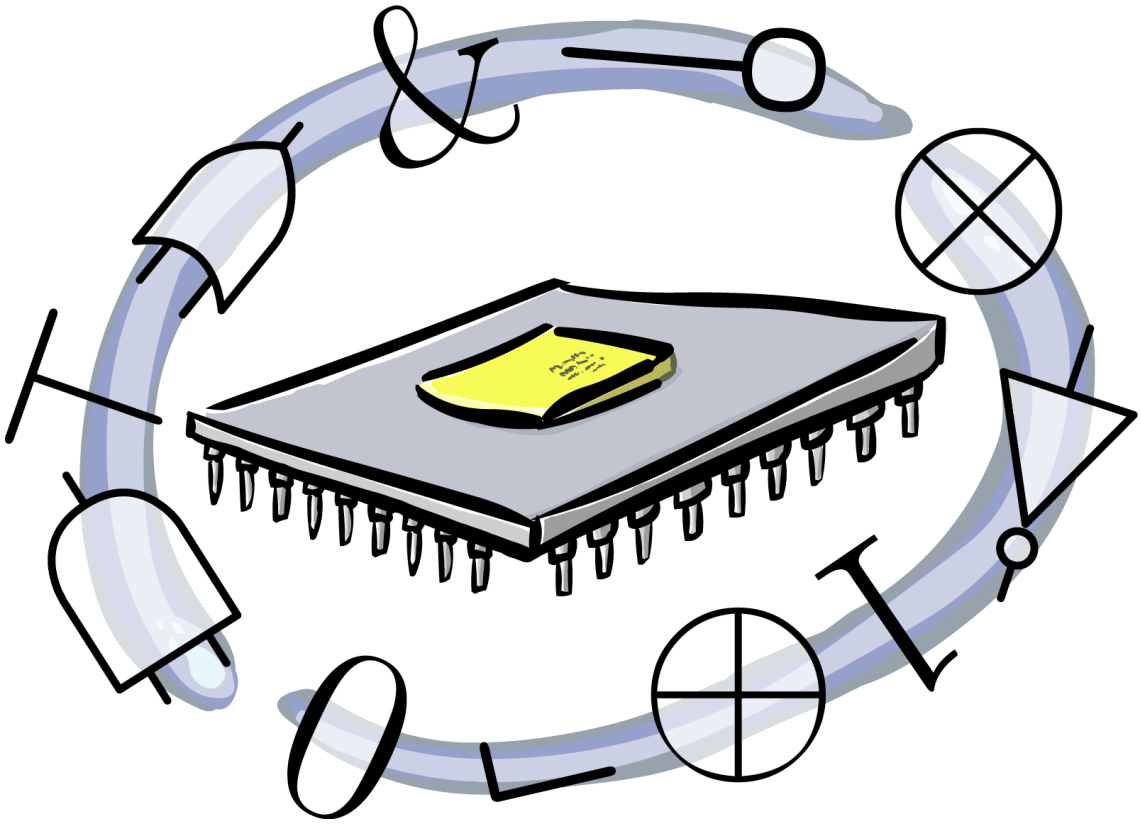
FPGA

Field Programmable Gate Array



Autores:

Javier Moreno García
Jose Durán Martín
Victor M. Delgado Hipólito
Joaquin Llano Montero

INDICE

1 COMPONENTES DEL GRUPO

Componente 1: Moreno Gracía, Javier.

Componente 2: Durán Martín, José.

Componente 3: Delgado Hipólito, Víctor Manuel

Componente 4: Llano Montero, Joaquín Manuel

2 INTRODUCCIÓN

En este trabajo vamos a tratar las FPGA (Field Programable Gate Array), o matrices de puertas lógicas programables.

Son una alternativa bastante válida con respecto a otro tipo de soluciones, como sistemas microprocesados o circuitos electrónicos dedicados.

Comenzaremos definiendo qué es una FPGA, cómo es su arquitectura.

Posteriormente comentaremos los tipos de FPGA que existen así como algunas de las aplicaciones alas que pueden destinarse.

Después comentaremos las ventajas e inconvenientes de este tipo de dispositivos para comentar seguidamente los fabricantes de FPGA más importantes.

Seguidamente comentaremos el ciclo de desarrollo a seguir cuando se trabaja con este tipo de dispositivos, la fomra de programarlos, el lenguaje que se emplea para ello, así como algunas de las herramientas que se emplean para ello, en toda esta parte haremos más hincapié puesto que existen multitud de herramientas, así como recursos con códigos libres y probados, listos para usarse. También se hablará en esta parte del hardware que se emplea para grabar este tipo de dispositivos, así como del software que lo controla.

Finalmente comentaremos algunos proyectos realizados con FPGA, todos ellos abiertos y con licencia GPL.

Por último mencionar que hemos creado un blog, donde hemos colgado la información que hemos encontrado, enlaces a fabricantes, etc. Su dirección es http://cumfpga.wordpress.com

3 OBJETOS

El objetivo de este trabajo es aprender una de las alternativas a sistemas microprocesados, así como aprender a investigar, a sacar conclusiones.

Por otra parte se varía del temario de la asignatura y aprendemos que hay más sistemas aparte de los microprocesadores para hacer computación.

Otro de los objetivos es aprender a hablar en público, a mostrar los resultados de los trabajos realizados, perder el miedo escénico, etc., algo bastante útil de cara al mundo laboral.

4 TEMAS A TRATAR

4.1 ¿QUE ES UNA FPGA?

Un **FPGA** (*field programmable gate array* = matriz de puertas lógicas programable) es un dispositivo semiconductor que contiene componentes lógicos programables e interconexiones programables entre ellos.

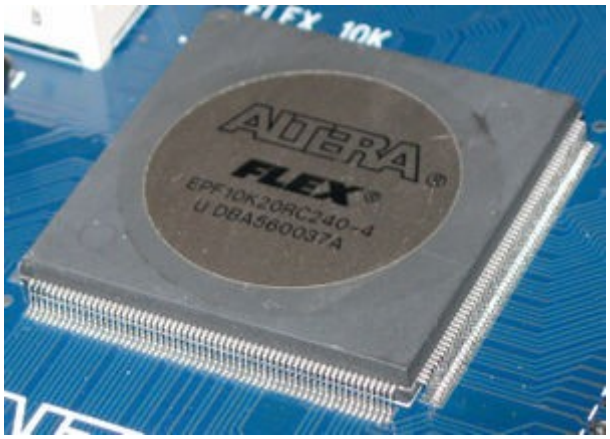


Ilustración 1Un FPGA Altera con 20.000 celdas

La raíz histórica de los FPGA son los dispositivos de lógica programable compleja (CPLD, Complex Programmable Logic Device) de mediados de los ochenta (1985). Su creador es Ross Freeman, co-fundador de Xilinx. Los componentes lógicos programables pueden ser programados para duplicar la funcionalidad de puertas lógicas básicas o funciones combinacionales más complejas tales como decodificadores o funciones matemáticas simples.

En muchos FPGA, estos componentes lógicos programables, también incluyen elementos de memoria, los cuales pueden ser simples flip-flops (biestables) o bloques de memoria más complejos.

Las FPGA consisten en una matriz bidimensional de bloques configurables que se pueden conectar mediante recursos generales de interconexión. Estos recursos incluyen segmentos de pista de diferentes longitudes, más unos conmutadores programables para enlazar bloques a pistas o pistas entre sí. En realidad, lo que se programa en una FPGA son los conmutadores que sirven para realizar las conexiones entre los diferentes bloques, más la configuración de los bloques.

4.2 ARQUITECTURA

La arquitectura de un FPGA consiste en arreglos de varias celdas lógicas las cuales se comunican unas con otras mediante canales de conexión verticales horizontales.

Cada celda lógica es funcionalmente similar a los bloques lógicos de un CLPD. La diferencia está en que un FPGA normalmente utiliza generadores de funciones en vez de puertas.

Cada uno de estos generadores es como una memoria en donde en vez de implementar la función lógica mediante puertas, se precalcula el resultado y se almacena en el generador. Las entradas al genrador funcionan como un bus de direcciones, y mediante las diferentes combinaciones de las entradas al generador se selecciona el resultado correcto. Esto le da una gran densidad al dispositivo ya que se manejan gran número de generadores, pero el tiempo de propagación al implementar una función lógica en estos generadores es menor al que se necesitaría si utilizáramos puertas. La estructura de las celdas lógicas y las formas en que estas pueden ser interconectadas, tanto salidas como entradas de la celda, varían de acuerdo al fabricante. En general una celda lógica tiene menos funcionalidad que la combinación de sumas de productos y macroceldas de un CPLD, pero como cada FPGA tiene una gran cantidad de celdas lógicas es posible implementar grandes funciones utilizando varias celdas lógicas en cascada. Además de las celdas lógicas también es importante la tecnología utilizada para crear las conexiones entre los canales, las más importantes son las siguientes:

- La tecnología **SRAM** es utilizada por Altera, Lucent Technologies, Atmel, Xilinx y otros.
- La tecnología **ANTIFUSE** es utilizada por Cypress, Actel, QuickLogic, y Xilinx.

4.2.1 ANTIFUSE

Al igual que la tecnología PROM, un FPGA que utiliza este tipo de tecnología sólo se puede programar una sola vez y utilizan algo similar a un fusible para realizar las conexiones. Una vez que éste es programado ya no se puede recuperar. La diferencia radica en que en un fusible normal se desactiva deshabilitando la conexión, en tanto que en estos *anti - fusibles* cuando son programados se produce una conexión por lo que normalmente se encuentran abiertos. La desventaja obvia es que no son reutilizables, pero por el contrario disminuyen considerablemente el tamaño y costo de los dispositivos.

4.2.2 SRAM

Las celdas SRAM son implementadas como generadores de funciones para remplazar la lógica combinacional mediante compuertas y, además, son usadas para controlar multiplexores e interconectar las celdas lógicas entre ellas. En estas el contenido se almacena mediante un proceso de configuración en el momento de encendido del circuito que contiene al FPGA. Ya que al ser SRAM, el contenido de estos bloques de memoria se pierde cuando se deja de suministrar la energía.

La información binaria de las celdas SRAM generalmente se almacena en memorias seriales EEPROM conocidas como memorias de configuración. En el momento de encendido del circuito toda la información binaria es transferida a las celdas del FPGA mediante el proceso de configuración el cual es generalmente automático y el propio FPGA contiene un circuito interno que se encarga de hacer todo el proceso.

Un FPGA que tiene una gran cantidad de canales de interconexión tiende a tener pequeñas celdas lógicas con muchas entradas y salidas en comparación con el número de compuertas que tiene la celda, este tipo de FPGAs generalmente utilizan tecnología ANTIFUSE.

Un FPGA que tiene una estructura pequeña en canales de interconexión tiende a tener grandes celdas lógicas con pocas entradas y salidas en comparación con el
número de compuertas que hay en la celda. Este tipo de FPGA generalmente está fabricado con tecnología SRAM.

Una arquitectura con celdas lógicas pequeñas permite utilizar todos los recursos del dispositivo. Sin embargo, si las celdas lógicas son muy pequeñas entonces tendremos que utilizar un gran número de estas para poder implementar funciones lógicas de varios términos, lo cual agrega un tiempo de retardo por cada celda lógica implementada.

Cuando el tamaño de la celda lógica es grande sucede lo contrario, en este tipo de celdas lógicas es posible utilizar un gran número de compuertas por lo que podemos implementar funciones lógicas de varios términos con pocas celdas lógicas. El que el tamaño de la celda sea grande no afecta la frecuencia máxima de trabajo porque estamos hablando de que existe un gran numero de compuertas que pueden ser usadas en la función paralelamente, siendo el mismo tiempo de retardo para todas. Sin embargo, cuando las funciones son pequeñas en comparación con el tamaño de la celda no es necesario utilizar todas las compuertas de la celda, por lo que este tipo de celdas no son precisamente las más indicadas para desempeñar pequeñas funciones.

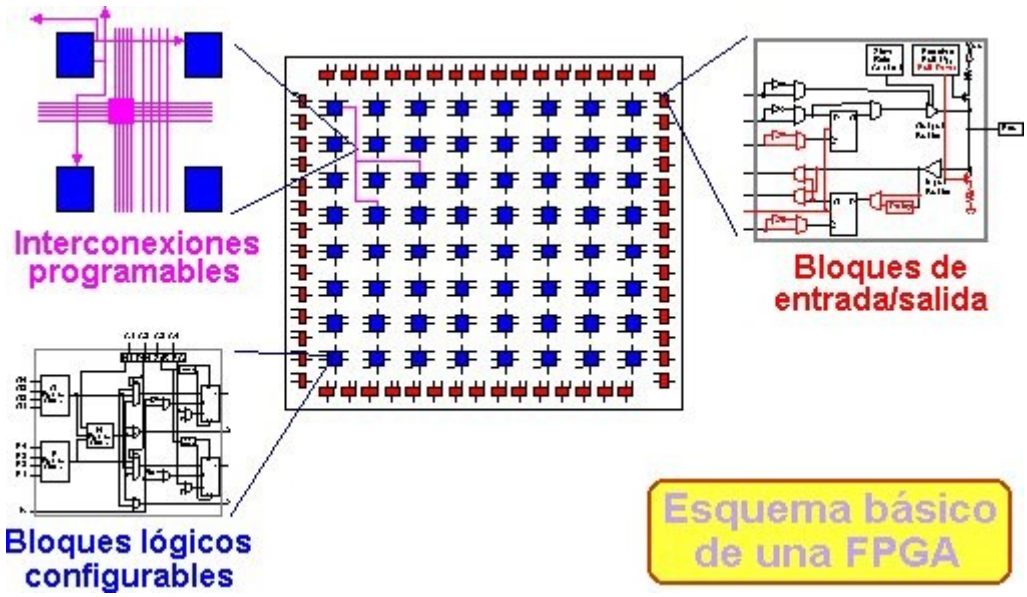


Ilustración 2 Esquema básico de la arquitectura de una FPGA

A continuación se muestran algunas celdas lógicas de distintos fabricantes.

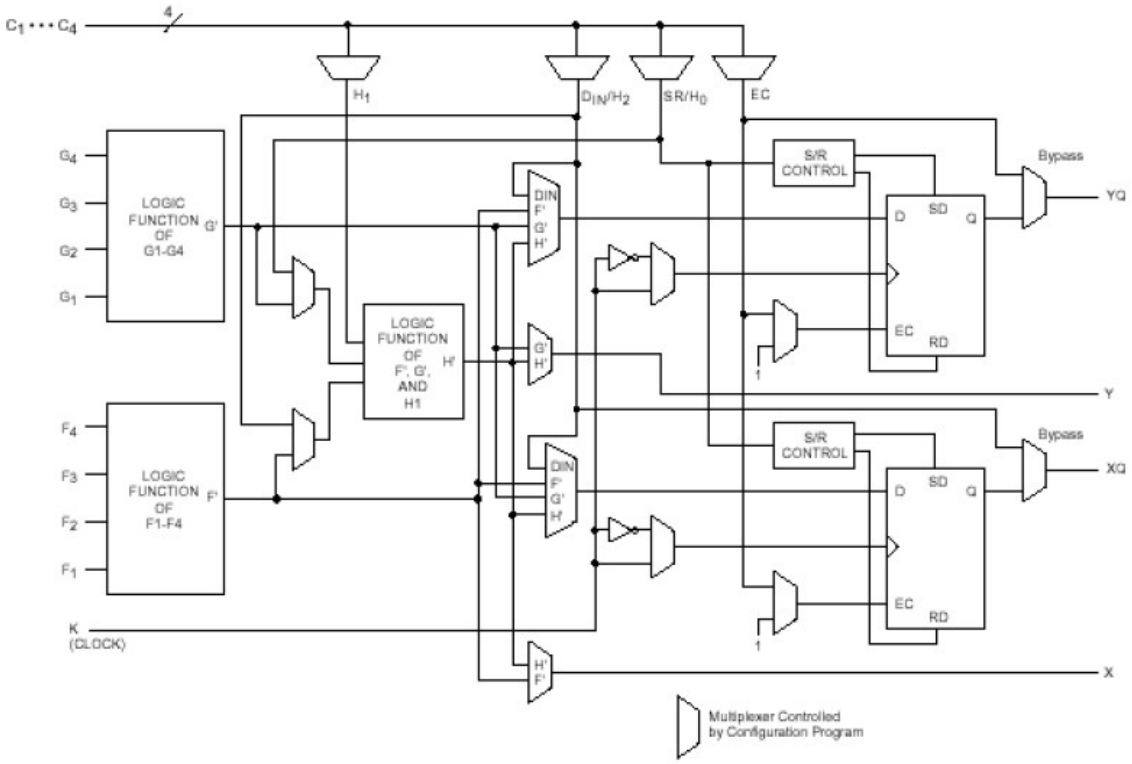


Ilustración 3 Bloque lógico configurable de la familia XC4000 de Xilinx, inc.

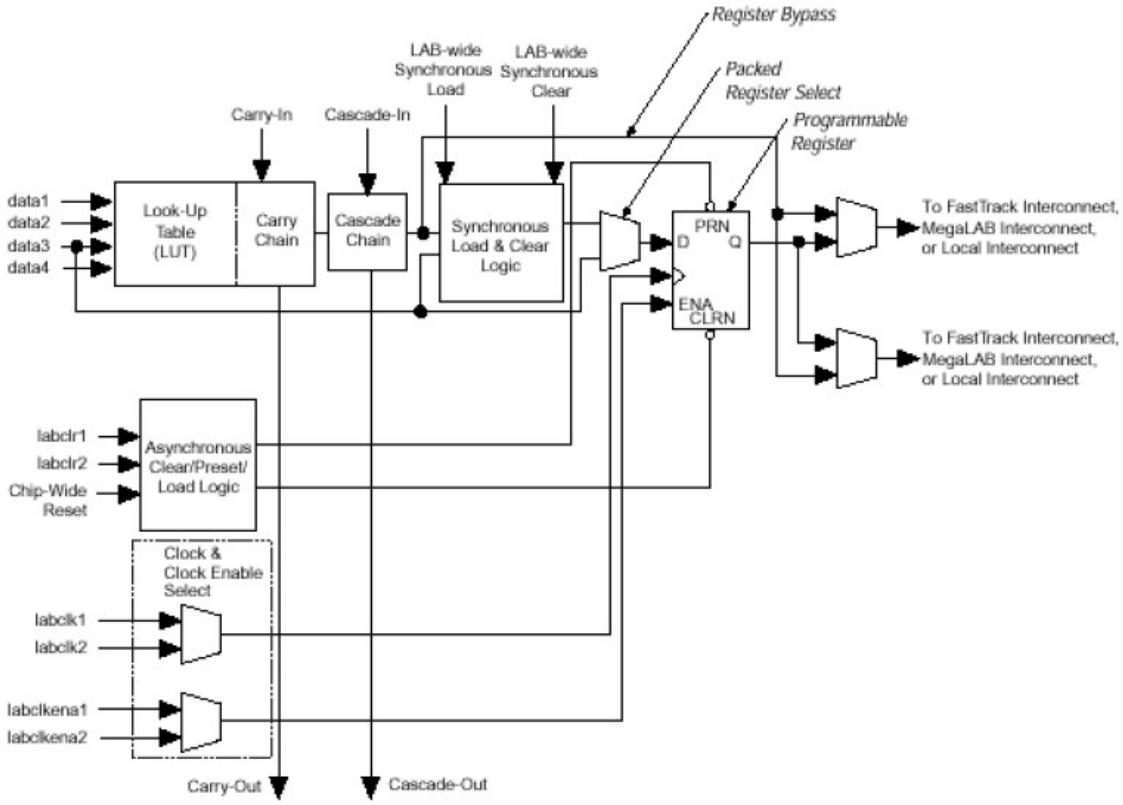


Ilustración 4 Elemento lógico de la familia APEX20K de Altera Corporation.

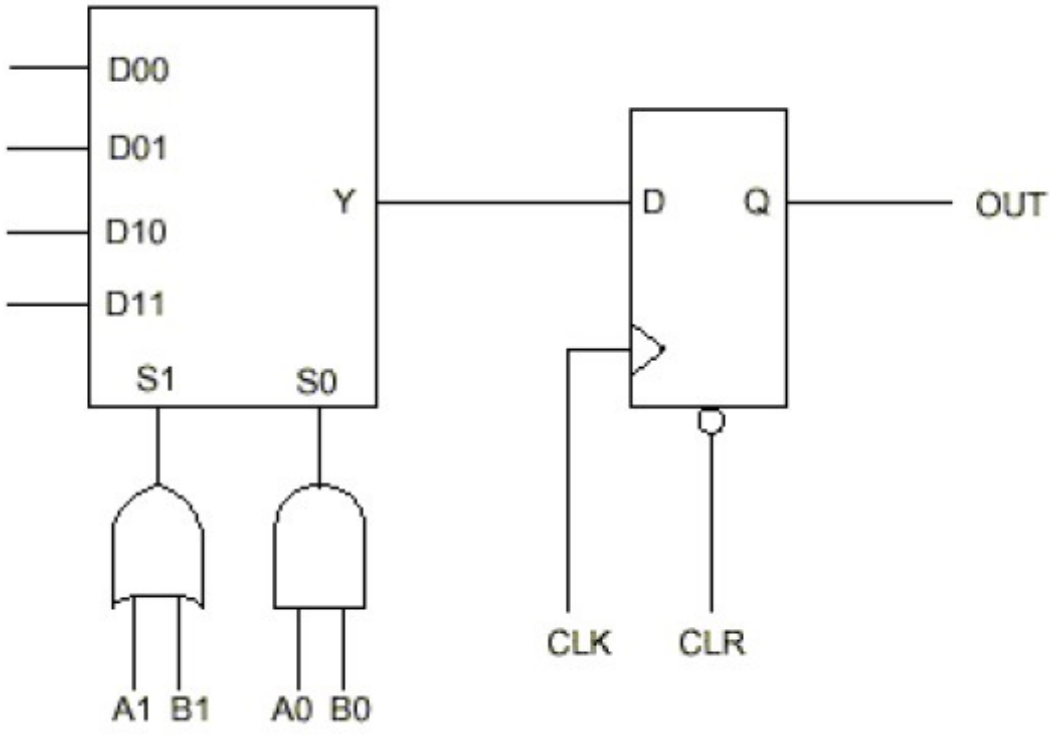


Ilustración 5 Módulo lógico de la familia ACT3 de Actel Corporation.

http://es.wikipedia.org/wiki/FPGA

http://fpgalibre.sourceforge.net/

4.3 TIPOS

Por la tecnología de la memoria de programación:
Volátiles-> Basadas en RAM
Su programación se pierde al quitar la alimentación. Requieren una memoria externa no valátil para configurarlas al arrancar (antes o durante el reset)
No volátiles-> Basadas en ROM
Reprogramables-> Basadas en EPROM (Enable-Programmable ROM) o flash. Se borran y se pueden volvera programar.
No reprogramables-> Basadas en fusibles.
Sólo se pueden programar una vez. Noaptas para laboratorios, pero sí para el espacio.

4.4 APLICACIONES

En un FPGA cualquier circuito de aplicación específica puede ser implementado, siempre y cuando esta disponga de los recursos necesarios. Las aplicaciones donde más comúnmente se utilizan los FPGA son entre otras cosas:

- Radio definido por software
- Sistemas aeroespaciales y de defensa
- Sistemas de imágenes para medicina
- Sistemas de visión para computadoras
- Reconocimiento de voz
- Bioinformática
- Emulación de hardware de computadora

Las áreas en las que las FPGA son usadas, son sobre todo en aquellas aplicaciones que requieren un alto grado de paralelismo.

4.5 VENTAJAS Y DESVENTAJAS

Ventajas:
- Bajo costo de las herramientas
- Verificación efectiva del diseño mediante simuladores o en el chip
- Ventajas de ser un producto de fabricación estándar
- Ventajas de ciclo de vida de una aplicación
- La densidad de integración crece bastante (hasta 4 millones de puertas)

Desventajas:
Tamaño y costo del chip
No pueden competir en velocidades máximas ni en consumo, pero muchas aplicaciones no son críticas en estos factores.

4.6 FABRICANTES

A principios de 2007, el mercado de los FPGA se ha colocado en un estado donde hay dos productores de FPGA de propósito general que están a la cabeza del mismo, y un conjunto de otros competidores quienes se diferencian por ofrecer dispositivos de capacidades únicas.

- Xilinx, es uno de los dos grandes líderes en la fabricación de FPGA.
- Altera, es el otro gran líder.
- Lattice Semiconductor, lanzó al mercado dispositivos FPGA con tecnología de 90nm. En adición, Lattice es un proveedor líder en tecnología no volátil, FPGA basadas en tecnología Flash, con productos de 90nm y 130nm.
- Actel, tiene FPGAs basados en tecnología Flash reprogrammable. También ofrece FPGAs que incluyen mezcladores de señales basados en Flash.
- QuickLogic, tiene productos basados en fusibles (programables una sola vez).

1. Atmel, es uno de los fabricantes cuyos productos son reconfigurables (el Xilinx XC62xx fue uno de estos, pero no están siendo fabricados actualmente). Ellos se enfocaron en proveer microcontroladores AVR, con FPGAs, todo en el mismo encapsulado.
   - *Achronix Semiconductor*, tienen en desarrollo FPGAs muy veloces. Planean sacar al mercado a comienzos de 2007 FPGAs con velocidades cercanas a los 2GHz.
   - *MathStar, InC*., ofrecen FPGA que ellos llaman FPOA (Arreglo de objetos de matriz programable).

4.7 CODIGO HDL

Selección del HDL para realizar los diseños.

Existen varios HDLs que pueden utilizarse para realizar la descripción de un diseño. Los más utilizados y populares son Verilog, VHDL y SystemC. Debido a que el VHDL es de uso común en muchas instituciones universitarias y proyectos de gran importancia.

El lenguaje VHDL está definido en el estándar IEEE 1076. VHDL es el acrónimo que representa la combinación de *VHSIC* y *HDL*, donde **VHSIC** es el acrónimo de ***Very High Speed Integrated Circuit*** y **HDL** es a su vez el acrónimo de ***Hardware Description Language***.

4.8 CORES

Se llaman cores al código fuente de sistemas como microprocesadores, microcontroladores, filtros, módulos de comunicación y memorias entre otros.

Su funcionamiento se puede interpretar como el de un puente entre los dos buses, el PCI y el Wishbone, es decir envía y recibe datos de un bus a otro.

La ventaja de pasar de un tipo de bus a otro radica en que la especificación Wishbone esta pensada para interconectar diseños dentro de un mismo integrado. Las interfaces y los ciclos de transferencia de datos son iguales para todos los cores IP sin importar su función (controlador de memoria, interfaz PCI, registros, etc.) y no es necesario conocer el funcionamiento del bus PCI para hacer un diseño. Esto además de facilitar la tarea, permite la reutilización. Si ya existe un controlador de memorias con interfaz Wishbone, basta con diseñar la interconexión entre dicho controlador y el core IIE-PCI.

Existen una gran cantidad de ***cores IP*** disponibles en la red, con licencias que permiten su uso sin ningún tipo de restricciones.

Se descarta la utilización de ***cores IP*** que no sigan los principios del software libre, por no brindar el código fuente, restringir su campo de aplicación o solicitar el pago de *royalties*.

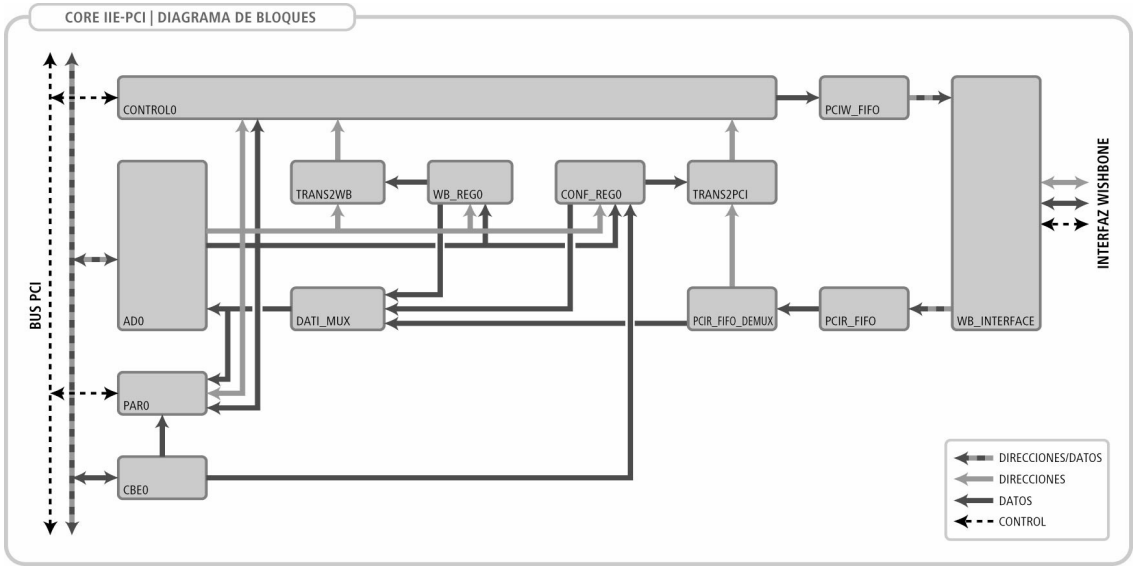
De todos los ***cores IP*** con estas licencias, este proyecto tiene preferencias por aquellos escritos en **VHDL**, que compilen correctamente con GHDL y posean interfase Wishbone.

Existen gran cantidad de *cores* IP disponibles en la red, con licencias que permiten su uso sin ningún tipo de restricciones.

Algunos de estas webs son: Wishbone o OpenCores.

En OpenCores encontramos:

- SPIFlash FPGA te permite configurar Altera y Xilinx FPGAs usando el flash de SPI y un microcontrolador pequeño.



4.9 SIMULACION

Herramienta de simulación.

Los simuladores nos permiten verificar el funcionamiento de nuestra descripción de hardware sin necesidad de pasar a la síntesis. Todos los simuladores importantes convierten nuestro *código* HDL en un binario ejecutable que se comporta como nuestro hardware simulando los cambios de señales. A través del uso de *testbenches* (bancos de prueba) es posible realizar una verificación del funcionamiento de nuestro hardware e informar si algo no funciona como se esperaba.

Algunos de las herramientas que se pueden utilizar son : SAVANT, FreeHDL, Verilog ,ABEL Alliance y GHDL.

4.10 EXTENSIONES Y ASISTENTES A LENGUAJES HDL

Comparado con otros lenguajes de programación como el C, el lenguaje VHDL posee limitaciones que dificultan y demoran el desarrollo. Para solucionar algunos de estos inconvenientes se desarrollan herramientas complementarias para el lenguaje. Estas son algunas herramientas desarrolladas hasta ahora:

1. VHDLSPP (VHDLSimple Pre-Processor)

Permite fácilmente incluir archivos. Similar a la sentencia *#include* del lenguaje C.

En ocasiones es necesario generar ciertas estructuras en forma automática evitando transcri- birlas cada vez que se realiza un cambio. Un ejemplo es cuando deseamos inicializar un bloque de memoria con una imagen o el código que correrá nuestra CPU.

En lenguajes como el C cuando se desea hacer esto basta con colocar los datos en un header (archivo de encabezado) y utilizar la directiva #include del preprocesador de C. En VHDL no existe el concepto de preprocesador y por lo tanto esto no es posible.

Con esta finalidad se creó vhdlspp[11]. Este script Perl permite reemplazar tags (marcadores) del tipo @nombre_archivo@ por el contenido del archivo indicado. A diferencia de C esto puede colocarse en cualquier lugar de nuestro código VHDL.

2. HEX2VHDL

Conversor de archivos en formato HEX a un array **VHDL**. Utilizado por ejemplo para incluir el contenido de la memoria de programa de un procesador dentro de un diseño.

Cuando creamos nuestro core compatible con el PIC16C84 nos encontramos en la necesidad de incluir los .hex que contenían el código del PIC en nuestros fuentes VHDL para inicializar la memoria de programa del controlador.

Para solucionar este problema creamos esta herramienta que trabajando en conjunto con
vhdlspp y GNU Make nos permitieron que bastara con modificar el código de ensamblador
para el PIC para que nuestro bitstream se actualizara conteniendo el programa para el microcontrolador.

3. XTRACTH

Este es un script muy simple para extraer definiciones de un *package* **VHDL** y colocarlas en un .h y/o un .inc.

Un problema muy frecuente que aparece cuando se diseña un periférico que será mapeado en una dirección de memoria o entrada/salida es que cuando se realizan cambios en el orden de sus registros se hace necesario actualizar los cambios en el código fuente que accede al periférico.

Para facilitar esta tarea desarrollamos xtracth[13]. Xtrach sólo necesita el nombre del package donde se definieron las constantes VHDL que definen los registros en cuestión. A partir del package xtrach extrae las constantes y las declara en un .inc y en un .h. Usando xtrach en conjunto con GNU Make se puede automatizar por completo esta tarea.

Cuando se utiliza el bus Wishbone xtrach se complementa con WISHBONE Builder que puede generar archivos .inc y .h con las direcciones base de los periféricos.

4. OTROS

4.1 SETEdit, Edición del Código Fuente

4.2 BAKALINT, Verificación de Estilo

4.3 INTERCON, Generador de Interconexión

4.4 EUBERANT C Tags, Búsqueda en el código HDL

4.11 LIBRERIAS HDL

Existen varias librerías que facilitan y aceleran el desarrollo de *cores*.

La IEEE creó la libreria IEEE VHDL y el tipo std_logic con el standard 1664. Fue ampliada por Synopsys, cuyas distribución es libre.

Partes de la libreria IEEE pueden ser inclidas en una entidad, por la inserccion de lineas, antes de la declaración de la entidad.

Ejemplo**:**

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
```

Estas son las mas utilizadas:

1. std_logic_1164

Se basa en el manejo básico de bits. Pertenece a la IEEE .

Esta es la librería que define los tipos basicos ***std_logic*** y algunas funciones. Probablemente este inlcuida en cada entidad creada.

El codigo std_logic_1164, no incluye informacion sobre permisos o copyright.
Se encuentra en la dirección ${SYNOPSYS}/packages/IEEE/src/.

2. numeric_std

Manejo básico de enteros, Pertenece a la IEEE .

Caracteristicas:

- Con signo: complemento a 2. (signo + valor absoluto).
- Sin Signo: Representación binaria de enteros positivos.
- Operadores matemáticos sobrecargados.
- Permite mezcla de vector y valores enteros. (vector <= vector + 1)
- Operadores relaciones sobrecargados .
- Evita problemas al ocuparse de diversas longitudes del vector.
- Comparación de vectores con valores enteros.

El uso de 'bit' y 'bit_vector' no esta recomendado.

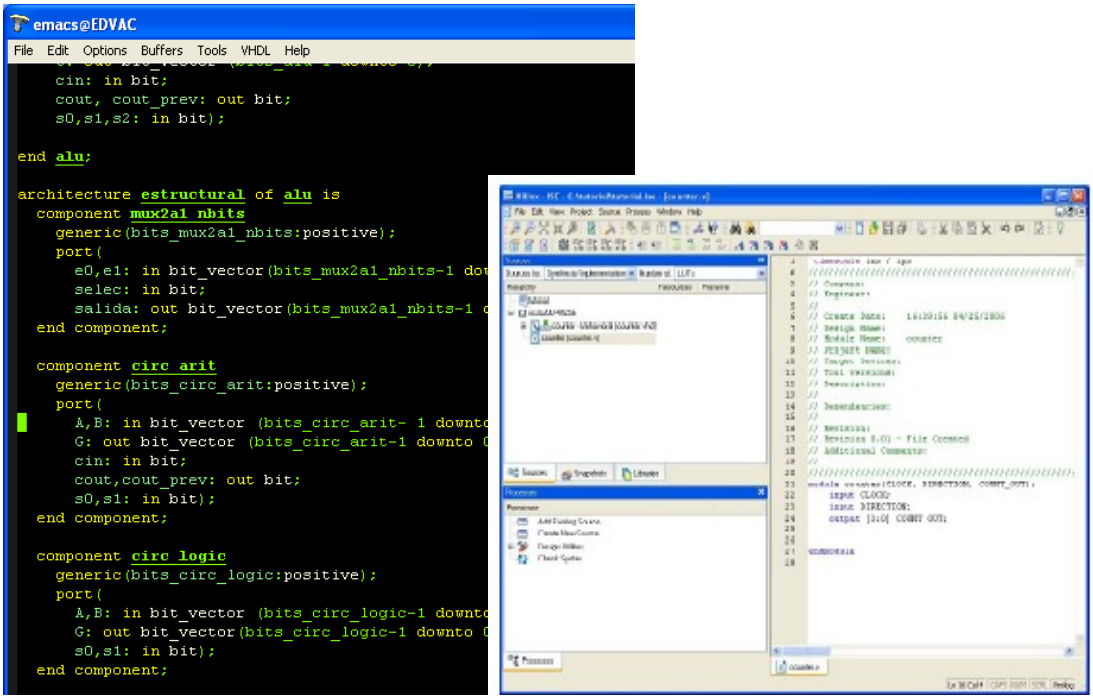
3. Otras std_logic

3.1 _arith, define algunos tipos y operaciones aritmeticas basicas. (Extensión de Synopsys).
3.2 _unsigned,
3.3 _signed,

El proyecto **FPGA Libre** recomienda el uso de los tipos definidos en **std_logic_1164** y **numeric_std** y no los definidos por Synopsys.

4.12 EDICION DEL CODIGO

Para editar el código puede emplearse cualquier editor de texto simple, incluso el bloc de notas de Windows, aunque existen editores



más potentes como GNU Emacs, SetEdit, etc.

Los fabricantes también ponen a disposición de los desarrolladores entornos integrados de desarrollo, así como simuladores, etc.

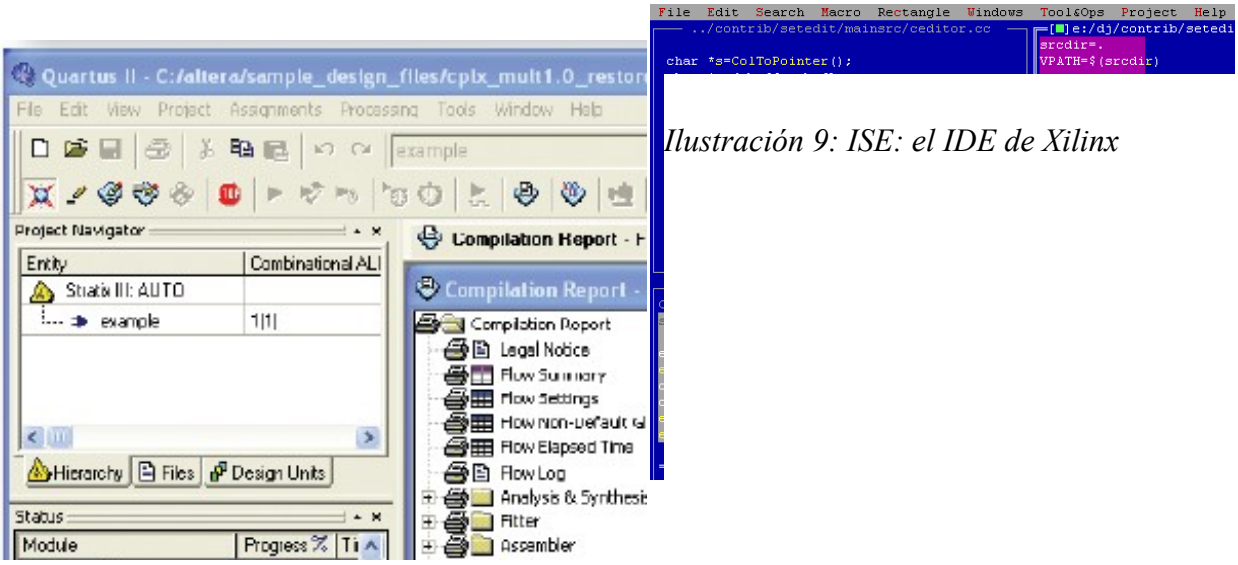


Ilustración 9: ISE: el IDE de Xilinx

4.13 SINTESIS

La síntesis es el proceso por el cual convertimos nuestro HDL en una descripción de bajo nivel que permitirá realizar nuestro hardware. En el caso de las **FPGA** esto es simplemente la configuración de las macroceldas y recursos de interconexión de la misma (entre otros).

Entre las herramientas de síntesis que proporcionan los fabricantes de FPGA, cabe destacar ISE, de Xilinx, que funciona tanto en entronos Windows como Linux.

4.14 TRANSFERENCIA A LA FPGA

Para transferir la sínteis a la FPGA es necesario emplear un hardware específico, así como un software que gestione dicho harware. Cada fabricante proporciona ambos para sus dispositivos, aunque tambíen existen alternativas libres, como es el caso del Jbit.

Casi todas las FPGA soportan el estándar IEEE 1149 (JTAG), por lo que disponiendo de una interfaz como esta podemos programarlas. Esta interfaz es relativamente fácil de construir, además de barata.

Además de para programar, la interfaz JTAG permite comunicar dispositivos entre sí e incluso la depuración de los mismos.

a interfaz JTAG consta de 4 o 5 líneas, una de entrada de datos, otra de salida de datos, otra para el reloj, una para seleccionar el modo de testeo y una opcional de reset.

No sólo las FPGA disponen de este tipo de interfaz, de hecho está presente en muchos dispositivos, como por ejemplo los decodificadores de satélite, con lo que se facilita enormemente su reparación, actualizacipon de firwmares, etc.

Ejemplo de Interfaz JTAG para puerto paralelo:

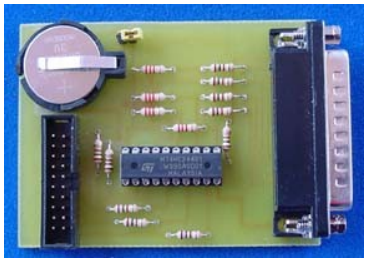


Ilustración 10: Programador para FPGAs de Lattice

Además cada fabricante nos proporciona kits de desarrollo para sus productos, en el ámbito del hardware / software libre tenermos S2PROTO.

4.15 PROYECTOS FPGA

1 Tarjeta gráfica libre

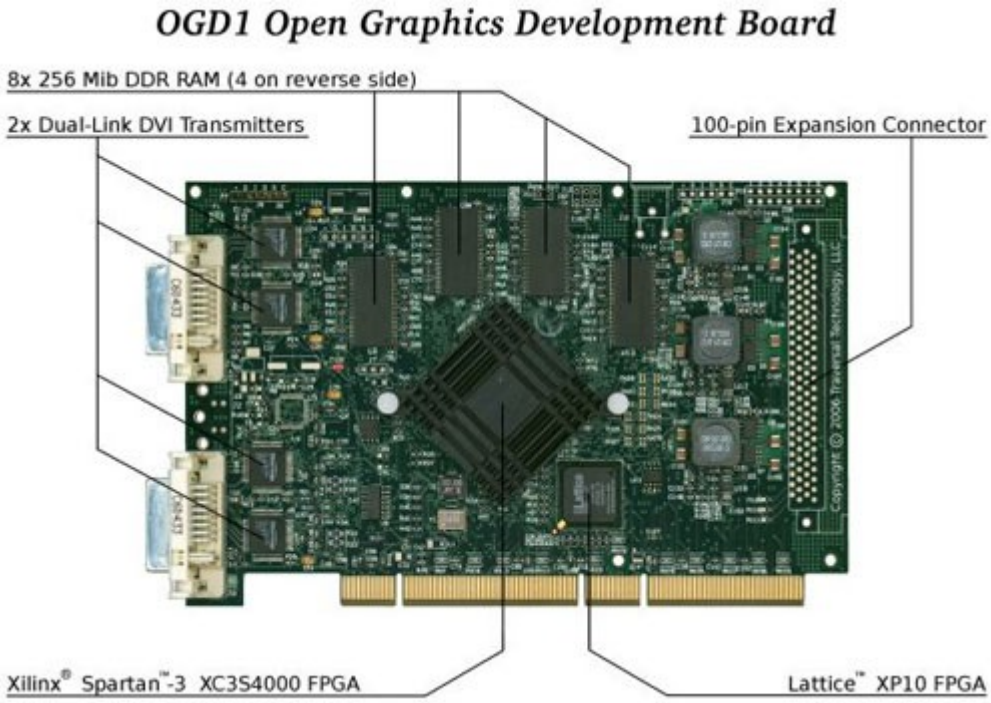


Ilustración 11: Tarjeta gráfica libre

Esta tarjeta posee 256MB de RAM, dos conectores DVI, soporte para otras dos salidas de vídeo (televisión analógica y vídeo). La GPU está formada por dos FPGA, una Xilinx Spartan-3 y una Lattice XP10. La Xilinx tiene una interfaz de memoria incorporada de 128 bits, lo que le permite un ancho de banda de 1,6GB/s.Puede soportar una resolución de hasta 2560x1600 píxeles. Se conecta al bus PCI.

2 Proyecto Cyclope

Es un SOC (system on a chip), que integra un computador dentro de una FPGA. Está orientado al visión artificial.

En la FPGA se encuentra un procesador Z80 para comandar el resto del sistema, una capturadora de vídeo, el propio sistema de visión artificial, salída de vídeo VGA, dos puertos serie RS232, un controlador programable para 6 servomotores y 4kB de memoria ROM.

Para construir este artificio, primero se emularon los módulos como simulaciones en lenguaje C, para después migrarlos a VHDL, con esto se ahorra mucho tiempo de depuración.

3 Sistema de comunicación seguro

Este diseño se ha implementado con dos tarjetas de Xilinx, que incorporan entre otros componentes, una FPGA XC4010XL y 32k de RAM.

El proyecto trata de comunicar las dos FPGA mediante un protocolo serie asíncrono, pero en el que los datos viajen encriptados, de forma que sólo puedan leerse con la clave que fueron cifrados.

Las principales características del sistema de comunicación son las siguientes:

- **Posibilidad de comunicación bidireccional**. Aunque por motivos de capacidad se ha implementado la transmisión solo en un sentido, los módulos están diseñados para que un transmisor y un receptor puedan convivir en la misma placa.
- **Clave de codificación de 64 bits** (8 bytes)
- **Palabra de datos de 32 bits**. Es la longitud del dato que se envía por la línea serie. Tanto codificador/decodificador como transmisor y receptor trabajan con datos de 32 bits.
- **Velocidad de transmisión variable**, sin más que cambiar un valor en el código. Por defecto se encuentra a 375 Kbits/s.
- Se incorpora un **control de flujo** para regular la transmisión: el receptor puede frenar al transmisor si aún no ha procesado el dato anterior.
- **Algoritmo de codificación reversible**: el mismo módulo sirve para codificar y para decodificar.
- **Funcionamiento totalmente transparente al usuario**.

4 Proyecto OpenUP, microprocesador abierto.

Desarrollado por la Universidad de Valladolid, se trata de un procesador CSIC de 8 bits, con 12 bits para el bus de direcciones y 32 registos. Se denomina uP1232.
Se distribuye bajo licencia GPL y según sus autores se propone su uso para producción en serie.

Tiene las siguientes características:

- Contiene **32 registros** internos de 8 bits, actuando todos ellos como acumuladores.
- Su código ejecutable, dependiendo de la versión, varía entre 4 KB y 64 KB.
- Accede a **64 KB** de memoria externa (datos, código ejecutable y puertos de E/S).

- Dispone internamente de hasta 256 bytes de RAM, con un mínimo de 32 bytes.
- La pila también es interna, tiene sólo 32 palabras, pero es ampliable.
- Su código de instrucciones dispone de 33 operaciones distintas.
- Las instrucciones requieren sólo uno o dos bytes en su codifiÿación.
- Su ALU de 8 bits puede realizar 16 operaciones distintas sobre cualquier registro.
- Responde a cuatro flags: cero, acarreo, signo y paridad.
- Atiende tres interrupciones priorizadas y enmascarables individualmente.
- En su momento, dispondrá de 15 puertos de entrada/salida.
- Ocupa unas **200 CLBs** en una XC4000E (unas 5000 puertas equivalentes).

Inicialmente se ha implementado en una XC4000E de Xilinx, aunque puede migrarse fácilmente a las de Atmel y Lucent.

Además han desarrollado un ensamblador, un simulador y un emulador para el procesador.

Todo se distribuye bajo licencia GPL.

- Proyecto OpenDSP, procesador digital de señal abierto.
Denominado DSPuva16, opera en coma fija de 16 bits y hasta 24 bits con precisión extendida.
Está enfocado a aplicaciones de filtrado, donde no haya que manipular demasiada información simultáneamente, por ejemplo electrónica de potencia, convertidores de CC a CA, control de motores de CA, filtros activos, etc.

Las principales características del procesador son las siguientes:

- Dispone de **16 registros** internos de 24 bits (**r0** a **r15**), actuando todos ellos como acumuladores excepto **r0**.
- Su código ejecutable, dependiendo de la versión, varía entre 256 y 4K palabras de 16 bits.
- Multiplica y acumula en coma fija normalizada de 16/24 bits con signo en un sólo ciclo de instrucción.
- Su ALU de 24 bits puede realizar hasta 8 operaciones distintas sobre cualquier registro.
- Opera con dos registros sobre un tercero (rD = rS op rT) o con registro y constante (rD = rS op K).
- Su código de instrucciones, altamente ortogonal, dispone de 17 operaciones distintas.

- Permite las comparaciones habituales: eq, ne, gt, lt, ge, le, ov, nv.
- Accede a **256** puertos de entrada/salida de 16 bits.
- No dispone de pila, pero se puede montar externamente.
- De momento no dispone de interrupciones.
- Ocupa unos 250 slices (500 LCs) en una Spartan-II de Xilinx.

5 CONCLUSIONES

El empleo de las FPGA como alternativa a los sistemas microprocesador a veces conlleva bastantes ventajas.

Aunque no son capaces de superar en velocidad de proceso a un computador, tienen la ventaja de que pueden integrar todo un sistema computacional dentro de un único chip, lo cual supone un ahorro en espacio, consumo y calor. Por otra parte son sistemas mucho más abiertos que los microprocesados son reprogramables.

Al poderse emplear HDLs para programarlos, tienen la ventaja de que en algunas aplicaciones su ejecución es mucho más rápido, puesto que la FPGA va a comportarse como un circuito electrónico hecho a la medida de la aplicación a satisfacer.

Cabe mencionar que muchos modelos de procesador se han probado primero en FPGA antes de construir los chips, lo que indica la potencia de este tipo de dispositivos.

Como hemos visto en los apartados anteriores los campos de aplicación de las FPGA llegan hasta donde podamos imaginar: visión artificial, telecomunicaciones, proceso digital de señales, robótica, tarjetas gráficas…, el límite está en la imaginación del diseñador.

Otra de sus ventajas es la interfaz de programación, casi todos los dispositivos se programan me diate la interfaz JTAG, que además permite comunicarlos e incluso conectarlos en serie para programar varios dispositivos conectados al mismo bus.

Por otra parte la industria aeroespacial las emplea bastante, básicamente porque permiten almacenar todo un sistema dentro de un único chip, con el consiguiente ahorro en peso. De hecho algunos fabricantes tienen series especiales de FPGA resistentes a radiaciones cósmicas.

Como conclusión final decir que nos ha resultado bastante interesante la realización de este trabajo, y nos hemos dado cuenta de las posibilidades del empleo de este tipo de dispositivos, así como la potencia de los mismos.

6 Herramientas

No hemos necesitado ninguna herramienta, ya que nuestro trabajo se ha basado en recabar información, siendo teorico y no práctico. En cualquier caso, en el interior se comentan algunas de las utilizadas en este proyecto.

7 BIBLIOGRAFIA

[Plaza] J. Plaza: "*Apuntes asignatura EC*".

[Intel] Intel: "Página web de Intel" (http://www.intel.com).

Direcciones Web:

Proyecto fpgalibre
Web del profesor
FPGA según la Wikipedia
Foro sobre FPGA
Web de Altera (fabricante)
Web de Xilinx (fabricante)
Web de Atmel (fabricante)
Web sobre FPGA
Microprocesador abierto
Noticia sobre la tarjeta grafica abierta
Proyecto de la tarjeta grafica de codigo abierto
Cores libres para FPGA
Proyecto Ciclope (visión artificial)
Sistema de comunicación seguro con FPGA (Universidad de Sevilla)
DSP Abierto
Interesante introducción a las FPGA
Tesis sobre vision artificial con FPGA (PDF)
Introduccion a las FPGA
Más info sobre FPGA
Sistemas operativos en FPGA
Dispositivos PCI
Cores de un fabricante
Sintesis con HDL
FPGA Libre