al bus de datos para darle los unos o ceros que se quiere escribir en la memoria(guardar) y que se desconecten del bus cuando se van a leer, ya que si se quedan conectados los interruptores físicamente podemos dañar la memoria, imagine que el interruptor se quedó conectado a tierra y la memoria está sacando un uno en esa línea, dañaría la memoria. El circuito para esto quedaría como en la siguiente figura:

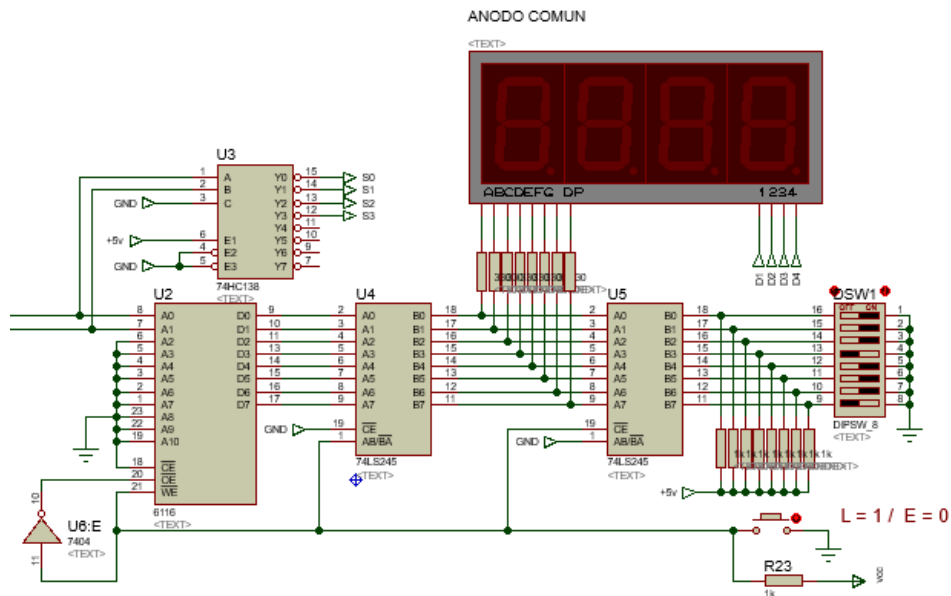


Figura 4.140 Circuito para mostrar en el display el dato leído o a escribir.

Ya conectado el segundo 74245 con los interruptores ya casi está listo para poder leer y escribir en la memoria. El circuito funciona de la siguiente manera: si el botón está sin presionar, al segundo 74245 U5 le llega un nivel alto a la terminal OE por lo tanto está en tercer estado es decir alta impedancia y es como si no estuviera eléctricamente, note que la terminal AB/BA está conectada a tierra por lo tanto, este circuito sólo deja pasar datos de B a A, para poder escribir en la memoria, en este estado del botón, a el otro 74245 le llega a la terminal AB/BA un uno, lo que hace que lo que tenga en las entradas A pasan a B, la memoria recibe un uno en la terminal WE y un cero en la terminal OE debido al inversor, lo que pone a la memoria en forma de lectura, el dato que ponga en sus líneas D0 a D7 pasan por 74245 U4 y llegan al display, el transistor que este en on permitirá que se vea la información de la memoria. Si presionamos el botón el 74245 U5 se activará en dirección de B a A y lo que tengan los interruptores se verán en el display cuyo transistor este en on, al mismo tiempo el otro 74245 recibe un cero en la terminal AB/BA dejando pasar el dato

de los interruptores de B a A y llegan a la memoria que está en forma de escritura ya que la terminal WE recibe un nivel bajo y la terminal OE un nivel alto.

Circuito para ir seleccionando las localidades de memoria y que al mismo tiempo vaya habilitando los transistores uno a la vez, haciendo que coincida que cuando este seleccionada la localidad de memoria cero, este habilitado sólo el primer transistor y los demás apagados y cuando este seleccionada la segunda localidad de memoria se habilita el segundo transistor y los demás apagados y así sucesivamente, se usa un contador 74193 y un decodificador 74138 para la tarea mencionada. El circuito queda de la siguiente manera:

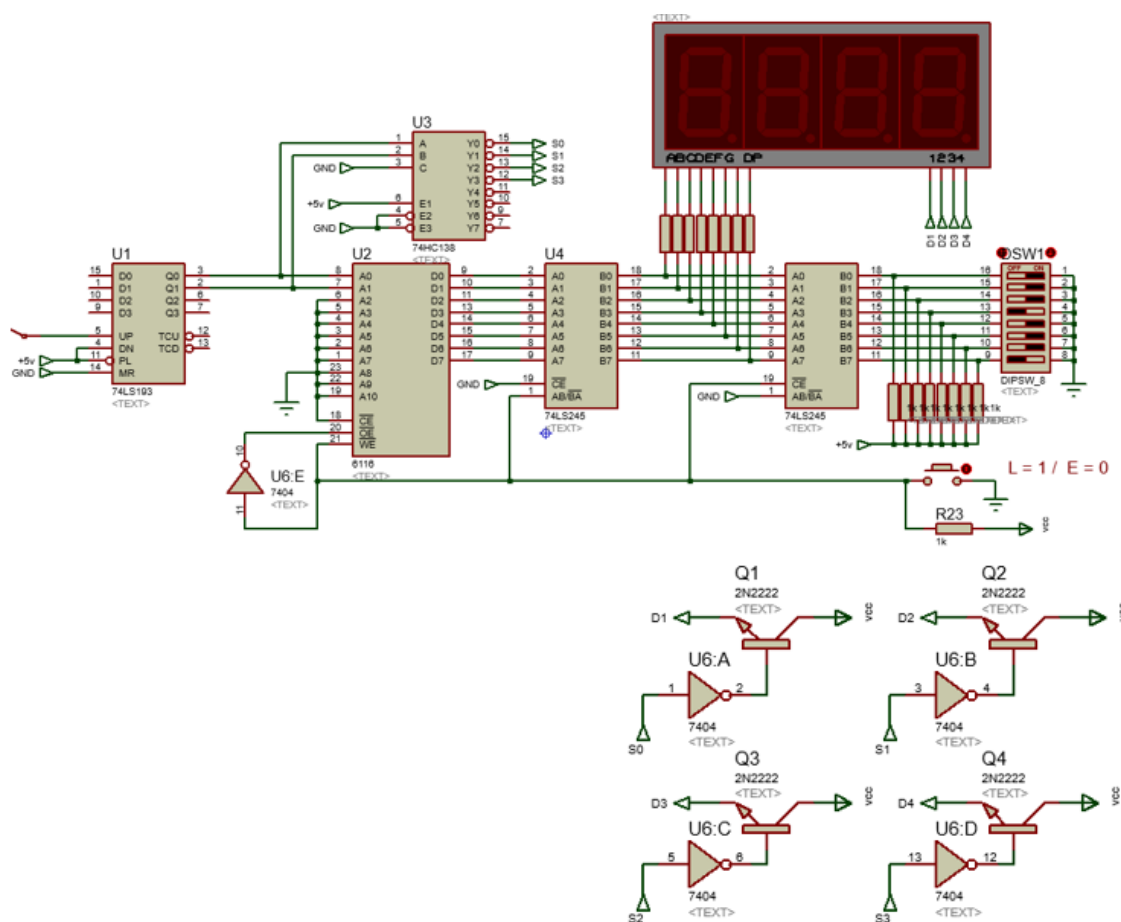
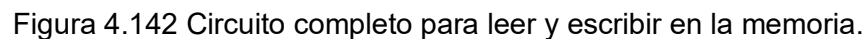


Figura 4.141 Circuito con el control de dirección y selección del display.

Por último, solo falta la señal de reloj para ir seleccionando las localidades de memoria, primero manualmente con un multivibrador monoestable a un segundo, que permita ir seleccionando la localidad a grabar, una por una y que se vaya prendiendo el transistor que le corresponda a esa localidad, también un multivibrador astable de una frecuencia de no menos de 60Hz por display del circuito es decir 240 Hz (60 x 4) para poder engañar al ojo y no ver parpadear los displays. El circuito final queda de la siguiente forma:



- a) Poner el selector del reloj del 74193 en posición de 1 seg.
- b) Poner el botón L=1/E=0 en cero asegúrese que se quede en esa posición. (Presionando en el punto rojo)
- c) Presionar el botón de pulso (multivibrador monoestable) hasta seleccionar el display que se quiera.
- d) Poner los interruptores en la posición adecuada para que se vea en este caso la "H"
- e) Poner el botón L=1/E=0 en uno y regresarlo a cero y dejarlo así. (Presionando en el punto rojo)
- f) Seleccionar otro display con el botón de pulso y poner los interruptores para que se vea "O"
- g) Poner el botón L=1/E=0 en uno y regresarlo a cero y dejarlo así. (Presionando en el punto rojo)
- h) Seleccionar otro display con el botón de pulso y ponga los interruptores para que se vea "L"
- i) Poner el botón L=1/E=0 en uno y regresarlo a cero y dejarlo así. (Presionando en el punto rojo)

j) Seleccionar el ultimo display con el botón de pulso y poner los interruptores para que se vea "A"

k) poner el botón L=1/E=0 en uno y regresarlo a cero y dejarlo así. (Presionando en el punto rojo)

Al llegar aquí ya no cambiar de display ya que si se regresa a cualquier otro display que ya se grabó se borra la información anterior. Dejar el botón L=1/E=0 en uno (abierto) y si se presiona el botón pulso ira mostrando lo que quedó grabado en la memoria.

l) Cambiar el interruptor del reloj a (240Hz) y se verá la palabra completa si todo está bien.

Hay que recordar que es una memoria RAM y, por lo tanto, si apaga el Proteus se borra (es equivalente a quitarle la energía a la RAM) lo que se grabó y tiene que repetir el procedimiento para grabarla nuevamente.

Grabe cualquier mensaje.

4.7 Diseño de circuitos mediante lenguajes de descripción de hardware.

En el apartado 3.5 se vio el lenguaje de descripción de hardware y el manejo del software WinCupl con algunos ejercicios en los dispositivos GAL's , se puede bajar el programa ISE de Xilinx para programar dispositivos más poderosos como lo son los FPGA's.

El software se descarga de la página Xilinx.com donde se elige crear una cuenta en la que pedirá los siguientes datos:

AMD Productos Soluciones Descargas y Asistencia Técnica Tienda

Creación de cuenta AMD

Para crear una cuenta, completa el siguiente formulario.

Se enviará por correo electrónico un mensaje de activación de cuenta con un **Token de acceso** a la dirección que especifique a continuación.

Nombre *

Apellido *

Correo electrónico *

Pedimos aos usuários empresariais que forneçam o endereço de e-mail de suas empresas para que obtenham acesso completo a licenciamento, suporte e serviços. A todos os demais usuários, pedimos que usem seu endereço de e-mail pessoal.

Preferred Language *

Inglés

Ubicación *

Elige una opción

Si creas una cuenta, aceptas los [Términos de uso](#) y la [Política de privacidad](#) de AMD.

Figura 4.143 Ventana para crear la cuenta.

El correo debe ser institucional, una vez llenados los datos mandará un correo con el token para permitir generar el User ID y la contraseña. Ya creada la cuenta ya se puede entrar y mostrará la siguiente ventana.

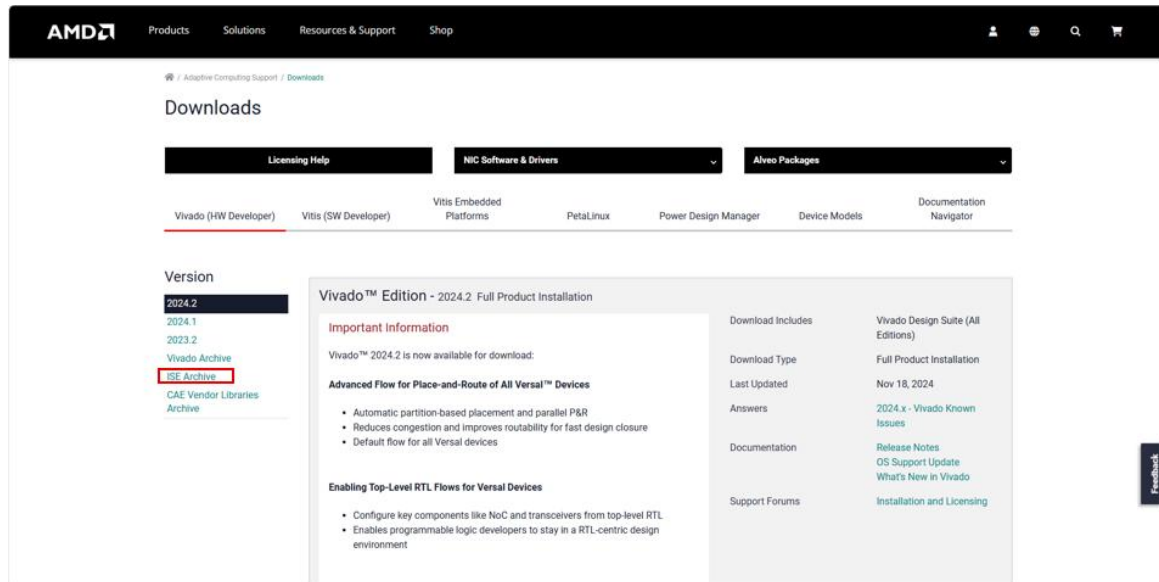


Figura 4.144 ventana inicial de Xilinx ahora AMD con la opción ISE Archive seleccionada.

Al darle clic mostrará la ventana donde se elige la versión de ISE.

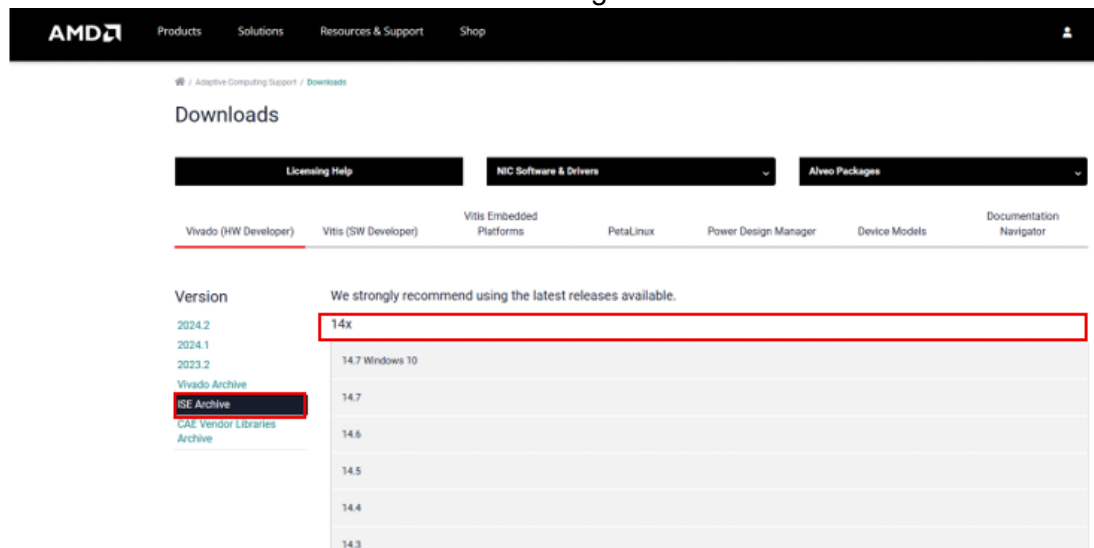


Figura 4.145 Ventana para seleccionar la versión de ISE.

En esta ventana se selecciona ISE Archive y la versión 14.7 Windows 10, al darle clic muestra la siguiente ventana:

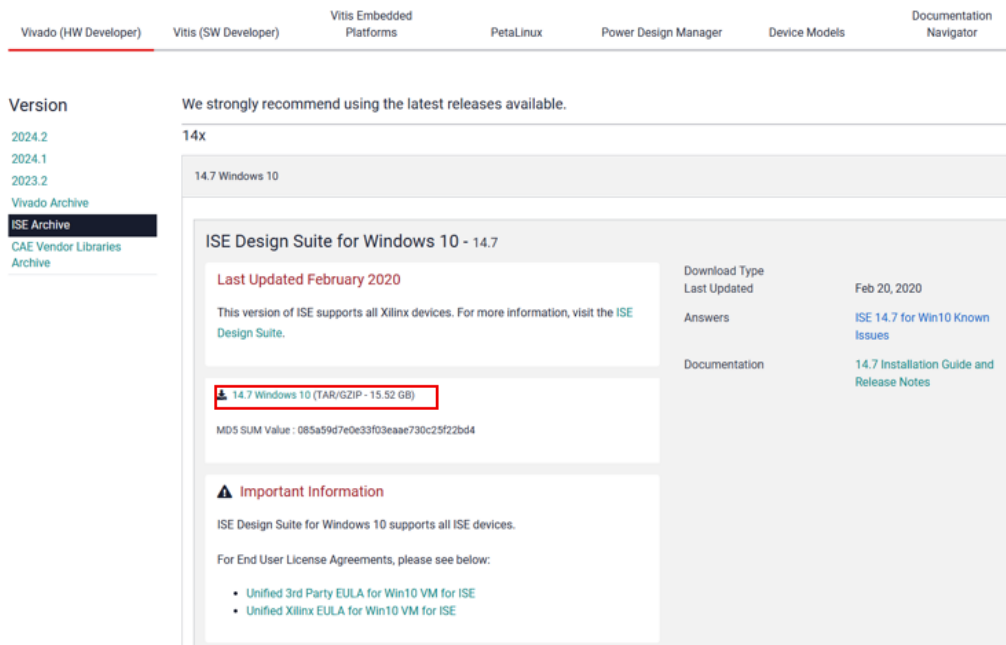


Figura 4.146 Ventana para seleccionar la descarga del archivo de ISE 14.7.

Al darle clic en la ventana de descarga de ISE 14.7 iniciará la descarga del archivo de 15.52 GB.

Ya descargado el archivo se descomprime y se mostrarán lo siguiente:

bin	11/05/2025 06:41 p. m.	Carpeta de archivos	
compartir	11/05/2025 06:54 p. m.	Carpeta de archivos	
data	11/05/2025 06:41 p. m.	Carpeta de archivos	
lib	11/05/2025 06:37 p. m.	Carpeta de archivos	
ova	11/05/2025 06:37 p. m.	Carpeta de archivos	
payload	11/05/2025 06:37 p. m.	Carpeta de archivos	
tps	11/05/2025 06:41 p. m.	Carpeta de archivos	
msvcp110.dll	11/05/2025 06:41 p. m.	Extensión de la ap...	523 KB
msvc110.dll	11/05/2025 06:41 p. m.	Extensión de la ap...	855 KB
vccorlib110.dll	11/05/2025 06:41 p. m.	Extensión de la ap...	247 KB
xsetup	11/05/2025 06:41 p. m.	Aplicación	435 KB

Figura 4.147 Archivos generados al descomprimir ISE 14.7.

Se ejecuta el archivo xsetup y dará la bienvenida indicando que se instalará en un entorno virtual para Windows 10. Si no se tiene Windows 10 se deberá instalar una máquina virtual para que funcione ISE 14.7

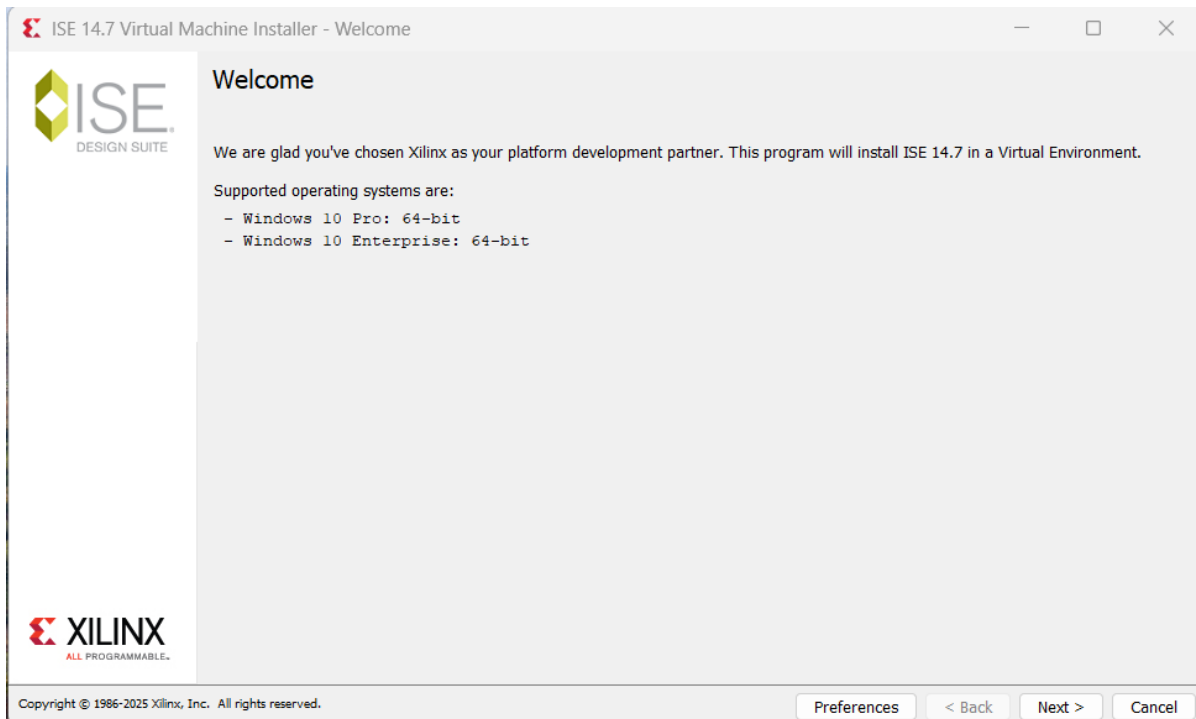


Figura 4.148 Ventana inicial de instalación de ISE 14.7.

Al darle clic en la opción Next mostrará la siguiente ventana:

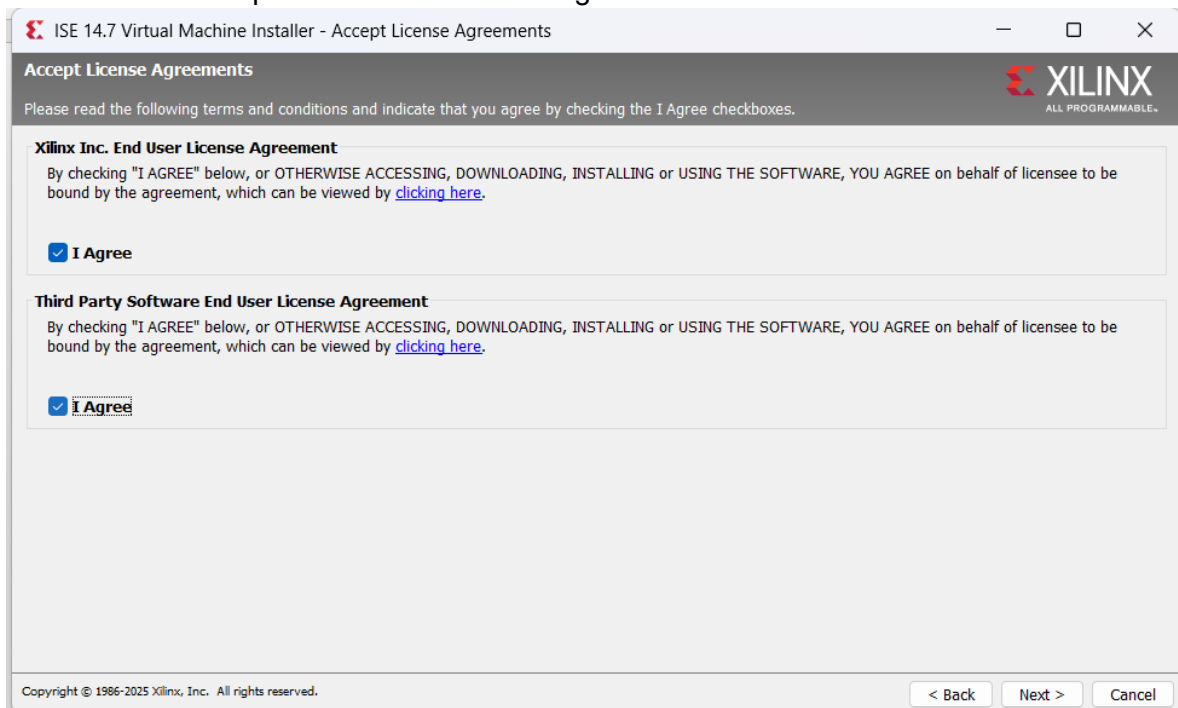


Figura 4.149 Ventana de aceptación de condiciones.

Dar clic Next una vez seleccionado I Agree en los dos casos y mostrará la siguiente ventana:



Figura 4.150 Ventana que indica lo que se instalará.

Dar clic en Next y se mostrará la siguiente ventana:

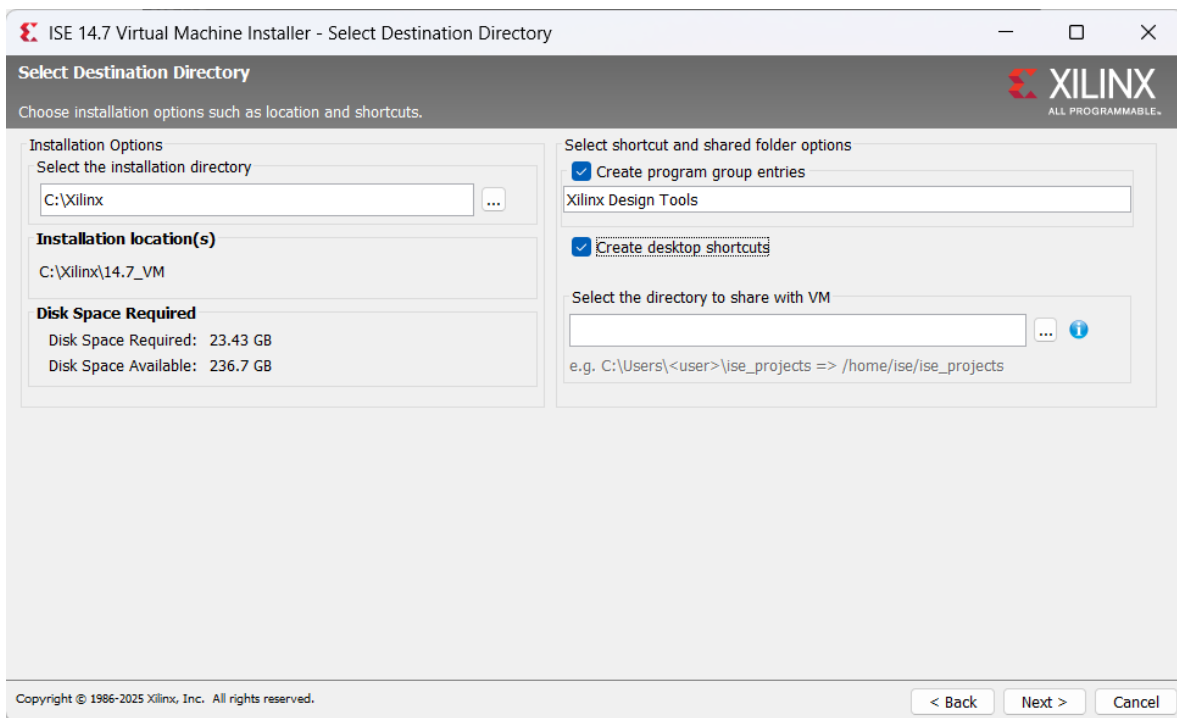


Figura 4.151 Ventana para la ruta donde estará la carpeta para compartir los archivos de Windows 10 y la máquina virtual.

Al darle la ruta de la carpeta donde compartirán los archivos y darle Next mostrará la siguiente ventana:

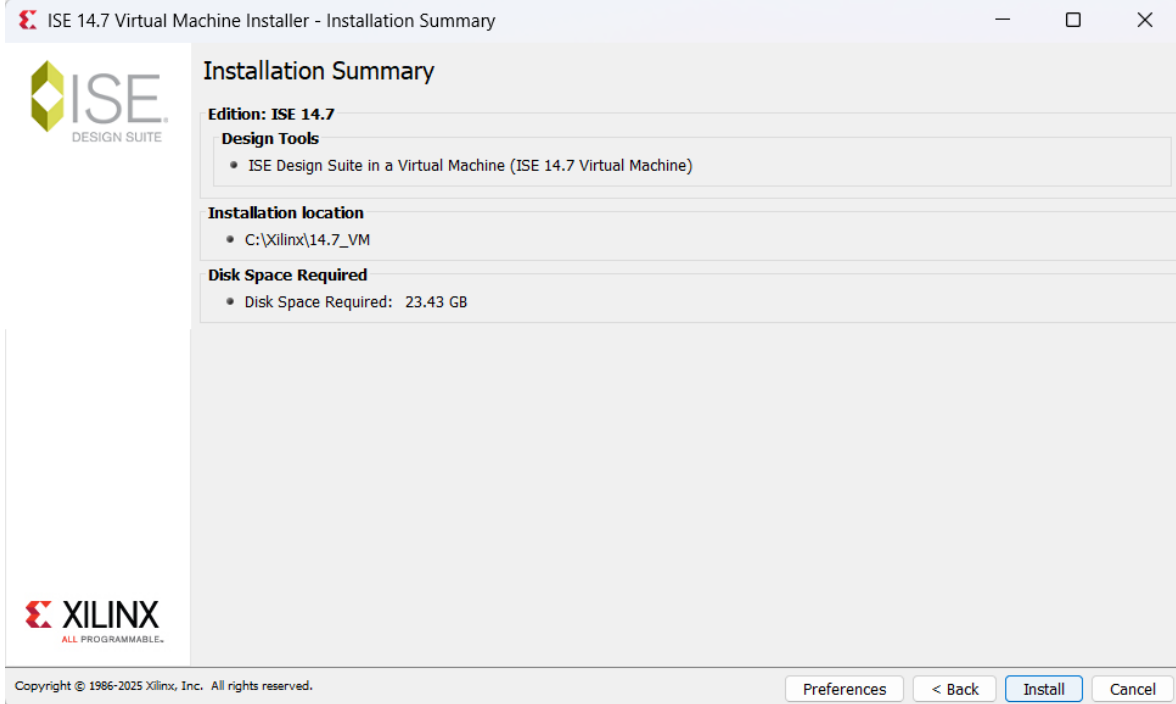


Figura 4.152 Ventana para iniciar la instalación de ISE.
Cuando se le da Next inicia la instalación y se mostrara el avance de esta.

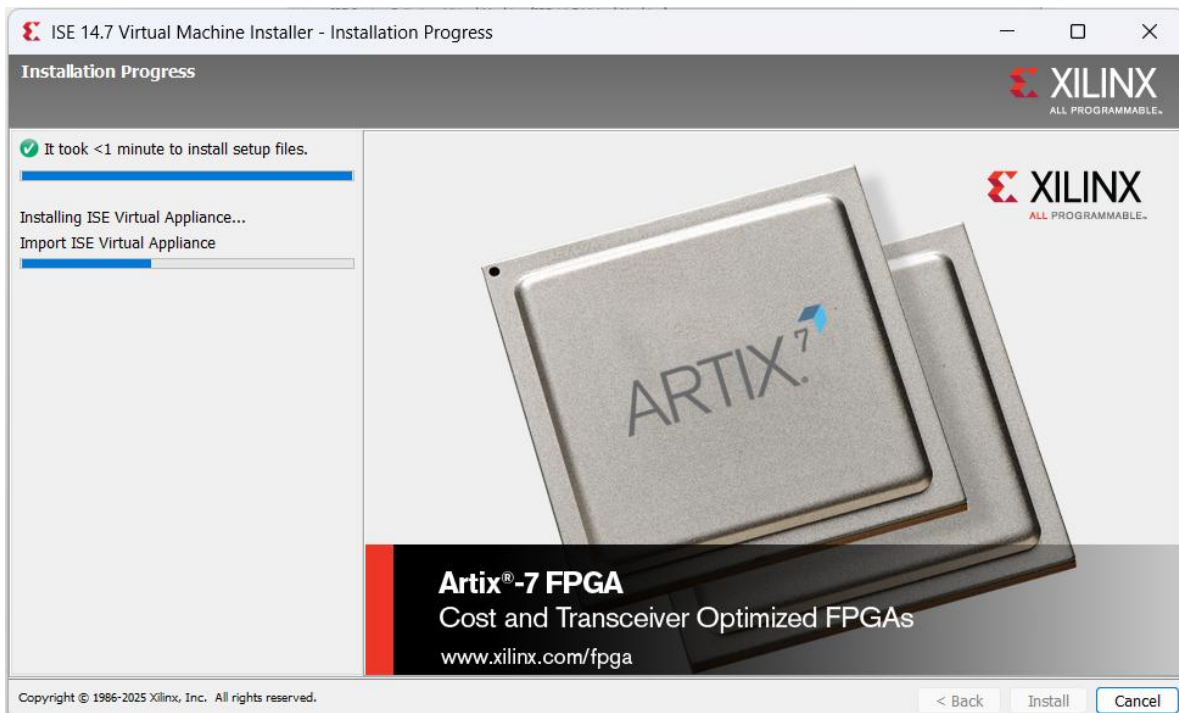


Figura 4.153 Ventana de instalación en proceso.

Una vez terminada la instalación mostrara la siguiente ventana:

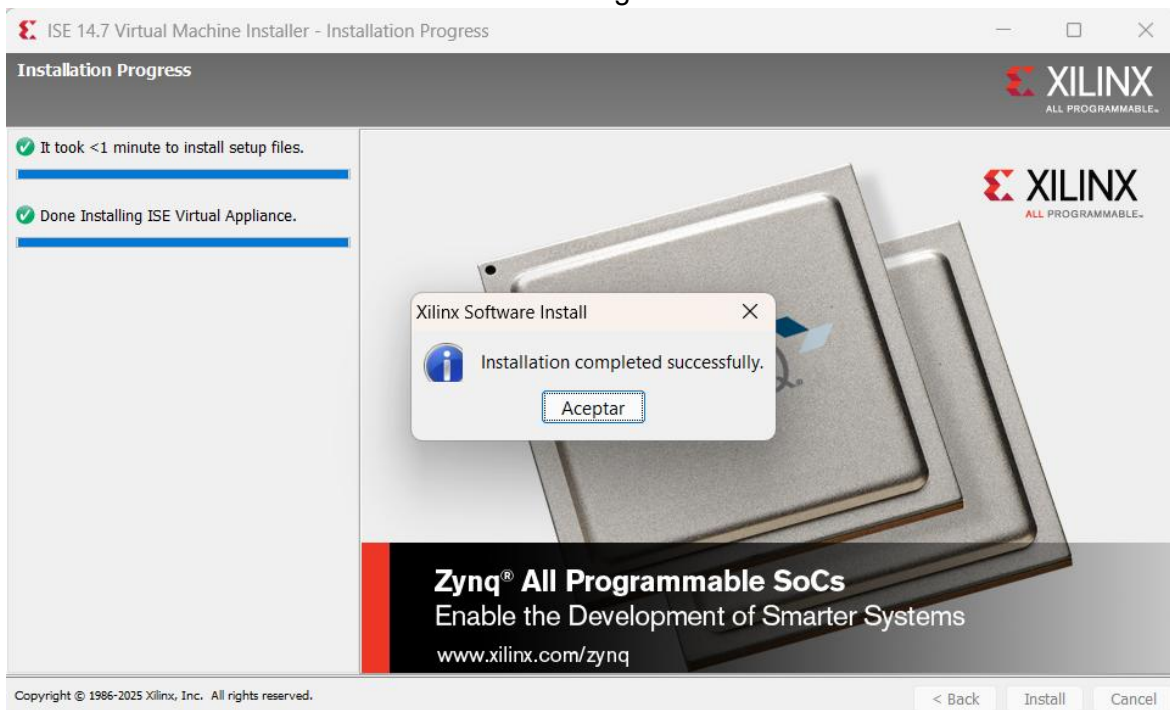


Figura 4.154 Ventana de instalación completa.

Recuerde que ISE corre sobre la su máquina virtual instalada en este caso es la Oracle VM VirtualBox, el ejecutarlo se mostrará los programas que pueden correr sobre ella, debe aparecer ISE_14.7 como se muestra en la siguiente figura:

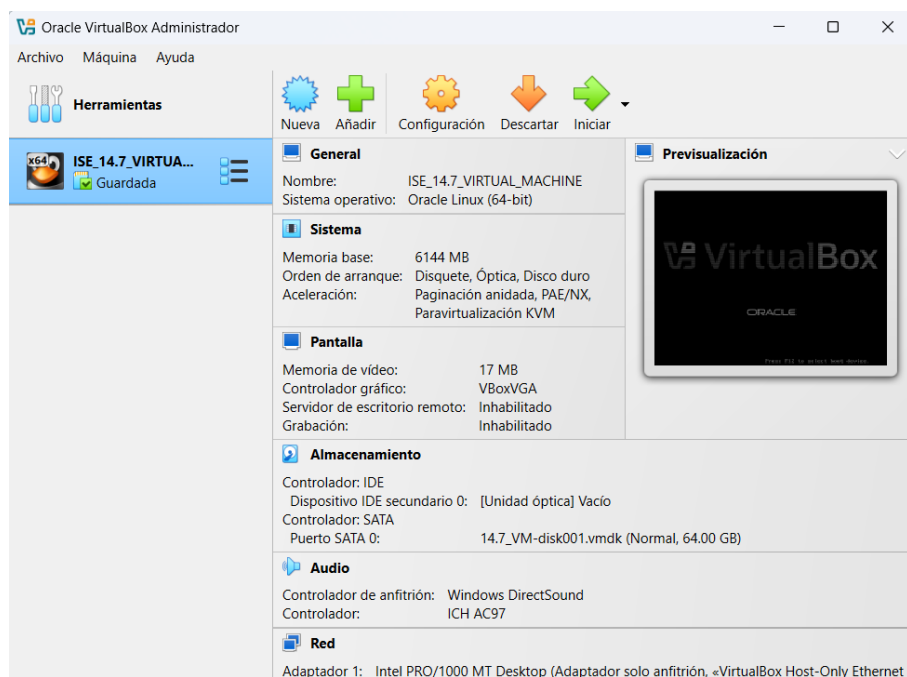


Figura 4.155 Ventana de Oracle donde se ejecutará ISE.

Al darle clic al icono ISE_14.7_VIETUAL.. se mostrará la siguiente ventana:

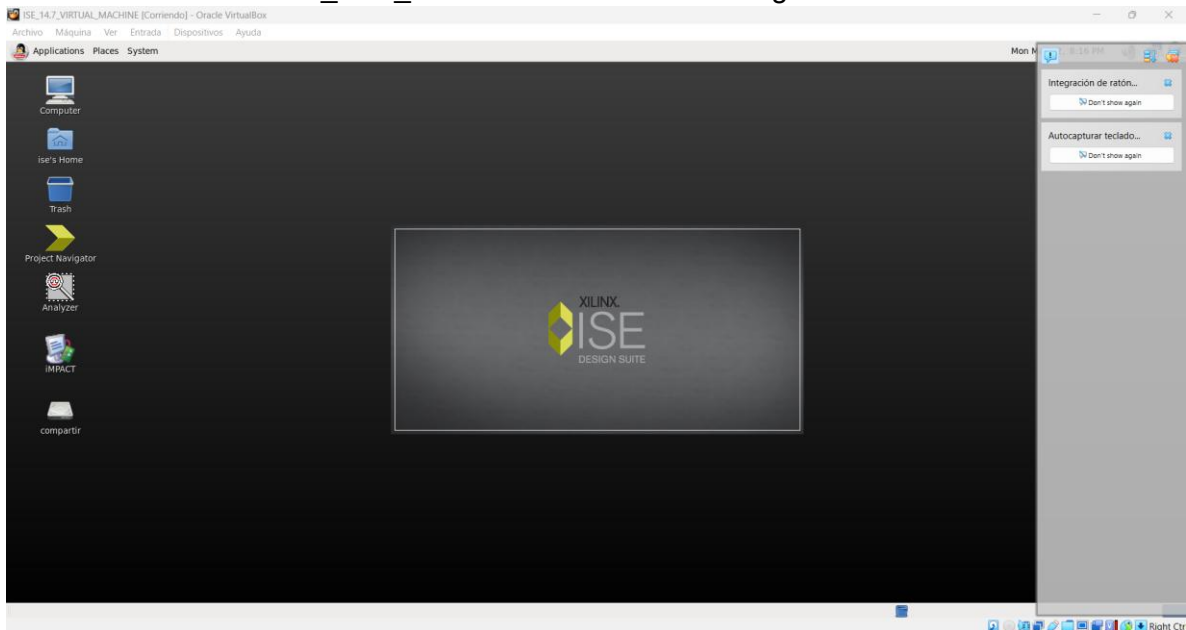


Figura 4.156 Ventana de la máquina virtual ya trabajando.

En esta ventana se le da doble clic al icono de la carpeta Project Navigator para arrancar ISE, mostrará la ventana principal de ISE.

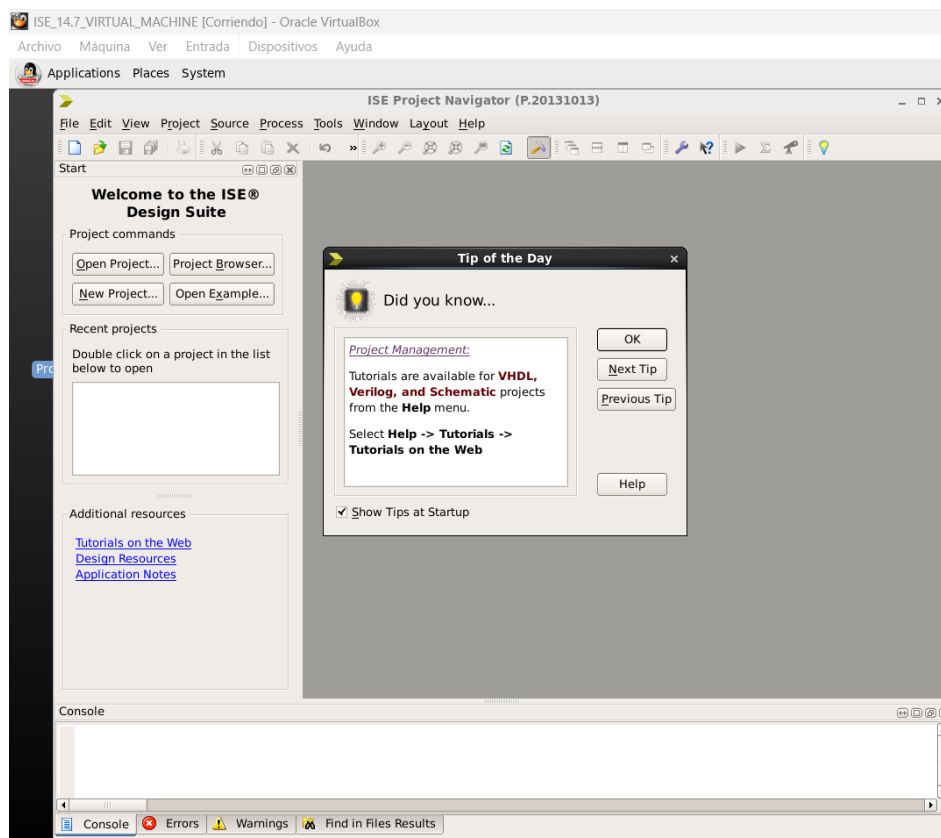


Figura 4.157 Ventana de inicio de ISE.

En la ventana de Tip of Day leemos las novedades de ISE y le damos OK.
Para probar que todo quedó bien instalado se hará un pequeño programa a manera de “Hola Mundo”, que será una compuerta NAND como se muestra en la siguiente figura:



Figura 4.158 Compuerta NAD para implementarla en ISE.

Para iniciar damos clic en la ceja di File y en New Proyect.

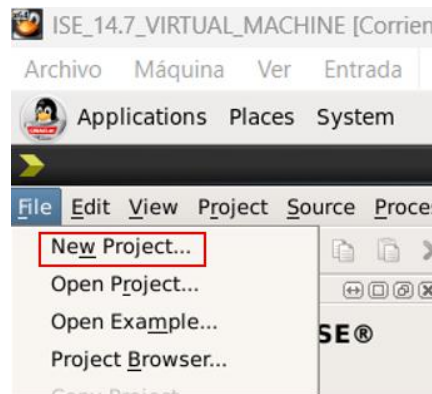


Figura 4.159 Ventana para iniciar un nuevo proyecto.

Mostrará la ventana pada darle el nombre al proyecto y la carpeta donde se guardarán.

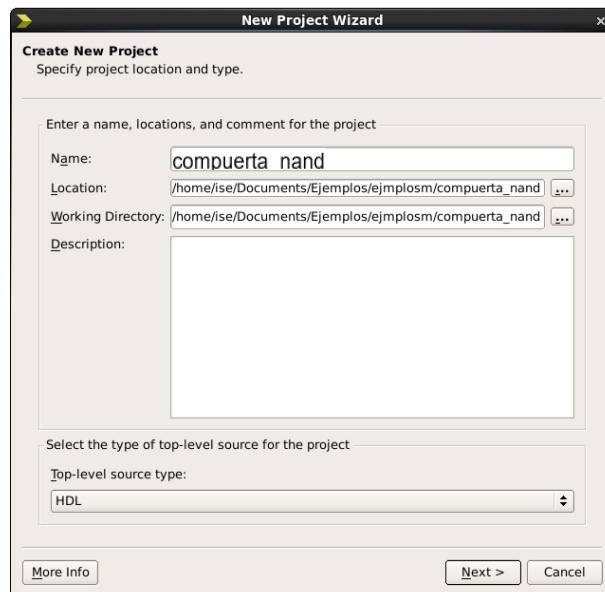


Figura 4.160 Ventana para dar el nombre al proyecto y la carpeta donde se guardará.

En el área de descripción podemos documentar el proyecto, al darle Next mostrará la ventana donde se puede escoger la FPGA, se puede seleccionar con la que se cuente en

la ceja de Family, se escogerá la Spartan3E, en la ceja del simulador escogemos ISim (VHDL/Verilog), en la ceja Preferred Language debe ser VHDL lo mismo que en la ceja VHDL Source Analysis Estándar, lo demás lo demos dejar como está, le damos Next y mostrará la siguiente ventana:

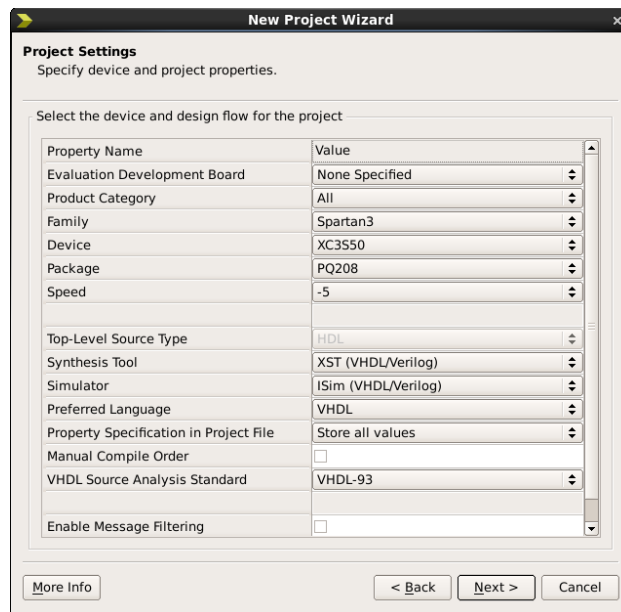


Figura 4.161 Ventana de selección del dispositivo, lenguaje y simulador.

Al darle Next mostrará la siguiente ventana:

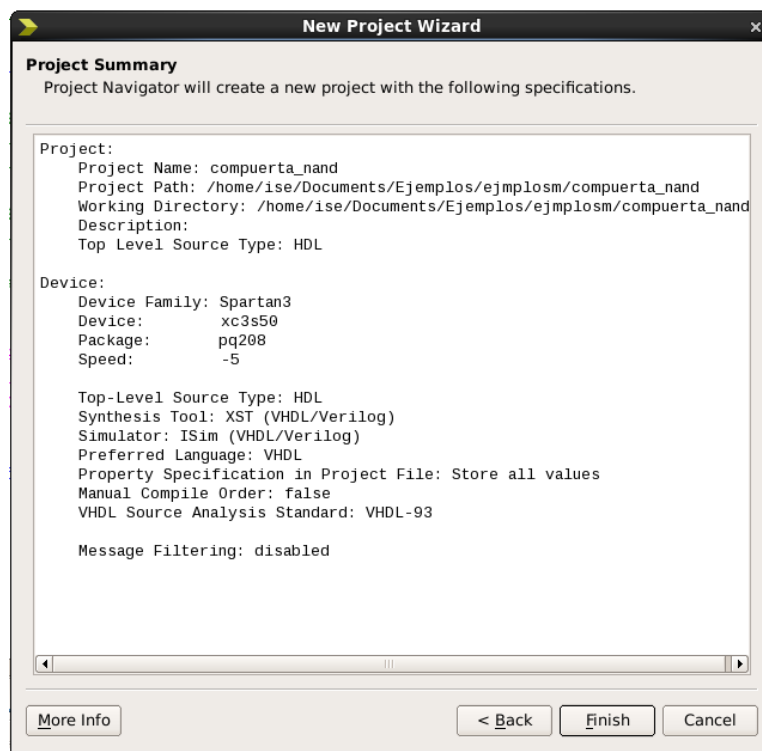


Figura 4.162 Ventana de la descripción del nuevo proyecto.

Le damos clic en Finish y ya se ha creado un nuevo proyecto.

Nos mostrará en la parte izquierda de la ventana el nombre del proyecto y el dispositivo como se ve en la siguiente figura:

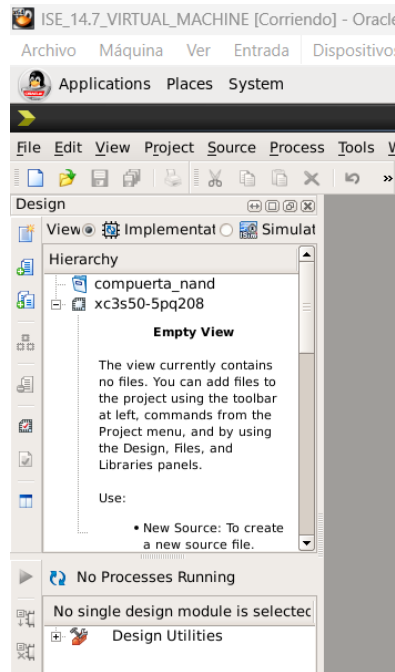


Figura 4.163 Ventana del nuevo proyecto.

Le damos clic con el botón derecho del ratón en el número del dispositivo y nos mostrará una ventana de nuevas fuentes.

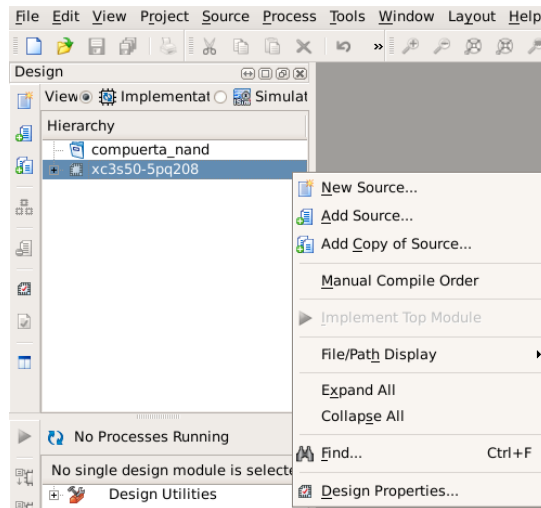


Figura 4.164 Ventana de nuevas fuentes.

Se le da clic en New Source y se selecciona VHDL Module, le ponemos el nombre, puede ser el mismo del proyecto.

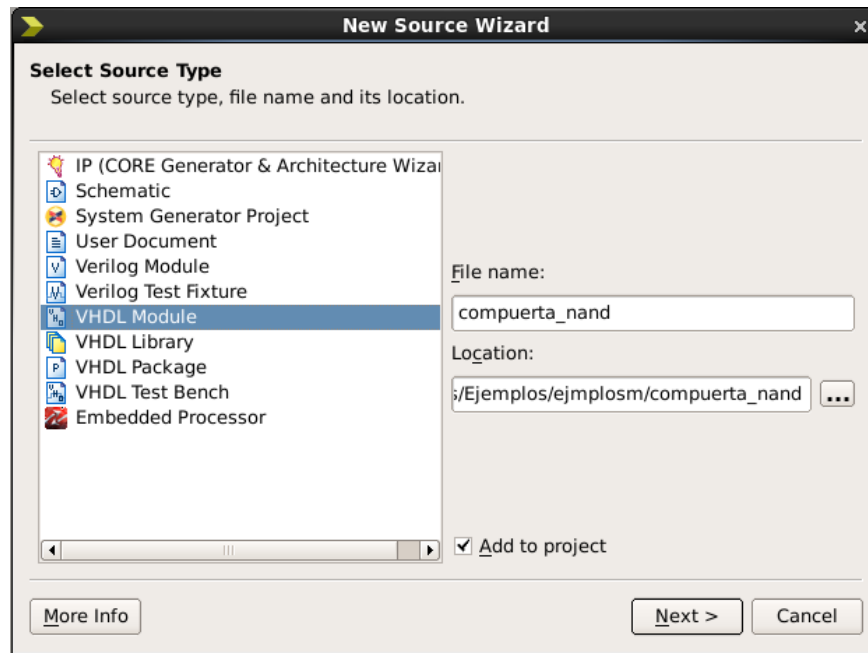


Figura 4.165 Ventana para seleccionar el módulo VHDL y el nombre.

Al darle Next, abre una nueva ventana donde se asigna las entradas y salidas del proyecto, allí ponemos el nombre que se le dio a las entradas y salida de la NAND.

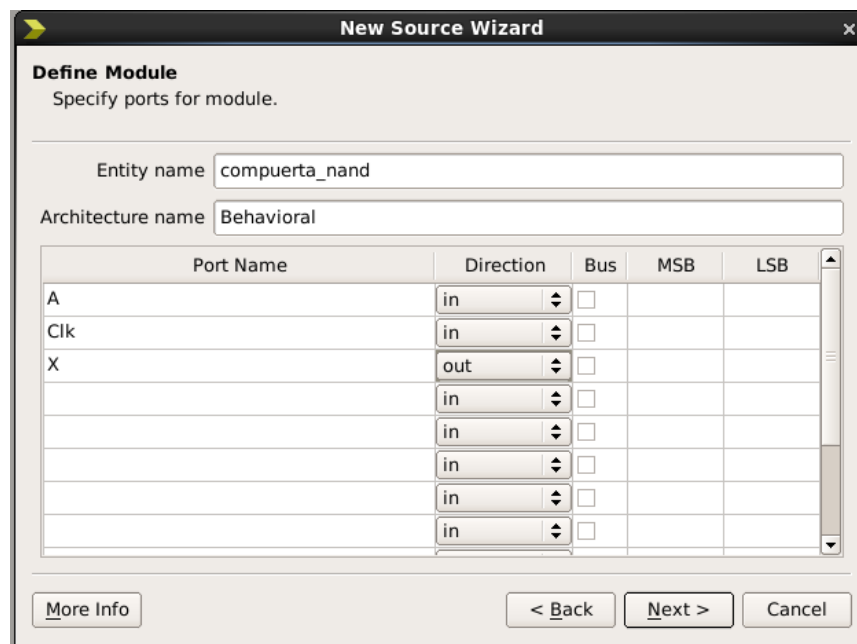


Figura 4.166 asignación del nombre y dirección de las entradas y salidas del proyecto.

Al darle Next mostrara una nueva ventana con la entidad(Entity) y su nombre, la arquitectura (Architecture) y los puertos(Port).

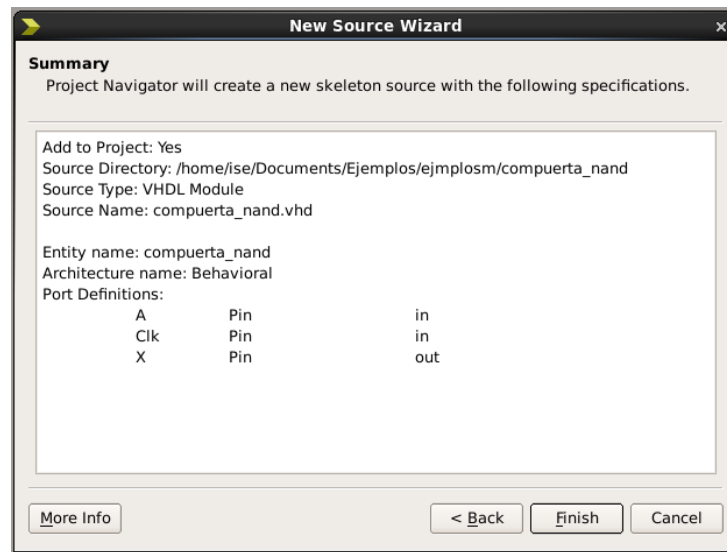


Figura 4.167 Ventana con los puertos definidos.

Se le da Finish y mostrara en la ventana los comentarios del proyecto, las librerías que se usarán, la entidad y la arquitectura.

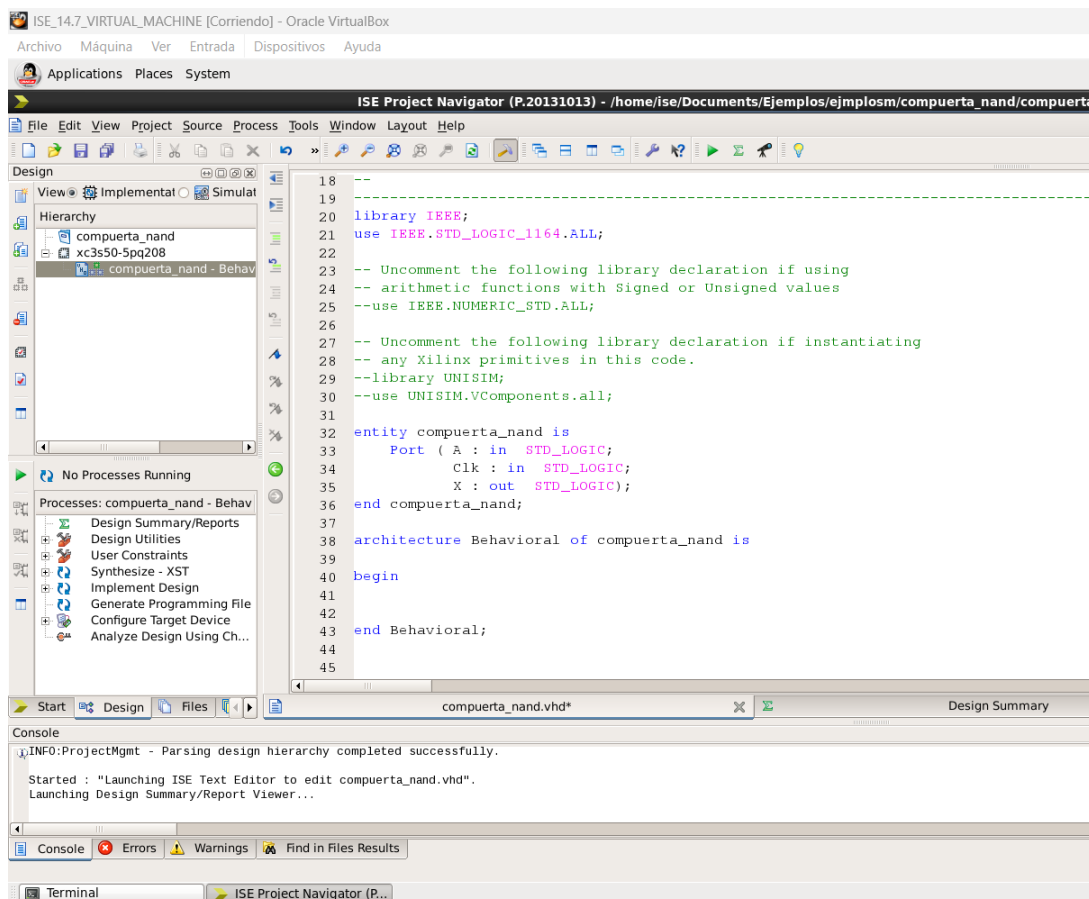


Figura 4.168 Ventana para describir la arquitectura del proyecto.

Se posiciona después del begin y se describe el comportamiento del proyecto como se vio en la sección 3.5, $X \leq \text{Clk nand } A$;.

```
32 entity compuerta_nand is
33     Port ( A : in  STD_LOGIC;
34           Clk : in  STD_LOGIC;
35           X : out  STD_LOGIC);
36 end compuerta_nand;
37
38 architecture Behavioral of compuerta_nand is
39
40 begin
41     X <= A nand Clk ;
42
43 end Behavioral;
```

Figura 4.169 descripción de compuerta_nand en la arquitectura.

Se le da guardar, si no hay errores mostrará la siguiente ventana:

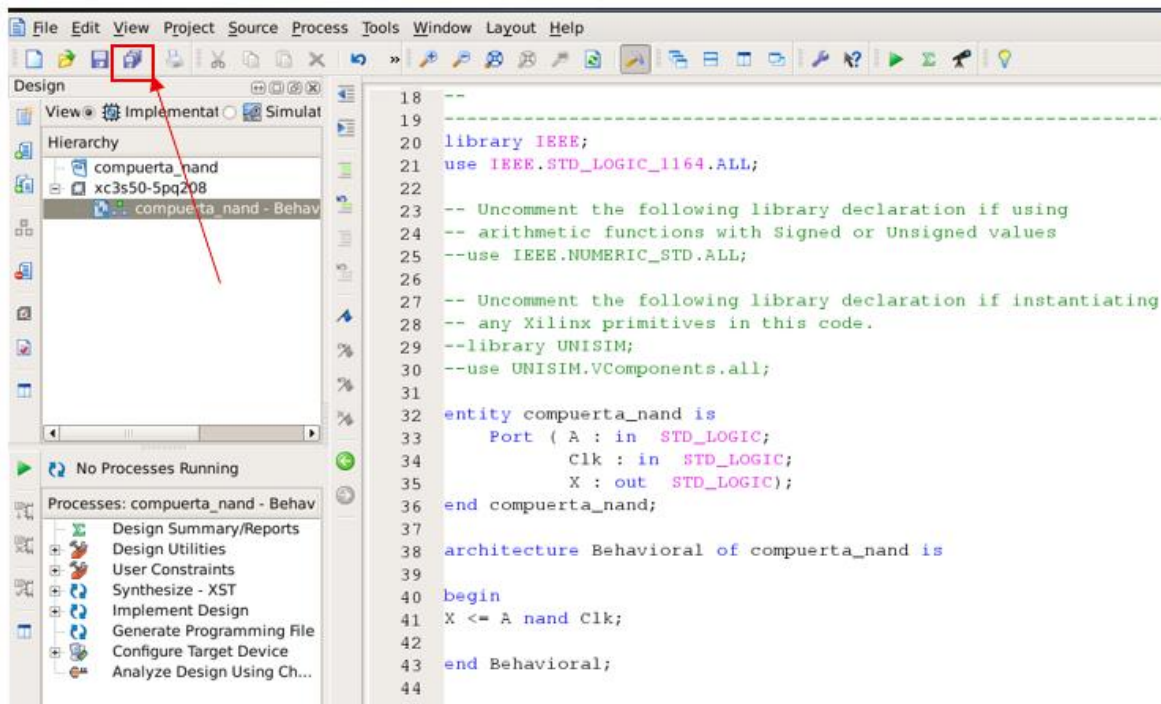


Figura 4.170 Proyecto ya guardado sin errores.

Ahora se selecciona la opción Synthesize – XST y mostrará la siguiente ventana:

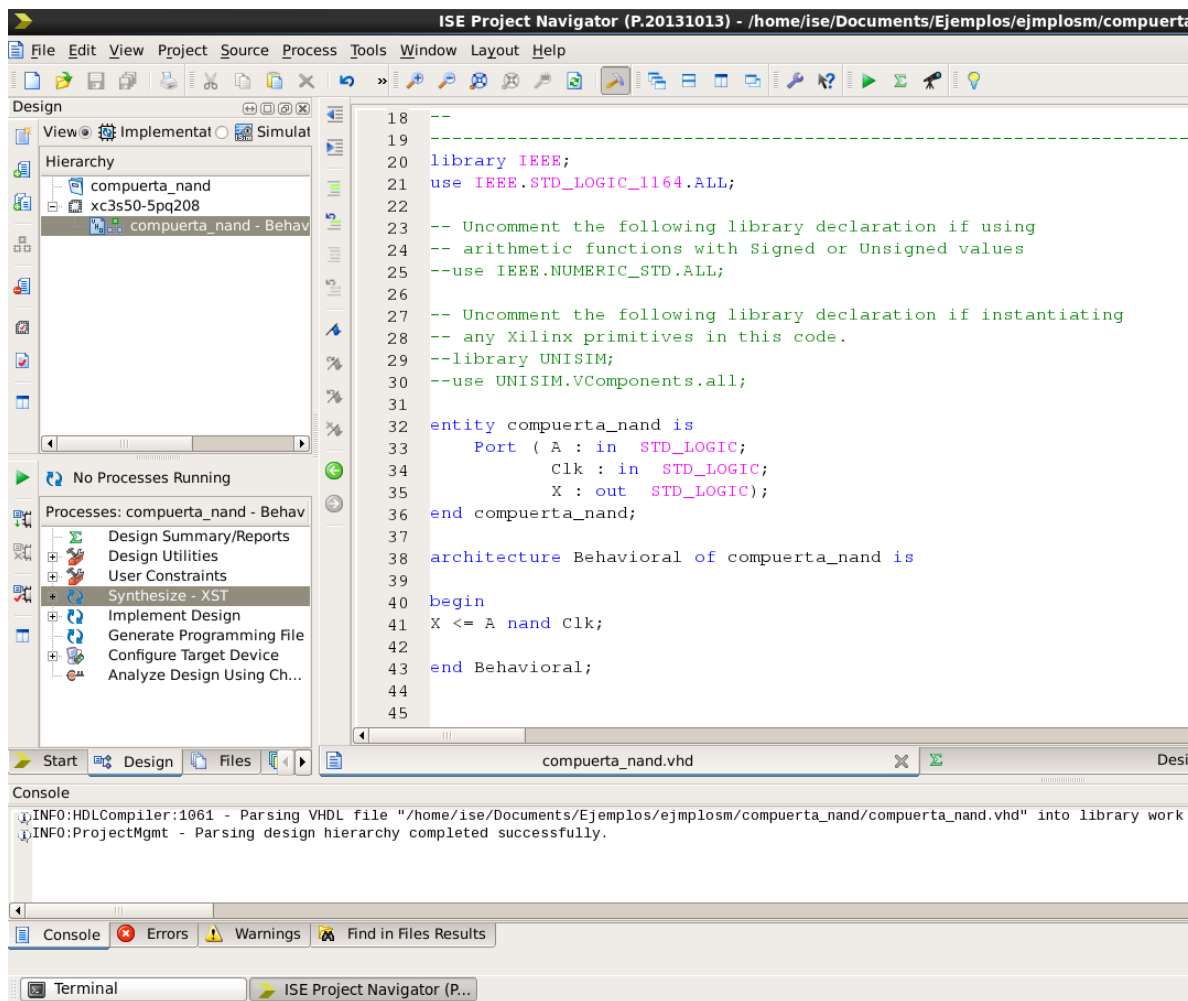


Figura 4.171 Proyecto sintetizado.

Se puede seleccionar la opción View RTL Schematic en la caja Synthesize – XST para que muestre como interpreto el proyecto.

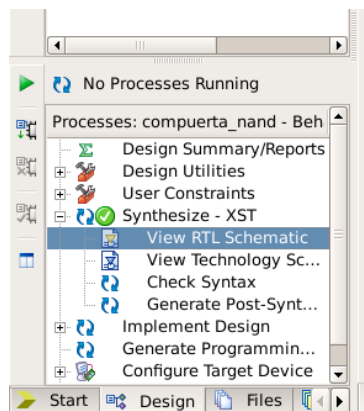


Figura 4.172 ventana para seleccionar el esquemático.

Se mostrará la siguiente ventana en donde se selecciona la opción Start with a schematic.

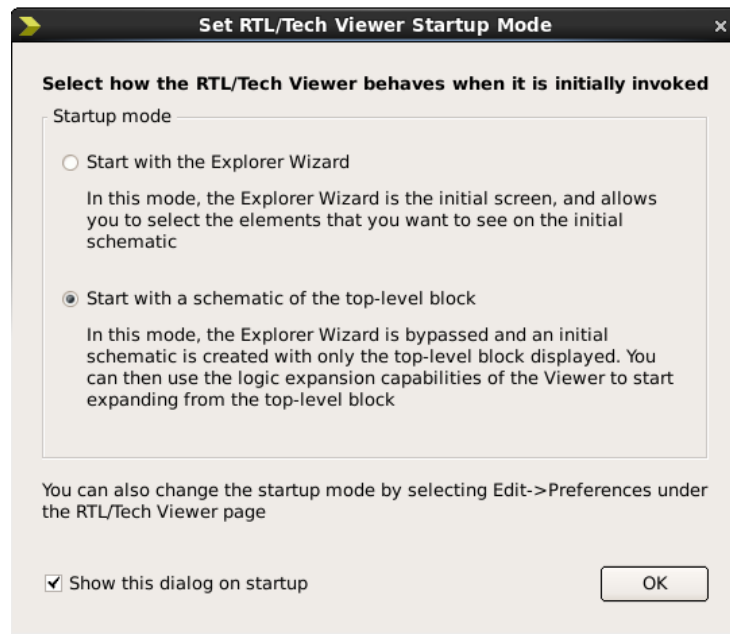


Figura 4.173 Ventana para seleccionar el esquemático.

La ventana que muestra es cómo interpreto el proyecto.

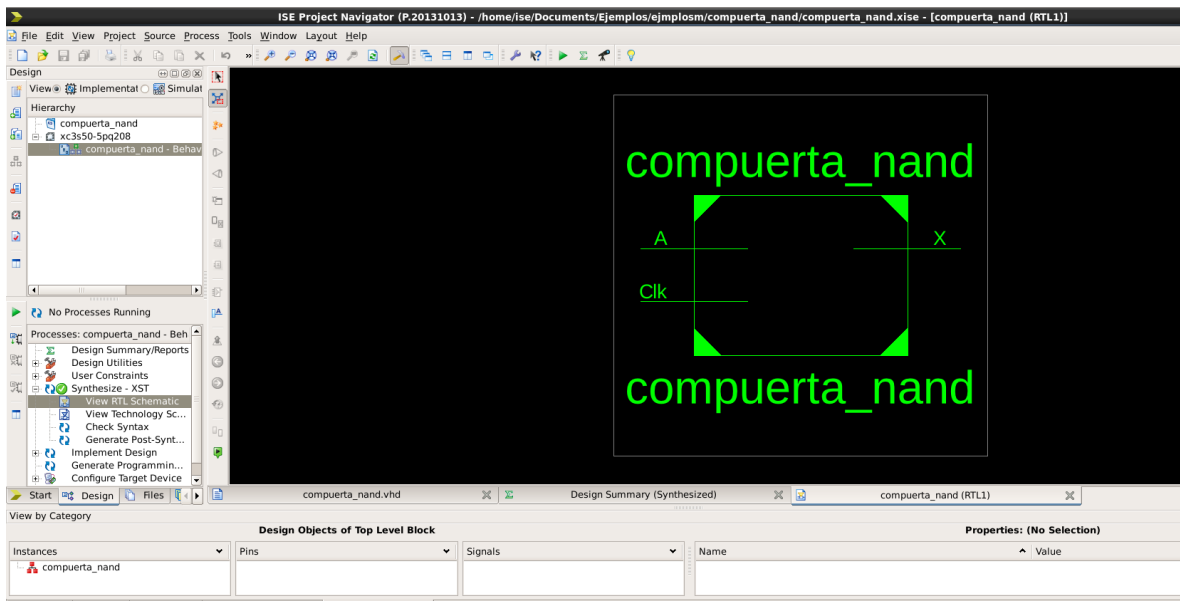


Figura 4.174 Ventana con la interpretación del proyecto.

Se le da doble clic en la caja donde están las entradas y salidas y nos mostrará la compuerta.

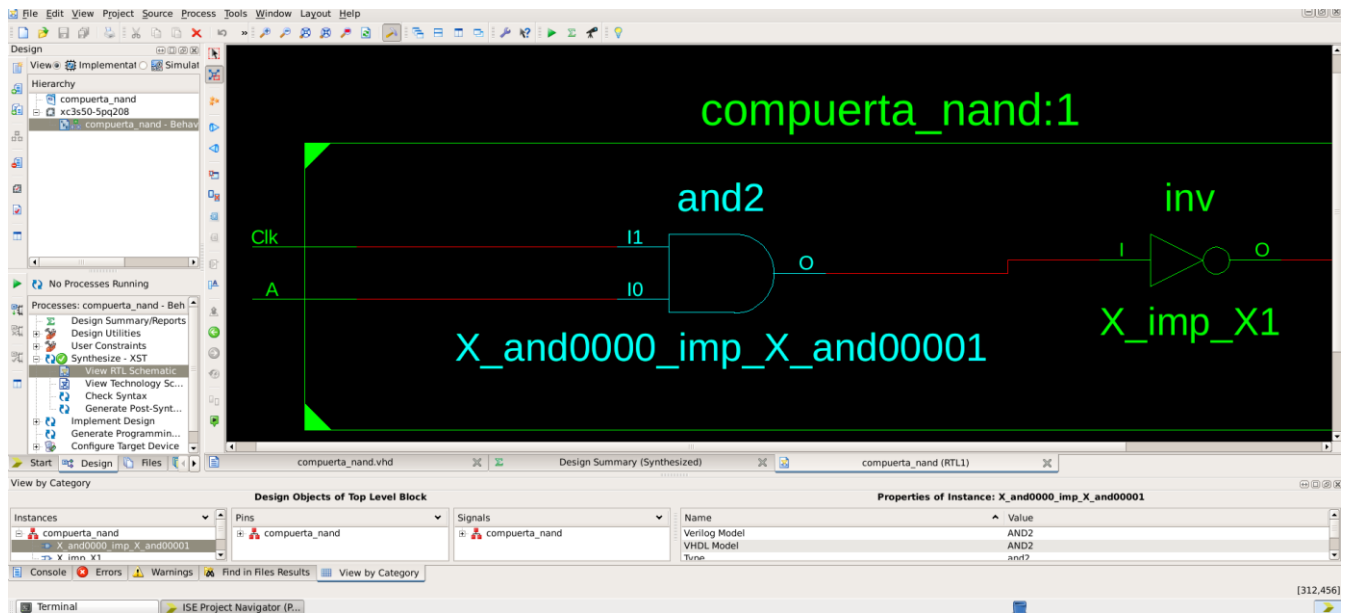


Figura 4.175 Ventana de la interpretación de la arquitectura del proyecto.

Para ver que la compuerta verdaderamente trabaje, seleccionamos la opción Simulation luego en la compuerta_nand y la opción nueva fuente y mostrara la siguiente ventana:

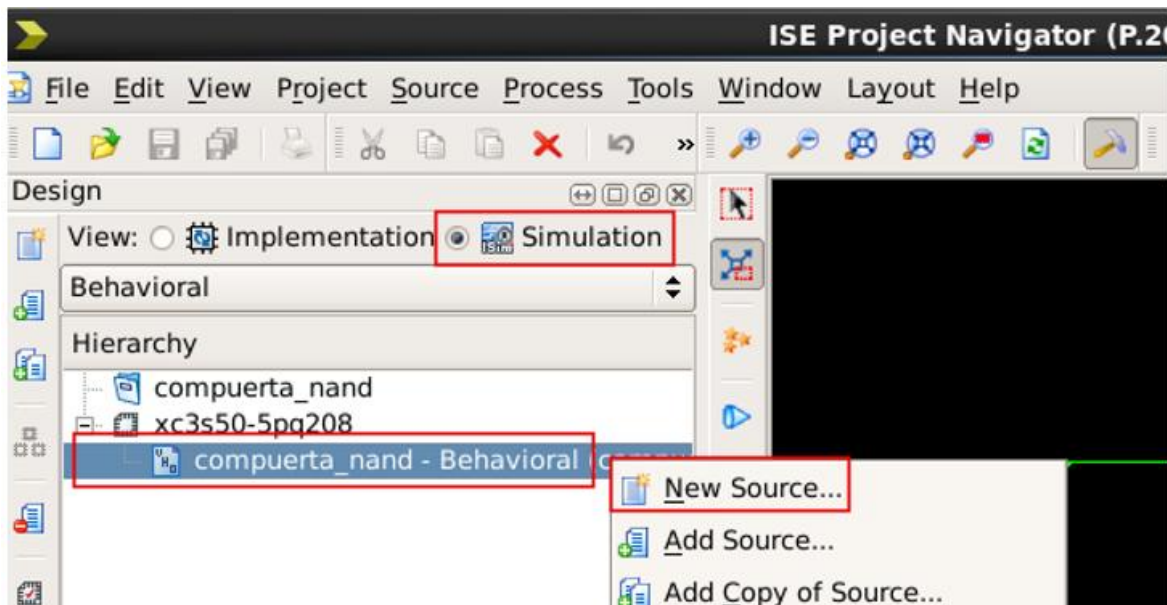


Figura 4.176 Ventana para la selección de la simulación.

Al darle clic nos mostrará una ventana en la que se selecciona la penúltima opción VHDL TEST Bench, le damos un nombre, se le da siguiente y siguiente y finalizar. Se generará una plantilla para la simulación como se muestra en la siguiente figura:

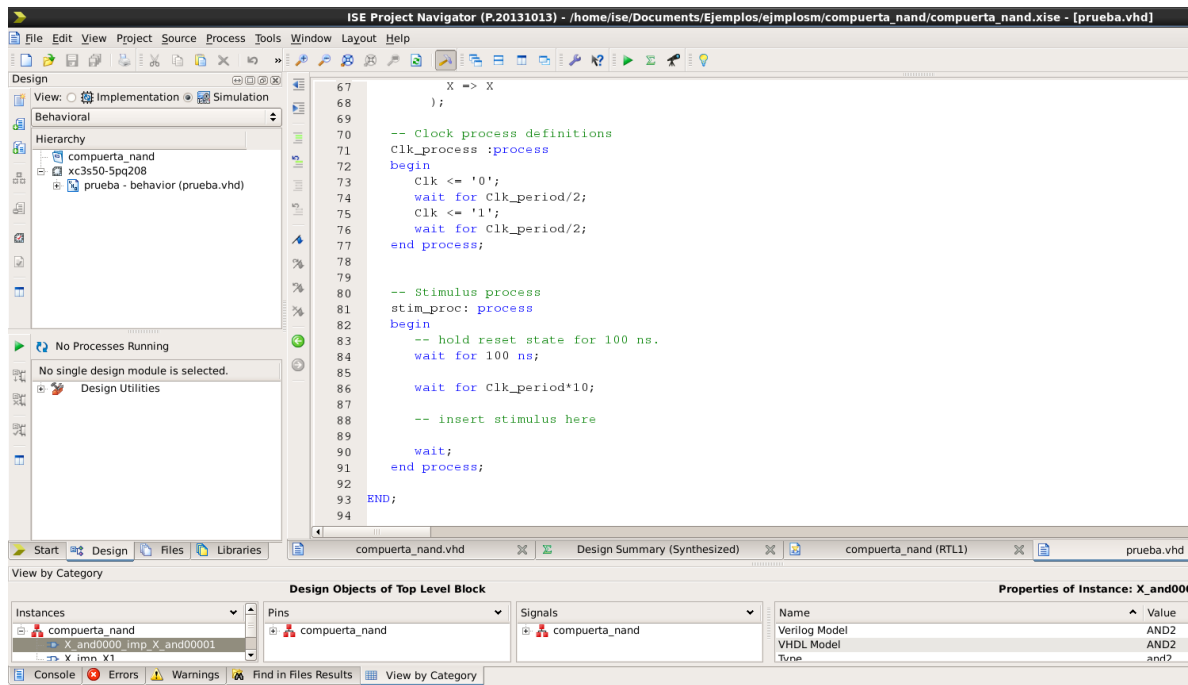


Figura 4.177 Ventana con la plantilla para escribir los estímulos(insert stimulus here) para la simulación.

Se escribe el siguiente código para valores de 0 y 1 para A y diferentes frecuencias del Clk.

```

A <= '1';
wait for Clk_period*10;
A <= '0';
wait for Clk_period*20;
A <= '1';
wait for Clk_period*30;
A <= '0';
wait for Clk_period*40;

```

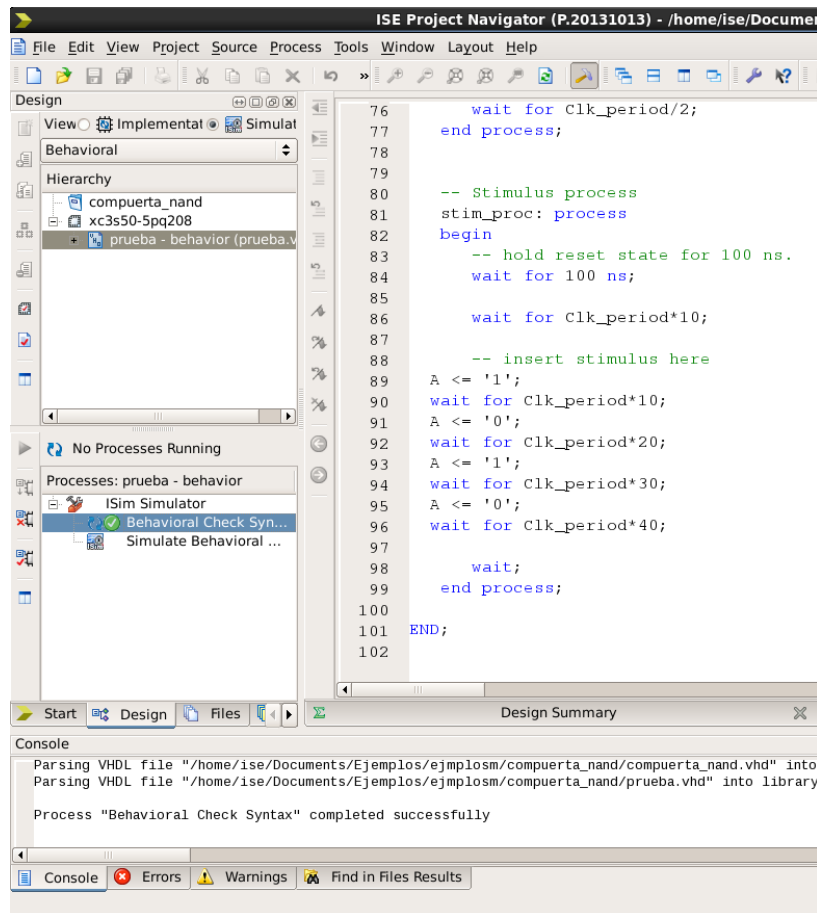


Figura 4.178 Ventana mostrando los estímulos para la simulación.

Se le da guardar y se selecciona la ceja prueba, aparecerá en la parte de abajo la opción ISim Simulator se le da clic y se selecciona la opción Simulate Behavioral Model.

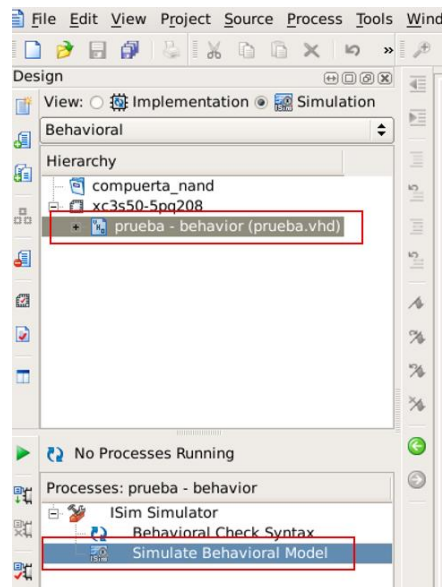


Figura 4.179 Ventana para seleccionar la simulación.

Dar clic en la caja Behavioral Chek Syntax para asegurar que no se tenga ningún error, si no lo hay aparecerá con una paloma en verde, luego Simulate Behavioral Model y nos mostrará la simulación.

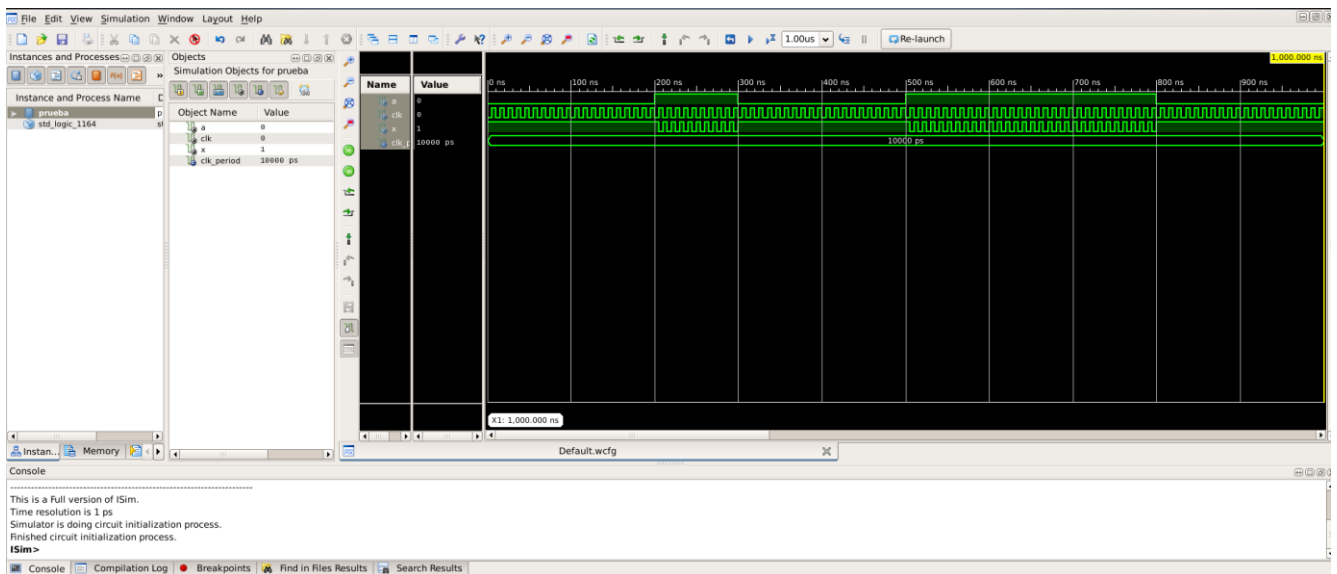


Figura 4.180 Ventana de la simulación del proyecto.

En la caja de herramientas se puede usar la lupa para hacer más grande o pequeña la señal de la simulación.

Ejemplo 2

Diseñar en VHDL un contador binario de 8 bits ascendente descendente con la señal de reset.

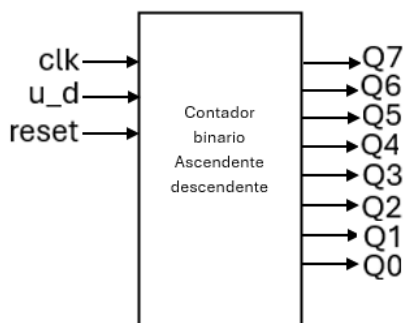


Figura 4.181 Contador ascendente descendente de 8 bits.

1. Abrir un nuevo proyecto (File > New Project) y darle el nombre.

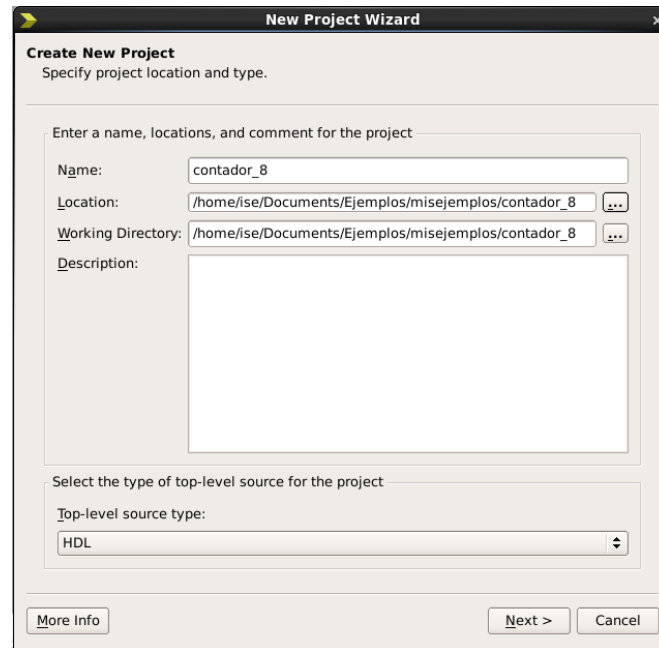


Figura 4.182 Ventana para dar el nombre y ruta del nuevo proyecto.

2. Darle Next para elegir la tarjeta y dispositivo en la que se desarrollará el proyecto y el lenguaje, se trabajará la Spartan3 y el lenguaje VHDL, una vez hecha la elección le damos Next.

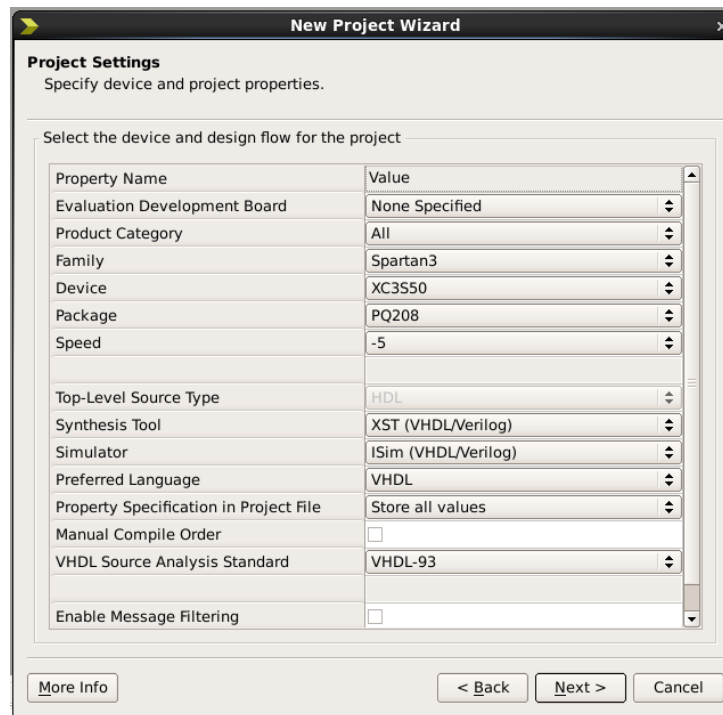


Figura 4.183 Ventana de selección de tarjeta y el lenguaje.

3. Se mostrará la ventana que los detalles del proyecto, verificamos que todo esté bien y seleccionamos Finish.

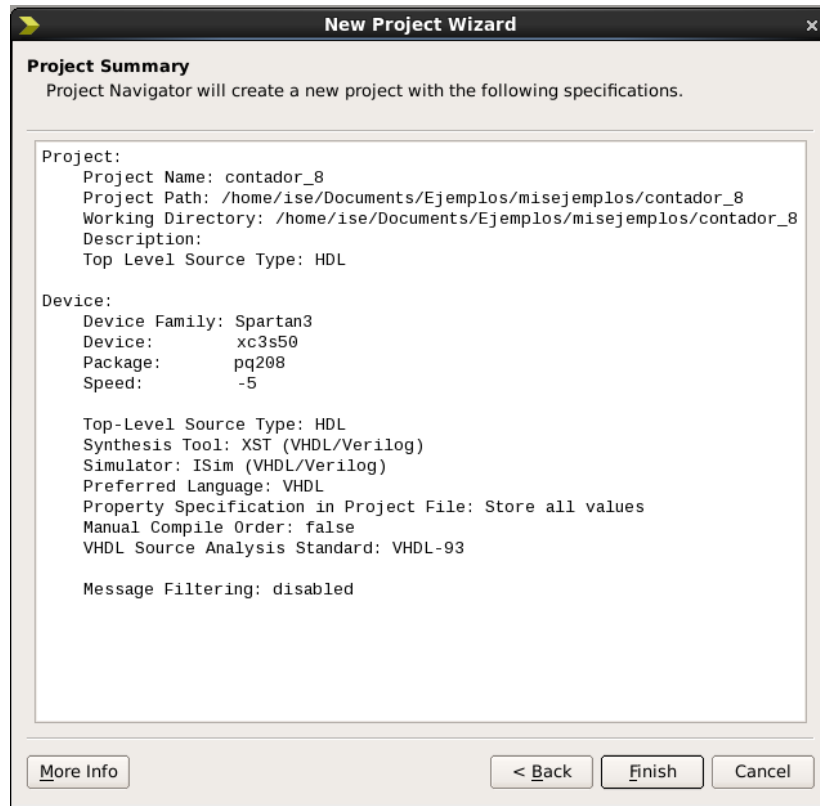


Figura 4.184 Ventana de los detalles del proyecto.

4. En la ventana donde esta implementa1 seleccionamos el número del dispositivo y se da doble clic y seleccionamos New Source como se muestra en la siguiente figura.

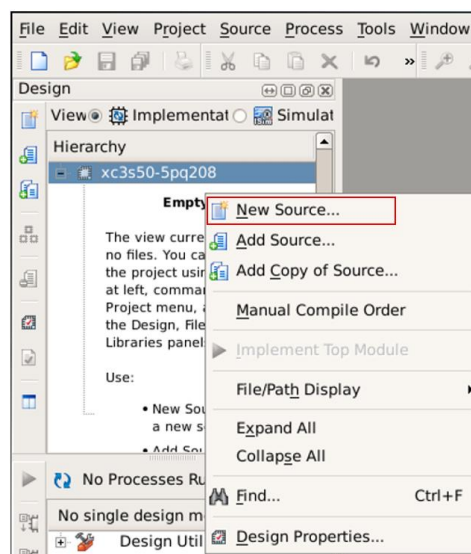


Figura 4.185 Ventana para seleccionar un nuevo recurso.

5. En la nueva ventana seleccionar la opción “VHDL module” y se le da un nombre, puede ser el mismo del proyecto, dar clic en Next.

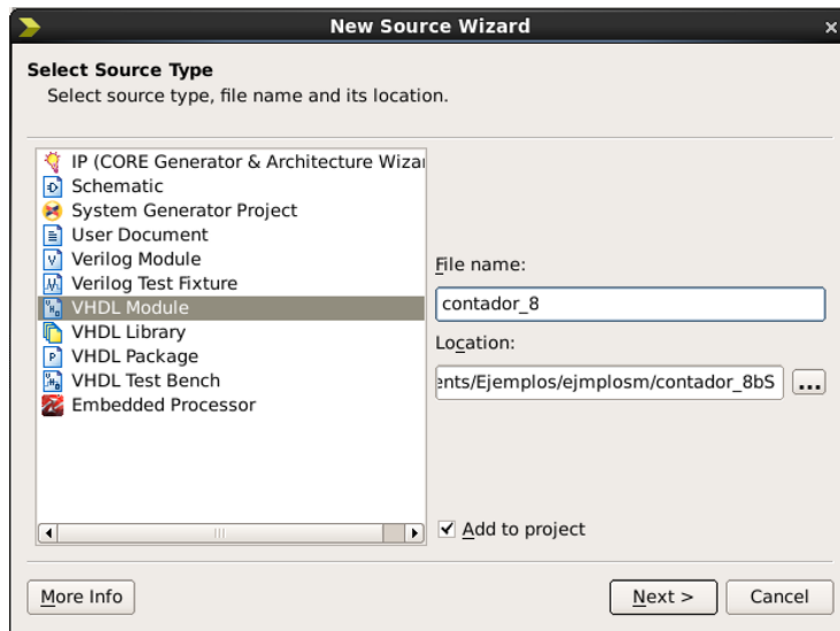


Figura 4.186 Ventana para seleccionar una nueva fuente “VHDL Module”.

6. en la siguiente ventana se ingresan las entradas y salidas de la entidad, reset, u_d, clk y el bus Q de 7 a 0 para las salidas del contador, dar clic en Next aparecerá una ventana detallando las entradas y salidas, se da clic en Finish.

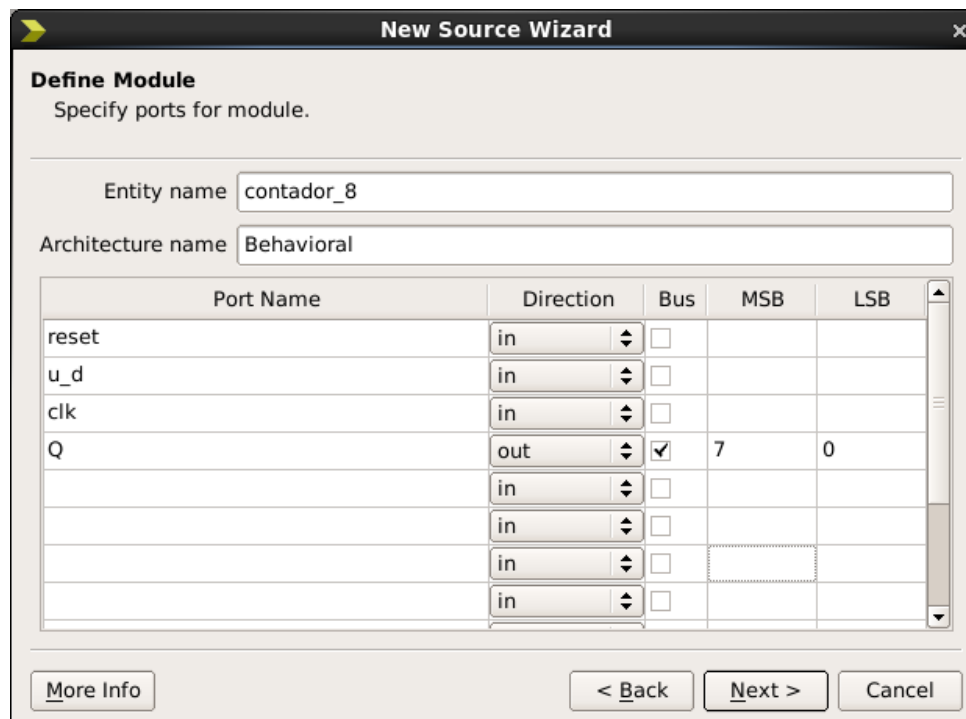


Figura 4.187 Ventana para ingresar las entradas y salidas del proyecto.

7. ventana que detalla la entidad, dar clic en Finish.

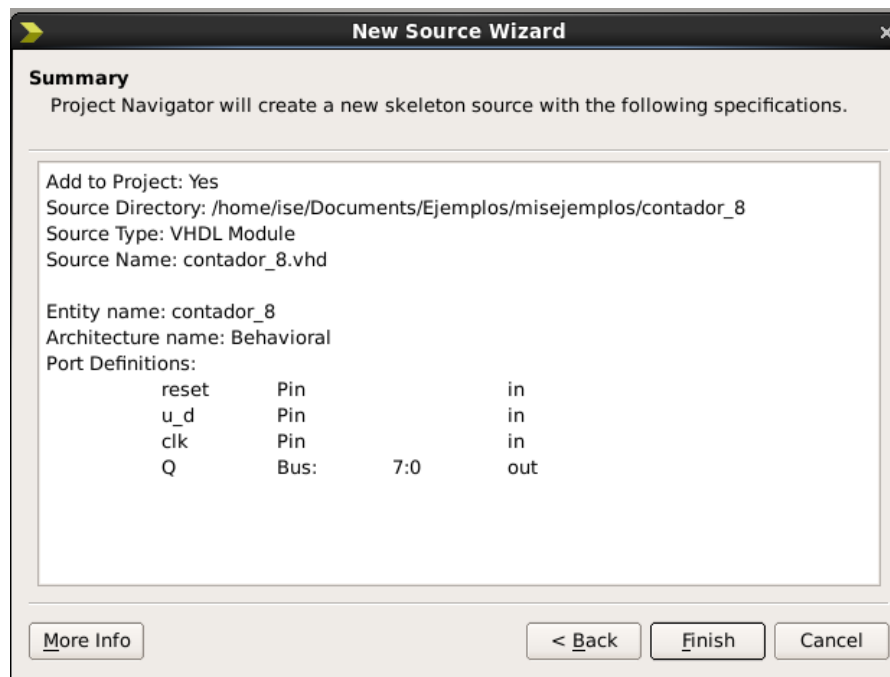


Figura 4.188 Ventana de los detalles de la entidad.

Al dar clic en la opción Finish mostrará la ventana con los comentarios, librerías, la entidad y el cuerpo de la arquitectura lista para editar en VHDL.

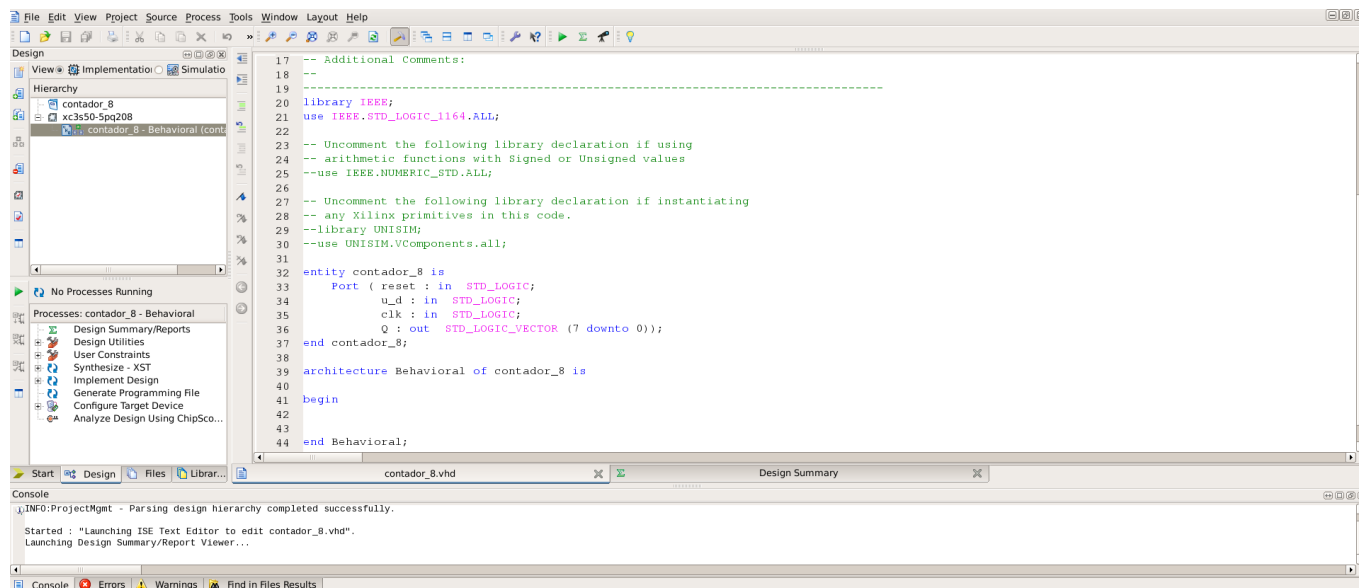


Figura 4.189 Ventana para editar la arquitectura del proyecto.

Código para el contador de 8 bits ascendente descendente con reset:

entity contador_8 is

Port (reset : in STD_LOGIC;

u_d : in STD_LOGIC;

clk : in STD_LOGIC;

Q : out STD_LOGIC_VECTOR (7 downto 0));

end contador_8;

architecture Behavioral of contador_8 is

signal Contador_8 : STD_LOGIC_VECTOR (7 downto 0) := "00000000";

begin

process(reset,clk)

begin

if reset='1' then

contador_8 <= "00000000";

elsif clk'event and clk = '1' then

if u_d = '1' then

contador_8 <= Contador_8 + 1;

else

contador_8 <= Contador_8 - 1;

end if;

Q <= contador_8;

end if;

end process;

end Behavioral;

Una vez escrito el código se da guardar y si hay errores los mostrará para que puedan ser corregidos.

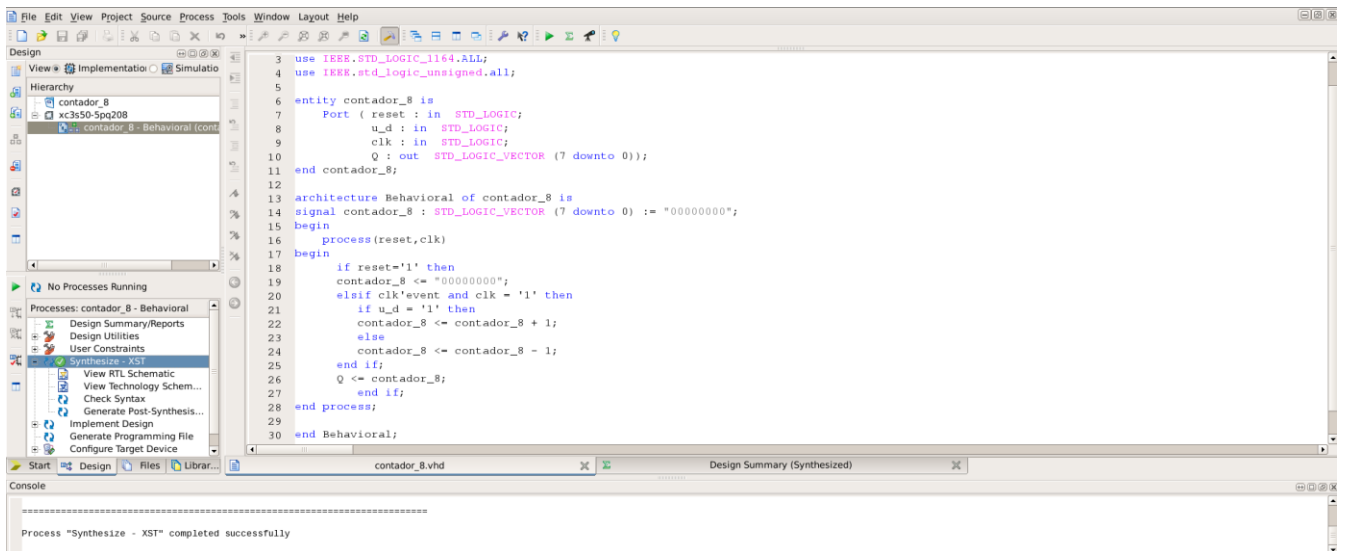


Figura 4.190 Código del proyecto ya terminado.

Se selecciona la opción Synthesize – XST luego la opción View RTL Schematic y mostrará la ventana donde se ve la interpretación del proyecto.

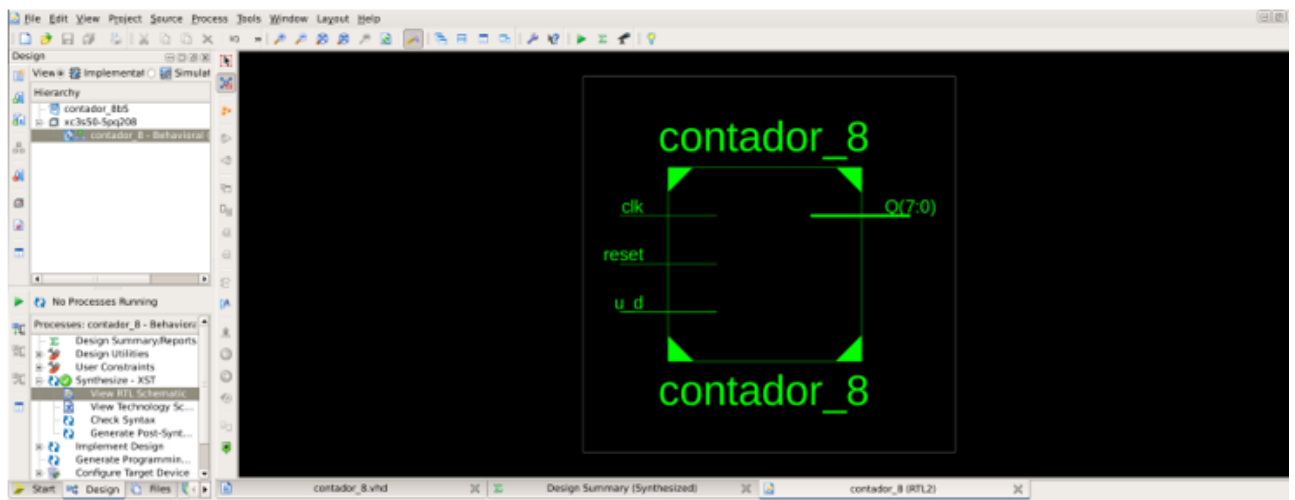


Figura 4.191 Ventana mostrando el esquemático del proyecto.

Se le da doble clic sobre la caja del contador y con la lupa reducimos el tamaño y nos mostrará el esquemático interno del contador.

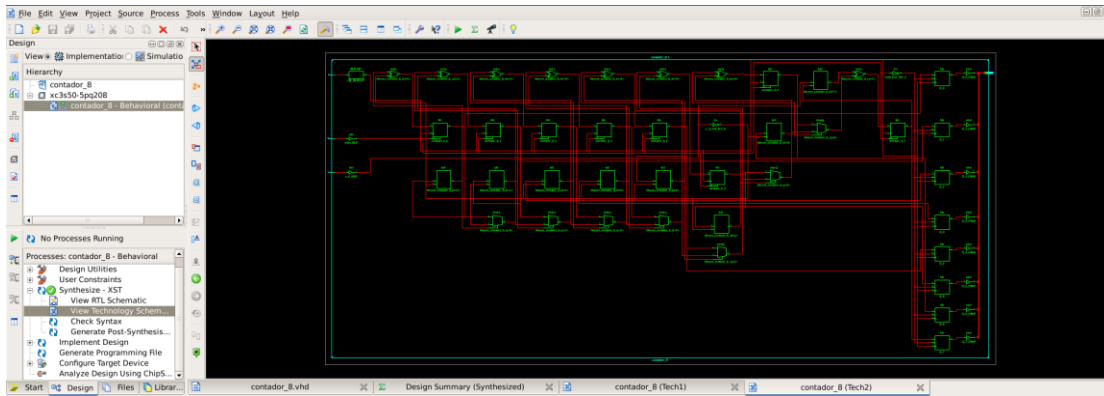


Figura 4.192 Ventana mostrando la esquemático interno.

Para hacer la simulación, se cambia en la ventana izquierda a la ceja Simulation, el proyecto y Behavioral Check Syntax para checar que no haya errores, si no hay pondrá una paloma en verde.

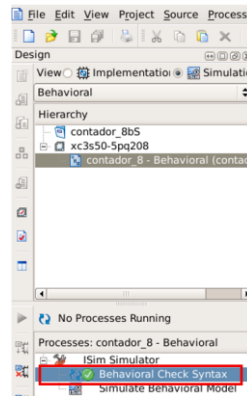


Figura 4.193 Ventana para checar que no hay errores en el código.

Ahora se da clic en la ceja Simulate Behavioral Model y se mostrará la ventana de la simulación, en ella se puede ingresar los valores de las variables dándole clic en la variable que desea cambiar.

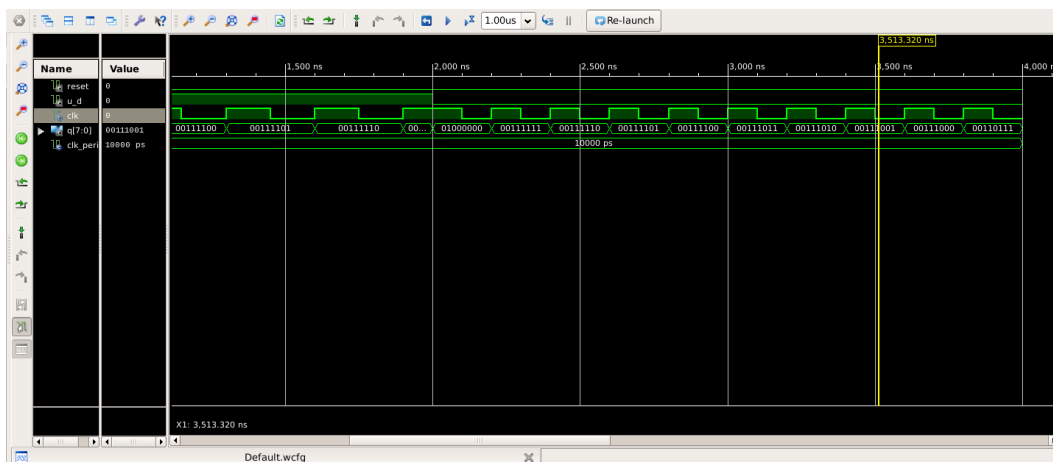


Figura 4.194 Ventana para la simulación del proyecto.

Para cambiar el valor de alguna variable que se quiera simular se sigue la secuencia para cada variable como se muestra en la siguiente figura:

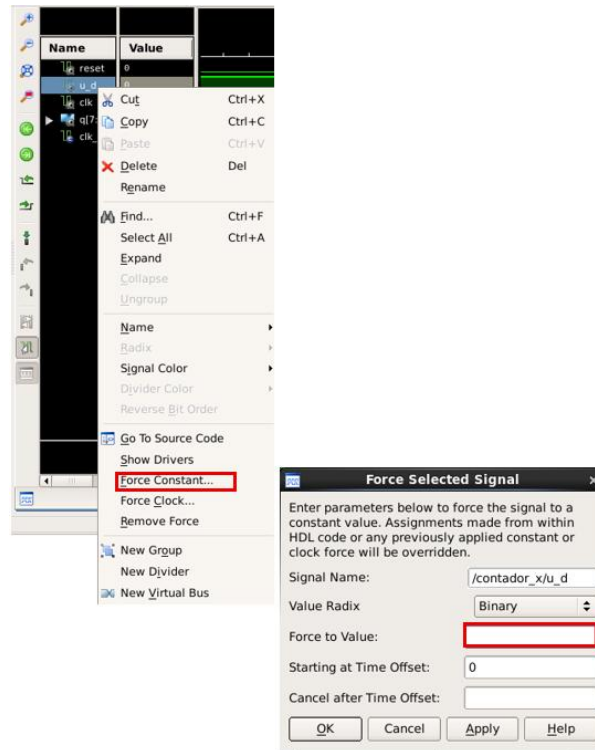


Figura 4.195 Ventana para cambiar la variable u_d.

Ejemplo 3

Diseñar en VHDL un multiplexor de 8 entradas de 8 bits y una salida de 8 bits como se muestra en la siguiente figura:

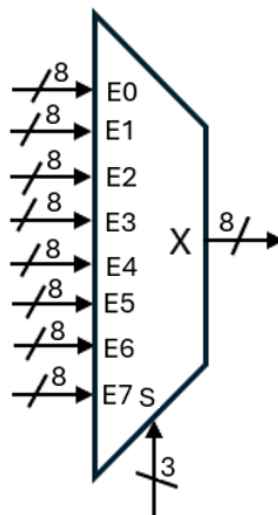


Figura 4.196 Multiplexor de 8 buses de 8 bits a un bus de 8 bits de salida.

1.. Iniciamos con un nuevo proyecto al que se la llamara mux_8, ya sólo se mostraran las ventanas más importantes del mismo.

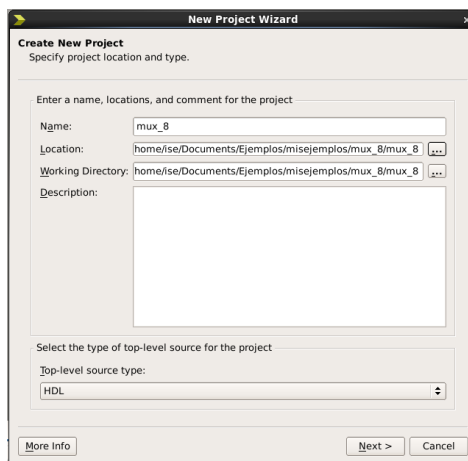


Figura 4.197 Ventana de inicio del nuevo proyecto mux_8.

Se elije la tarjeta espartan3, lenguaje VHDL y las opciones que correspondan a la tarjeta en al que se trabaja.

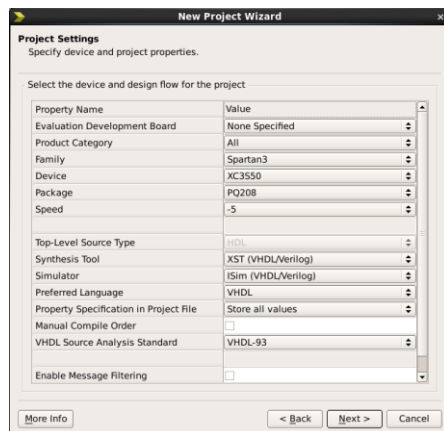


Figura 4.198 Ventana para la selección de la tarjeta y sus parámetros.

Una vez seleccionada la nueva fuente, VHDL Module y haberle dado el nombre nos mostrará una nueva ventana para definir las entadas, salidas y su tipo.

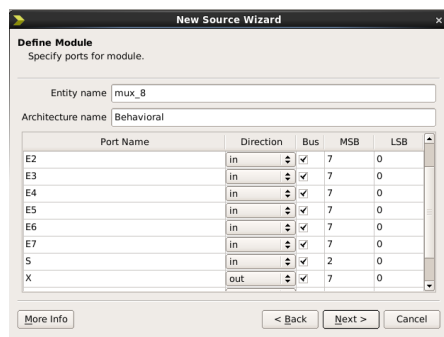


Figura 4.199 Ventana con la definición de entradas y salidas y su tipo.

Al darle Next y luego Finish nos mostrará la ventana principal para editar el código del proyecto, en este caso se le quito los comentarios, estos serán de acuerdo con las necesidades de quien haga el diseño.

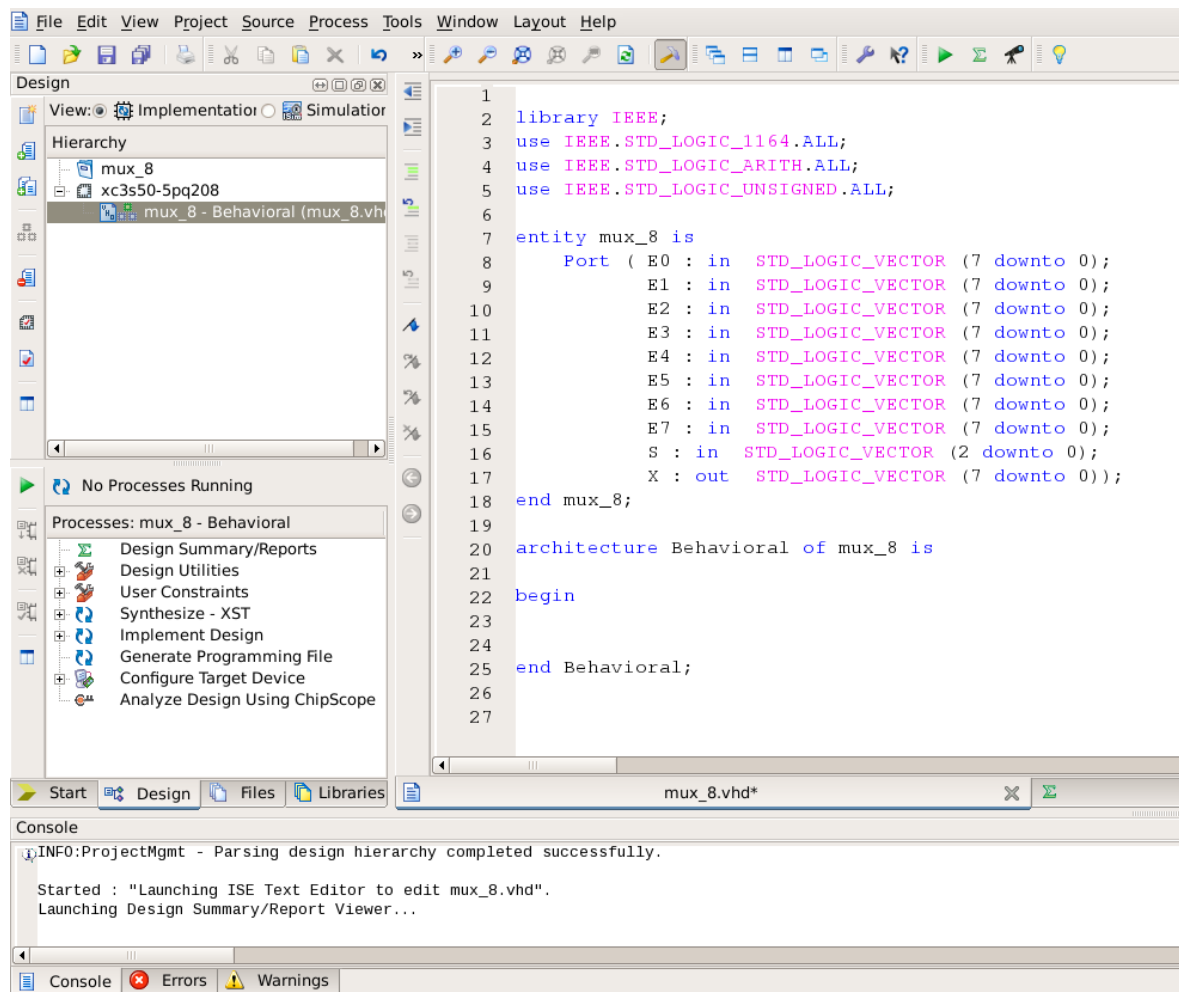


Figura 4.200 Ventana del proyecto mux_8 para editar el código.

Para esta aplicación es recomendable usar la opción with < choice_expression> select. pero queda al gusto de quien lo diseña.

architecture Behavioral of mux_8 is

begin

with S select

X <= E0 when "000",
 E1 when "001",
 E2 when "010",
 E3 when "011",

E4 when "100",
 E5 when "101",
 E6 when "110",
 E7 when others

end Behavioral;

Al terminar de escribir el código se guarde y se revisa que no haya errores de sintaxis, si todo está bien quedará de la siguiente manera:

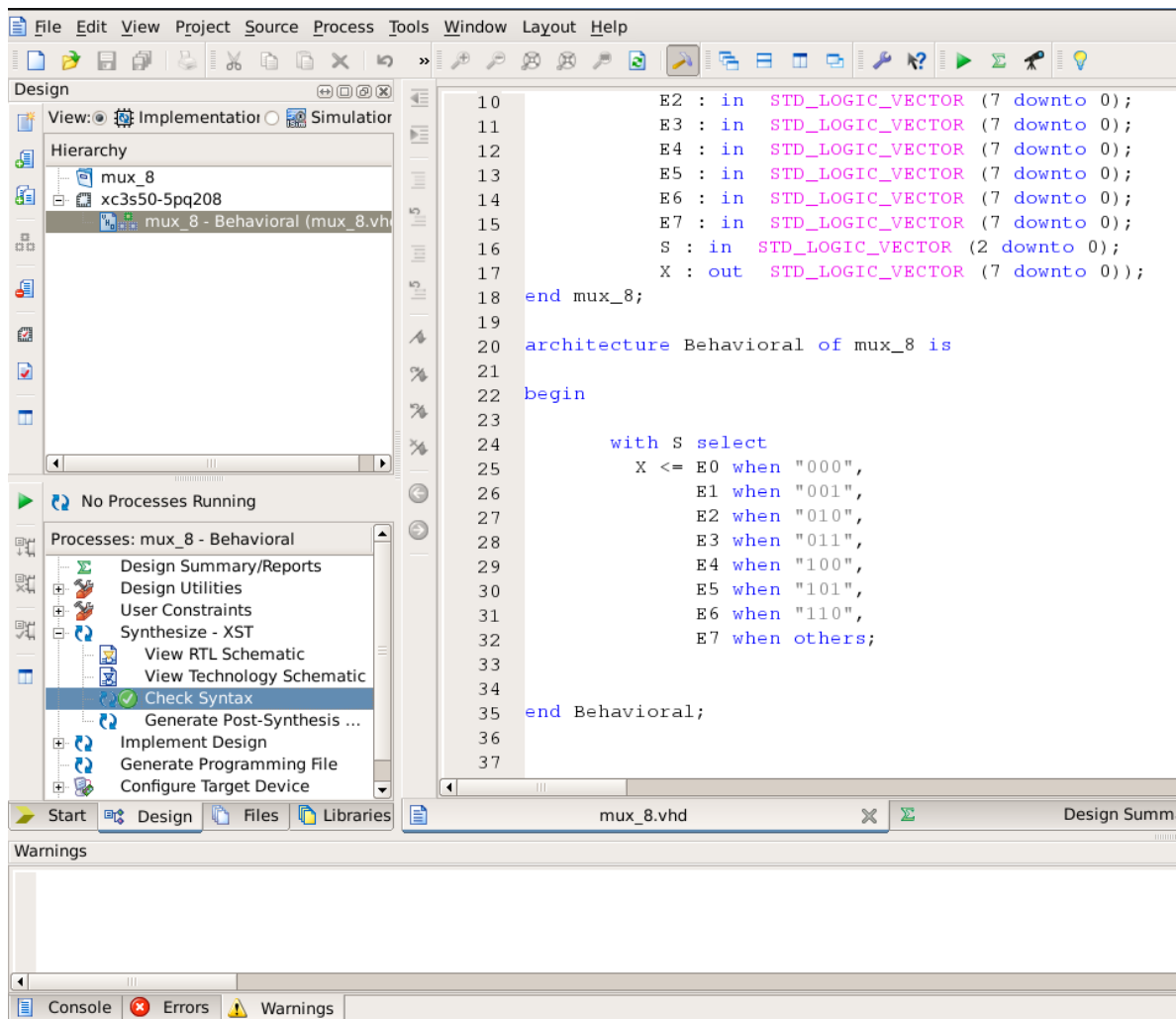


Figura 4.201 Código terminado y revisada su sintaxis.

Ahora se sintetiza y se selecciona la opción que muestra el esquemático y muestra la siguiente ventana:

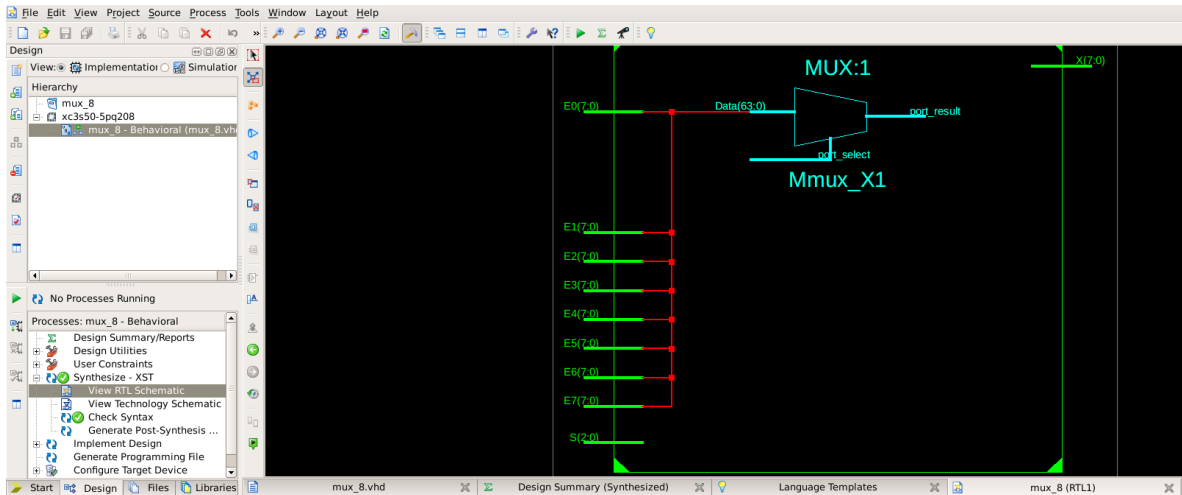


Figura 4.202 Ventana del RTL Schematic.

Se cierra la ventana y se selecciona la opción de simulación como en los ejemplos anteriores, se selecciona la ceja de Simulation y Simulte Behavioral Model, una vez estando en la simulación se pueden actualizar las variables al seleccionarla y darle clic, mostrará una ventana para introducir el nuevo valor de la variable.

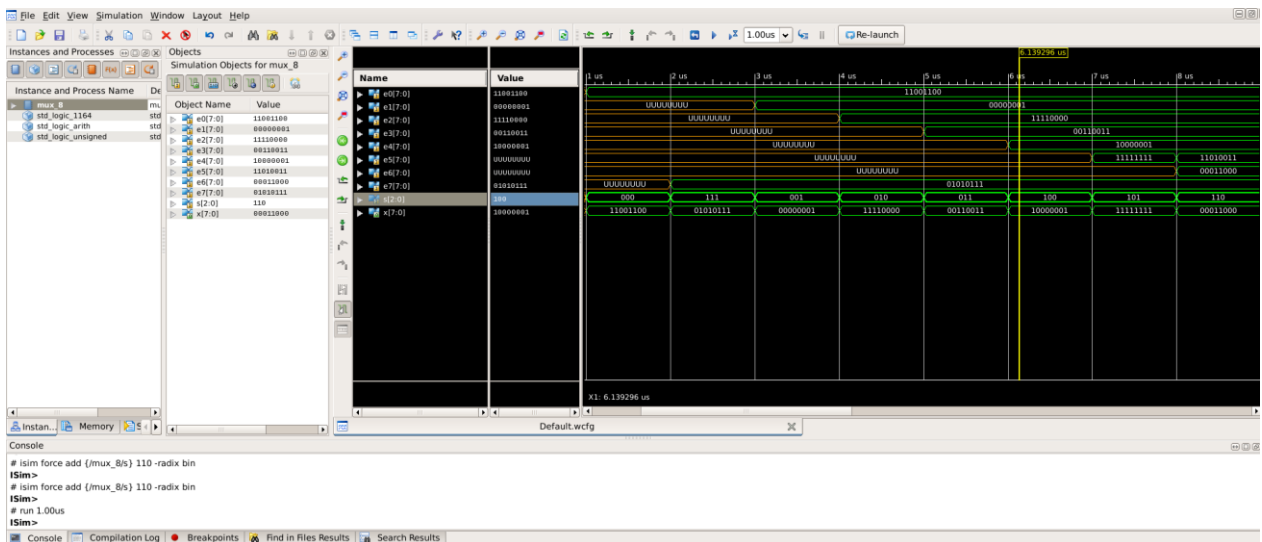


Figura 4.203 Ventana de simulación con el proyecto funcionando.

Ejemplo 4

Diseñar un circuito comparador de magnitud para dos datos de 8 bits y que tenga tres salidas para indicar el resultado de la comparación, el diagrama de bloque se muestra en la siguiente figura:

Para este diseño se usará la estructura process vista en la unidad 3.5.

Se crea el nuevo proyecto.

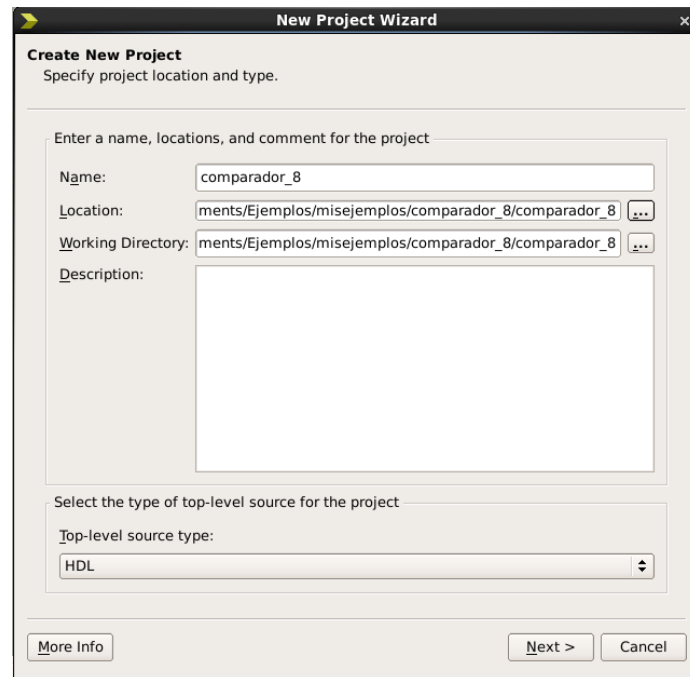


Figura 4.204 Ventana de creación de nuevo proyecto.

Selección de la nueva fuente y módulo VHDL.

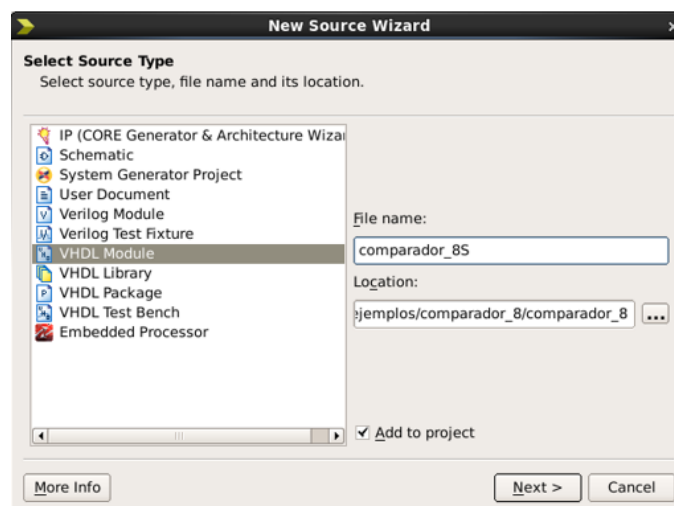


Figura 4.205 Ventana de nueva fuente y selección del módulo VHDL.

Avanzamos hasta la ventana para asignar el nombre de las variables y su tipo.

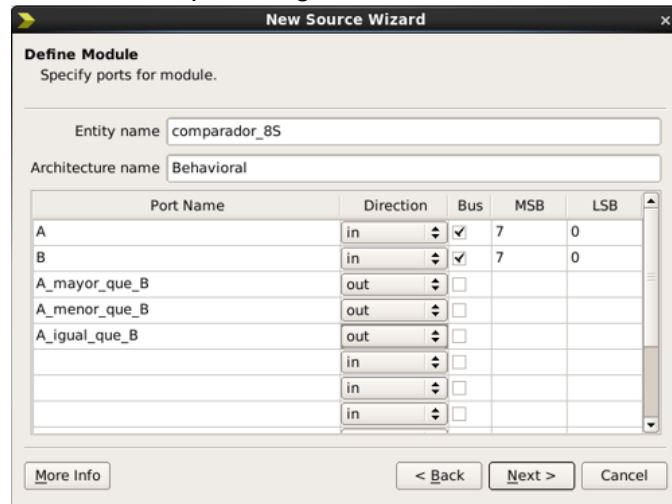


Figura 4.206 Ventana con la asignación de las variables y su tipo.

Al avanzar i finalizar se muestra la ventana principal para editar el código, es conveniente quitar los comentarios que trae por default y agregar los que se crea conveniente.

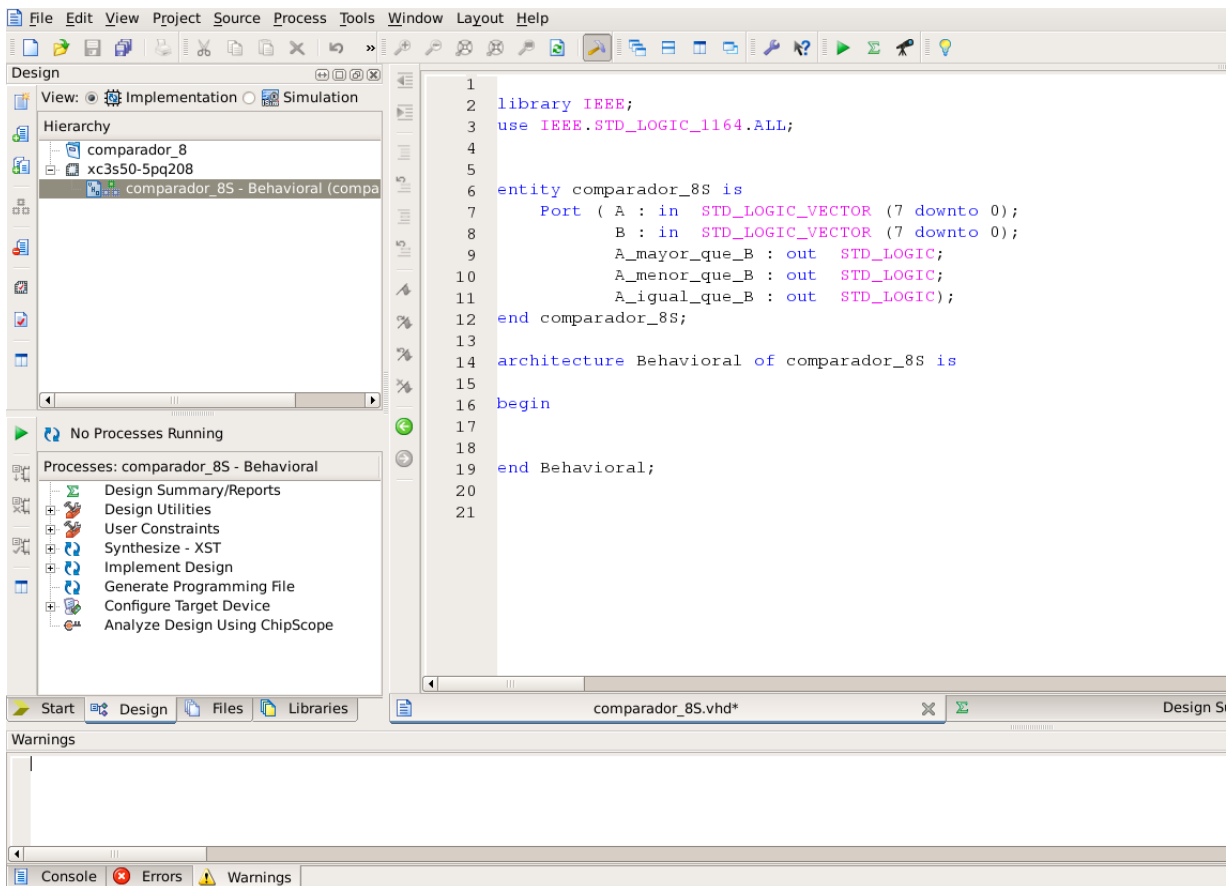


Figura 4.207 Ventana para editar el código.

El código para el comparador_8 usando process es el siguiente:

```
process(A,B)
begin
    if (A>B) then
        A_mayor_que_B <= '1';
        A_menor_que_B <= '0';
        A_igual_que_B <= '0';
    elsif (A=B) then
        A_mayor_que_B <= '0';
        A_menor_que_B <= '0';
        A_igual_que_B <= '1';
    elsif (A<B) then
        A_mayor_que_B <= '0';
        A_menor_que_B <= '1';
        A_igual_que_B <= '0';
    end if ;
end process;
```

Al escribir el código y checar la sintaxis, si hay error queda de la siguiente manera:

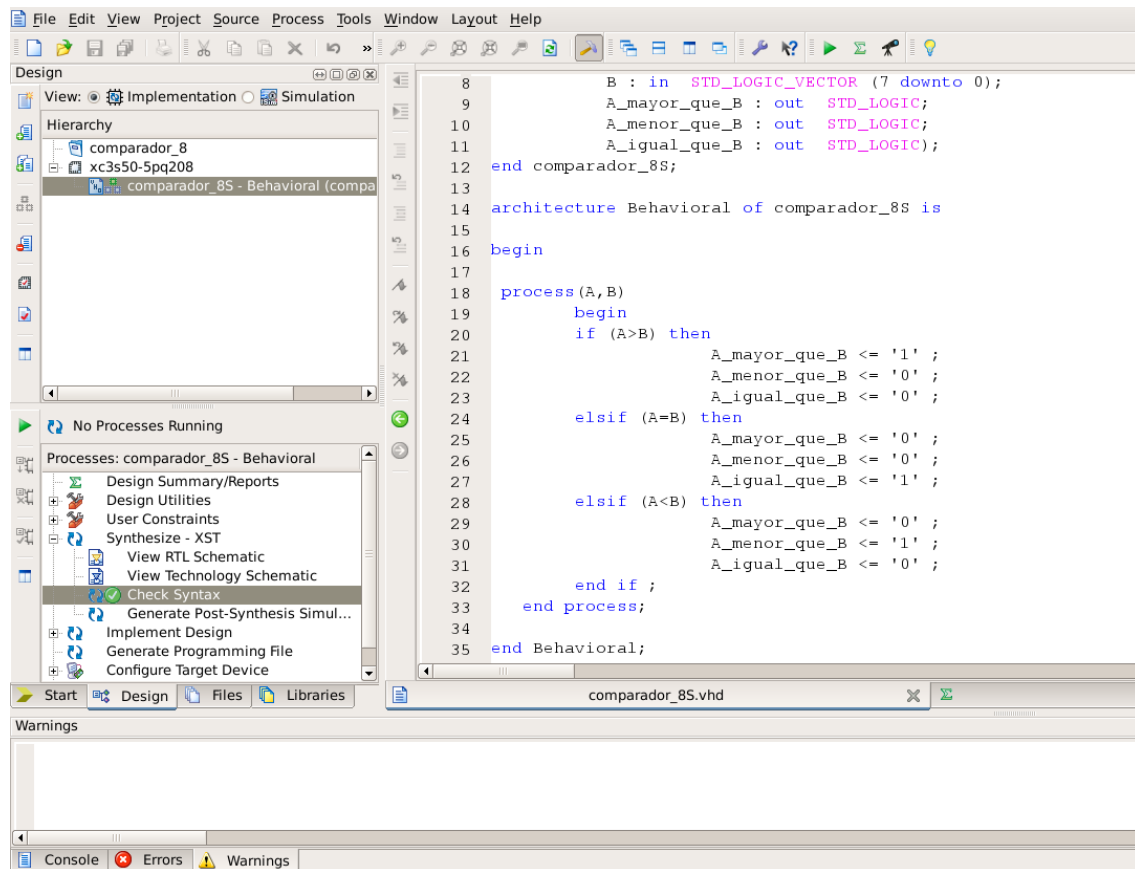


Figura 4.208 Ventana con el código y sintaxis correcta.

En seguida se sintetiza y si no hay errores se pide que muestre en la opción View RTL Schematic el esquemático.

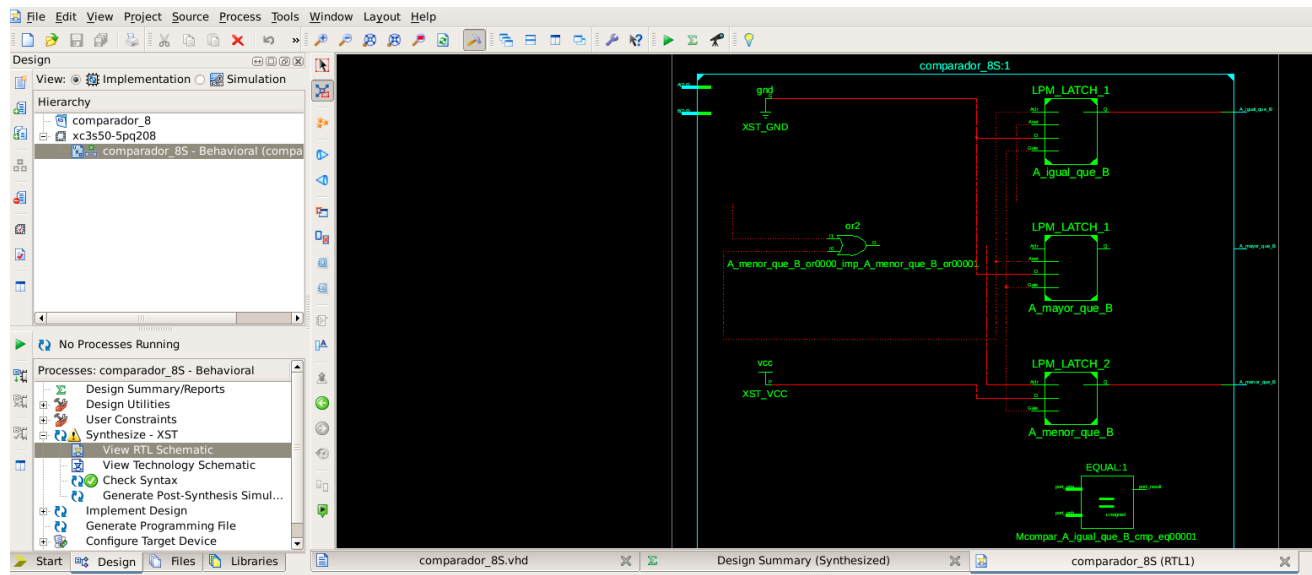


Figura 4.209 Ventana del esquemático del comparador_8.

Ya terminado el comparador_8 se selecciona la opción de simulación y en la ventana que aparece se hace la prueba con distintos valores de A y B dando clic en la variable para cambiar sus valores.

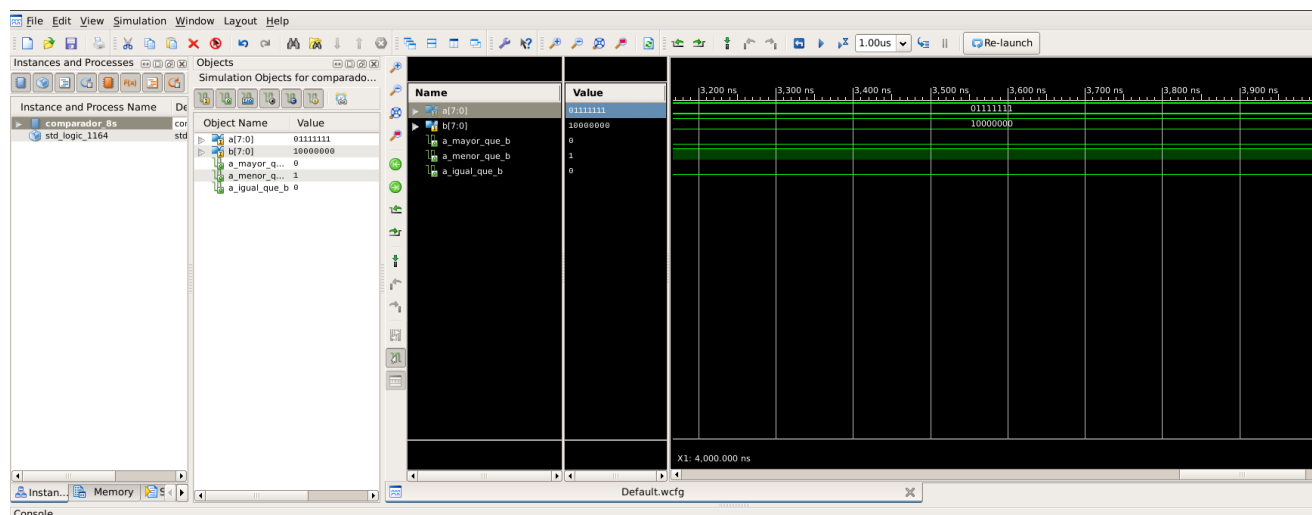


Figura 4.210 Ventana del simulador con el proyecto comparador_8.

El comparador trabaja correctamente.

Ejemplo 5

Diseñar en VHDL un registro de corrimiento de 8 bits entrada serie salida serie, la lógica de funcionamiento se muestra en la siguiente figura:

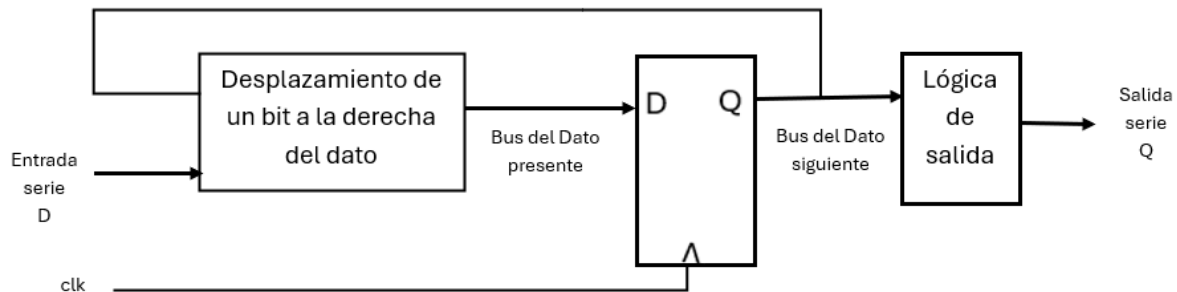


Figura 4.211 estructura del registro de corrimiento entrada serie salida serie.

Al bus del dato siguiente le llamaremos B_S y a la salida Q y al dato de entrada DS.

Iniciando el nuevo proyecto se verá la ventana siguiente:

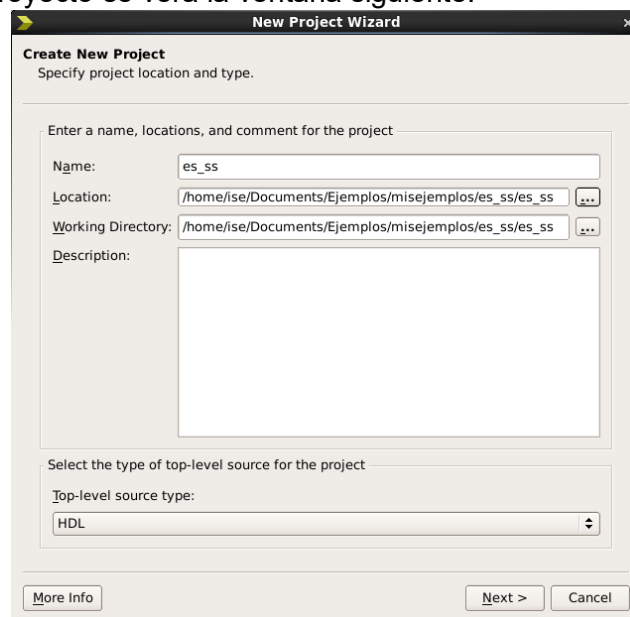


Figura 4.212 Ventana de nuevo proyecto es_ss.

Al seleccionar la nueva fuente queda de la siguiente manera:

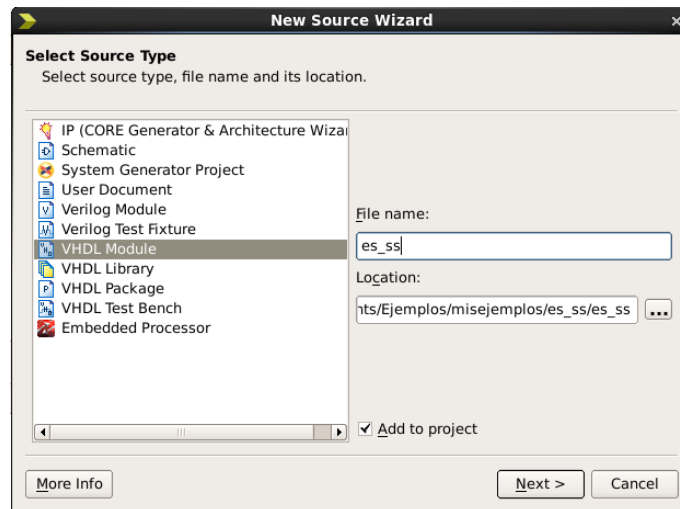


Figura 4.213 Selección del módulo VHDL en la nueva fuente.

Al darle Next mostrará la ventana para asignar las entradas y salidas, así como su tipo, esto se muestra en la siguiente figura:

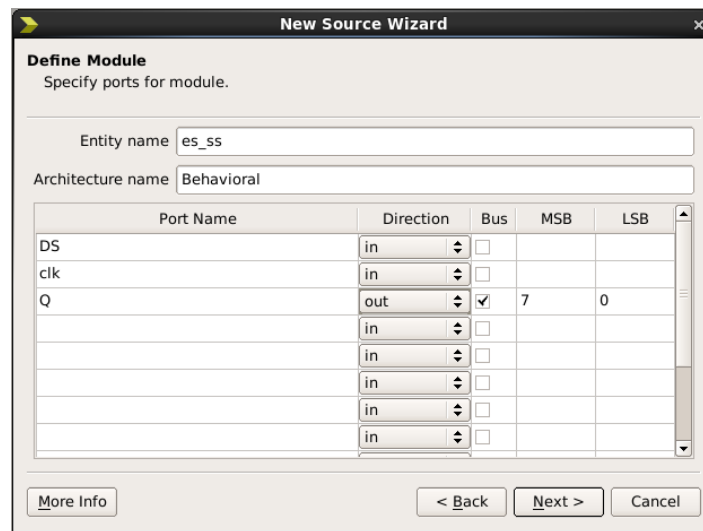


Figura 4.214 Ventana de asignación de entradas, salidas y tipo.

Al darle Next y Finish mostrará la ventana para editar el código, se le quita los comentarios que genera por default y si es necesario agregar los comentarios propios.

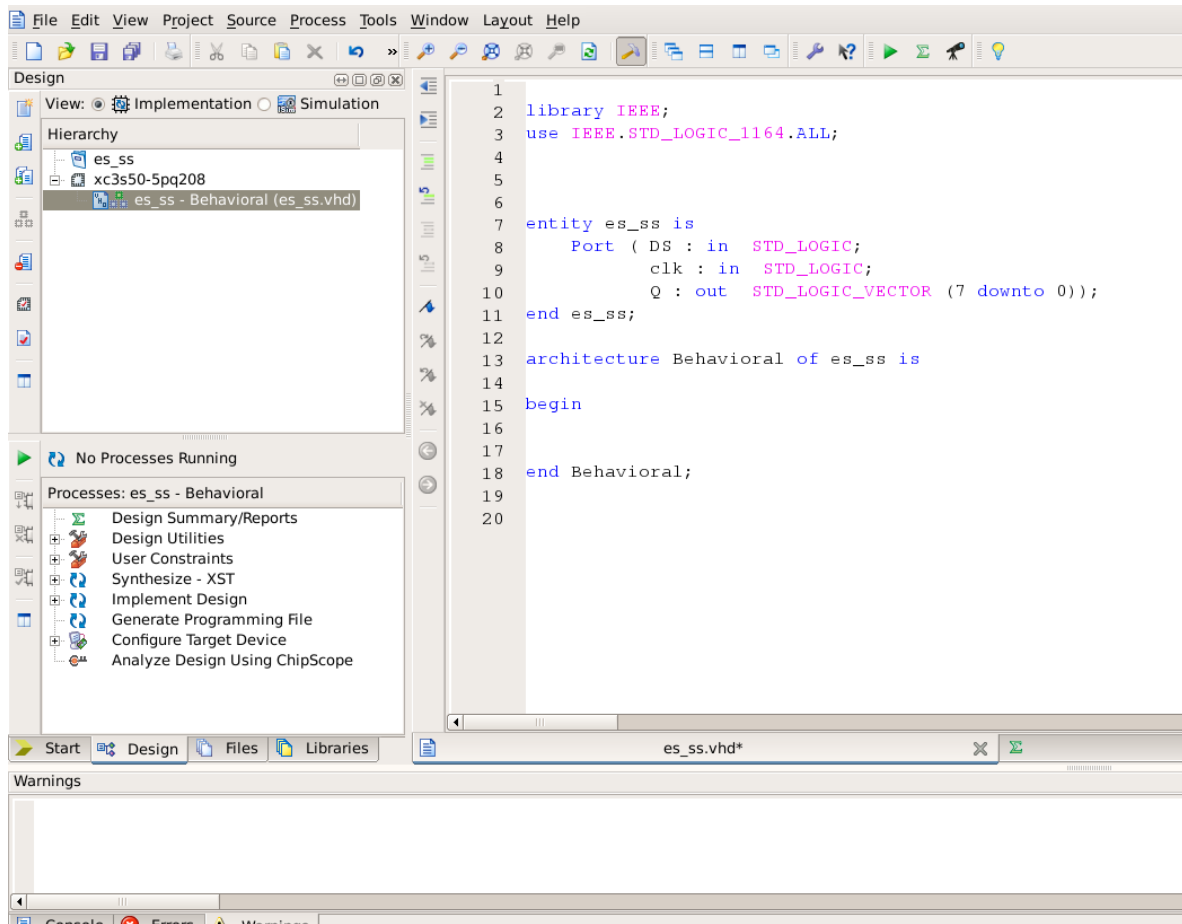


Figura 4.215 Ventana para la edición del código del registro entrada seri salida serie.

Es necesario agregar dos señales internas para el manejo de los buses el B_S y Q, esta quedaría de la siguiente manera:

```
signal bus_presente, bus_siguiente : std_logic_vector(7 downto 0):= (others => '0');
```

En estas señales se asegura que al inicio tengan 0.

Para el manejo del Flip flop llamado registro de estado se puede manejar un process de la siguiente manera:

```
Process (clk)
```

```
Begin
```

```
    If clk' evento and clk = '1' then
```

```
        bus_presente <= bus_siguiente;
```

```
    endif
```

```
end process;
```

Para la lógica del estado siguiente se hace una concatenación del bit de entrada DS como bit más significativo con los 7 bits más significativos del estado siguiente descartando el bit 0, esto queda de la siguiente manera:

```
bus_presente <= DS & bus_siguiete(7 downto 1);
```

Para la salida sólo asignamos a Q el bit menos significativo de la siguiente manera:

```
Q <= bus_siguiete (0);
```

El código completo es el siguiente

```
library IEEE;  
use IEEE.STD_LOGIC_1164.ALL;
```

```
entity es_ss is  
  Port ( DS : in  STD_LOGIC;  
        clk : in  STD_LOGIC;  
        Q : out STD_LOGIC);  
end es_ss;
```

```
architecture Behavioral of es_ss is
```

```
  signal bus_presente , bus_siguiete : std_logic_vector(7 downto 0):= (others => '0');
```

```
begin
```

```
  Process (clk)
```

```
  Begin
```

```
    If clk' event and clk = '1' then
```

```
      bus_presente <= bus_siguiete;
```

```
    end if;
```

```
  end process;
```

```
  bus_presente <= DS & bus_siguiete(7 downto 1);
```

```
  Q <= bus_siguiete (0);
```

```
end Behavioral;
```

El código editado y con la prueba de sintaxis queda de la siguiente forma:

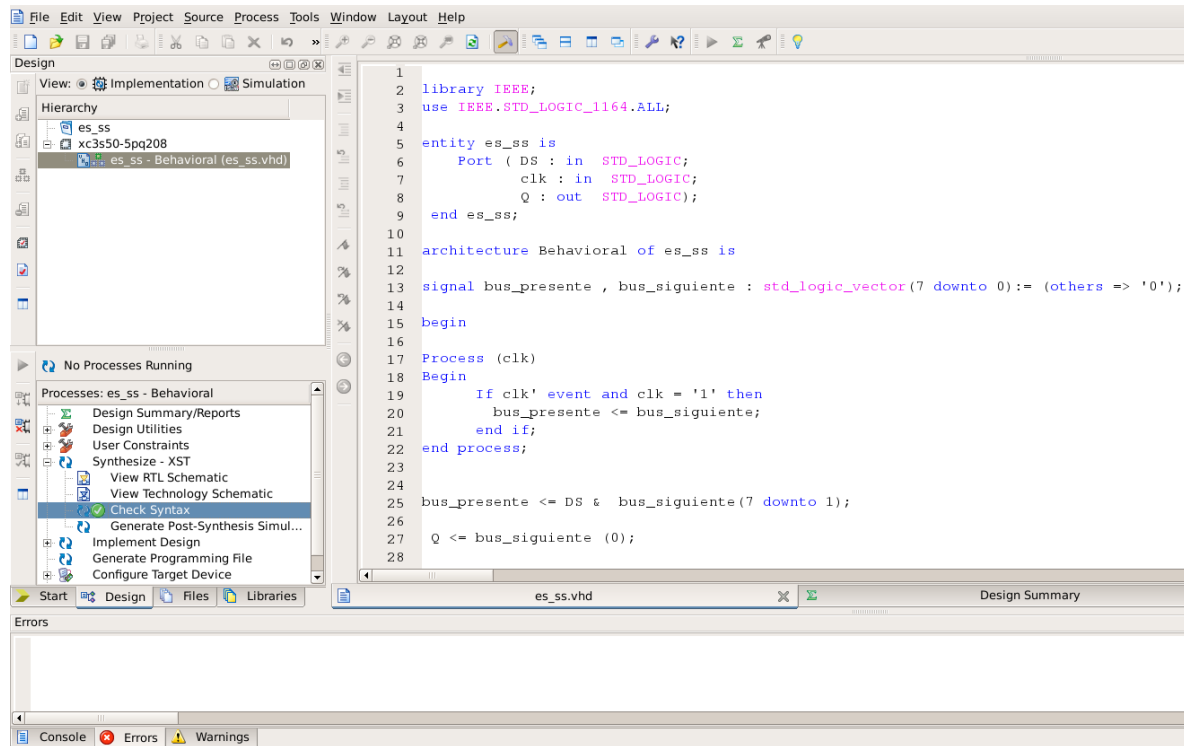


Figura 4.216 Ventana con el código del proyecto del registro de corrimiento.

Seleccionando la opción de View Technology Schematic muestra la siguiente ventana:

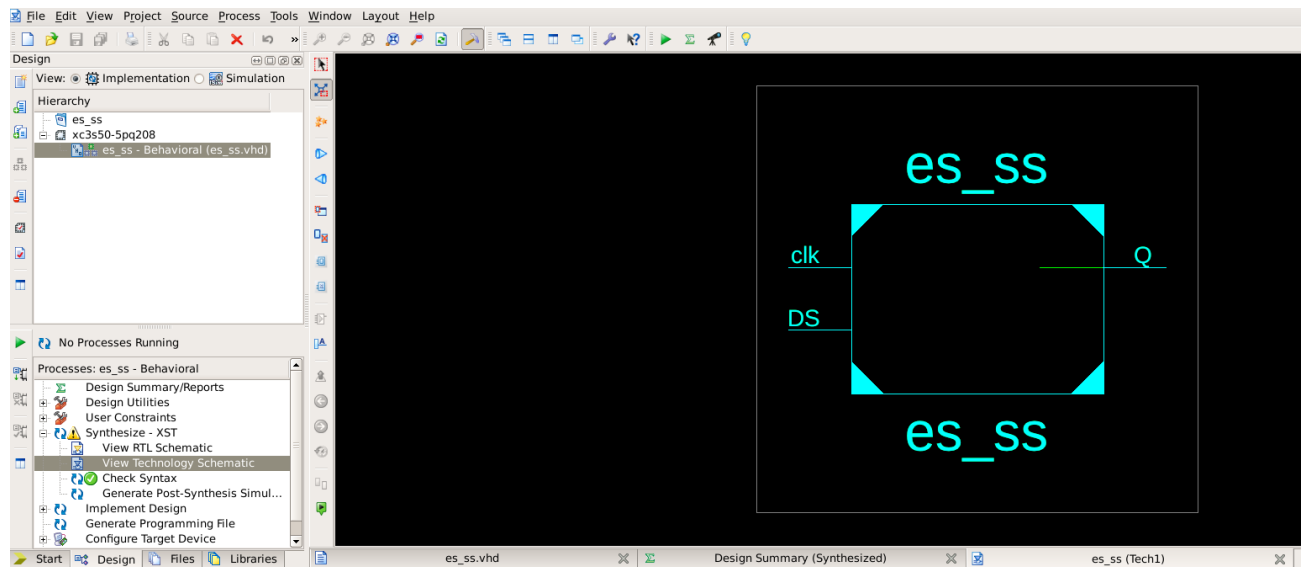


Figura 4.217 Ventana mostrando el esquemático del proyecto sintetizado.

Se selecciona la opción de simulación y en la ventana que aparece se hace la prueba con distintos valores de dando clic en la variable para cambiar sus valores.

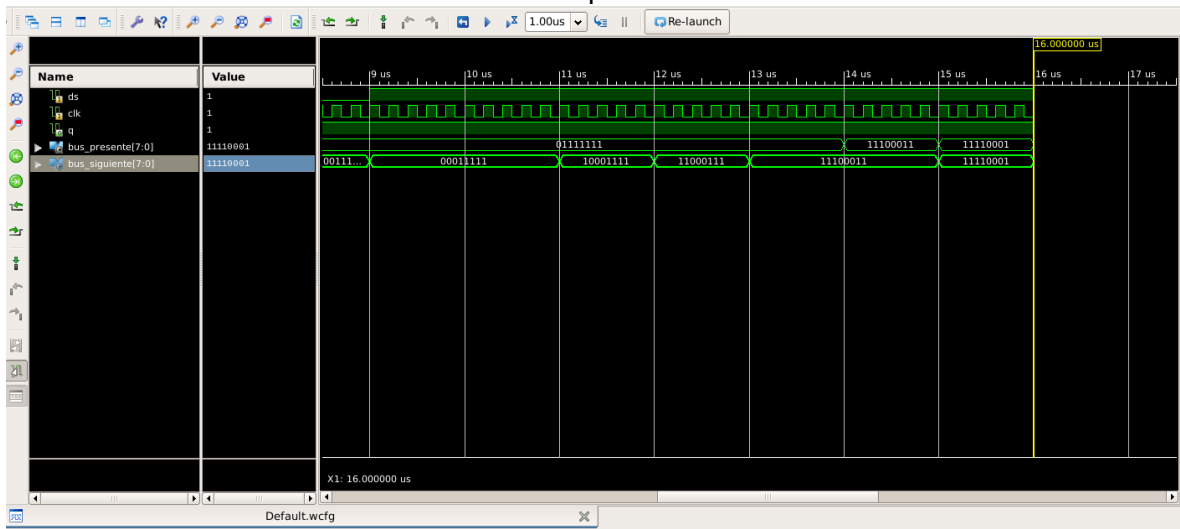


Figura 4.218 Simulación del registro entada serie salida serie.

Ejemplo 6

Desarrollar un programa en VHDL para el diagrama de estados(Máquina de Moore) de la siguiente figura:

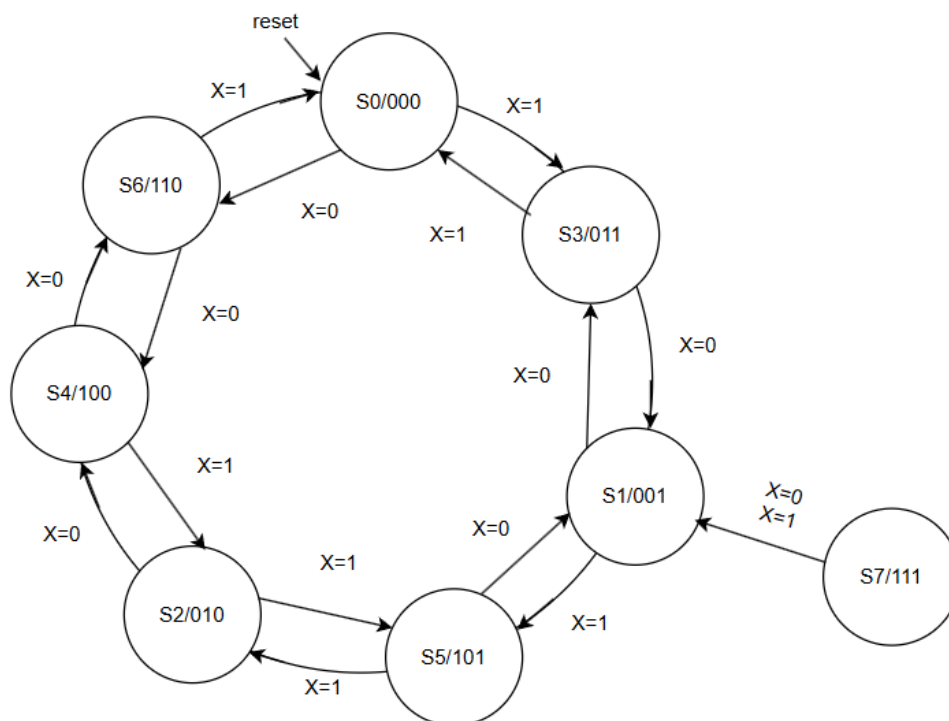


Figura 4.219 Diagrama de estados(Máquina de Moore) para implementar en VHDL.

Iniciando el nuevo proyecto se verá la ventana siguiente:

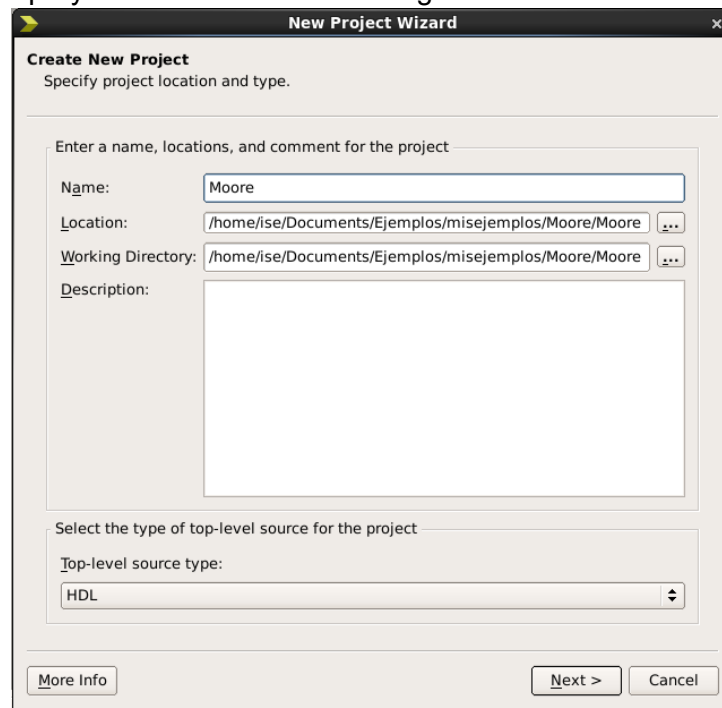


Figura 4.220 ventana de inicio del proyecto Moore.

Al darle Next, se selecciona una nueva fuente en la opción VHDL Module y se le da el nombre.

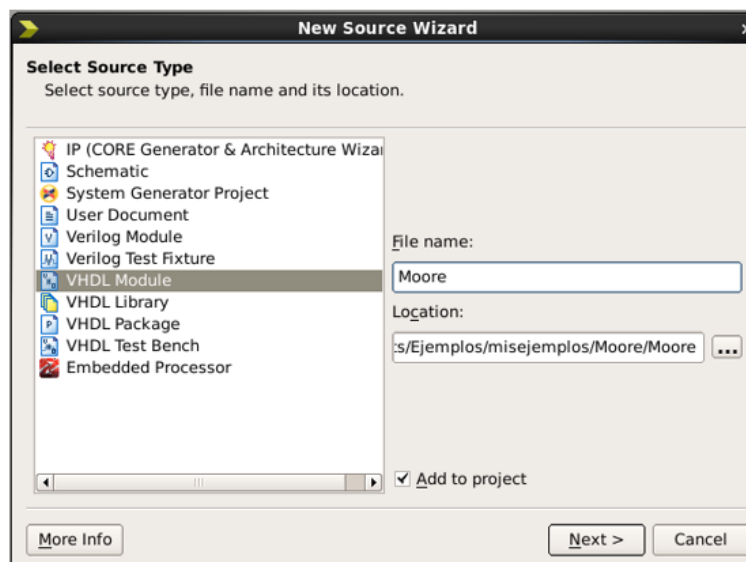


Figura 4.221 Ventana de selección del módulo VHDL.

Se le da siguiente para que muestre la ventana donde le daremos las entradas y salidas, así como su tipo, esto sería la entidad.

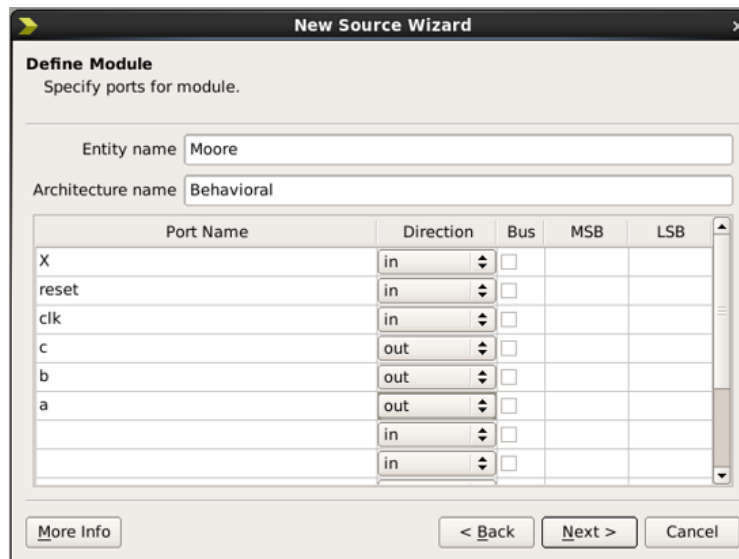


Figura 4.222 Ventana para asignar las entradas, salidas y tipo.

Al darle Next y Finish se muestra la ventana para editar el código, quitando los comentarios queda de la siguiente manera:

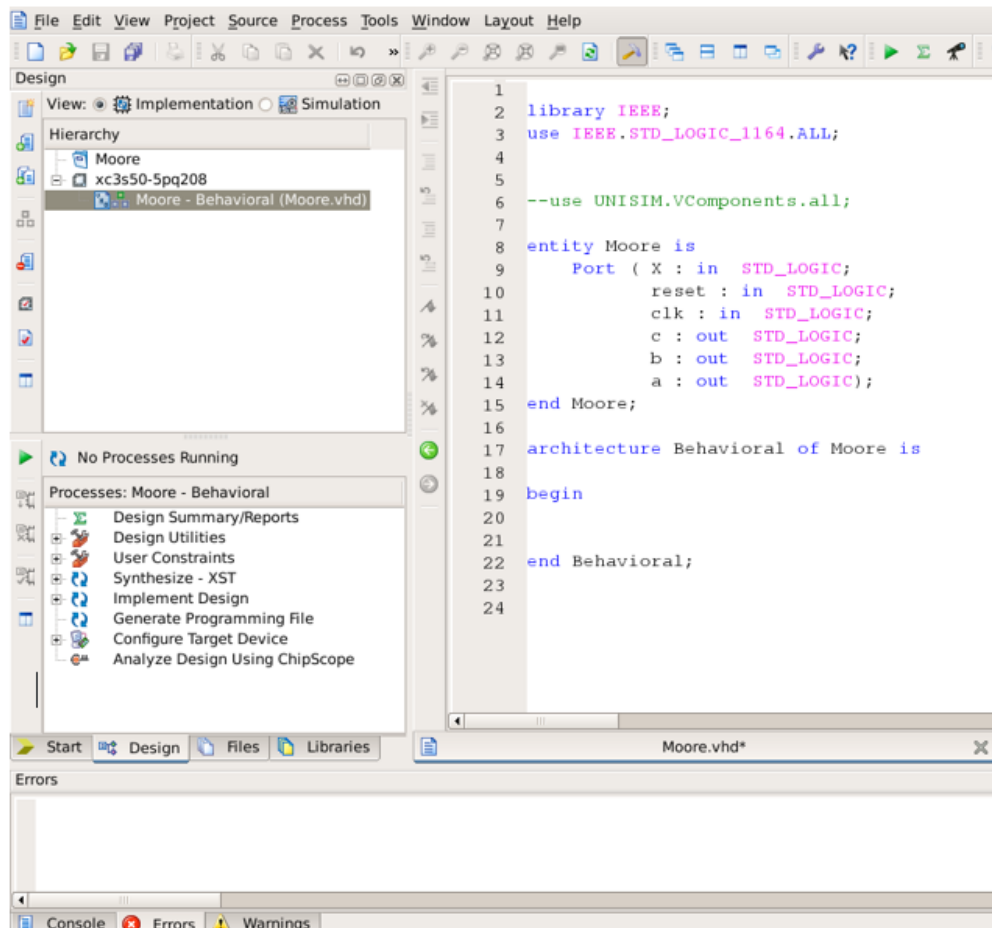


Figura 4.223 Ventana para editar el código.

Para desarrollar el proyecto de una máquina de estados se hace en tres pasos que son:

Registro de estado.

Lógica de estado siguiente.

Lógica de salida.

Antes se editan las señales que se usaran en el código estas son:

```
type estados is ( S0, S1, S2, S3, S4, S5, S6, S7);
```

```
signal p_bus, S_bus : estados;
```

```
signal salidas : std_logic_vector(2 downto 0);
```

Registro de estado.

Esta se hace con un process, para la sintaxis se puede auxiliar de la herramienta “Language Templates”, se selecciona VHDL, Coding Examples, Flop Flop, D Flip Flop, Posedge y w/Synchronous Active High Reset como se muestra en la figura:

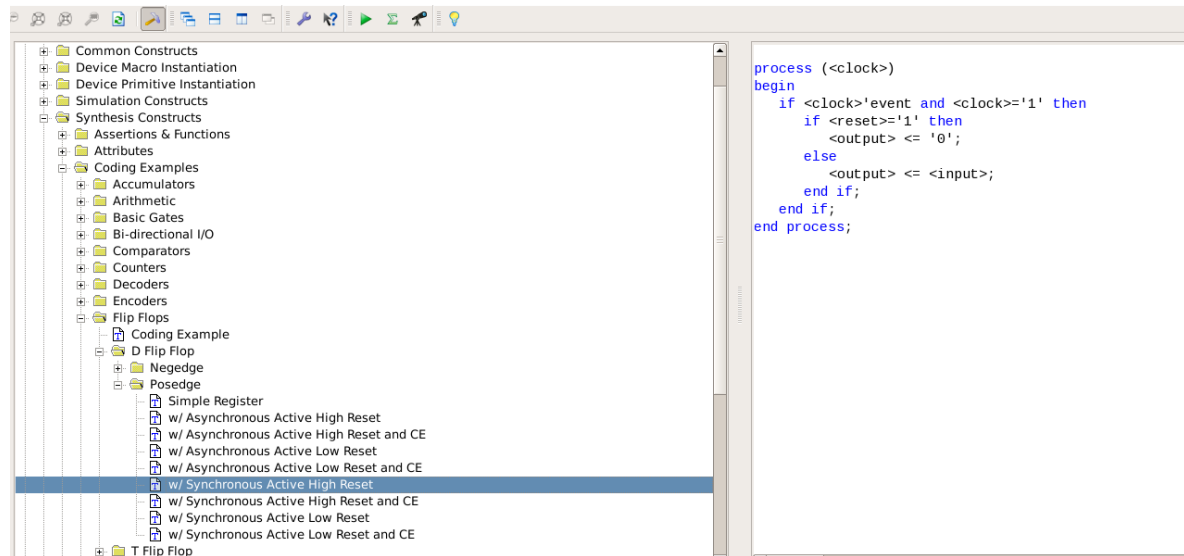


Figura 4.224 Sintaxis para un Flip Flop tipo D con flanco de subida y reset.

```
process (clk)
```

```
begin
```

```
    if clk'event and clk='1' then
```

```
        if reset='1' then
```

```
            s_bus <= S0;
```

```
        else
```



```

        s_bus <= p_bus;

    end if;

end if;

end process;

```

Lógica de estado siguiente.

Esta lógica se hace con un Process y dentro de este un case y dentro de este un When y un if else, para la sintaxis se puede auxiliar con la herramienta de ayuda como en el caso anterior para el registro de estado, la selección queda de la siguiente manera:

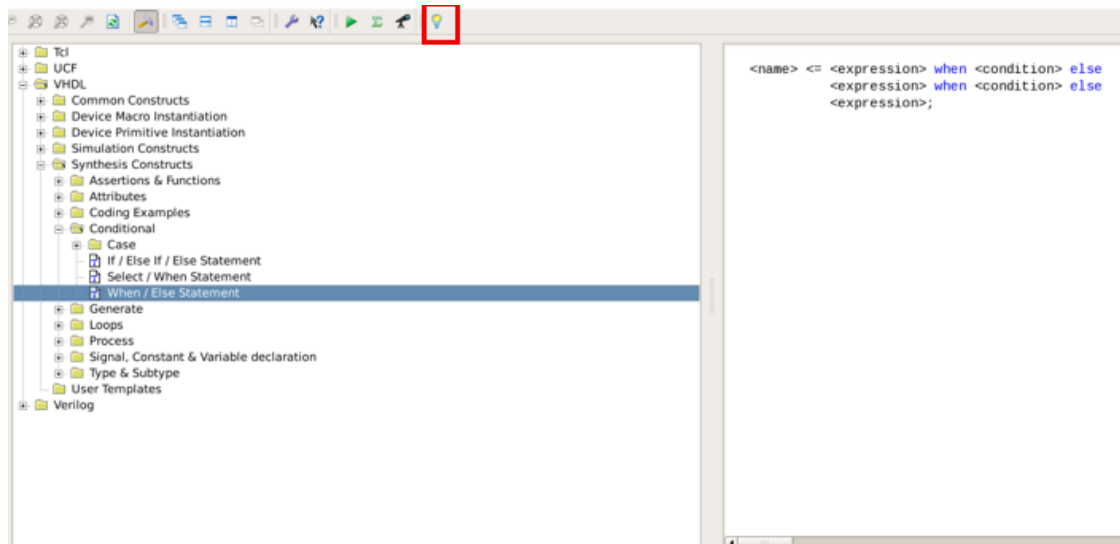


Figura 4.225 Ayuda para la sintaxis de la herramienta utilizada.

Código para la lógica del estado siguiente, s_bus es el estado siguiente y p_bus es el estado presente.

```

process (s_bus,X)
begin

```

```

    case (s_bus) is
        when S0 =>
            if X = '1' then
                p_bus <= S3;
            else
                p_bus <= S6;
            end if;

```

```

    when S3 =>
        if X = '1' then
            p_bus <= S0;
        else
            p_bus <= S1;

```

```

    end if;

when S6 =>
    if X = '1' then
        p_bus <= S0;
    else
        p_bus <= S4;
    end if;

when S4 =>
    if X = '1' then
        p_bus <= S2;
    else
        p_bus <= S6;
    end if;

when S2 =>
    if X = '1' then
        p_bus <= S5;
    else
        p_bus <= S4;
    end if;

when S5 =>
    if X = '1' then
        p_bus <= S2;
    else
        p_bus <= S1;
    end if;

when S1 =>
    if X = '1' then
        p_bus <= S5;
    else
        p_bus <= S3;
    end if;

    when S7 =>
        if X = '1' then
            p_bus <= S1;
        else
            p_bus <= S1;
        end if;

end case;

end process;

```

Lógica de salida.

Para la lógica de salida se usará el with select para asignarle a cada estado la salida que le corresponde en el diagrama de estados, en este caso corresponde a la combinación binaria del mismo estado, pero puede ser diferente, en seguida se asigna a las variables de salida el bit que le corresponde, a c el bit2, a b el bit 1 y a a el bit 0.

```
with s_bus select
  salidas <= "000" when S0,
            "001" when S1,
            "010" when S2,
            "011" when S3,
            "100" when S4,
            "101" when S5,
            "110" when S6,
            "111" when S7,
            "000" when others;
```

```
c <= salidas(2);
b <= salidas(1);
a <= salidas(0);
```

En seguida se enlista el código completo:

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4
5 --use UNISIM.VComponents.all;
6
7 entity Moore is
8     Port ( X : in  STD_LOGIC;
9           reset : in  STD_LOGIC;
10          clk : in  STD_LOGIC;
11          c : out  STD_LOGIC;
12          b : out  STD_LOGIC;
13          a : out  STD_LOGIC);
14 end Moore;
15
16 architecture Behavioral of Moore is
17
18     type estados is ( S0, S1, S2, S3, S4, S5, S6, S7);
19     signal p_bus, s_bus : estados;
20     signal salidas : std_logic_vector(2 downto 0);
21
22 begin
23     -----Registro de estado-----
24     process (clk)
25     begin
26         if clk'event and clk='1' then
27             if reset='1' then
28                 s_bus <= S0;
29             else
30                 s_bus <= p_bus;
31             end if;
32         end if;
33     end process;
34     -----Lógica del estado siguiente-----
35     process (s_bus,X)
36     begin
37
38         case (s_bus) is
39             when S0 =>
40                 if X = '1' then
41                     p_bus <= S3;
42                 else
43                     p_bus <= S6;
44                 end if;
45
46             when S3 =>
47                 if X = '1' then
48                     p_bus <= S0;
49                 else
50                     p_bus <= S1;
51                 end if;
52
53             when S6 =>
54                 if X = '1' then
55                     p_bus <= S0;
56                 else
57                     p_bus <= S4;
58                 end if;
59
60             when S4 =>
61                 if X = '1' then
62                     p_bus <= S2;
63                 else
64                     p_bus <= S6;
65                 end if;
66
67             when S2 =>
68                 if X = '1' then
69                     p_bus <= S5;
70                 else
71                     p_bus <= S1;
72                 end if;
73
74             when S5 =>
75                 if X = '1' then
76                     p_bus <= S2;
77                 else
78                     p_bus <= S1;
79                 end if;
80
81             when S1 =>
82                 if X = '1' then
83                     p_bus <= S5;
84                 else
```

```

85     p_bus <= s3;
86   end if;
87
88   when s7 =>
89     if X = '1' then
90       p_bus <= s1;
91     else
92       p_bus <= s1;
93     end if;
94   end case;
95
96
97
98
99 end process;
100 -----Lógica de salidas-----
101 with s_bus select
102   salidas <= "000" when s0,
103              "001" when s1,
104              "010" when s2,
105              "011" when s3,
106              "100" when s4,
107              "101" when s5,
108              "110" when s6,
109              "111" when s7,
110              "000" when others;
111
112 c <= salidas(2);
113 b <= salidas(1);
114 a <= salidas(0);
115
116
117 end Behavioral;

```

La siguiente figura es el código ya checado en su sintaxis:

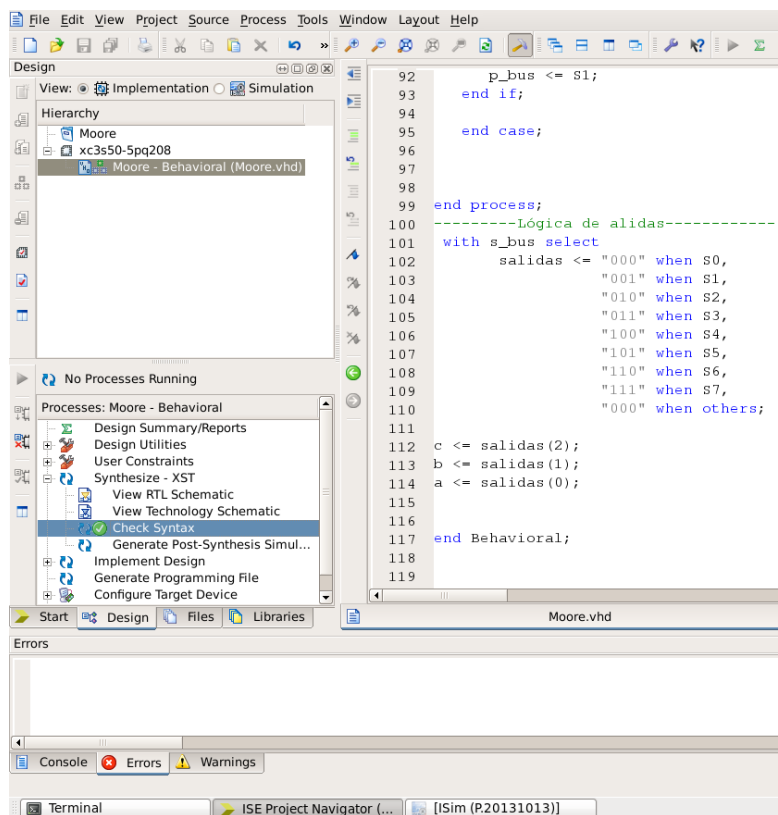


Figura 4.226 Código terminado con la sintaxis correcta.

Como no hay errores se puede ver el RTL y se muestra de la siguiente manera:

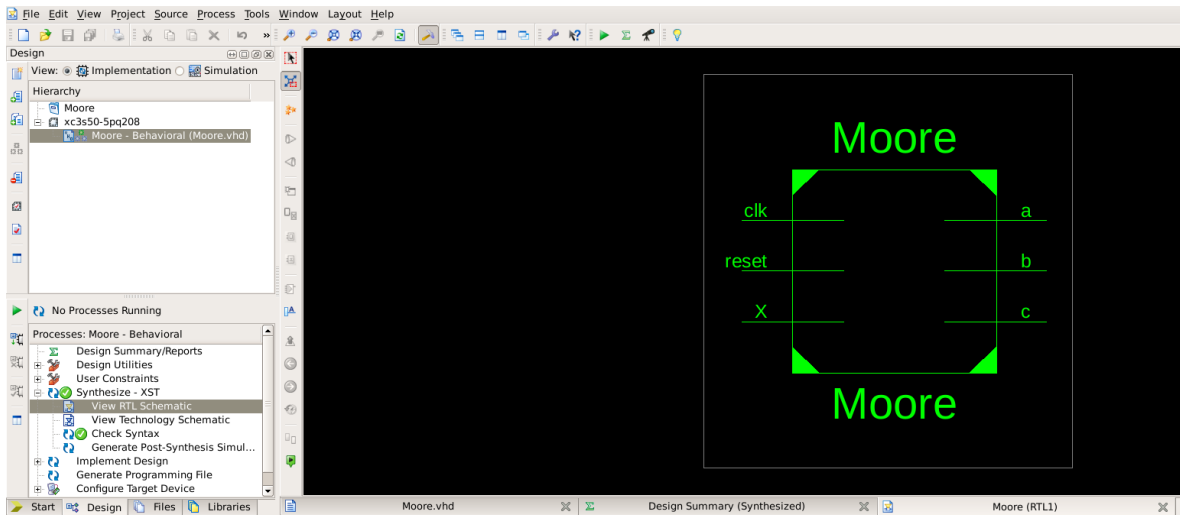


Figura 4.227 Esquemático del proyecto máquina de estados Moore.

Se puede ir dando doble clic en la ventana del esquemático para que muestre lo que hay en la caja hasta llegar al circuito final.

Para la simulación se cambia a la pestaña Simulation, se corre checar Behavioral Check Syntax y si no hay errores se procede a la simulación, recuerde que puede cambiar los valores de las variables dando un clic en la que se desea modificar.

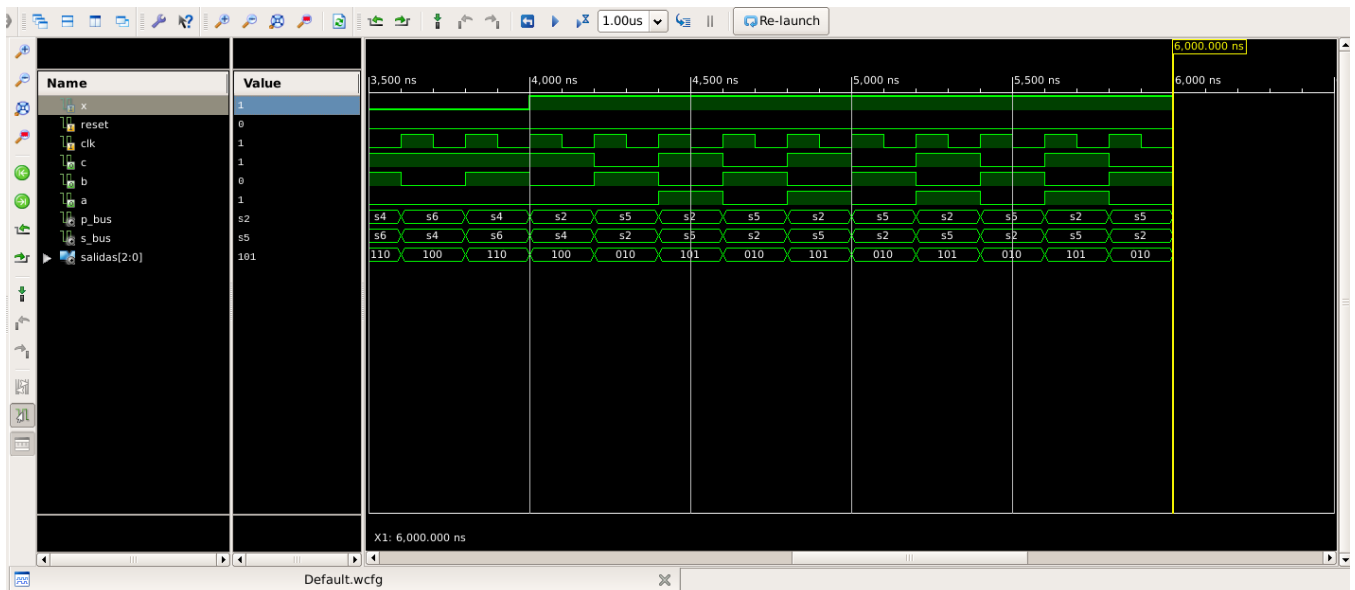


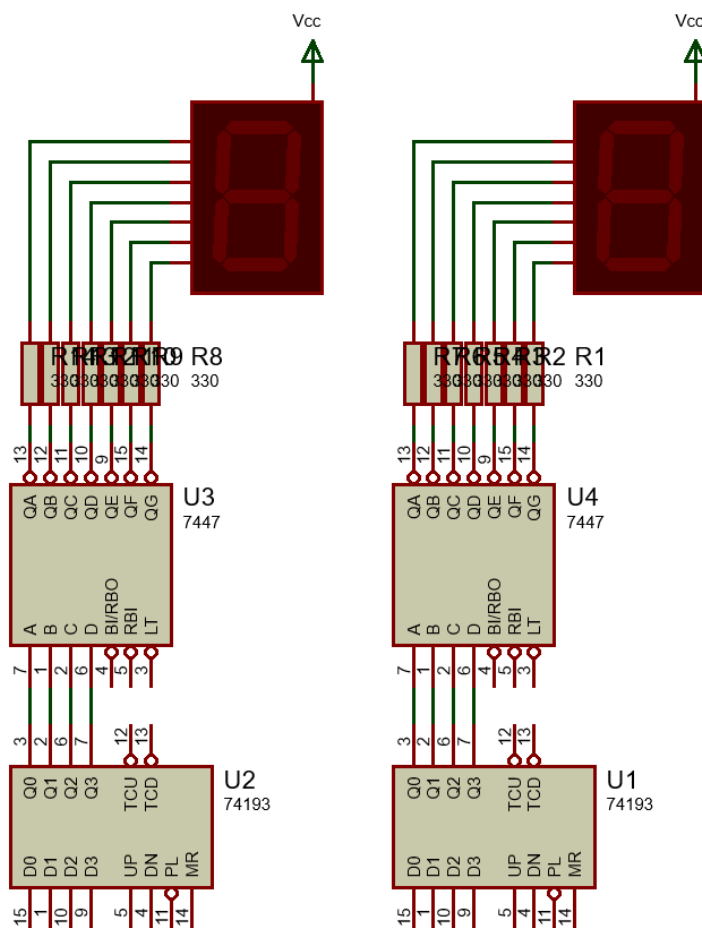
Figura 4.228 Simulación del proyecto funcionando correctamente.

Contadores de dos dígitos.

Diseñe un contador de dos dígitos para hacer un conteo desde cero hasta un valor mayor de 20 y menor de 95, utilice los contadores 71193, los decodificadores y los displays de 7 segmentos, haga las conexiones necesarias para su diseño.

Objetivo

Comprender el funcionamiento de los contadores con módulo diferente de 2^n .



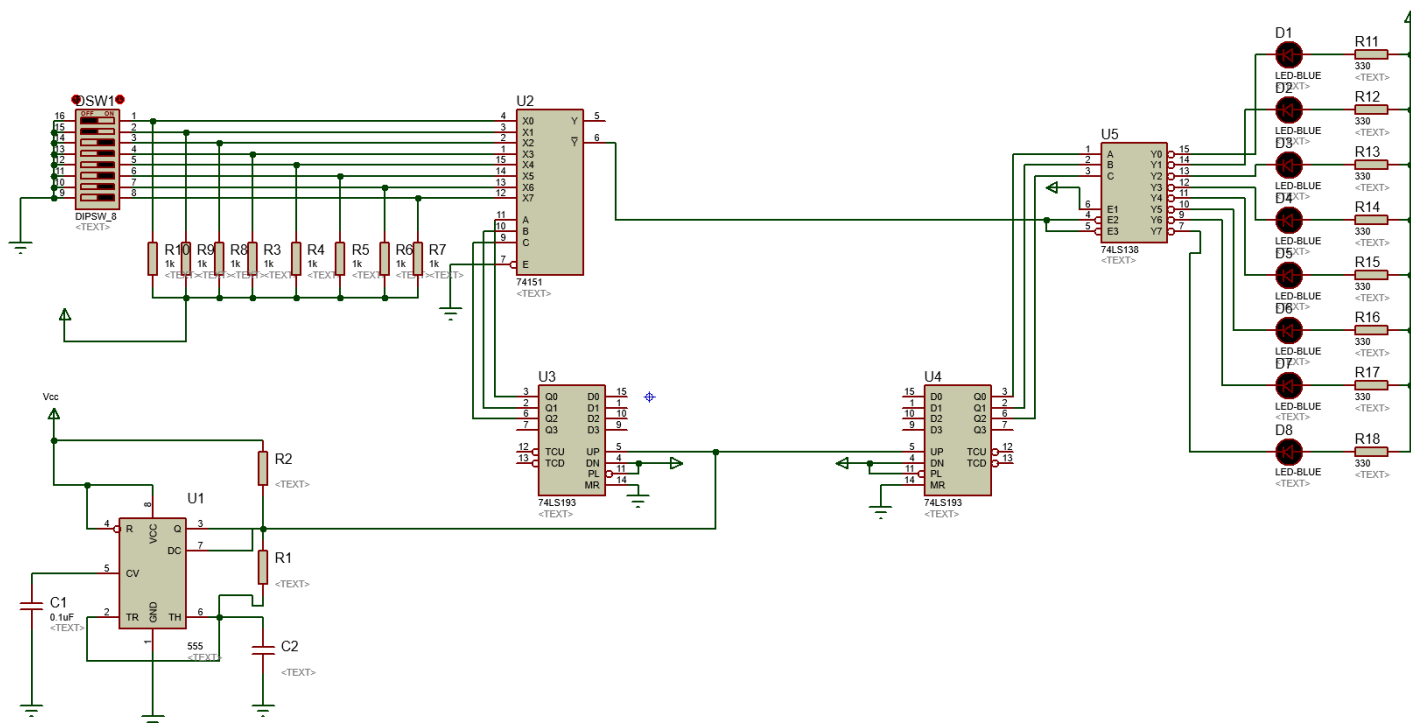
Práctica

Transmisor de estados por dos hilos.

Implemente el siguiente circuito y estudie su comportamiento, calcule las resistencias del 555 para tener una frecuencia de 1 KHz o mayor, pruebe primero el circuito con una frecuencia de 1Hz para ver si se muestra el estado de los interruptores en los leds.

Objetivo

Comprender el funcionamiento de los multiplexores y demultiplexores.



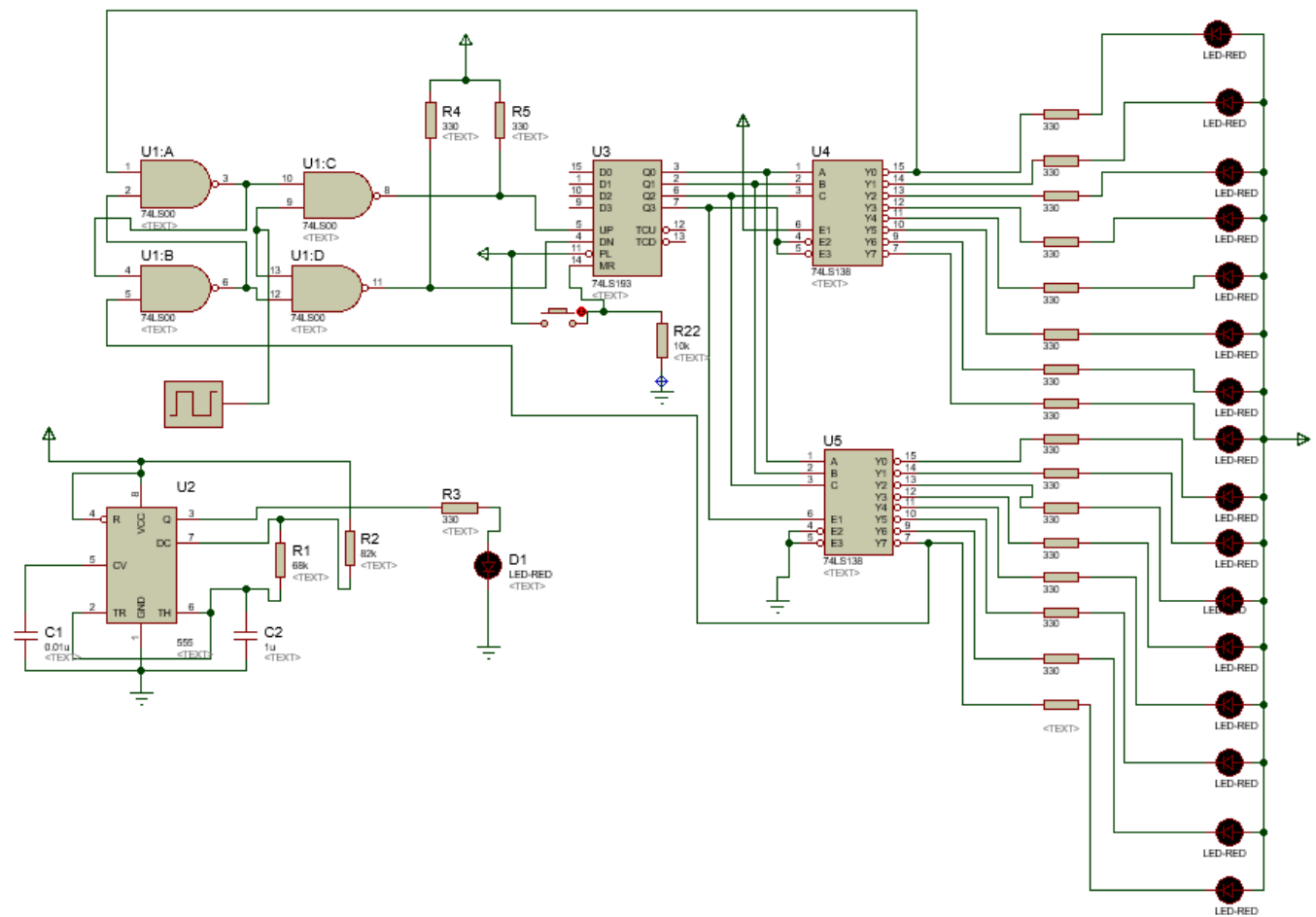
Práctica

Luces secuenciales con decodificadores.

Implemente el siguiente circuito de luces secuenciales y analice el comportamiento de cada circuito que lo compone, explique el funcionamiento de ellos, en el 555 utilice un potenciómetro para cambiar la frecuencia de barrido de los leds.

Objetivo

Comprender el funcionamiento de los decodificadores.



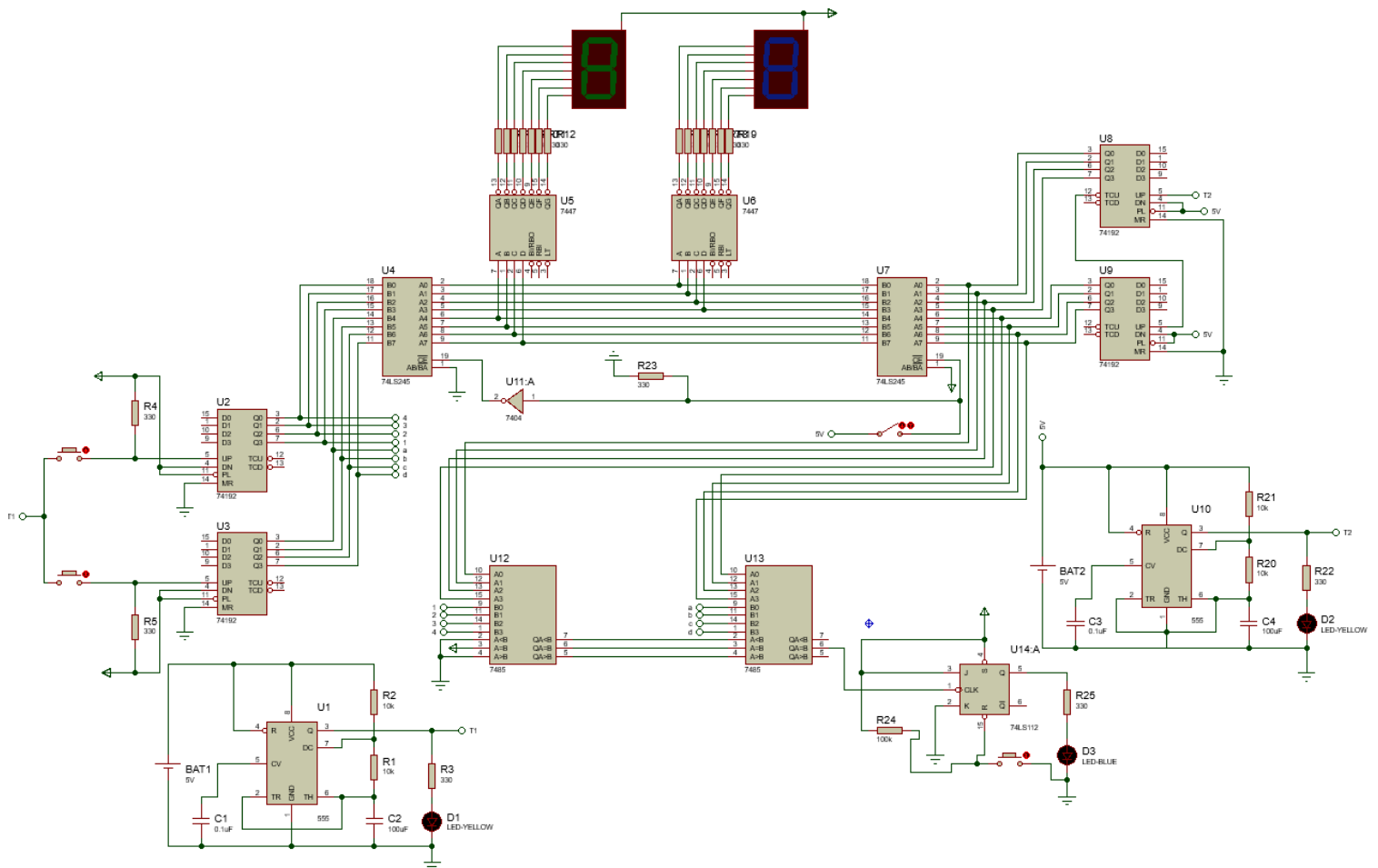
Práctica

Alarma de dos dígitos.

Implemente el siguiente circuito “Alarma de dos dígitos”, arme el circuito por partes y cheque su funcionamiento, primero los contadores que llevaran el conteo normal(555 con frecuencia de un segundo, los dos contadores, el buffer bidireccional, los decodificadores BCD a 7 segmentos y los displays) en seguida los contadores para poner el valor a comparar con el buffer que le corresponde, el 555 que incrementaran estos contadores y por último los comparadores de magnitud y el flip flop para la alarma. Analice el funcionamiento de cada circuito integrado que conforman la alarma.

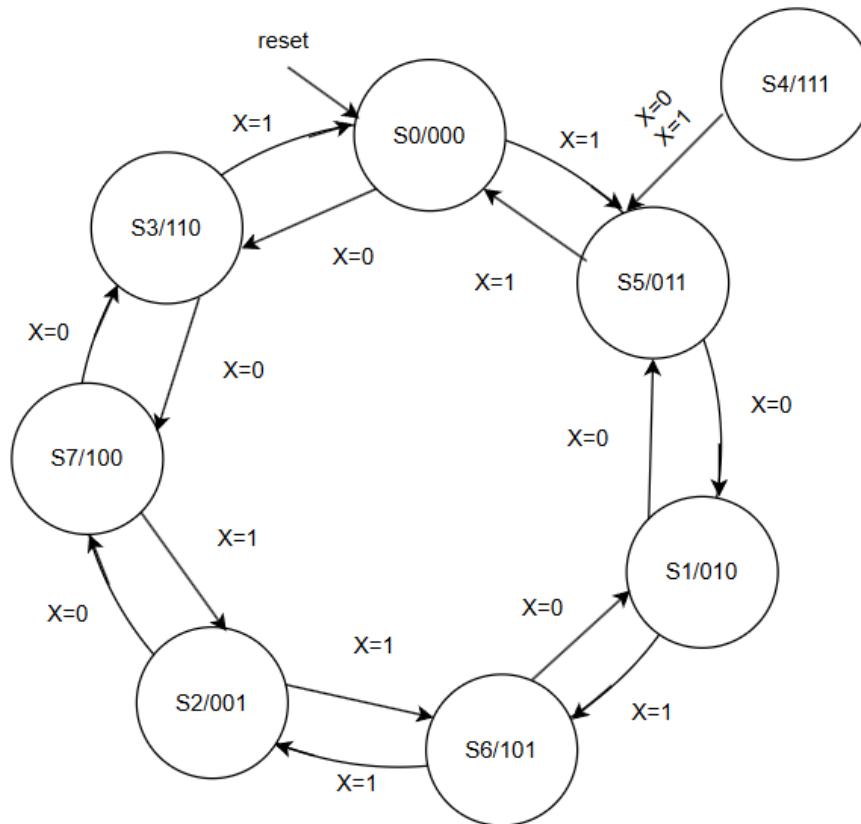
Objetivo

Comprender el funcionamiento de los comparadores de magnitud y los circuitos de tercer estado bidireccionales.



Práctica

Diseñe en lenguaje VHDL y haga la simulación de la siguiente máquina de Moore.



Práctica

Diseñe en VHDL un registro de corrimiento de 8 bits entrada serie salida paralelo y haga su simulación.

Examen

Selecciona la opción correcta para cada pregunta. Solo hay una respuesta válida por pregunta.

1. ¿Cuál es la principal diferencia entre una máquina de Moore y una de Mealy?

- a) La máquina de Moore depende solo del estado actual, mientras que la de Mealy depende del estado actual y las entradas.
- b) La máquina de Mealy depende solo del estado actual, mientras que la de Moore depende del estado actual y las entradas.
- c) La máquina de Moore no tiene salidas, mientras que la de Mealy sí.
- d) La máquina de Mealy no tiene estados definidos.

2. ¿Qué componente se utiliza para generar señales temporizadas en circuitos electrónicos?

- a) Flip-flop.
- b) Temporizador 555.
- c) Contador binario.
- d) Registro de desplazamiento.

3. ¿Qué tipo de flip-flop cambia su salida únicamente en el flanco de subida del reloj?

- a) Asíncrono.
- b) Síncrono.
- c) Latch.
- d) Multivibrador.

4. ¿Cuál es la condición inválida en un flip-flop SR?

- a) $S=0, R=0$.
- b) $S=1, R=0$.
- c) $S=0, R=1$.
- d) $S=1, R=1$.

5. ¿Qué flip-flop se considera universal debido a su versatilidad?

- a) D.
- b) T.
- c) JK.
- d) SR.

6. ¿Qué representa un diagrama de estados en un circuito secuencial?

- a) La relación entre entradas y salidas.
- b) Las transiciones entre diferentes estados del sistema.
- c) La configuración física del circuito.
- d) La tabla de verdad del circuito.

7. ¿Qué caracteriza a un circuito secuencial síncrono?

- a) Opera sin una señal de reloj.
- b) Sus cambios de estado dependen de una señal de reloj común.
- c) Sus salidas dependen solo de las entradas actuales.
- d) No utiliza flip-flops en su diseño.

8. ¿Qué componente se utiliza para almacenar múltiples bits de datos y permite desplazarlos?

- a) Contador.
- b) Registro de desplazamiento.
- c) Temporizador.
- d) Multiplexor.

9. ¿Cuál es la función principal de un contador en electrónica digital?

- a) Multiplexar señales.
- b) Almacenar datos de forma permanente.
- c) Contar eventos o pulsos.
- d) Generar señales analógicas.

10. ¿Qué tipo de memoria permite la lectura y escritura de datos?

- a) ROM.
- b) RAM.
- c) PROM.
- d) EPROM.

11. ¿Qué lenguaje se utiliza comúnmente para describir hardware digital?

- a) Python.
- b) VHDL.
- c) Java.
- d) C++.

12. ¿Qué significa VHDL?

- a) Very High Digital Logic.
- b) Virtual Hardware Description Language.
- c) VHSIC Hardware Description Language.
- d) Verified Hardware Design Language.

13. ¿Qué tipo de flip-flop tiene una única entrada además del reloj?

- a) SR.
- b) JK.
- c) D.
- d) T.

14. ¿Qué componente se utiliza para detectar secuencias específicas de bits en una entrada?

- a) Contador.
- b) Detector de secuencia.
- c) Registro.

d) Multiplexor.

15. ¿Cuál es la principal ventaja de un circuito secuencial síncrono sobre uno asíncrono?

- a) Mayor velocidad de operación.
- b) Menor consumo de energía.
- c) Mayor predictibilidad y estabilidad en las transiciones de estado.
- d) Menor complejidad en el diseño.

16. ¿Qué tipo de flip-flop cambia su estado con cada pulso de reloj?

- a) D.
- b) T.
- c) SR.
- d) JK.

17. ¿Qué componente se utiliza para generar una señal de reloj en un circuito?

- a) Flip-flop.
- b) Temporizador 555.
- c) Contador.
- d) Registro.

18. ¿Qué tipo de memoria es no volátil y puede ser programada una sola vez?

- a) RAM.
- b) ROM.
- c) PROM.
- d) EPROM.

19. ¿Qué componente se utiliza para almacenar datos de forma temporal durante la ejecución de un programa?

- a) ROM.
- b) RAM.

- c) EPROM.
- d) PROM.

20. ¿Qué herramienta se utiliza para representar gráficamente las transiciones de estado en un sistema secuencial?

- a) Tabla de verdad.
- b) Diagrama de flujo.
- c) Diagrama de estados.
- d) Esquema eléctrico.

21. ¿Qué tipo de modelado en VHDL se utiliza para describir un circuito conectando componentes existentes?

- a) Conductual
- b) Estructural
- c) Flujo de datos
- d) Secuencial

22. ¿Cuál de las siguientes opciones es una sentencia concurrente en VHDL?

- a) process
- b) if
- c) case
- d) with-select

23. En VHDL, ¿cuál es el propósito de la sentencia generate?

- a) Declarar señales internas
- b) Crear múltiples instancias de componentes de manera parametrizada
- c) Asignar valores a constantes
- d) Iniciar un proceso secuencial

24. ¿Qué tipo de señal se utiliza comúnmente para describir circuitos síncronos en VHDL?

- a) bit
- b) std_logic
- c) integer
- d) boolean

25. ¿Cuál es la diferencia principal entre una señal (signal) y una variable (variable) en VHDL?

- a) Las señales se actualizan inmediatamente; las variables al final del proceso.
- b) Las variables se actualizan inmediatamente; las señales al final del proceso.
- c) Las señales solo se usan en procesos; las variables fuera de ellos.
- d) No hay diferencia; ambas se comportan igual.

26. ¿Qué sentencia se utiliza para seleccionar entre múltiples opciones en VHDL?

- a) if
- b) case
- c) select
- d) choose

27. ¿Cuál de las siguientes opciones representa una asignación condicional en VHDL?

- a) `Y <= A when S='1' else B;`
- b) `Y = A if S='1' else B;`
- c) `if S='1' then Y <= A; else Y <= B; end if;`
- d) `Y := A when S='1' else B;`

28. ¿Qué instrucción se utiliza para crear bucles en VHDL?

- a) loop
- b) for
- c) while
- d) Todas las anteriores

29. ¿Qué palabra clave se utiliza para incluir bibliotecas estándar en VHDL?

- a) include
- b) library
- c) use
- d) import

30. ¿Cuál de las siguientes opciones es un tipo de modelado en VHDL que describe cómo los datos fluyen a través del sistema?

- a) Conductual
- b) Estructural
- c) Flujo de datos
- d) Secuencial

Unidad 5

Convertidores.

Los convertidores analógicos-digital (ADC) y los convertidores digitales-analógico (DAC) son fundamentales en los sistemas de control digital en los que se manejan señales físicas de origen analógico, estas señales deben ser adecuadas para que pueda ser interpretadas y procesada por el sistema utilizando, estos sistemas digitales pueden ser: un microprocesador, microcontrolador, sistema embebido o una computadora.

En el procesamiento de las señales físicas como temperatura, presión o humedad entre otras, se siguen diferentes etapas, como se ilustra en la siguiente figura:

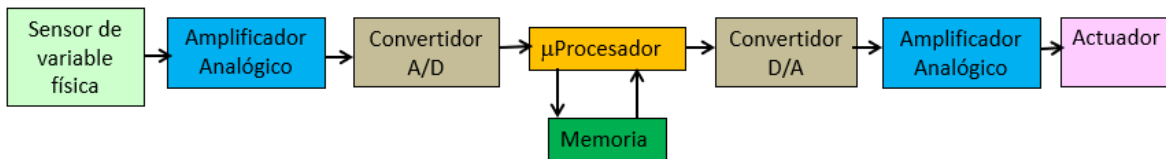


Figura 5.1 Sistema de control de temperatura.

La primera etapa consiste en adecuar la señal entregada por el sensor utilizado. Normalmente los sensores de señales físicas generan una señal de baja amplitud (corriente o voltaje), por lo que es necesario amplificarla para que cumpla las características que requiere el ADC, una vez digitalizada puede ser procesada por el sistema digital, este regresa una palabra binaria que es convertida nuevamente a una señal analógica por medio del DAC para regresarla a una señal analógica que puede ser amplificada si se requiere para accionar el actuador.

Por lo anterior, es fundamental comprender el funcionamiento y la correcta utilización de los convertidores ADC y DAC.

5.1 Funcionamiento de DAC.

La tarea de un “convertidor digital-analógico (DAC)” es transformar una señal digital presentada en forma de unos y ceros a una señal analógica continua en el tiempo ya sea un voltaje o una corriente, que debe ser proporcional a la señal digital.

La figura 5.2 muestra el símbolo para un convertidor digital-analógico de cuatro bits.

Se tiene cuatro bits de entrada, con ellos se pueden hacer 2^4 combinaciones diferentes posibles, además se tiene un voltaje de referencia. Por cada combinación binaria de entrada se tendrá un voltaje único de salida equivalente a:

$$V_{sal} = V_{ref} \times (D / (2^n - 1))$$

D es el valor binario de la palabra digital de entrada que puede tomar un valor de 0 hasta $(2^n - 1)$

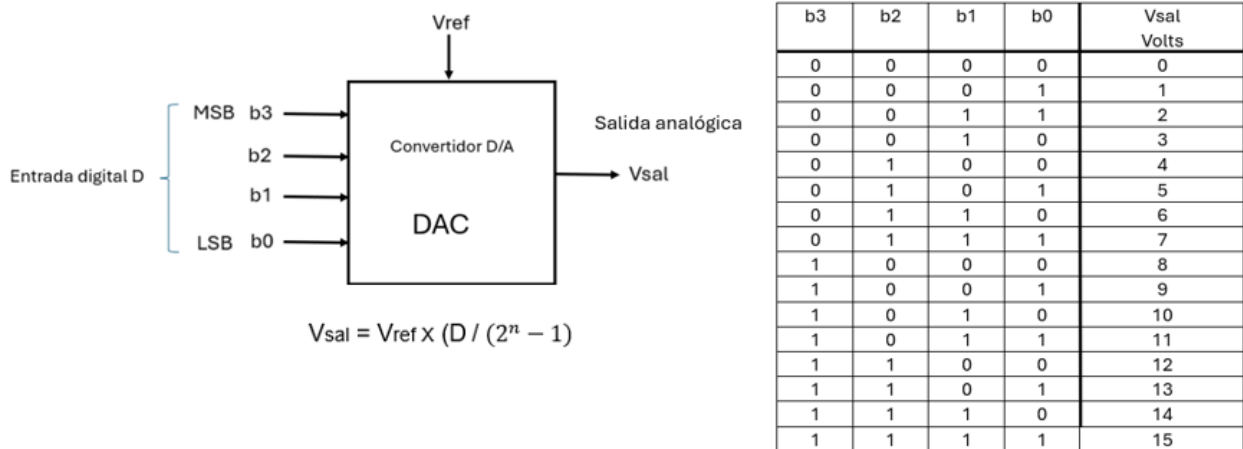


Figura 5.2 Convertidor digital-analógico de 4 bits y salidas para un Vref de 15 Volts.

Resolución de un DAC es el voltaje mínimo que puede representar por paso, se obtiene dividiendo el Vsal máximo (normalmente Vref) entre $(2^n - 1)$, donde n es el número de bits, por ejemplo, si se tiene un convertidor de 8 bits y el voltaje de salida máximo es de 5 Volts, la resolución es de $5 \text{ volts} / 255 = 19.6 \text{ milivolts}$, lo que significa que, por cada incremento en la palabra digital de entrada, la salida se incrementará en 19,6 milivolts, si la palabra digital de entrada es 0111111_2 es decir 127_{10} el voltaje de salida será $0.0196 \times 127 = 2.489 \text{ volts}$.

¿cuál es la resolución de un DAC que maneja 10 bits de entrada y su voltaje máximo de salida es de 2.5 volts?

Con 10 bits se pueden tener 1024 combinaciones diferentes lo tanto, la resolución es:

Resolución = $2.5 \text{ volts} / 1023 = 2.443 \text{ milivolts}$.

Se puede visualizar la salida de un DAC como una escalera de $2^n - 1$ escalones, donde cada escalón corresponde a una combinación binaria de entrada.

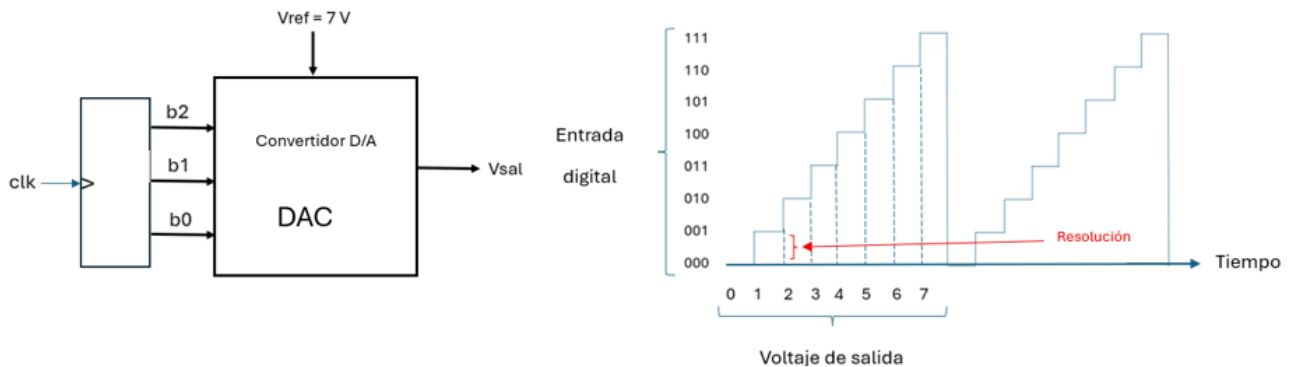


Figura 5.3 Salida del DAC de 3 bits con 7 volts de referencia y una resolución de 1volt.

Resolución porcentual.

Puede ser útil presentar la resolución porcentual que es igual a la resolución del convertidor entre la escala completa y se expresa como un porcentaje, para el ejemplo del DAC de 3 bits, quedaría de la siguiente manera:

Porcentaje de resolución = $1v / 7v = 14.28 \%$; 1v es la resolución y 7v es el valor de salida a escala completa es decir con una entrada de 111.

Si se tiene un DAC de 10 bits y su voltaje máximo de salida es de 5v, la resolución es de :

$$\begin{aligned} \text{Resolución} &= 5v / (2^n - 1) \\ &= 5v / 1023 \\ &= 4.88 \text{ mV} \end{aligned}$$

El porcentaje de resolución es:

$$\begin{aligned} \text{Porcentaje de resolución} &= (4.88 \text{ mv} / 5) \times 100 \\ &= 0.097 \% \end{aligned}$$

Métodos para convertir una señal digital a señal analógica:

Método de resistencias ponderadas.

Es una técnica común para convertir una señal digital en una señal analógica utilizando una red de resistencias con valores ponderados según la posición del bit en el número binario, los valores de las resistencias son: $R, 2R, 4R, \dots$

Este es un convertidor sencillo, lo difícil es tener los valores de las resistencias adecuadas.

Cada bit controla un interruptor que conecta la resistencia a un voltaje de referencia V_{ref} si el bit es 1 o a tierra si es 0.

El amplificador operacional trabaja como un sumador inversor, combina las corrientes generadas y las convierte en un voltaje de salida.

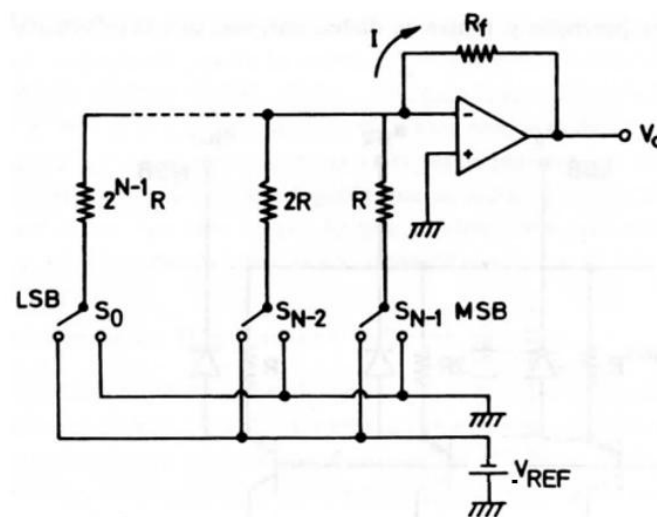


Figura 5.4 Convertidor digital-analógico de resistencias ponderadas.

$$V_o = \frac{V_{REF} R_f}{R} \left(a_{N-1} + a_{N-2} \frac{1}{2} + \dots + a_0 \frac{1}{2^{N-1}} \right)$$

Como se puede ver en la ecuación el bit más significativo genera V_{ref} el siguiente bit $V_{ref}/2$, el tercer bit $V_{ref}/4$ y así sucesivamente hasta $V_{ref} / 2^n$.

La resolución la define el bit menos significativo, es decir si el DAC es de 4 bits la resolución es $V_{ref} / 8$.

Ejemplo

Si la relación de R_f/R es igual a 1 y $V_{ref} = -5\text{ v}$, hacer la tabla de valores de salida para todas las combinaciones de la palabra digital de 4 bits.

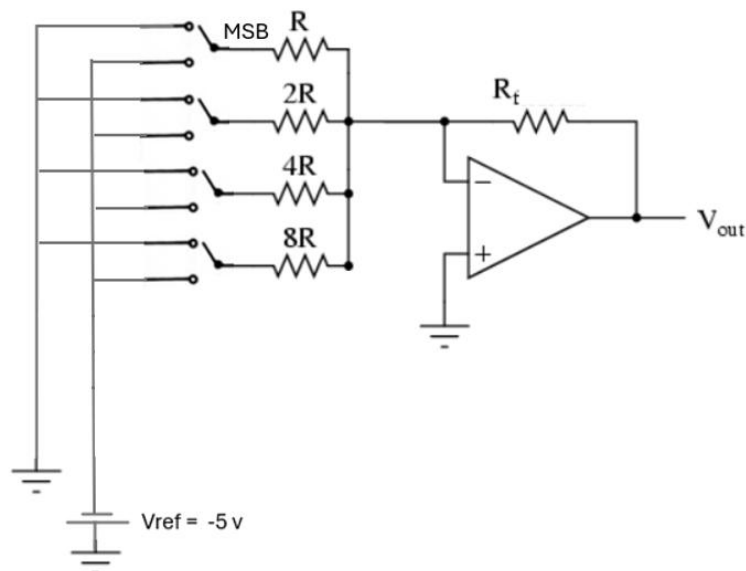


Figura 5.5 Convertidor digital-analógico de resistencias ponderadas de 4 bits.

b3	b2	b1	b0	Vout (volts)
0	0	0	0	0
0	0	0	1	0.625
0	0	1	0	1.250
0	0	1	1	1.875
0	1	0	0	2.500
0	1	0	1	3.125
0	1	1	0	3.750
0	1	1	1	4.375
1	0	0	0	5.00
1	0	0	1	5.625
1	0	1	0	6.250
1	0	1	1	6.875
1	1	0	0	7.500
1	1	0	1	8.125
1	1	1	0	8.750
1	1	1	1	9.375

$$V_{out} = 5v(b3 + b2/2 + b1/4 + b0/8)$$

Figura 5.6 Respuesta del convertidor de 4 bits.

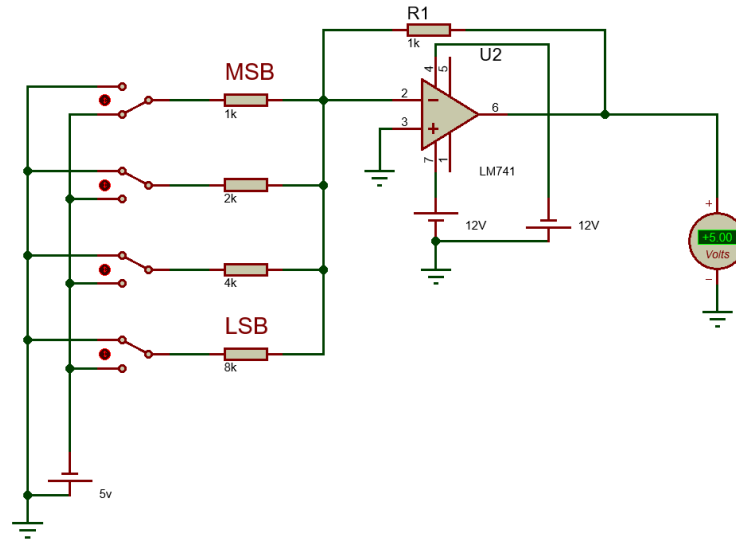


Figura 5.7 Simulación del DAC en Proteus.

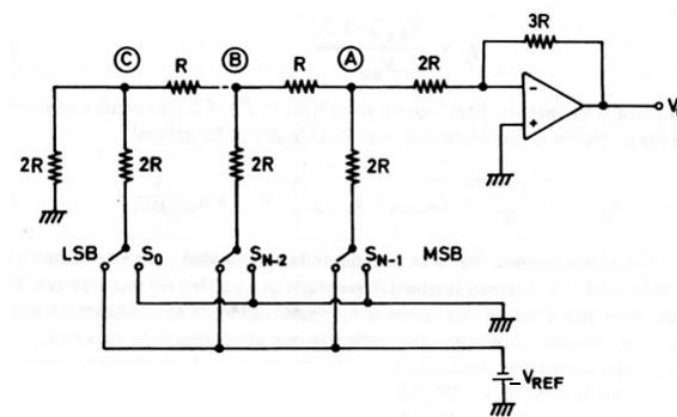
La precisión en la conversión.

Esta depende mucho de los valores exactos de las resistencias utilizadas y de la exactitud de los voltajes de entrada.

Convertidor digital-analógico red escalera R / 2R.

El problema que se tiene con los DAC's de resistencias ponderadas, son los valores de resistencia muy elevados cuando maneja una gran cantidad de bits, por ejemplo, si se tienen 12 bits y la resistencia para el bit más significativo es de 1KΩ, el valor de la resistencia para el bit menos significativo, el número 12, sería mayor de 2MΩ.

Para evitar ese problema, se usa la técnica, red escalera R/2R, que sólo maneja dos valores de resistencias y es el más utilizado por su sencillez y precisión.



$$V_o = V_{REF} \times \left(\frac{D_{N-1}}{2} + \frac{D_{N-2}}{2^2} + \frac{D_{N-3}}{2^3} + \dots + \frac{D_0}{2^N} \right)$$

Figura 5.8 Convertidor digital-analógico red escalera R/2R y ecuación de salida para Vo.

También se puede representar el voltaje de salida con la siguiente ecuación:

$$V_o = V_{REF} \times \left(\frac{D}{2^N} \right)$$

D es la palabra digital
N es el número de bits

En la figura 5.7 se puede observar que la resistencia que esta delante del bit menos significativo está conectada a tierra y queda en paralelo con la de este bit, por lo tanto, la resistencia equivalente del paralelo es R, esta se suma con la resistencia que queda en serie hacia el siguiente bit, dando como resultado el valor de 2R, que queda en paralelo con la de 2R del siguiente bit, esto se repite hasta llegar al bit más significativo quedando un valor de R que se suma con la de 2R conectada a la entrada no inversora del operacional, dando un valor de 3R, mismo valor que la resistencia de retroalimentación del operacional 3R, por lo que la ganancia de este es de 1, Algunos circuito no manejan la resistencia 2R conectada a la entrada inversora, pero manejan una resistencia de retroalimentación de R para seguir teniendo ganancia de 1.

Resolución

La resolución se obtiene multiplicado el voltaje de referencia entre 2^n , n es el número de bits del convertidor, por ejemplo, si el voltaje de referencia es de 5v, y el convertidor de 4 bits, la resolución es $5v / 16 = 0.3125$ volts.

Ejemplo

Encontrar el valor de V_o para cada combinación binaria del DAC R/2R que se muestra en la siguiente figura:

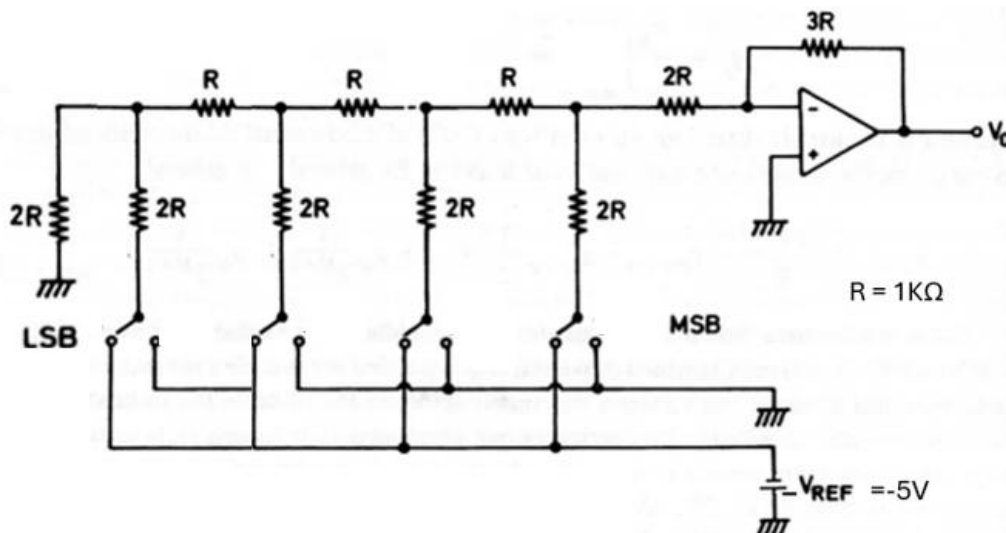


Figura 5.9 DAC R/2R de 4 bits.

MSB			LSB	Vout (volts)
0	0	0	0	0
0	0	0	1	0.312
0	0	1	0	0.624
0	0	1	1	0.936
0	1	0	0	1.248
0	1	0	1	1.560
0	1	1	0	1.872
0	1	1	1	2.184
1	0	0	0	2.496
1	0	0	1	2.808
1	0	1	0	3.120
1	0	1	1	3.432
1	1	0	0	3.744
1	1	0	1	4.056
1	1	1	0	4.368
1	1	1	1	4.680

$$V_o = V_{REF} \times \left(\frac{D}{2^N} \right)$$

D es la palabra digital
N es el número de bits

Figura 5.10 Respuesta del DAC de 4 bits con V_{REF} de 5v y R de 1KΩ.

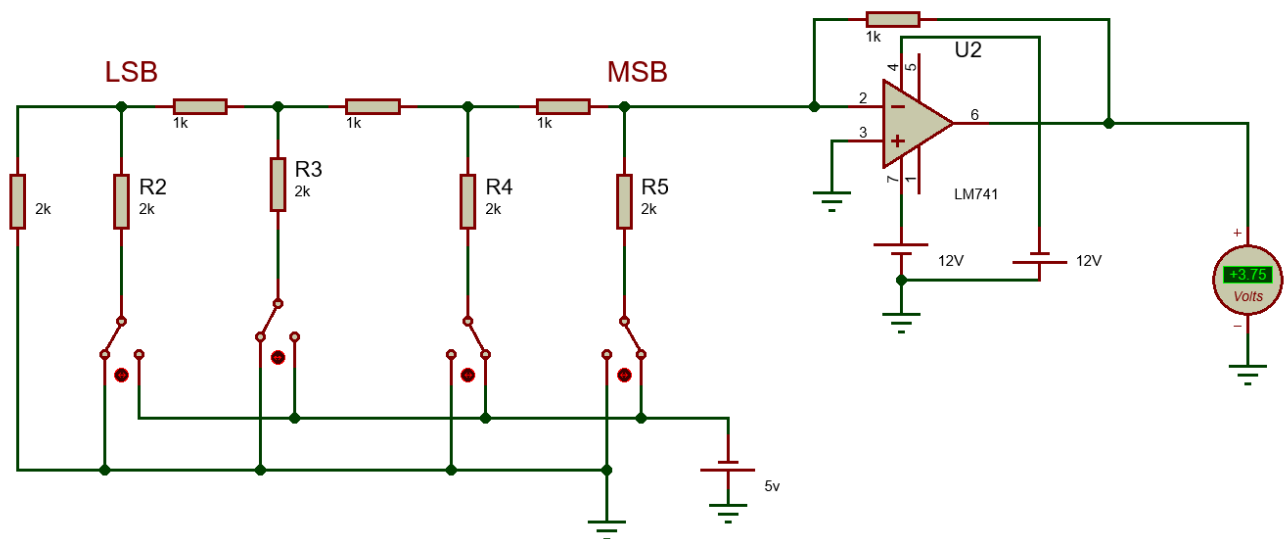


Figura 5.11 DAC R/2R del ejemplo simulado en Proteus.

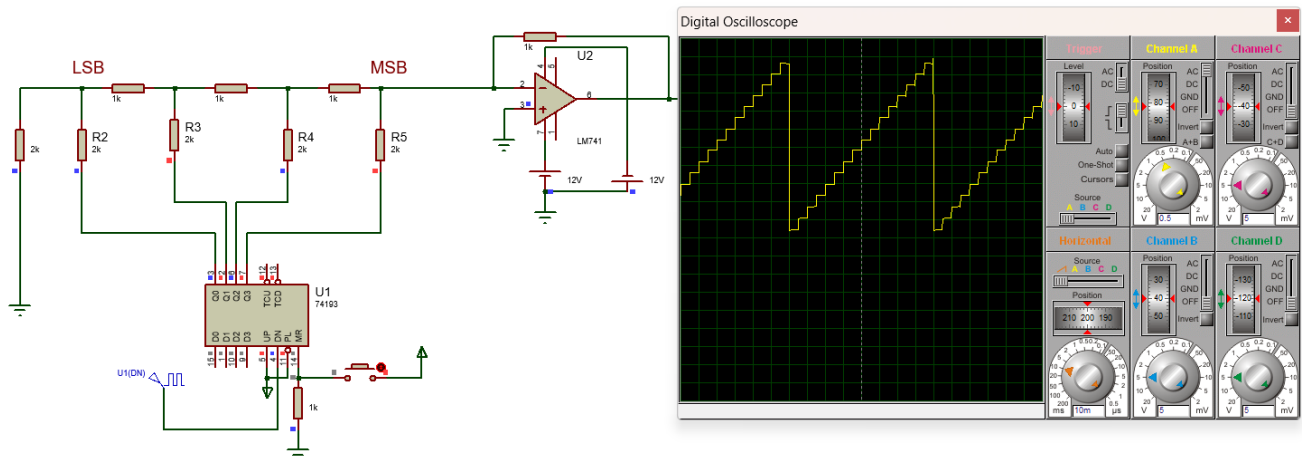


Figura 5.12 Simulación del DAC R/2R con un contador binario.

Convertidores digitales-analógicos integrados.

Modelo	Resolución (bits)	Interfaz	Descripción / Aplicación
DAC0808	8 bits	Paralelo	DAC básico, económico, muy usado en aprendizaje. se conecta un amplificador operacional.
MCP4725	12 bits	I ² C	Muy popular en Arduino y proyectos embebidos.
MCP4921	12 bits	SPI	Alta precisión, voltaje de salida ajustable.
AD558	8 bits	Paralelo	Salida de voltaje ajustable, ideal para instrumentación.
AD7303	8 bits × 2 canales	SPI	Dual DAC, pequeño y eficiente.
MAX5216	16 bits	I ² C	Alta resolución, ideal para instrumentación precisa,
DAC60004	12 bits × 4 canales	SPI	Multicanal, buena para automatización industrial.
LTC2640	12-16 bits (variante)	SPI	Ultra precisa, bajo ruido.
AD7524	8 bits	Paralelo	Para obtener un voltaje de salida, se conecta un amplificador operacional.
AD7424	12 bits	Paralelo	No requiere un convertidor I/V como el DAC0808, ya que su salida es directamente en voltaje.

Figura 5.13 Lista de convertidores digitales-analógicos para distintas aplicaciones.

DAC0808

El DAC0808 es un convertidor digital a analógico de 8 bits, muy utilizado por su facilidad de manejo, bajo costo y bastante compatible con microcontroladores. Es ideal para aplicaciones educativas, prototipos y sistemas básicos de control analógico.

Características del DAC0808:

Tipo de entrada	8 bits.
Tipo de entrada	Digital paralela (8 líneas de datos).
Tipo de salida	Corriente (requiere op-amp para convertir a voltaje).
Corriente de salida (I _{out})	Hasta 2 mA.
Tiempo de establecimiento	150 ns (para cambios rápidos).
Interfaz de referencia	Voltaje de referencia externo (usualmente 5V o 10V).
Precisión	Linealidad típica ±0.19 LSB.
Rango de temperatura	0°C a +70°C (versión comercial).
Voltaje de alimentación	±15V típico (puede funcionar con ±5V a ±15V).
Compatibilidad TTL	Compatible con salidas lógicas TTL.
Encapsulado típico	DIP-16 o SOIC-16.

Funcionamiento básico

Toma una señal digital de 8 bits (valores de 0 a 255).

Genera una corriente de salida proporcional a ese valor digital.

Para obtener una salida de voltaje, se necesita un amplificador operacional (op-amp) conectado como convertidor I/V (corriente a voltaje).

$$I_{out} = \left(\frac{V_{ref}}{256} \right) \times D$$

D es el valor decimal de la entrada digital de 8 bits (0 a 255).

V_{ref} es el voltaje de referencia externo.

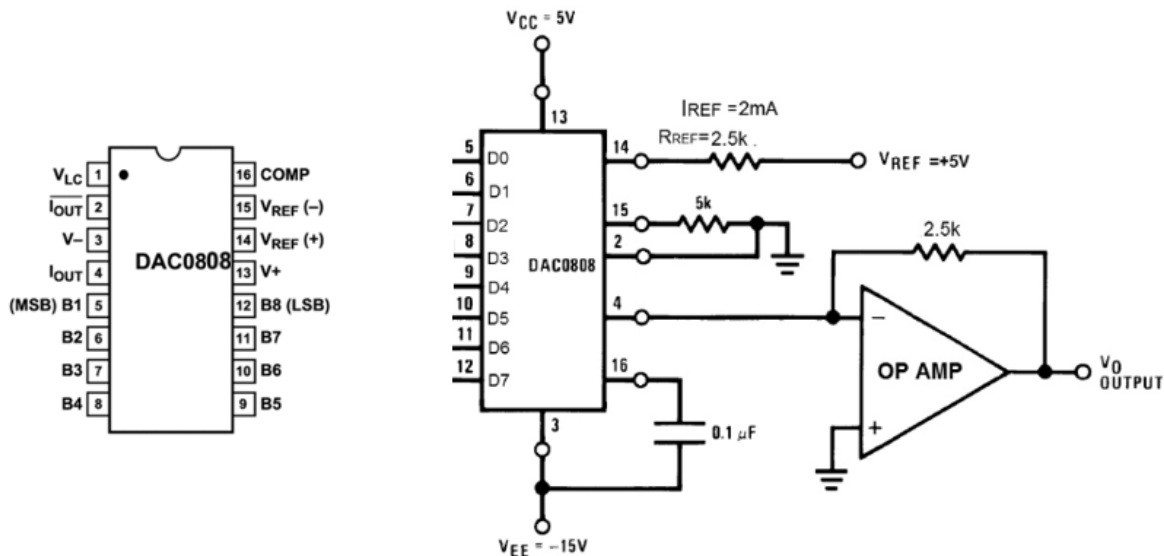


Figura 5.14 Convertidos digital-analógico DAC0808.

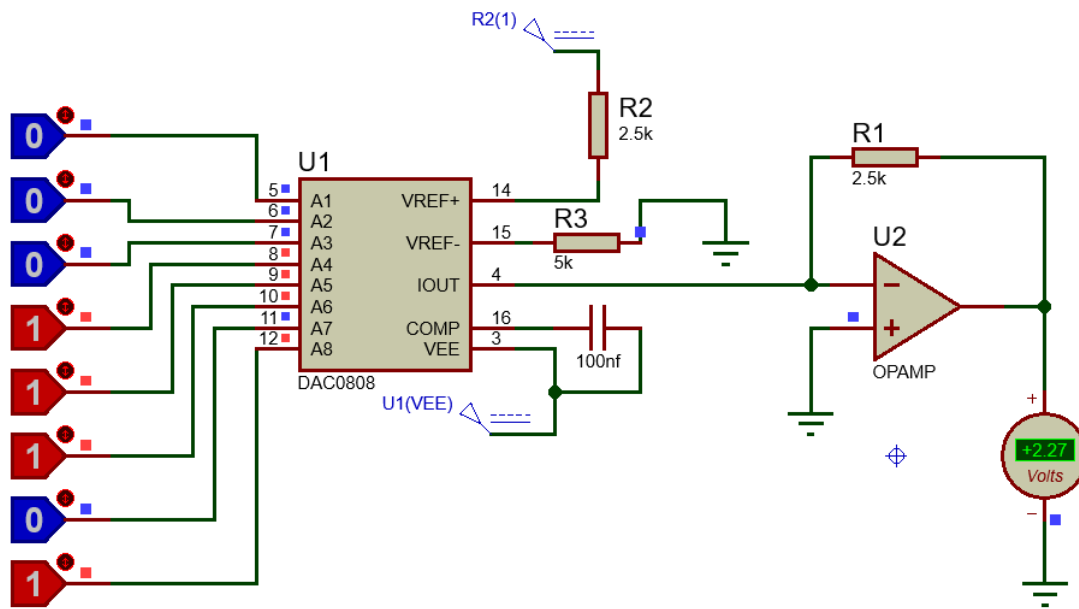


Figura 5.15 simulación en Proteus del DAC0808.

5.2 Diseño con lenguaje de descripción de hardware.

Diseñar un convertidor digital-analógico (DAC) no se puede ya que VHDL es un lenguaje de descripción de hardware digita, lo que se puede hacer es controlar un DAC externo desde un FPGA para generar las señales digitales de control necesarias.

Algunas opciones para trabajar con un DAC en VHDL son:

Controlar un DAC externo mediante interfaces digitales:

Muchos DACs se comunican a través de protocolos como SPI o I²C. se puede escribir código VHDL para generar las señales de control y datos necesarias para interactuar con el DAC.

Implementar un DAC de modulación por ancho de pulso (PWM): Puedes generar una señal PWM con VHDL y luego filtrarla con un filtro pasa bajos para obtener una señal analógica.

Ejemplo

Generar una rampa digital con el DAC0808 utilizando VHDL.

Las variables que se manejan son reset, clk, dato y una señal “contador”

El código que se puede escribir una vez iniciado el proyecto es el siguiente:

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_unsigned.all;

entity rampa_DAC0808 is
    Port ( clk : in  STD_LOGIC;
          reset : in  STD_LOGIC;
          dato : out STD_LOGIC_VECTOR (7 downto 0));
end rampa_DAC0808;

architecture Behavioral of rampa_DAC0808 is

    signal contador : std_logic_vector(7 downto 0) := (others => '0');

begin
    process(clk, reset)
    begin
        if reset = '1' then
            contador <= (others => '0');
        elsif rising_edge(clk) then
            contador <= contador + 1;
        end if;
    end process;

    dato <= std_logic_vector(contador);

```

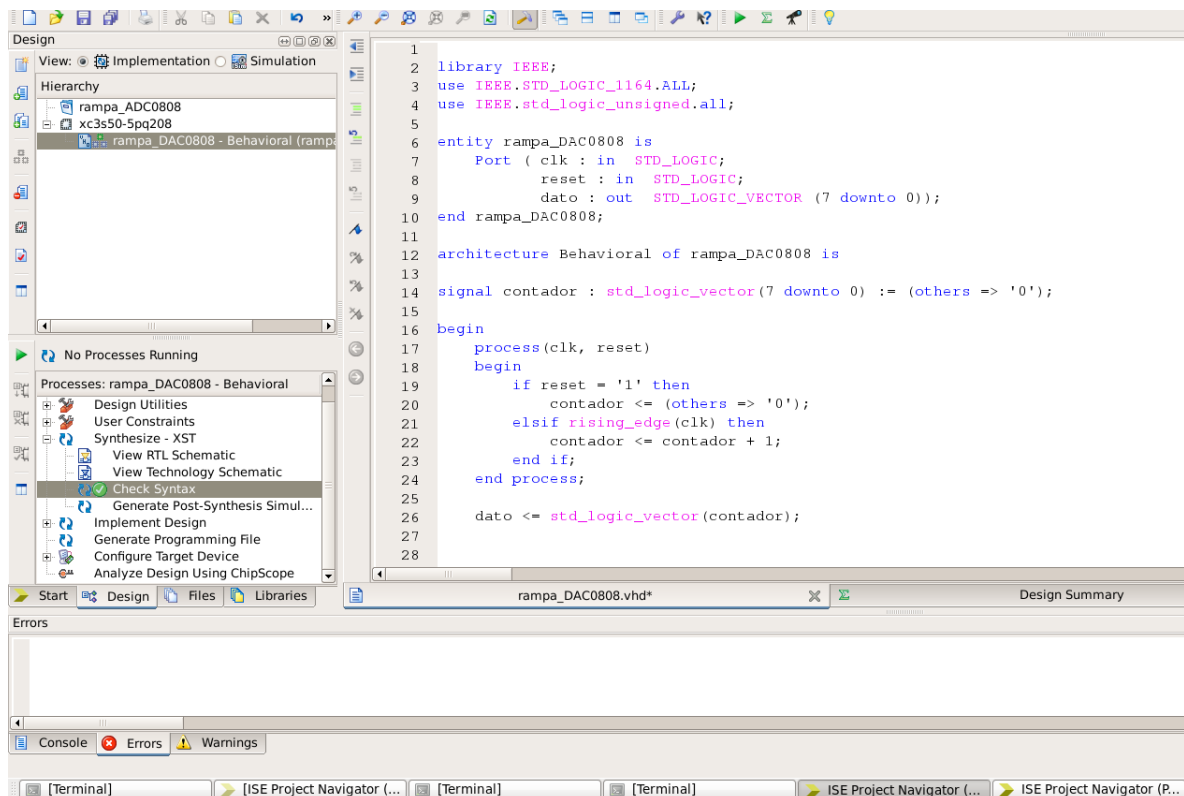


Figura 5.16 Código de rampa_ADC0808 con la sintaxis revisada.

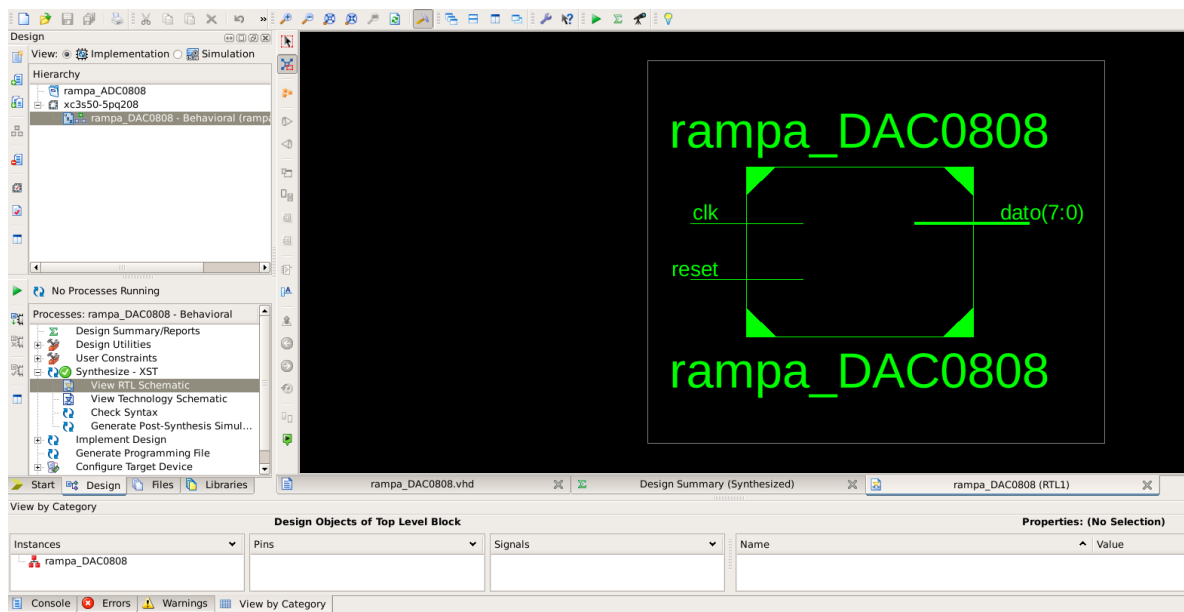


Figura 5.17 RTL Schematic par rampa_ADC0808.

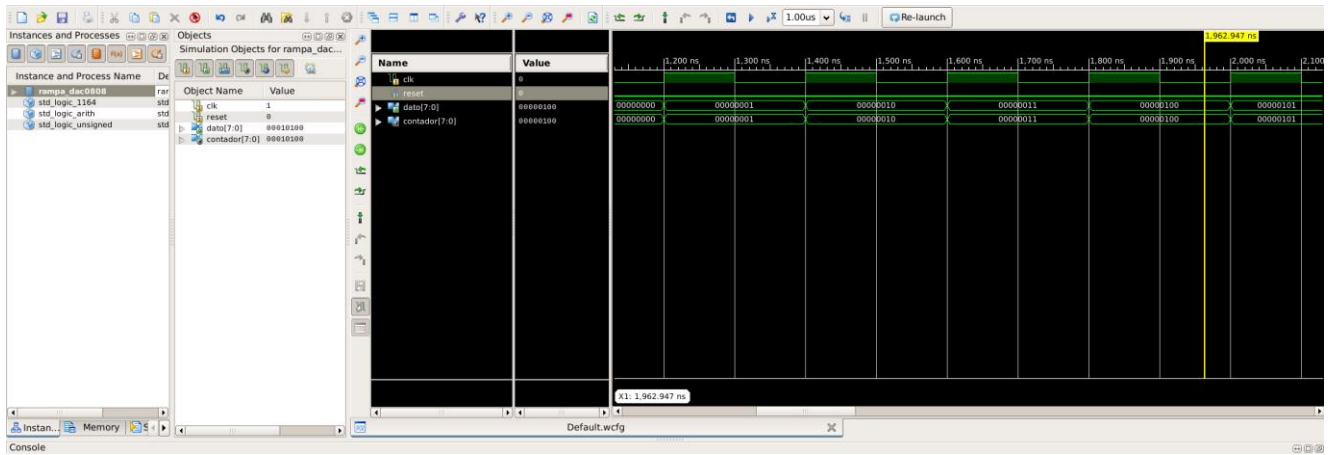


Figura 5.18 Simulación de rampa_ADC0808.

Las salidas “dato” del FPGA se conectarán al ADC0808 como la palabra digital que debe convertís a un voltaje analógico, a la salida del DAC0808 se debe conectar un amplificador operacional para convertir la corriente que entrega el DAC a un voltaje.

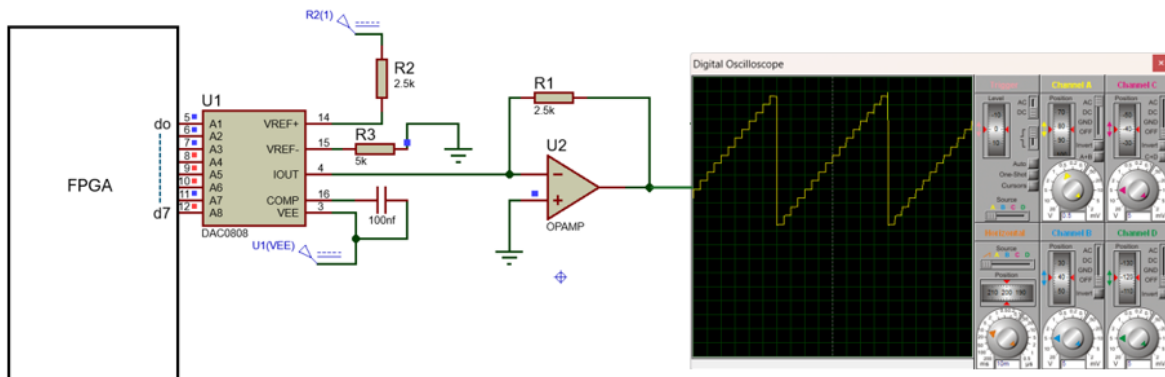


Figura 5.19 Conexiones del FPGA al DAC0808.

5.3 Funcionamiento del ADC.

Un convertidor de una señal analógico a una digital (ADC) es un circuito electrónico, normalmente un circuito integrado que recibe señales analógicas continuas, como voltajes o corrientes variables, y los convierte en datos digitales unos y ceros, para que puedan ser procesados por sistemas digitales como los microprocesadores, microcontroladores, computadoras o FPGAs. Esta conversión es fundamental para que los sistemas digitales puedan interpretar y manipular la información que se toma del mundo real como la temperatura, presión, sonido etc.

Proceso de conversión.

Para realizar la conversión de una señal analógica a digital, se puede decir que son tres etapas fundamentales y son:

Muestreo

En esta etapa se toma una porción de la señal analógica que se desea digitalizar, la frecuencia del muestreo debe ser por lo menos dos veces más rápida que la máxima frecuencia que compone la señal que se está digitalizando(según el teorema de Nyquist) para poder reproducirla lo más cercano la señal original cuando se aplique el proceso inverso, es decir convertirla nuevamente a una señal analógica. Es imposible obtener la señal analógica original debido a que se requiere tiempo en convertir a digital la muestra, es por eso por lo que lo ideal es tomar el mayor número de muestra en un intervalo de tiempo.

Cuantificación

Consiste en asignar a cada muestra el valor más cercano dentro de un conjunto finito de niveles discretos. En este proceso se introduce un pequeño error que se le conoce como error de cuantificación.

Codificación

A cada muestra cuantificada se le asignan un código binario que representan su valor digital.

Normalmente los convertidores analógicos-digitales (DAC's) se clasifican en función de los siguientes parámetros:

Velocidad de conversión

La cantidad de conversiones que puede hacer en un segundo un ADC es lo que le da la posibilidad de manejar variables físicas que varían de una forma lenta como la temperatura a muy rápidas como señales de video, es decir debe manejarse un ADC adecuado a la señal que se desea digitalizar.

Cantidad de bits (resolución)

A mayor número de bits que maneje un ADC para representar la señal analógica, se puede acercar más al valor original de esta, cuando se requiera más precisión en la representación de la señal analógica es muy importante manejar un ADC del mayor número de bits posible.

Técnicas de conversión ADC

Existen diversas técnicas para convertir una señal analógica a digital, sólo se revisarán algunas de ellas para entender el funcionamiento de estos convertidores, en la actualidad existen una gran cantidad de circuitos integrados que manejan alguna de estas técnicas, lo importante es saber escoger el convertidor que pueda manejar adecuadamente la señal que se quiera procesar, algunas de estas técnicas son:

Conversión de rampa digital simple.

Conversión de doble rampa.

Conversión de aproximaciones sucesivas.

Conversión flash.

Conversión de rampa digital simple.

Este método utiliza un convertidor digital-analógico para genera una señal de rampa que se compara con la señal de entrada en el operacional. El tiempo que tarda la rampa en igualar la señal de entrada se mide y se convierte en un valor digital. Es sencillo y económico, pero relativamente lento, por lo que se utiliza en aplicaciones donde la velocidad no es crítica.

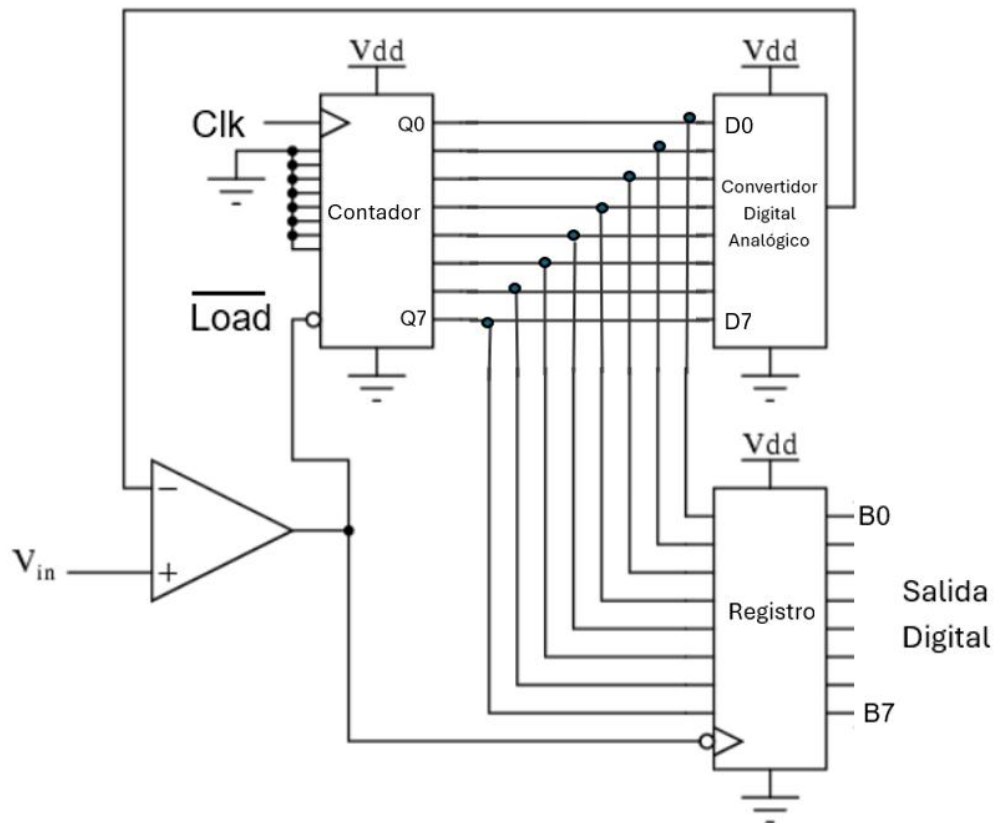


Figura 5.20 Convertidor analógico-digital de rampa digital simple.

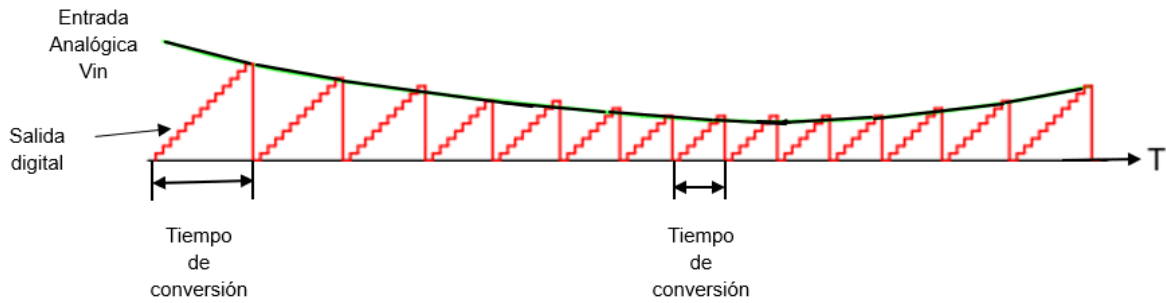


Figura 5.21 Señal de salida del ADC de rampa digital.

Funcionamiento.

Cuando no hay voltaje en la entrada no inversora del comparador, este da como salida un nivel bajo provocando que el registro tome el valor del contador y enseguida el contador se cargue con ceros a la salida, la salida del registro es el valor de conversión. Al poner un valor diferente de cero en la terminal no inversora este dará un nivel alto de salida, lo que permite que el contador se incremente por cada pulso de reloj, al mismo tiempo el DAC irá incrementando su voltaje de salida, cuando este sea un poco mayor que el voltaje de entrada V_{in} en la terminal no inversora, el comparador cambia de estado a nivel bajo permitiendo primero que el registro tome el valor del contador en el flanco negativo y enseguida el contador se carga con ceros.

El tiempo más largo de conversión es cuando se requiere de 255 pulsos de reloj para que el DAC alcance el valor del voltaje de entrada a convertir, por lo tanto, si se tiene una señal de reloj de 255 KHz, se lograrían aproximadamente 1000 conversiones por segundo, este convertidor es lento. Se logra más conversiones cuando el voltaje de entrada no es el máximo que puede convertir, sin embargo, siempre se toma el tiempo más largo de conversión como referencia de velocidad de conversión.

Tiempo de conversión para este caso es de 1 ms.

Resolución = $V_{in} \text{ más} / (2^n - 1)$; $5v / (256 - 1) = 19.6$ milivolts.

El error de un convertidor analógica-digital es un medio de la resolución ($\frac{1}{2}$ LSB), quiere decir que el bit menos significativo puede estar oscilando entre 0 y 1.

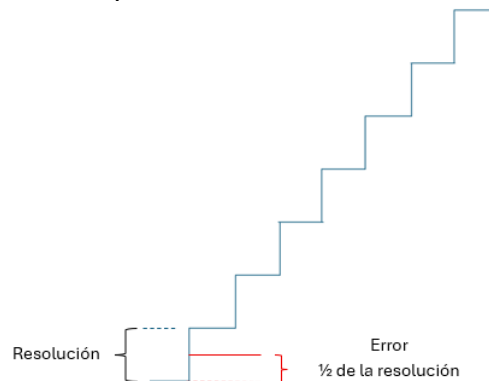


Figura 5.22 Resolución y error de un convertidos analógico-digital.

Convertidor analógico-digital de rampa analógica.

Este tipo de convertidor no maneja un DAC para generar el voltaje que se compara con el de la entrada a convertir, utiliza un circuito integrador inversor con un voltaje de referencia negativo generar una rampa positiva.

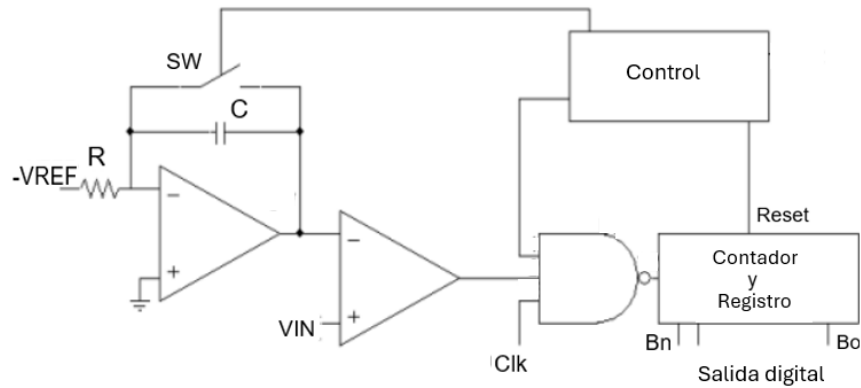


Figura 5.23 Convertidor analógico-digital de rampa analógica.

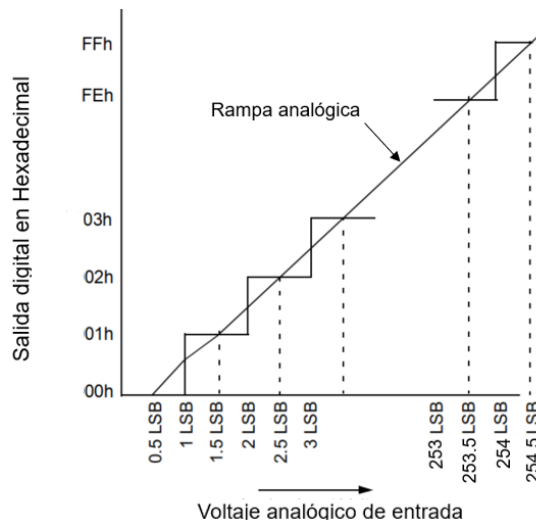


Figura 5.24 Salida del convertidos analógico-digital de rampa analógica.

Funcionamiento

Inicialmente el circuito de control cierra el switch para descargar el condensador, al abrir el switch comienza a generar una rampa de voltaje positiva que entra al comparador, si el voltaje de entrada es mayor que el valor de la rampa, el circuito de control deja pasar los pulsos de reloj al contador, cuando la rampa alcance el valor del voltaje de entrada VIN el comparador cambia de estado su salida para bloquear los pulsos de reloj deteniendo el conteo, el valor con el que quedo el contador es transferido al registro para mostrarlo a la salida, este es el valor equivalente en binario al voltaje de entrada.

Si el convertidor es de 8 bits, la rampa se calcula de tal manera que alcance su valor máximo cuando hayan pasado 256 pulsos de reloj, por lo tanto, si el reloj tiene una frecuencia de 500 kHz, el tiempo que debe transcurrir para que la rampa alcance su valor máximo será de $(1/500 \text{ KHz}) \times 256$ esto es, 512 μ segundos, este es el tiempo que se toma como el necesario para una conversión completa, aproximadamente 1953 conversiones por segundo.

Si el voltaje máximo de la rampa es de 5 volts, se puede calcular los valores del capacitor y resistencia con la función de voltaje de salida del integrador.

$$V_{sal} = -\left(\frac{V_{ent}}{RC}\right) * t$$

Si se tiene un voltaje de referencia de -1 volt, se puede proponer el valor del capacitor, por ejemplo 100 nF y se despeja el valor de R.

$$5v = (1v / (10nF \times R)) \times 512\mu\text{segundos}$$

$$R = 10.24K\Omega$$

Resolución

Igual que en el convertidor analógico-digital de rampa digital, la resolución viene dada como el voltaje máximo a convertir entre $(2^n - 1)$, por lo tanto, para este caso será:

$$\text{Resolución} = 5v / 255 = 19.6 \text{ milivolts.}$$

El error es la mitad de la resolución, es decir $\frac{1}{2}$ de 19.6 milivolts = 9.8 milivolts.

Circuito de captura y sostenimiento S&H.

Normalmente en un convertidor analógico-digital, la conversión no se hace directamente del voltaje de entrada, este se captura en un circuito S&H, se desconecta la entrada y enseguida inicia la conversión con el voltaje capturado, al terminar la conversión se reinicia el circuito S&H para una nueva captura y conversión.

Ejemplo de un circuito S&H del convertidor ADC del microcontrolador PIC18F4550.

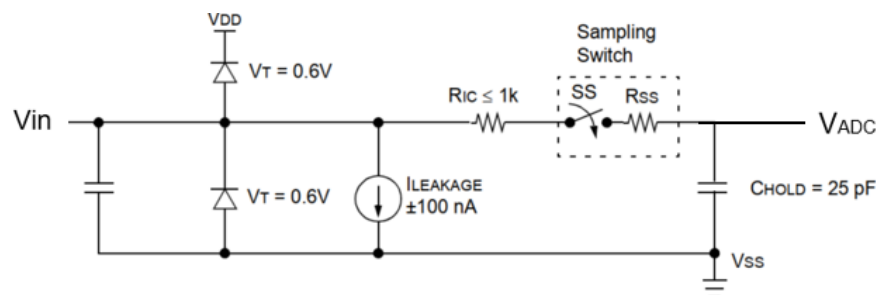


Figura 5.25 Circuito S&H del ADC del microcontrolador PIC18F4550.

Este tipo de circuitos obliga a que el voltaje a ser convertido, este presente de manera estable un determinado tiempo en la entrada, para asegurar que la conversión es correcta, a este tiempo se le llama tiempo de adquisición (TACQ) y cada circuito convertidor integrado lo especifica.

Convertidor analógico-digital de aproximaciones sucesivas(SAR).

Este tipo de convertidor está formado básicamente de cinco bloques que son un registro de aproximaciones sucesivas(SAR), un convertidos digital-analógico(DAC), un circuito operacional comparador, un circuito de captura y sostenimiento(S&H) y un registro de salida.

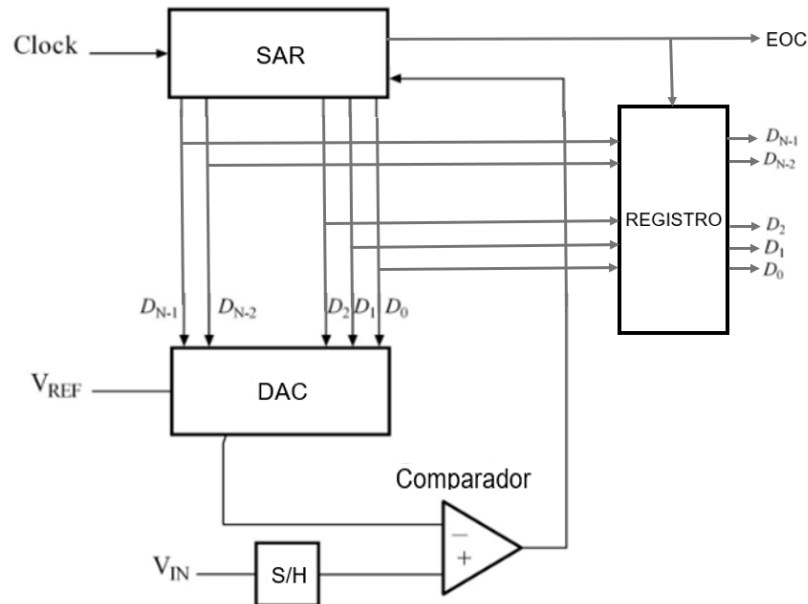


Figura 5.26 Convertidor analógico-digital de aproximaciones sucesivas.

Funcionamiento

El registro de aproximaciones sucesivas inicia poniendo el bit más significativo (MSB) en 1 y los demás bits en 0. Estos bits se introducen en el DAC y produce un voltaje analógico equivalente a $(V_{ref}/2)$, este voltaje llega al circuito comparador para su comparación con el voltaje de entrada muestreado. Si este voltaje analógico excede V_{in} , entonces el comparador hace que el SAR restablezca este bit(lo regresa a 0); de lo contrario, el bit se deja como 1. Luego, el siguiente bit se establece en 1 y se realiza la misma prueba, continuando esta búsqueda binaria hasta que se haya probado cada bit en el SAR. El código resultante es la aproximación digital del voltaje de entrada muestreado y finalmente es emitido por el SAR al final de la conversión (EOC) para capturar el valor convertido en el registro.

Por cada prueba de un bit se requiere de un pulso de reloj, lo que indica que una conversión de n bits, requiere de n pulsos de reloj, lo que hace a este tipo de convertidor uno de los más rápidos y utilizados por muchos microcontroladores y sistemas embebidos que lo incluyen internamente.

VCD volts	Inicio				
15				1111	1111
14			1110		1110
13				1101	1101
12		1100			1100
11				1011	1011
10			1010		1010
9				1001	1001
8	1000				1000
7				0111	0111
6			0110		0110
5				0101	0101
4		0100			0100
3			0011	0011	
2		0010		0010	
1			0001	0001	
0				0000	
CLK	inicio	Primer	Segundo	Tercer	Cuarto

← Pulsos de reloj

Figura 5.27 Respuesta de un ADC de aproximaciones sucesivas de 4bits.

La resolución se obtiene de la misma manera que en los convertidores anteriores, es decir el voltaje máximo a convertir entre $(2^n - 1)$, para el ejemplo de la tabla anterior la resolución es $15V / 15 = 1V$.

El error es la mitad de la resolución, es decir $\frac{1}{2}$ (LSB), esto es $\frac{1}{2}1 \text{ volt} = 0.5 \text{ volts}$.

Los errores más relevantes del ADC de SAR

Error de cuantización.

$$E_{\text{cuantización}} = \pm \frac{1}{2} \cdot \text{LSB}$$

Error de offset

Es un desplazamiento constante en todos los niveles de salida. Puede deberse a errores en el comparador o desajustes en el DAC interno.

Error de ganancia.

Sucede si la amplitud total de salida no se ajusta correctamente al rango completo del ADC. Causado por una referencia incorrecta o errores acumulados en el DAC.

No linealidad diferencial (DNL).

Mide cuánto se desvía el tamaño de cada paso de 1 LSB respecto al ideal.

Idealmente: $\text{DNL} = 0 \text{ LSB}$.

Si $\text{DNL} > 1 \text{ LSB}$, pueden producirse códigos faltantes (el ADC no genera ciertos valores).

No linealidad integral (INL).

Es el error acumulado entre la curva real del ADC y la curva ideal.

Mide la distorsión general de la conversión.

Errores del DAC interno.

En un SAR ADC, la precisión del DAC interno es crucial.

Cualquier error de referencia, resistencia o capacitancia en el DAC se traduce directamente en error de conversión.

Error del comparador.

El comparador determina si el valor analógico generado por el DAC es mayor o menor que la entrada.

ADC0804

Un convertidor analógico-digital de aproximaciones sucesivas integrado es el ADC0804.

Es de 8 bits, tiene un circuito de reloj interno el cual requiere de un capacitor y una resistencia externas, la frecuencia viene dada por la relación:

$$F = 1 / (1.1RC)$$

También puede ser utilizada una señal de reloj externa aplicándola a la terminal CLK IN.

Este convertidor es una interfaz para microprocesador ya que maneja el tercer estado(HZ) en su bus de datos(D0 a D7).

El tiempo de acceso es 135 ns, maneja un rango de voltaje de entrada analógica de 0 a 5 volts con suministro único de alimentación de 5volts.

El tiempo de conversión es de 100 μ s.

La resolución viene dada por: $V_{REF} / (2^n - 1)$, n es el número de bits y V_{REF} es de 5 volts, También puede ser 2.5 volts.

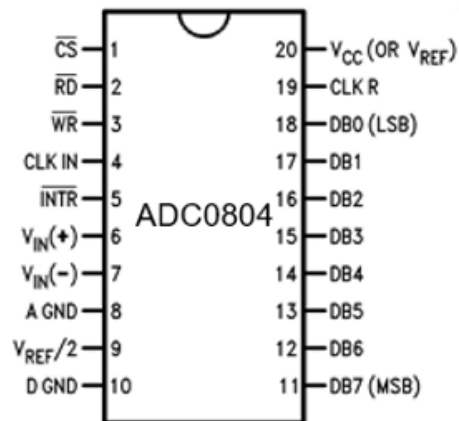


Figura 5.28 Convertidor analógico-digital de aproximaciones sucesivas ADC0804.

Descripción de las terminales:

CS (selector del chip) entrada activa a nivel bajo, si la terminal está a nivel alto, el bus de datos estará en tercer estado(HZ) y no tendrá ningún efecto las terminales RD y WR.

RD (lectura) línea de entrada activa a nivel bajo, activa el buffer de salida para mostrar el valor de la conversión en el bus de datos(D7 a D0).

WR (escritura) línea de entrada activa a nivel bajo para iniciar una conversión.
INTR (interrupción) línea de salida activa a nivel bajo para indicar que ha terminado una conversión, es equivalente a la de fin de conversión EOC, puede ser utilizada si se requiere hacer conversiones continuas automáticas, se conecta a la línea de entrada WR.

Vref/2 Line de entrada para reducir el voltaje de referencia interno a 2.5 volts por lo que la entrada analógica cambia de 0 a 2.5volts.

CLK OUT línea de salida que muestra la señal del reloj interno.

CLK IN línea de entrada para señal de reloj externa o para conectar el capacitor si se maneja el reloj interno.

A GND terminal de tierra analógica.

D GND terminal de tierra digital.

DB7 a DB0 Salidas digitales, db7 es el MSB, este bus es de tercer estado se activa cuando se selecciona el CS y una operación de RD o WR.

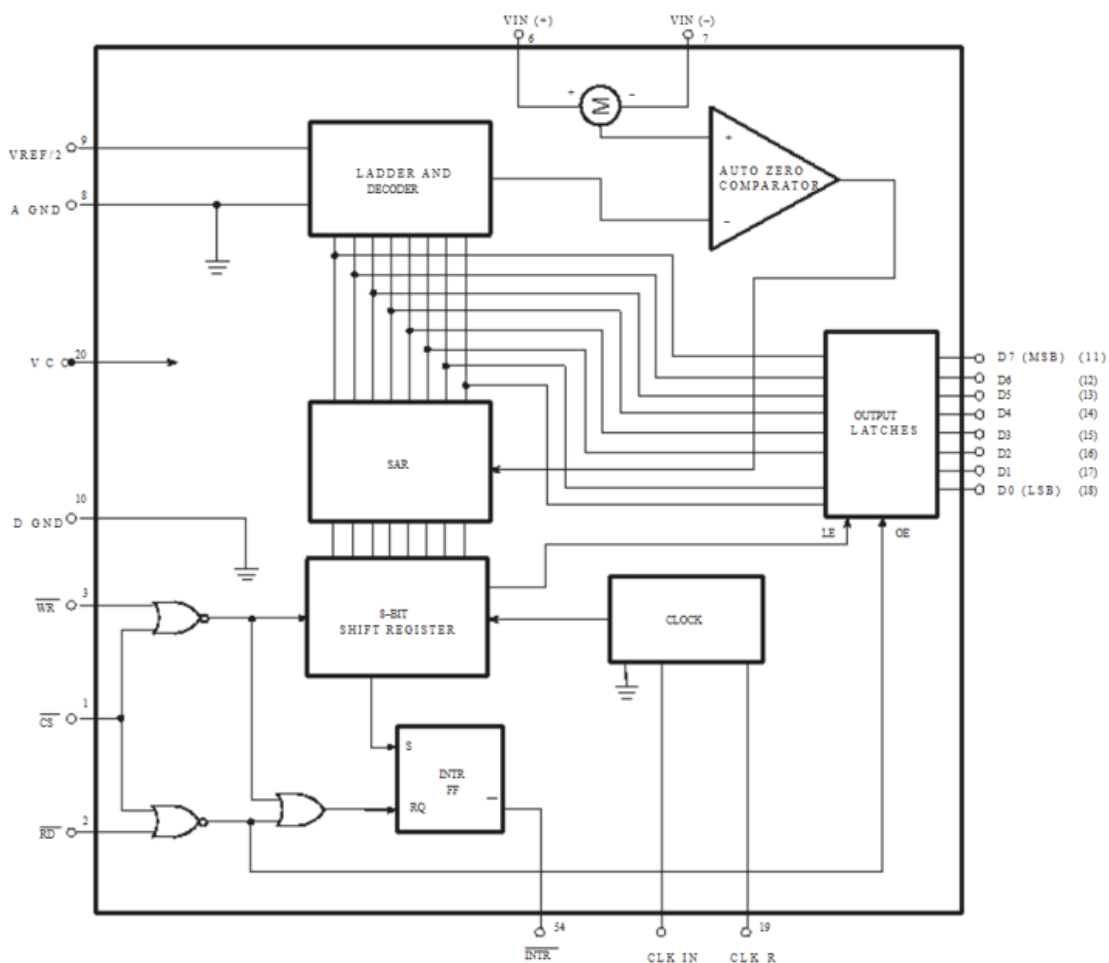


Figura 5.29 Diagrama a bloques del ADC0804. [22]

Para probar el funcionamiento del ADC0804 puede retroalimentar la salida INT a la entrada WR, de esta forma cada que termine una conversión iniciara una nueva automáticamente, el circuito para la prueba se muestra la siguiente figura:

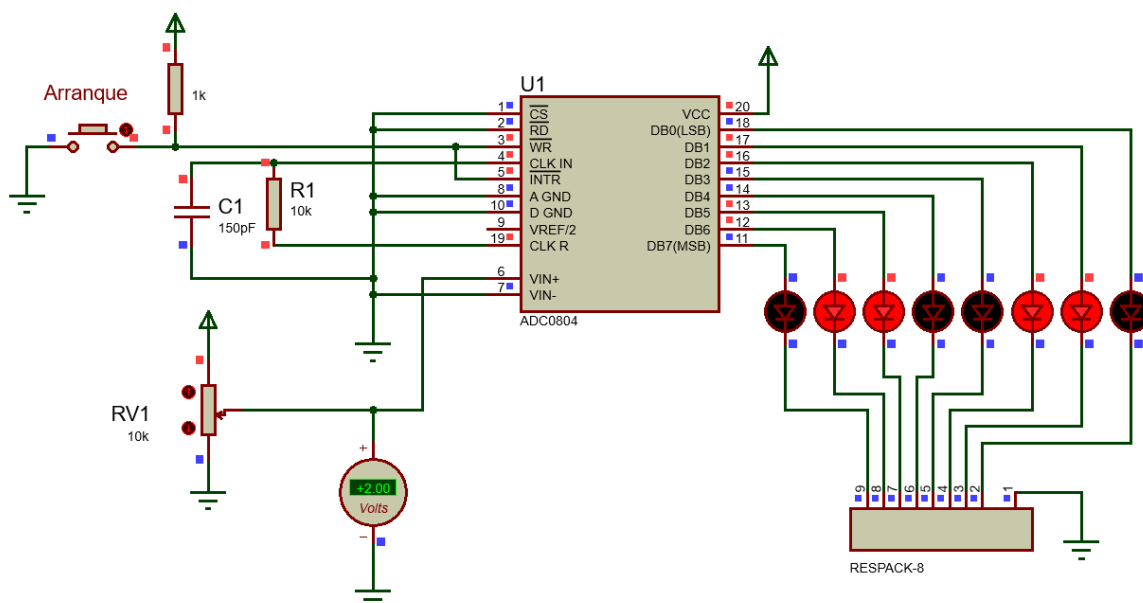


Figura 5.30 Prueba del ADC080 en Proteus.

En la simulación están llegando 2 volts a la entrada VIN+ y como la alimentación está a 5 volts, aplicando la formula, la resolución es de 19.6 milivolts, la salida digital que presenta el convertidor en decimal es 102, si se multiplica este valor por la resolución, 102×19.6 milivolts de un voltaje de 1.99 volts, muy cercano al voltaje de entrada. Hay que recordar que el error que presenta el convertidor es $\frac{1}{2}$ de la resolución, es decir $\frac{1}{2}$ del LSB.

Los valores típicos de resistencia y capacitor para el reloj interno propuestas por el fabricante son de 10K Ω y 150p respectivamente, para generar una frecuencia de aproximadamente 666 KHz.

5.4 Aplicaciones.

Los convertidores digitales-analógico (DAC's) y los convertidores analógicos-digitales(ADC's) son fundamentales en la electrónica, estos dispositivos permiten interactuar a los dos grandes campos, el digita y el analógico, gracias a ellos se puede hacer un control completo de las variables físicas o digitales desde su lectura hasta el actuador.

Algunas de las aplicaciones de los convertidores analógicos-digitales ADC son:

Procesamiento digital de señales.

La mayoría de los sensores que se usan en control, entregan una señal analógica en forma de voltaje o corriente, estas señales son convertidas a una señal digital por estos dispositivos para que se pueda aplicar a un algoritmo ejecutado por algún microprocesador, microcontrolador, sistemas embebidos o una computadora.

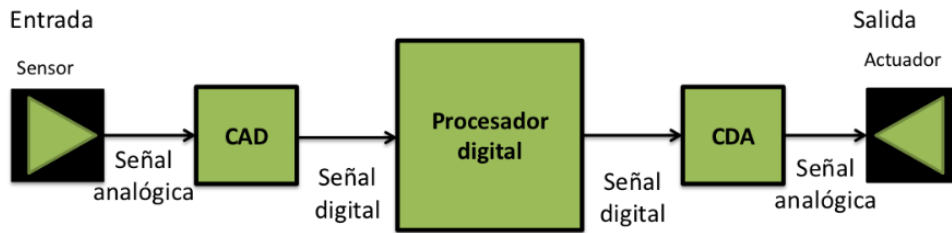


Figura 5.31 Procesamiento digital de señales. [23]

Grabación y procesamiento de audio.

El audio es una señal analógica que es traducida en un pequeño voltaje o corriente por un micrófono, esta se digitaliza y se almacena para su procesamiento.

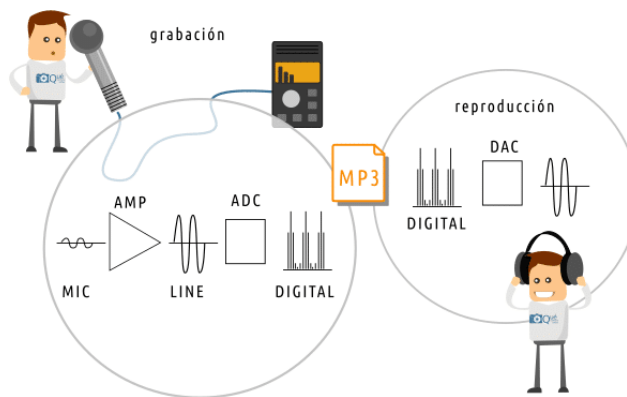


Figura 5.32 Digitalización de la señal de audio y su recuperación. [24]

Instrumentación.

Los instrumentos de medición convierten las señales de los sensores (temperatura, presión, nivel, etc.) en datos digitales para su análisis.

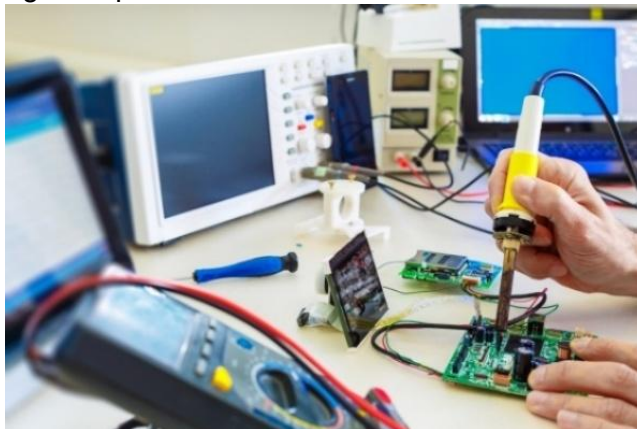


Figura 5.33 Instrumentos de medición que utilizan ADC's. [25]

Automatización industrial.

Monitoreo y control de procesos con el manejo de señales analógicas que entregan los sensores.



Figura 5.34 Automatización industrial con equipos que manejan ADC's. [26]

Algunas de las aplicaciones de los convertidores digitales-analógicos DAC son:

Reproducción de audio y video.

Convierte los datos digitales almacenados en audio y video para ser mostrados en pantalla, televisores etc.

Control de sistemas físicos.

Algunos sistemas de control de motores o temperatura entre otros requieren señales analógicas para hacer su función, estos convertidores toman los valores digitales entregados por el sistema y regresan la señal analógica de control.

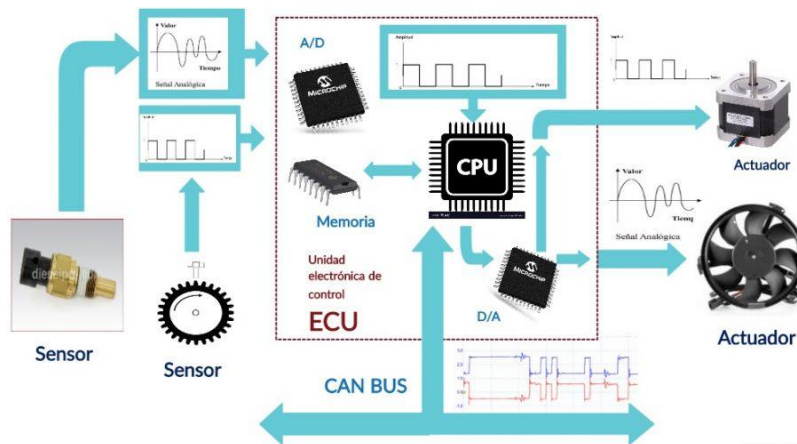


Figura 5.35 Control de motores utilizando ADC y DAC.

Instrumentación y prueba.

Los DAC se utilizan en equipos de prueba para generar señales analógicas precisas, necesarias en la calibración y evaluación de dispositivos electrónicos.

La mayoría de los sistemas electrónicos que se utilizan para llevar a cabo el control sobre variables físicas, trabajan los dos tipos de convertidores, no con circuitos integrados separados, normalmente lo trabajan con microcontroladores, sistemas embebidos o PLC's que integran los dos tipos de convertidores.

Ejemplo Voltímetro con el ADC0804.

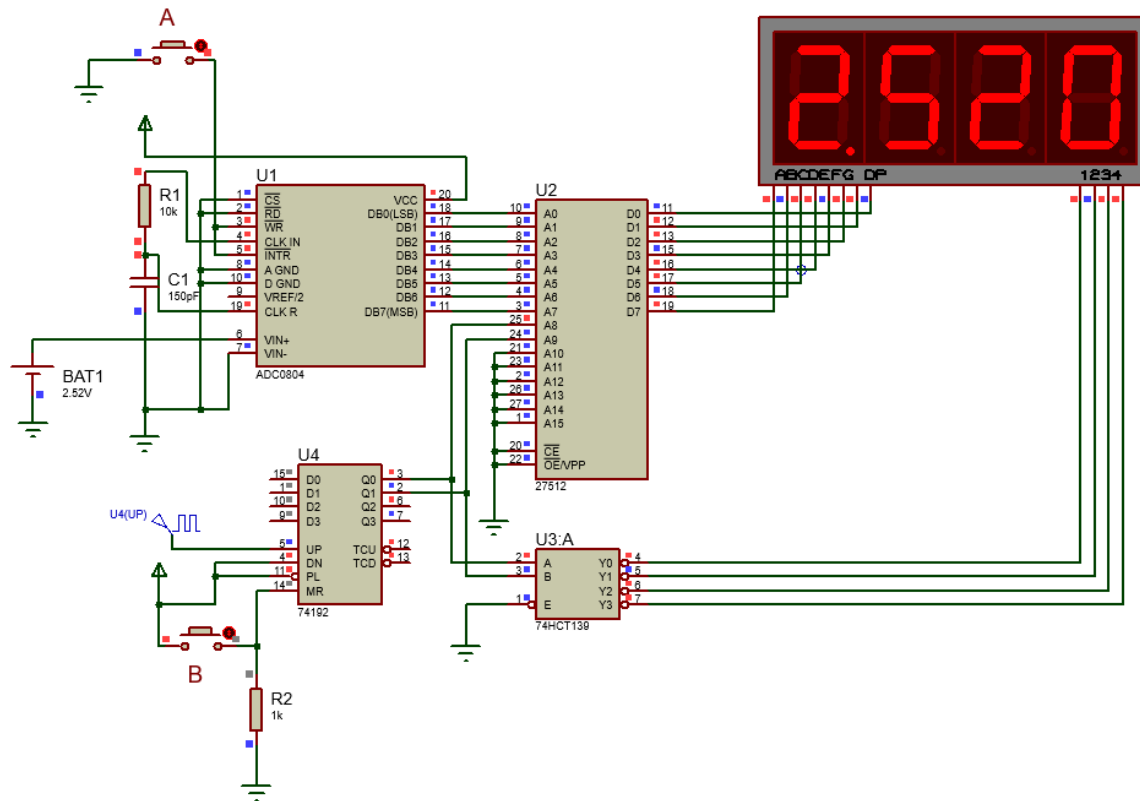


Figura 5.36 Voltímetro con el ADC0804.

Funcionamiento

Se configura el ADC en forma de carrea libre, es decir sólo requiere de un pulso inicial en el botón A para hacer conversiones consecutivas debido a la conexión de la terminal INT a WR, cada que termina una conversión el nivel bajo que se genera en la terminal INT conectada a WR inicia una nueva conversión, la salida del ADC conectadas como direcciones de la memoria pueden seleccionar de la localidad 0 a la 255 para (de 00H a FFh en hexadecimal)

Por cada conversión se manejan cuatro localidades de memoria, una para cada display, de la localidad 00h a la FFh al primer display, de la 100h a 1FFh para el segundo display, de la 200h a la 2FFh el tercer display y de la 300 a la 3FFh el cuarto display.

Si se supone que el convertidor tiene 0 volts de entrada, la salida es 00h si agrupamos los bits en grupos de 4, el display debe mostrar 0.000, esto se logra con el contador y el decodificador, cuando el contador tiene 00 de salida el decodificador activa la señal de salida y0 activando el display 0, en esta localidad de memoria 000h debe estar guardado el código del display de 7 segmentos para que exhiba el 0., si se mantiene el valor de salida del ADC, el contador se incrementa en uno, lo que direcciona la memoria en la localidad 100h, y el decodificador activa la salida y1 activando el display 1, en esta localidad de memoria se guarda el código para mostrar el 0, si el contador se incrementa en uno sin modificar la salida el ADC, estará dando la dirección de memoria 200h, y activando la salida del decodificador y2 activando el display 2, en esta localidad se guarda el código para mostrar también un 0, por último si se incrementa el contador sin modificar la salida el ADC,

estará dando la dirección 300h y la salida del decodificador y3 activando el tercer display, debe estar guardado el código para que se vea el 0.

La frecuencia del contador debe ser por lo menos 240 Hz para poder ver la información en los cuatro displays sin que se vean parpadeando.

De acuerdo con lo anterior, se deben guardar 1024 datos en la memoria para poder mostrar el voltaje en la entrada del convertidor, 256 códigos por cada display.

El código que se guarda en la memoria depende del display si es de ánodo o cátodo común, no se requiere decodificador para el display ya que en la memoria se guarda los segmentos que deben prender para cada número, se guardan unos para prender el segmento si es de cátodo y ceros si es de ánodo.

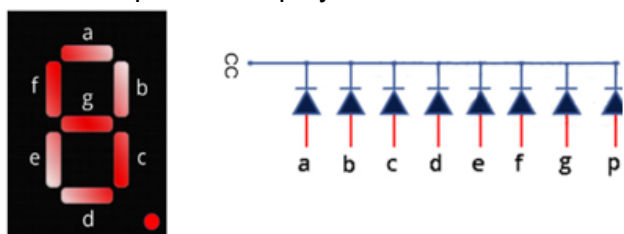
Existe un programa libre para editar un archivo hexadecimal que es el que se carga en la memoria de Proteus se llama HxD y la liga donde se puede bajar es:

[HxD Hex Editor Descargar \(2025 Última versión\)](#)

La resolución del convertidor es 19.6 milivolts escala máxima daría 4.998 volts.

Se puede redondear la resolución a 20 milivolts si se quiere, solo hay que tener en cuenta el error, a máxima escala será 0.1 volt.

Los datos para los números para un display de cátodo común son:



a	b	c	d	e	f	g	p.d	Número	Código hexadecimal	Número con punto	Código hexadecimal
1	1	1	1	1	1	0	0	0	FC	0.	FD
0	1	1	0	0	0	0	0	1	60	1.	61
1	1	0	1	1	0	1	0	2	DA	2.	DB
1	1	1	1	0	0	1	0	3	F2	3.	F3
0	1	1	0	0	1	1	0	4	66	4.	67
1	0	1	1	0	1	1	0	5	B6	5.	B7
1	0	1	1	1	1	1	0	6	BE	6.	BF
1	1	1	0	0	0	0	0	7	E0	7.	E1
1	1	1	1	1	1	1	0	8	FE	8.	FF
1	1	1	0	0	1	1	0	9	E6	9.	E7

Figura 5.37 Tabla de valores hexadecimales para los números en display de 7 segmentos.

Los datos almacenados en la memoria para el voltaje que muestra la figura 2.520, son:

Localidad 07Eh = DBh

Localidad 17Eh = B7h

Localidad 27Eh = DBh

Localidad 37Eh = FCh

Observe que las localidades del convertidor 7Eh no cambian en el barrido de los cuatro display.

Trabajo

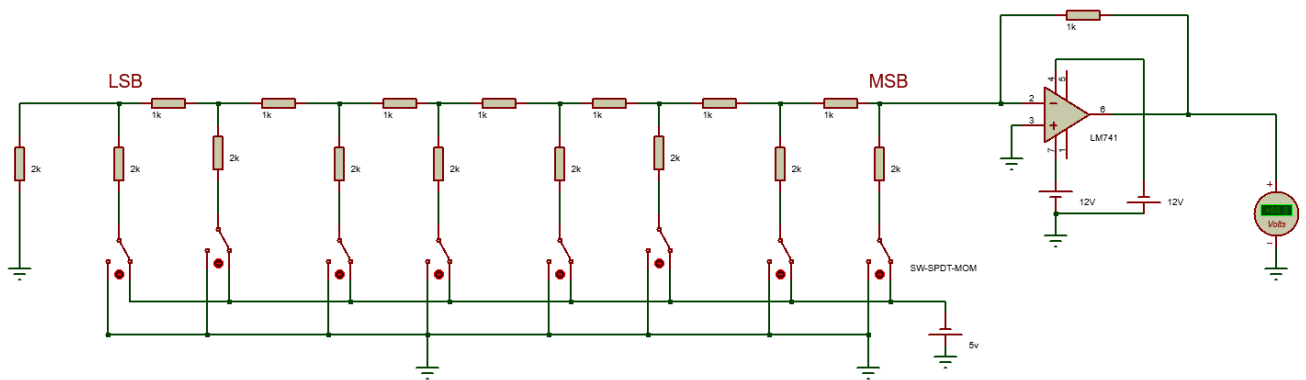
Realizar una investigación sobre las diferentes técnicas de conversión analógica-digital y digital-analógica y sus aplicaciones en un documento de por lo menos 5 cuartillas.

Práctica

Objetivo

Comprender el funcionamiento de un convertidor digital-analógico DAC.

Implementar el siguiente circuito convertidor digital-analógico, calcule la resolución y el error si el voltaje de referencia es de 5 volts, haga una tabla de los valores teóricos y reales medidos para 20 combinaciones digitales de entrada utilizando todos los bits por lo menos una vez.



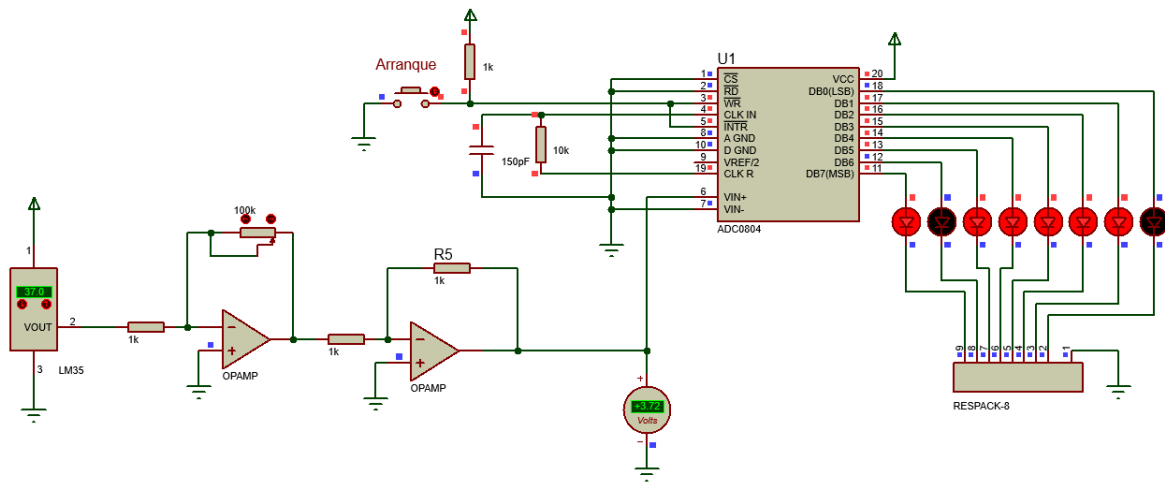
Palabra digital	Valor teórico	Valor medido

Práctica

Objetivo

Comprender el funcionamiento de un convertidor analógico digital ADC.

Implementar el siguiente circuito y ajuste el potenciómetro del amplificador inversor para que entregue un voltaje de salida de 0 a 5 volts para una temperatura del sensor lm35 de 0 a 50 grados Celsius, calcule la resolución y el error, haga una tabla para 10 valores de temperatura, voltaje de entrada al ADC y la palabra digital entregada por el ADC.



Temperatura	Vin+	Palabra digital

Examen

1. ¿Cuál es la función principal de un convertidor analógico-digital (ADC)?
 - a) Generar señales analógicas.
 - b) Convertir señales digitales en analógicas.
 - c) Convertir señales analógicas en digitales.
 - d) Filtrar señales de ruido.
2. ¿Qué hace un convertidor digital-analógico (DAC)?
 - a) Convierte señales analógicas en digitales.
 - b) Convierte señales digitales en analógicas.
 - c) Amplifica señales digitales.
 - d) Reduce el ruido de señales analógicas.
3. ¿Qué es la resolución en un convertidor?
 - a) La velocidad máxima de conversión.
 - b) La precisión en la conversión.
 - c) La capacidad de manejar múltiples señales.
 - d) La capacidad de amplificar señales.
4. ¿Qué tipo de convertidor es más rápido en la conversión A/D?
 - a) De rampa.
 - b) De aproximación sucesiva.
 - c) Flash (comparador paralelo).
 - d) Integrador dual.
5. ¿Qué tipo de convertidor A/D es más lento?
 - a) Flash.
 - b) De aproximación sucesiva.
 - c) Integrador dual.
 - d) De rampa.
6. ¿Qué tipo de ADC utiliza un comparador para cada nivel de cuantificación?
 - a) De rampa.
 - b) Flash.
 - c) De aproximación sucesiva.

d) Integrador dual.

7. ¿Cuántos comparadores se necesitan para un ADC Flash de 4 bits?

a) 4

b) 15

c) 16

d) 8

8. ¿Qué tipo de ADC utiliza un contador para generar la señal digital?

a) De rampa.

b) Flash.

c) De aproximación sucesiva.

d) Integrador dual.

9. ¿Qué tipo de ADC utiliza un integrador y un comparador para convertir la señal?

a) De rampa digital.

b) Flash.

c) De aproximación sucesiva.

d) Integrador dual.

10. ¿Qué tipo de ADC utiliza un registro de aproximación sucesiva?

a) De rampa.

b) Flash.

c) De aproximación sucesiva.

d) Integrador dual.

11. ¿Qué tipo de ADC es más adecuado para aplicaciones de alta velocidad?

a) De rampa.

b) Flash.

c) De aproximación sucesiva.

d) Integrador dual.

12. ¿Qué tipo de ADC es más adecuado para aplicaciones de alta precisión?

a) De rampa.

b) Flash.

c) De aproximación sucesiva.

d) Integrador dual.

13. ¿Qué tipo de ADC es más adecuado para aplicaciones de bajo costo?

a) De rampa.

b) Flash.

c) De aproximación sucesiva.

d) Integrador dual.

14. ¿Qué tipo de ADC es más adecuado para aplicaciones de bajo consumo de energía?

a) De rampa.

b) Flash.

c) De aproximación sucesiva.

d) Integrador dual.

15. ¿Qué tipo de ADC es más adecuado para aplicaciones de alta resolución?

a) De rampa.

b) Flash.

c) De aproximación sucesiva.

d) Integrador dual.

16. ¿Qué tipo de DAC utiliza una red de resistencias de dos valores?

a) De rampa.

b) Escalera R-2R.

c) De aproximación sucesiva.

d) Integrador dual.

17. ¿Qué tipo de DAC es más adecuado para aplicaciones de alta velocidad?

a) De rampa.

b) Escalera R-2R.

c) De aproximación sucesiva.

d) Integrador dual.

Referencias

- [1 Shoptime.com.br, «americanas,» 9 9 2015. [En línea]. Available:
] https://www.americanas.com.br/especial/carnavalr?chave=dk_hm_tp_1_0_liquida-carnaval25. [Último acceso: 2025].
- [2 ar.inspidedpencil.com, «Mercury in glass,» 20 8 2017. [En línea]. Available:
] <https://ar.inspidedpencil.com/pictures-2023/mercury-in-glass-thermometer-diagram>. [Último acceso: 2025].
- [3 Badecol, «www.badecol.com,» Badecol, 24 9 2021. [En línea]. Available:
] <https://www.badecol.com/collections/basculas-analogas>. [Último acceso: 2025].
- [4 articulo.mercadolibre.com.co, «mercado libre,» 24 6 2016. [En línea]. Available:
] https://articulo.mercadolibre.com.co/MCO-612949629-velocimetro-digital-universal-con-retroiluminacion-_JM. [Último acceso: 2025].
- [5 S. NAGARDAS, «snmestore.com,» 16 11 2013. [En línea]. Available:
] <https://snmestore.com/Tfa-Electronic-Thermometer-Digital-W-Probe-Min-Max-p147181679>. [Último acceso: 2025].
- [6 M. libre, «articulo.mercadolibre.com.mx,» 4 7 2019. [En línea]. Available:
] https://articulo.mercadolibre.com.mx/MLM-761114885-bascula-digital-analitica-de-laboratorio-3000g001g-_JM. [Último acceso: 2025].
- [7 A. Bueno, «portaleso.com,» 3 9 2012. [En línea]. Available:
] https://www.portaleso.com/web_digital/digital_indice.html. [Último acceso: 2025].
- [8 D. T. SISTEMAS, «isabelitac.blogspot.com,» 23 1 2017. [En línea]. Available:
] <https://isabelitac.blogspot.com/2012/06/diferencias-entre-computadoras.html>. [Último acceso: 2025].
- [9 e. a. d. acustica, «www.eumus.edu.uy,» 5 7 2013. [En línea]. Available:
] <https://www.eumus.edu.uy/eme/ensenanza/acustica/presentaciones/electro/electro/electro01.html>. [Último acceso: 2025].
- [1 C. A. -. D. M. MAP, «www.bing.com,» 29 11 2009. [En línea]. Available:
0] <https://www.mindomo.com/ru/mindmap/conversion-analogica-digital-87fd128e96ec42ad9fdb5f901ca607c5>. [Último acceso: 2025].
- [1 NGFW, «www.researchgate.net,» 5 2 2015. [En línea]. Available:
1] https://www.researchgate.net/figure/Arithmetic-Logic-Unit-14-The-combinational-logic-circuitry-of-the-74-181-integrated_fig1_352092850. [Último acceso: 2025].

- [1 A. INFORMATICA, «blogspot.com,» 11 1 2012. [En línea]. Available:
2] <https://misterioenlainformatica.blogspot.com/2015/07/ascii.html>. [Último acceso: 2025].
- [1 e. a. U. Character sets, «www.gammon.com.au,» 21 3 2017. [En línea]. Available:
3] <https://www.gammon.com.au/unicode/>. [Último acceso: 2025].
- [1 R. J. Tocci, «Familias lógicas,» de *Sistemas Digitale Principios y Aplicaciones*, México,
4] Pearson, 2007, pp. 488 - 497.
- [1 B. J.L, «electronicaonline.net,» [En línea]. Available:
5] <https://electronicaonline.net/electronica/multiplexor-y-demultiplexor/>. [Último acceso: 9 4 2025].
- [1 L. S. Corporation, «alldatasheet.com,» Electronic Components Datasheet Search, [En
6] línea]. Available: <https://www.alldatasheet.com/datasheet-pdf/view/70193/LATTICE/GAL22V10.html>. [Último acceso: 2025].
- [1 C. L. M. G. E. S. M. Luis Entrena, «ocw.uc3m.es,» 2008. [En línea]. Available:
7] https://ocw.uc3m.es/pluginfile.php/2308/mod_page/content/22/Tema9_dispositivos_logicos_programables.pdf. [Último acceso: 2025].
- [1 I. Artetxe, «addi.ehu.es,» 6 2013. [En línea]. Available:
8] <https://addi.ehu.es/bitstream/handle/10810/10620/TFG.pdf?sequence=1>. [Último acceso: 4 2025].
- [1 Gijon, «slideplayer.es,» Universidad de Oviedo, [En línea]. Available:
9] <https://slideplayer.es/slide/109442/>. [Último acceso: 4 2025].
- [2 B. J.L, «www.bing.com,» 22 5 2021. [En línea]. Available:
0] <https://electronicaonline.net/electronica/circuitos-logicos/>.
- [2 T. Instrument, «www.ti.com,» SNAS548D, 2015. [En línea]. Available:
1] <https://www.ti.com/lit/ds/symlink/lm555.pdf?ts=1743031306782>. [Último acceso: 2025].
- [2 ALLDATASHEET.COM, «www.alldatasheet.com,» 17 10 2002. [En línea]. Available:
2] www.alldatasheet.com/datasheet-pdf/pdf/83230/PHILIPS/ADC0804.html. [Último acceso: 2025].
- [2 C. urbina, «www.bing.com,» 2014. [En línea]. Available:
3] <https://www.bing.com/images/search?view=detailV2&ccid=luW%2BQKSp&id=F30A07C6DBDD08627FDA5443D7D26EE19663F1FD&thid=OIP.luW-QKSpop0jZ6luW4K3GgHaEr&mediaurl=https%3A%2F%2Fth.bing.com%2Fth%2Fid%2FR.22e5be40a4a9a29d2367a96e5b82b71a%3Frik%3D%252ffFjluFu0tdDVA%2>. [Último acceso: 2025].

- [2 ELECTROPREGUNAS.COM, «www.bing.com,» 2024. [En línea]. Available:
- 4] <https://electropreguntas.com/wp-content/uploads/2023/07/como-se-puede-mejorar-la-calidad-de-la-senal-en-cables-de-audio.png>. [Último acceso: 2025].
- [2 +. DIGITAL, «maselectronica18.blogspot.com,» 2021. [En línea]. Available:
- 5] <https://maselectronica18.blogspot.com/2016/12/circuitos-combinacionales.html>. [Último acceso: 2025].
- [2 CybernetiK, «www.cybernetik.com,» 2024. [En línea]. Available:
- 6] https://www.cybernetik.com/case_study/drum-ibc-decanting-system/. [Último acceso: 2025].
- [2 Makerguides, «makerguides.com,» 22 1 2019. [En línea]. Available:
- 7] <https://www.makerguides.com/character-lcd-arduino-tutorial/>. [Último acceso: 2024].
- [2 VISHAY, «WWW.vishay.com,» 9 10 2012. [En línea]. Available:
- 8] <https://www.vishay.com/docs/37314/lcd020n004l.pdf>. [Último acceso: 2024].
- [2 NeoDigiMundo, «www.bing.com,» 5 7 2013. [En línea]. Available:
- 9] <https://neodigimundo.blogspot.com/2011/11/maquina-de-mealy.html>. [Último acceso: 4 2025].