INSTITUTO TECNOLÓGICO SUPERIOR DE ATLIXCO

Organismo Público Descentralizado del Gobierno del Estado de Puebla

SISTEMA DE VISIÓN Y PROCESAMIENTO DE VIDEO EMBEBIDO EN FPGA PARA PROCESOS DE MANUFACTURA DE ALTA VELOCIDAD

OPCIÓN I.

TESIS PROFESIONAL

QUE PARA OBTENER EL TÍTULO DE:

Ingeniero Mecatrónico

P R E S E N T A:
Norma Lucia Grande Rodriguez

ASESOR: **Dra. Mariana Natalia Ibarra Bonilla**

ATLIXCO, PUE. JULIO DE 2019

AGRADECIMIENTOS

Quiero expresar mi gratitud a Dios, quien con su bendición llena siempre mi vida, por guiarme y acompañarme en estos 5 años de mi vida profesional, por permitirme tener experiencias las cuales me han enseñado y ayudado a crecer en todos los ambitos de mi vida, asi como tambien me ha dado la oportunidad de conocer a personas extraordinarias que me ayudaron y ayudan a crecer cada día con sus consejos y experiencias.

Pero sobre todo agradezco con todo mi corazón a toda mi familia por estar siempre presentes, por nunca dejarme caer, por siempre estar conmigo en las buenas y en las malas.

Un agradecimiento especial a mis padres Raymundo Grande y a mi mami Lucia Rodriguez , quien nunca dejo de creer en mi, por que nunca dejo de ayudarme, hasta en la cosa más minima estuvo preocupada por mi carrera y que la pudiera culminar con éxito, por todos sus consejos, por nunca dejarme sola, por ser una mujer luchadora, llena de amor para todos sus hijos y nietos, no sabes lo agradecida que estoy con todas las cosas que has hecho por mi mamá, nunca me cansare en decirte cuanto TE AMO.

A mi hermano y hermanas por ser parte importante de mi vida, por sus conejos, por escucharme cuando lo necesitaba, por sus porras de “Si se puede hermanita la mas pequeñita, Animo nosotros creemos en ti y te queremos mucho ”, y por representar la unidad familiar, los amo mucho.

De igual forma, agradezco a mi tutora de Tesis a la Dra. Mariana Natalia Ibarra Bonilla, quien me motivo a realizar este proyecto , por creer en mi, por sus consejos, correcciones, por haberme guiado, no solo en la elaboración de este trabajo de titulación, sino a lo largo de mi carrera universitara, pero sobre todo agradezco su amistad, que se fue creando en platicas, risas, anécdoras y consejos.

Por ultimo, pero no menos importante agradezco a todos mis profesores, por su esfuerzo, dedicación, paciencia, por su confianza y por siempre darme una mano de ayuda cuando lo necesitaba.

A todos con mucho cariño, respeto, admiracion y amor,

NORMA LUCIA GRANDE RODRIGUEZ

ÍNDICE

INTRODUCCIÓN

La tecnología de montaje superficial, SMT (Surface Mount Technology) es una nueva generación de montaje electrónico, que ha reemplazado la técnica de agujero pasante (*through hole*), en donde el componente atraviesa la placa de circuito impreso, conocida como PCB, del inglés *Printed Circuit Board [1]*. La SMT es el sistema para soldar componentes de montaje superficial, SMD (Surface Mount Component), en una PCB cumpliendo con los requerimientos de tiempos, calidad y costos para su manufactura. Los SMD son componentes micro miniaturizados con o sin terminales, como resistencias, capacitores y circuitos integrados que se soldan directamente en unas zonas conductoras situadas en la superficie de la PCB llamadas huellas (*lands*), sin la necesidad de atravesar la tarjeta para ser insertados. Sin embargo, debido a su reducido tamaño, el ensamblado manual de las piezas se dificulta, por lo que se necesita un proceso de automatización en las líneas de producción, y también se requiere la implementación de técnicas más avanzadas de diseño para que los SMD funcionen adecuadamente, aún en ambientes con altos índices de interferencia electromagnética (EMI).

Actualmente la manufactura de tarjetas electrónicas con la tecnología SMT está altamente automatizada, y a pesar de que el avance tecnológico es muy rápido, en México existe una limitación en cuanto a la manufactura de productos innovadores y funcionales que puedan competir en el mercado internacional, pues siempre se recurre a la maquinaria extranjera. Es por ello que la empresa INTESC implementó una máquina *pick & place,* a partir de la necesidad de diseñar y fabricar las tarjetas de desarrollo basadas en FPGAS y microcontroladores utilizando SMT. Sin embargo, esta máquina requiere perfeccionar el posicionamiento y orientación de los componentes electrónicos. El problema se presenta cuando los circuitos integrados son colocados sobre bandejas alimentadoras que no proporcionan una posición exacta y resulta en un desfase en la posición de los componentes, por lo que es necesario incorporar un sistema de corrección en la posición y así eliminar el desfase.

El presente trabajo presenta el desarrollo de un sistema visión que se incorporará a una máquina *pick & place* que manipula componentes SMT. El sistema de “visión artificial” generará un proceso más autónomo, la cual engloba todas las técnicas y elementos que proporcionan “ojos” a las máquinas, siendo parte esencial en el proceso de manufactura electrónica para mejorar la calidad de los productos ensamblados con la tecnología de montaje superficial. Con ello se pretende mejorar el desarrollo de la tecnología mexicana para comenzar a crear productos de calidad por empresas mexicana que compitan internacionalmente y a su vez ofrecer productos de menor costo y tiempo.

La organización del presente doumento esta dividida en los siguientes capitulos :

Capítulo 1: Presenta la parte del protocolo de investigación que ha dado surgimiento a este trabajo de tesis.

Capítulo 2: Presenta información, relacionada con los conceptos utilizados para realizar este proyecto, así como la información del sistema embebido FPGA, su arquitectura y los fabricantes más importantes a nivel internacional..

Capítulo 3: Describe el software ISE Desing Suit, en el que explica su entorno de diseño para productos de FPGA y la relacionado que tiene con la arquitectura de dichos chips.

Capítulo 4: Se explica el desarrollo de la programación del módulo principal en VHDL que se implementó para hacer trabajar la cámara en la FPGA de la tarjeta ASSERTA y del proceso de orientación en base al algoritmo basado en los segundos momentos de inercia del trabajo de tesis del estudiante Jesús Ángel Aragón Morales, el cual presento su proyecto con el titulo de *“Algoritmo de visión artifical para el cálculo de la orientación de componentes electrónicos aplicados a procesos de manufactura de alta velocidad”[1].*

Capítulo 5: Se expone el analisis general de la traducción del código generado en matlab a codigo VHDL, se explicará el procedimiento realizado para obtener el algoritmo de orientación en FPGA, que será utilizado dentro de este

proyecto, haciendo uso de las herramientas del software Matlab e ISE Design Suite 14.7.

Capitulo 6: Se presentan los resultados finales de la traducción y vinculación del algoritmo de orientación en la tarjeta FPGA.

CAPÍTULO 1. PROTOCOLO DE INVESTIGACIÓN

En el presente capítulo se describe el protocolo de la investigación que ha dado surgimiento a este trabajo de tesis.

1.1. Planteamiento del problema

INTESC Electrónica & Embebidos [2] es una empresa mexicana dedicada al diseño y desarrollo de sistemas embebidos basadas en FPGAs (del inglés, *Field-Programmable Gate Array*) y SoCs (del inglés, *Systems on Chips*), así como a la producción de nueva tecnología. Sin embargo la fabricación de las tarjetas que desarrollan utilizan la tecnología de montaje superficial (SMT, Surface-Mount Technology), puesto que es menos costosa y ahorran básicamente espacio, permitiendo hacer placas reducidas y ligeras que ocupan la cuarta parte de un espacio, reduciendo la longitud de las pistas, permitiendo obtener el manejo de circuitos de alta velocidad, ya que el comportamiento a altas frecuencias es mucho mejor. Actualmente, el proceso de ensamble manual de los componentes es difícil debido a los tamaños reducidos de los componentes electrónicos.

En INTESC, para la fabricación de sus tarjetas de desarrollo, los ingenieros utilizan la tecnología de montaje superficial (SMT, Surface-Mount Technology). Para ensamblar las tarjetas existe una persona encargada del proceso de ensamble de los componentes electrónicos SMT sobre la PCB, lo que ocasiona que su producción sea menor y con mayor costo. Por ello la importancia de desarrollar una máquina *Pick & Place* que permita realizar la colocación de los componentes electrónicos de forma autónoma utilizando un dispositivo programable FPGA, y así no requerirá de un operador para su manejo. Con la máquina se espera un aumento de producción, disminución de costos y la reducción de tiempos de ensamble.

Sin embargo existe un problema importante en la programación de la máquina *Pick & Place* que limita su correcto funcionamiento, la cual consiste en el posicionamiento y orientación de los componentes cuando estos son colocados

sobre la PCB para su ensamble, pues al ser colocados por la máquina sobre la placa, éstos algunas veces son colocados con cierto ángulo de inclinación, lo que provoca que un operador tenga que corregir la posición de los componentes sobre la placa, lo cual ocasiona en la disminución de la producción.

Debido a lo anterior, como solución se ha propuesto incorporar un sistema de visión que sirva como retroalimentación al lazo de control en la máquina, que pueda disminuir el desfase al orden de las micras. Para ello el sistema de visión incorpora un algoritmo de procesamiento de imágenes que es capaz de calcular de forma autónoma el ángulo de rotación de los componentes.

De modo que el cálculo realizado para su solución fue descrito en un lenguaje de alto nivel, y para este caso se necesita de un lenguaje HDL (Hardware Description Language), el cual manejan los dispositivos FPGAs, puesto que este será utilizado para el procesamiento de imágenes a alta velocidad y cálculo del ángulo de rotación de los componentes. La capacidad de procesamiento en paralelo de un FPGA lo hace más eficiente en todas las funciones a manejar en la maquina *Pick & Place* de manera automática.

1.2. Objetivos

A continuación se describen los objetivos que se persiguen con la realización del presente proyecto.

1.2.1. Objetivo General

Traducir un algoritmo de visión realizado en Matlab a un lenguaje de descripción de hardware HDL para su implementación en una tarjeta FPGA aplicado a una máquina Pick & Place.

1.2.2. Objetivos Específicos

- Revisar la bibliografía existente acerca de las implentaciones de los algoritmos de procesamiento de imagen en FPGA.

- Calcular el ángulo de rotación de los componentes, implementado en la máquina Pick & Place para soldar y posicionar los componentes electrónicos de manera correcta sobre las placas (PCB), con respecto a una cámara RGB 0V7670.

- Familiarizarse con el software ISE Design Suite, para desarrollar el procesamiento y manipulación en el FPGA.

- Incorporar el sistema en una única unidad y montar a la máquina.

- Realizar pruebas de validación.

1.3. Alcances y Limitaciones

Una vez descritos los objetivos, pasamos a describir los alcances y las limitaciones del presente proyecto, con la finalidad de dar una idea general del trabajo que va a realizarse.

1.3.1. Alcances

Los alcances del trabajo son los siguientes:

- Generar el código bajo las condiciones que exige el lenguaje VHDL.
- Incorporar el sistema de visión a la máquina y se espera concluir el proceso de automatización.

1.3.2. Limitaciones

- La solución solo se hará para un FPGA y no otros dispositivos.

1.4. Justificación

La mayoría de los sistemas electrónicos de hoy día utilizan, en cierta medida, componentes de tecnología de ensamblaje superficial (SMT). Sin la SMT, la reducción en tamaño de los componentes y productos sería extremadamente complicada. Sin embargo el proceso de ensamble SMT requiere trabajar a alta velocidad y precisión, es por ello que para el sistema de visión y procesamiento de

video propuesto en la maquina *Pick & Place* deben implementarse en una plataforma computacional capaz de procesar imágenes con alto rendimiento, como lo serían los procesadores de señales digitales (DSP, Digital Signal Processors).

Los DSP están basados en un procesador o microprocesado ampliamente y son usados en aplicaciones de alto rendimiento y procesamiento de una gran cantidad de datos, no obstante estos dispositivos dependen de un conjunto fijo de instrucciones para su ejecución secuencial o en serie, cada una de ellas está asociada a un hardware conectado de forma fija, y el programador no puede usar más instrucciones que las definidas y configuradas por el fabricante. Por otra parte, la FPGA, al no tener una arquitectura definida, no tiene nada conectado de forma fija, ya que está compuesta por una red de conexiones que el usuario puede unir o romper determinando el comportamiento lógico del dispositivo, usando lenguajes de descripción de hardware, como VHDL, Verilog, entre otros. Por otro lado, cuando se describe la lógica digital para una FPGA, el circuito resultante contendrá múltiples señales que variarán al mismo tiempo, en una especie de ejecución paralela, proporcionando mayor velocidad por lo que la capacidad de procesamiento de un FPGA es más eficiente y económico que un DSP.

CAPÍTULO 2. MARCO TEÓRICO

En el presente capítulo se presenta información, relacionada con los conceptos utilizados para realizar este proyecto.

2.1. Tecnología SMT

La tecnología de montaje superficial o SMT fue desarrollada en los años 1960 y se volvió ampliamente utilizada a fines en el año de 1980. La labor principal en el desarrollo de esta tecnología fue gracias a IBM (“International Business Machines Corporation”) y Siemens [3]. La estructura de los componentes fue rediseñada para que tuvieran pequeños contactos metálicos que permitiesen el montaje directo sobre la superficie del circuito impreso. Es por esto que los componentes SMT se construyen pequeños y livianos.

La tecnología SMT se utiliza ampliamente en la industria electrónica, debido al incremento de tecnologías que permiten reducir cada día más el tamaño y peso de los componentes electrónicos. La evolución del mercado y la inclinación de los consumidores hacia productos de menores tamaños y peso, causando que este tipo de industria creciera y se expandiera; componentes tan pequeños en su dimensión como 0,5 milímetros son montados por medio de este tipo de tecnología, ver Figura 2.1.

Figura 2.1: Montaje Superficial [3].

2.2. Circuitos Integrados (CI)

El primer Circuito Integrado fue desarrollado en 1958 por el Ingeniero Jack St. Clair Kilby, justo meses después de haber sido contratado por la firma Texas Instruments y fue posible gracias a descubrimientos experimentales que mostraban que artefactos semiconductores podían realizar las funciones de los tubos de vacío, así como a los avances científicos de la fabricación de semiconductores a mediados del siglo XX. [4]

2.2.1. ¿Qué son los Circuitos Integrados?

El circuito Integrado (IC), es una pastilla o chip muy delgado en el que se encuentran una cantidad enorme de dispositivos microelectrónicos, principalmente diodos y transistores, además de componentes pasivos como resistencias o condensadores, ver Figura 2.2.

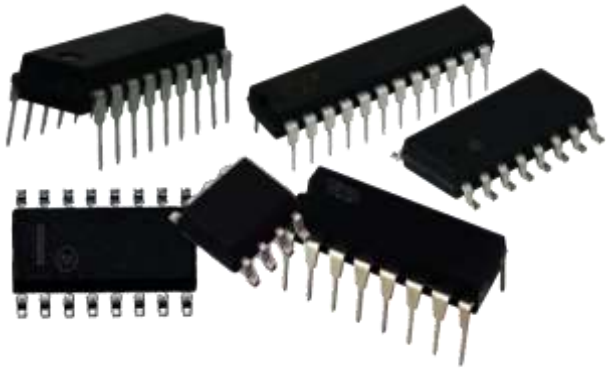


Figura 2. 2: Circuitos Integrados.

Existen dos ventajas principales de los circuitos integrados sobre los circuitos convencionales: costo y rendimiento [5]. El bajo costo es debido a que los chips, con todos sus componentes, son fabricados en un solo proceso por fotolitografía y no construidos por transistores de a uno por vez. Existen dos clasificaciones fundamentales de circuitos integrados:

- **Circuitos integrados digitales:** Pueden ser desde puertas lógicas básicas hasta los más complicados microprocesadores. Éstos son diseñados y fabricados para cumplir una función específica dentro de un sistema. En general, la fabricación de los circuitos integrados es compleja ya que tienen una alta integración de componentes en un espacio muy reducido de forma que llegan a ser microscópicos. Sin embargo, permiten grandes simplificaciones con respecto a los antiguos circuitos, además de un montaje más rápido.

- **Circuitos integrados analógicos:** Pueden constar desde simples transistores encapsulados juntos, sin unión entre ellos, hasta dispositivos completos como amplificadores, osciladores o incluso receptores de radio completos.

2.2.2. Componentes de dispositivos de montaje superficial

Los componentes diseñados para SMT reciben el nombre de SMD ("dispositivos de montaje superficial"), ver Figura 2.3. Los SMD son de dimensiones reducidas y pueden ser empaquetados en recipientes especiales para una manipulación más eficiente a la hora de su colocación. Generalmente los recipientes suelen ser carretes de papel en donde se acomodan los componentes, para su distribución. Los SMD se diseñan de forma que permiten ser soldados superficialmente en la placa de circuito impreso. Su pines o patas eléctricas de conexión, se construyen en formas geométricas simples como rectángulos, círculos o una combinación de ambos. Estas formas permiten generar áreas de soldadura planas, lo que a su vez provoca que las conexiones eléctricas sobre las placas se diseñen con estas formas. Los SMD son colocados sobre las placas de manera que ambas áreas se toquen entre si superficialmente y la soladura se adhiera a ambas partes. Esta técnica de soldadura y montaje da el nombre a la tecnología SMT [6].

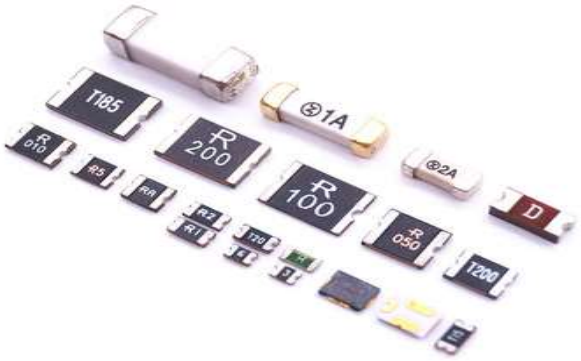


Figura 2. 3: Dispositivos de montaje superficial.

2.2.3. Placas de Circuito Impreso

Las placas de circuito impreso o PCB, son placas especiales que proporcionan el cableado eléctrico para la interconexión de los SMD, además de un soporte físico de los mismos, Figura 2.4. El Circuito impreso, PCB o tarjeta electrónica es muy usada cuando se trabaja en diseño electrónico, mecatrónico, eléctrico, también en investigación, producción y manufactura, desarrollo de productos innovadores [7]. Actualmente todos los productos (electrónicos o no) tienen por dentro de sí tarjetas electrónicas con diferentes formas, características, tamaños, componentes, colores.

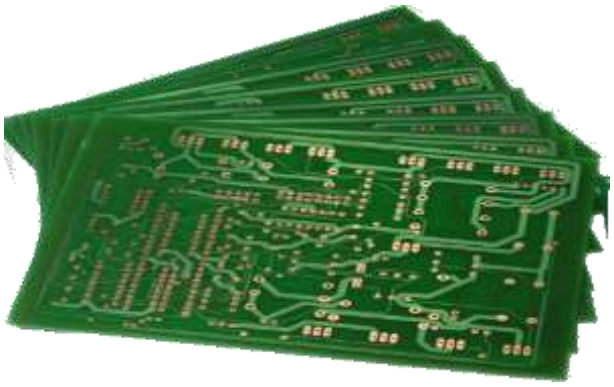


Figura 2. 4: Placa de circuito impreso.

Las PCB se fabrican con diferentes materiales como por ejemplo baquelita, fibra de vidrio, teflón o una combinación de estos materiales; así mismo emplea metales para la interconexión de los componentes como: el cobre, oro, plata, platino, plomo y estaño. Para proteger la superficie de las placas y evitar la oxidación de los metales, se emplean resinas dieléctricas a prueba de agua que se aplican sobre la superficie de las placas en forma de pintura de diferentes colores [8]. El diseño de una PCB está orientado a generar la menor emisión electromagnética para disminuir la interferencia a otros dispositivos electrónicos. Las interconexiones eléctricas en una PCB se diseñan en capas o niveles, de manera que una capa puede emplearse para interconectar todas las líneas de voltaje; otro nivel para interconectar todas las líneas de la terminal común o GND, y por último niveles adicionales pueden emplearse para interconectar las líneas de comunicación entre los SMD. [9] Generalmente se diseñan placas de doble capa o de dos niveles, debido a los costos de producción de las mismas, ya que a medida que se incrementa los niveles de interconexión, lo mismo ocurre con el precio de fabricación. Las líneas o cables eléctricos que conectan a los SMD siempre son terminados en áreas de interconexión llamados pads. Los pads proporcionan el área metálica sobre la cual se soldán los SMD.

2.2.4. Equipo de montaje superficial

En la industria se emplean equipos de montaje superficial o SME para realizar las tareas de montaje de los SMD sobre las PCB. Estos equipos realizan tareas repetitivas de manera rápida, eficiente e incrementa la producción de forma drástica, en comparación con un operador humano. Un operador humano podría producir una placa completa, de unos 200 componentes en alrededor de 2hrs, mientras que una SME podría producir hasta 100 PCBs de las mismas características en el mismo tiempo [9]. Es por ello que, en la industria de fabricación de circuitos impresos, las máquinas SMD son ampliamente empleadas. Y resulta evidente que los costos de producción utilizando equipos decrecen considerablemente.

Un laboratorio de manufactura de SMT, involucra diferentes tipos de equipos, sin embargo, en esencia solos se requieren de 2 equipos para cubrir la tarea de montaje. Los 2 equipos más importantes son:

- **El equipo PPE *("Pick and Place Equipment")***, del idioma ingles se traduce como equipo de recoger y colocar, como se muestra en la Figura 2.5; son sistemas robóticos utilizados para colocar los componentes de montaje superficial en una tablilla electrónica (PCB) [10]. Se utilizan por sus velocidades altas y su extrema precisión al momento de colocar los componentes electrónicos tales como circuitos integrados, capacitores, resistencias, entre otros en las PCBs las cuales, a su vez, son parte fundamental de productos de consumo, computadoras o sistemas médicos, automotrices, militates, de telecomunicaciones o industriales.

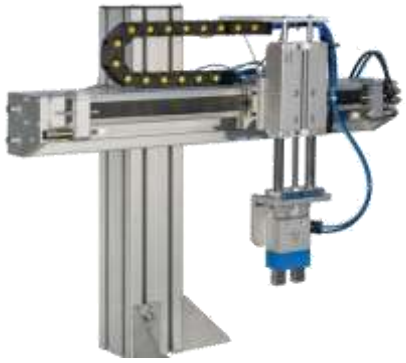


Figura 2. 5: Maquina pick and place.

- **El equipo horno de reflujo**, como se muestra en la Figura 2.6, la cual calienta la PCB junto con los SMD en ella, a una temperatura suficientemente alta como para derretir la soldadura de la PCB y soldar los componentes sobre la placa. Sin embargo Para el perfilado del horno que no es más que la medición de la temperatura de cada una de las zonas del mismo es necesario usar software y hardware especializados los más populares son el kick y el datapaq.

Figura 2. 6: Maquina horno de reflujo de SMT.

Por otro lado, el PPE se encarga de tomar los SMD de diferentes carretes alimentadores y los coloca en su posición final correcta en la PCB. Dependiendo del tipo de PCB que se ensamble, los PPE pueden variar en su configuración mecánica y tipo de manipulador mecánico. Existen equipos simples que solo colocan componentes pasivos como resistencias, capacitores, bobinas y que no poseen una retroalimentación con respecto a la posición final de los componentes. Equipos de este tipo cuestan alrededor de unos $2000 USD a $3000 USD [10]. Cuando el PPE incluye un sistema más complejo como la capacidad para colocación de circuitos integrados y retroalimentación por medio de procesamiento de imágenes, el costo puede rondar de entre $4100 USD a $20000 USD o incluso más elevado.

Los sistemas de visión por computador son una parte importante en la colocación de componentes en la tecnología SMT. Existen sistemas de procesamiento de imágenes con la capacidad de reconocimiento de objetos, y formas. Estos se emplean para determinar la posición inicial y final del componente que un PPE coloca en el PCB.

2.3. Módulo de la cámara VGA OV7670

Como elemento de captación de imagen se ha utilizado un sensor CMOS de bajo coste, concretamente el OV7670, fabricado por Omnivisión [11]. Es un SoC (sistema en chip) por lo que es capaz de realizar procesamiento de imágenes, ver Figura 2.7.

Figura 2. 7: Cámara VGA OV7670.

Es capaz de proveer imágenes en distintos formatos (YUC,RGB,GRB y Raw RGB) y en distintas resoluciones (VGA, QVGA, CIF,QCIF). Puede obtener un máximo de 30 imágenes por segundo en VGA con control completo del usuario respecto a calidad de imagen, formato y salida de datos. Todas las funciones del procesador de imagen (control de exposición, gamma, balance de blancos, saturación de color, control de tono y más) son programables a través de una comunicación SCCB (compatible con comunicación I2C).

El chip incorporado también es capaz de mejorar la calidad de la imagen reduciendo o eliminando ruido, defectos, manchas, brillos, etc. para producir una imagen limpia y estable [11]. Cuenta con un buffer FIFO AL422B de 380 KB de memoria, que permite trabajar de manera más sencilla, pues los datos se almacenan en el buffer temporalmente y así disminuyen la carga computacional del controlador, el diagrama de bloques funcional del módulo se muestra en la Figura 2.8.

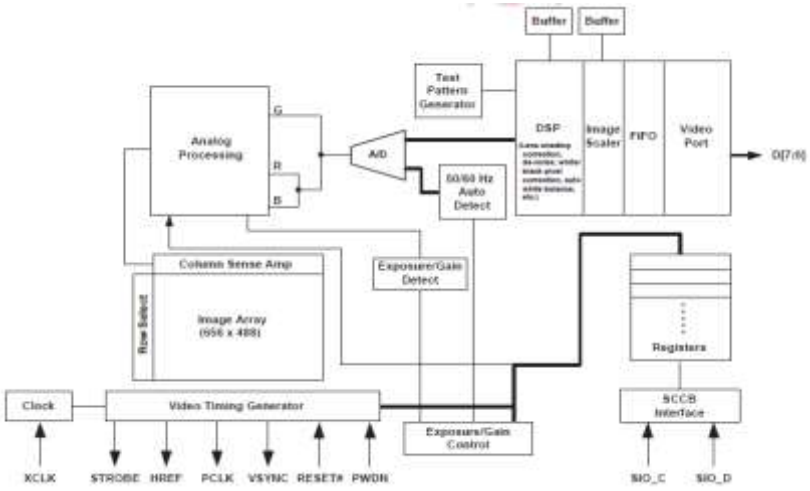


Figura 2. 8: Diagrama de bloques funcional del módulo.

Sus características son:

- Voltaje de Operación: 3.3V DC.
- Consumo de energía: 60mW/15fpsVGAYUV.
- Corriente Sleep: <20uA.
- Cámara: OV7670.
- Buffer: AL422B.
- Transmisión de datos en paralelo (8 bits).
- Interface de control estándar SCCB, compatible con I2C.
- Necesita una frecuencia de 8MHz.
- Lente óptico de 1/6”.
- Ángulo de visión (FOV): 25°.
- Resolución: 640x480 VGA.
- Sensibilidad: 1.3V / (Lux-sec).
- Ratio Señal-Ruido (SNR); 46 dB.
- Rango Dinámico: 52 dB.
- Modo de vista: Progresivo.
- Lente de alta calidad F1.8/6 mm.
- Formatos de Salida: Raw RGB (8 digit). RGB (GBR 4:2:2, RGB 565/555/444), YUV (4:2:2) y YCbCr (4:2:2).

- Máximo refresco de cuadro: 30 fps VGA.
- Exposición electrónica: 1 a 510 filas.
- Covertura de pixel: 3.6 um x 3.6 um [12].

2.3.1. Configuración de los pines del módulo VGA OV7670

El módulo cuenta con 22 pines, los cuales incluyen los de la cámara OV7670 y el FIFO, ambos alimentados con una tensión de 3.3V, ver Figura 2.9 [12].

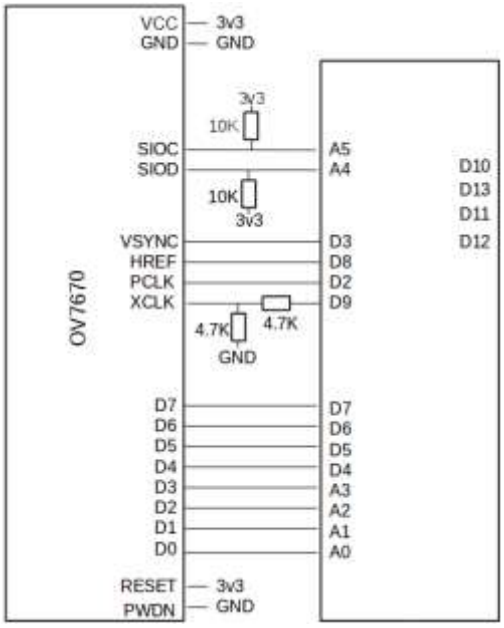


Figura 2. 9: Pines del módulo OV7670.

Los pines más importantes se mencionan a continuación:

- ***VCC****:* Pin de alimentación a 3.3V.
- ***GND:*** Pin de tierra 0V.
- ***SIO_C****:* Señal del reloj de la comunicación SCCB.
- ***SIO_D****:* Señal de datos de la comunicación SCCB.
- ***VSYNC:*** Señal de sincronización vertical (indica el final de salida de información del “frame” anterior y el comienzo del siguiente).
- ***HREF***: Señal de sincronización horizontal (indica el final de salida de información de la línea anterior y el comienzo de la siguiente).

- ***XCLK**:* Señal de reloj de entrada (indica una base de tiempos al módulo).
- ***PCLK:*** Señal de reloj de salida del pixel (indica el comienzo de salida de la información del siguiente pixel).
- ***D [7-0]:*** 8 bits de datos (representan la máxima cantidad de información que se extrae en cada instante, 1 Byte).
- ***Reset:*** Señal de reset (restablece todos los registros a sus valores por defecto y reinicia el módulo).

2.3.2. Funcionamiento del módulo VGA OV7670

Para implementar el módulo de captura de imagen, se tiene que saber cómo transmite el sensor las secuencias de bits que componen la imagen con cada espacio de color y las señales de sincronía que emplea. Es por ello que antes de centrarnos en cada espacio de color, conviene hacer un breve repaso de las distintas señales de sincronización de las que dispone el sensor. Éste transmite un flujo de bits controlado por las señales HREF (referencia horizontal), VSYNC (sincronía vertical) y PCLK (reloj de píxel):

- ***HREF:*** Señal de referencia horizontal. Esta señal indica cuando acaba una línea de píxeles poniéndose a nivel bajo, de forma que el sensor está enviando datos válidos siempre que esta señal se encuentre a nivel alto.
- ***VSYNC:*** Señal de sincronía vertical. Indica cuando termina un fotograma y comienza el siguiente. Cambia a nivel alto cuando la imagen ha terminado de transmitirse y vuelva a nivel bajo cuando comienza el siguiente.
- ***PCLK:*** Reloj de píxel. Este reloj indica cuando leer el dato de los pines de salida. El dato (8 bits) está listo para leerse en cada flanco ascendente.

El diagrama de tiempos que se presenta en la Figura 2.10, muestra las señales de sincronía que emplea el funcionamiento de las salidas digitales más importantes del módulo de la cámara

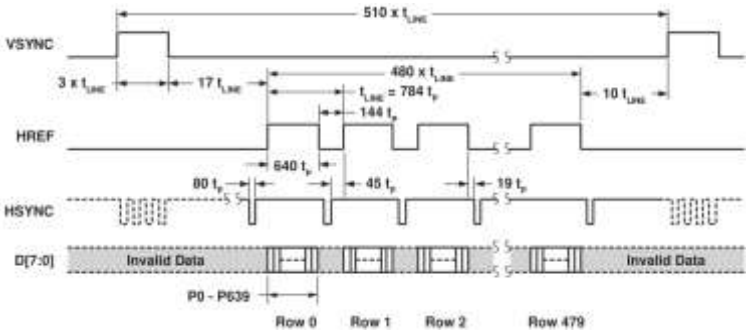


Figura 2. 10: Diagrama de temporización de los Frame VGA.

Se puede observar que la señal de VSYNC indica el comienzo de una imagen (frame) cuando se pone a estado Alto (1), éste dura un corto tiempo para después volver a su estado inicial Bajo (0) en espera de indicar el comienzo de la siguiente imagen. Cuando HREF se pone a 1 inicia la transferencia de información de una fila, esto continua hasta que cambia al estado 0, lo que indica que ya finalizó la transferencia de información de toda la fila [13]. Éste proceso se repite hasta terminar todas las filas, por ejemplo, en la resolución de 640x480 dicho proceso se repetirá 480 veces para a completar 1 frame. Mientras HREF permanezca en 0, toda la información que sale por los pines D [7-0] no es válida.

Es por ello que los cambios se producen por defecto por flanco de bajada de la señal de reloj de salida (PCLK), por cada ciclo de dicho reloj se obtiene 1 Byte de información, por lo tanto, por cada pixel se necesitan 2 Bytes, como se muestra en la Figura 2.11.

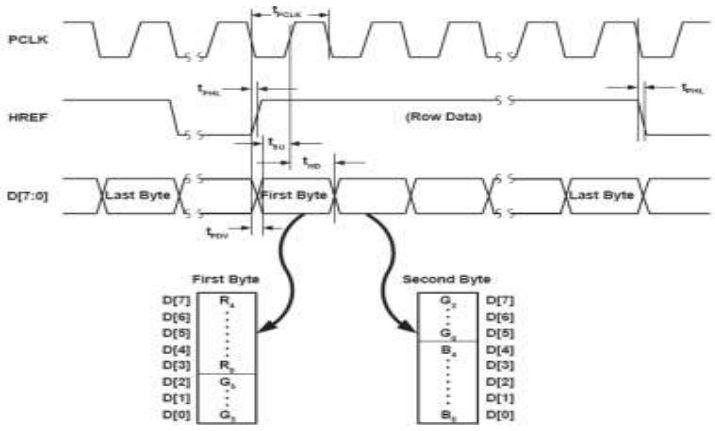


Figura 2. 11: Diagrama de temporización del formato de salida RGB 565.

2.3.3. Configuración del módulo VGA OV7670

Para configurar los distintos modos de operación de la cámara se utilizan los registros, que son dispositivos digitales donde se almacena información de manera temporal. Los registros están constituidos por flip-flops o biestables, cada uno de los cuales almacena un bit con el valor 0 o 1. Lo más común es que los registros estén constituidos por 8 biestables, es decir, almacenan 8 bits, este es el caso de los registros de la cámara OV7670 [14]. Los registros de la cámara nos permiten configurar opciones como la frecuencia de reloj de salida, el formato de colores de los píxeles, la resolución de la imagen y el modo en que actúan las distintas señales.

Cabe mencionar que todos los registros tienen un valor por defecto, de forma que la cámara ya viene configurada para funcionar de un determinado modo al encenderla. La configuración del resto de registros se puede ver en el datasheet del fabricante.

2.4. Tarjeta FPGA ASSERTA

Para realizar la programación del módulo OV7670 y hacer que la cámara funcione, se empleó la tarjeta de desarrollo ASSERTA. La tarjeta de desarrollo ASSERTA, diseñada y manufacturada por la empresa mexicana INTESC, ha sido diseñada para satisfacer las necesidades de estudiantes, investigadores y profesionistas que trabajan con FPGAs, la tarjeta cuenta con numerosos recursos lo que la hace versátil para desarrollos de aplicaciones embebidas, además la hacen ideal para hacer adquisición de datos.

El núcleo de *ASSERTA* contiene un FPGA Spartan 6 XC6SLX16 que le permite diseñar sistemas de procesamiento en paralelo, cuenta con 12 convertidores analógico/digital (ADC), 4 convertidores digital/analógico (DAC), un compartimiento para integrar una pantalla TFT de 4.3” con opción de touchscreen capacitivo lo que le permite ser usada como un sistema autónomo de adquisición,

3 fuentes internas para el FPGA, una fuente externa de alimentación y un oscilador de 50 MHz que es la principal fuente de reloj, además de la circuitería [15]. En la Figura 2.12 se puede observar la tarjeta ASSERTA.

Figura 2. 12: Tarjeta FPGA ASSERTA.

2.4.1. Características de la Tarjeta de desarrollo FPGA ASSERTA

La tarjeta ASSERTA es un sistema de desarrollo que contiene elementos que la hacen ideal para hacer adquisición de datos. Es por ello que es de gran importancia mencionar las características de esta tarjeta.

- FPGA Familia Spartan 6 modelo XC6SLX16 Empaquetado 2FTG256
- 576 kb de Block RAM
- Oscilador de 50 MHz
- Convertidor USB-RS232 (FTDI FT2232HL)
- Memoria Flash 64 Mb
- 16 pines de entrada/salida digital (FPGA)
- Puerto para TFT de 4,3”
- ADC modelo AD7265, contiene 2 ADC de 6 canales cada uno
- DAC modelo AD5684, contiene 1 DAC de 4 canales
- 9 LEDs
- 8 Switches
- 3 push button
- Interruptor de encendido o apagado
- Fuente Principal: 5V desde alimentación externa

- Un Transceiver USB 2.0 de Cypress CYC68013A
- 17 Conectores SMA: 6 para ADC, 4 para DAc y 1 para voltaje de salida con selector de 5V o 3.3V
- 1 jumper para habilitar o deshabilitar fuente de 19V
- 1 jumper para programar Transceiver desde memoria EEPROM
- 5 jumpers para configurar ADC y DAC
- 1 puerto de 3 pines para seleccionar voltaje en VAN utilizando un jumper

La siguiente Figura 2.13 muestra los diferentes dispositivos contenidos en ASSERTA

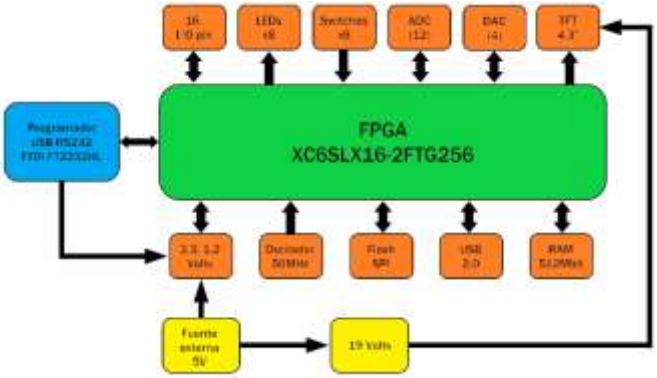


Figura 2. 13: Diagrama de bloques de la tarjeta ASSERTA.

2.5. Sistemas embebidos

La definición de “sistema embebido” es muy diversa y muy difícil de precisar. Debido a la constante evolución en los avances tecnológicos y la disminución en el costo para implementar diversos componentes hardware y software [15]. Las siguientes son algunas de las descripciones más comunes de un sistema embebido.

- Un sistema embebido está más limitado funcionalmente en hardware y/o software que una computadora personal (PC). Posee limitaciones de hardware, como en rendimiento del procesador, consumo de potencia, memoria, funcionalidad de hardware, entre otros. En software las limitaciones son relativas, cuentan con aplicaciones a escala reducida, no

poseen un sistema operativo (OS) ó es limitado. Actualmente ésta definición es parcialmente cierta, tanto hardware y software encontrados en los ordenadores de ayer y hoy han sido construidos en los más complejos diseños de sistemas embebidos.

- Un sistema embebido en un sistema computacional con alta calidad y requisitos de fiabilidad como otros tipos de sistemas computacionales. Algunas familias de dispositivos embebidos poseen una alta calidad y requisitos de fiabilidad.

En el diseño y desarrollo de un sistema embebido es necesario considerar optimizar las soluciones en hardware y software, así como reducir la memoria, hacer uso eficiente de las baterías y sobre todo que regularmente están asociados con aplicaciones de tiempo real, utilizando los sistemas de comunicación disponibles actualmente y tratando de ofrecer dispositivos autónomos e inteligentes.

Un sistema embebido tiene tres componentes principales:

- Hardware.
- Un software primario o aplicación principal. Este software o aplicación lleva a cabo una tarea en particular, o en algunas ocasiones una serie de tareas.
- Un sistema operativo que permite supervisar la(s) aplicación(es), además de proveer los mecanismos para la ejecución de procesos. En muchos sistemas embebidos es requerido que el sistema operativo posea características de tiempo real.

En el diseño de circuitos electrónicos industriales, se tiene que considerar varios criterios. Así como el costo, el consumo de energía (esencial en caso de los sistemas embebidos), el rendimiento de las aplicaciones y, sobre todo, compatibilidad para integración con hardware elegido [16].

2.5.1. Arquitectura de un sistema embebido

La arquitectura de un sistema embebido es una abstracción de un hardware embebido, siendo una generalización del sistema pues no muestra detalles e información, tal como el código fuente de un software ó el diseño de un hardware electrónico.

En el nivel de arquitectura, los componentes hardware y software en un sistema embebido están representados como algunos elementos de interacción. Los elementos son representaciones de hardware y/o software, cuyo detalle de implementación ha sido solo abstraída, dejando solamente información del comportamiento e inter-relación. Los elementos de la arquitectura pueden estar integrados dentro del mismo dispositivo embebido ó externamente interactuando con los elementos internos.

2.5.2. Modelo de un sistema embebido

El modelo presente cuenta con una arquitectura de alto nivel, introduce la mayoría de elementos localizados dentro de un sistema embebido, ver Figura 2.14. Todos los modelos tienen al menos una capa, hardware layer, o todas las capas (hardware, system software and application software layers).

La capa Hardware tiene la mayoría de componentes físicos. Las otras capas contiene el software el cuál es procesado por el sistema embebido [17]. El modelo de un sistema Embebido se puede asociar a un modelo universal, como es el, Open Systems Interconnection (OSI) reference model. Los requerimientos de un sistema pueden ser agrupados en un modelo OSI, el cual fue creado en 1980 por la International Organization for Standardization (ISO).

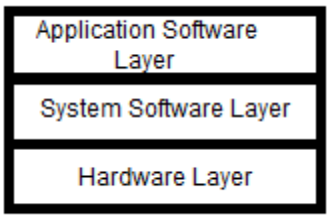


Figura 2. 14: Modelo de sistema embebido.

2.6. Control Visual

El control visual se refiere al empleo de un sistema de visión con el fin de controlar el movimiento de un robot. Los datos necesarios para lograr dicho fin se adquieren desde una cámara que se fija directamente en un robot manipulador o móvil. Para ello, se utiliza la información captada de una escena por una o más cámaras conectadas al sistema de visión por computador, con el fin de controlar la localización del extremo del robot con relación a la pieza de trabajo [18]. Esta realimentación visual en tiempo real proporciona interesantes ventajas al sistema robótico. Una de ellas, quizás la más interesante, es la flexibilidad que se gana, permitiendo una mayor capacidad de interacción con el entorno, de modo que se dota al robot de la capacidad de reaccionar ante imprevistos. De este modo se consigue realizar tareas en las que la orientación y las posiciones de las piezas de trabajo no están predefinidas.

Además, se obtienen movimientos de mayor precisión, puesto que el sistema de visión puede estar implantado en el extremo del robot, lugar en el que se realiza la acción. Por otra parte, se obtiene un incremento en velocidad de respuesta, ya que se tiene una imagen cada cierto intervalo corto de tiempo, permitiendo detectar problemas a gran velocidad [18]. Según dónde se encuentre situado el sistema de visión, los sistemas de control visual se clasifican como: ***eye-in-hand*** (cuando el sistema de adquisición de imágenes se encuentra el extremo final del robot), o como ***eye-to-hand*** (cuando el sistema de adquisición se encuentra montado en una referencia externa al robot de manera que se puedan visualizar los movimientos del robot, y así obtener la información necesaria para realimentar el sistema de control), como se muestra en la Figura 2.15. En este proyecto la

cámara se encuentra unida al extremo del robot, y los movimientos del robot se traducen en movimientos de la cámara.

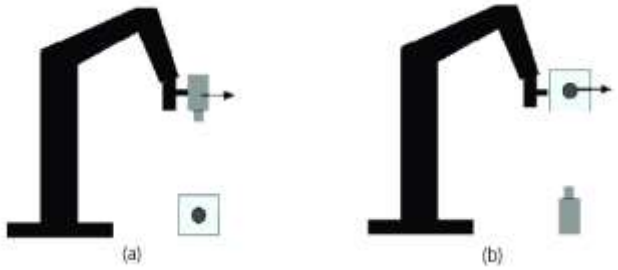


Figura 2. 15: Esquemas de la configuración de la cámara: a) *Eye in hand* y b) *Eye to hand.*

2.6.1. Tipos de control visual

El control visual incluye variados métodos de aplicación, cada uno aplicable a un tipo diferente de problemas de gestión.

2.6.1.1. Control visual basado en posición

En control visual, el basado en posición trata de obtener las coordenadas del extremo del robot y las del objeto con el que se desea interactuar a través del sistema de visión. Para ello es necesario un modelo geométrico del entorno. A partir del error, dado por la diferencia entre la posición deseada y la posición obtenida del objeto en el procesamiento visual, se realimenta el bucle de control para llegar a la posición deseada, ver Figura 2.16. La principal desventaja de dicho tipo de control, es la ineficacia en entornos no estructurados. Por otra parte, es necesario el conocimiento preciso del modelo cinemático y geométrico del robot, puesto que en caso de encontrar errores en dichos modelos, se obtendrían graves errores en el posicionamiento. Otra desventaja es la importancia del correcto funcionamiento de calibración de las cámaras presentes, ya que a partir de ellas se obtiene la posición del objeto respecto al sistema de referencia de la cámara.

De modo que los movimientos que realice el robot se verán reflejados en dicho sistema de adquisición, para así tomar las decisiones oportunas para el guiado de

dicho robot según la evolución de las características observadas en la imagen [19].

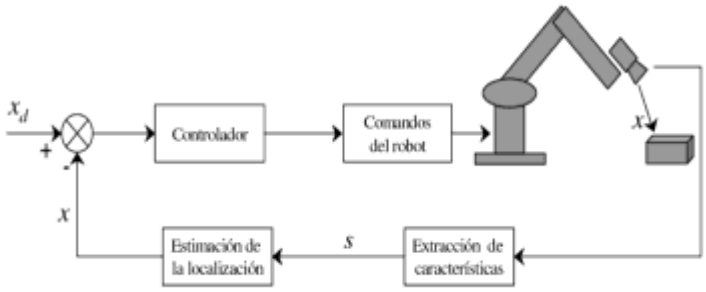


Figura 2. 16: Modelo de control visual basado en posición.

2.6.1.2. Control visual basado en imagen

En cuanto a los sistemas basados en imagen, el control visual se realiza directamente a partir de las características extraídas por el sistema de visión, que representan la proyección del objetivo a alcanzar en el plano imagen. De forma que la entrada al regulador será una comparación entre las características observadas y las deseadas o de referencia, ver Figura 2.17.

El control visual basado en imagen nos permite realizar el guiado de robots donde la ley de control se expresa en términos de características en las imágenes. Es decir, se minimiza el error entre las características de la imagen medidas y las deseadas. De esta forma, los sistemas de control visual basados en imagen implican la determinación de una función de error, e, la cual valdrá cero cuando la tarea se haya desarrollado correctamente y haya llegado a su fin. En dicho momento, las características observadas en la imagen, se corresponderán con las características deseadas. De modo que la acción de control se calcula en el espacio 2-D de la imagen.

En una aplicación típica de control visual en la que el robot debe posicionarse a cierta distancia respecto a un objeto del espacio de trabajo. De forma que, el robot describirá una determinada trayectoria a lo largo de la cual las imágenes captadas por el sistema de visión se irán modificando progresivamente. Es el controlador el

que se encargará de ejecutar las acciones oportunas sobre el robot, de manera que las características observadas en las imágenes se vayan aproximando progresivamente a las deseadas, es decir, a la entrada de referencia del bucle de control. Generalmente, las características de imagen que se utilizan en control visual son formas geométricas elementales (como pueden ser puntos característicos o esquinas) que permiten reconocer en una imagen la proyección de un objeto.

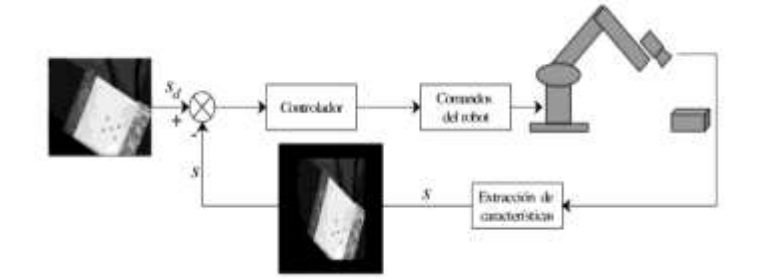


Figura 2. 17: Modelo de control basado en imagen.

2.6.1.3. Control visual indirecto

Los sistemas de control visual clásicos parten de la información visual obtenida por la cámara situada en el extremo del robot (configuración eye-in-hand) para calcular en cada iteración una nueva posición del robot que acabe guiándolo hacia la posición final deseada. Para ello, el controlador obtiene entre iteración e iteración una nueva velocidad de la cámara que minimiza el error calculado como la resta entre la posición de las características visuales actuales medidas en el plano imagen y la posición de las características visuales deseadas. Al resultar conocida la posición relativa entre la cámara y el extremo del robot, es posible transformar las velocidades de la cámara proporcionadas por el controlador a velocidades del extremo del robot. Ésta es la base del control visual basado en imagen con configuración eye-in-hand. Dado que la acción de control del sistema es una velocidad del extremo, es el controlador interno del robot el que debe calcular qué pares articulares se deben aplicar para que el extremo del robot se desplace realmente a esa velocidad calculada.

2.6.1.4. Control visual directo

En el control visual directo se calcula y envía los pares y fuerzas articulares que deben guiar al robot hacia la posición requerida. El resultado es un control mucho más rápido y preciso que si se utiliza un control visual indirecto clásico. El principal inconveniente de estos controladores es que se debe trabajar directamente con los parámetros dinámicos no lineales del sistema robótico [19]. Cuando se diseña un robot, se conocen de forma precisa sus parámetros dinámicos (masas, localización del centro de gravedad, términos de inercia y parámetros de fricción).

2.7. Tecnología FPGA

Los FPGA (siglas del inglés "*Field Programmable Gate Arrays*"), son dispositivos semiconductores que se basan en torno a una matriz de bloques lógicos configurables (CLBs), conectados a través de interconexiones programables. Un FPGA, es una tarjeta que según como se configure, puede realizar cualquier circuito digital, ver Figura 2.18. Los componentes lógicos programables pueden ser programados para duplicar la funcionalidad de puertas lógicas básicas tales como AND, OR, XOR, NOT o funciones combinacionales más complejas tales como decodificadores o simples funciones matemáticas [20].

Figura 2. 18: FPGA SPARTAN-6.

En los últimos años la utilización de los dispositivos FPGA's (Field Programmable Gate Array) para la realización de procesado digital ha tomado gran relevancia, reemplazando cada vez más a los ASIC's y a los DSP's que requieren gran

cómputo de datos, así como, el operador FFT, filtros FIR, IIR. Las FPGA tienen la ventaja frente a circuitos comerciales preprogramados en que no tienen especificados los tiempos que tarda cada instrucción, es decir, es posible realizar un diseño estableciendo la frecuencia de reloj adaptada al circuito.

Son los dispositivos programables por el usuario de aplicación más general. Un FPGA puede ser reprogramado para requisitos de las aplicaciones o funcionalidades deseadas después de la fabricación. Esta característica distingue a los FPGAs de circuitos integrados de aplicaciones específicas (ASIC), que se fabrican a medida para las tareas de diseño específicos. Una jerarquía de interconexiones programables permite a los bloques lógicos de un FPGA, ser interconectados según la necesidad del diseñador del sistema. Estos bloques lógicos pueden ser programados después de los procesos de manufactura por el usuario/diseñador, así que el FPGA puede desempeñar cualquier función lógica necesaria [20]. A diferencia de los procesadores que se encuentran en una computadora personal, al programar un FPGA el chip se vuelve a cablear para implementar su funcionalidad en lugar de ejecutar una aplicación de software.

2.7.1. Arquitectura FPGA

La arquitectura del FPGA (Field Programmable Gate Arrays) está conformada por una matriz de bloques lógicos configurables (Configurable Logic Blocks CLBs) interconectados entre sí. Los cuales implementan la lógica generadora de funciones. Ver Figura 2.19 muestra la arquitectura de un FPGA de la marca XILINX.

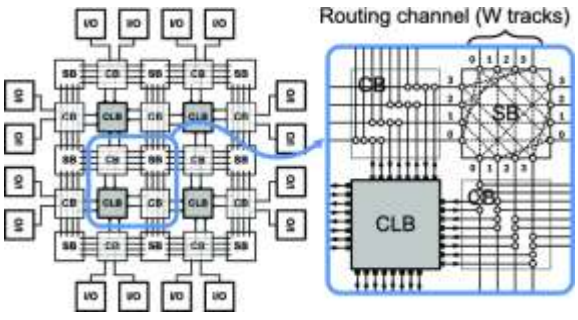


Figura 2. 19: Arquitectura general de un FPGA.

Un sistema FPGA contiene cinco elementos principales los cuales son:

- **Bloques de entrada y salida (IOB):** Proveen la interface entre los pines del integrado y la lógica interna. Las interfaces de entrada-salida son otro de los componentes particulares que tienen los FPGA's. Estos están divididos en bancos configurados, para interconectar la lógica de diferentes estándares eléctricos de manera independiente.
- **Bloques de lógica configurable (CLB):** Son bloques básicos que se utilizan en la implementación de un circuito digital. Todas las FPGA tienen algún tipo de CLB. Este es el corazón de la FPGA, y permite implementar las diferentes funciones lógicas.
- **Bloques de compensación y distribución de reloj (DLL del inglés "*Delay Locked Loop*"):** Estos bloques controlan los dominios de reloj dentro del integrado y compensan los retardos que pueda haber entre el reloj externo e interno.
- **Bloque de memoria RAM ("*Block RAM*"):** Son memorias dedicadas, integradas dentro de la lógica programable. Es una memoria de puerto doble, es decir, que puede leer y escribir al mismo tiempo.
- **Bloques de enlaces o matriz de interconexión:** Es una estructura versátil y multinivel de interconexión entre los componentes de la FPGA. La estructura de interconexión interna de un FPGA consiste en un conjunto de cables que se unen mediante elementos programables.

Con respecto a la arquitectura, cada chip de FPGA está hecho de un número limitado de recursos predefinidos con interconexiones programables para implementar un circuito digital reconfigurable y bloques de Entradas y Salidas para permitir que los circuitos tengan un acceso al mundo exterior. Los bloques de lógica configurables, también conocidos como CLBs, son la unidad de lógica básica de un FPGA. Algunas veces referido como segmentos o células de lógica, los CLBs están hechos de dos componentes básicos: Flip-Flops y Tablas de consulta (LUTs) o de sus siglas del inglés "look Up Table". Es importante tomar

esto en cuenta porque distintas familias de FPGAs se diferencian en la manera en que los Flip-Flops y las LUTs están empacados [21].

2.7.1.1. Aplicaciones

Las aplicaciones actuales de las FPGAs son: el procesado digital de señal, sistemas aeroespaciales y de defensa, prototipado de ASICs, visión artificial, reconocimiento de voz, criptografía, bioinformática, emulación de hardware, redes neuronales, etc. Las FPGAs competían con las CPLDs (Complex Programmable Logic Device) para reemplazar los circuitos digitales discretos, y con el tiempo han ido abarcando funciones cada vez más complejas, hasta incluso implementar SOCs (System-on-a-chip) e instalarse en sistemas reservados para chips DSP (Digital Signal Processing).

Se usan cada vez más en aplicaciones convencionales de computación de altas prestaciones HPC (High Performance Computing) como para calcular la transformada rápida de Fourier en FPGAs en vez de en microprocesadores. Aunque el uso de FPGAs en HPC está limitado por la complejidad del diseño con FPGAs, inconveniente que mejorará conforme haya disponibles mejores herramientas de programación de estos dispositivos.

2.7.1.2. FPGA y procesamiento de imágenes

En una FPGA la lógica requerida por una aplicación se implementa mediante hardware separado para cada función, por ello se considera de procesamiento paralelo. Tiene la ventaja de obtener una velocidad propia de un diseño hardware siendo posible su reprogramación. Por tanto, son muy adecuadas para el procesamiento de imágenes, en concreto en niveles bajo e intermedios para poder explotar mejor el paralelismo [21]. En una arquitectura segmentada, cada bloque hardware se construye para cada operación de procesamiento de imagen. En un sistema síncrono, cada dato se pasa de la salida de un bloque de una operación a la entrada del siguiente. Si no es síncrono se emplean buffers intermedios entre las operaciones de datos.

El paralelismo lógico en una operación de procesamiento de imagen se adapta perfectamente a una FPGA, consiguiendo mejorar el rendimiento de los algoritmos de procesamiento de imagen de manera significativa gracias al hardware paralelo. Ya que los datos de la imagen viajan en serie, si se implementan las operaciones de procesamiento en “stream”, el algoritmo puede resultar muy eficiente aprovechando los recursos de la FPGA.

2.7.1.3. Principales Fabricantes

En 1985, Xilinx Incorporated introdujo una idea completamente nueva en el campo de dispositivos programables: combinar el control del diseñador y el tiempo de desarrollo (time to Market) de los PLDs con la densidad y bajo costo de arreglos de compuertas. Rápidamente la idea tomó vuelo, los FPGAs se empezaron a fabricar, y hoy en día Xilinx es el principal vendedor de FPGAs. Hasta ese año los dispositivos que se usaban (y todavía se usan) cuando se necesitaba una gran cantidad de lógica combinacional y secuencial eran los Application Specific Integrated Circuit (ASIC), cuyo tiempo de desarrollo es muy largo, muchos meses llegando hasta años, y sumamente costosos [22]. Por ello los FPGAs logran insertarse rápidamente en un mercado que no tenía competencia. Hoy en día hay varios fabricantes entre los que se destacan FPGAs de empresas como:

- ***Xilinx***: Es uno de los grandes líderes de la producción de FPGAs. Xilinx es una compañía de tecnología americana, primeramente, un distribuidor de dispositivos lógicos programables, ver Figura 2.20. Es reconocida mundialmente por inventar los FPGAs y también por ser la primera compañía con modelos de manufactura Fabless.

Figura 2. 20: Fabricante Xilinx.

- ***Altera:*** Es el otro gran líder de la producción de los FPGAs. Altera corporation es uno de los pioneros de la lógica programable, ver Figura

2.21. Desarrolla algunas características que están orientadas hacia capacidad de sistemas en chips programables.

Figura 2. 21: Fabricante Altera.

- ***Lattice Semiconductor:*** Es un fabricante de FPGAs que lanzo grandes FPGAs con tecnología de 90nm, ver Figura 2.22. En adición, Lattice es un proveedor líder en tecnología no volátil, FPGAs basados en tecnología flash, con productos de 90nm y 130nm.

Figura 2. 22: Fabrica Lattice Semiconductor.

- ***Actel***: Tiene FPGAs basadas en tecnologías flash reprogramables, ver Figura 2.23. También ofrece FPGAs que incluyen mezcladores de señales basados en Flash.

Figura 2. 23: Fabrica Actel.

- ***QuickLogic***: Tiene productos basados en fusibles, es decir, solo son programables una sola vez, ver Figura 2.24.

Figura 2. 24: Fabrica QuickLogic.

❖***Atmel:*** Es uno de los fabricantes que sus productos no son re configurables, ver Figura 2.25. Ellos se enfocaron en proveer microcontroladores AVR con FPGAs, todo en el mismo encapsulado.

Figura 2. 25: Fabrica Atmel.

2.7.2. Lenguaje de descripción de hardware

A partir del desarrollo de circuitos Integrados Digitales programables con una gran cantidad de componentes lógicos y la necesidad de sistema digitales para aplicaciones más complejas, las herramientas de diseño tradicionales se vuelven cada vez más ineficientes y poco efectivas para lograr desarrollos adecuados, por lo tanto las empresas fabricantes de circuitos integrados desarrollan herramientas más útiles, originándose así los HDL ó Lenguajes de Descripción de hardware. Así cada empresa crea el suyo estableciendo una diversidad de lenguajes muy grande.

Para tratar de unificar estas herramientas, entre los años 1984 y 1987, el IEEE y el Departamento de 2 Unidos (DoD) patrocinan el desarrollo de un Lenguaje llamado VHDL. Su nombre viene de VHSIC HDL, o sea Lenguaje de Descripción de hardware para circuitos integrados de muy alta velocidad [23]. Considerando que un lenguaje de descripción de hardware es una herramienta formal que permite describir la estructura y comportamiento de un sistema para lograr una adecuada especificación, documentación y simulación del mismo antes de su realización real; para su implementación se establecieron ciertas características fundamentales, que aún hoy siguen siendo válidas, y son:

- ❖ Cada elemento de diseño tiene una interfaz única y perfectamente definida, que permite conectarla a otros elementos.
- ❖ Cada elemento tiene un comportamiento preciso y unívocamente definido, que permiten su posterior simulación.

- La especificación de comportamiento que permite definir la operatividad puede realizarse a través de un algoritmo ó de una estructura de hardware real.
- Los diseños mantienen una estructura jerárquica, que permite descomponerlo adecuadamente.
- Las características concurrentes, temporizadas y de sincronismo (por ej. reloj) pueden ser modeladas adecuadamente.
- Se puede simular cualquier operación lógica y de temporización.

Luego del desarrollo de este lenguaje aparecieron las herramientas adecuadas de síntesis que completan el panorama de diseño de un sistema digital.

El diseño de un proyecto en VHDL consiste en:

- Establecer las funciones que implementará el circuito.
- Codificar el programar con un lenguaje HDL.
- Análisis de la sintaxis y simulación del programa.
- Programar la FPGA y comprobar el funcionamiento.

Así de esta forma se puede decir que si utiliza VHDL se puede diseñar, simular y sintetizar cualquier sistema digital, desde el combinacional más simple hasta la estructura secuencial más compleja. Los lenguajes HDL son independientes del hardware, lo que significa que un código se puede usar en otros diseños más complicados y con otros dispositivos compatibles. Las nuevas versiones de HDL permiten también el desarrollo de circuitos analógicos.

2.7.2.1. Estructura

Este lenguaje está diseñado con principios de programación estructurada basado en Pascal y ADA. Damos las siguientes definiciones:

- **Entidad VHDL (*Entity*):** Declaración de los vectores del módulo, generalmente entradas y salidas. En otras palabras, podemos decir que se trata de identificador Vectorial del sistema.
- **Arquitectura VHDL (*Architecture*):** Descripción detallada de la estructura interna ó comportamiento del módulo.

Observado el siguiente diagrama en blocks, Figura 2.26,

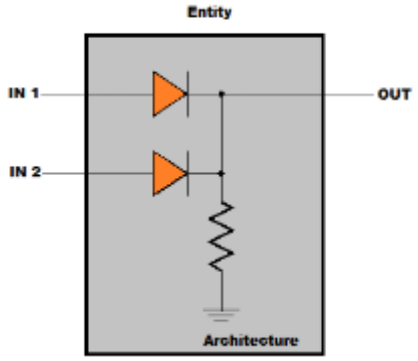


Figura 2. 26: Estructura de VHDL.

Vemos que la entidad implica todo lo externo que tiene un sistema, mientras que la arquitectura describe el interior del bloque.

Para realizar un circuito a través de VHDL se deben declarar ambas características, la entidad y la arquitectura [24]. Por lo tanto, un programa VHDL es un archivo de texto dónde se declara en primer lugar la entidad y luego se define la arquitectura, ver la Figura 2.27, para un decodificador.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Tabla is
port (
      entrada: in std_logic_vector(3 downto 0);
      salida: out std_logic_vector(6 downto 0)
);
end Tabla;

architecture Behavioral of Tabla is

begin
with entrada select
      salida <= "0000001" when "0000",
                "1001111" when "0001",
                "0010010" when "0010",
                "0000110" when "0011",
                "1001100" when "0100",
                "0100100" when "0101",
                "0100000" when "0110",
                "0001111" when "0111",
                "0000000" when  "1000",
                "0000100" when others;


end Behavioral;
```

ENTIDAD
Declaración de Entradas y Salidas.

ARQUITECTURA
Descripción del sistema.

Figura 2. 27: Programa VHDL.

CAPÍTULO 3. SOFTWARE DE XILINX VERSIÓN 14.7

Para el diseño sobre la FPGA se ha utilizado el entorno de desarrollo ISE Design Suite 14.7. Este entorno está formado por una serie de herramientas integradas entre sí de manera que se pueden hacer diseño hardware sobre esquemático u VHDL, integrar el hardware creado con otro entorno para diseño de procesadores embebidos y, finalmente, exportar todo a otro entorno para programar el código que se ejecutará sobre los procesadores anteriormente definidos.

El entorno de desarrollo incluye muchas herramientas de simulación, provisión de recursos dentro de la FPGA, etc. A continuación se describen cada una de las herramientas utilizadas dentro de este trabajo.

3.1. ISE Design Suite

ISE Design Suite 14.7, Figura 2.28 es un software desarrollado por Xilinx para sintetizar y analizar diseños de HDL (*Hardware Description Language*), el cual permite al desarrollador sintetizar ("compilar") sus diseños, realizar análisis de tiempo, examinar diagramas RTL, simular la reacción de un diseño, y configurar diversos dispositivos de destino con el programador [25] Xilinx ISE (Integrated Synthesis Environment) es un entorno de diseño para productos FPGA y está estrechamente relacionado con la arquitectura de dichos chips, ver Figura 3.1.

Figura 3. 1: Software ISE Design Suite v14.7.

Esta herramienta de diseño, permite crear sistemas complejos de procesadores embebidos dentro de una FPGA que interactúan con periféricos que pueden diseñar dentro de las propias FPGA o bien externos a la misma. Ofrece un entorno de desarrollo gráfico que permite añadir los periféricos usando *drag and drop*, y crear buses para interconectarlos.

3.1.1. Xilinx CORE Generator System (*LogiCore*)

Xilinx CORE Generator System es una funcionalidad incluida en el pack *ISE Design Suite* facilita en gran medida el diseño de componentes usados con frecuencia pero que requieren mucho tiempo para su programación. Para ello *Xilinx* ofrece componentes predefinidos, denominados *Intellectual Properties (IP),* para FPGAs configurables por el usuario. Estos *IPs* abarcan un rango muy amplio de aplicaciones, siendo los más comunes bloques de memoria, FIFOs o filtros y adaptadores. Usar estos *IPs* puede ahorrar días o incluso meses de tiempo de diseño, haciendo que un determinado proyecto pueda salir antes al mercado.

3.1.2. ISE Simulator (ISim)

ISim es un completo simulador HDL integrado en el pack *ISE*. La simulación HDL puede ser un paso fundamental en el diseño de cualquier proyecto, pudiendo testear el funcionamiento de cualquier módulo programado antes de aplicarlo sobre la plataforma real. Por este motivo, *ISim* se convierte en una herramienta básica la trabajar con la programación de FPGAs.

3.1.3. Interfaz de usuario

La interfaz de usuario principal de ISE es el Project Navegator, incluye varios paneles que se mencionan a continuación, ver Figura :

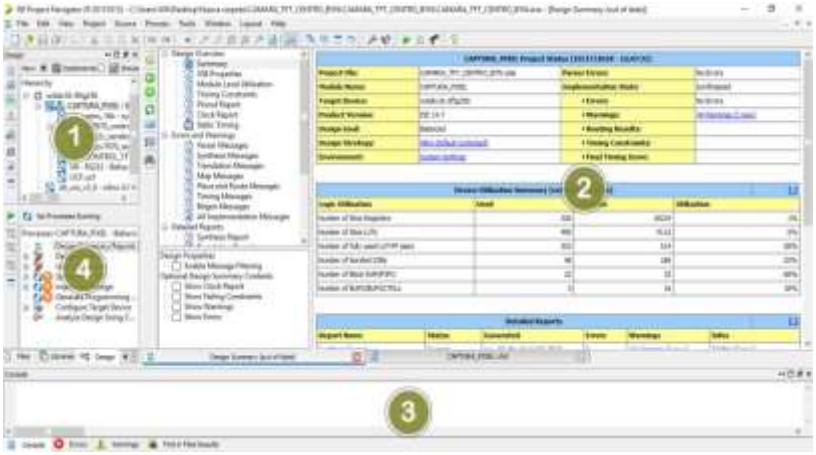


Figura 3. 2: Interfaz de usuario ISE *Design Suite*.

- ***Hierachy:*** La Jerarquía de diseño muestra los archivos de diseño, es decir los módulos VHDL, de descripción de hardware o de estímulos (test bench), que componen el proyecto, cuyas dependencias son interpretadas y mostradas por el ISE como una estructura de árbol, puede haber un módulo principal con otros módulos incluidos por el módulo principal, dichos módulos incluyen la configuración y asignación de pines, además se detalla el nombre del dispositivo FPGA utilizado y los nombres de todos los módulos.
- ***Workplace:*** Es un editor de código fuente en el cual se puede mostrar desde un módulo VHDL y un reporte de uso del FPGA, hasta resúmenes de los recursos que la FPGA está consumiendo para el diseño implementado.
- ***Console:*** La consola de salida o de transcripción proporciona el estado de las operaciones que están en ejecución además de informar sobre los problemas que se pueden encontrar en el diseño, dichos problemas pueden ser filtrados para mostrar Advertencias, Errores o ambos.
- ***Process:*** En el árbol de procesos se describen las operaciones que realizará el ISE en el módulo actualmente activo. La jerarquía incluye funciones de compilación, sus funciones de dependencia y otras utilidades. También denota problemas o errores que surgen con cada función.

CAPÍTULO 4. DESARROLLO DEL SISTEMA DE VISIÓN EN FPGA

El desarrollo de la programación en VHDL para la tarjeta ASSERTA y el módulo OV7670, se hace esquema a bloques general mostrado en la Figura 4.1, el cual se puede observar la interacción de los diferentes dispositivos utilizados para el sistema de visión. La cámara posee un sensor de imagen CMOS VGA OV7670, capaz de trabajar a un máximo de 30 cuadros por segundo a una resolución de 640x480 pixeles. Se utiliza la tarjeta de desarrollo Asserta (INTESC, 2019) que contiene un FPGA Spartan 6 XC6SLX16 y memoria SDRAM de 64 Mb. El FPGA controla la adquisición de las imágenes y visualiza el video en una pantalla TFT de 4.3 pulgadas de 480 por 272 pixeles de resolución.

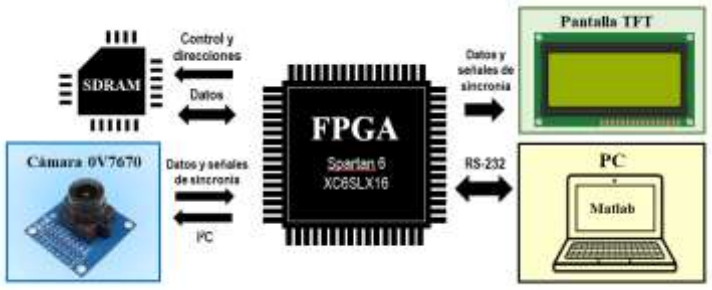


Figura 4. 1: Esquema a bloques general del sistema de visión en FPGA.

El funcionamiento del sistema de visión se basa en lo siguiente y se puede ver en la Figura 4.1: el FPGA recibe y procesa los datos de la cámara OV7670, cada cuadro recibido se guarda en la memoria SDRAM y se envía a la pantalla TFT. Adicionalmente se incorpora una PC que se comunica con el FPGA por medio del protocolo de comunicación RS-232, por lo que el FPGA puede enviar a la PC la imagen capturada, si recibe la instrucción. Entonces, para el sistema de visión, el FPGA incorpora 4 núcleos IP o bloques de controladores para operar la cámara OV7670, la comunicación RS-232, la pantalla TFT y la memoria SDRAM. La programación de los bloquees se realizó en lenguaje VHDL usando el software de Xilinx ISE-Design Suite. Todos los bloques son interconectados en el interior de una entidad general para su implementación física en la FPAGA Spartan 6. Los

bloques implementados en el FPGA se presentan de manera esquematica en en la Figura 4.2.

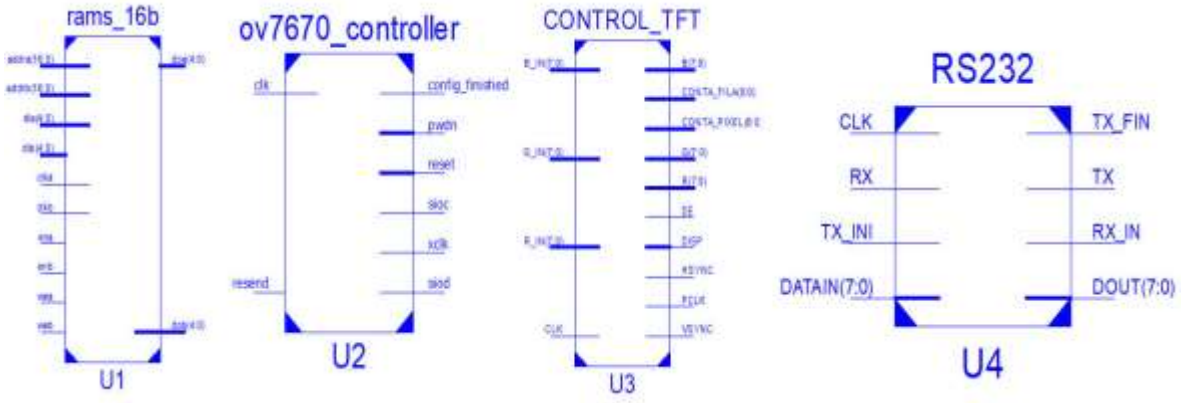


Figura 4. 2: Módulos de hardware en VHDL.

El código programado en VHDL se encargará de realizar un bloque o módulo principal, el cual se desarrollan 3 procesos en paralelo, el cual realizara la interconexión de todos los submódulos para obtener y procesar los datos enviados por el módulo OV7670, enviar las imágenes a la TFT y para la comunicación RS232 entre la computadora y la tarjeta.

4.1. Bloque de control de la cámara.

En la recepción del vídeo con la cámara y el procesamiento del vídeo, se utilizo un bloque de control representado en la Figura 4.2. como “U2”, el cual consiste en una máquina de 6 estados que se presenta en la Figura 4.3.

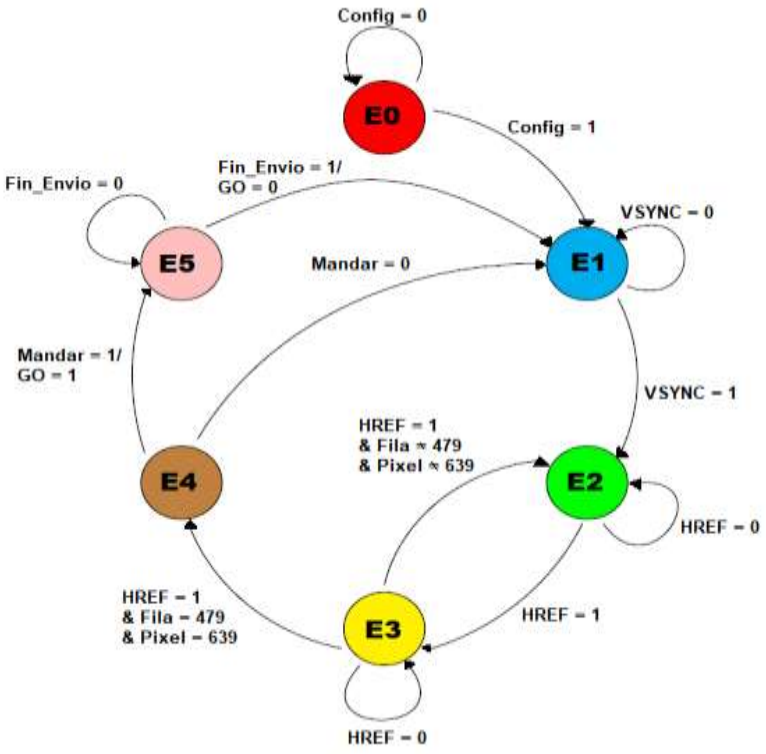


Figura 4. 3: Máquina de 6 estados general del sistema propuesto.

Cada estado se encarga de realizar una tarea específica en la cámara, el cual consiste en lo siguiente:

- **Estado E0:** Realiza una configuración inicial en los registros internos de la cámara. La carga de los valores apropiados de configuración se realiza mediante un puerto comunicación serial I2C; que incorpora la cámara para este propósito y que también fue incluido en el bloque de control de la cámara. El estado E0 también verifica si se ha realizado una configuración correcta de la cámara, y la máquina no cambia al siguiente estado, si la cámara no responde con una confirmación positiva.
- **Estado E1:** Monitorea si la cámara está lista para iniciar la transmisión de un cuadro de imagen. El monitoreo se realiza a través de la señal de sincronización de la cámara VSYNC, que indica el momento en que inicia una imagen nueva.

- **Estado E2:** Captura los bytes provenientes de la cámara y los agrupa en un buffer temporal con capacidad para almacenar una línea completa de 640 pixeles. Los pixeles son almacenados en dicho buffer en un formato RGB-565. Cada vez que un pixel es capturado la máquina pasa al siguiente estado llamado E3.
- **Estado E3:** El controlador monitorea si la captura de la actual línea de la imagen ya se ha completado. Esto lo realiza verificando la señal de sincronización HREF proveniente de la cámara. Si una línea completa se ha capturado, el bloque pasa al estado E4.
- **Estado E4:** Se encarga de guardar los pixeles en la SDRAM y pasar al estado E5.
- **Estado E5:** La máquina monitorea si una nueva línea de la imagen, será enviada por la cámara o si una nueva imagen será enviada por la cámara. Si continúa el envío de una línea, la máquina regresa del estado E5 al E2, no obstante, si una nueva imagen será enviada, el control saltará hacia el estado E1 para comenzar un nuevo ciclo de captura.

4.2. Bloque de control de la SDRAM.

Este bloque de control se encarga de almacenar de forma ordenada los pixeles hacia la memoria SDRAM. Además realiza una compresión de la imagen mediante la binarización por umbral de cada pixel de la imagen. De esta forma se almacena la información original y la información binarizada de la imagen. Este proceso se realiza cada vez que el bloque de control de la cámara, manda la orden de almacenar una línea de la imagen. De esta manera se aprovecha el tiempo de espera entre cada captura de línea, para almacenar la línea actual. Además de lo anterior, este bloque puede leer una imagen almacenada en la SDRAM y enviarla hacia el bloque de control de la pantalla TFT o hacia el bloque de comunicación RS-232. Este proceso se realiza entre cada tiempo de espera de captura de una nueva imagen proveniente de la cámara. esclavo, que espera la orden del bloque de control de la cámara o del bloque de RS-232. Adicionalmente a estos dos procesos, de manera autónoma transfiere la información desde la SDRAM hacia el

control de pantalla TFT para la visualización de la imagen. Sin embargo, si una orden de transferencia, proveniente del control RS-232, es recibida, el bloque interrumpe la transmisión de la imagen hacia la pantalla y la redirige hacia el bloque RS-232; dando prioridad de la transferencia hacia el ordenador de un usuario.

4.3. Bloque de control de RS-232

Este bloque de control se puede observar en la Figura 4.2, identificado como "U4", el cual permite enviar una imagen almacenada en la SDRAM, hacia un ordenador para su procesamiento. El usuario puede seleccionar el tipo de imagen que se transfiere, entre la original o la binarizada. Para realizar esta tarea, el bloque acepta dos comandos simples de 8 bits; la recepción de un número 2, le indica transferencia de la imagen original y un número 3 le indica la transferencia de una imagen binarizada. La información se envía mediante un formato estándar de 1 bit de inicio, 8 bits de datos y un bit de paro. El bloque además verifica que todos los pixeles de la imagen se envíen mediante el conteo de los mismos. La velocidad de trasferencia que alcanza este bloque es de 115200 baudios.

4.4. Bloque de control de la pantalla TFT

Su principal función es generar las señales de control para la pantalla TFT, ver Figura 4.2. el bloque "U3". La pantalla utilizada es de la marca Newhaven display que funciona bajo un esquema VGA. Es decir, el controlador debe generar las señales de sincronización vertical *VSYNC* y horizontal *HSYNC*, para indicarle a la pantalla cuando iniciará el despliegue de una nueva imagen. Los datos de los pixeles se envían en una configuración de 24 bits (8 bits para cada color). El bloque emplea una señal de reloj PCLK para sincronizar los datos y el despliegue de cada línea de la imagen. La velocidad de refresco de la imagen, está fijada a 15 cuadros por segundo. Como se menciona anteriormente este bloque es autónomo en lo referente al despliegue de las imágenes y control de la pantalla, sin embargo, si depende del bloque de control de la memoria SDRAM para obtener la información de los pixeles. Debido a que dicho bloque le envía la información

cuando están disponibles en la SDRAM. Sin embargo cuando se requiere transferir la imagen hacia un ordenador, el bloque de control de la SDRAM, interrumpe el envío normal de la información hacia este bloque y en su lugar envía pixeles negros. Por este motivo, cuando una imagen se está transfiriendo hacia el usuario, el despliegue de la imagen en la pantalla se ve interrumpida y se pone en color negro.

4.5. Binarización en el procesamiento de imagen

El procesamiento que ejecuta el FPGA sobre las imágenes consiste en una binarización. Aplicar el cálculo de la rotación en imágenes binarias reduce el tiempo de procesamiento, pues en este tipo de imagen es más fácil y rápido obtener las propiedades geométricas y topológicas de los objetos capturados. Para obtener la binarización, el primer paso es trabajar con la imagen en escala de grises, de esta manera los datos de los pixeles son valores entre 0 y 255. Se fija el valor de un umbral en la escala de grises que sirve para convertir a un valor binario 0 todos los pixeles cuyo nivel de gris sea inferior a ese umbral, y un 1 para todos aquellos pixeles que superen el umbral. Cabe mencionar que para aplicar un umbral fijo implica controlar los parámetros del entorno en que se capturará la imagen, pues cualquier variación en la iluminación pueden ocasionar cambios en los niveles de gris de la imagen que invaliden el umbral fijado. Para ejecutar el proceso de binarización en código VHDL se adquiere la matriz del plano del color verde, del formato RGB, pues al ser un formato 565, es el plano que tiene mayor información. El pseudocódigo que se ejecuta en el FPGA, es el siguiente:

- Binarización con umbral fijo de 75

Algoritmo. Binarización fija
im : imagen en escala de grises de tamaño (*filas*, *columnas*) imB : imagen binaria *umbral* = 75 **Inicializar** imB = 0

Mientras se recorre im(*x*,*y*) *x* = 1… *filas*, y = 1… *columnas*
  Si im(*x*, *y*) > *umbral*
    imB = 1
  Fin
Fin
Devolver *imB*

4.6. Algoritmo para el cálculo del ángulo de rotación en Matlab.

Es el algoritmo basado en los segundos momentos de inercia del trabajo de tesis del estudiante Jesús Ángel Aragón Morales, el cual presento su proyecto con el titulo de *"Algoritmo de visión artifical para el cálculo de la orientación de componentes electrónicos aplicados a procesos de manufactura de alta velocidad"*, enfocado a determinar el área de los objetos captados por la cámara OV7670, además de los momentos de inercia para calcular los grados de desfase que tienen.

Es por ello que realizo un analisis a partir de la orientación de un objeto el cual puede determinarse mediante el cálculo del eje respecto al cual el momento de inercia es mínimo (De la Fuente y Trespaderne, 2012). El eje de inercia mínimo de una región será la recta *Ax + By + C = 0*, tal que la suma de las distancias al cuadrado entre los píxeles del objeto y dicha línea es mínima.

Como se muestra en la Figura 4.4, el vector *(A, B)= (cos θ, sen θ)* es un vector perpendicular a la recta, siendo θ el ángulo que forma la perpendicular desde el origen a la recta con el eje x. Así, el ángulo θ se obtiene fácilmente tras el cálculo de los momentos de segundo orden I_{xx}, I_{yy} *e* I_{xy}.

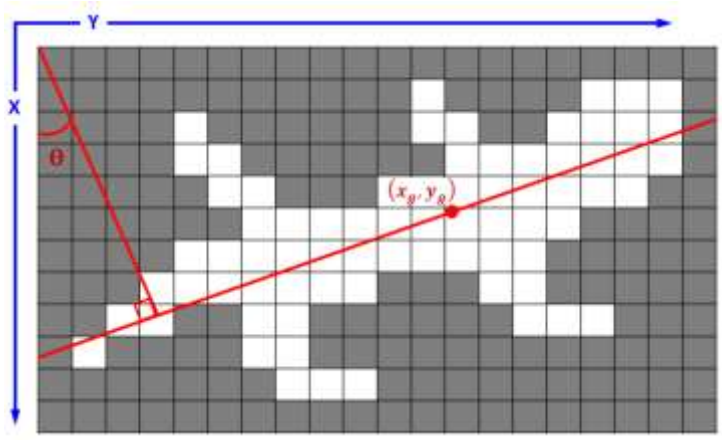


Figura 4. 4: Orientación de un objeto sobre una imagen.

Los ejes de inercia pasan por el centro de gravedad del objeto (x_g, y_g). De acuerdo con los investigadores De la fuente y Trespaderne, los tres momentos de inercia de segundo orden de una región $I_{xx}, I_{yy}\ e\ I_{xy}$ se expresan:

$$I_{xx} = \sum_{x=1}^{f} \sum_{j=1}^{c} (x - x_g)^2 \cdot B(x,y) = \sum_{i=1}^{N} (x_i - x_g)^2 \quad (1)$$

$$I_{xy} = \sum_{x=1}^{f} \sum_{j=1}^{c} (x - x_g) \cdot (y - y_g) \cdot B(x,y) = \sum_{i=1}^{N} (x_i - x_g) \cdot (y_i - y_g) \quad (2)$$

$$I_{yy} = \sum_{x=1}^{f} \sum_{j=1}^{c} (y - x_g)^2 \cdot B(x,y) = \sum_{i=1}^{N} (y_i - x_g)^2 \quad (3)$$

La suma de todas las distancias al cuadrado $\sum_{i=1}^{N} {d_i}^2$ está dada por:

$$\sum_{i=1}^{N} {d_i}^2 = cos^2\theta \cdot I_{xx} + sen^2\theta \cdot I_{yy} + 2cos\theta sen\theta \cdot I_{xy} \quad (4)$$

Para determinar la recta que minimiza la sumatoria de las distancias al cuadrado, se deriva la ecuación (4) respecto al parámetro *θ*, y se iguala a 0, lo que resulta:

$$-2cos\theta sen\theta \cdot I_{xx} + 2cos\theta sen\theta \cdot I_{yy} + 2(cos^2\theta - sen^2\theta) \cdot I_{xy} = 0 \quad (5)$$

Finalmente realizando el despeje de *θ,* se obtiene:

$$\theta = 0.5 \arctan \frac{2I_{xy}}{I_{xx} - I_{yy}} \quad (6)$$

El cual con la implementación de la ecuación (6) se obtiene el ángulo de rotación del objeto capturado por la cámara, pero dicha fórmula puede generar divisiones por cero lo que ocasionaría indeterminaciones en los resultados, para evitar eso se recomienda usar la siguiente formula:

$$\theta = \frac{atan2(y,x)}{2} \tag{7}$$

La función atan2 tiene la ventaja de tomar en cuenta los 4 cuadrantes por lo que los resultados son más fiables, se comienza con el cálculo de los valores x,y para después ser usados en la ecuación (7).

Sin embargo para el cálculo de los momentos de inercia es necesario extraer algunas características de la imagen: área y centro de gravedad del objeto detectado. De modo que el área de una región sobre la imagen está dada por el número de pixeles que la constituyen, por lo que está dada por la expresión:

$$\acute{A}rea = \sum_{x=1}^{filas} \sum_{y=1}^{columnas} B(x,y) \tag{8}$$

El área también es denominada momento de orden cero. El centro de gravedad se obtiene a partir de los momentos de primer orden divididos por el área.

$$x_g = \frac{\sum_{x=1}^{f} \sum_{y=1}^{c} x \cdot B(x,y)}{\acute{A}rea} \tag{9}$$

$$y_g = \frac{\sum_{x=1}^{f} \sum_{y=1}^{c} y \cdot B(x,y)}{\acute{A}rea} \tag{10}$$

Los algoritmos para la obtención del centro de gravedad y el cálculo de los momentos de inercia se presentan continuación :

❖ Obtención del centro de gravedad del objeto sobre la imagen

Algoritmo. Centro de gravedad
imB: imagen binaria (0=Fondo, 1=Objeto) de tamaño (*filas*, *columnas*) area = 0 Ix = 0 Iy = 0 **Mientras** se recorre imB(*x*,*y*) *x* = 1… *filas*, y = 1… *columnas* **Si** imB(*x*, *y*) ≠ 0 area = area + 1 Ix = Ix + *y* Iy = Iy + *x* **Fin** **Fin** Cx = Ix/area Cy = Iy/area Centro = [Cx Cy]

❖ Obtención de los momentos de segundo orden

Algoritmo. Momentos de segundo orden de inercia
imB: imagen binaria (0=Fondo, 1=Objeto) de tamaño (*filas*, *columnas*) Centro: vector con las coordenadas del centro de gravedad (*fila*, *columna*) Ixx = 0 Iyy = 0 Ixy = 0 **Mientras** se recorre imB(*x*,*y*) *x* = 1… *filas*, y = 1… *columnas* **Si** imB(*x*, *y*) ≠ 0

```
        Ixx = Ixx + (y – Centro(1)) ^ 2
        Iyy = Iyy + (x – Centro(2)) ^ 2
        Ixy = Ixy + (y – Centro(1)) * (x – Centro(2))
    Fin
Fin
```

CAPÍTULO 5. DESARROLLO DEL ALGORITMO DE ORIENTACIÓN EN VHDL.

Dentro de este capítulo se explicará el procedimiento realizado para obtener el algoritmo de orientación en FPGA, que será utilizado dentro de este proyecto, haciendo uso de las herramientas del software Matlab e ISE Design Suite 14.7.

5.1. HDL coder

Matlab cuenta con herramientas que permiten la transformación de código en Matlab como funciones o programas a código VHDL lo cual facilita el desarrollo de algoritmos para trabajar como xilinx y FPGAs.

HDL Coder ™ genera códigos portátiles, sintetizables VHDL® y Verilog® a partir de las funciones MATLAB®, los modelos Simulink® y las tablas de Stateflow®. El código HDL generado se puede utilizar para la programación de FPGA o el diseño y creación de prototipos de ASIC. HDL Coder proporciona un asesor de flujo de trabajo que automatiza la programación de Xilinx® y Altera® FPGA. Puede controlar la arquitectura e implementación de HDL, resaltar rutas críticas y generar estimaciones de utilización de recursos de hardware. HDL Coder proporciona trazabilidad entre su modelo de Simulink y el código HDL generado, lo que permite la verificación del código para aplicaciones de alta integridad que cumplen con el DO-254 y otros estándares.

HDL Coder es una herramienta de uso muy intuitivo, el cual convierte automáticamente código Matlab de punto flotante a de punto fijo y genera VHDL sintetizable y código Verilog. Es por ello que se usó una ruta popular probada en producción para tomar un algoritmo MATLAB DSP a través de Fixed-Point Designer ™, y HDL Coder ™ para después ejecutarlo en una tarjeta FPGA o ASIC, ver Figura 5.1.

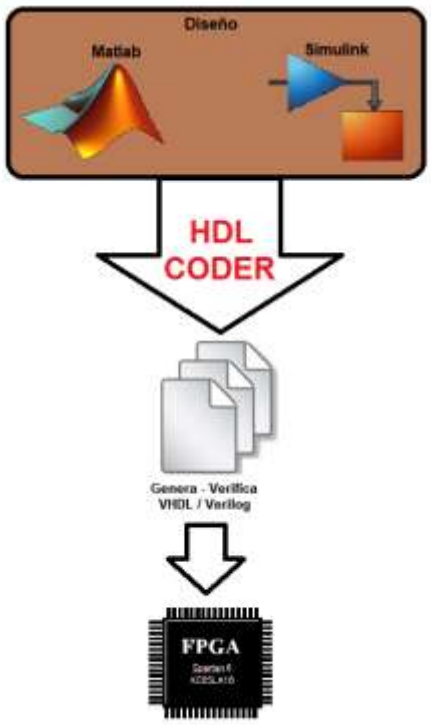


Figura 5. 1: Diagrama de diseño en HDL CODER.

5.1.1. Características HDL coder

Las características que describen a esta herramienta están basadas a que pueden desarrollar aplicaciones en un lenguaje de alto nivel para luego transformarlo a código HDL para su implementación en software.

- Código VHDL y Verilog son independiente del objetivo y sintetizable.
- Compatible con la generación de código para funciones MATLAB, y bloques en Simulink.
- Asesor de flujo de trabajo para la programación de tableros de aplicación Xilinx y Altera.
- Controla el contenido, optimización y el estilo de código generado.
- Genera VHDL y Verilog bancos de prueba para la verificación rápida y validación de código generado.
- Las bibliotecas proporciona funciones complejas, tales como el decodificador viterbi, FFT, filtros, CIC, y filtros FIR, para sistemas de

procesamiento de señales de modelado y de comunicación y el código de generación de HDL.

5.1.2. Fixed Point Designer

"*Fixed point designer*" modela y optimiza algoritmos de punto fijo y punto flotante, de tal manera que explora los tipos de datos de punto flotante y punto fijo para analizar la compensación en la precisión numérica.

"*Fixed point designer*" proporciona tipos de datos y herramientas para desarrollar algoritmos de punto fijo y precisión simple para optimizar el rendimiento en *hardware* integrado. "*Fixed point designer*"analiza su diseño y propone tipos de datos y atributos, como la longitud de la palabra y la escala. Puede especificar atributos de datos detallados, como el modo de redondeo y la acción de desbordamiento, y mezclar datos de precisión simple y de punto fijo. Puede realizar simulaciones de bits verdaderos para observar el impacto del rango y la precisión limitados sin implementar el diseño en el *hardware*. "*Fixed point designer*" admite la generación de códigos C, HDL y PLC.

En el *hardware* digital, los números se almacenan en palabras binarias. Una palabra binaria es una secuencia de bits de longitud fija (1 y 0). La forma en que los componentes de hardware o las funciones de software interpretan esta secuencia de 1 y 0 está definida por el tipo de datos.

Un tipo de datos de punto fijo se caracteriza por la longitud de la palabra en bits, la posición del punto binario es el medio por el cual se escalan e interpretan los valores de punto fijo. Una representación común de un número de punto fijo binario, se muestra en la siguiente Figura 5.2.

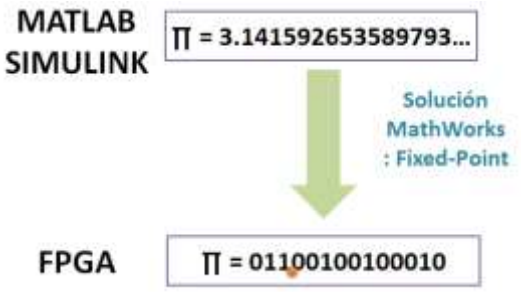


Figura 5. 2: Ejemplo de conversión *"Fixed Point".*

5.2. Algoritmo de Matlab a VHDL

Para obtener la implementación en un FPGA es necesario desarrollar y verificar los diseños de hardware a un alto nivel de abstracción y generar automáticamente un código RTL sintetizable. El diagrama de la Figura 5.3 presenta el proceso de conversión que se diseñó para generar el código en VHDL.

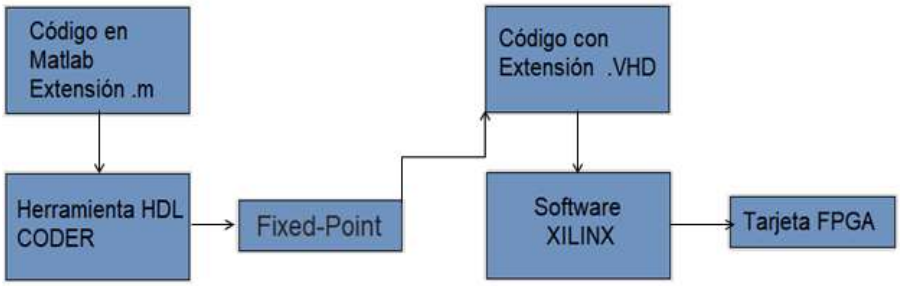


Figura 5. 3: Diagrama de conversión Matlab a VHDL.

Como prueba preliminar, para evaluar que el diseño es funcional, se realizó una prueba muy sencilla. Para ello se desarrollaron dos archivos en Matlab, una función y un script simulando una compuerta *"AND"* como se muestra en el diagrama de la Figura 5.4.

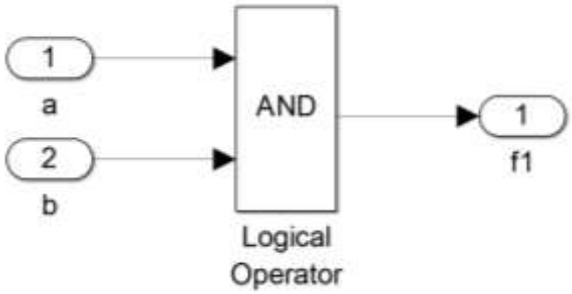


Figura 5. 4: Diagrama de la prueba con compuerta AND

Posteriormente se abrió la herramienta HDL coder que se encuentra en el apartado de APPS de Matlab R2015a, donde se colocan los dos archivos anteriormente dichos, ver Figura 5.5.

Figura 5. 5: Ventana para agregar los archivos

Luego se abrirá una nueva ventana en el que se definen las características del código a generar, el cual se tiene que habilitar el tipo de dato de conversión a realizar, para este caso será de tipo punto fijo (Fixed Point) ver Figura 5.6.

Figura 5. 6: Ventana para habilitar tipo de conversión.

Finalmente se da clic en “RUN” para generar el código en VHDL, el cual mostrará los archivos al terminar de ejecutarse, ver Figura 5.7.

Figura 5. 7: Ventana con los archivos generados.

Una vez convertido el código en lenguaje de VHDL, se procede abrir el software ISE Design Suite para comprobar que no existan errores en el algoritmo, para ello se abre un nuevo archivo, el cual solicita las especificaciones del tipo de FPGA en el que se estará implementando el código, ver Figura 5.8.

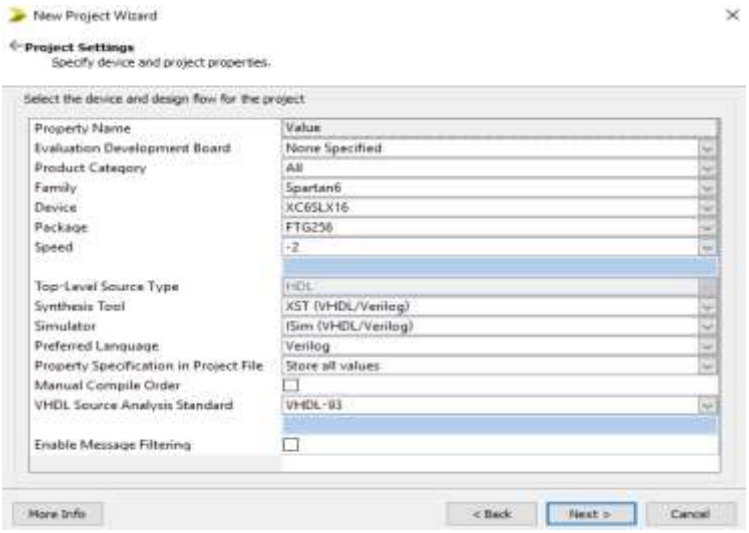


Figura 5. 8: Especificaciones de FPGA Spartan-6.

Al configurar los datos de la tarjeta a utilizar, se procede a abrir un Módulo en VHDL, el cual contendrá el código generado en Matlab. Posteriormente se procede a verificar si no existen errores en el código dando doble clic en "Synthesize" ver Figura 5.9.

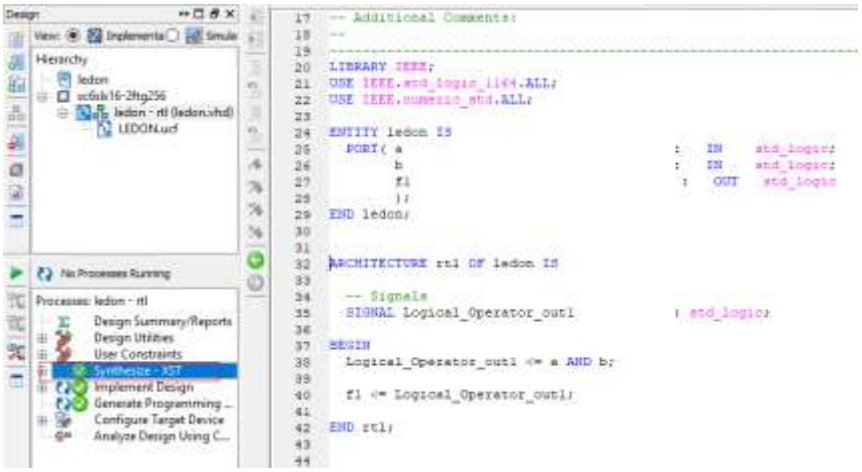


Figura 5. 9: Ventana de verificación en ISE desing suite.

Cuando esté sea correcto y aparezca una palomita de color verde, procederemos a abrir un archivo de restricciones de implementación con extensión ".ucf", ver Figura 5.10. En él se mapearán las entradas y salidas representadas con los pines físicos del FPGA Spartan. Ver Figura 5.11. Paso seguido se verifica nuevamente todos los archivos para observar si no existen errores.

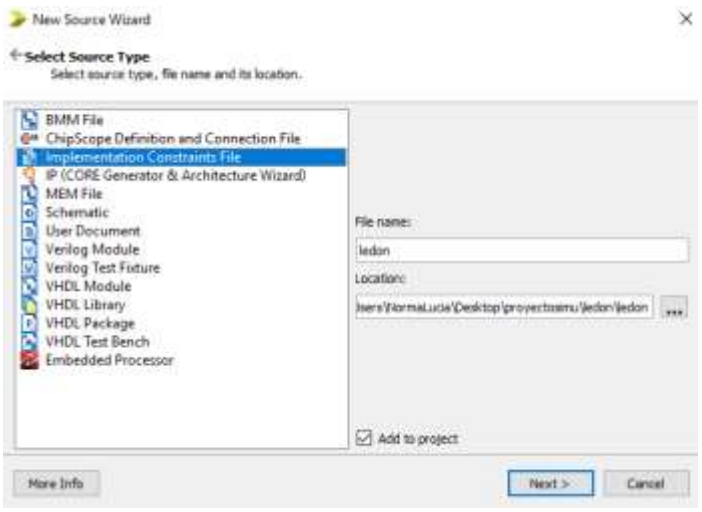


Figura 5. 10: Creación de archivo con extensión *".ucf".*

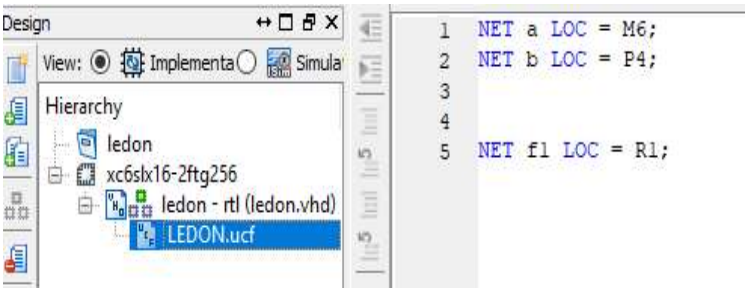


Figura 5. 11: Relación de entradas y salidas con el FPGA:

Por último, para implementar el código en la FPGA y observar su funcionamiento, se utiliza el software Integra de la compañía de Intesc, para programar la tarjeta. Para esto se conecta la tarjeta al computador por USB y se abre integra para realizar la vinculación, se busca el archivo ".bit", se selecciona FPGA y seguido se da clic en programar ver Figura 5.12.

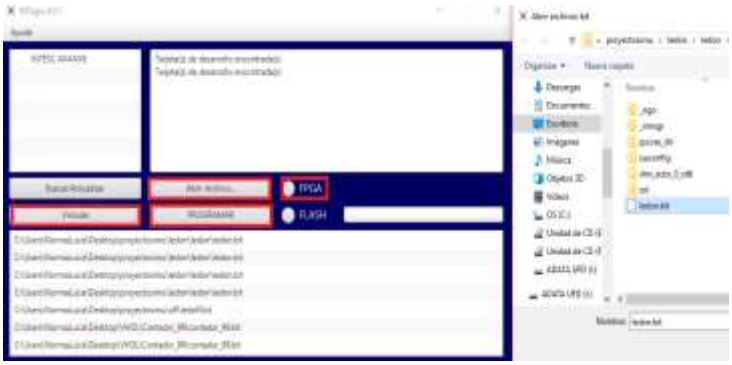
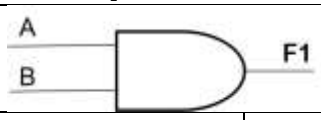


Figura 5. 12: Vinculación Integra con el FPGA.

Finalmente se realiza la prueba sobre el FPGA, donde se tiene como entrada dos switch y como salida un led, de tal manera que al realizar las combinaciones binarias que representa una compuerta AND, se tiene lo siguiente, ver Tabla 1:

Tabla 1 Resultados

Compuerta AND			
			
Combinación binaria			**Combinación física en FPGA**
Entrada **"a"**	Entrada **"b"**	Salida **"F1"**	Resultados en tarjeta AVANXE
0	0	**0**	
0	1	**0**	
1	0	**0**	

1	1	**1**			

Con lo anterior explicado, se procede a seguir los mismos pasos con los archivos que describen el código de orientación en Matlab para realizar la traducción del mismo, teniendo como resultado el siguiente código, ver Figura 5.13. Posteriormente se realizará las pruebas para observar y verificar los resultados, sobre la FPGA.

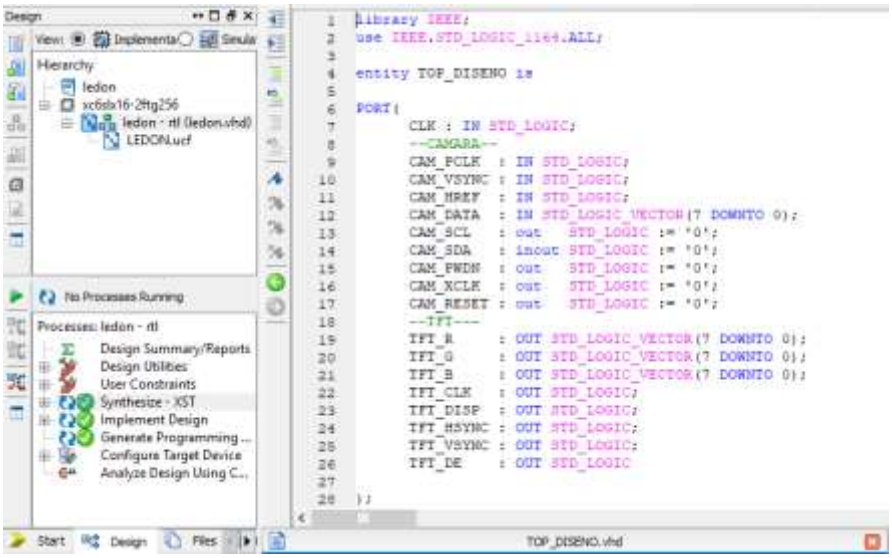


Figura 5. 13: Traducción del código en VHDL

CAPÍTULO 6. RESULTADOS

En este capítulo se presentan los resultados finales de la traducción y vinculación del algoritmo de orientación en la tarjeta FPGA.

La Figura 6.1 presenta el esquema de conexiones del diseño estructural en VHDL del sistema de visión. Como puede observarse el diseño final consiste en 7 bloques de cada proceso realizado previamente en Matlab.

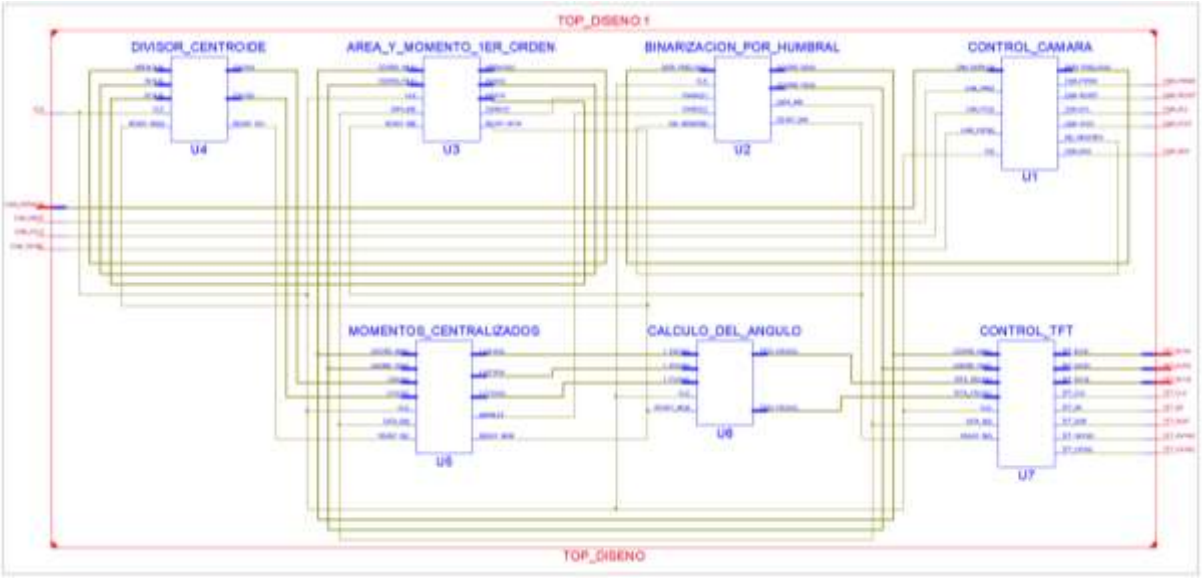


Figura 6. 1: Diseño estructural del TOP del sistema de visión en VHDL.

Para realizar las pruebas del sistema, la Figura 6.2 presenta el ensamble del sistema de visión. Hasta ahora, las capturas se realizan en un ambiente controlado, pues el circuito integrado se coloca sobre una superficie de color blanco y a 10 cm de distancia de la cámara. Estos circuitos integrados son de tecnología SMT, tal como los que manipula la máquina pick & place. La Figura 6.3 presenta los 5 tipos de encapsulados de los circuitos integrados y la Tabla 2 los resultados de dichas imágenes.

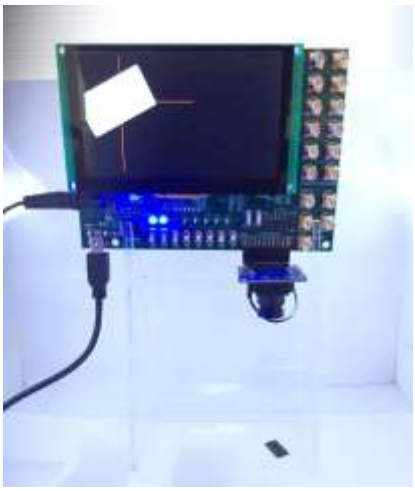
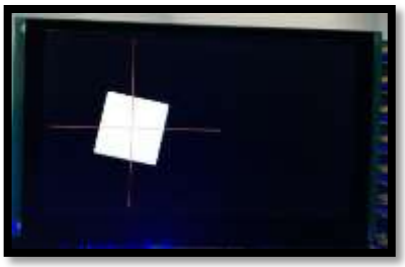
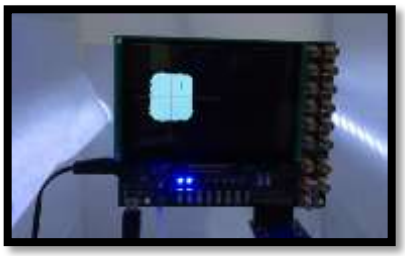
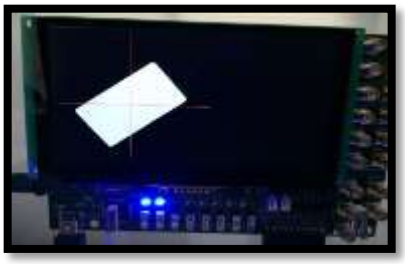


Figura 6. 2: Ensamble del sistema de visión.

Figura 6. 3: Circuitos integrados.

Tabla 2 Resultados del algoritmo traducido sobre la FPGA

Resultados del Sistema de visión sobre la FPGA	Valor del ángulo calculado
	**Ángulo = 18.3°**
	**Ángulo = 14.5°**
	**Ángulo = 1.2°**
	**Ángulo = 53.4°**
	**Ángulo = 6.2°**

Tal como se reportó en el trabajo de tesis previo [1], el algoritmo reportó un 99.9% de precisión. Por lo que se considera que se ha alcanzado ese mismo parámetro en este sistema. El trabajo en progreso es montar el sistema sobre la máquina, sin embargo se está llevando a cabo por el personal de la empresa INTESC.

CONCLUSIONES

- Se presentó un sistema de visión con captura de video y cálculo del ángulo de orientación en FPGA basado en los segundos momentos de inercia.
- El trabajo se dividió en 2 etapas; la primera consistió en la traducción del algoritmo de la orientación al lenguaje VHDL para vincularlo sobre la tarjeta Asserta por la compañía de instesc y la segunda en realizar pruebas para verificar si el codigo determinar los grados de orientación de los objetos vistos por la cámara.
- Con base a los resultados obtenidos con el algoritmo desarrollado se puede concluir que se cumplieron con todos los objetivos planteados.
- Este trabajo es la etapa final del proyecto “Sistema de visión y procesamiento de video en tiempo real embebido en FPGA para procesos de manufactura de alta velocidad”, con clave 529.18-PD, financiado por el TecNM a través de la Convocatoria 2018 de Apoyo a la investigación científica y tecnológica en los programas educativos de los Institutos Tecnológicos Descentralizados.
- El trabajo futuro consiste en montar el sistema a la máquina pick & place, lo cual será ejecutado por personal de la empresa INTESC.
- En base a los resultados obtenidos con el algoritmo desarrollado se puede concluir que se cumplieron con todos los objetivos planteados al comienzo de éste documento, cabe mencionar que éste proyecto queda como referente y motivación para que los estudiantes de mecatrónica del Instituto Tecnológico Superior de Atlixco colaboren con empresas para innovar y desarrollar nuevas tecnologías dando solución a los problemas con los que se enfrentan, de esa forma amplían sus conocimientos en su formación profesional.

REFERENCIAS

[1]. J.A. Aragón Morales, *"Algoritmo de Visión Artificial para el Cálculo de la Orientación de Componentes Electrónicos Aplicados a Procesos de Manufactura De Alta Velocidad*", Tesis Ing. Mecatrónico, ITSA, México, Pue, 2019.

[2]. INTESC "*Electronic & Embedded*" (2017) [Online]. Disponible: https://www.intesc.mx/nosotros.

[3]. M.F. Puerto Ordaz. Tecnologia de Montaje Superficial, [Online]. Disponible: https://tecnologiademontajesuperficial.es.tl.

[4]. X. Crespo. (2017, Julio 18). "*Qué es una FPGA y por qué jugarán un papel clave en el futuro"* [Online]. Available: https://planetachatbot.com/qu%C3%A9-es-una-fpga-y-por-qu%C3%A9-jugar%C3%A1n-un-papel-clave-en-el-futuro-e76667dbce3e.

[5]. J.C Soto Bohórquez, "*Desarrollo de sistema FPGA aplicado a control e identificación de un módulo experimental para regulación de pH*", Tesis de licenciatura, Ing. Mecanica, Universidad de Piura, Piura, Marzo 2013.

[6]. B. Alacid Soto, "*Control visual embebido en dispositivos FPGA*", M.S. tesis, M.S. Universitario en Automatización y Robótica, Universidad de Alicante, Julio 2014.

[7]. O. Bajío, E.Rodríguez Gónzalez, "¿Sabes qué es un sistema embebido?" El Financiero, Mayo 2018.

[8]. A. Quintero, E. Vallejo, "*Image Processing Algorithms Using FPGA Algoritmos de Procesamiento de Imágenes en FPGA*", Revista Colombiana de Tec. De Avanzada, vol.1, no.pp.7, 2006.

[9]. M. Wnuk. "*Remarks on Hardware Implementation of Image Processing Algorithms*". International Journal of Applied Mathematics and Computer Science, 18(1):105110, 2008.

[10]. Omni Vision, "User Manual Data Sheet OV7670" (2006) [Online]. Available: http://www.haoyuelectronics.com/Attachment/OV7670%20+%20AL422B%28

FIFO%29%20Camera%20Module%28V2.0%29/OV7670%20datasheet%28V1.4%29.pdf

[11]. H. Kalasua (2017, Septiembre 11). OV7670+Arduino Uno+SD module, [Online]. Disponible: https://hkalasua.wordpress.com/2017/09/11/ov7670-arduino-sd.

[12]. D.G. Maxinez, Programación de sistemas digitales con VHDL. 1ra ed. México:Patria,2013.

[13]. MathWorks. Codificador HDL, [Online]. Disponible: https://la.mathworks.com/products/hdl-coder.html.

[14]. MathWorks. Uso de Simulink para implementar un algoritmo MATLAB en un FPGA o ASIC, [Online]. Disponible: https://la.mathworks.com/videos/using-simulink-to-deploy-a-matlab-algorithm-on-an-fpga-or-asic-1484667134277.html.

[15]. MathWorks (2010). Accelerate FPGA Prototyping with MATLAB and Simulink, [Online]. Disponible: http://www2.fhi.nl/de/archief/2010/images/mathworks_accelerate_fpga_prototyping_with_matlab__simulink_s.van_beek.pdf.

[16]. MathWorks (2012). HDL CODER, [Online]. Disponible: https://www.mathworks.com/tagteam/72320_hdl-coder.pdf.

[17]. H. Fabregat Marcos , “Generación de código mediante HDL Coder para procesamiento y clasificación de imágenes biomédicas”, M.S. Ing. Informática, Fac. Inform, U.C.M, Madrid, Julio 2017.

[18]. M. Nuñez (2015, Agosto 13). Generar código VHDL con Matlab [Online]. Disponible: https://prezi.com/feef7jiytkwv/hdl-coder-es-una-herramienta-de-uso-muy-intuitivo-convierte.

[19]. E. San Millán Heredia, M. Portela García (2011,Octubre 25). Circuitos integrados y Microelectronica. Universidad Carlos III de Madrid [Online]. Disponible:http://ocw.uc3m.es/tecnologia-electronica/circuitos-integrados-y-microelectronica/practicas-1/practica-de-vhdl.

[20]. MathWorks (2012). Fixed-Point Designer, [Online]. Disponible: https://la.mathworks.com/products/fixed-point-designer.html.

[21]. MathWorks (2012). Fixed-Point Basics, [Online]. Disponible: https://la.mathworks.com/help/fixedpoint/fixed-point-basics-2.html.

[22]. Pablo M. Blanco, “Desarrollo de un sistema de transmisión de vídeo con módulo WiFi”, Trabajo de Fin de grado, Universidad Politécnica de Madrid, septiembre, 2016.

[23]. Eusebio de la F. Lopez, Félix M. Trespaderne. “Visión artificial industrial: Procesamiento de imágenes para inspección automática y robótica”, 1era Edición, Universidad de Valladolid, 2012.

[24]. Atan2, “Definition the function atan2”. Visitado: 20-07-2018. On line: https://en.wikipedia.org/wiki/Atan2

[25]. Simon J. Bleiker, “Performance Analysis of Nanoelectromechanical Relay-Based Field-Programmable Gate Arrays”. Vistado 20-05-2018. On line: https://www.researchgate.net/publication/323820898_Performance_Analysis_of_Nanoelectromechanical_Relay-Based_Field-Programmable_Gate_Arrays.

ÍNDICE DE FIGURAS

Capítulo 2

Capítulo 3

Capítulo 4

Capítulo 5

Capítulo 6

ÍNDICE DE TABLAS