2018

ESTUDIO DEL MUNDO DE LAS FPGAS COMPARATIVAS E IMPLEMENTACION

DAROCH MONTOYA, DANIEL

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA SEDE
CONCEPCION- REY BALDUINO DE BELGICA

Estudio del mundo de las FPGAs, comparativas e implementación

Trabajo de Titulación para optar al Título de Técnico Universitario en Automatización y Control

Alumnos:
Daniel Hernán Daroch Montoya
Salomón Damián Antiñire Monje

Profesor Guía:
Helmut Contreras Novoa

2018

DEDICATORIA

Dedico este proyecto a mi familia que me ha brindado un apoyo fundamental durante toda esta experiencia, por su entrega desinteresada y su infinito amor. A mis compañeros y amigos por los buenos momentos vividos. ¡Muchas Gracias!

Salomón Damián Antiñire Monje

RESUMEN

El presente trabajo tiene el propósito de realizar un estudio sobre las FPGAs reconociendo sus aspectos más importantes a tener en cuenta si una persona quiere empezar a trabajar con esta tecnología.

Esto se conseguirá mediante la inducción a su historia para observar el desarrollo a través del tiempo, reconociendo la arquitectura interna típica de las FPGA, conociendo sus aplicaciones y principales beneficios, estableciendo diferentes tipos de comparativas para tener una visión más clara de esta tecnología y realizando una introducción a uno de los lenguajes de descripción de hardware.

Posteriormente se procede a la segunda etapa de este trabajo donde se refiere al trabajo físico con la placa Spartan-3e Starter Board a disposición, cómo se trabaja con ella y la información correspondiente a su implementación en un proceso.

INDICE DE CONTENIDOS

INDICE DE GRAFICOS

INDICE DE FIGURAS

INDICE DE TABLAS

SIGLA Y SIMBOLOGIA

SIGLA

FPGA	:	Field Programmable Gate Array (Matriz de Puertas Programables).
ASIC	:	Application Specific Integrated Circuit (Circuito Integrado de Aplicación Específica).
HDL	:	Hardware Description Language (Lenguaje de Descripción de Hardware).
IEEE	:	Institute of Electrical and Electronics Engineers (Instituto de Ingeniería Electrica y Electrónica).
PLD	:	Programmable Logic Device (Dispositivo Lógico Programable).
CLB	:	Configurable Logic Block (Bloque Lógico Configurable).
PROM	:	Programmable Read-Only Memory (Memoria Programable de Sólo Lectura).
EPROM	:	Erasable Programmable Read-Only Memory (ROM Programable Borrable).
ROM	:	Read-Only Memory (Memoria de Sólo Lectura).
LUT	:	Look-Up Table (Tabla de Búsqueda).
SRAM	:	Static Random Acces Memory (RAM Estática).
RAM	:	Random Access Memory (Memoria de Acceso Aleatorio).
BRAM	:	Block Random Access Memory (Bloque de RAM).
GPU	:	Graphics Processing Unit (Unidad de Procesamiento Gráfico).
CPU	:	Central Processing Unit (Unidad Central de Procesamiento).
IOB	:	Input/Output Block (Bloque de Entrada/Salida).
DCM	:	Digital Clock Manager (Administrador de Reloj Digital).
DLL	:	Delay Locked Loop (Bucle Bloqueado por Retardo).
IA	:	Artificial Intelligence (Inteligencia Artificial).
IC	:	Integrated Circuit (Circuito Integrado).
VHSIC	:	Very High Speed Integrated Circuit (Circuito Integrado de Muy Alta

Velocidad).

DDR : Double Data Rate (Doble Velocidad de Transmisión de Datos).

JTAG : Join Test Action Group (Grupo de Acción de Pruebas Conjuntas).

SPI : Serial Peripheral Interface (Interfaz Periférica Serial).

LCD : Liquid Crystal Display (Pantalla de Cristal Líquido).

USB : Universal Serial Bus (Bus Serial Universal).

SIMBOLOGÍA (De acuerdo con lo estipulado por el SI)

Hz : Hertz

V : Volts

s : Segundos

INTRODUCCIÓN

En la actualidad existe un panorama de cambios en el mercado mundial, donde la libre competencia crea la necesidad de adecuar eficientemente las industrias a fin de satisfacer los retos que se presenten en los posteriores años. Una de las alternativas, que se utilizan en el campo industrial para enfrentar estos retos, es la integración de elementos que permitan la automatización de equipos y sistemas de una forma adecuada.

Hace unos años atrás todo se centraba en usar los microcontroladores para realizar las tareas de automatización ya que presentaban grandes ventajas como ser reprogramables, aunque son más lentos que un ASIC ofrecen una versatilidad al momento de trabajar con ellos.
Pero últimamente hay una tecnología que está cobrando fuerza pese a no ser reciente diversos factores están provocando que se comiencen a popularizar.

Estamos hablando de las FPGA o matriz de puertas programables (en inglés “Field Programmable Gate Array”) es un dispositivo programable que en su interior contiene celdas de lógica cuya interconexión y funcionalidad puede ser configurada mediante un lenguaje de descripción de hardware (HDL).

En la carrera de Técnico Universitario en Automatización y Control nace la necesidad de abordar estas nuevas tecnologías que vienen en auge y prometen ser importantes en los próximos años, debido a distintos factores esto no es así y se busca hacer de este trabajo de título un método de inducción a esta tecnología.

Para ello se realizará un estudio de su historia y aspectos más importantes tanto de hardware como software, además del estudio se realizará una inducción al trabajo con las FPGA, para concluir al final con un proyecto donde demostrar su efectividad y su versatilidad con respecto a otras tecnologías.

OBJETIVO GENERAL

- Efectuar un estudio y montaje con los cuales inducir al lector en el mundo de las FPGA, además de dar una visión de la utilidad de esta tecnología.

OBJETIVOS ESPECÍFICOS

- Introducir al lector con un instructivo de las FPGA.
- Conseguir información precisa y de utilidad sobre dicha tecnología.
- Plantear comparativas de las FPGA para tener una visión más clara.
- Obtener instructivo del trabajo con la Placa.
- Conseguir realizar una implementación de la Placa Spartan-3e Starter Board donde evidenciar la utilidad de esta tecnología.

ALCANCE

- Generar el escrito para la introducción de lleno a la tecnología de las FPGA.
- Implementación de la placa FPGA Spartan-3E Starter Board.

METODOLOGÍA DE TRABAJO

Para ello se realizará un estudio donde contenga los aspectos más relevantes tanto de software como de hardware incluyendo por supuesto el desarrollo de su historia, además del estudio se realizará una inducción al trabajo con las FPGA, para terminar con un proyecto donde demostrar su efectividad y su versatilidad con respecto a otras tecnologías.

Con el escrito que vamos a generar va a resultar sumamente fácil poder meterse de lleno en el mundo de las FPGA y para que en un futuro cualquier estudiante pueda tomar el tema y desarrollarlo en mayor profundidad.

Si comparas dos cosas desconocidas es realmente difícil tener una clara apreciación de las ventajas de una cosa u otra.

Se establecerá este tipo de comparativa debido a que los microcontroladores ya son conocidos para cualquier persona dentro del área de la tecnología, y esto hace mucho más fácil tener una visión de sus ventajas y desventajas, pero no va a bastar solo con una idea teórica de sus ventajas por eso surge lo del montaje

Vamos a crear un documento con la información necesaria sobre las FPGA para que un técnico de la carrera sea capaz de usarla.

CAPÍTULO 1: ESTUDIO DE LAS FPGAS

1. HISTORIA DE LAS FPGA

1.1. RESUMEN

La FPGA desde sus comienzos ha pasado por diversas modificaciones que han dado paso al producto que conocemos hoy en día, en este capítulo se revisará el desarrollo de las FPGA desde su creación hasta la situación de las mismas en este 2018.

Se considera que las FPGA han pasado por 3 etapas o "eras" las cuales consideran un gran cambio que ha ocurrido en ese periodo para las fabricantes de estas tarjetas, las eras consideradas abarcan desde 1984 hasta 2007 cuyo periodo fue continuamente cambiante para el sector.

Tenemos "La Era de la Invención", La Era de la Expansión" y "La Era de la Acumulación". Si vemos a las FPGA desde su creación han aumentado su capacidad en un factor de 10000 aproximadamente y en su rendimiento en un factor de 100, así como también su costo y la energía necesaria para la operación han disminuido en un factor de 1000.

En la imagen se puede ver los atributos de las FPGA relativos a 1988 publicados por Xilinx donde se consigue apreciar la evolución de la tecnología de FPGA. La capacidad es el número de céldas lógicas, la velocidad es el rendimiento, el precio está considerado por cada celda lógica y la energía también por celda lógica. La evolución que han tenido estas tarjetas en más de 30 años vamos a verla lo más detalladamente en éste capítulo.

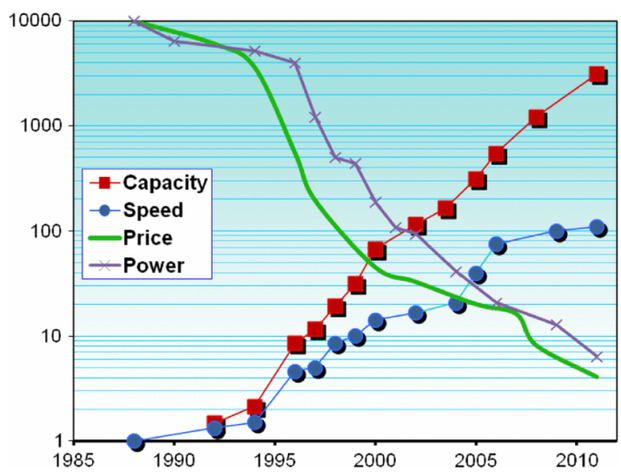


Gráfico 1-1. Atributos de FPGA de Xilinx relativos a 1988.
Fuente: Stephen M. Trimberger. Tres edades de los FPGA: una retrospectiva de los primeros treinta años de la tecnología FPGA. Proceedings of the IEEE, Volumen 103 (3): 318. 2015. ISSN 0018-9219.

La continua modificación a la que se vio sometido el sector se vieron impulsados por el alza de la tecnología de proceso, también la Ley de Moore y sus efectos han impulsado cambios cualitativos en la FPGA.

Toda la información de software de las FPGA es privada, los fabricantes tienen control total de ellas por eso en este capítulo también se incluye un hito importante que abrió estas tecnologías al uso libre de la humanidad, este fue el llamado proyecto IceStorm.

1.2. ERA DE LA INVENCIÓN (1984 – 1991)

Las FPGAs fueron creadas en 1984 por Ross Freeman y Bernard Vonderschmidt, cofundadores de Xilinx. Las FPGA son resultado de la unión de los PLDs y los ASIC.

La primera FPGA creada fue el Xilinx XC2064 cuyas características eran: 64 bloques lógicos complejos (CLB), en una cuadrícula de 8x8, 18MHz, 58 I/O pins, tenía 1200 puertas y un circuito integrado muy grande. La tecnología de proceso de 2,5 micrones apenas pudo producirse. La contención de costos (procedimiento que consiste en mantener los costes de una organización dentro de un presupuesto establecido restringiendo los gastos para tratar de conseguir los objetivos financieros previamente fijados) fue indispensable para el éxito de las FPGA.

El tamaño de matriz y costo por función fue algo vital. El XC2064 era realmente algo insignificante comparado con los chips de FPGA que hay en la actualidad, pero en ese entonces tenía un valor de cientos de dólares.

Las FPGA basados en memoria estáticas eran reprogramables y requerían una PROM externa para almacenar la programación cuando la alimentación estaba apagada. La reprogramación no se consideraba un activo y Xilinx lo minimizó para evitar las preocupaciones de los clientes sobre lo que sucedió con su lógica luego de desalimentar.

El antifusible desarrollado por Actel era una estructura de transistor único. El ahorro de área de los antifusibles sobre las celdas de memoria era ineludible. Actel lo llevó al mercado y en 1990 el FPGA de mayor capacidad fue el Actel 1280. Quicklogic y Crosspoint siguieron a Actel y también desarrollaron dispositivos basados en las ventajas de la tecnología antifusibles.

Las arquitecturas basadas en LUT de cuatro entradas de Xilinx se consideraron de “grano grueso” (tipo de paralelismo en la arquitectura de un objeto cada hilo actúa independientemente de los demás), pero el análisis de las listas de redes mostró que muchas configuraciones LUT no estaban en uso desperdiciando espacio. Varias compañías implementaron arquitecturas de grano fino (a diferencia del grano grueso en este caso los hilos actúan en consecuencia de los demás) con funciones fijas para eliminar el desperdicio. Así como surgieron diferentes alternativas de parte de cada empresa donde cada una presentaba su propia solución al tema.

A finales de la era de la invención los cables largos y lentos se reemplazaron por conexiones cortas entre bloques adyacentes que se podían unir según las necesidades mediante la programación para formar rutas de enrutamiento más largas.

La Era de la Invención terminó con un desgaste brutal en el negocio de las FPGA. Muchas de las compañías simplemente desaparecieron, otros vendieron sus activos al salir del negocio de las FPGA. Hubo cambios importantes en la tecnología y las compañías que no aprovecharon estos cambios no pudieron competir. Los cambios cuantitativos debido a la Ley de Moore dieron como resultado cambios cualitativos en los FPGA construidos con tecnología de los semiconductores.

1.3. ERA DE LA EXPANSIÓN (1992 - 1999)

La era de la expansión podemos resumirla como el periodo donde las fabricantes de FPGA comenzaron a abordar el problema del tamaño, como resultado la complejidad del diseño se volvió clave. Esto significó un gran cambio para las FPGA que había hasta aquel momento ya que provocó el nacimiento del Lenguaje de Descripción de Hardware (HDL), el método utilizado anteriormente eran herramientas en las cuales se designaban lugar y ruta.

Eso es, en resumen, pero viendo con más detalle lo que sucedió comenzamos con señalar el desarrollo rápido que tuvo la Ley de Moore a lo largo de la década del 1990 duplicando la cantidad de transistores cada dos años. Las compañías de FPGA pioneras del modelo de negocio fabbles (fabricante de semiconductores que carece de fábrica para las obleas de silicio) por lo cual las compañías no podían conseguir tecnología de silicio de vanguardia para esos años. Cada nueva generación de silicio duplicó el número de transistores disponibles, lo que duplico el tamaño del FPGA más grande posible y redujo a la mitad el costo por función. En la imagen se refleja el crecimiento de las LUT y los cables de interconexión.

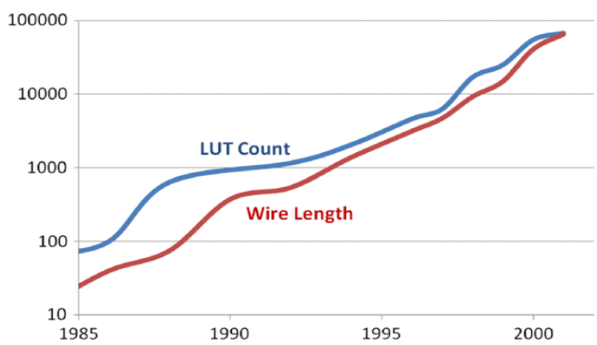


Gráfico 1-2. Crecimiento de las LUT de FPGA y los cables de interconexión.
Fuente: Stephen M. Trimberger. Tres edades de los FPGA: una retrospectiva de los primeros treinta años de la tecnología FPGA. Proceedings of the IEEE, Volumen 103 (3): 322. 2015. ISSN 0018-9219.

En la era de la expansión, la capacidad de los dispositivos FPGA aumentó rápidamente a medida que disminuían los costos. Las aplicaciones FPGA se hicieron demasiado grandes para el diseño manual. En 1992, el buque insignia de Xilinx XC4010 entregó un proyecto de

máximo 10000 puertas. En 1999 el Virtex XCV1000 fue calificado en un millón. A principios de la década de 1990 se prefería la ubicación y el enrutamiento automáticos, pero a finales de la década la síntesis automatizada, la colocación y el enrutamiento eran pasos necesarios en el proceso de diseño.

Ahora la vida de una empresa de FPGA dependía de la capacidad de las herramientas de automatización de diseño para apuntar al dispositivo. Aquellas empresas FPGA que controlaban su software controlaban su futuro. Las herramientas de diseño automatizadas requerían arquitecturas fáciles de automatizar, arquitecturas con recursos de interconexión regulares y abundantes para simplificar la toma de decisiones algorítmicas.

Los bloques lógicos en la era de la invención eran pequeños y simples, esto resultaba atractivo porque su retraso lógico era corto. Las LUTs eran poco eficiente ya que no se utilizaban todas las celdas de memorias en ellas. Para funciones más grandes, la necesidad de conectar varios bloques pequeños genera una mayor demanda en la interconexión. En la Era de la Expansión, no solo había más bloques lógicos, sino que los bloques mismos se volvieron más complejos.

El rápido progreso de la Ley de Moore produjo una necesidad de estar a la vanguardia de la tecnología de procesos. Ante esta necesidad los proveedores de FPGA que no podían adaptar sus tecnologías a las necesidades de mejora se vieron en una gran desventaja estructural, este fue el caso de las memorias no volátiles (EPROM, Flash y antifusible).

De estas tecnologías sólo el antifusible fue considerado como una opción válida, aunque para cuando se consiguió adaptar tenían que ser el doble de eficientes que las SRAM solo para mantener la paridad del producto. Además, la falta de reprogramación se volvió una necesidad al momento de que los clientes utilizaban las FPGA SRAM comenzaron a valorar las ventajas de la programabilidad en el sistema y la actualización del hardware.

Las arquitecturas basadas en LUT eran objetivos fáciles para las herramientas de síntesis. En un principio los proveedores de síntesis señalaban que una FPGA era incompatible con la síntesis. Esto ocurrió en un principio porque las herramientas de síntesis eran desarrolladas inicialmente para apuntar a los ASIC que tenían una arquitectura diferente a las FPGA. Esto cambió a mediados de 1990 cuando los mapeadores LUT específicos explotaron la simplicidad de mapear funciones arbitrarias en LUTs.

La programación de celdas de memoria distribuida permitió la libertad arquitectónica y dio a los proveedores de FPGA acceso casi universal a la tecnología de proceso. Utilizar LUTs para la implementación lógica alivió la carga de la interconexión.

La Ley de Moore aumentó rápidamente la capacidad de los FPGA, lo que generó una demanda de automatización del diseño y permitió una segmentación de interconexión más

prolongada. Las arquitecturas demasiado eficientes que no podían ser automatizadas efectivamente simplemente desaparecieron.

1.3.1. Marketing de Xilinx para el futuro

La Era de la Expansión se entendía bien en el negocio FPGA. Los proveedores de FPGA perseguían agresivamente la tecnología de procesos como la solución a sus problemas de tamaño, rendimiento y capacidad. Cada nueva generación de procesos trajo consigo numerosas aplicaciones nuevas.

Podemos observar en la imagen el Virtex 1000, el FPGA más grande disponible en ese momento, es representado como el pequeño rectángulo negro en la parte inferior izquierda. La imagen muestra la expectativa de que la era de la expansión continuara sin disminuir, aumentando el número de puertas a 50 millones en los siguientes cinco años. Esto no sucedió, a pesar del progreso inquebrantable de la Ley de Moore. En la era de la acumulación veremos la razón.

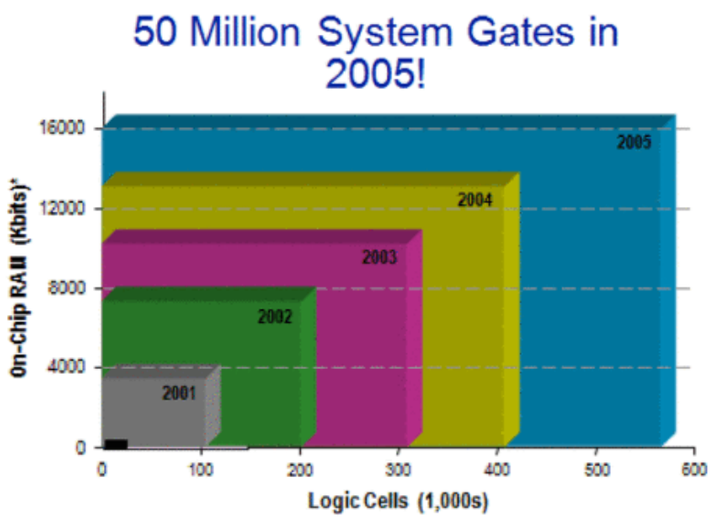


Gráfico 1-3. Xilinx Marketing 2000.
Fuente: Stephen M. Trimberger. Tres edades de los FPGA: una retrospectiva de los primeros treinta años de la tecnología FPGA. Proceedings of the IEEE, Volumen 103 (3): 322. 2015. ISSN 0018-9219.

1.4. ERA DE LA ACUMULACIÓN (2000 - 2007)

Al comienzo del nuevo milenio los FPGA eran componentes comunes de los sistemas digitales. La capacidad y el tamaño del diseño estaban creciendo y las FPGA habían encontrado un gran mercado en la industria de las comunicaciones de datos. La caída de las punto-com a principios de la década de 2000 creó una necesidad de un menor costo.

Al igual que en la era de la expansión, el ritmo inexorable de la Ley de Moore hizo que los FPGA fueran cada vez más grandes. Ahora eran más grandes que el tamaño del problema típico. No hay nada malo en tener una capacidad superior a la que se necesita, pero tampoco hay nada particularmente virtuoso en ella. Como resultado, los clientes no estaban dispuestos a pagar una gran prima por el mayor FPGA.

El aumento de la capacidad por sí solo tampoco fue suficiente para garantizar el crecimiento del mercado. El hecho que había generado un producto exitoso en la era de la expansión atrajo menos clientes en los años siguientes.

Los proveedores de FPGA tomaron decisiones diferentes para la gama baja y alta del mercado. En el caso de la gama baja se centraron en la eficiencia y produjeron familias de "bajo costo" de menor capacidad y rendimiento donde podemos encontrar: Spartan de Xilinx, Cyclone de Altera y EC/ECP de Lattice.

En el caso de la gama alta la decisión fue facilitar que los clientes llenen las espaciosas FPGAs. Produjeron bibliotecas de soft logic para funciones importantes. Las funciones lógicas más notables fueron los microprocesadores (Xilinx MicroBlaze y Altera Nios) controladores de memorias y varios protocolos de comunicaciones.

Las características de los diseños cambiaron en los años 2000. Los grandes FPGA admitían grandes diseños que eran subsistemas completos. A medida que el FPGA creció como una fracción de la lógica general del sistema del cliente, su costo y potencia aumentaron en consecuencia. Estos problemas se volvieron mucho más importantes de lo que eran en la Era de la Expansión.

La presión para adherirse a los estándares, disminuir el costo y la potencia llevó a un cambio en la estrategia de arquitectura de simplemente agregar lógica programable y montar la Ley de Moore, como se hizo en la Era de la Expansión, a agregar bloques lógicos dedicados. Estos bloques contenían memorias grandes, microprocesadores, multiplicadores, e/s flexibles y transceptores sincrónicos de origen. Para las aplicaciones que los usaron, redujeron la sobrecarga de programación en área, rendimiento, potencia y esfuerzo de diseño.

El resultado de estos cambios se ve en la "Plataforma FPGA" observada en la imagen, si se compara con la imagen que se tenía para el futuro de las FPGA en el final de la era de la

expansión se observa que ya no existe la referencia a los millones de puertas, esto cambia su enfoque hacia los bloques dedicados de alto rendimiento.

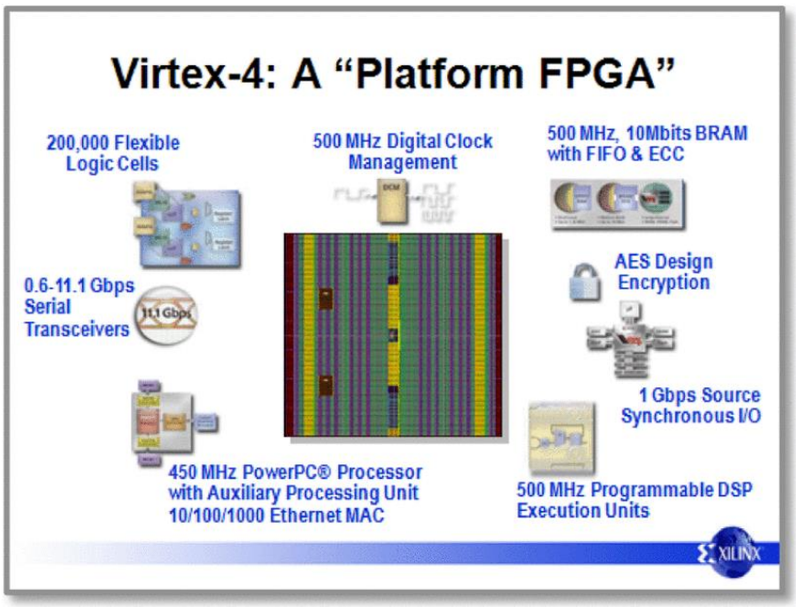


Figura 1-1. Xilinx Marketing 2005.
Fuente: Stephen M. Trimberger. Tres edades de los FPGA: una retrospectiva de los primeros treinta años de la tecnología FPGA. Proceedings of the IEEE, Volumen 103 (3): 327. 2015. ISSN 0018-9219.

Pero con esto volvemos al mismo problema que había anteriormente ¿Qué sucede con los clientes que no necesita todas las herramientas que se les ofrecen? Al principio los proveedores de FPGA intentaron asegurarse de que esas funciones pudieran usarse para la lógica si no se necesitaban para su propósito principal. Pero estas medidas fueron consideradas como poco importantes. Con esto se aceptó el hecho de que habría funciones desperdiciadas, un vicepresidente de Xilinx comentó que proporcionaría cuatro procesadores Power-PC en una FPGA y no le importaba si los clientes no utilizaban ninguno de ellos. "Les damos los procesadores de forma gratuita".

1.5. EL DESPUÉS DE LAS ERAS

Al final de la Era de la Acumulación, los FPGA no eran matrices de puertas, sino colecciones de bloques acumulados integrados con la lógica programable. Todavía eran programables, pero no estaban restringidos a la lógica programable. Las dimensiones adicionales de la capacidad de programación adquiridas en la era de la acumulación agregaron una carga de diseño. El esfuerzo de diseño, una ventaja para los FPGA en su competencia con ASIC, fue una desventaja en la competencia con los procesadores de múltiples núcleos recién llegados y las GPU.

Las presiones continuaron aumentando en los desarrolladores de FPGA. La desaceleración económica que comenzó en 2008 continuó impulsando el deseo de un menor costo.

Los nuevos FPGA tenían nuevos requisitos de diseño, era programables por hardware y por software. El microprocesador ahora incluye un entorno con cachés, buses, Network-on-Chip y periféricos. El software incluye sistemas operativos, compiladores y middleware. Era un ecosistema completo a diferencia de un bloque de funciones integrados que contenía antiguamente.

Los sistemas completos requieren interfaces de señal mixta para la interfaz en el mundo real. Estos también monitorean el voltaje y la temperatura. Todo esto es necesario para que el FPGA sea un sistema completo en un chip. Como resultado, los FPGA han crecido hasta el punto en que la matriz de la puerta lógica suele ser menos de la mitad del área. En el camino, las herramientas de diseño FPGA han crecido para abarcar el amplio espectro de problemas de diseño.

Costo, la capacidad y la velocidad eran precisamente aquellos atributos en los cuales los FPGA estaban en desventaja con respecto al ASIC en los años 80 y 90. Sin embargo, prosperaron. Un enfoque limitado en esos atributos sería erróneo, al igual que el enfoque limitado de las compañías ASIC en los años 90 los llevó a subestimar los FPGA. La programabilidad le dio a los FPGA una ventaja a pesar de sus inconvenientes.

Los conjuntos de herramientas FPGA modernas incluyen compilación de síntesis de alto nivel desde C, Cuda y OpenCL hasta lógica o microprocesadores integrados. Las bibliotecas de funciones lógicas y de procesamiento proporcionadas por los proveedores cubren los costos de diseño. Las funciones de diseño del equipo, incluido el control de construcción, están integradas en los sistemas de diseño FPGA. Algunas capacidades son creadas por los propios proveedores, otras son parte del creciente ecosistema de FPGA.

1.6. ACTUALIDAD

Después de ver toda la historia de las FPGA la pregunta es ¿Cómo están actualmente? Observando que grandes empresas han puesto sus ojos sobre estos chips siendo la mayor noticia la compra de Altera por parte de Intel en 16700 millones de dólares demuestra la importancia que se espera tomen las FPGA en un futuro no muy lejano.

La competencia en el mercado tecnológico por conseguir el mejor rendimiento ha llevado a buscar alternativas diferentes de las ya utilizadas, hay algunas que ya se han dado cuenta que las FPGA pueden volver a ser consideradas después del bajón que tuvieron posterior a la era de la expansión tanto así que Microsoft también centró sus ojos en esta tecnología con el Project Catapult cuyo objetivo era explorar alternativas para obtener una mayor potencia informática para desarrollar nuevos procesadores. Según declaraciones del líder el proyecto ellos estaban explorando arquitecturas entre las que se encontraban la GPU, ASIC y FPGA. Siendo la matriz de puerta programable la elegida ya que ofrece la combinación única de velocidad, capacidad de programación y flexibilidad. Amazon, Bing, Office 365 y Azure son otros que se han unido al uso de las FPGA para mejorar sus servicios.

Microsoft ha presentado este año el primer modelo de hardware acelerado donde presentan como una de las características su capacidad de actuar como un acelerador de cómputo local, un procesador en línea o un acelerador remoto para computación distribuida. Si las cosas salen tan bien como se prevea en el futuro las FPGA pueden convertirse en un elemente esencial en cada equipo de procesamiento.

1.7. PROYECTO ICESTORM

Las FPGA son una tecnología privada donde existen pocos fabricantes y el software es de su total pertenencia, las empresas de FPGA buscaban tener el control total sobre ellas. Hasta hace poco se desconocía totalmente el formato del bitstream y del software en general, pero esto cambió en marzo del 2015 cuando Clifford Wolf un profesor universitario austriaco realizó ingeniería inversa a la familia ICE40 de Lattice y liberó la primera toolchain de herramientas para a través de Verilog conseguir el bitstream y su posterior carga para la configuración de la FPGA. Ahora las empresas de FPGA ya no poseían el poder total sobre las aplicaciones para los chips cada usuario sólo necesitaba comprar la tarjeta y conseguía control total para usar todas sus posibilidades.

Aunque esto es solo el primer paso ya que las tarjetas que se encuentran liberadas son pequeñas y su uso se limita a experimentos domésticos o de laboratorio.

CAPÍTULO 2: DESCRIPCION DE LA FPGA

2. ¿QUÉ ES UNA FPGA? Y SU UTILIDAD

2.1. DESCRIPCIÓN

Una FPGA es un chip que contiene componentes lógicos programables e interconexiones programables entre ellos. Sus componentes lógicos pueden ser programados como compuertas lógicas (AND, OR, XOR, NOT), o funciones combinacionales más complejas como decodificadores o también puede programarse como simples funciones matemáticas. También incluyen elementos de memoria, estos pueden ser simples flip-flops o bloques de memoria más complejos.

La FPGA tiene una arquitectura que se basa en CLBs, donde se realizan las funciones lógicas. Las FPGA tienen interconexiones programables que son las que construyen los circuitos que hemos desarrollado esas interconexiones son activadas a través del bitstream que le designa el estado on/off al cable.

Una jerarquía de interconexiones programables permite a los bloques lógicos de un FPGA ser interconectados según la necesidad del diseñador del sistema. Ese diseño se realiza a través de un software en un lenguaje de descripción de hardware luego se sintetiza, se emplaza y enruta para posteriormente cargar un bitstream en la FPGA.

2.2. EL BISTREAM

Es la información para los bits de configuración se transmite por un bus serial que contiene toda la información para cada punto de configuración, esta tira de bits se llama bitstream cuyo origen viene desde el software.

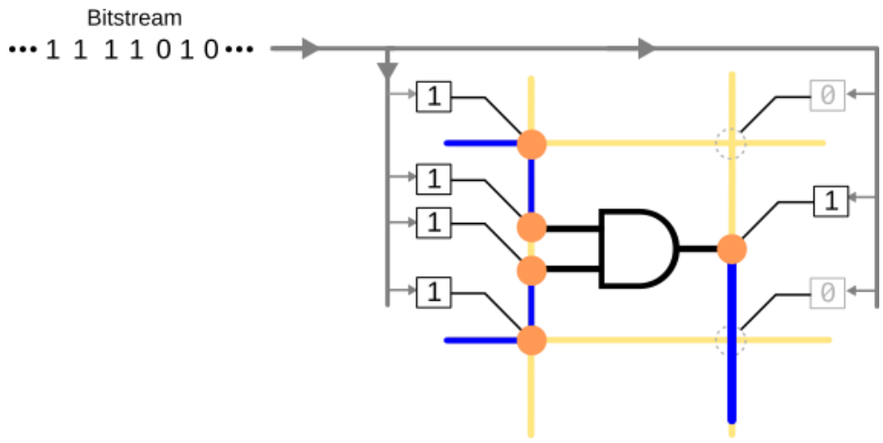


Figura 2-1. Esquema bitstream.
Fuente: FPGAwars. Explorando el lado libre de las FPGAS. [en línea]. <http://fpgawars.github.io/>. [Consulta: 14 de N0oviembre de 2018].

2.2.1. Generación del bitstream

Vimos anteriormente que el bitstream es un tramo de datos que se transportan por un bus serial a cada punto de configuración dentro del chip. Pero la pregunta es ¿Cómo se generan estos bitstream de qué manera nosotros definimos la conexión de cientos de miles puntos configurables? La respuesta es simple, esto se realiza a través de las siguientes etapas:

- Primero se diseñan utilizando algún lenguaje de descripción de hardware donde los más populares son Verilog y VHDL.
- Luego la herramienta de síntesis concluye a partir de la descripción los elementos hardware básicos, y obtiene un fichero netlist que describe las uniones entre ellos (esto no depende del modelo de FPGA). Podríamos decir que la síntesis es el proceso mediante el cual se designa la factibilidad de lo descrito y los tipos de componentes que serán necesarios para hacerlo.
- Luego está la fase de emplazado y enrutado donde los componentes del fichero netlist se asocian a los elementos físicos de la FPGA (aquí va a depender del modelo ya que confiere a la asignación física), luego se determina la colocación y se realiza el enrutado. Luego toda la información se adapta al formato del bitstream para poder enviarlo a la placa.

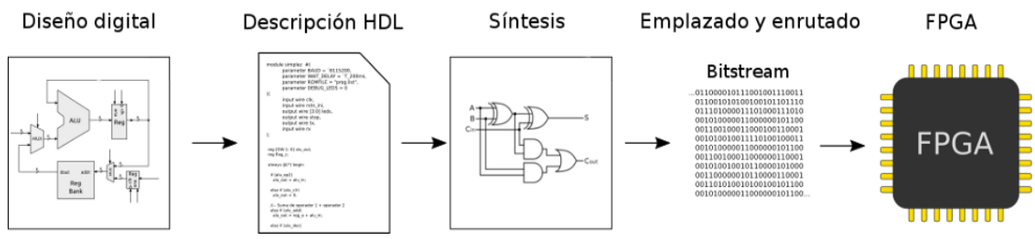


Figura 2-2. Generación del bitstream.
Fuente: FPGAwars. Explorando el lado libre de las FPGAS. [en línea]. <http://fpgawars.github.io/>. [Consulta: 14 de Noviembre de 2018].

2.3. ARQUITECTURA DE UNA FPGA

Es complicado referirse a este tema como un concepto general, ya que cada compañía presenta sus propias arquitecturas y se encuentran en constante desarrollo. Básicamente en una FPGA la lógica se divide en un gran número de bloques lógicos, estos bloques se encuentran distribuidos a través de todo el chip en una infinidad de interconexiones programables, en los márgenes del chip se forma una fila de IOBs. En esta sección se describirá la arquitectura de la familia Spartan 3E de Xilinx, debido a que será la FPGA a utilizar para la implementación.

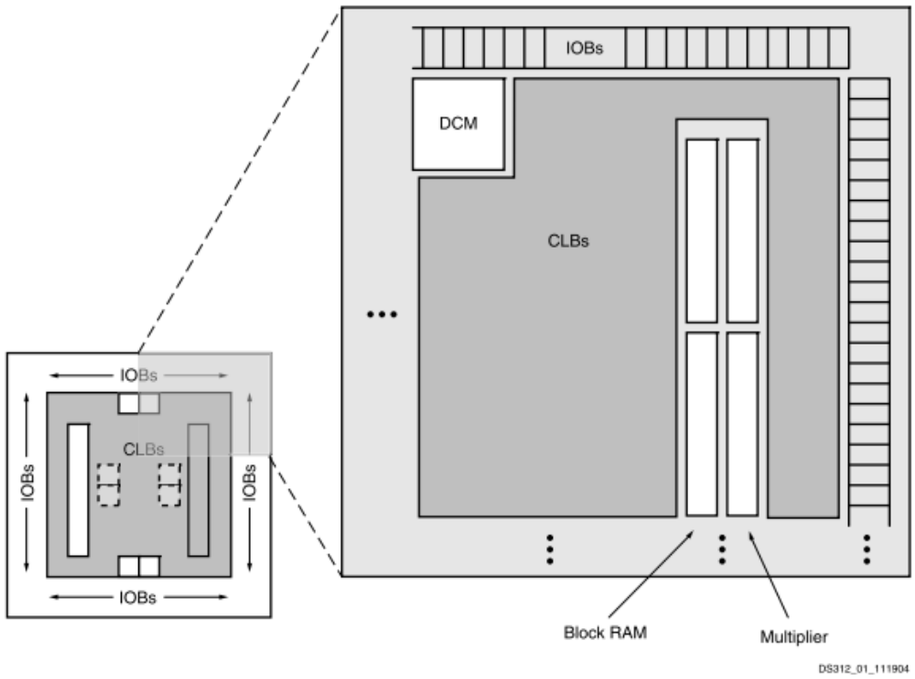


Figura 2-3. Arquitectura Spartan-3E.
Fuente: Xilinx. Spartan-3E FPGA Family Data Sheet. [PDF]. Estados Unidos, 19 de Julio de 2013. [Consulta: 14 de Noviembre de 2018].

Se puede identificar cuatro elementos que son a grandes rasgos los componentes más importantes y ahora veremos más en profundidad que es cada uno.

CLB: Un CLB es la pieza fundamental de un FPGA y es lo que le da su capacidad para asumir diferentes configuraciones de hardware. Un FPGA en su forma más básica es un chip de CLB, juntos hacen un FPGA. Los muchos miles de estos que se pueden encontrar en los FPGA modernos pueden programarse para realizar virtualmente cualquier función lógica. Un CLB individual consiste en un conjunto de slices.

En el caso de la familia Spartan-3E cada CLB contiene cuatro slices, y cada slice contiene dos tablas de consulta (LUT) para implementar la lógica y dos elementos de almacenamiento dedicados que se pueden usar como flip-flops o latches. Los LUT se pueden usar como una memoria 16x1 o como un registro de desplazamiento de 16 bits, y los multiplexores adicionales y la lógica de transporte simplifican las funciones de lógica amplia y aritmética. La lógica de propósito más general en un diseño se asigna automáticamente a los recursos de la porción en los CLB.

Slices: Cada CLB comprende cuatro slices interconectados. Estos slices se agrupan en pares. Cada par está organizado como una columna con una cadena de acarreo independiente. El par izquierdo admite funciones lógicas y de memoria y sus slices se denominan SLICEM. El par derecho admite solo la lógica y sus slices se denominan SLICEL. Por lo tanto, la mitad de las LUT admiten tanto la lógica como la memoria mientras que la mitad solo admite la lógica, y los dos tipos se alternan en las columnas de la matriz.

En resumen, un slice incluye dos generadores de funciones LUT y dos elementos de almacenamiento, junto con lógica adicional, como se verá en la siguiente figura.

Tanto SLICEM como SLICEL tienen los siguientes elementos en común para proporcionar funciones de lógica, aritmética y ROM:

• Dos generadores de función LUT de 4 entradas, F y G.

• Dos elementos de almacenamiento.

• Dos multiplexores de función amplia, F5MUX y FiMUX.

• Lógica de acarreo y aritmética.

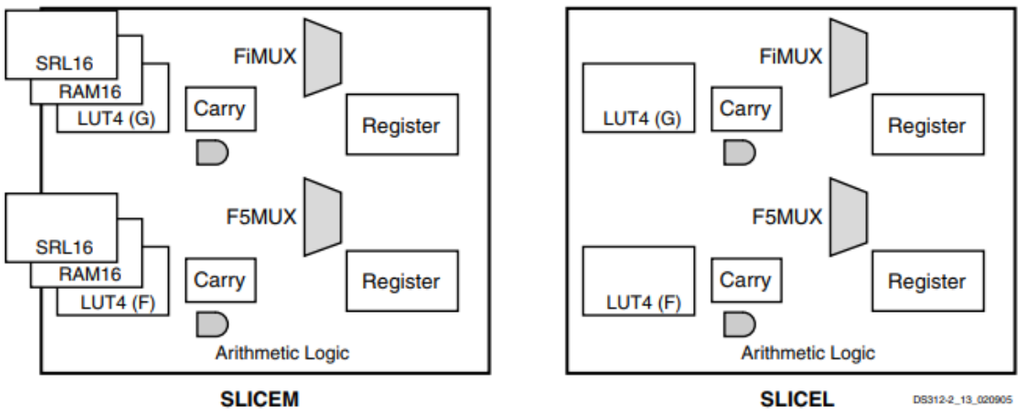


Figura 2-4. Slices Spartan-3E.
Fuente: Xilinx. Spartan-3E FPGA Family Data Sheet. [PDF]. Estados Unidos, 19 de Julio de 2013. [Consulta: 14 de Noviembre de 2018].

LUT (Look-up tables): La tabla de consulta o LUT es un generador de funciones basado en RAM y es el recurso principal para implementar funciones lógicas. Además, las LUT en cada par de SLICEM se pueden configurar como RAM distribuida o un registro de desplazamiento de 16 bits. Cada una de las dos LUT (F y G) en un segmento tiene cuatro entradas lógicas (A1-A4) y una sola salida (D). Cualquier operación lógica booleana de cuatro variables se puede implementar en una LUT. Las funciones con más entradas se pueden implementar mediante LUT en cascada.

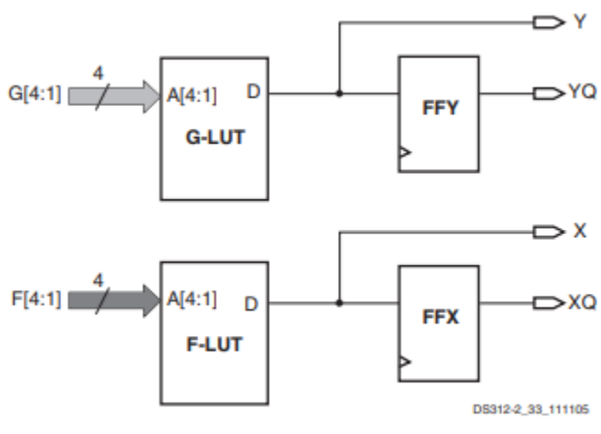


Figura 2-5. LUT Spartan-3E.
Fuente: Xilinx. Spartan-3E FPGA Family Data Sheet. [PDF]. Estados Unidos, 19 de Julio de 2013. [Consulta: 14 de Noviembre de 2018].

Esta tabla de verdad define efectivamente cómo se comporta su lógica combinatoria, que son celdas de memoria SRAM volátiles y multiplexores para seleccionar la salida.
En otras palabras, cualquier LUT puede implementar cualquier comportamiento que se obtenga al interconectar cualquier cantidad de puertas (como AND, NOR, etc.).

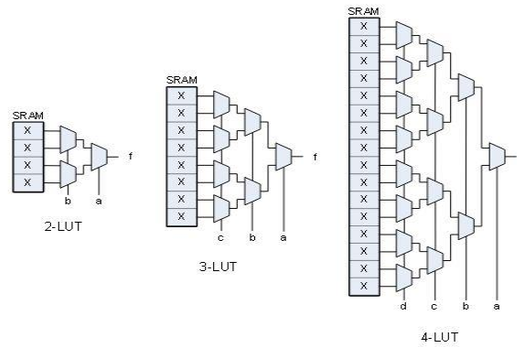


Figura 2-6. Ilustración de una LUT.

BRAM (bloque de memoria de acceso aleatorio): La memoria dedicada en el chip en sí se conoce como BRAM. Si bien cada bloque individualmente tiene un tamaño, estos bloques pueden subdividirse o conectarse en cascada para hacer que BRAM tenga disponibles tamaños más pequeños o más grandes. También se puede realizar una variedad de configuraciones operativas y pueden admitir una funcionalidad especial como la corrección de errores.

Flip-Flop: Es un dispositivo de almacenamiento que puede almacenar un solo bit de información. Cada slice contiene ocho flip-flops.

DCM (Digital Clock Manager): Un DCM es un conjunto en bloques de control integrados a la red de distribución de reloj. La red de distribución de reloj en las FPGA asegura retardos parejos a todos los bloques lógicos de la FPGA.

Hay bloques específicos dedicados al control de reloj, estos son los DLL (Delay Locked Loop). Estos bloques se ocupan de sincronizar el reloj interno al reloj externo del sistema, además controlan el desplazamiento de fase entre los relojes, sincronizan los diferentes dominios de reloj y aseguran un retardo de distribución del reloj pareja para la lógica interna de la FPGA.

El DCM realiza tres funciones principales:

-Eliminación del desalineamiento de la fase cero (skew). Esto se refiere a en qué grado las señales de reloj pueden desviarse del alineamiento de la fase cero. Esto ocurre cuando pequeñas diferencias en los retardos de las rutas causan que la señal de reloj llegue a diferentes puntos del circuito en tiempos diferentes. Este desalineamiento de reloj puede incrementar los requerimientos del tiempo de establecimiento y de retención, lo que puede perjudicar al desempeño de aplicaciones de alta frecuencia. El DCM se encarga de alinear la

salida de la señal de reloj que genera con otra versión de la misma señal que es retroalimentada. Así se establece una relación de cero desfases entre ambas señales.

-Síntesis de frecuencia: Con una señal de reloj de entrada, el DCM puede generar diferentes frecuencias de salida. Ello se logra multiplicando y/o dividiendo la frecuencia del reloj de entrada.

-Corrimiento de fase: Puede producir desfases controlados de la señal de reloj de entrada y con ello producir señal de salida con diferentes fases.

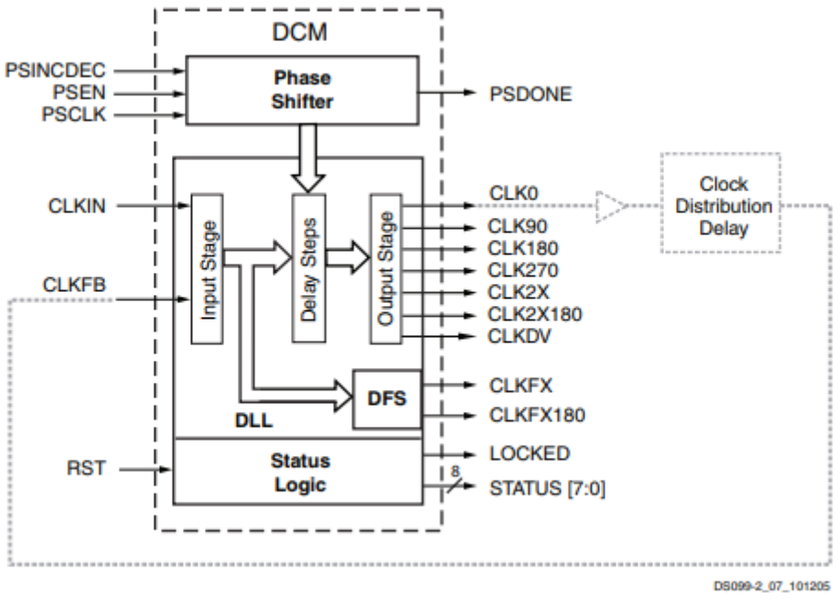


Figura 2-7. Digital Clock Manager Spartan-3E.
Fuente: Xilinx. Spartan-3E FPGA Family Data Sheet. [PDF]. Estados Unidos, 19 de Julio de 2013. [Consulta: 14 de Noviembre de 2018].

Interconexiones: Para que una FPGA pueda implementar los circuitos lógicos programados por el usuario, los bloques lógicos y de entrada/salida no solo deben configurarse adecuadamente, sino que también deben conectarse entre sí. La estructura de interconexión interna de una FPGA consiste en un conjunto de conexiones o trazas que pueden conectarse mediante elementos de paso programables. Las herramientas de "particionado, localización e interconexión" son las encargadas de decidir en qué elementos lógico se implementará la lógica diseñada por el usuario y como deben programarse las interconexiones para que el diseño funcione según las especificaciones de tiempo y retardo que se han definido.

La interconexión es la red programable de vías de señal entre las entradas y salidas de los elementos funcionales dentro del FPGA, como IOB, CLB, DCM y bloque de RAM. Las conexiones a los CLB adyacentes permiten optimizar los diseños al evitar los retardos y la utilización de recursos de la matriz general de interconexionado.

La mayor parte de las señales se conectan a través de la matriz general de interconexionado. Los elementos de esta matriz se encuentran entre los CLB, en los canales de interconexión horizontales y verticales de la FPGA. Permiten hacer la unión entre las trazas horizontales y

verticales y hacia los CLB. A través de ellos se configuran las conexiones entre CLBs no adyacentes y hacia los bloques de entrada/salida.

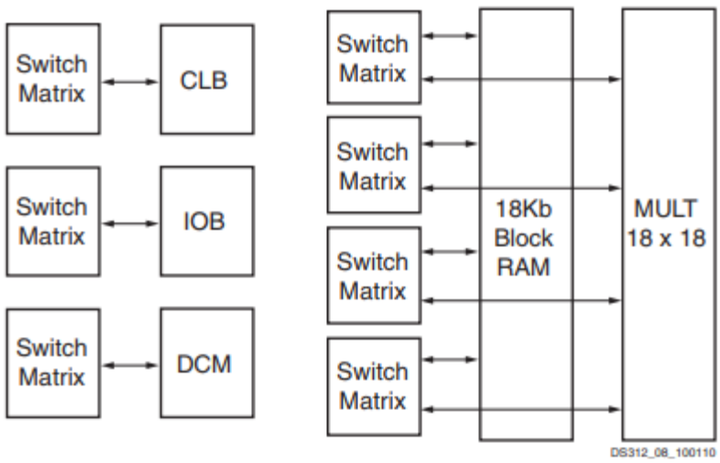


Figura 2-8. Switch Matrix Spartan-3E.
Fuente: Xilinx. Spartan-3E FPGA Family Data Sheet. [PDF]. Estados Unidos, 19 de Julio de 2013. [Consulta: 14 de Noviembre de 2018].

IOB (Input output blocks): El bloque de entrada/salida (IOB) proporciona una interfaz programable, unidireccional o bidireccional entre un pin del paquete y la lógica interna del FPGA.

La siguiente figura es un diagrama simplificado de la estructura interna de la IOB. Hay tres rutas de señal principales dentro de la IOB: la ruta de salida, la ruta de entrada y la ruta de 3 estados. Cada ruta tiene su propio par de elementos de almacenamiento que pueden actuar como registros o latches.

Cada uno tiene sus elementos de almacenamiento que pueden actuar como registros, los cuales permiten controlar mejor la entrada y salida.

La tarea principal de este bloque es el de conducir las señales al nivel apropiado de tensión para que cada pin pueda ser configurado como entrada, salida o bidireccional. Cada bloque de entrada/salida tiene una resistencia de pull-up y pull-down.

Los pines I/O tienen un buffer de salida tri-estado que puede ser controlado por señales internas. La señal de entrada del IOB se dirige al módulo de enrutamiento con el fin de que se encamine al bloque lógico apropiado.

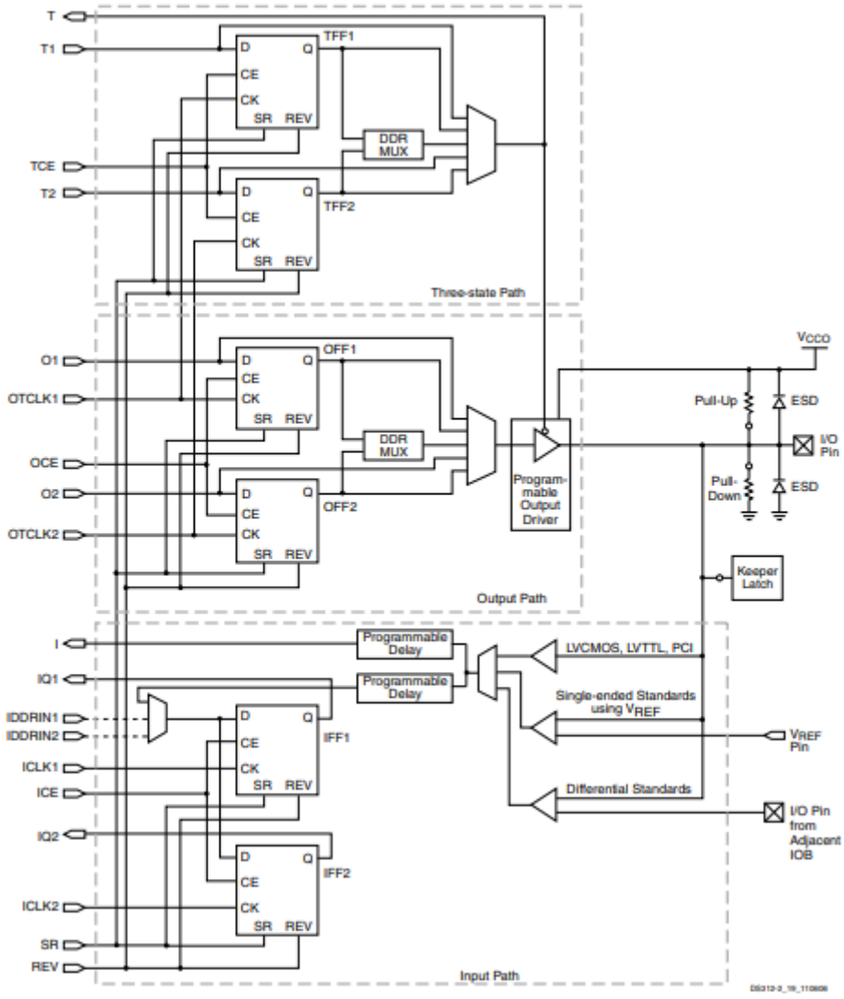


Figura 2-9. Bloques de entrada y salida Spartan-3E.
Fuente: Xilinx. Spartan-3E FPGA Family Data Sheet. [PDF]. Estados Unidos, 19 de Julio de 2013. [Consulta: 14 de Noviembre de 2018].

2.4. BENEFICIOS Y APLICACIONES DE LA FPGA

En esta sección el objetivo es ver algunos de los beneficios de estos chips y sus principales aplicaciones a nivel mundial. A nivel local esta tecnología es bastante desconocida, incluso entre profesionales del área en gran parte sólo habrán escuchado el nombre FPGA sin mayor conocimiento sobre esta tecnología, al ser un país con retraso tecnológico seguramente tardará un par de años más conseguir popularizar estos chips.

A continuación, veremos algunos de los motivos por los cuales se han popularizado:

-El primer punto es el rendimiento, las FPGA se aprovechan del paralelismo del hardware, los FPGAs sobrepasan la potencia de cómputo de los procesadores digitales de señales la ventaja resulta aplastante en este caso, la ejecución secuencial que caracteriza a los dispositivos procesadores no pueden hacer nada contra la velocidad que puede proporcionar una FPGA. El controlar entradas y salidas (E/S) a nivel de hardware ofrece tiempos de

respuesta más veloces y funcionalidad especializada que coincide con los requerimientos de una aplicación.

-Como segundo punto tenemos el tiempo que demora en llegar al mercado, este punto se refiere a que el desarrollo de prototipos tiene una gran flexibilidad y con una velocidad que permite llegar a tiempo al mercado. Se puede probar conceptos o ideas en hardware si tener la necesidad de hacer el largo proceso de fabricación que realiza un ASIC. El aumento en disponibilidad de herramientas de software de alto nivel disminuye la curva de aprendizaje con niveles de abstracción. Estas herramientas frecuentemente incluyen importantes núcleos IP (funciones pre-construidas) para control avanzado y procesamiento de señales.

-Como tercer punto más importante aparece el precio, quizá este punto sea el que genere más dudas debido a los costos de compra que se conocen sobre las FPGA donde el software es bastante caro, pero en comparación con otras tecnologías esto no es realmente cierto, ya que te ofrece un trabajo más eficiente para la labor que se requiere a un precio menor (obviamente hablando a niveles más avanzados). También es importante considerar que la naturaleza programable del silicio implica que no haya precio de fabricación o largos tiempos de ensamblado. Los requerimientos de un sistema van cambiando con el tiempo, y el precio de cambiar incrementalmente lo diseños FPGA es insignificante al compararlo con el precio de implementar cambios en un sistema no reprogramable.

-La fiabilidad de funcionamiento, esto implica que las herramientas de software ofrecen un entorno de programación, mientras los circuitos de un FPGA son una implementación segura de la ejecución de un programa. Los sistemas basados en procesadores suelen implicar varios niveles de separación para ayudar a programar las tareas y compartir los recursos entre varios procesos. El núcleo de un procesador ejecuta una instrucción a la vez y existe el riesgo que en esos sistemas las tareas se obstruyan entre sí. En cambio, un FPGA que no necesita sistema operativo, ejecutan tareas de forma paralela sin riesgo de obstrucción, además de tener el hardware preciso para realizar cada tarea.

-El último punto que se le destaca a la FPGA al ser reprogramables, pueden mantenerse al tanto con modificaciones a futuro que pudieran ser necesarias. Mientras el producto o sistema se va desarrollando, se le puede implementar mejoras funcionales sin la necesidad de invertir tiempo rediseñando el hardware o modificando el diseño de la placa.

Aplicaciones de la FPGA: Hay múltiples áreas en las que actualmente se usan las FPGA desde dispositivos electrónicos hasta aplicaciones en ámbito aeroespacial.

Actualmente su aplicación más destacada es como acelerador de IA o acelerador de hardware, aunque también tiene múltiples otras aplicaciones que ahora nombraremos.

La FPGA tiene aplicación en:

-Aeroespacial y defensa

-Audio

-Automotor

-Bioinformática

-Emisión

-Electrónica de consumo

-Centro de datos

-Industrial

-Médico

-Seguridad

-Comunicaciones inalámbricas

-Computación de alto rendimiento

-Diseño de circuito integrado

-Instrumentos científicos

-Procesamiento de video e imagen

-Comunicaciones por cable

Es llamativo que las FPGA después del declive que tuvieron en el año 2000 actualmente tenga presencia en todas las áreas mencionadas. Flexibilidad, velocidad, integración y reducción en costos totales son el lema para el uso de las FPGA.

Figura 2-10. Beneficios de la FPGA.
Fuente: Altera. Beneficios de la FPGA. [en línea]. <https://www.intel.com/content/www/us/en/products/programmable/fpga.html>. [Consulta: 14 de Noviembre de 2018].

CAPÍTULO 3: COMPARATIVAS

3. FABRICANTES DE FPGA Y COMPARATIVAS

3.1. FABRICANTES DE FPGA

El número de empresas que se dedica al negocio de las FPGA son limitadas, son pocas las que quedan desde su creación hasta hoy, muchas han caído por no poder adaptarse al mercado y sólo las más consistentes conseguirán ver el crecimiento final de esta tecnología.

Los dos gigantes del sector Xilinx y Altera, también tenemos la compañía de la cual se liberó los primeros programas de software gratuito para FPGA Lattice Seminconductors y otras como Microsemi, Quicklogic, Cypress PSoC, entre otros.

3.2. VISTA AL GIGANTE DE LAS FPGA

En esta sección se verá el resultado de la evolución de las FPGA en la compañía formada por los creadores, observar características técnicas de los modelos más importantes y comparar precios entre ellos.

Pueden surgir varias preguntas ¿Por qué Xilinx? ¿No sería mejor ver un poco de todas las empresas para saber cómo están en general las FPGA?

Bueno la verdad es que Xilinx es la compañía más grande del negocio incluso aún se mantiene por sobre Altera después de que Intel lo haya adquirido. Xilinx es la mejor representación de las FPGA y cómo se han desarrollado desde sus inicios; tienen una variedad de productos muy interesantes que vale la pena ver ya que abarca todo tipo de clientes y esto no es una exageración, la gran variedad de desarrollo que presentan es algo que realmente llama la atención y al revisarlo se puede tener una visión clara de la diferencia en las capacidades de las familias de gama baja con las de media y alta. Y también obviamente la diferencia de precios que conlleva la diferencia en las capacidades de cada chip. Después de todo, como estudiantes nos interesa principalmente la tecnología más barata que está más orientada a ese tipo de cliente, pero siempre es importante saber cómo va el desarrollo actual de la mejor tecnología.

Xilinx es la compañía fundada por Ross Freeman, Bernie Vonderschmitt y Jim Barnett creadores de la FPGA en 1984 mismo año que fundarían la compañía. Tienen sede en San José, California.

Trabajan con cuatro líneas familiares que se separan para suplir una porción del negocio siendo la familia Virtex de gama alta, los dispositivos de esta serie están destinados a ocupar las labores que requieran un mayor rendimiento, esta familia es la que va orientada a aplicaciones de nivel más avanzado.

La familia Kintex de gama media con un rendimiento menor al de la Virtex, pero con precios menores y consumo menor.

Y luego se encuentran las familias de gama baja Spartan y Artix. La familia Artix está diseñada para atender las necesidades de factor pequeño y de rendimiento de bajo consumo de energía de los equipos de baterías portátiles de ultrasonido, control de lente de la cámara digital comercial, aviónica militar y equipo de comunicaciones. La serie Spartan está dirigida a aplicaciones con una huella de baja potencia, sensibilidad extrema y de alto volumen; por ejemplo pantallas, decodificadores, routers inalámbricos y otras aplicaciones.

3.2.1. Comparativa de capacidades

Es un poco complicado hacer este tipo de sección ya que como veremos en la imagen cada familia posee diferentes categorías. Esta sección va a ir orientada a comparar las categorías de una "generación", en este caso se usará como referencia las series 7 de cada familia.

Figura 3-1. Productos Xilinx más recientes.
Fuente: Xilinx. Dispositivos FPGA y 3D ICs. [en línea]. <https://www.xilinx.com/products/silicon-devices/fpga.html> [Consulta: 15 de Noviembre de 2018].

Ahora compararemos las características técnicas de las 4 familias, para poder clarificar y tener una mirada más subjetiva sobre las diferencias entre las familias. Se verá que cada versión trae varios chips de diferentes capacidades.

Spartan-7 FPGAs

		I/O Optimization at the Lowest Cost and Highest Performance-per-Watt (1.0V, 0.95V)					
	Part Number	XC7S6	XC7S15	XC7S25	XC7S50	XC7S75	XC7S100
Logic Resources	Logic Cells	6,000	12,800	23,360	52,160	76,800	102,400
	Slices	938	2,000	3,650	8,150	12,000	16,000
	CLB Flip-Flops	7,500	16,000	29,200	65,200	96,000	128,000
Memory Resources	Max. Distributed RAM (Kb)	70	150	313	600	832	1,100
	Block RAM/FIFO w/ ECC (36 Kb each)	5	10	45	75	90	120
	Total Block RAM (Kb)	180	360	1,620	2,700	3,240	4,320
Clock Resources	Clock Mgmt Tiles (1 MMCM + 1 PLL)	2	2	3	5	8	8
I/O Resources	Max. Single-Ended I/O Pins	100	100	150	250	400	400
	Max. Differential I/O Pairs	48	48	72	120	192	192
Embedded Hard IP Resources	DSP Slices	10	20	80	120	140	160
	Analog Mixed Signal (AMS) / XADC	0	0	1	1	1	1
	Configuration AES / HMAC Blocks	0	0	1	1	1	1
Speed Grades	Commercial Temp (C)	-1,-2	-1,-2	-1,-2	-1,-2	-1,-2	-1,-2
	Industrial Temp (I)	-1,-2,-1L	-1,-2,-1L	-1,-2,-1L	-1,-2,-1L	-1,-2,-1L	-1,-2,-1L
	Expanded Temp (Q)	-1	-1	-1	-1	-1	-1

Tabla 3-1. Spartan-7 características técnicas.
Fuente: Xilinx. Programmable 7 series product selection guide. [PDF]. Estados Unidos. [Consulta: 15 de Noviembre de 2018].

Artix-7 FPGAs

		Transceiver Optimization at the Lowest Cost and Highest DSP Bandwidth (1.0V, 0.95V, 0.9V)							
	Part Number	XC7A12T	XC7A15T	XC7A25T	XC7A35T	XC7A50T	XC7A75T	XC7A100T	XC7A200T
Logic Resources	Logic Cells	12,800	16,640	23,360	33,280	52,160	75,520	101,440	215,360
	Slices	2,000	2,600	3,650	5,200	8,150	11,800	15,850	33,650
	CLB Flip-Flops	16,000	20,800	29,200	41,600	65,200	94,400	126,800	269,200
Memory Resources	Maximum Distributed RAM (Kb)	171	200	313	400	600	892	1,188	2,888
	Block RAM/FIFO w/ ECC (36 Kb each)	20	25	45	50	75	105	135	365
	Total Block RAM (Kb)	720	900	1,620	1,800	2,700	3,780	4,860	13,140
Clock Resources	CMTs (1 MMCM + 1 PLL)	3	5	3	5	5	6	6	10
I/O Resources	Maximum Single-Ended I/O	150	250	150	250	250	300	300	500
	Maximum Differential I/O Pairs	72	120	72	120	120	144	144	240
Embedded Hard IP Resources	DSP Slices	40	45	80	90	120	180	240	740
	PCIe® Gen2[1]	1	1	1	1	1	1	1	1
	Analog Mixed Signal (AMS) / XADC	1	1	1	1	1	1	1	1
	Configuration AES / HMAC Blocks	1	1	1	1	1	1	1	1
	GTP Transceivers (6.6 Gb/s Max Rate)[2]	2	4	4	4	4	8	8	16
Speed Grades	Commercial Temp (C)	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2
	Extended Temp (E)	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3
	Industrial Temp (I)	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L

Tabla 3-2. Artix-7 características técnicas.
Fuente: Xilinx. Programmable 7 series product selection guide. [PDF]. Estados Unidos. [Consulta: 15 de Noviembre de 2018].

Cada familia de FPGAs de Xilinx tiene diversos chips, ya que van orientado a satisfacer diferentes clientes, las familias de gama baja Spartan y Artix no difieren mucho en sus características técnicas ni al sector de público que va dirigido, ya que después de todo son limitadas como para realizar trabajos más complejos.

Kintex-7 FPGAs

		Optimized for Best Price-Performance (1.0V, 0.95V, 0.9V)						
	Part Number	XC7K70T	XC7K160T	XC7K325T	XC7K355T	XC7K410T	XC7K420T	XC7K480T
	EasyPath™ Cost Reduction Solutions[1]	—	—	XCE7K325T	XCE7K355T	XCE7K410T	XCE7K420T	XCE7K480T
Logic Resources	Slices	10,250	25,350	50,950	55,650	63,550	65,150	74,650
	Logic Cells	65,600	162,240	326,080	356,160	406,720	416,960	477,760
	CLB Flip-Flops	82,000	202,800	407,600	445,200	508,400	521,200	597,200
Memory Resources	Maximum Distributed RAM (Kb)	838	2,188	4,000	5,088	5,663	5,938	6,788
	Block RAM/FIFO w/ ECC (36 Kb each)	135	325	445	715	795	835	955
	Total Block RAM (Kb)	4,860	11,700	16,020	25,740	28,620	30,060	34,380
Clock Resources	CMTs (1 MMCM + 1 PLL)	6	8	10	6	10	8	8
I/O Resources	Maximum Single-Ended I/O	300	400	500	300	500	400	400
	Maximum Differential I/O Pairs	144	192	240	144	240	192	192
Integrated IP Resources	DSP48 Slices	240	600	840	1,440	1,540	1,680	1,920
	PCIe® Gen2[2]	1	1	1	1	1	1	1
	Analog Mixed Signal (AMS) / XADC	1	1	1	1	1	1	1
	Configuration AES / HMAC Blocks	1	1	1	1	1	1	1
	GTX Transceivers (12.5 Gb/s Max Rate)	8	8	16	24	16	32	32
Speed Grades	Commercial Temp (C)	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2
	Extended Temp (E)	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3
	Industrial Temp (I)	-1, -2	-1, -2, -2L	-1, -2, -2L	-1, -2, -2L	-1, -2, -2L	-1, -2, -2L	-1, -2, -2L

Tabla 3-3. Kintex-7 características técnicas.
Fuente: Xilinx. Programmable 7 series product selection guide. [PDF]. Estados Unidos. [Consulta: 15 de Noviembre de 2018].

Virtex-7 FPGAs

		Optimized for Highest System Performance and Capacity (1.0V)										
	Part Number	XC7V585T	XC7V2000T	XC7VX330T	XC7VX415T	XC7VX485T	XC7VX550T	XC7VX690T	XC7VX980T	XC7VX1140T	XC7VH580T	XC7VH870T
	EasyPath™ Cost Reduction Solutions[1]	XCE7V585T	—	XCE7VX330T	XCE7VX415T	XCE7VX485T	XCE7VX550T	XCE7VX690T	XCE7VX980T	—	—	—
Logic Resources	Slices	91,050	305,400	51,000	64,400	75,900	86,600	108,300	153,000	178,000	90,700	136,900
	Logic Cells	582,720	1,954,560	326,400	412,160	485,760	554,240	693,120	979,200	1,139,200	580,480	876,160
	CLB Flip-Flops	728,400	2,443,200	408,000	515,200	607,200	692,800	866,400	1,224,000	1,424,000	725,600	1,095,200
Memory Resources	Maximum Distributed RAM (Kb)	6,938	21,550	4,388	6,525	8,175	8,725	10,888	13,838	17,700	8,850	13,275
	Block RAM/FIFO w/ ECC (36 Kb each)	795	1,292	750	880	1,030	1,180	1,470	1,500	1,880	940	1,410
	Total Block RAM (Kb)	28,620	46,512	27,000	31,680	37,080	42,480	52,920	54,000	67,680	33,840	50,760
Clocking	CMTs (1 MMCM + 1 PLL)	18	24	14	12	14	20	20	18	24	12	18
I/O Resources	Maximum Single-Ended I/O	850	1,200	700	600	700	600	1,000	900	1,100	600	300
	Maximum Differential I/O Pairs	408	576	336	288	336	288	480	432	528	288	144
Integrated IP Resources	DSP Slices	1,260	2,160	1,120	2,160	2,800	2,880	3,600	3,600	3,360	1,680	2,520
	PCIe® Gen2[2]	3	4	—	—	4	—	—	—	—	—	—
	PCIe Gen3	—	—	2	2	—	2	3	3	4	2	3
	Analog Mixed Signal (AMS) / XADC	1	1	1	1	1	1	1	1	1	1	1
	Configuration AES / HMAC Blocks	1	1	1	1	1	1	1	1	1	1	1
	GTX Transceivers (12.5 Gb/s Max Rate)[3]	36	36	—	—	56	—	—	—	—	—	—
	GTH Transceivers (13.1 Gb/s Max Rate)[4]	—	—	28	48	—	80	80	72	96	48	72
	GTZ Transceivers (28.05 Gb/s Max Rate)	—	—	—	—	—	—	—	—	—	8	16
Speed Grades	Commercial Temp (C)	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2
	Extended Temp (E)[5]	-2L, -3	-2L, -2G	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L	-2L, -2G	-2L, -2G	-2L, -2G
	Industrial Temp (I)	-1, -2	-1	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1	-1	—	—

Tabla 3-4. Virtex-7 características técnicas.
Fuente: Xilinx. Programmable 7 series product selection guide. [PDF]. Estados Unidos. [Consulta: 15 de Noviembre de 2018].

La diferencia entre cada familia es notoria. Virtex es la familia insignia de Xilinx, es el que se busca conseguir lo más capacitado posible. En las familias de gama baja y media se regula la arquitectura para conseguir un producto que no supere ciertos estándares, el objetivo es conseguir que cualquier usuario pueda conseguir una FPGA.

Es la estrategia que han adoptado para surgir en este negocio.

3.2.2. Comparativa de precios

Es difícil comparar precios en estos casos, ya que cuando se requiere una FPGA para motivos universitarios o ensayos domésticos lo que importa realmente es comprar una placa con todo el circuito integrado alrededor del chip y eso afecta en gran parte el precio de adquisición, los precios de estas placas no son lineales. Xilinx trabaja en conjunto con una empresa llamada Digilent Inc. La cual construye casi todas las placas que presentan a la venta, son empresas socias. Digilent presenta una diversa cantidad de diseños para las placas, donde sin ir más allá se encuentran las familias Basys, Nexys, Arty entre otras. Por ello la comparativa de precios se establecerá de acuerdo a los chips únicamente proveídos por una de las empresas distribuidoras socias de Xilinx, Digi-Key Electronics.

Incluso el chip se puede dividir por categorías, Xilinx utiliza el siguiente modelo para la numeración de los chips, es igual para cada uno hasta las celdas lógicas después el precio comienza a variar dependiendo del grado de temperatura, grado de velocidad, etc.

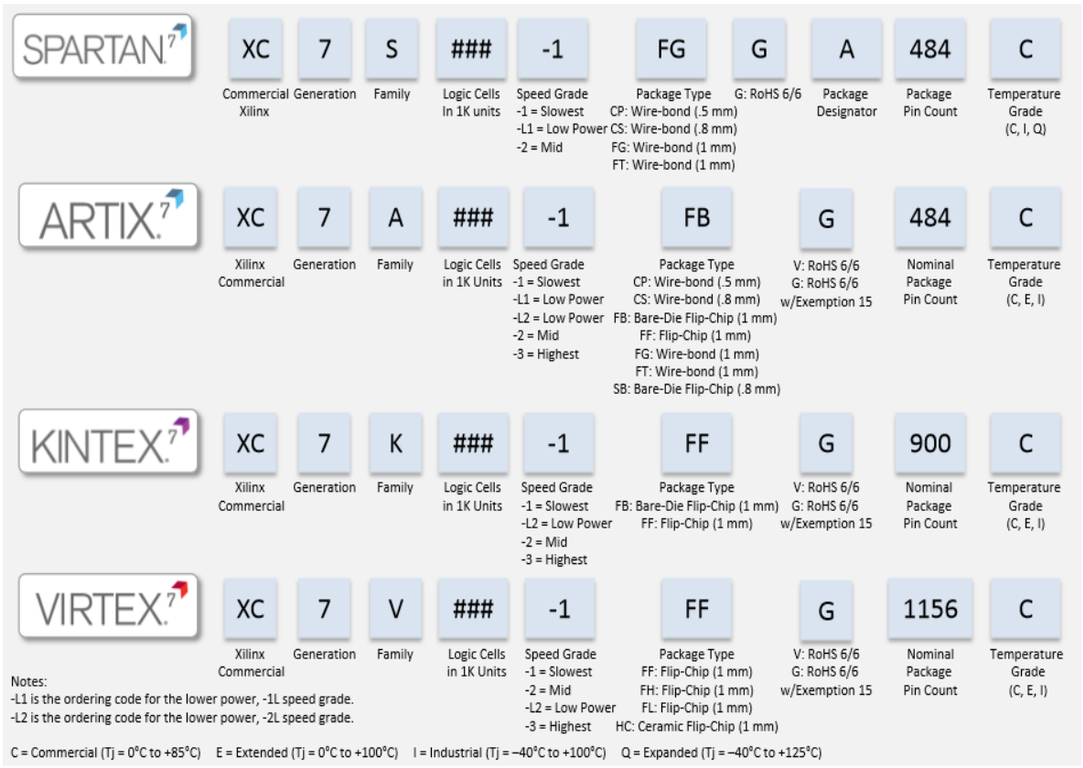


Figura 3-2. Información de pedido del dispositivo.
Fuente: Xilinx. Programmable 7 series product selection guide. [PDF]. Estados Unidos. [Consulta: 15 de Noviembre de 2018].

Los valores que presentan cada familia son:

Spartan 7

Número de Pieza	XC7S6	XC7S15	XC7S50	XC7S75	XC7S100
Precio en Dólares	12,92-18,62	16,48-20,85	46,68-73,29	78,4125-99,2	104,98-132,84

Tabla 3-5. Precios de chips de FPGA Spartan-7 en distribuidora oficial de Xilinx.

Artix 7

Número de Pieza	XC7A12T	XC7A15T	XC7A25T	XC7A35T	XC7A50T	XC7A75T	XC7A100T	XC7A200T
Precio en Dólares	22,5707-48,79	27,93-56,98	32,9-54,39	34,37-80,57	53,2-127,54	92,61-189,8	109,2-239,2	193,7-431,6

Tabla 3-6. Precios de chips de FPGA Artix-7 en distribuidora oficial de Xilinx.

Kintex 7

Número de Pieza	XC7K70T	XC7K160T	XC7K325T	XC7K355T	XC7K410T	XC7K420T	XC7K480T
Precio en Dólares	133,91-247	217,1-448,5	934,7-2418	1593,8-2713,38	1353,3-3313,62	2451,39-4872	3434,16-6510

Tabla 3-7. Precios de chips de FPGA Kintex-7 en distribuidora oficial de Xilinx.

Virtex 7

Número de Pieza	XC7V585T	XC7V2000T	XC7VX330T	XC7VX415T	XC7VX485T	XC7VX550T
Precio en Dólares	3479,67-7814,18	18003,51-35163	2542,41-5709	3560,2-8078	3690-11913,78	4182-10801,86

Tabla 3-8. Precios de chips de FPGA Virtex-7 en distribuidora oficial de Xilinx.

Número de Pieza	XC7VX690T	XC7VX980T	XC7VX1140T	XC7VH580T	XC7VH870T
Precio en Dólares	5587,8-14432,8	14580-26945	16801-32815	6146-12004	23062

Tabla 3-9 . Precios de chips de FPGA en distribuidora oficial de Xilinx.

¿Se nota la diferencia? Para las FPGA de gama baja hay modelos que son sumamente accesibles para cualquier usuario, cada modelo tiene chips para cada uno de los ambientes y con las diferentes construcciones.

Es bastante marcado la diferencia de precios para cada una de las gamas, se observa claramente la orientación para cada familia, lo cual es el objetivo de Xilinx para poder competir en el mercado. Proporcionar para cada tipo de cliente la FPGA con las cualidades necesitadas.

Ahora veremos la diferencia de precio de algunos modelos con placa o sin ella:

La Basys 3 una de los modelos de placa de Digilent tiene montado un chip Artyx-7 XC7A35T-1CPG236C. El valor del conjunto es de $149, mientras que únicamente el chip tiene un valor de $40. En este caso la placa tiene un valor añadido claramente superior al de la propia FPGA.

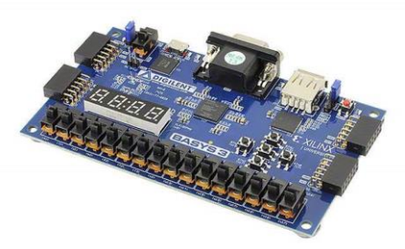


Figura 3-3. Placa Basys 3 de Digilent con FPGA de Artix 7

Un segundo caso es el del modelo Nexys Video de Digilent que utiliza un chip Artix-7 al igual que el anterior de numeración XC7A200T-1SBG484C donde el valor del chip es de $227 y el de la placa en conjunto $490.

Figura 3-4. Placa Nexys Video de Digilent con FPGA de Artix 7.

3.3. LA IMPORTANCIA DE LA TECNOLOGÍA DE DISEÑO

En la construcción de dispositivos electrónicos como FPGAs, procesadores, CPU, etc. La importancia de la eficiencia de energía y tamaño se debe en su mayoría a los transistores. Los transistores son la unidad fundamental en cada uno de ellos siendo el componente que permite emular el comportamiento básico de un bit, como se sabe en un transistor se controla permitiendo o no el paso de energía por la base cuyo control permite tener un estado de 1 o 0. Con los transistores se crean puertas lógicas para realizar operaciones básicas, que son las que gestionan las instrucciones de código máquina.

La técnica actualmente permite crear transistores imposibles de ver para el ojo humano, se necesita utilizar un microscopio para poder observarlos.

¿Qué tipo de ventajas trae reducir el tamaño de los transistores? Son varias, aunque ya se puede deducir de cuales se trata. El tamaño para la fabricación esto permite diseñar equipos electrónicos con mayor eficiencia y un tamaño físico más reducido. El rendimiento y consumo de energía son menores, aunque las estadísticas no mejoran de forma lineal al tamaño de los transistores.

Esto se repite cada vez que evoluciona el tamaño de los transistores utilizados en los chips, se muestra en la siguiente gráfica un ejemplo de la evolución del nanómetro en Intel, una de las empresas tecnológicas más importantes a nivel mundial.

El cambio es evidente y también se aprecian en los equipos tecnológicos de las distintas épocas, se pierde el sentido a la gráfica a partir del 2007 pero en la subsiguiente se distingue

claramente la evolución que se espera para los próximos años proyectos en los que ya se ha comenzado a trabajar.

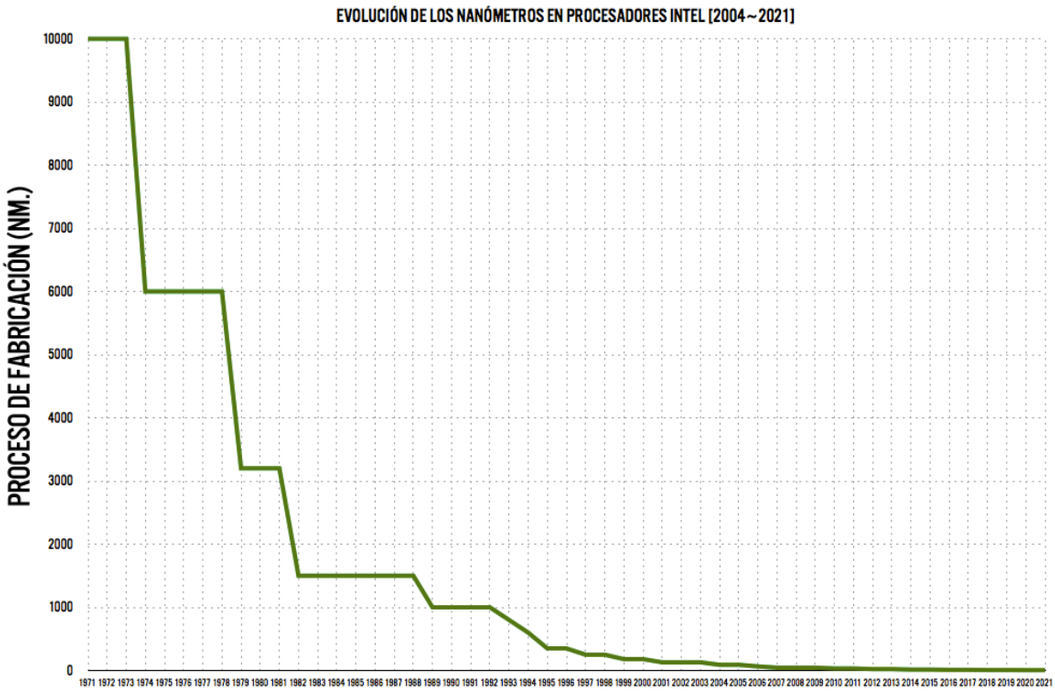


Gráfico 3-1. Evolución de los nanómetros en procesadores Intel 1971-2021.
Fuente: Xataka. La importancia de los nanómetros en los procesadores. [en línea]. <https://www.xataka.com/componentes/la-importancia-de-los-nanometros-en-los-procesadores>. [Consulta: 15 de Noviembre de 2018].

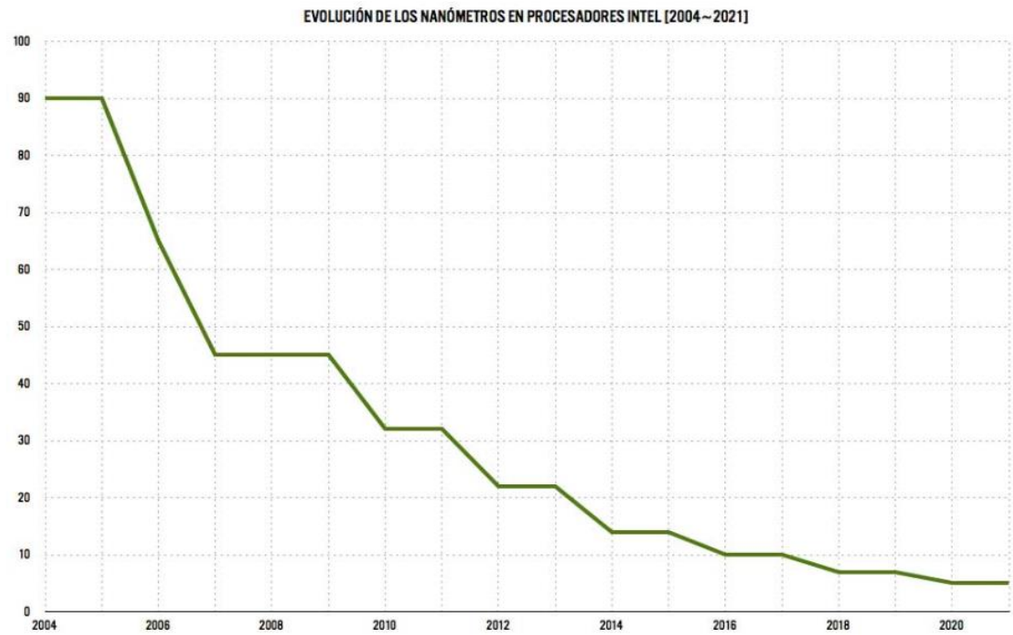


Gráfico 3-2. Evolución de los nanómetros en procesadores Intel 2004-2021.
Fuente: Xataka. La importancia de los nanómetros en los procesadores. [en línea]. <https://www.xataka.com/componentes/la-importancia-de-los-nanometros-en-los-procesadores>. [Consulta: 15 de Noviembre de 2018].

3.4. COMPARANDO LOS FABRICANTES

Vamos a realizar una comparación de los aspectos más importantes a tener en consideración cuando se quiera "afiliar" a alguna de las 3 compañías que van a estar en debate.

¿Por qué solo estas 3? Altera y Xilinx son por gran margen las empresas más importantes del sector, por lo cual lo más común es buscar la adquisición de un producto producido por una marca que de confianza. Y Lattice es la compañía que produce la familia ICE-40, de la cual se consiguió la creación de los primeros softwares libres.

	Altera	Xilinx	Lattice
Disponibilidad de Software	Intel Quartus Design Software desde $2995	Vivado Design Suite desde $2995	Lattice Diamond Design Software $895 - Lattice Radiant Design Suite $0 - iCEcube2 Design Suite $0

Tabla 3-10. Comparativa de software de fabricantes de FPGA.

En este punto se utiliza la comparación de HDL pero cada compañía tiene otro tipo de formas para el diseño de sus circuito, por ejemplo la opción esquemática siempre está. Al principio puede ser lo más fácil, pero se considera una mala costumbre el aprender con ese tipo de diseño. Lo que realmente importa cuando se quiere trabajar con las FPGA son los lenguajes de descripción de hardware que tenga disponible.

Los softwares mencionados son los necesarios para operar una FPGA de base; Xilinx y Altera presentan otras versiones que pueden irse complementando con la de base mientras más avanzado necesiten que sea el software. Para el caso de Lattice el Diamond Design Suite es el principal, pero atiende unas familias en específico y así para los otros softwares.

	Altera	Xilinx	Lattice
Software (con soporte)	Intel Quartus Design Software	Vivado Design Suite	Lattice Radiant Design Suite – iCEcube2 Design Suite – Lattice Diamond Design Software
Sistema Operativo	Microsoft Windows - Linux	Microsoft Windows - Linux	Microsoft Windows -Linux
Lenguaje Utilizado	Verilog-VHDL-AHDL	Verilog-VHDL-System Verilog	Verilog-VHDL

Tabla 3-11. Comparativa de disponibilidad de los softwares de los fabricantes de FPGA.

Como se mencionaba anteriormente en el caso de Xilinx y Altera estos son los softwares de base (obviamente considerando que son de pago por ende son más completos que la versión gratuita), ambos tienen una versión gratuita que sirve para probar la placa y en caso de FPGAs de bajo costo su funcionalidad debería ser suficiente.

Lattice presenta un software gratuito temporalmente que es el Radiant, uno gratuito permanente el iCEcube2 y un software de pago el Diamond. Puede resultar raro que haya un software de pago y como se mencionaba anteriormente sólo atiende unas familias, pues claro obviamente esas son las familias principales, los otros dos atienden básicamente a la familia ICE40.

	Altera	Xilinx	Lattice
Obsolecescencia	Buena	Regular	Excelente

Tabla 3-12. Comparativa de la obsolescencia de los productos de los fabricantes de FPGA.

La obsolescencia es un tema presente siempre en el mundo de la electrónica, lo rápido de la evolución del mercado provoca que haya productos que continuamente vayan quedando descontinuados del mercado.

Esto también afecta a las FPGA, es un problema para el cliente que haya comprado una FPGA hace un tiempo y ahora el software ya no le de soporte o quiera comprar una del mismo modelo de la que tenía, pero el producto ha sido descontinuado.

Comenzando por Lattice, esta empresa presenta una página con todos sus productos descontinuados y la información de donde puede ser adquirido en caso de querer adquirirlo. Y señala que todos los modelos en la lista presentan soporte con la versión gratuita de Lattice Diamond Design Software, es por ello que se evalúa como excelente su trabajo en este ámbito.

Xilinx presenta a los clientes un listado con todos los modelos para los cuales el software actual no presenta soporte y a que versión anterior deben recurrir para usarla. Si bien están todas las versiones anteriores disponibles para su descarga el hecho de que las últimas versiones de Vivado vayan dejando de lado el soporte a las versiones anteriores representa un problema sobre todo para la gente que quiera adquirir la versión de pago. Sin mencionar que incluso para las familias principales (Virtex y Kintex) esto también sucede y tener una tarjeta del precio que valen no resulta conveniente utilizarlas con una versión gratuita.

En cuanto a Altera en su momento fue muy criticada por la descontinuación de algunas familias sin previo aviso, pero actualmente han mejorado en ese sentido, de hecho, el soporte de software que presentan es muy bueno y las versiones actuales presentan soportes para FPGAs de hasta 15 años, algo realmente útil.

	Altera	Xilinx	Lattice Semiconductors
Facilidad de Adquisición	Posible en tiendas limitadas con opciones muy limitadas en el área local - Internacionalmente costeando gastos de envío y con espera considerable	Posible en tiendas limitadas con opciones muy limitadas en el área local - Internacionalmente costeando gastos de envío y con espera considerable	Adquisición local imposible - Internacional costeando gastos de envío y con espera considerable

Tabla 3-13. Comparativa de la facilidad para adquirir un producto de FPGA.

Vamos a ver uno por uno la compañía con sus distribuidores y que tienda nos encontramos en Chile que nos pueda proveer de esos productos. No se considera el tiempo, ya que cuando se realiza la cotización es una información que te entregan una vez cancelado el producto.

El costo de envío más el tiempo de espera que va desde las 2 semanas aproximadamente hacen complicado la adquisición de una tarjeta. En Chile hay algunas empresas de productos electrónicos que tienen FPGAs en stock como MCI Electronics y Rs Components pero son en cantidades mínimas y obligan a elegir entre una cantidad de productos limitados, además solo poseen algunos productos de Xilinx y Altera.

Obviamente las distribuidoras no son las únicas empresas donde podemos encontrar FPGAs, solamente son las oficiales de cada fabricante de estos chips.

Siempre hay empresas alternativas donde se pueden encontrar ofertas, pero hacer referencias a ellas es algo complicado ya que son poco estables y no representan tampoco una confiabilidad absoluta en caso de falla o de cualquier otro problema.

La adquisición de una tarjeta no resulta en un tema sencillo es el problema de vivir en un país subdesarrollado en el cual reina la ignorancia. Antes debió ser realmente difícil si se quería conseguir una tecnología desde un país tan lejano, pero hoy gracias a la globalización se puede conseguir prácticamente cualquier cosa solamente asumiendo los costos de envío y esperando un tiempo “corto” para lo que solía ser.

	Altera	Xilinx	Lattice
Variedad de diseños	Gama alta-gama media-gama baja	Gama alta-gama media-gama baja	Gama baja – Gama ultra baja

Tabla 3-14. Comparativa de la variedad de diseños que ofrece cada fabricante de FPGA.

Las compañías Xilinx y Altera compiten en las mismas áreas, esta competencia hace crecer el negocio de las FPGA, ambas orientan sus productos de la misma forma generando una familia mínima para cada tipo de gama.

Resulta un poco injusto meter a Lattice en este tipo de comparativa ya que su orientación como fabricante de FPGAs no es la misma que los otros dos. Lattice tiene un enfoque de

diseños móviles de bajo costo y bajo consumo. Lattice encontró un hueco en el mercado de las FPGA con destinos de consumo ultra bajo y bajo precio como dispositivos móviles.

¿Cuál es mejor? Pues de momento resulta difícil dar una opinión así ya que se requeriría una investigación muy a fondo e incluso conseguir probarlos, lo que sí es seguro es que toda la competencia en gama alta y media se la llevan Xilinx y Altera, pero de gama baja y ultra baja se recomendaría Lattice sobre todo por la especialización en ese sector.

CAPÍTULO 4: MICROCONTROLADOR VS FPGA

4. COMPARATIVA CON MICROCONTROLADOR

En la carrera de Técnico Universitario en Automatización y Control de la USM Concepción a lo largo de la carrera una de las tecnologías que más énfasis se le presta son los microcontroladores, una tecnología más que conocida y muy popular en el área de la electrónica.

Como se señala al principio de este documento en esta carrera las FPGAs son mencionadas como una tecnología a la que prestarle atención para el futuro, aunque no se enseñan como tal, por lo cual ante el desconocimiento lo mejor para tener una visión clara de sus virtudes es compararla con un similar en cuanto al área de programación; en este caso los microcontroladores.

Normalmente a las FPGA se les compara con los ASIC o los CPLD, pero el tema es que las FPGA se hicieron a partir de ambas tecnologías por lo que podría considerarse una evolución de ellas.

Sabemos que un FPGA es básicamente un circuito integrado que se construye con una gran cantidad de componentes de procesamiento lógico además de multiplexores, flip-flips, tablas de búsqueda y sumadores. Pero estos componentes no son únicamente utilizados para la construcción de FPGAs, ya que son piezas básicas en un IC para que se pueda adquirir una señal de entrada y procesarla en una salida significativa. Lo que hace único a una FPGA es que las interconexiones en su interior son programables, o sea pueden ser reconfiguradas por el usuario cuando desee. Ya vimos anteriormente que esto se realiza mediante un lenguaje de descripción de hardware, esto hace a la FPGA tan especial.

En cambio, un microcontrolador es un circuito integrado que contiene un núcleo de procesador dedicado, memoria y puertos de E/S. Todo esto está formado por los mismos componentes a nivel básico que se encuentran en una FPGA, hablamos de multiplexores, sumadores, flip-flops, etc. La diferencia es que las interconexiones entre todos sus elementos son permanentes, esto quiere decir que no tiene la opción de reconfigurar las interconexiones como hace la FPGA.

Tal como se demuestra en la imagen tienen esas 3 principales características compartidas, pero no son tan iguales como aparenta hasta el momento y lo vamos a ver en la siguiente tabla:

N°	FPGA	Microcontrolador
1	La FPGA puede realizar varias instrucciones a la vez, esto es una ejecución en paralelo de las tareas	Sólo puede realizar una instrucción a la vez, ya que su ejecución es secuencial
2	Son peores en la comunicación en serie	Son mejores comparado con la FPGA en la comunicación en serie
3	Se puede hacer un microcontrolador con una FPGA	No se puede hacer una FPGA con el microcontrolador
4	Se puede considerar como un circuito electrónico que diseñamos configurando interconexiones y bloques lógicos	Ejecuta piezas de código a través de la memoria de instrucciones
5	Consumen mayor cantidad de energía	Tienen un menor consumo de energía
6	Tiene una curva de aprendizaje más pronunciada	La curva de aprendizaje es más sencilla
7	Se utiliza punto fijo, el flotante presenta complicaciones de uso	Están ampliamente disponibles tanto puntos fijos como flotantes
8	No hay método universal para el diseño en FPGA	Es fácil portar diseños entre microcontroladores ya que los lenguajes de programación son universales
9	Es superior en cuanto a flexibilidad para reprogramar el hardware	Tiene software para reprogramación solamente
10	El cambio en el código toma más tiempo hacerlo y un mayor trabajo para el software realizar el cambio ya que debe hacer una reubicación y re encaminamiento del circuito implementado	Se pueden implementar cambios de código después de la compilación
11	Difícil adquisición	En cualquier tienda electrónica se encuentra algún microcontrolador

Tabla 4-1. Comparación entre FPGA y Microcontrolador.

En resumen, si bien es cierto que tienen bastantes puntos en común cada una es mejor en cierto aspecto y realmente vale la pena aprender a usar ambos.

CAPÍTULO 5: HDL

5. LENGUAJE DE DESCRIPCIÓN DE HARDWARE

5.1. INTRODUCCIÓN A HDL

Algo muy importante cuando se trabaja con equipos programables es el lenguaje de programación que emplea, eso aplica para equipos que se basan en un procesamiento secuencial de la programación. Esto no aplica para las FPGAs, ya que son de hardware programable.

Se utiliza un lenguaje de descripción de hardware, estos son utilizados para describir la arquitectura y el comportamiento que tendrá la FPGA internamente. Los HDL utilizan un lenguaje estándar basado en texto que busca proyectar la estructura del circuito electrónico que se desea describir.

Los lenguajes de descripción hardware son lenguajes especializados en la descripción de estructuras, el diseño de las mismas y el comportamiento del hardware. Con estos lenguajes se pueden representar diagramas lógicos, circuitos con diferentes grados de complejidad, desde expresiones booleanas hasta circuitos más complejos. Estos lenguajes sirven para representar sistemas digitales de manera legible para las máquinas y para las personas.

5.2. VENTAJAS DE HDL SOBRE UN LENGUAJE DE PROGRAMACIÓN

Comparado con un lenguaje de programación los HDLs presentan varias ventajas donde podemos encontrar:

-Se puede simular el diseño sin necesidad de implementar el circuito.

-Las herramientas de síntesis permiten convertir una descripción en compuertas lógicas que te ejemplifican lo que entendió el software que se planeaba hacer.

-Las descripciones en un lenguaje de descripción de hardware entregan documentación de la funcionalidad del diseño independientemente de la tecnología utilizada.

5.3. TIPOS DE HDL

Hay varios tipos de HDL, pero podemos encontrar 2 que van a estar presentes en cada FPGA los demás son creados a veces para las de una compañía o alguna familia en específico, Pero Verilog y VHDL se encuentran presentes en todas y son reconocidos en como los lenguajes más importantes y por lo tanto si alguien quiere comenzar a aprender cómo utilizar una FPGA debe hacerlo aprendiendo uno de estos HDLs.

Si se va a elegir un lenguaje del cual aprender hay que tener en cuenta una pequeña descripción de ellos:

VHDL es acrónimo proveniente de la combinación de dos acrónimos: VHSIC (Very High Speed Integrated Circuit) y HDL (Hardware Description Language). El lenguaje VHDL es un estándar de dominio público llamado IEEE 1076-1993. Al ser un estándar no depende de ningún fabricante o dispositivo, es independiente; esto también provoca que se puedan reutilizar los diseños; y por último, al ser un estándar tiene la ventaja de que es un diseño jerárquico, por lo que se mantiene un orden y se siguen ciertas reglas jerárquicas. Su diseño se asemeja a Java.

El lenguaje Verilog al igual que VHDL, es un estándar de dominio público que soporta el diseño, la prueba y la implementación de circuitos analógicos, digitales y de señales mixtas a diferentes niveles de abstracción. Este lenguaje se diseñó basándose en el lenguaje de programación C, con el fin de que resultara familiar para los diseñadores y así fuese rápidamente aceptado. Por ello, Verilog tiene un preprocesador como C y la mayoría de sus palabras reservadas son similares a las de C.

5.4. VERILOG

Para este trabajo se eligió el lenguaje de descripción de hardware Verilog, ya que debido a su semejanza con el lenguaje de programación C resulta mucho más fácil de aprender a usar.

Verilog es uno de los lenguajes de descripción hardware más utilizados. Como se señaló anteriormente con este lenguaje puede utilizarse para describir cualquier hardware a cualquier nivel. Este lenguaje es de dominio público, al igual que VHDL, porque es un estándar IEEE.

Como se señaló anteriormente este lenguaje se diseñó basando en el lenguaje de programación C, pero hay algunas diferencias entre estos lenguajes:

- Verilog carece de estructuras, punteros y funciones recursivas.
- En el lenguaje de programación C no se encuentra el concepto de tiempo, tan importante en los lenguajes HDL.

Un diseño en Verilog consiste en una jerarquía de módulos. Estos módulos son definidos con conjuntos de puertos de entrada, salida y, de entrada-salida. Se utilizan sentencias concurrentes y secuenciales las cuales definen el comportamiento del módulo, definiendo las conexiones entre los puertos, los calves y los registros. Las sentencias secuenciales son puestas dentro de un bloque y ejecutadas en orden secuencial. Por otro lado, todas las sentencias concurrentes y los bloques donde se ponen las sentencias secuenciales se ejecutan en paralelo en el diseño.

5.4.1. Introducción a Verilog

En esta sección se enseñarán los comandos básicos para comenzar a trabajar en Verilog.

Un diseño en Verilog comienza con la sentencia:

```
module <nombre_módulo> <(definición las señales de interfaz)>;
```

Luego se declaran las entradas/salidas:

```
input a,b,c,etc;
output x,y,z,etc;
```

Al final se termina con la sentencia “endmodule”.

A continuación, un pequeño ejemplo para observar la implementación de lo señalado:

```
module ejemplo1(a,b,sel,clk,en,f);

input a,b,sel,clk,en;
output f;


endmodule
```

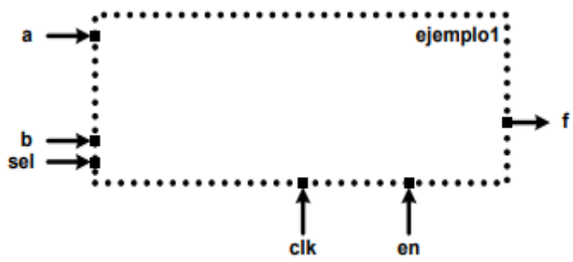


Figura 5-1. Figura sintetizada.

En ese párrafo hemos definido un bloque que cuenta con 5 entradas y 1 salida.

Hay algunos puntos importantes que se deben tener en cuenta para el trabajo en Verilog:

- Los comentarios se pueden ingresar de dos formas; la primera es con // al principio de la sentencia, esto sólo sirve para insertar comentarios de 1 línea; la segunda sirve para generar un espacio de varias líneas donde se considera comentario todo lo ingresado dentro de ese espacio, esto se genera con /* al principio y */ al final.

```
// <comentario>
```

```
/*
<comentario>
*/
```

- Se debe tener en consideración que es sensible al uso de mayúsculas. Se recomienda el uso únicamente de minúsculas.
- Los identificadores en Verilog deben comenzar con un carácter, pueden contener cualquier letra de la “a” a la “z”, caracteres numéricos además de los símbolos “_” y “$”. Con un tamaño máximo de 1024 caracteres.

5.4.2. Números en Verilog

Las constantes numéricas en Verilog pueden especificarse en decimal, hexadecimal, octal o binario. Los números negativos se representan en complemento a 2 y el carácter "_" puede utilizarse para una representación más clara del número, si bien no se interpreta. Si bien el lenguaje permite el uso de números enteros y reales, por brevedad y simplicidad sólo utilizaremos la representación de constantes numéricas enteras.

Entero	Almacenado como	Descripción
1	00000000000000000000000000000001	Unzised 32 bits
8'hAA	10101010	Sized hex
6'b10_0011	100011	Sized binary
'hF	00000000000000000000000000001111	Unzised hex 32 bits
6'hCA	001010	Valor truncado
6'hA	001010	Relleno de 0's a la izquierda
16'bz	zzzzzzzzzzzzzzzz	Relleno de z's a la izquierda
8'bx	xxxxxxxx	Relleno de x's a la izquierda
8'b1	00000001	Relleno de 0's a la izquierda

Tabla 5-1. Números en Verilog.

La tabla muestra algunos ejemplos de definición de números enteros. Verilog expande el valor hasta rellenar el tamaño especificado.

Los números negativos se especifican con el signo "-" delante del tamaño de la constante como se verá a continuación:

```
-8'd2
```

5.4.3. Tipos de datos

Existen dos tipos de datos que se utilizan principalmente:

- reg: Este tipo de dato representa variables con la capacidad de almacenar información en ellas.
- wire: Este tipo de dato representa conexiones estructurales entre componentes, no tiene capacidad de almacenamiento.
- real: Registro con la capacidad de almacenar números en coma flotante.
- time: Registro con capacidad de almacenar hasta 64 bits sin signo.
- integer: También es un registro, pero este presenta la capacidad de almacenar 32 bits.

5.4.4. Operadores

Afortunadamente, los operadores son las mismas cosas aquí que en algún lenguaje de programación. Para hacernos la vida más fácil, casi todos los operadores (al menos los de la lista a continuación) son exactamente iguales a sus equivalentes en el lenguaje de programación C.

Operadores aritméticos:

Símbolo del operador	Operación realizada
*	Multiplicación
/	División
-	Sustracción
%	Resto
+	Suma

Tabla 5-2. Operadores aritméticos.

El único operador que puede resultar desconocido es el del resto; pero este comando es exactamente la obtención del resto de la división de dos números.

Operadores relacionales:

Símbolo del operador	Operación realizada
>	Más grande que
<	Menor que
<=	Menor o igual que
>=	Mayor o igual que

Tabla 5-3. Operadores relacionales.

Operadores lógicos:

Símbolo del operador	Operación realizada
!	Negación lógica
&&	y lógico
||	ó lógico

Tabla 5-4. Operadores lógicos.

El ! en este caso, se coloca delante de un operando, y tiene como función cambiar el valor lógico del mismo.

Cuando se utiliza && el resultado de esta operación es la combinación de los dos operandos. Es decir, para que sea verdadero, ambos operandos tienen que serlo, en otro caso el resultado es falso.

El || al igual que and es el resultado de la combinación de dos operandos. En este caso para que sea verdadero sólo es necesario que uno de ellos cumpla esa condición.

Operadores de igualdad:

Símbolo del operador	Operación realizada
==	Igualdad
!=	Diferente

Tabla 5-5. Operadores de igualdad.

El comando == es una pregunta, se utiliza para comparar si dos comandos son iguales entrega verdadero.

El comando != es lo contrario que el anterior en este caso si son diferentes entrega verdadero.

Operadores a lógica de bit:

Símbolo del operador	Operación realizada
~	not
~&	nand
&	and
|	O
~|	nor
^	xor
^~	xnor
~^	xnor

Tabla 5-6. Operadores a lógica de bit.

Estos permiten efectuar operaciones lógicas con los bits de los operandos.

El comando nand tiene el resultado contrario de and por ejemplo si tiene 1&1=0, si tiene 0&0=1.

El comando xor es el resultado de la combinación de los operandos será verdadero (1) solo si los operandos son diferentes, en otro caso (los operandos son iguales) el resultado es 0.

El comando xnor con el cual obtienes el resultado contrario al que obtendrías con el operador xor.

Otro tipo de operadores:

Tipo de operador	Símbolo del operador	Operación Realizada
Desplazamiento	<<	Desplazamiento a la izquierda
	>>	Desplazamiento a la derecha
Concatenación	{}	Concatenación
Condicional	?	Condicional

Tabla 5-7. Operadores poco comunes.

El comando {} se utiliza para concatenar 2 operandos de cualquier tipo.

```
// A: 4'0110 B: 4'b0010
C = {A, B} D = {2{B}}
//Resultado:
C: 8'0110_0010 D: 0110_0110
```

El comando << desplaza bits a la izquierda. Es necesario indicar el número de desplazamientos y los bits que se quedan vacíos se rellenan con 0.

Por ejemplo:

```
// A: 8'b1111_1001
(A << 2)
// Resultado: A: 8'b1110_0100


// B: 8'b10001110
(B << 3)
// Resultado: B: 8'b01110000
```

El comando >> desplaza a la derecha, es básicamente lo mismo, pero hacia el otro lado.

El comando ? dependiendo del resultado (verdadero o falso) el resultado cambiará.

```
// A: 1'b1 B: 3'b111 C: 3'b001
 A == 1? B : C
 // Resultado: A: 3'b111
 //Si se cumple la igualdad, B cambia su valor C
```

5.4.5. Procesos

El concepto de procesos que se ejecutan en paralelo es una de las características fundamentales del lenguaje, siendo ese uno de los aspectos diferenciales con respecto al lenguaje procedural como el lenguaje C.

Toda descripción de comportamiento en lenguaje Verilog debe declararse dentro de un proceso, aunque existe una excepción que trataremos a lo largo de este apartado. Existen en Verilog dos tipos de procesos, también denominados bloques concurrentes.

- Initial: Este tipo de proceso se ejecuta una sola vez comenzando su ejecución en tiempo cero. Este proceso no es sintetizable. Su uso está íntimamente ligado a la realización del testbench (testbench o banco de pruebas se utilizan para probar el circuito dándole valores y viendo la respuesta).
- Always: Este tipo de proceso se ejecuta continuamente a modo de bucle. Tal y como su nombre indica, se ejecuta siempre. Este proceso es totalmente sintetizable. La ejecución de este proceso está controlada por una temporización (es decir, se ejecuta cada determinado tiempo) o por eventos. En este último caso, si el bloque se ejecuta por más de un evento, al conjunto de eventos se denomina lista sensible.

Ejemplos de su utilización:

```
initial
begin
 clk = 0;
 reset = 0;
 enable = 0;
 data = 0;
end
```

```
always @(a or b or sel)
begin //Sobra en este caso
 if (sel == 1)
 y = a;
 else
 y = b;
 end //Sobra en este caso
```

Si el proceso engloba más de una asignación (=) o más de un comando de control (if-else,case,etc) estas deben estar delimitadas por begin y end.

El comando initial se ejecuta a partir del instante cero y, en el ejemplo, en tiempo 0, si bien las asignaciones contenidas entre begin y end se ejecutan de forma secuencial comenzando por la primera. En caso de existir varios bloques initial todos ellos se ejecutan de forma concurrente a partir del instante inicial.

En el ejemplo, el comando always se ejecuta cada vez que se produzcan los eventos variación de la variable a o variación de b o sel (estos tres eventos conforman su lista de sensibilidad) y en tiempo 0. En el ejemplo, el proceso always sólo contiene una estructura de control por lo que los delimitadores begin y end pueden suprimirse.

Un tipo de error que se puede cometer es el siguiente:

```
wire clk,reset;
reg enable,data;
initial
 begin
 clk = 0; //Error
 reset = 0; //Error
 enable = 0;
 data = 0;
 end
```

El motivo es simple, dentro de un proceso initial o always no se puede realizar asignaciones en variables tipo wire.

5.4.6. Módulos

Cada módulo dispone de entradas y salidas, por las cuales se puede interconectar con otros módulos, aunque existe el caso de los testbench que no cuentan con ningún tipo de entrada o salida. Se puede describir cada módulo de diferente forma estructural o de comportamiento.

Tiene la siguiente estructura:

```
module <nombre> (<señales>);
<declaración de señales>
<funcionalidad del módulo>
endmodule
```

5.4.7. Estructura de control

Las estructuras de control más utilizadas son:

If-else: Esta estructura de control es del tipo condicional y tiene la capacidad de controlar la ejecución de otras sentencias y/o asignaciones. En el caso de haber más de una sentencia o asignación, es necesario usar el bloque begin/end. La forma de este bloque es la siguiente:

```
if (expresión)
 begin
 . . . . // Sentencias 1
 end
else
 begin
 . . . . // Sentencias 2
 end
```

El funcionamiento consiste en que si la expresión es cierta se ejecuta lo contenido en el bloque if, en caso de que sea falta se ejecuta el contenido del bloque else.

Case: También es condicional, esta sentencia evalúa la expresión y dependiendo de su valor ejecutará el caso que corresponda. Si no se cubren todos los posibles casos explícitamente, es necesario crear un caso por defecto (default) que se ejecutará cuando la expresión no coincida con ningún caso. Al igual que en if-else, si existen varias sentencias en un caso se deben agrupar en un bloque begin-end. Su estructura es la siguiente:

```
case (expresión)
caso 1:
 begin
 . . . . // Sentencias 1
 end
caso 2:
 begin
 . . . . // Sentencias 2
 end
. . . .
default:
 begin
 . . . . // Sentencias D
 end
endcase
```

For: El bloque for es igual al utilizado en lenguaje C. Se rige por la misma idea de las anteriores donde si hay varias sentencias se debe incluir el begin/end. Su estructura es la siguiente:

```
for (<valor inicial>, expresión,<incremento>)
 begin
 . . . . // Sentencias
 end
```

While: Este bucle se ejecuta de forma continuada mientras se cumpla la condición que evalúa. Al igual que en los casos anteriores, si existe más de una sentencia es necesario agruparlas en un bloque begin-end. Su estructura es la siguiente:

```
while (<expresión>)
 begin
 . . . . // Sentencias
 end
```

Repeat: Es del tipo bucle y se ejecuta un número determinado de veces, siendo este número la condición de salida del bucle. Si es necesario se usa el begin/end.

Su estructura es la siguiente:

```
repeat (<numero>)
 begin
 . . . . // Sentencias
 end
```

Forever: Este bloque se ejecuta infinitamente, ya que no tiene condición de finalización, al igual que los demás se utiliza el begin/end de ser necesario.

Tiene la siguiente estructura:

```
forever
 begin
 . . . . // Sentencias
 end
```

Wait: Esta sentencia utiliza la palabra reservada wait y de una expresión que iniciará la condición para parar. Esta sentencia puede estar incluida dentro de las sentencias anteriormente mencionadas. Su sintaxis es:

```
wait (expresión)
```

5.4.8. Asignaciones

En verilog existen dos maneras de asignar valores:

- Asignación continua: Este tipo de asignación se utiliza únicamente para modelar lógica combinacional, sin la necesidad de usar una lista de sensibilidad para la asignación. La sintaxis es la siguiente:

assign variable = asignación

La variable solo puede ser declarada de tipo wire. Y la asignación solo puede ser declarada fuera de cualquier proceso o bloque initial o always.

- Asignación procedural: Este tipo de asignación es la más usual hasta el momento. Su sintaxis es la siguiente:

variable = asignación

Estas asignaciones se usan en el interior de los procesos y el tipo de variable a la que se asigna el dato puede ser de cualquier tipo.

5.4.9. Temporizaciones

Las temporizaciones se consiguen mediante el uso de retardos. El retardo se especifica mediante el símbolo # seguido de las unidades de tiempo de retardo. La siguiente imagen muestra el efecto del operador retardo. En ella se observa que los retardos especificados son acumulativos:

```
initial
 begin
 clk = 0;
 reset = 0;
 #5 reset = 1;
 #4 clk = 1;
 reset = 0;
 end
```

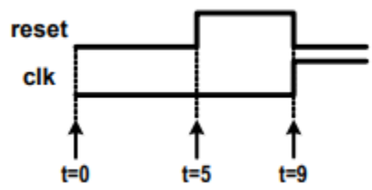


Figura 5-2. Retardos.

En la imagen anterior se ve que las dos primeras asignaciones se ejecutan en tiempo 0. Cinco unidades de tiempo más tarde se realiza la tercera asignación y cuatro unidades más tarde de esta última se ejecutan la cuarta y quinta asignación. La siguiente figura muestra un ejemplo de control de asignación en un proceso always:

```
always
 #5 clk = !clk;
```

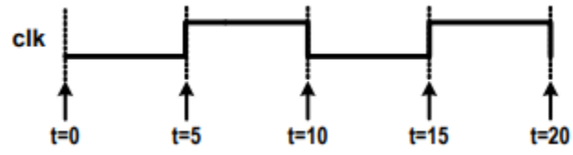


Figura 5-3. Retardo con always.

5.4.10. Eventos

Las asignaciones procedurales pueden ser controladas por el cambio de una variable en un proceso always, esto se denomina control por evento. Para realizar este control se utiliza el carácter @ seguido del evento que permitirá la ejecución de la asignación.

Podemos encontrar 2 tipos de eventos:

-Eventos de nivel: El cambio del valor de una o de un conjunto de variables controla la asignación.

Ejemplo:

```
always @(A or B)
 C = A + B;
```

En este caso si se da un cambio en la variable de A o B se realiza la asignación descrita.

-Evento de flanco: Cuando sucede un flanco de subida (de 0 a 1) o de bajada (de 1 a 0). Para determinar que se están usando la combinación de flancos de subida se usa la palabra

“posedge”, y para el caso de los flancos de bajada se usa la palabra “negedge”. Un ejemplo de este tipo de eventos es el siguiente:

```
always @(posedge clk)
 C <= A + 1;
```

5.4.11. Parámetros

Los parámetros equivalen a las denominadas constantes en otros lenguajes. Su definición sólo se realiza dentro de un módulo.

Se define así: `parameter nombre = valor;`

Se utiliza normalmente para definir el periodo de un reloj o para el tamaño de un bus de datos.

5.4.12. Ejemplos de programación en Verilog

El primer ejemplo es un multiplexor de 2 entradas realizado:

```
module  mux_using_case(
din_0       , // Mux primera entrada
din_1       , // Mux Second entrada
sel           , // Seleccionar entrada
mux_out         // Mux salida
);
// ----------- Puertos de entrada ---------------
input din_0, din_1, sel ;
// ----------- Puertos de salida ---------------
output mux_out;
// ------------ Variables internas --------
reg  mux_out;
// ------------- El código comienza aquí ---------
always @ (sel or din_0 or din_1)
begin : MUX
 case(sel )
    1'b0 : mux_out = din_0;
    1'b1 : mux_out = din_1;
 endcase
end

endmodule
```

Figura 5-4. Ejemplo de multiplexor con always.

Al sintetizar este diseño el software interpreta el multiplexor y nos entrega este esquema RTL:

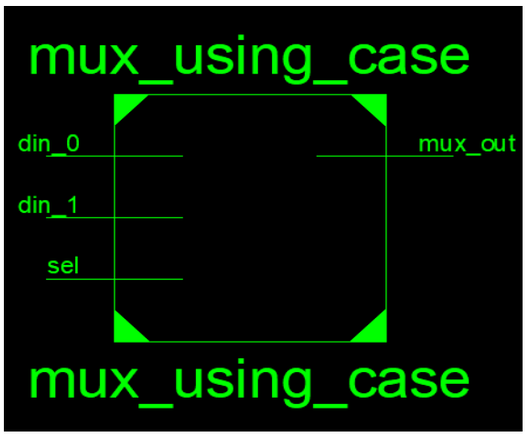


Figura 5-5. Esquema RTL de multiplexor.

Con eso confirmamos la buena descripción del diseño y no habrá problemas al implementarlo. El código consiste simplemente en 2 entradas din_0 y din_1, una salida y una entrada sel que se encargará de seleccionar cual entrada llega a la salida.

El segundo ejemplo será un contador se subida de 8 bits:

```
module up_counter    (
out     ,  // Salida del contador
enable  ,   // habilitar para contador
clk     ,  // reloj Entrada
reset       // restablecer Entrada
);
// ---------- Puertos de salida --------------
    output [7:0] out;
 // ------------ Puertos de entrada --------------
     input enable, clk, reset;
// ------------ Variables internas --------
    reg [7:0] out;
// ------------- Código comienza aquí -------
always @(posedge clk)
if (reset) begin
  out <= 8'b0 ;
end else if (enable) begin
  out <= out + 1;
end


endmodule
```

Figura 5-6 . Ejemplo contador con always.

Al sintetizar el software reconoce correctamente lo que le queríamos pedir: un enable que permite contar, el reloj que realiza el conteo y un reset para volver a iniciar de 0.

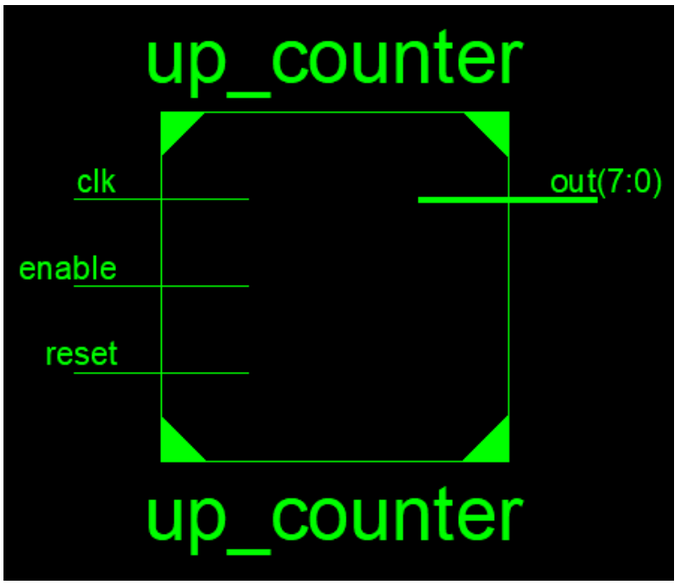


Figura 5-7. Esquema RTL de contador.

CAPÍTULO 6: PLACA FPGA

6. PLACA FPGA

6.1. CARACTERISTICAS GENERALES

La placa elegida a disposición para su implementación es la Spartan-3E Starter Board que cuenta con el chip XC3S500E. Es la solución para problemas en cuales se necesite un alto rendimiento y un gran volumen de datos a un bajo costo puesto que:

- Es una tecnología de avanzada reducido tamaño 90 nanómetros.
- Puede ser conectado en múltiples voltajes y múltiples estándares.
- Cuenta con hasta 376 pines de entrada y salida o 156 entradas diferenciales.
- Señalización de 3.3V, 2.5V, 1.8V, 1.5V y 1.2V.
- Velocidad de transferencia de datos por E/S es de 622+ Mb/s.
- DDR SDRAM soporta hasta 333 Mb/s.
- Recursos lógicos abundantes y flexibles.
- Densidades de hasta 33192 celdas lógicas, incluido el desplazamiento opcional.
- Multiplexores anchos eficientes, lógica amplia.
- Lógica de transporte anticipada rápida.
- Multiplicadores 18 x 18 mejorados con canalización opcional.
- Hasta 648 Kbits de RAM de bloque rápido.
- Hasta 231 Kbits de RAM distribuida eficiente.
- Hasta ocho administradores de reloj digital (DCM).
- Amplio rango de frecuencia (5 MHz a más de 300 MHz).
- Ocho relojes globales más ocho relojes adicionales por cada mitad de dispositivo, más abundante enrutamiento de baja inclinación.
- Interfaz de configuración para PROMs estándar de la industria.

6.2. CARACTERISTICAS DE LA FPGA XC3S500E

El XCS3S500E cuenta con los siguientes atributos principales:

- Compuertas 500k.
- Filas: 46.
- Columnas: 34.
- 1164 CLBs.
- 4656 Slices.
- RAM distribuidas: 73k.
- Bloque de RAM: 360k.
- Multiplicadores dedicados: 20.
- DCMs: 4.
- Cantidad máxima de I/O: 232.
- Cantidad máxima de I/O diferenciales: 92.

El modelo de chip con el que cuenta la placa es el siguiente:

Figura 6-1. Características de chip de la placa.

Al utilizar la placa como un conjunto no es necesario saber mucho sobre el chip, pero al momento de utilizar el software se necesita saber este tipo de información para ingresar al programa.

6.2.1. Tipos de pines

La mayoría de los pines de la tarjeta son de usos generales esto quiere decir que pueden ser entradas o salidas definido por quien programe. Hay, sin embargo, hasta 11 tipos de pines diferentes. Los tipos de funciones están resumidas en el siguiente párrafo y son codificados por color según el pin.

Tipo /codigo de color	Descripcion	Pin Nombre (s) en
I/O	Pines de uso general se pueden usar como I/Os diferenciales según la necesidad del usuario	IO IO_Lxxy_#
IMPUT(ENTRADAS)	Este pin solo puede ser usado comoentrada , no cuenta como salida	IP IP_Lxxy_#
DUAL	Pin de doble propósito usado en algunos modos de configuración. Procesar y luego generalmente disponible como de E / S después de la configuración. . Algunos de Los pines de doble propósito también son entradas de reloj de borde o globales (GCLK).	M[2:0] HSWAP CCLK MOSI/CSI_B D[7:1] D0/DIN CSO_B RDWR_B BUSY/DOUT INIT_B A[23:20] A19/VS2 A18/VS1 A17/VS0 A[16:0] LDC[2:0] HDC
VREF	proporciona una entrada de voltaje de referencia para ciertos estándares de E / S.	IP/VREF_# IP_Lxx_#/VREF_#

Tabla 6-1. Traducción de tabla original de Xilinx.

CLK	Cada paquete Cuenta con 16 entradas de reloj global que, opcionalmente, registran todo el dispositivo. El RHCLK Las entradas opcionalmente registran el lado derecho del dispositivo. Las entradas de LHCLK opcionalmente, registre el lado izquierdo del dispositivo. Se comparte con los pines de configuración de doble propósito y se considera de tipo DUAL.	GCLK[15:0], LHCLK[7:0], RHCLK[7:0]
CONFIG	Pin de configuración dedicado. No disponible como pin de E / S de usuario. Cada paquete tiene Dos pines de configuración dedicados. Estos pines son alimentados por VCCAUX.	DONE, PROG_B
JTAG	Cada paquete tiene cuatro pines JTAG dedicados. Estos pines son alimentados por VCCAUX.	TDI, TMS, TCK, TDO
GND	pin de tierra	GND
VCCAUX	fuente de poder auxiliar	VCCAUX
VCCINT	Fuente de alimentación lógica de núcleo interno dedicado. El número de pines VCCINT Depende del paquete utilizado. Todos deben estar conectados a + 1.2v	VCCINT
VCCO	Junto con todos los otros pines VCCO en el mismo banco, este pin suministra energía a los buffers de salida dentro del banco de E / S y establece el voltaje de umbral de entrada para Algunos estándares de E / S	VCCO
NC	Este pin del paquete no está conectado en esta combinación específica de dispositivo / paquete pero se puede conectar en dispositivos más grandes en el mismo	NC

Tabla 6-2. Traducción de tabla original de Xilinx.

Sumintros de entrada	Descripcion	voltaje
VCCINT	Tensión interna de alimentación del núcleo. Suministra todas las funciones lógicas internas, como CLBs, bloque RAM, y multiplicadores. Entrada al circuito de reinicio de encendido	1.2 V
VCCAUX	Tensión de alimentación auxiliar. Suministro de gestores de reloj digital (DCM), diferencial controladores, pines de configuración dedicados, interfaz JTAG. Entrada para reinicio de encendido Circuito (POR)	2.5V
VCCO_0	suministra los buffers de salida en el Banco de E / S 0, el banco a lo largo del borde superior del FPGA	Seleccionable, 3.3V, 3.0V, 2.5V, 1.8, 1.5V o 1.2V
VCCO_1	Suministra los buffers de salida en I / O Bank 1	Seleccionable, 3.3V, 3.0V, 2.5V, 1.8, 1.5V o 1.2V
VCCO_2	Suministra los búferes de salida en I / O Bank 2, el banco a lo largo del borde inferior del FPGA. Se conecta al mismo voltaje que la fuente de configuración de FPGA. Entrada al circuito de reinicio de encendido (POR).	Seleccionable, 3.3V, 3.0V, 2.5V, 1.8, 1.5V o 1.2V
VCCO_3	Suministra los búferes de salida en I / O Bank 3, el banco a lo largo del borde izquierdo del FPGA.	Seleccionable, 3.3V, 3.0V, 2.5V, 1.8, 1.5V o 1.2V

Tabla 6-3. Traducción de tabla original de Xilinx.

6.3. COMPONENTES DE LA PLACA

En la placa nos podemos encontrar con todos estos elementos:

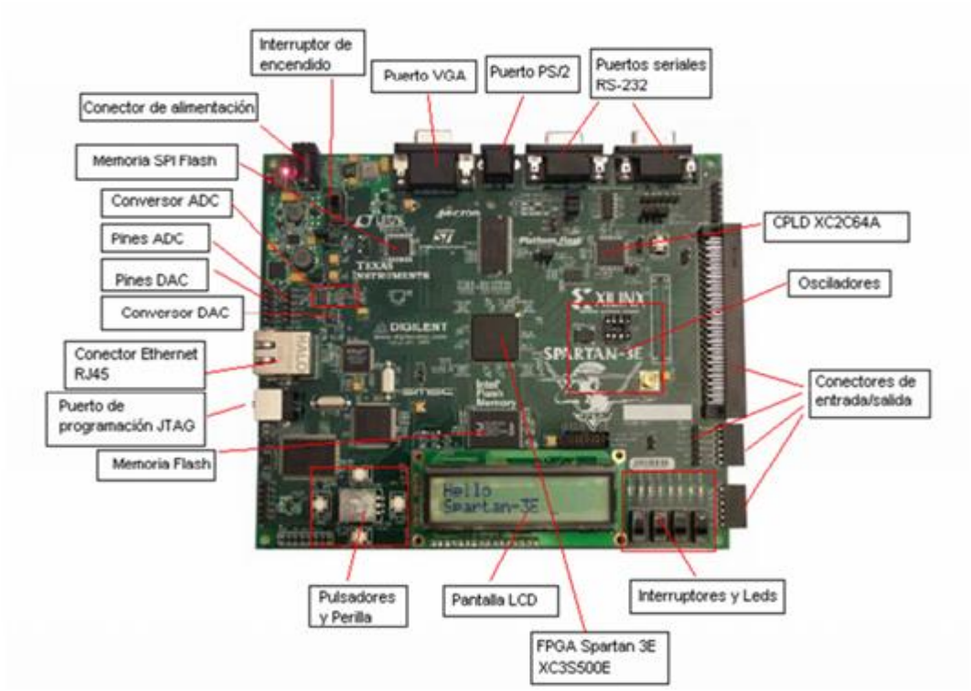


Figura 6-2. Componentes de la placa.

Todo el circuito integrado alrededor de la FPGA tiene como objetivo permitirnos un uso más fácil del chip en sí, tal como se vio en capítulos anteriores la compañía fabricante produce el chip y eso es lo que ofrece generalmente al público, aunque hay pocas que fabrican sus placas para el uso de ellas a veces resulta más fácil encargar a otra empresa que diseñe estas placas realizando así un trabajo conjunto factible para ambos. En el caso de Xilinx su compañero para la fabricación de placas es Digilent.

La placa nos permite distintos tipos de puertos, pines, interruptores, etc. Para la práctica con FPGA cada uno tiene su pin definido y señalado al costado.

Características:

- Xilinx Spartan-3E (500K gates) XC3S500E FPGA
- Xilinx XCF04 Platform Flash para almacenar configuraciones de FPGA
- Texas Instruments TPS75003 Administración de energía de suministro triple IC
- JTAG & SPI Flash programación con cable USB JTAG
- Fuentes de alimentación Linear Technology
- 64MB Micron® DDR SDRAM
- 16MB Numonyx StrataFlash™
- 2MB ST Microelectronics Serial Flash
- Programación JTAG a través del puerto USB2 incorporado

- SMSC LAN83C185 Ethernet PHY
- PS/2 keyboard
- RJ-45 Ethernet
- Cabezal de 16 pines para módulos LCD opcionales
- Dos conectores DB9 RS-232
- Conector Hirose FX2 de 100 pines
- Tres puertos Pmod de 6 pines
- DB15HD VGA

6.3.1. XCF04 Platform Flash

La familia PROM de Platform Flash proporciona almacenamiento no volátil, así como un mecanismo de entrega de flujo de bits integrado para su uso con FPGA de destino.

6.3.2. Administrador de energía TPS75003

El TPS75003 es una solución completa de administración de energía para FPGA. Las habilitaciones independientes para cada salida permiten que la secuencia minimice la demanda de la fuente de alimentación en el inicio. Arranque suave en cada límite de suministro de corriente de entrada durante el arranque. Dos controladores integrados permiten una conversión de voltaje eficiente y rentable para suministros de baja y alta corriente, como el núcleo y la E / S. El TPS75003 está completamente especificado de -40°C a +85°C, lo que produce un tamaño de solución total muy compacto con capacidad de disipación de alta potencia.

6.3.3. SPI Flash

Es un estándar de comunicaciones (Serial Peripheral Interface), utilizado principalmente para la transferencia de información entre circuitos integrados de equipos electrónicos. El bus SPI es un estándar para controlar prácticamente cualquier dispositivo electrónico que acepte un flujo de bits serie, este debe ser regulado por un reloj por lo cual es comunicación síncrona.

6.3.4. DDR SDRAM

Es un tipo de memoria utilizada en computadoras, emplean DDR esto es transferencia de datos tanto en el flanco de subida como el de bajada de la señal de un reloj.

6.3.5. Fuentes de reloj

La placa Spartan-3E Starter Kit admite tres relojes primarios fuentes de entrada, todas ellas ubicadas debajo del logotipo de Xilinx, cerca del logotipo de Spartan-3E.

- La placa incluye un oscilador de reloj de 50 MHz incorporado.
- Los relojes pueden suministrarse fuera de la placa a través de un conector de estilo SMA. Alternativamente, el FPGA puede generar señales de reloj u otras señales de alta velocidad en el conector de estilo SMA.

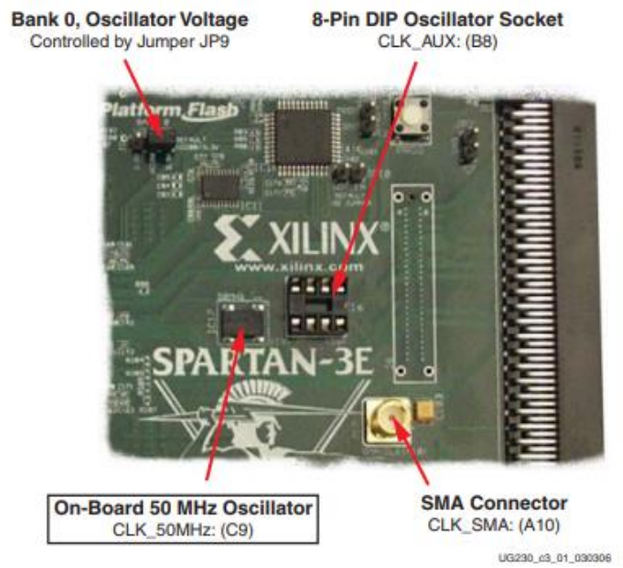


Figura 6-3. Entradas de reloj disponibles.
Fuente: Xilinx. Spartan-3E Starter Kit Board User Guide. [PDF]. Estados Unidos. [Consulta: 8 de Noviembre de 2018].

6.3.6. Programación de la FPGA

Un FPGA se programa por medio de la carga de los datos de configuración en celdas de memoria estática, las que colectivamente controlan todos los elementos funcionales y los recursos de interconexión. Luego de aplicar la alimentación se escribe la trama de configuración en dicha memoria utilizando uno de los siguientes modos: Maestro - Paralelo, Esclavo - Paralelo, Maestro - Serial, Esclavo - Serial o Boundary-Scan (JTAG). Estos modos difieren en el origen del reloj (proviene de la FPGA en los modos Maestro y es externo en los modos Esclavo) y en la forma en que se escriben los datos, por lo que los modos paralelos son más rápidos. Este modo está siempre disponible en la FPGA y al activarlo se desactivan los otros modos indicados. El proceso de configuración de la FPGA ocurre en tres etapas. Primero la memoria interna de configuración es borrada, luego los datos de configuración son cargados en dicha memoria y finalmente la lógica es activada por un proceso de partida.

CAPÍTULO 7: DESARROLLO DEL PROYECTO

7. DESARROLLO DE PROYECTO

7.1. INTRODUCCIÓN

Este capítulo está planteado con el objetivo de visualizar, comprender y demostrar la implementación de una FPGA. En este caso se utilizará la placa Spartan-3E Starter Kit para adquirir señales digitales e implementar una señalización en el LCD de la placa. Simulando el efecto de un estacionamiento se realizará un mini montaje para visualizar el efecto.

Con este proyecto podremos contemplar la forma en la que trabaja la FPGA con señales de tipo digital, la comunicación basada en reloj y una forma de diseño de hardware con Verilog relativamente compleja.

Como el objetivo de este trabajo es introducir al lector al mundo de las FPGA esto se hará paso a paso tratando de dejar claro el porqué de cada acción tomada y enseñar brevemente a utilizar el ISE Design Suite 14.5.

Al principio del trabajo se enfrentaron varios problemas que vienen de la mano con lo que sucede hoy en día con las FPGA, los problemas principales podemos resumirlos en que cuando empiezas a aprender de FPGA se debe hacer absolutamente por cuenta propia hay muy pocas personas al menos en este país que tengan conocimiento del uso de esta tecnología, por ende cada duda o consulta hay que investigar por cuenta propia a ver si hay suerte y aparece la respuesta a la duda que se tiene, esto retrasa considerablemente la velocidad de aprendizaje, además dentro de este sector es conocido que la curva de aprendizaje para el uso de FPGA tiene una curva más empinada que otras tecnologías.

Un problema que complica bastante es la obsolescencia de los productos de Xilinx, ya que tal como se menciona en capítulos anteriores es bastante malo y al tener a disposición un modelo relativamente antiguo al momento de trabajar con ella el usuario se encuentra con la sorpresa que el software actual no da soporte y hay que retroceder al ISE Design Suite 14.5 para tener compatibilidad con la placa aunque el real problema es que era para Windows 7 peor gracias a la comunidad internacional se consiguió encontrar un método para utilizarlo en Windows 8, 8.1 y 10.

El avance tecnológico siempre invita a renovar estos dispositivos que en un principio estaban destinados para otros protocolos, tecnologías, etc. La placa Spartan 3E Starter Kit se comunica por puerto usb 2.0 al computador para recibir el bitstream a la FPGA y esta placa no se comunica correctamente con puertos usb 3.0 por lo cual condiciona aún más el equipo con el que quiera trabajar.

7.2.PRENDIENDO UN LED

Para comenzar a utilizar la FPGA lo más típico es comenzar con un encendido/ apagado de un led, a continuación, la imagen de la ubicación y el número de pin:

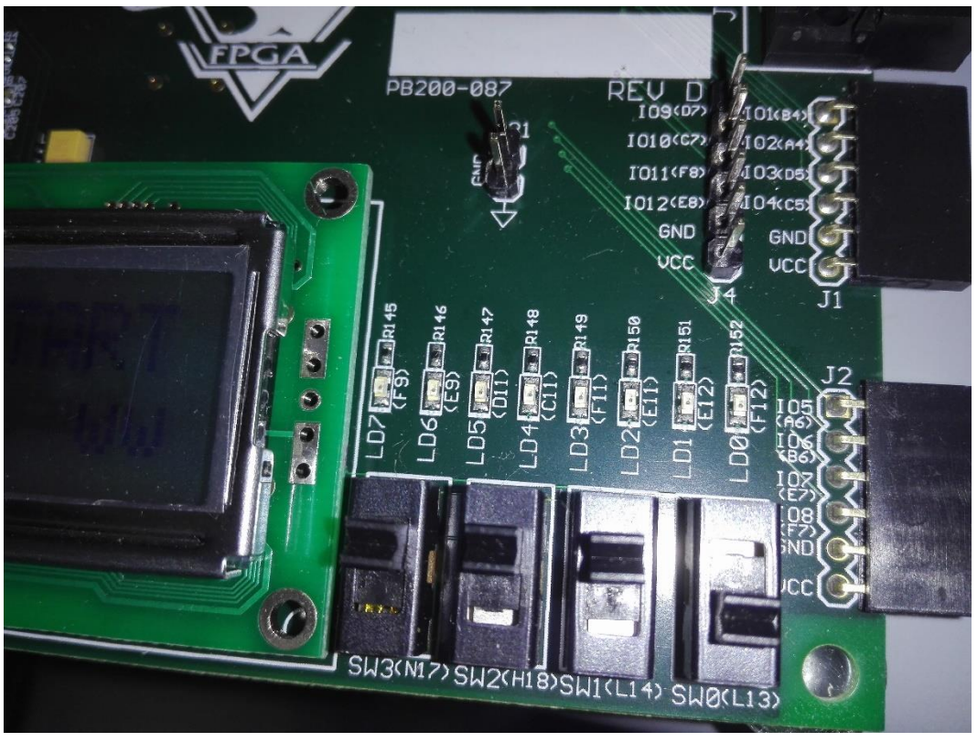


Figura 7-1. Imagen de placa Spartan 3E Starter Kit.

En la imagen se puede observar los interruptores que trae la placa para hacer pruebas y los leds con el mismo objetivo. Debajo se encuentra el pin al cual se le debe asignar si se quiere utilizar ese elemento.

7.2.1. Paso a paso

Lo primero para crear un proyecto es asignar las características de la placa a utilizar, anteriormente se vieron los valores que se les debe asignar.

Para comenzar con esto se debe abrir el ISE Design Suite 14.5 y presionar en New Project, luego de eso aparece unas pestañas donde la importante es como se mencionaba anteriormente la cual le indica al software cual es la placa que estamos utilizando, esto es sumamente importante al momento de utilizar la placa ya que al estar trabajando en descripción de hardware se necesita conocer perfectamente cuál es la placa y los elementos que se puede utilizar. Al momento de sintetizar no es necesario ya que sólo asigna los elementos necesarios para implementar el programa. Pero al momento de generar el archivo

de programación es vital conocer donde implementar el programa, sino el programa no funcionará.

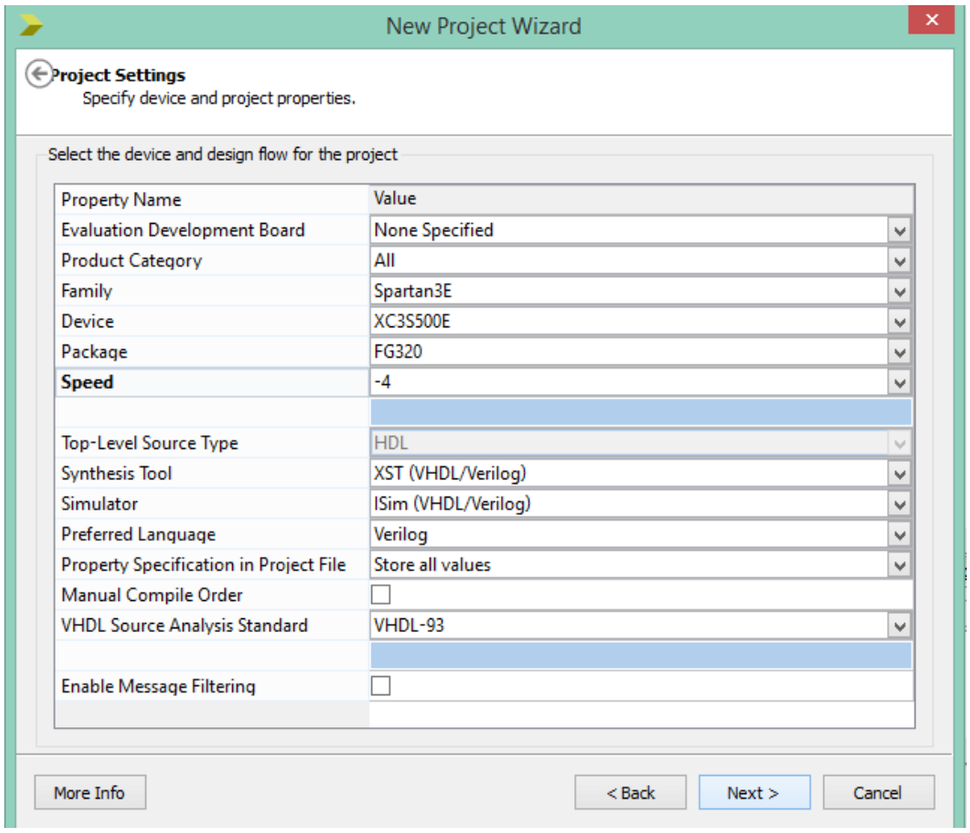


Figura 7-2. Imagen datos de la FPGA a introducir.

Luego de completar la tabla con los datos se puede proceder al crear una fuente en New Source y aparecerán varias opciones donde las opciones para programar son Schematic, VHDL y Verilog, Schematic es más sencillo y puede utilizar módulos hechos con anterioridad como compuertas and, nor, xor, etc. Aunque se considera una mala costumbre ya que sus limitaciones son muchas y no es más que una alternativa para circuitos más sencillos. Aunque si quiere combinar módulos de Verilog se puede hacer. Lo que se hará en este caso es seleccionar Verilog module.

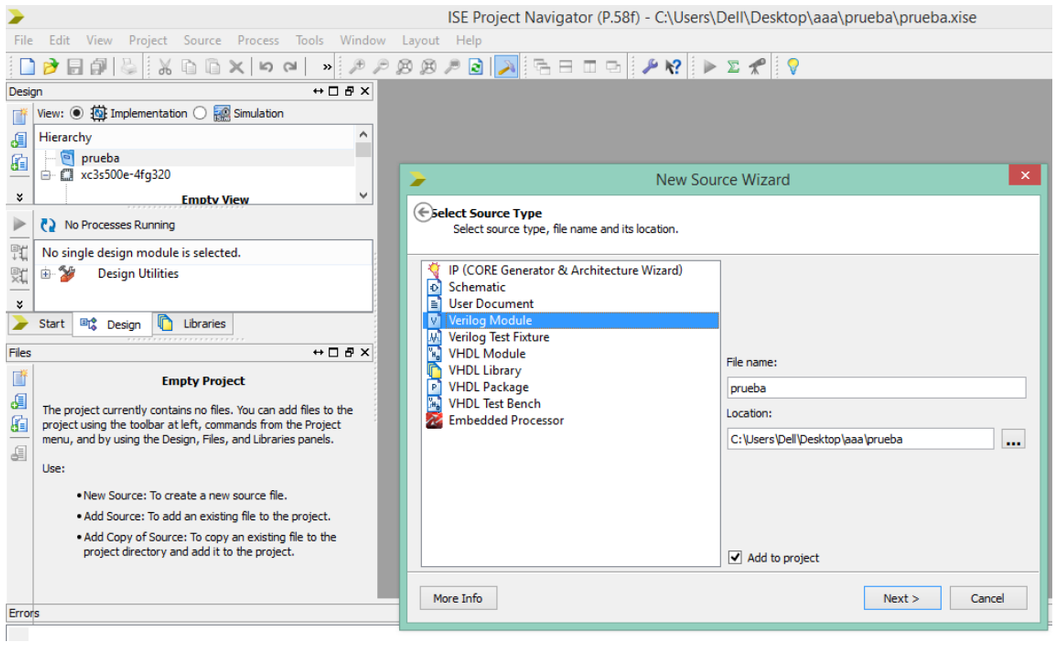


Figura 7-3 . Creando un módulo Verilog.

Para hacer la prueba de encendido de un led se hará un programa con una compuerta and donde la salida será el led.

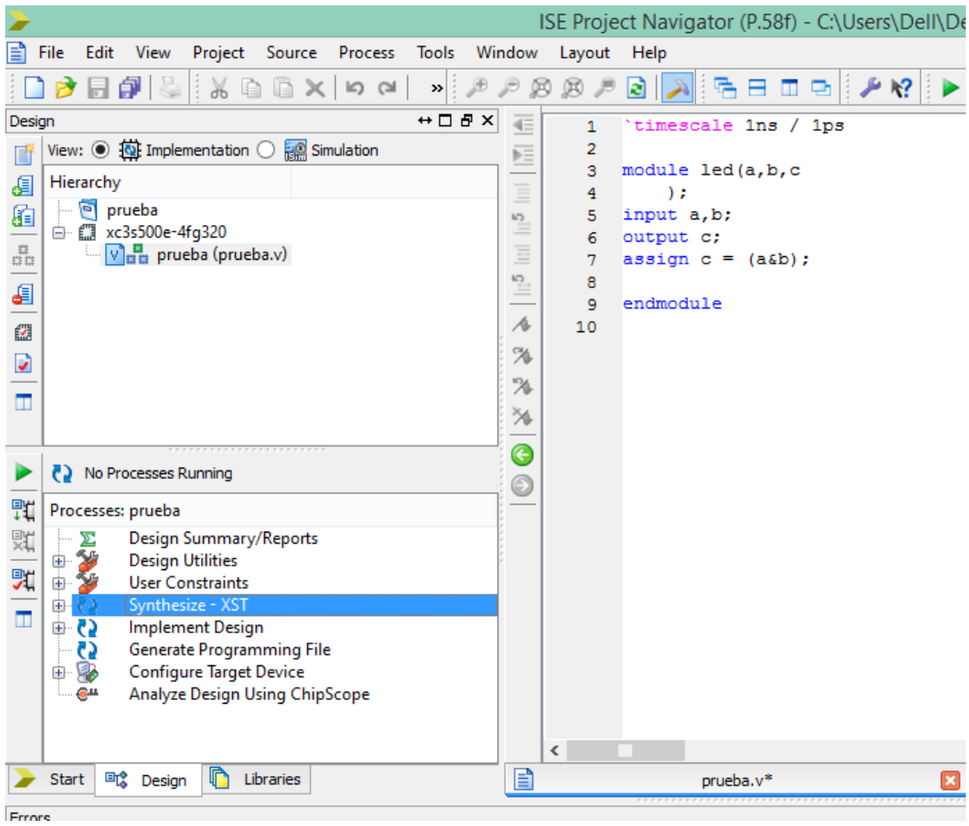


Figura 7-4. Diseño compuerta and.

Una vez desarrollado el programa necesitamos verificar si está correctamente diseñado, para ello hay que sintetizarlo.

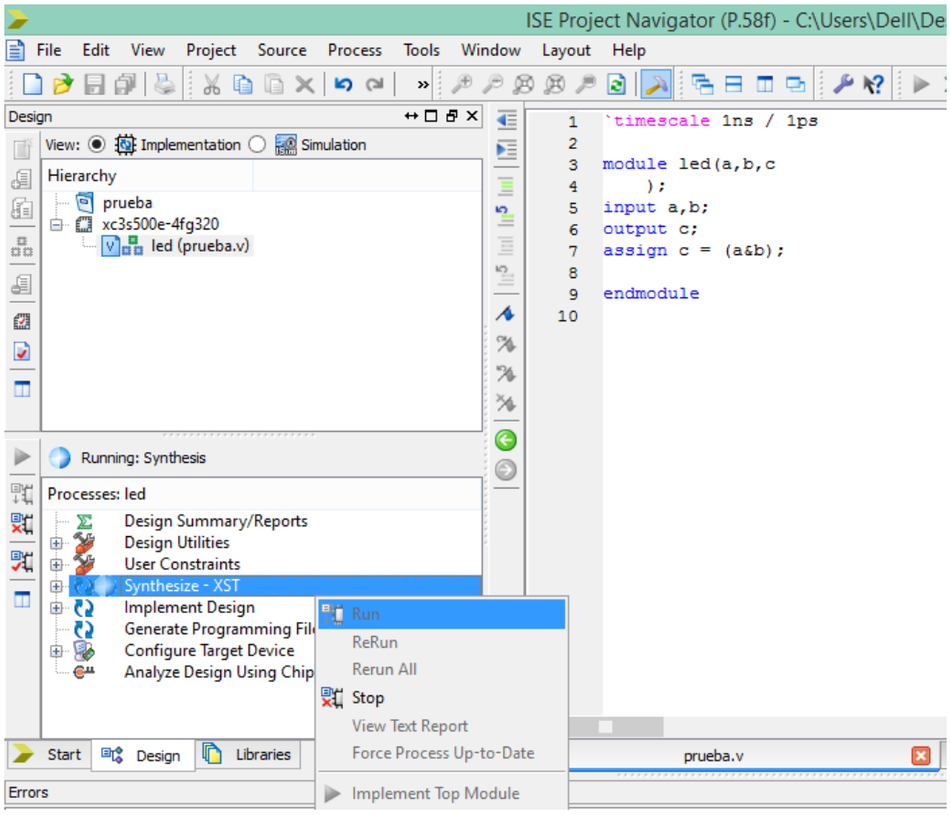


Figura 7-5. Sintetizando el diseño.

Si no salta ningún problema es porque no tiene fallas y ha podido interpretar sin falla alguna el diseño que se ha realizado. También se puede observar el archivo sintetizado y en este caso el software interpreto lo siguiente:

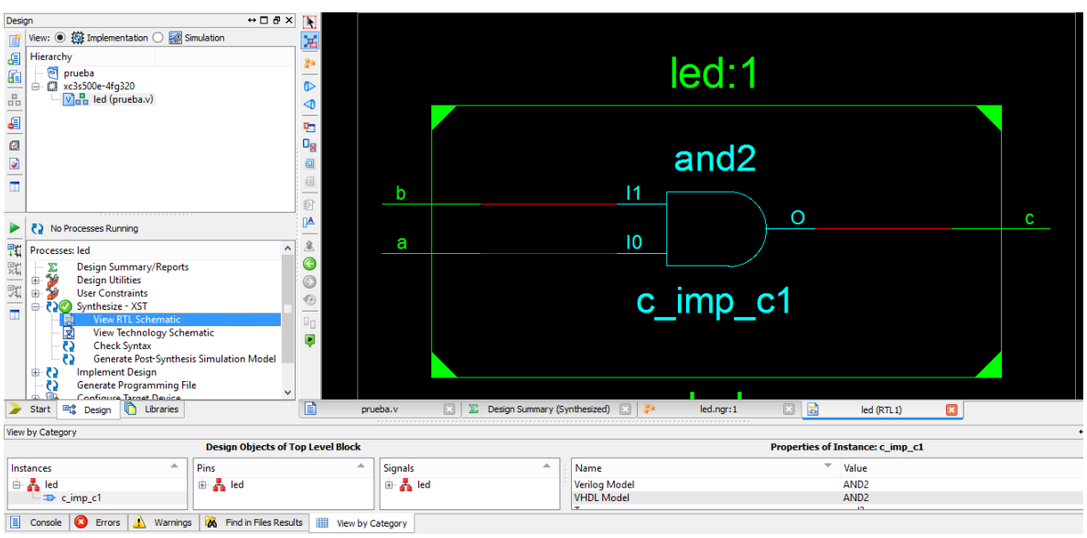


Figura 7-6. Compuerta and sintetizada.

El software ha interpretado correctamente lo que le estábamos indicando e interpretó la compuerta and que se requería.

Luego se debe ejecutar el PlanAhead cuyo programa sirve para la asignación de pines.

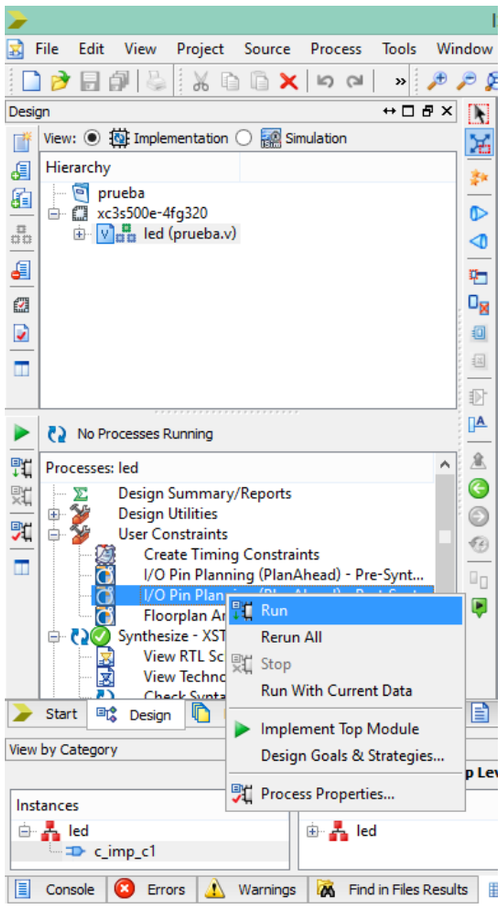


Figura 7-7. Ejecutando PlanAhead.

Luego hay que asignar pines, como se mencionaba anteriormente y L13, L14 son dos switchs (SW0 y SW1), F12 será el pin del led elegido (LD0).

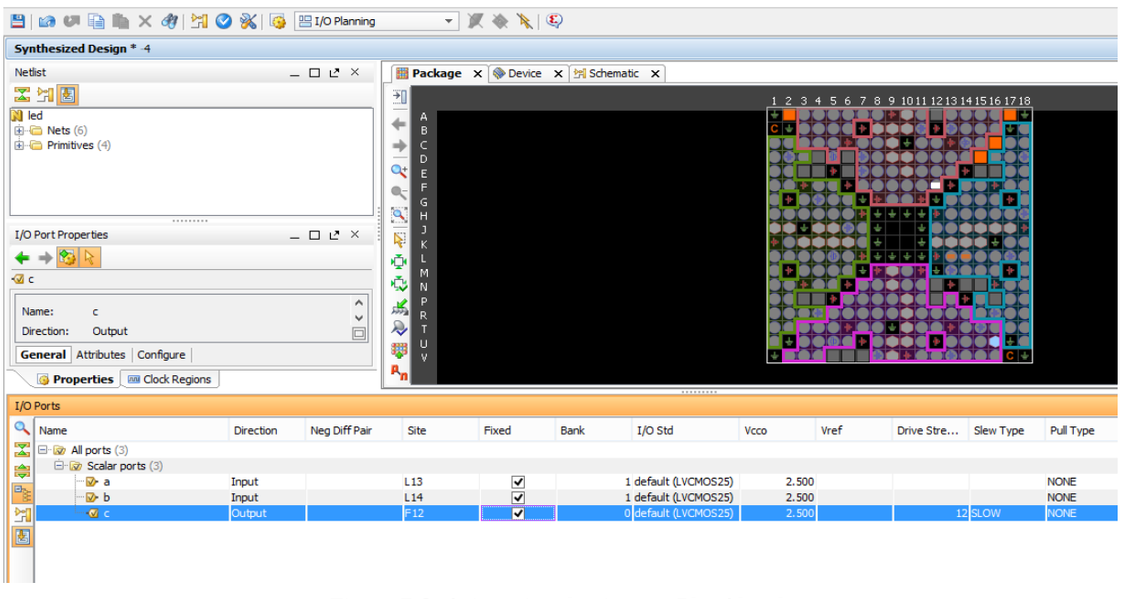


Figura 7-8. Asignacion de pines en PlanAhead.

Una vez asignados los pines se puede generar el archivo de programa, donde el software realiza enrutamiento.

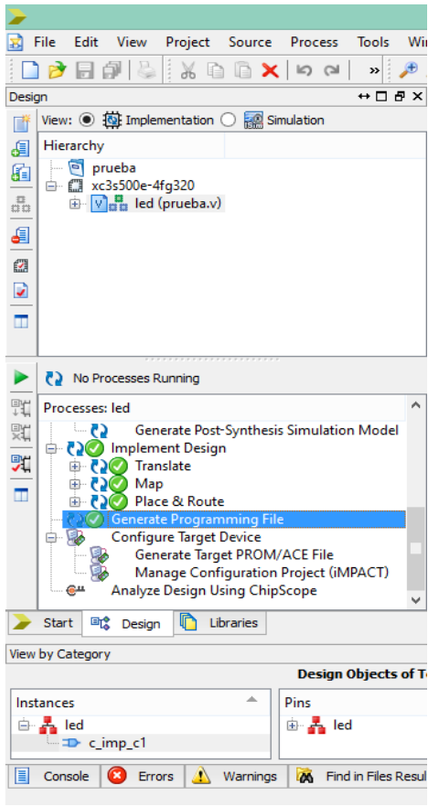


Figura 7-9. Generando Archivo de programa.

Luego hay que abrir el programa iMPACT, el cual sirve para cargar el archivo al FPGA.

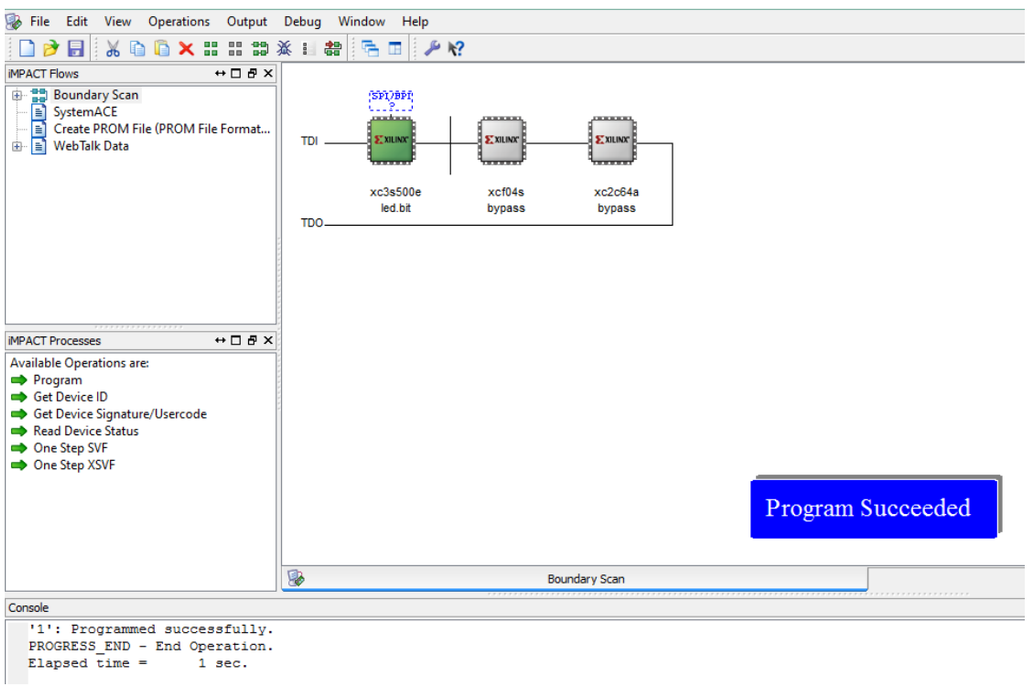


Figura 7-10. Utilizando iMPACT.

Y ahora a comprobar en la placa, según lo definido en la lógica si SW0 y SW1 están en high el LD0 prenderá.

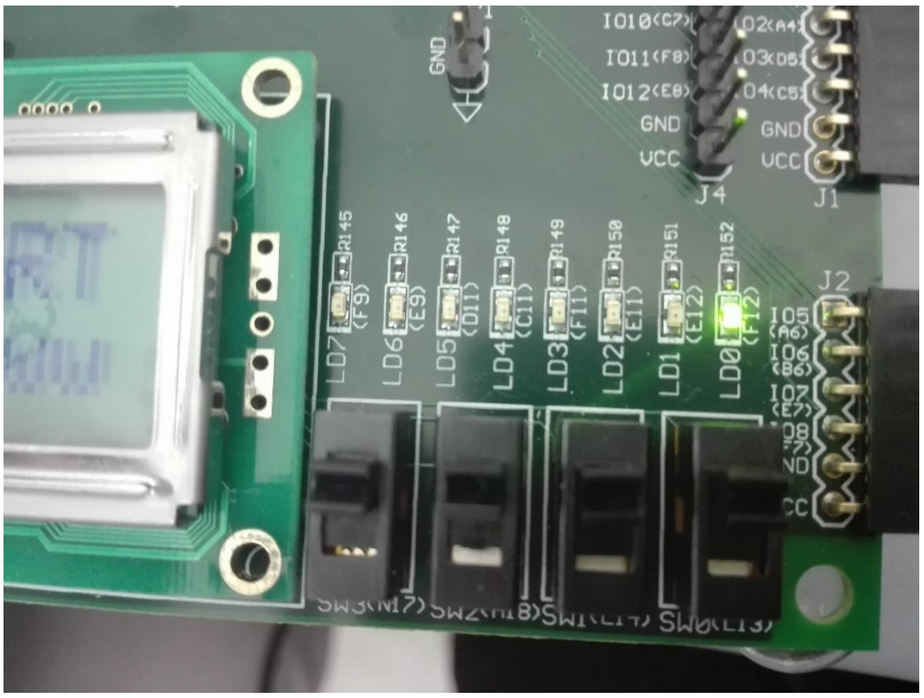


Figura 7-11. Comprobando el and.

7.3. UTILIZANDO EL LCD

La placa cuenta con un LCD que se conecta a través de una interfaz de 4 bits, esto no beneficia al momento de utilizar la placa, ya que se debe enviar los datos para un carácter en dos partes, la inferfaz es de 4 bits ya que se comparte con el dispositivo de almacenamiento StrataFlash y así reducir la cantidad de pines. Para trabajar con el LCD debemos tener en cuenta todas las señales a utilizar.

En la próxima imagen se puede apreciar internamente la conexión de la Spartan-3E con el LCD, en este ejercicio se utilizará obviamente 8 señales para el envío de datos hacia el LCD, donde 4 serán para enviar los caracteres, y las demás para el control.

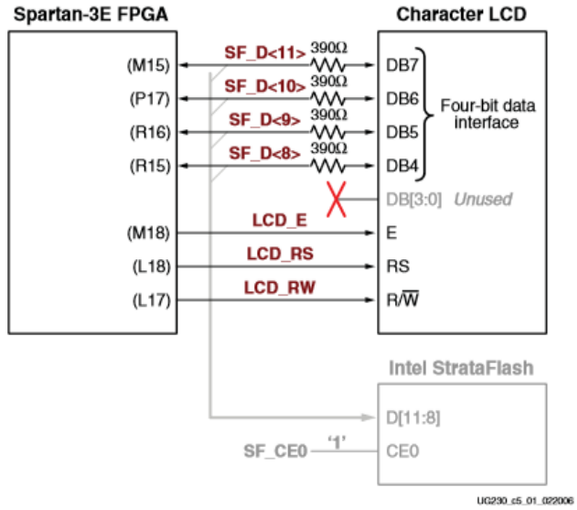


Figura 7-12. Interfaz caracteres de LCD.
Fuente: Xilinx. Spartan-3E FPGA Starter Kit Board User Guide. [PDF]. Estados Unidos. [Consulta: 24 de Diciembre de 2018].

A continuación, se presentan las señales en conjunto con su pin de asignación:

- LCD_E (M18) = Pulso para habilitar (1) lectura/escritura, 0 para deshabilitar.
- LCD_RS (L18) = Seleccionar registro, 0: Registro de instrucciones durante las operaciones de escritura. Flash ocupado durante las operaciones de lectura 1: Datos para operaciones de lectura o escritura.
- LCD_RW (L17) = Control de lectura/escritura 1= LCD presenta los datos, 0=LCD lee los datos.
- SF_D<8> (R15) = data bit DB4.
- SF_D<9> (R16) = data bit DB5.
- SF_D<10> (R17) = data bit DB6.
- SF_D<11> (M15) = data bit DB7.
- SF_E (D16) = Se coloca en 1 para habilitar completamente el uso del LCD y deshabilitar el Strata Flash.

Vamos a trabajar con tres tipos de memoria los cuales son importantes de tener en consideración especialmente al momento de enviar datos para configurar el LCD.

DD RAM	RAM de visualización de datos, hace referencia a los datos que se mostrarán en pantalla.
CG RAM	RAM del generador de caracteres, almacena patrones definidos por el usuario.
CG ROM	ROM del generador de caracteres, incluye una serie de patrones predefinidos que corresponden a los símbolos ASCII.

Tabla 7-1. Memorias del controlador del LCD.

Se envían caracteres en ASCII por lo cual se utilizan 8 bits por carácter. Como se señalaba anteriormente los datos se envían en paquetes de 4 bits por ello se realiza el envío de datos primero enviando los bits superiores en el orden señalado y luego el paquete inferior.

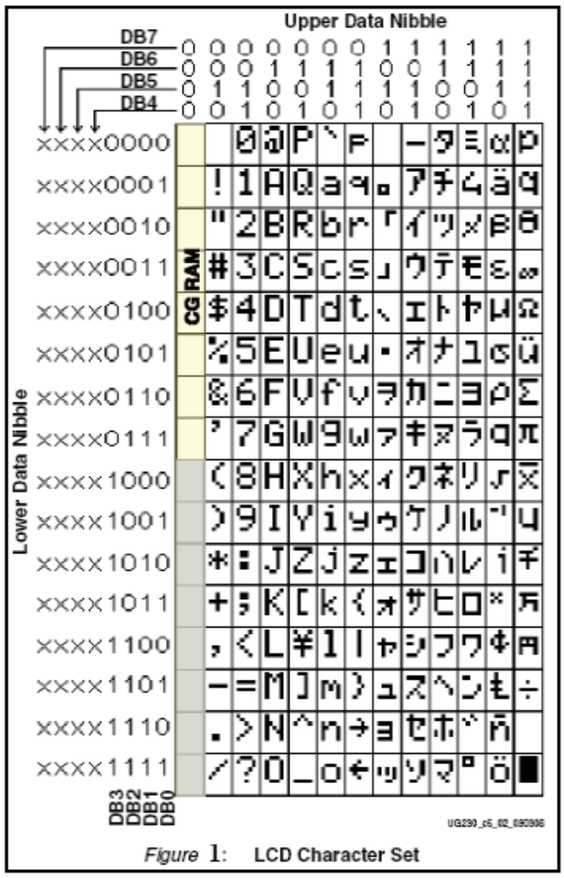


Figura 7-13. Caracteres del LCD.

Luego las siguientes direcciones son ubicaciones en el DD-RAM que corresponden físicamente a las ubicaciones de los caracteres que se muestran en la pantalla LCD.

Hay tres pasos principales para usar la pantalla:

- El primero es la inicialización de la interfaz de cuatro bits.
- El segundo son los comandos para configurar las opciones de pantalla.
- El tercero es la escritura de datos de caracteres.

Se debe realizar cada uno sin excepción o la transmisión podría no resultar como esperamos.

Se comienza inicializando enviándole estamos en un orden predefinido según lo que se señala en el manual. Este proceso es para inicializar el display.

Luego se debe configurar el display para indicarle donde debe partir, el parpadeo del cursor, etc.

Y por último viene el envío de caracteres.

Primero que nada, es necesario saber que el fabricante da las indicaciones de los datos a enviar al momento de la inicialización y configuración, además de una tabla con las funciones disponibles para enviar.

El programa va a contar con:

- Una entrada (clk) la señal de reloj en el cual se va a basar el control del tiempo para el envío de señales.
- 8 salidas definidas como output reg para el envío de los datos hacia el LCD. Las señales tipo reg permite almacenar valores, esto sirve para asignar valores secuenciales a la salida dentro de un always.
- Una variable tipo reg, que será el contador de 27 bits para las señales de reloj, cuya función es servir de referencia al momento de administrar el tiempo.
- Variable tipo reg de 6 bits el cual nos permitirá almacenar los datos a enviar en (rs,rw,d,c,b,a).
- Variable tipo reg para almacenar la información a enviar al LCD para el control de activación. (El fabricante indica que esta señal se debe activar mientras se envían los paquetes de bits).

Funcion	LCD_RS	LCD_RW	Upper Nibble				Lower Nibble			
			DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Limpiar Display	0	0	0	0	0	0	0	0	0	1
Inicio del cursor de retorno	0	0	0	0	0	0	0	0	1	-
Conjunto de modo de entrada	0	0	0	0	0	0	0	1	I/D	S
Display On/Off	0	0	0	0	0	0	1	D	C	B
Cursor y cambio de pantalla	0	0	0	0	0	1	S/C	R/L	-	-
Conjunto de funciones	0	0	0	0	1	0	1	0	-	-
Establecer dirección CG RAM	0	0	0	1	A5	A4	A3	A2	A1	A0
Establecer dirección DD RAM	0	0	1	A6	A5	A4	A3	A2	A1	A0
Leer bandera ocupada y direccion	0	1	BF	A6	A5	A4	A3	A2	A1	A0
Escribir datos en CG RAM o DD RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0

Read Data from CG RAM or DD RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0

Tabla 7-2. Conjunto de comandos de visualización de caracteres LCD.

Ahora a revisar el código, el desarrollo empieza desde la línea 17 designando un bloque always que se activará cada vez que haya un flanco de subida en la señal de reloj (por lo cual se ejecutará cada 20 ns). Posteriormente se debe utilizar el comando begin ya que se va a realizar más de una asignación por ende se vuelve necesario su uso. Antes que nada, hay que definir (en la línea 18) el contador, este contador nos va a dejar presencia de cada flanco de subida de la señal de clk.

Luego se inicializa el case, este case va a tomar referencia de los valores del contador, pero solo cuando se registre cambios desde el bit 22 al 27, Esto permite tener periodos de case más largos lo cual es necesario para la inicialización, posteriormente para el envío de datos esto no tiene mayor relevancia.

A la variable código se le asignan valores de 6 bits, esto es debido a que al final del código se concatena con las variables: rs, rw, d, c, b, a. A cada bit respectivamente por lo cual los dos primeros bits (los más significativos) se asignarán a rs y rw, luego los 4 bits restantes a d,c,b y a.

La inicialización del LCD se realiza en las líneas 20 al 23 donde se envían valores señalados por el fabricante.

Luego de la línea 24 a la 31 se realiza la configuración del LCD donde se le envía un comando de conjunto de funciones, luego un comando de establecimiento de modo de entrada seguidos para un comando para encender y apagar el LCD para finalizar con una limpieza del LCD.

Después de la línea 32 a la 77 se realiza el envío de los caracteres al LCD.

Luego algo importante es definir el valor default del case donde se designará un código para permitir la visualización del código en el LCD.

Al final a sf_e se le asigna un valor de 1 permanente (para permitir el control completo de la interfaz por parte del LCD).

A e se le define un tiempo menor al del envío de los datos (esto es definido por el fabricante).

Y por último se procede a realizar la concatenación de código con las variables de salida.

```
1  `timescale lns / lps
2
3
4  module lcdfinal(clk, sf_e, e,rs, rw, d,c,b,a);
5  input clk;  //  "C9"
6   output reg sf_e;  // "D16"
7  output reg e;  // "M18"
8   output reg rs; // "L18"
9  output reg rw;  // "L17"
10  output reg d;  // "M15"
11  output reg c;  // "P17"
12  output reg b;  //"R16"
13  output reg a;   //"R15"
14 reg [26:0] count=0;
15 reg [5:0] codigo;
16 reg valorparae;
17 always @(posedge clk) begin
18 count <= count + 1;
19 case (count [26:21])
20 0: codigo <= 6'h03;
21 1: codigo <= 6'h03;
22 2: codigo <= 6'h03;
23 3: codigo <= 6'h02;
24 4: codigo <= 6'h02;
25 5: codigo <= 6'h08;
```

Figura 7-14. Diseño en Verilog código LCD parte 1.

```
26 6: codigo <= 6'h00;
27 7: codigo <= 6'h06;
28  8: codigo <= 6'h00;
29  9: codigo <= 6'h0C;
30  10: codigo <= 6'h00;
31  11: codigo <= 6'h01;
32  12: codigo <= 6'b100100;//E
33  13: codigo <= 6'b100101;
34  14: codigo <= 6'b100110;//l
35  15: codigo <= 6'b101100;
36  16: codigo <= 6'b100010;//(espacio)
37  17: codigo <= 6'b100000;
38  18: codigo<= 6'b100110;//m
39  19: codigo <= 6'b101101;
40  20: codigo <= 6'b100111;//u
41  21: codigo <= 6'b100101;
42  22: codigo <= 6'b100110;//n
43  23: codigo <= 6'b101110;
44  24: codigo <= 6'b100110;//d
45  25: codigo <= 6'b100100;
46  26: codigo <= 6'b100110;//o
47  27: codigo <= 6'b101111;
48  28: codigo <= 6'b100010;//(espacio)
49  29: codigo <= 6'b100000;
50  30: codigo <= 6'b100110;//d
51  31: codigo <= 6'b100100;
52  32: codigo <= 6'b100110;//e
53  33: codigo <= 6'b100101;
54  34: codigo <= 6'b001100; //para correrlo
```

Figura 7-15. Diseño en Verilog código LCD parte 2.

```
35: codigo <= 6'b000000;
 36: codigo <= 6'b100110;//l
 37: codigo <= 6'b101100;
 38: codigo <= 6'b100110;//a
 39: codigo <= 6'b100001;
 40: codigo <= 6'b100111;//s
 41: codigo <= 6'b100011;
 42: codigo <= 6'b100010;//(espacio)
 43: codigo <= 6'b100000;
 44: codigo <= 6'b100100;//F
 45: codigo <= 6'b100110;
 46: codigo <= 6'b100101;//P
 47: codigo <= 6'b100000;
 48: codigo <= 6'b100100;//G
 49: codigo <= 6'b100111;
 50: codigo <= 6'b100100;//A
 51: codigo <= 6'b100001;
 52: codigo <= 6'b100010;//(espacio)
 53: codigo <= 6'b100000;
 54: codigo <= 6'b100011;//:
 55: codigo <= 6'b101010;
 56: codigo <= 6'b100010;//)
 57: codigo <= 6'b101001;
  default: codigo <= 6'h10;
  endcase
  valorparae <= count [20];
  sf_e <= 1;
  {e, rs,rw, d,c, b,a}<= {valorparae, codigo};
  end
  endmodule
```

Figura 7-16. Diseño en Verilog código LCD parte 2.

Posteriormente se procede a cargar el código y se tiene el siguiente resultado:

Figura 7-17. Foto resultado de código.

7.4. PROYECTO DE ESTACIONAMIENTO

7.4.1. Pre desarrollo

Uno de los objetivos planteados inicialmente con este trabajo es realizar un montaje con el cual apreciar el funcionamiento de una FPGA.

Esta es la implementación de un estacionamiento a escala de maqueta en la cual se consta de una entrada donde hay un sensor inductivo que detectará el ingreso del vehículo y lo mismo para la salida donde se detectará su partida.

Este proyecto está planteado con el objetivo de visualizar, comprender y demostrar la implementación de una FPGA. En este caso se utilizará la placa Spartan 3E Starter Kit para adquirir señales digitales e implementar una señalización en el LCD de la placa. Simulando el efecto de un estacionamiento se realizará un mini montaje para visualizar el efecto.

Con este proyecto podremos contemplar la forma en la que trabaja la FPGA con señales de tipo digital, la comunicación basada en reloj y una forma de diseño de hardware con Verilog relativamente compleja.

Al plantear este trabajo como una fuente de introducción al mundo de las FPGA el proyecto se plantea de forma diferente, esto consiste en comenzar por mostrar el proceso completo para la implementación de una compuerta and en la placa.

Estas señales serán adquiridas por los pines de entrada digital de la FPGA, en este caso serán B4 y A4.

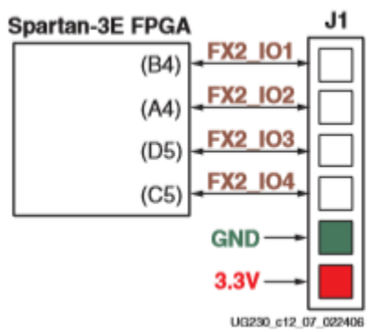


Figura 7-18. Conexiones FPGA al cabezal de pines J1.
Fuente: Xilinx. Spartan-3E FPGA Starter Kit Board User Guide. [PDF]. Estados Unidos. [Consulta: 10 de Enero de 2018].
.

Como se mencionaba anteriormente la implementación tratará de una maqueta donde se consta de 2 sensores colocados en la entrada y salida. Se realizará un conteo de las entradas y salidas que se visualizará en el LCD de la FPGA.

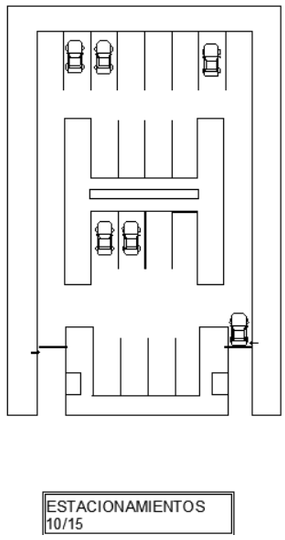


Figura 7-19. Esquema de estacionamiento.

7.4.2. Concepto General

El objetivo es conseguir con un cambio en cualquiera de los sensores genere una variación en el mensaje mostrado en el LCD.

Anteriormente ya se mostró la forma de trabajar con el LCD y como enviarle un mensaje, esta vez se le debe añadir la posibilidad de un cambio en el mensaje, para ello se va a utilizar el diseño esquemático.

El diseño esquemático proporciona una facilidad para conectar diferentes módulos mientras las entradas y salidas tengan el mismo formato de datos, cada módulo puede generar un archivo de este tipo.

El diseño para este estacionamiento se basa en 4 módulos principales donde cada uno cumple una función específica y un quinto módulo que es simplemente un punto común para obtener más de una señal de reloj.

La creación de tantos módulos tiene como objetivo facilitar el flujo de datos y evitar cualquier interferencia entre ellos, con eso se consigue que cada módulo cumpla su labor y pase los datos a la siguiente. El diseño en ISE Design Suite es bastante complejo debido a que el más mínimo error produce una falla completa en el diseño, así como también una estructura de diseño inadecuada donde se mezclen tipos de variables que puedan toparse o que simplemente no sean compatibles.

Como se mencionaba anteriormente serán 2 sensores los encargados de detectar la entrada y salida de los vehículos, estos sensores son de tipo inductivo NPN, necesitan una alimentación de 24VDC que se realizará por medio de una fuente, la señal no se puede enviar directo a la FPGA ya que el nivel de voltaje es demasiado para la placa, por ende, se utilizó un relé de estado sólido para detectar la señal de manera indirecta y así con un contacto normal abierto

medir el estado del sensor. En la FPGA se realizó un circuito de tipo PULL-DOWN para recibir un 0 normalmente y cundo detecte cambie a 1.

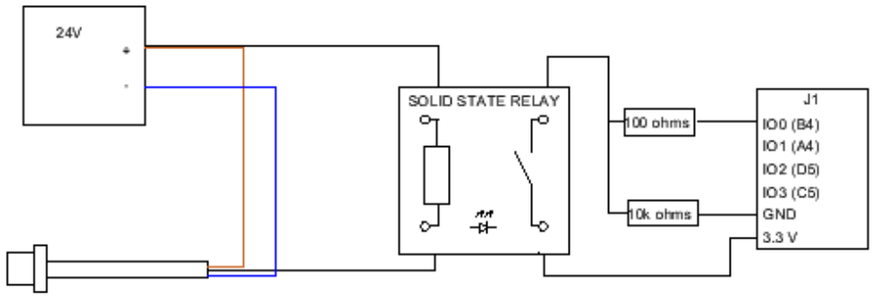


Figura 7-20. Conexionado de sensor de entrada a la FPGA.

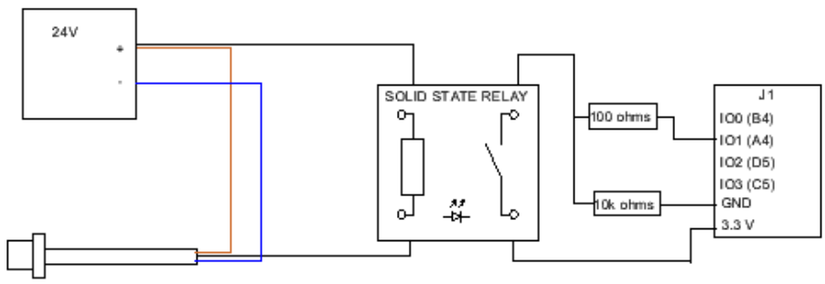


Figura 7-21. Conexionado sensor de salida a la FPGA.

El código arbitrariamente se dividió en 3 secciones:

- El contador se encargará de detectar la señal y contabilizarla para entregarla en un formato de 4 bits.
- La conversión se encarga de tomar la contabilización de cada entrada y obtener el resultado de la resta, para posteriormente asignar el resultado al código ASCII correspondiente.
- El LCD utiliza el código ASCII de 16 bits y se implementa en el mensaje, el código se basa en el mismo formato que se señaló en la sección anterior.

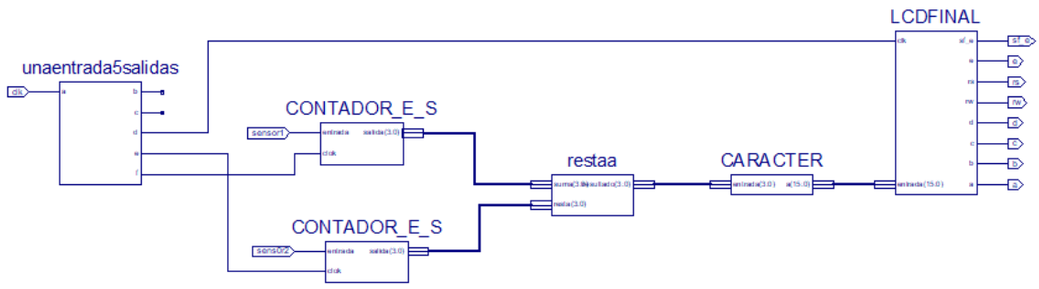


Figura 7-22 Diseño final de proyecto en esquemático

7.4.3. Contador

El contador es un esquemático compuesto por cinco módulos principales que realizan la tarea principal y dos que sirven como punto de derivación, para conseguir un poco de orden con las señales de entrada y evitar el acumulamiento de cables, ya que a veces cuando se cruzan el programa los toma como un nodo.

El contador recibe una señal de reloj, la señal del sensor y entrega una contabilización de 4 bits.

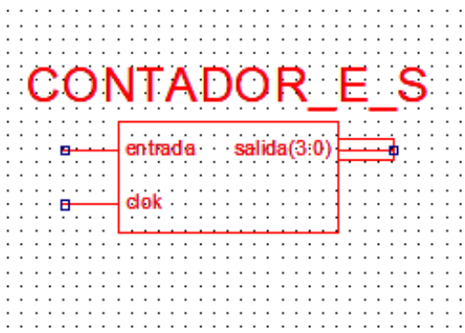


Figura 7-23. Símbolo de contador.

Como se señalaba anteriormente el esquemático está conformado por cinco módulos y dos puntos de derivación donde la única función que cumplen es entregar en cada una de las salidas el mismo valor que se tiene en la entrada.

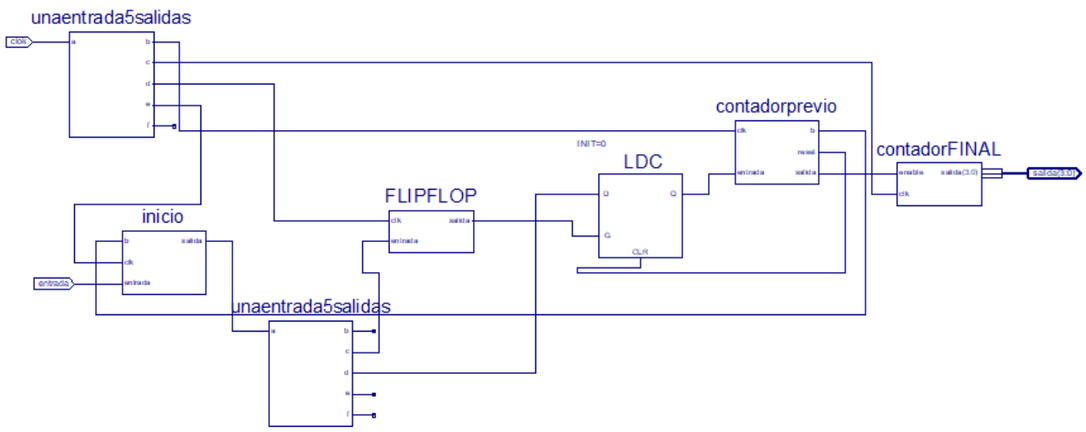


Figura 7-24. Esquemático de contador.

El módulo de inicio cumple la labor de detectar la activación de la entrada de sensor y entregarlo en su salida hasta que le llegue la señal del contador previo que le bloquea hasta que termine el proceso de contabilización (para contabilizar un valor), para evitar que vuelva a contabilizar el mismo en caso de dejar activado el estado HIGH se le asignó otro bloqueo que le impide volver a generar una señal de salida HIGH a menos que el sensor pase por el estado LOW.

```
1  `timescale 1ns / 1ps
2
3  module inicio(b,salida,clk,entrada);
4  input clk,b,entrada;
5  output reg salida;
6  reg bloqueando=0;
7  always @(posedge clk)
8  begin
9  if (b==1)begin
10 salida<=0;
11 bloqueando<=1; end
12 if (entrada==0)begin
13 bloqueando<=0;end
14 if (b==0 & bloqueando==0)begin
15 salida<=entrada; end
16 end
17 endmodule
```

Figura 7-25. Código módulo inicio.

FLIPFLOP es un esquemático compuesto por cinco flip-flops y un not cuyo objetivo es permitir la lectura de la entrada en el latch LDC, normalmente tiene una salida de estado HIGH. hasta que reciba la señal de entrada HIGH que al estar negado tendrá como resultado una señal LOW. Pero los flip-flops provocan un retraso de cinco ciclos de reloj con respecto

a la entrada esto resulta en que la salida del latch almacena el valor HIGH en su salida y cinco ciclos después se desactiva la lectura de valores.

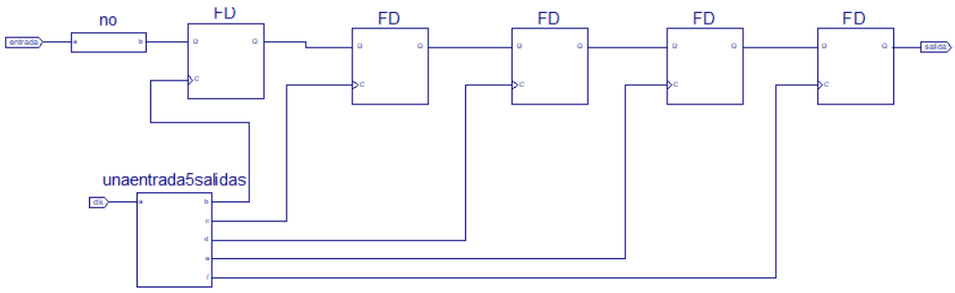


Figura 7-26. Esquemático de flip-flops.

El latch LDC es un símbolo proporcionado por el programa que cuenta con una señal de entrada, una de salida un habilitador y un reset. El propósito de este latch en el contador es almacenar el valor de estado HIGH para el conteo del módulo de contador previo y le emita la señal de reset que terminará por resetear el circuito y volver todas las señales a su punto de inicio.

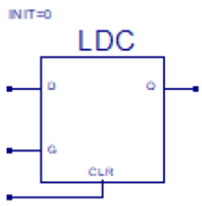


Figura 7-27. Símbolo de latch.

Los módulos de contabilización podrían estar contenidos en uno perfectamente, pero con el objetivo de tener un orden en el diseño se optó por separarlos. Primero el contador previo realiza un conteo mientras la señal de entrada (la que entrega el latch) se encuentra en estado HIGH, también entrega una señal HIGH en la salida y el conteo sea inferior a 40000000 ciclos de reloj lo que son 0,8 s y al momento de cumplirse los 40000000 envía una señal de reset al latch, baja a LOW la señal de salida, resetea su propio contador y desactiva el bloqueo que realiza en el módulo de inicio.

```
1  `timescale 1ns / 1ps
2
3  module contadorprevio(clk,b,reset,entrada,salida );
4  output reg b=0;
5  input clk,entrada;
6  output reg reset;
7  output reg salida=0;
8  reg iniciolel=0;
9  reg [26:0] contador=27'b0;
10 always @ (posedge clk) begin
11 if (entrada==1)
12 iniciolel<=1'b1;
13 reset<=0;
14 if (iniciolel==1 & contador<40000000) begin
15 contador<=contador+1;
16 salida<=1;
17 reset<=0; end
18 else if  (contador==40000000) begin
19 reset<=1;
20 contador<=27'b0;
21 salida<=0;
22 iniciolel<=1'b0;
23 b<=0; end
24 if (contador==10)begin
25 b<=1;end
26 end
27 endmodule
```

Figura 7-28. Código módulo contador previo.

El contador final inicia el conteo al momento que el contador previo por lo cual van sincronizados y realiza un conteo hasta que la señal de entrada sea HIGH y su conteo sea inferior a 40000000 pero al momento que cumple ese valor al contador de 4 bits que tiene definido como salida le suma uno y resetea el contador. En el momento que ocurre ello se resetea todo el contador menos el módulo de inicio que necesita un reset a 0 de la señal del sensor.

```
1  `timescale 1ns / 1ps
2
3  module contadorFINAL( salida , enable  ,  clk  );
4
5  reg[27:0] count=28'b0;
6  input enable, clk;
7  output reg [3:0] salida=4'b0;
8
9  always @(posedge clk)begin
10 if (enable & count<40000000) begin
11   count<= count + 1;
12 end
13 else if  (count==40000000)begin
14 salida<=salida+1;
15 count<=28'b0;end
16 else begin
17 count<= count + 0;
18 salida<=salida+0; end
19 end
20 endmodule
21
22
```

Figura 7-29. Código módulo contador final.

7.4.4. Conversión

Esta sección engloba los módulos resta y carácter, el módulo resta adquiere la señal de 4 bits de ambos sensores y realiza una resta que entrega un resultado el cual se asigna como salida.

```
1  `timescale 1ns / 1ps
2
3  module restaa(suma,resta,resultado
4      );
5  input [3:0] suma;
6  input [3:0] resta;
7  output[3:0] resultado;
8  assign resultado = (suma-resta);
9  endmodule
10
11
```

Figura 7-30. Código módulo contador resta.

Luego el resultado llega al módulo carácter allí dependiendo de cuál valor tenga la señal (0 a 15 en binario) asignará a la salida de 16 bits su valor correspondiente codificado en ASCII. Un carácter en ASCII está compuesto por 8 bits, la idea es enviarle dos caracteres (desde 00 a 15). Los valores en ASCII ya están asignados previamente.

```
1  `timescale 1ns / 1ps
2
3  module CARACTER(entrada,a);
4  input [3:0]entrada;
5  output [15:0]a;
6
7
8
9  assign a= (entrada==5'b0000)?16'b0011000100110101: //15
10  (entrada==5'b0001)? 16'b0011000100110100: //14
11  (entrada==5'b0010)? 16'b0011000100110011: //13
12 (entrada==5'b0011)? 16'b0011000100110010: //12
13  (entrada==5'b0100)?  16'b0011000100110001: //11
14  (entrada==5'b0101)? 16'b0011000100110000://10
15  (entrada==5'b0110)? 16'b0011000000111001: //09
16  (entrada==5'b0111)? 16'b0011000000111000: //08
17  (entrada==5'b1000)? 16'b0011000000110111: //07
18 (entrada==5'b1001)?   16'b0011000000110110: //06
19 (entrada==5'b1010)?  16'b0011000000110101: //05
20 (entrada==5'b1011)?   16'b0011000000110100: //04
21  (entrada==5'b1100)? 16'b0011000000110011: //03
22  (entrada==5'b1101)? 16'b0011000000110010: //02
23 (entrada==5'b1110)?  16'b0011000000110001: //01
24 (entrada==5'b1111)?  16'b0011000000110000: //00
25 16'b0011000000110000;  //00
26
27
28 endmodule
```

Figura 7-31. Código módulo contador carácter.

7.4.5. LCD

Este módulo en su mayoría es igual al de la sección anterior donde se enviaba el mensaje “EL MUNDO DE LAS FPGA”, en este caso tiene un mensaje predeterminado “ESTACIONAMIENTOS ??/15”, en donde están los signos de interrogación será el espacio donde se ubicará el mensaje a cargar. Esto se consigue simplemente considerando que el mensaje de 16 bits viene compuesto en un orden preparado, sólo basta separarlos.

```
1  `timescale 1ns / 1ps
2
3  module LCDFINAL (clk, sf_e, e,rs, rw, d,c,b,a, entrada);
4
5  input clk;  //  "C9"
6  input [15:0] entrada;
7  output reg sf_e;  // "D16"
8  output reg e;  // "M18"
9   output reg rs; // "L18"
10 output reg rw;  // "L17"
11  output reg d;  // "M15"
12  output reg c;  // "P17"
13  output reg b;  //"R16"
14  output reg a;   //"R15"
15
16 reg [26:0] count=0;
17 reg [5:0] codigo;
18 reg valorparae;
19 parameter envio= 2'b10;
20
21 always @(posedge clk) begin
22
23 count <= count + 1;
24 case (count [26:21])
25 0: codigo <= 6'h03;
26 1: codigo <= 6'h03;
27 2: codigo <= 6'h03;
28 3: codigo <= 6'h02;
29 4: codigo <= 6'h02;
30 5: codigo <= 6'h08;
```

Figura 7-32. Código módulo lcd final parte1.

```
31  6: codigo <= 6'h00;
32  7: codigo <= 6'h06;
33  8: codigo <= 6'h00;
34  9: codigo <= 6'h0C;
35  10: codigo <= 6'h00;
36  11: codigo <= 6'h01;
37  12: codigo <= 6'b100100; //E
38  13: codigo <= 6'b100101;
39  14: codigo <= 6'b100101; //S
40  15: codigo <= 6'b100011;
41  16: codigo <= 6'b100101; //T
42  17: codigo <= 6'b100100;
43  18: codigo <= 6'b100100; //A
44  19: codigo <= 6'b100001;
45  20: codigo <= 6'b100100; //C
46  21: codigo <= 6'b100011;
47  22: codigo <= 6'b100100; //I
48  23: codigo <= 6'b101001;
49  24: codigo <= 6'b100100; //O
50  25: codigo <= 6'b101111;
51  26: codigo <= 6'b100100; //N
52  27: codigo <= 6'b101110;
53  28: codigo <= 6'b100100; //A
54  29: codigo <= 6'b100001;
55  30: codigo <= 6'b100100; //M
56  31: codigo <= 6'b101101;
57  32: codigo <= 6'b010000;
58  33: codigo <= 6'b010000;
59  34: codigo <= 6'b100100; //I
60  35: codigo <= 6'b101001;
```

Figura 7-33. Código módulo lcd final parte2.

```
61  36: codigo <= 6'b100100; //E
62  37: codigo <= 6'b100101;
63  38: codigo <= 6'b100100; //N
64  39: codigo <= 6'b101110;
65  40: codigo <= 6'b100101; //T
66  41: codigo <= 6'b100100;
67  42: codigo <= 6'b100100; //O
68  43: codigo <= 6'b101111;
69  44: codigo <= 6'b100101; //S
70  45: codigo <= 6'b100011;
71  46: codigo <= 6'b001100; //para correrlo
72  47: codigo <= 6'b000000;
73  48: codigo <= {envio,entrada[15:12]};
74  49: codigo <= {envio,entrada[11:8]};
75  50: codigo <= {envio,entrada[7:4]};
76  51: codigo <= {envio,entrada[3:0]};
77  52: codigo <= 6'b100010; // /
78  53: codigo <= 6'b101111;
79  54: codigo <= 6'b100011; // 2
80  55: codigo <= 6'b100001;
81  56: codigo <= 6'b100011;//0
82  57: codigo <= 6'b100101;
83    default: codigo <= 6'b010000;
84    endcase
85    valorparae <= count [20];
86    sf_e <= 1;
87    {e, rs,rw, d,c, b,a}<= {valorparae, codigo};
88    end
89  endmodule
```

Figura 7-34. Código módulo lcd final parte3.

7.5. PRUEBAS Y ANÁLISIS

Las pruebas se realizaron en un orden que permitiera ir detectando paso por paso los errores, con las FPGA ocurre un problema y es que al diseñar un circuito en el software y simularlo realmente no asegura que vaya a funcionar de la forma en que se espera, hay que recordar que con las FPGA se crean circuitos físicos y se sabe que en la práctica los resultados no son perfectos. Por ello el primer punto fue tener el cuidado con la señal de reloj de sincronizarla para todos, enviándoles la señal a partir del mismo punto.

Uno de los problemas a enfrentar fue que el contador se debe implementar con una sincronización de reloj y esto es un gran problema ya que el reloj de la FPGA trabaja a 50 MHz y si simplemente se cuenta mientras esta activo el sensor puede resultar en un valor de millones en menos de un segundo. Por ello se tuvo que recurrir a otros métodos alternativos que no hacen el conteo directamente, pero si hacen un reflejo proporcional del valor.

La FPGA es demasiado veloz, sus velocidades de trabajo resultan un problema para la realización de este trabajo donde se mide por lapsos relativamente cortos. Un problema que se presentó con uno de los códigos que parecían exitosos cuando se probaron con switchs como entrada simulando al sensor; y al momento de conectar el sensor la señal hacía rebote, en el código realizado la medición era cada 4 ms, esto ya es un tiempo demasiado elevado y es capaz de sensar todos los cambios producidos por error.

Se eligió un ciclo de 0,8 s entre cada medición del sensor, para la aplicación que se está utilizando esto no presenta una dificultad es más por ello se solucionó todo el problema de los rebotes en el circuito.

La velocidad de ciclo del módulo del LCD debe ser consecuente a la velocidad del ciclo del contador para evitar que entre cada visualización haya más de un conteo, por esto se utiliza un contador de 27 bits, el cual permite un solo ciclo de reloj entre cada visualización del LCD.

CONCLUSIONES Y RECOMENDACIONES

CUMPLIMIENTO DE LOS OBJETIVOS

A continuación, se citarán los objetivos planteados al principio del trabajo para evaluar el cumplimiento de ellos.

El proyecto tenía como objetivo:

- Introducir al lector con un instructivo de las FPGA

El objetivo más general de los que se plantearon es el de conseguir desarrollar este trabajo de tal manera que pueda servir de introducción a esta tecnología tan poco conocida que promete y se está desarrollando de una forma que promete tomar un rol protagónico en un futuro no muy lejano, este punto es un poco difícil de evaluar, a este trabajo se le dio un formato y una orientación con el propósito de ser un material para el aprendizaje.

- Conseguir información precisa y de utilidad sobre dicha tecnología

Si se va a crear un material para el aprendizaje la información recopilada debe ser precisa por ello se buscó información de diferentes fuentes contrastando la información para asegurar su veracidad.

- Plantear comparativas de las FPGA para tener una visión más clara

Una de las formas más fáciles y mejores para generar una visión clara sobre algo es comparándolo con alguna tecnología para este caso que sea conocida como son los microcontroladores los cuales son conocidos por cualquier estudiante del área eléctrica, por ello se compararon en aspectos generales. Como esta tecnología tiene una gran variedad de fabricantes y niveles para cada modelo también se mencionó a las más importantes y la información más importante tratando de no extender demasiado la información para respetar los puntos anteriores.

- Obtener instructivo del trabajo con la Placa

Por diversos motivos el trabajo se tuvo que realizar con una placa relativamente antigua que utiliza un software ya obsoleto que actualmente está abandonado por la empresa fabricante y no resulta conveniente desarrollar un manual para ello. Lo que se hizo fue enseñar el desarrollo de un código básico como es prender un led, luego el código para el uso del LCD que tiene una complejidad mayor donde se puede observar y aprender formas de diseño de un código en Verilog y al final uno más completo que mezcla la adquisición de señales y su manipulación para generar cambios en otros elementos, en este caso el LCD.

- Conseguir realizar una implementación de la Placa Spartan-3e Starter Board donde evidenciar la utilidad de esta tecnología.

Una de las conclusiones más importante que se sacó al desarrollar la implementación es la dificultad que tiene trabajar con algo que se implementa físicamente y deja de lado el desarrollo perfecto de un programa desarrollado en un software. Las velocidades y capacidad

de realizar varias tareas en paralelo fueron un tema presente en el proyecto final desarrollado pese a que la velocidad no era algo necesaria para este proyecto con el código deja en evidencia la gran capacidad que poseen las FPGA donde hubo que reducir su capacidad para evitar otro tipo de problemas.

RECOMENDACIONES

Si se desea aprender a trabajar con las FPGA lo mejor es dedicarse a aprender un lenguaje de descripción de hardware que sea VHDL o Verilog ya que son los que van a estar disponibles en la FPGA que sea de la compañía que sea, el lenguaje Verilog tiene bastantes similitudes al C por lo cual es en general más sencillo, aunque no permite un desarrollo tan específico como podría conseguirse con VHDL. Si se va a invertir en comprar una placa para el desarrollo de circuitos a nivel de estudiante, las FPGA de gama baja son orientada a ese tipo de propósito y el software en general no presenta un problema ya que las versiones gratuitas proporcionan lo necesario para su uso.

CONCLUSIONES

El negocio de las FPGA desde el principio ha tenido un camino de buenos y malos momentos, en un principio surgió como la combinación de los ASICs y los PLD, en un principio no podían compararse con los ASICs en términos de velocidad bajo costo alta capacidad y alta velocidad. El avance de la tecnología ha jugado un papel muy importante a su favor principalmente la Ley de Moore, el que tiene un papel protagónico en el desarrollo de estos chips y ha permitido su crecimiento acelerado, que en momentos fue demasiado rápido y esto provocó la crisis de la era de la acumulación que enfrentaron de gran manera.

Actualmente las FPGA se han desarrollado de tal manera que su costo es incluso bajo comparado con otras tecnologías, obviamente esto referido a aplicaciones de nivel avanzado. No es sólo el precio, la velocidad y flexibilidad que proporcionan es superior en relación a otras tecnologías, por ello el auge que están teniendo, siendo reconocidas como la mejor alternativa para aplicaciones que necesiten de estos beneficios.

Las FPGA pueden utilizarse en una gran cantidad de aplicaciones, son capaces desarrollar desde un circuito tan simple como una compuerta and hasta sistemas complejos.

En un principio resulta difícil entender por qué recién ahora están teniendo un crecimiento y quizás si antes se pudo haber aprovechado estas placas para un desarrollo tecnológico más apresurado, para ello está la redacción de su historia que busca explicar este tema.

En algunos puntos el trabajo tomó una orientación hacia una compañía, pero la intención es tomar de ejemplo para ver cómo funciona una empresa de FPGA. Obviamente si se va a tomar una empresa de referencia lo mejor es utilizar la compañía más grande y que además

es más familiar ya que se utilizó una placa creada por ellos para el desarrollo de la implementación.

A nivel estudiante es difícil y quizá poco rentable enseñar FPGA ya que las aplicaciones que se le pueden dar en un nivel donde se pueda marcar claramente sus ventajas es a nivel avanzado y aún la velocidad de aprendizaje es más pronunciada que otras tecnologías por lo que requiere prácticamente una asignatura dedicada únicamente a estos chips.

En Chile son realmente un tema desconocido y esto se ve reflejado en las tiendas del país, donde las que ofrecen este producto son muy pocas y con una variedad de productos muy limitada, aunque ya vivimos en un mundo globalizado y comprar en el exterior está al alcance de cualquier persona, claro que teniendo que pagar costos adicionales. Estos problemas no ayudan a su desarrollo en el país y además las universidades que enseñan FPGA son muy pocas y tampoco motiva a alguien a estudiar ya que no tienen relevancia y su curva de aprendizaje es muy pronunciada, con este trabajo se busca marcar un precedente del cual se espera que sea el primer paso hacia la popularización de las FPGA al menos en la carrera.

BIBLIOGRAFIA

- ASIC-WORLD.COM. Verilog Tutorial [En línea]< http://asic-world.com/verilog/veritut.html> [Consulta Noviembre 2018].

- FPGAWARS. Explorando el lado libre de las FPGAs [En línea] <http://fpgawars.github.io/> [Consulta Agosto 2018].

- XILINX.COM. Products FPGAs & 3D ICs [En línea] <https://www.xilinx.com/products/silicon-devices/fpga.html> [Consulta Noviembre 2018].

- INTEL.COM. Intel FPGAs and programable devices [En línea] <https://www.intel.com/content/www/us/en/products/programmable/fpga.html> [Consulta Noviembre 2018].

- LATTICESEMI.COM. Lattice products [En línea] <http://www.latticesemi.com/Products.aspx#_D5A173024E414501B36997F26E842A31> [Consulta Noviembre 2018].

- Stephen M. Trimberger. Tres edades de los FPGA: una retrospectiva de los primeros treinta años de la tecnología FPGA. Proceedings of the IEEE, Volumen 103 (3): 2015. ISSN 0018-9219.

- XATAKA.COM. La importancia de los nanómetros en los procesadores [En línea] <https://www.xataka.com/componentes/la-importancia-de-los-nanometros-en-los-procesadores> [Noviembre 2018].

- DIGI-KEY.COM. Xilinx products [En línea] <https://www.digikey.com/es/supplier-centers/x/xilinx> [Septiembre 2018].

- XILINX. Programmable 7 series product selection guide. [PDF]. Estados Unidos [Consulta Septiembre 2018].

- XILINX. ISE In-Depth Tutorial [PDF]. Estados Unidos [Consulta Septiembre 2018].

- XILINX. Spartan-3E FPGA family datasheet [PDF] Estados Unidos [Consulta Diciembre 2018].

- XILINX. Spartan-3E FPGA Starter Kit Board User Guide [PDF] Estados Unidos [Consulta Diciembre 2018].

ANEXOS

ANEXO 1: GLOSARIO

Ley de Moore: Ley creada en 1965 por Gordon Earl Moore (cofundador de Intel), que previó que el número de total de transistores integrados en un circuito se duplicaría cada dos años.

RTL: Una descripción de RTL describe los registros de un circuito y la secuencia de transferencias entre estos registros, pero no describe el hardware utilizado para llevar a cabo estas operaciones. Una descripción RTL se convierte en una descripción del circuito a través de una herramienta de síntesis lógica.

Esquemático tecnológico: Este esquema muestra una representación del diseño en términos de elementos lógicos optimizados; por ejemplo, en términos de LUT, lógica de transporte, buffers de E/S y otros componentes específicos de la tecnología. La visualización de este esquema le permite ver una representación a nivel tecnológico del HDL optimizada para una arquitectura Xilinx específica.

Enrutar: Es la función de buscar un camino entre todos los posibles en una red de paquetes cuyas topologías poseen una gran conectividad. Dado que se trata de encontrar la mejor ruta posible, lo primero será definir qué se entiende por “mejor ruta” y en consecuencia cuál es la métrica que se debe usar para medirla.

Emplazar: Se emplea para aludir a la ubicación física de algo.

Resistencia de Pull-Down: Cuando se utilizan entradas digitales generalmente se emplea este tipo de resistencia, que debido a su configuración permite mantener en 0 la entrada cuando no hay señal y en 1 cuando si la hay.

Buffer: Es un espacio digital destinado para el almacenamiento temporal de información digital, mientras espera ser procesada.

Toolchain (Cadena de herramientas): Es un conjunto de programas informáticos, es decir herramientas que se utilizan para crear un determinado producto.

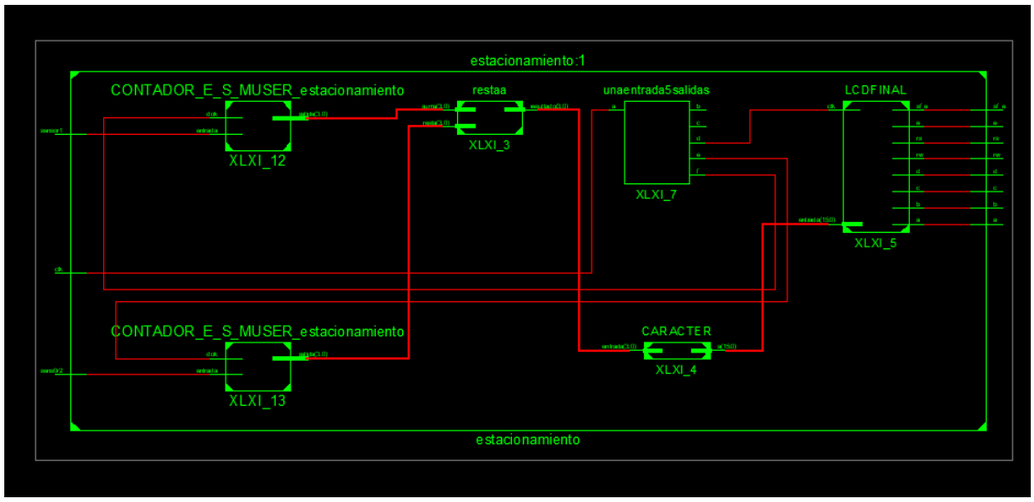
ANEXO 2: CARACTERÍSTICAS TÉCNICAS FAMILIA SPARTAN-3

			Spartan-3 FPGAs Optimized for High-Density and High I/O Designs								Spartan-3E FPGAs Logic Optimized				
	Part Number		XC3S50	XC3S200	XC3S400	XC3S1000	XC3S1500	XC3S2000	XC3S4000	XC3S5000	XC3S100E	XC3S250E	XC3S500E	XC3S1200E	XC3S1600E
Logic Resources	System Gates[1]		50K	200K	400K	1,000K	1,500K	2,000K	4,000K	5,000K	100K	250K	500K	1,200K	1,600K
	Slices[2]		768	1,920	3,584	7,680	13,312	20,480	27,648	33,280	960	2,448	4,656	8,672	14,752
	Logic Cells		1,728	4,320	8,064	17,280	29,952	46,080	62,208	74,880	2,160	5,508	10,476	19,512	33,192
	CLB Flip-Flops		1,536	3,840	7,168	15,360	26,624	40,960	55,296	66,560	1,920	4,896	9,312	17,344	29,504
Memory Resources	Maximum Distributed RAM (Kb)		12	30	56	120	208	320	432	520	15	38	73	136	231
	Block RAM (18 Kb each)		4	12	16	24	32	40	96	104	4	12	20	28	36
	Total Block RAM (Kb)		72	216	288	432	576	720	1,728	1,872	72	216	360	504	648
Clock Resources	Digital Clock Managers (DCMs)		2	4	4	4	4	4	4	4	2	4	4	8	8
I/O Resources	Maximum Single-Ended I/Os		124	173	264	391	487	565	633	633	108	172	232	304	376
	Maximum Differential I/O Pairs		56	76	116	175	221	270	300	300	40	68	92	124	156
	I/O Standards Supported		LVTTL, LVCMOS33, LVCMOS25, LVCMOS18, LVCMOS15, LVCMOS12, GTL, GTL+, HSTL15 Class I, HSTL15 Class III, HSTL18 Class I, HSTL18 Class II, HSTL18 Class III, PCI 3.3V 32/64bit 33MHz, SSTL2 Class I, SSTL2 Class II, SSTL18 Class I, Bus LVDS, LDT (ULVDS), LVDS_ext, LVDS25 & 33, LVPECL25, and RSDS25								LVTTL, LVCMOS33, LVCMOS25, LVCMOS18, LVCMOS15, LVCMOS12, HSTL18 Class I, HSTL18 Class III, PCI 3.3V 32/64bit 33MHz, PCI 3.3V 64bit/66MHz, SSTL2 Class I, SSTL18 Class I, Bus LVDS, LVDS25, LVPECL25, Mini-LVDS25, RSDS25				
Embedded Hard IP Resources	Dedicated Multipliers		4	12	16	24	32	40	96	104	4	12	20	28	36
Speed Grades	Commercial		-4, -5	-4, -5	-4, -5	-4, -5	-4, -5	-4, -5	-4, -5	-4, -5	-4, -5	-4, -5	-4, -5	-4, -5	-4, -5
	Industrial		-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4
Configuration	Configuration Memory (Mb)		0.4	1	1.7	3.2	5.2	7.7	11.3	13.3	0.6	1.4	2.3	3.8	6
	Package	Footprint Size	Maximum User I/Os												
	VQFP Packages (VQ): Very thin QFP (0.5 mm lead spacing)														
	VQ100	16 x 16 mm	63	63							66	66	66		
	Chip Scale Packages (CP): Wire-bond, chip-scale, BGA (0.5 mm ball spacing)														
	CP132	8 x 8 mm	89								83	92	92		
	TQFP Packages (TQ): Thin QFP (0.5 mm lead spacing)														
	TQ144	22 x 22 mm	97	97	97						108	108			
	PQFP Packages (PQ): Wire-bond, plastic, QFP (0.5 mm lead spacing)														
	PQ208	30.6 x 30.6 mm	124	141	141							158	158		
	FGA Packages (FT): Wire-bond, fine-pitch, thin BGA (1.0 mm ball spacing)														
	FT256	17 x 17 mm		173	173	173						172	190	190	
	FGA Packages (FG): Wire-bond, fine-pitch, BGA (1.0 mm ball spacing)														
	FG320	19 x 19 mm			221	221	221						232	250	250
	FG400	21 x 21 mm												304	304
	FG456	23 x 23 mm			264	333	333	333							
	FG484	23 x 23 mm													376
	FG676	27 x 27 mm				391	487	489	489	489					
	FG900	31 x 31 mm						565	633	633					

ANEXO 3: ARCHIVO UCF DE PROYECTO ESTACIONAMIENTO

```
1
2
3  NET "a" LOC = R15;
4  NET "b" LOC = R16;
5  NET "c" LOC = P17;
6  NET "clk" LOC = C9;
7  NET "d" LOC = M15;
8  NET "e" LOC = M18;
9  NET "rs" LOC = L18;
10 NET "rw" LOC = L17;
11 NET "sens0r2" LOC = B4;
12 NET "sensor1" LOC = A4;
13 NET "sf_e" LOC = D16;
14
15
```

ANEXO 4: ESQUEMÁTICO RTL DE PROYECTO ESTACIONAMIENTO



ANEXO 5: ESQUEMÁTICO TECNOLÓGICO DE PROYECTO ESTACIONAMIENTO