FIELD PROGRAMMABLE GATE ARRAYS (FPGAS)

Cátedra: Sistemas Digitales II / Electrónica Digital II

Autor: Cristian Sisterna, MSc

Tabla de contenido

Lista de Figuras

1. Introducción

Los dispositivos Field Programmable Gate Arrays, en español Arreglos de Compuertas Programable en el Campo, tal como su nombre lo indica son un arreglo (arrays) matricial de bloques lógicos (gates) programables (programmable) en cualquier espacio físico (field).

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

Un FPGA es un dispositivo que un diseñador de sistemas digitales puede programar, después que está soldado en el circuito impreso, para que funcione de un modo determinado. Los FPGAs son fabricados con conexiones y lógica programables. El diseñador desarrolla su sistema digital usando herramientas tipo EDA (Electronics Design Automation), sean dibujos esquemáticos o lenguaje de descripción de hardware (como VHDL), para poder plasmar el sistema en lógica digital. Luego de simular satisfactoriamente el sistema digital comprobando su funcionalidad se usan herramientas específicas del vendedor del FPGA para crear un archivo de configuración del FPGA, el cual describe todas las conexiones, interconexiones y lógica que necesita ser programada dentro del FPGA para poder implementar el sistema digital desarrollado. Entonces, a través de un cable USB se conecta el FPGA o el circuito impreso en cual está soldado el FPGA, a una PC y usando el software de configuración del FPGA se descarga el archivo de configuración. Una vez comprobado el correcto funcionamiento del sistema en el FPGA se graba el archivo de configuración en una memoria no-volátil que el FPGA leerá y usará para auto-configurarse cada vez que se aplica la tensión de alimentación al FPGA o cada vez que se desee re-configurar el FPGA.

Todos los FPGAs, independientemente del fabricante, tienen ciertos elementos en común, tienen un arreglo tipo matricial de elementos lógicos, como flips-flops y lógica combinacional, que se configuran usando cierta tecnología de programación. Los terminales de entrada y salida del FPGA usan celdas especiales de E/S que son diferentes de las celdas de elementos lógicos. Además tienen un esquema de interconexión programable que permite la conexión entre las celdas de elementos lógicos entre sí, y con las celdas de E/S. La programación de las interconexiones y de los elementos lógicos puede o no ser permanente, eso depende de la tecnología de programación usada tal como se verá más adelante. La Figura 1 muestra la composición y disposición de los componentes lógicos en un FPGA de la empresa Xilinx.

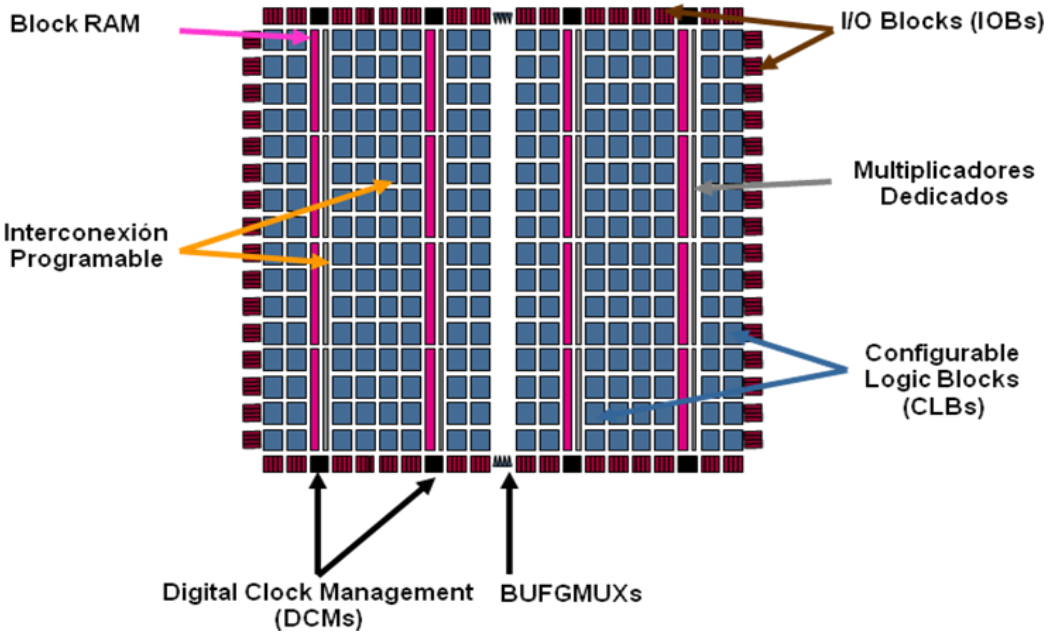


Figura 1 – Estructura y componentes de un FPGA de la empresa Xilinx

La Figura 2 muestra la composición y disposición de los componentes lógicos en un FPGA de la empresa Altera. Tal como se puede observar de las Figuras 1 y 2, los FPGAs de distintos fabricantes tienen elementos comunes, un arreglo matricial, interconexiones y bloques dedicados a funciones específicas como memoria, procesamiento digital de señales, control de reloj, y otras que son particulares de cada fabricante.

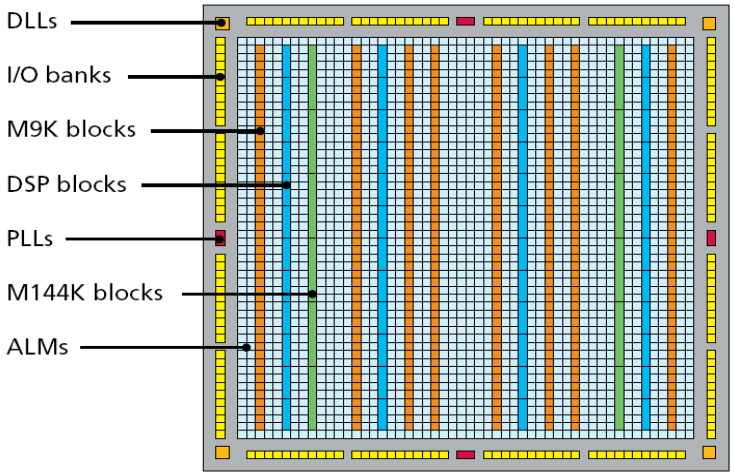


Figura 2 - Estructura y componentes de un FPGA de la empresa Altera

Si bien para la configuración de un FPGA particular se usa un software especifico del fabricante del FPGA, la tendencia actual es tratar de realizar el diseño digital, en un diagrama esquemático o en Lenguaje de Descripción de Hardware (Hardware Description Language, HDL), con la máxima abstracción del FPGA a usar. De modo que si por una razón u otra es necesario cambiar de fabricante de FPGA, sea posible una transición lo más fácil posible. Esto se le llama *diseño transportable*.

2. Principales Características de un FPGA

Los FPGAs son dispositivos orientados a satisfacer una muy amplia gama de aplicaciones, desde simple lógica combinacional hasta sistemas con microprocesador embebido, transmisión tipo Ethernet, transmisión de datos series a 3.5Gb/s, todo con el mismo dispositivo. Por ello los FPGAs tienen características diversas, pero se podría decir que las principales son las siguientes:

- Gran cantidad de terminales de E/S. Desde 100 hasta unos 1400 terminales de E/S
- Buffers de E/S programables: control de sesgo, control de corriente, configuración del estándar de E/S , pull-up y pull-down configurables
- Gran cantidad de Flips-Flops, los dispositivos mas grandes tienen unos 40.000 FFs
- Gran cantidad de Tablas de Búsqueda (Look-Up Tables), ~100.000
- Bloques de Memoria (BRAM) de doble puerto, puerto simple, de hasta 18Mbits, configurables como RAM, ROM, FIFO y otras configuraciones
- Bloques dedicados de Multiplicación
- Transceptores para transmisión serie de muy alta velocidad , entre 1.5 a- 10.0Gb/s
- Procesador en hardware embebido, tal como el Power-PC, ARM9
- Procesadores descriptos en software, HDL, tales como el 8051, ARM3
- Controladores de reloj tipo Delay Lock Loop (DLL) y Phase Lock Loop (PLLs) de hasta 550MHz. De 2 a 8 controladores por dispositivo
- Control de impedancia programable por cada terminal de E/S
- Interface DDR/DDR2 SDRAM soportando interfaces de hasta 800 Mb/s
- Interfaz con estándares de E/S tipo diferencial tales como LVDS, SSTL diferencial, etc.

3. Celda Básica de Configuración del FPGA

El elemento básico de un FPGA desde el punto de visto no-lógico, es decir que no tiene una función digital lógica, es la celda de configuración. Esta celda es la que va determinar la configuración de *cada* elemento lógico, por ejemplo si un flip-flop se va usar o no, y en caso de usarlo, si se configura como D o T. La celda de configuración también determina la configuración de los elementos de ruteo y de las interconexiones, tal como se verá más adelante.

Existen en la actualidad cuatro tipos de celdas de configuración de un FPGA. Una está basada en una *celda SRAM*, que como su nombre indica se usa una pequeña celda SRAM para mantener la configuración de cada parte configurable del FPGA. Otro tipo de celda involucra una celda llamada *anti-fuse* (anti-fusible), que consiste en una estructura microscópica la que, a diferencia de un fusible regular, esta normalmente abierta. Cuando se hace circular una cierta cantidad de corriente durante la configuración del dispositivo, , cerca de los 5mA, causa una gran potencia de disipación en un área muy pequeña, lo que provoca el derretimiento de un aislante dieléctrico entre dos electrodos formando una unión permanente muy fina. El tercer tipo de celda de configuración se basa en celdas tipo *Flash*, que a diferencia de las SRAM, permiten mantener la configuración aún después de desconectada la alimentación del dispositivo. Hay también FPGAs que tienen en el mismo dispositivo celdas *Flash* y celdas *SRAM*. Las celdas Flash se usan para mantener los datos de configuración del FPGA y las SRAM para la configuración lógica del FPGA.

3.1. FPGAs basados en celdas SRAM

La celda de configuración tipo *SRAM,* que como su nombre lo indica, es una pequeña celda SRAM que se usa para mantener la configuración de cada parte configurable del FPGA. La gran ventaja de los FPGA basados en celdas SRAM es que utilizan un proceso de fabricación estándar. La 'fabrica' de FPGA-SRAM tiene un procedimiento de fabricación muy conocido, que es el usado en la fabricación de memorias SRAM, lo que debido a la enorme cantidad de memorias SRAM que se producen para el mercado digital, permite lograr costos de producción muy bajos, muy alta performance y trabajar con un proceso de fabricación muy amortizado y de gran rendimiento (yield).

Como es sabido las celdas de memoria tipo SRAM pueden ser reprogramadas un sinnúmero de veces, del mismo modo un FPGA basado en celdas de configuración SRAM puede ser reprogramado un infinito número de veces, aún cuando el FPGAs ya esté montado y soldado en un circuito impreso (PCB). Esta reprogramación en PCB se denomina *Programable En Circuito* (In-Circuit Programmable, ISP). Esta tecnología SRAM es muy útil también para llevar a cabo una actualización rápida del sistema digital dentro del FPGA. Por ejemplo, algunos sintonizadores de televisión satelital usan FPGAs en su sistema para sus diversas funciones. Cuando por alguno motivo se necesita actualizar el sistema dentro del FPGA, se envía por satélite un nuevo archivo de configuración. La lógica del sintonizador, que interpreta el comando de actualización, procede a actualizar el archivod de configuración del FPGA (auto-actualización), evitando de este modo tener que cambiar el sintonizador.

La gran desventaja de las celdas SRAM es que son volátiles, lo que significa que aún un simple pulso espúreo (glitch) en la tensión de alimentación borraría la configuración del FPGA, quedando prácticamente sin ninguna funcionalidad hasta que se lo configure de nuevo. Otra desventaja es que, debido a que la selección del camino de conexión entre los diferentes bloques lógicos (llamado ruteo), se basa en celdas SRAM, se provocan grandes retardos de ruteo, lo que es un problema en diseños que requieren un rendimiento muy alto. Una desventaja más de estos dispositivos es que, en el producto final, una vez que el FPGA está en el PCB y listo para ser comercializado, necesita de una memoria externa pequeña (tipo Flash, serie o paralela) que mantiene el archivo de configuración del FPGA. El FPGA tiene una pequeña MEF que, cuando se le da tensión de alimentación, le indica al FPGA que tiene que ir a buscar la configuración del mismo a una memoria externa. Luego lee la información de la memoria y configura las celdas SRAM. Para algunas aplicaciones, como sistemas médicos de emergencia, este tiempo de lectura de la memoria y configuración es muy largo (50 ~ 500ms). Para otras aplicaciones, como interfaces series, este tiempo pasa desapercibido.

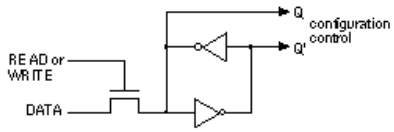


Figura 3 - Celda básica SRAM de configuración de los FPGAs de Xilinx

La Figura 3 muestra un ejemplo de la tecnología SRAM de configuración de los FPGAs. Esta celda básica está construida por dos inversores cruzados. La salida de la celda de configuración es

conectada al transistor de paso, para conectarlo o desconectarlo. La celda se programa usando las líneas Write y Data.

3.2. FPGAs basados en celdas Anti-Fuse

Otro tipo de FPGA usa una celda llamada anti-fuse (anti-fusible) como celda básica de configuración. Esta celda consiste en una estructura microscópica que, a diferencia de un fusible regular, está normalmente abierta. Está compuesta de tres capas, las conductoras arriba y abajo, y la aisladora en el medio (Metal-Insulator-Metal, MIM). Para configurar el dispositivo se hace circular una cierta cantidad de corriente (~5mA), lo que causa una gran potencia de disipación en un área muy pequeña, provocando el derretimiento del aislante dieléctrico (tipo Ooxide-Nitride-Oxide, ONO) entre los dos electrodos conductores (SiO_2, Dióxido de Silicio) formando una unión permanente muy fina de unos 20nm. La estructura anti-fuse fue creada y se usa habitualmente en ciertas familias de los FPGAs de la empresa ACTEL. Esta estructura es conocida como Programmable Logic Interconnect Circuit Element, PLICE.

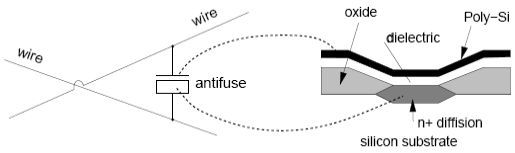


Figura 4 - Estructura del anti-fuse de Actel

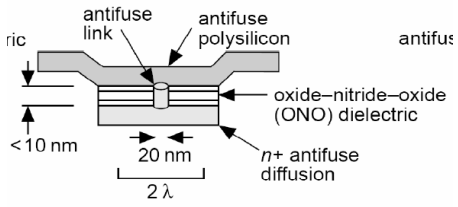


Figura 5 - Vista microscópica del anti-fuse de Actel

La principal ventaja de los FPGA-Anti-Fuse es que no son volátiles, lo que para algunas aplicaciones es sumamente crítico, por ejemplo aplicaciones espaciales y aplicaciones médicas. También, debido a esta tecnología, los retardos de conexión entra los bloques lógicos son muy reducidos, por lo que el rendimiento de estos dispositivos es bastante elevado.

Como desventajas se tienen: primero, que requieren un proceso de fabricación específico, bastante complejo, lo que lleva a que el costo de los mismos sea bastante elevado comparado con los FPGA-SRAM (como mínimo 200 veces más caros). También requieren un programador especial para poder programar el anti-fusible, y la mayor desventaja es que una vez que se han configurado con cierta lógica, ésta no se puede cambiar, lo que es conocido técnicamente como One Time Programmable, OTP . El hecho de que estos FPGAs sean OTP crea un proceso de verificación muy

meticuloso de la lógica a ser programada, a fin de no tener que descartar este dispositivo tan caro por errores de diseño. Este proceso de verificación tan largo, implica también incrementar los costos de desarrollo del diseño, al tener que disponer de más horas-ingeniero para tener listo y sin problemas el producto final. Una de las principales aplicaciones de este tipo de FPGA es para sistemas espaciales, ya que estos FPGAs son tolerantes a las radiaciones de partículas de alta energía (los bits de configuración no pueden cambiar si son golpeados por una partícula).

Como para tener una idea, un FPGA anti-fusible de término medio cuesta cerca de U$S2.000, mientras que uno de similares características tipo SRAM puede rondar los U$S 100. La diferencia de costos es abrumadora, claro que sí, está en juego la estabilidad, seguridad y confiabilidad de un satélite que puesto en órbita cuesta cerca de U$S270.000.000. No hay muchas opciones para elegir la tecnología a usar. (Referencias: http://www.ieco.clarin.com/notas/2008/09/28/01768728.html; http://www.en.argentina.ar/_en/science-and-education/C1477-argentina-on-space.php)

3.3. FPGAs basados en celdas FLASH

Los FPGA-Flash, tienen como ventaja lo mejor del FPGA-SRAM y del FGPA-Anti-Fuse, son reprogramables y no son volátiles. Sin embargo todavía son caros, ya que usan una tecnología más cara que la SRAM, y las celdas FLASH se usan no solo para guardar la configuración en si del FPGA, sino que también para todo lo que es ruteo, lo que hace que la cantidad de celdas FLASH por FPGA sea un gran número Los procesos de fabricación de celdas FLASH recién ahora son más comunes. Actualmente en el mercado están apareciendo más opciones de estos dispositivos, sobre todo en los tamaños de FPGAs medianos-chicos, pero como la competencia es muy grande y a veces centavos marcan la diferencia, la demanda todavía no es considerable. Otra desventaja para los FPGAsFLASH es que el proceso de reconfiguración toma varios segundos.

La Figura 6 detalla la estructura de una celda FLASH de la empresa ACTEL, que le llama FLASH Switch. Usa dos transistores que comparten la compuerta flotante, la que almacena la información de configuración. Uno es el transistor de sensado, el cual sólo se usa para escribir y verificar la tensión de compuerta flotante. El otro es el transistor de conexión (switching). Esta celda puede ser usada en el FPGA para conectar/separar rutas de conexiones o para configurar la lógica.

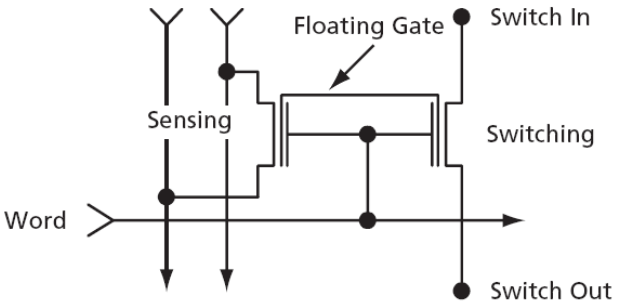


Figura 6 - ACTEL Flash switch

3.4. FPGAs basados en celdas Flash y SRAM

Finalmente hay algunos FPGAs que tienen celdas Flash y SRAM en el mismo dispositivo. Las celdas Flash se utilizan para guardar los datos de configuración del FPGA, mientras que las celdas SRAM para la configuración de la lógica del FPGA. Cuando se da tensión de alimentación, las celdas SRAM se configuran en forma casi instantánea desde la Flash, resultando una configuración del FPGA en menos de 1ms, a diferencia de un FPGA-SRAM cuyo tiempo de configuración típicamente va de los 50 hasta los 500 ms (dependiendo del tamaño del dispositivo). Esta disponibilidad casi instantánea del FPGA lo hace muy útil para aplicaciones críticas en tiempo. Estos FPGAs también permiten configurar solo las celdas SRAM, por ejemplo, durante el proceso de construcción del prototipo, sin tener que programar la Flash.

Una gran ventaja, y que a veces es decisiva para el diseñador al elegir entre este tipo de FPGA o las FPGA-FLASH, es que, al no tener que acceder a un chip de memoria Flash externo, no hay una conexión física entre el FPGA y la memoria Flash que permita que la configuración del FPGA pueda ser expuesta, y de este modo vulnerada, para una posterior re-ingeniería (o ingeniería inversa) sobre el producto final. Se han descubierto muchos casos de productos copiados a través de la lectura de los datos de configuración (bitstream) disponibles en las rutas del circuito impreso que hay entre el FPGA y la memoria FLASH (éste problema también está presente en las FPGAs-SRAM).

Por supuesto que estos dispositivos son un poco más caros que los FPGA-SRAM, pero más baratos que los FPGA-FLASH, ya que en este caso las celdas FLASH *solo se usan* para guardar la configuración, por lo que la cantidad de estas celdas es mucho menor que en el caso de FPGA-FLASH.

3.5. Ejemplos y Comparación

Ejemplos de FPGA-SRAM:

- Xilinx Spartan/Virtex
- Altera Cyclon/Stratix
- Lattice EP/SC

Ejemplos de FPGA-Flash

- Actel Fusion/Igloo (ultra low power)

Ejemplos de FPGA-FLASH-SRAM

- Lattice XP family
- Xilinx Spartan 3AN

Ejemplos de FPGA-Anti-Fuse

- Actel Sx/Mx
- QuicLogic pASIC

La Tabla 1 presenta una comparación entre las distintas tecnologías y las principales características de un FPGA.

Tabla 1 - Comparación de Tecnologías de las FPGAs

	SRAM	Anti_fuse	Flash
Velocidad	Más lenta	Mejor	Más lenta
Potencia	Varía/Peor	Segunda Mejor	Mejor
Densidad	Mejor	Segunda Mejor	Media
Tolerancia a la Radiación	Peor	Mejor	Media
Tamaño celda ruteo	1	1/10	1/7
Memoria Externa	Si	No	No
Reprogramable	Si	No	Si

3.6. Transición de FPGA a FPGA-ASIC

Finalmente cabe mencionar que empresas como Xilinx, y Altera ofrecen una conversión del FPGA que ya está listo para producción comercial a un dispositivo llamado FPGA-ASIC. Altera le llama HardCopy ASIC, mientras Xilinx le llama EasyPath. Para grandes volúmenes de producción (mayores a 500.000 unidades por año) ésta es una idea atractiva que baja mucho los costos e incrementa la seguridad del diseño (no external bitstream). La arquitectura de este FPGA-ASIC es similar a la arquitectura del FPGA, pero las celdas básicas programables, tanto las de ruteo como las de lógica, ya no son más celdas SRAM o FLASH, ni son programables, son conexiones fijas como en un ASIC. Pero a diferencia de un ASIC, la lógica del sistema que se ha diseñado ya ha sido verificada en hardware, en el FPGA, por lo que el riesgo de tener que re-hacer al ASIC es casi nulo. La parte crítica de esta conversión está referida a los retardos lógicos y de ruteo (delays). Durante la fase de prueba que se lleva a cabo en el FPGA se trabaja con las celdas SRAM, que como ya se sabe agregan bastante retardo, pero al pasar al FPGA-ASIC no hay celdas SRAM que retarden, por lo que el rendimiento del dispositivo cambia, aumentando mucho. Hay nuevas herramientas de software que hacen los cálculos de tiempo de una manera más realista, haciendo que el resultado final sea más previsible en lo que respecta a retardos de tiempo.

Otra ventaja que hace atractiva esta conversión es la reducción de potencia. FPGAs son dispositivos que en general consumen mucha potencia, por lo que para aplicaciones que se alimentan con baterías no son una buena alternativa. Con la conversión a FPGA-ASIC el consumo de potencia se reduce considerablemente. Sin embargo, se debe mencionar que últimamente los fabricantes están ofreciendo FPGAs de muy bajo consumo, tal como la familia IGLOO de Actel, cuyo mercado es el de los equipos portátiles.

4. Arquitectura de los FPGAs

La arquitectura de un FPGA varía de un fabricante a otro. En realidad es una discusión permanente, en la que cada fabricante se atribuye que tiene la mejor arquitectura. Se ejecutan cientos de estudios comparativos (benchmarks) para mostrar cuán buena es la arquitectura ofrecida por cada fabricante.

Independientemente del fabricante elegido para el diseño, ciertamente el más beneficiado en esta competencia es el usuario final de los FPGAs. En la carrera por diferenciarse como el mejor, cada fabricante incluye tecnología de punta en sus dispositivos, aumenta el rendimiento, baja los costos, ofrece bloques enteros de propiedad intelectual (Intellectual Property, IP) como así también bloques en silicio dentro del FPGA para aplicaciones complejas especificas, tales como procesador(es) PowerPC, Gigabit Transceivers, PCIe, controlador Ethernet, etc.

A pesar de tratar de diferenciarse uno del otro, en realidad los FPGAs de los diferentes fabricantes tienen muchos componentes en común, tales como bloques lógicos programables, bloques de memoria de doble puerto (dual port), bloques para ejecución de MACs (Multiplicador–Acumulador o bloques DSPs), bloques de control de reloj (generación de frecuencias a partir de una frecuencia base, corrimiento de fase), bloques de entrada/salida, etc. En este trabajo se presentará y se verá en detalle la arquitectura de uno de los fabricantes, los FPGAs de Xilinx, pero debido a la similitud con otros FPGAs, se sentarán las bases para comprender la arquitectura de cualquier otro FPGA.

La empresa Xilinx ofrece al mercado dos tipos de FPGAs: una de bajo costo, performance media y otra de alto costo, performance alta-muy alta. La primera se denomina la familia de los FPGA Spartan, y la segunda los FPGA Virtex. A su vez, y debido a la gran competitividad del mercado, estas familias se van renovando cada dos o tres años. Tal es así que, por ejemplo, la familia Spartan fué evolucionando como Spartan, Spartan 2, Spartan 3 y la recientemente lanzada Spartan 6. Del mismo modo para la Virtex, comenzando con Virtex, Virtex-E, Virtex 2, Virtex 2Pro, Virtex 4, Virtex 5 y la reciente Virtex 6.

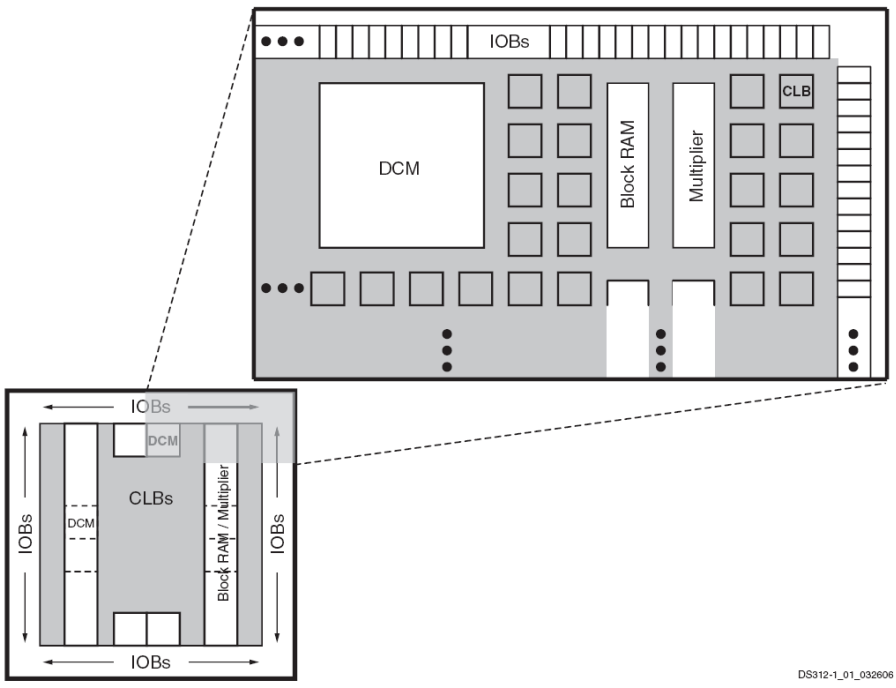


Figura 7 – Arquitectura de un FPGA Spartan 3

Por ejemplo, la Figura 7 muestra cómo están organizados los distintos componentes en un FPGA Spartan 3. Se puede observar el anillo de bloques de E/S (Input/Output Blocks, IOBs) que rodea el arreglo matricial de los bloques lógicos configurables (Configurable Logic Blocks, CLBs). Los

bloques de memoria RAM en realidad constan de varios bloques de memoria RAM de 18K bits. Cada bloque de memoria RAM está asociado a un multiplicador dedicado. Los controladores de reloj (Digital Clock Manager, DCM) están distribuidos de manera que hay dos en la parte superior, dos en la inferior y uno en cada uno de los costados del FPGA. En la Figura 7 no se muestran la gran cantidad de recursos destinados a la interconexión de los diferentes componentes del FPGA. A continuación se verán en detalle los diferentes componentes mencionados, que son parte de los FPGAs.

4.1. Bloque Lógico Configurable (CLB)

Conocidos comúnmente como Bloque Lógico Configurable, (Configurable Logic Block, CLB), estos bloques son la parte lógica mayoritaria dentro de un FPGA. Su estructura macroscópica se detalla en la Figura 8, donde se puede ver que un CLB está constituido por 4 rebanadas (Slices), SLICEM S0, SLICEM S1, SLICEL S2 y SLICEL S3, rutas de conexión para el acarreo matemático, Cin y Cout (carry in, carry out), conexiones a la matriz de conexiones (Switch Matrix) que proveen acceso a las rutas de conexiones generales y conexiones locales entre las rebanadas del mismo CLB y CLBs vecinos.

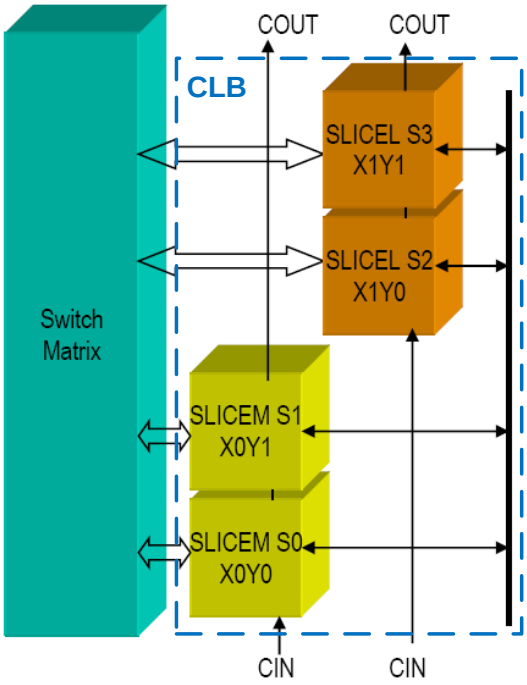


Figura 8 - Esquema de un CLB y su conexión a la matriz de conexión

Tal como se aprecia en Figura 8 los SLICES se individualizan en coordenadas, desde al más abajo a la izquierda como SLICE X0Y0, hasta el superior derecho identificado como SLICE X1Y1. Del mismo modo para el CLB. Cada CLB se identifica con coordinadas XY dentro del arreglo matricial. Donde XY igual a 00 identifica el primer CLB en la esquina inferior izquierda de la matriz de CLBs, la coordenada del CLB suprior derecho dependerá del tamaño de la matriz, es decir del tamaño del FPGA.

4.1.1. Rebanadas (SLICE) de un CLB

Hay dos tipos de rebanadas (SLICEs) dentro de un CLB:

- SLICEM: además de las funciones lógicas que se pueden implementar en el SLICE, ofrece opciones para implementar pequeñas memorias.
- SLICEL: solo puede implementar funciones lógicas.

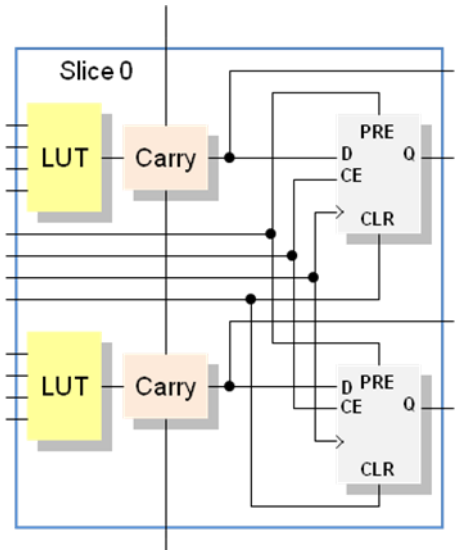


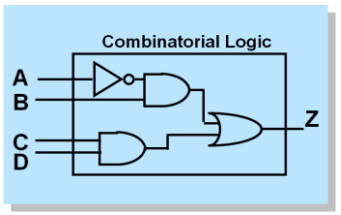
Figura 9 - Vista simplificada de un SLICEL de un FPGA Spartan/ FPGAVirtex

La Figura 9 muestra una vista simplificada de un SLICEL, en la que se destaca lo siguiente:

- Dos tablas de búsquedas (Look-Up Tables)
- Dos flip-flops
- Cuatro salidas, dos combinacionales y dos con registros
- Tiene entradas de control para los flip-flops
- Entradas para las LUTs
- Entrada y salida para la cadena de acarreo (Carry Chain)

4.1.2. Tablas de Búsqueda - Look-up Tables (LUTs)

En un FPGA toda la lógica combinacional se implementa utilizando tablas de búsqueda, LUT, es decir la función lógica se almacena en una tabla de verdad de 16x1 (para las LUTs de 4 entradas). En cierta literatura a las LUTs también se les llama "Generadores de Funciones" (Function Generators). La Figura 10 detalla la similitud entre una tabla de verdad y una LUT. La columna de valores Z, valores de la función combinacional, son los valores que realmente se almacenan en la LUT de 16x1. Vale recordar que, a menos que por alguna razón se quiera hacerlo manualmente, el almacenamiento de los valores en las LUTs lo realiza el Software del fabricante del FPGA, siendo el proceso totalmente transparente al diseñador del sistema digital.



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

Figura 10 - Cómo se implementa una función combinacional en una LUT.

Una característica que a veces es de mucha utilidad, sobre todo para sistemas de muy alta frecuencia, es que el retardo a través de la LUT es constante e independiente de la función implementada.

Para implementar funciones lógicas de más de cuatro entradas en una LUT, se usan multiplexores dedicados que están distribuidos en el SLICE(L/M) y en el CLB para poder implementar cualquier función con un mayor número de entradas. Por ejemplo, para implementar cualquier función lógica de cinco entradas se usa el multiplexer F5Mux, que multiplexa las salidas de las dos LUTs (4 entradas) y la quinta entrada funciona como señal de selección del F5Mux. Si bien estos multiplexores dedicados podrían ser implementados en las LUTs, al ser dedicados son más eficientes y dejan las LUTs disponibles para otras funciones. Estos multiplexores no se pueden ver en la Figura 9, al ser una vista simplificada de un SLICE. Pero sí se pueden ver en la Figura 11, donde se destacan los siguientes multiplexores:

- F5Mux: multiplexa las salidas de las LUTs dentro del SLICE
- F6Mux: multpilexa las salidas de los F5Mux de un SLICE
- F7Mux: multiplexa las salidas de los F6Mux de un CLB
- F8Mux: multiplexa las salidas de los F7Mux de dos CLBs

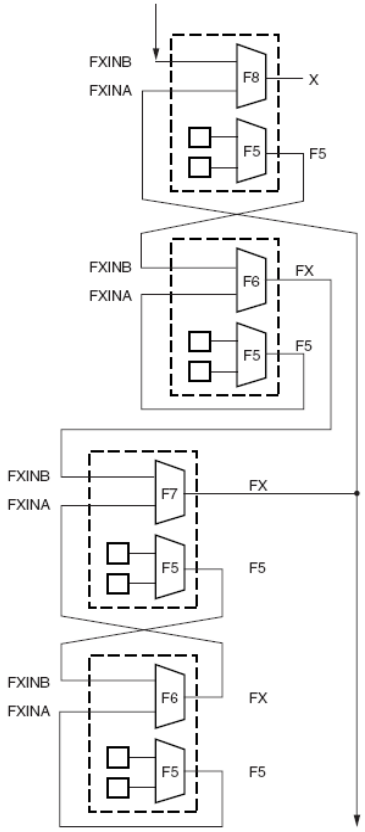


Figura 11 - Multiplexores dedicados y sus conexiones dentro de un CLB.

Como se puede ver en la Figura 11, cada SLICE tiene un F5Mux, y un segundo multiplexor llamado genéricamente FiMux. Esto se debe a que este multiplexor puede funcionar como F6Mux, F7Mux o F8Mux dependiendo de su ubicación y su conexión con los otros multiplexores.

Se destaca que las conexiones entre los multiplexores se hacen a través de rutas de conexiones dedicadas a tal fin, con retardo cero. De todos modos, las salidas de estos multiplexores también están disponibles en la salida combinacional del CLB para acceder a las conexiones de ruteo general. Nota: en la última versión de Spartan y Virtex, versión 6, las LUTs son de 6 entradas. Esto fue implementado después de trabajos de investigación llevados a cabo por Xilinx tratando de encontrar cual era el número óptimo de entradas para las LUTs. De todos modos, todo lo explicado en este punto para LUTs de 4 entradas, se aplica también para LUTs de 6 entradas.

4.1.3. Elementos de Almacenamiento. Flip-Flops

La Figura 12 detalla las posibles configuraciones del elemento de almacenamiento y sus respectivas señales de control. Cada SLICE posee dos elementos de almacenamiento programables que pueden funcionar como flip-flop D, o cerrojo (latch) transparente. El elemento de almacenamiento ubicado en la parte superior del SLICE se denomina FFY, mientras que el de la parte inferior se denomina FFX. Ambos elementos de almacenamiento tienen un multiplexor de selección para la entrada D, pudiendo seleccionar entre la salida de la respectiva LUT, DY o DX, o una entrada externa al SLICE, llamada BY para el elemento FFY y BX para el FFX.

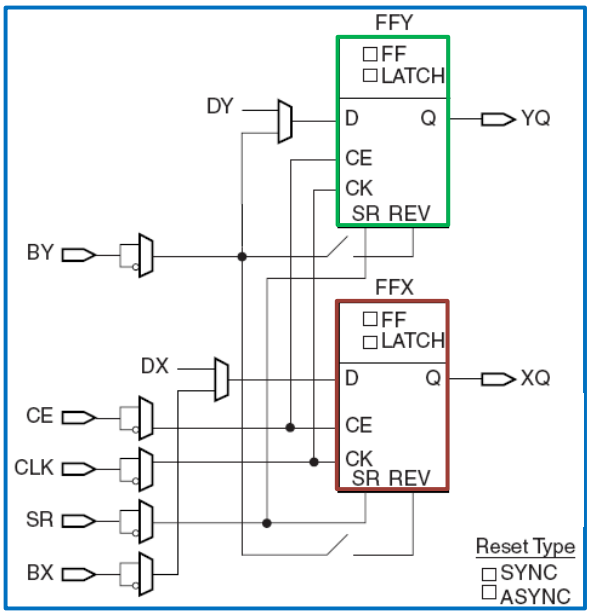


Figura 12 – Detalle de la lógica del Registro/Cerrojo en un SLICE

Se observa también que las señales de reloj, habilitación del reloj y reset (limpiar) son comunes al par de flip-flops en cada SLICE. El flip-flop tiene construido en su lógica interna la opción de poder poner en cero (reset) el flip-flop de manera sincrónica o asincrónica. La entrada REV (reverse) se usa cuando se desea invertir el valor lógico de activación de la señal de reset.

4.1.4. Lógica de Acarreo (Carry)

El CLB tiene lógica dedicada exclusivamente para el acarreo de la suma aritmética con el objeto de mejorar el rendimiento de sumadores, contadores, comparadores y funciones lógicas relacionadas.

En la Figura 13 se detalla la lógica, descrita en compuertas, de cada bit de un sumador total. Mientras que la Figura 14 muestra la analogía entre la lógica del sumador total y la lógica disponible en un SLICE para implementar el sumador total de una manera más eficiente que si fuera implementado solo con LUTs.

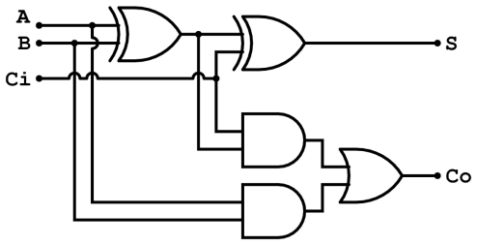


Figura 13 - Bloque lógico básico de un sumador total

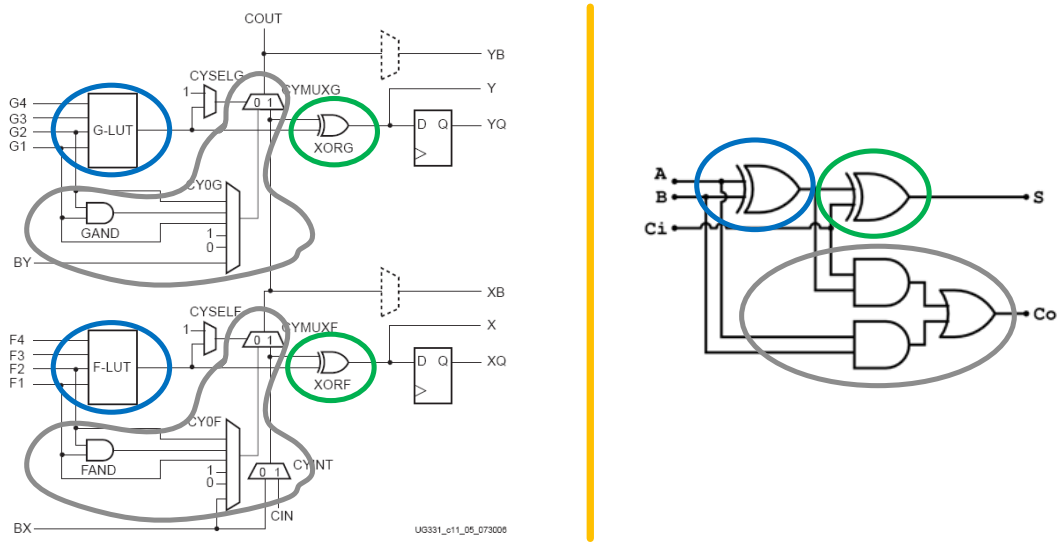


Figura 14 - Relación entre la lógica de acarreo en un SLICE y el sumador total

La lógica de acarreo disponible en los CLB está constituida por los siguientes elementos, detallados en la Figura 14:

- Compuertas lógicas dedicadas sólo para el acarreo
- Multiplexores dedicados
- Ruteo y conexiones dedicadas

4.1.5. Componentes del SLICE

Después de haber visto en detalle los distintos elementos componentes de un SLICE, es hora de ver como se utilizan e interconectan dentro del mismo. La Figura 15 muestra una vista generalizada de los componentes sin sus interconexiones.

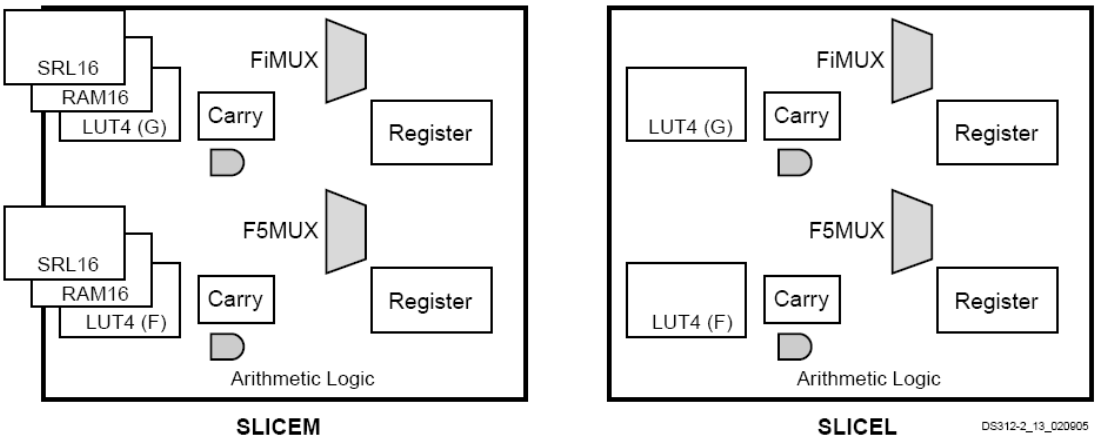


Figura 15 - Componentes de un SLICEM y SLICEL, sin sus interconexiones

Como ya de detalló en 4.1.1, los FPGA de Xilinx tienen los SLICELs y los SLICEMs. Tal como se ve en la parte izquierda de la Figura 15, el SLICEM permite implementar bloques de memoria, registros de

desplazamiento y lógica combinacional en una especie de LUT-multifunción. En los dispositivos actuales, aproximadamente un 50% del total de los CLBs de un FPGA contienen SLICEM y el otro 50% SLICEL. En versiones anteriores de FPGAs, el 100% de los CLBs eran SLICEM, pero esto fue cambiando hasta llegar a la relación actual de 50-50. Esto se debe principalmente a dos motivos: primero, si bien el SLICEM es más genérico, físicamente ocupa más lugar que el SLICEL y por ende, hace el dispositivo más costoso. Segundo, se llevaron a cabo intensos estudios del uso del CLB, y rara vez se usaba más del 50% de ellos como SLICEM.

La Figura 16 muestra todos los bloques vistos anteriormente por separado (excepto las funciones unidas por líneas de puntos que no se han descripto todavía), ahora juntos en un SLICEM. Se procederá a describir la mitad inferior del SLICE (que es exactamente igual a la mitad superior).

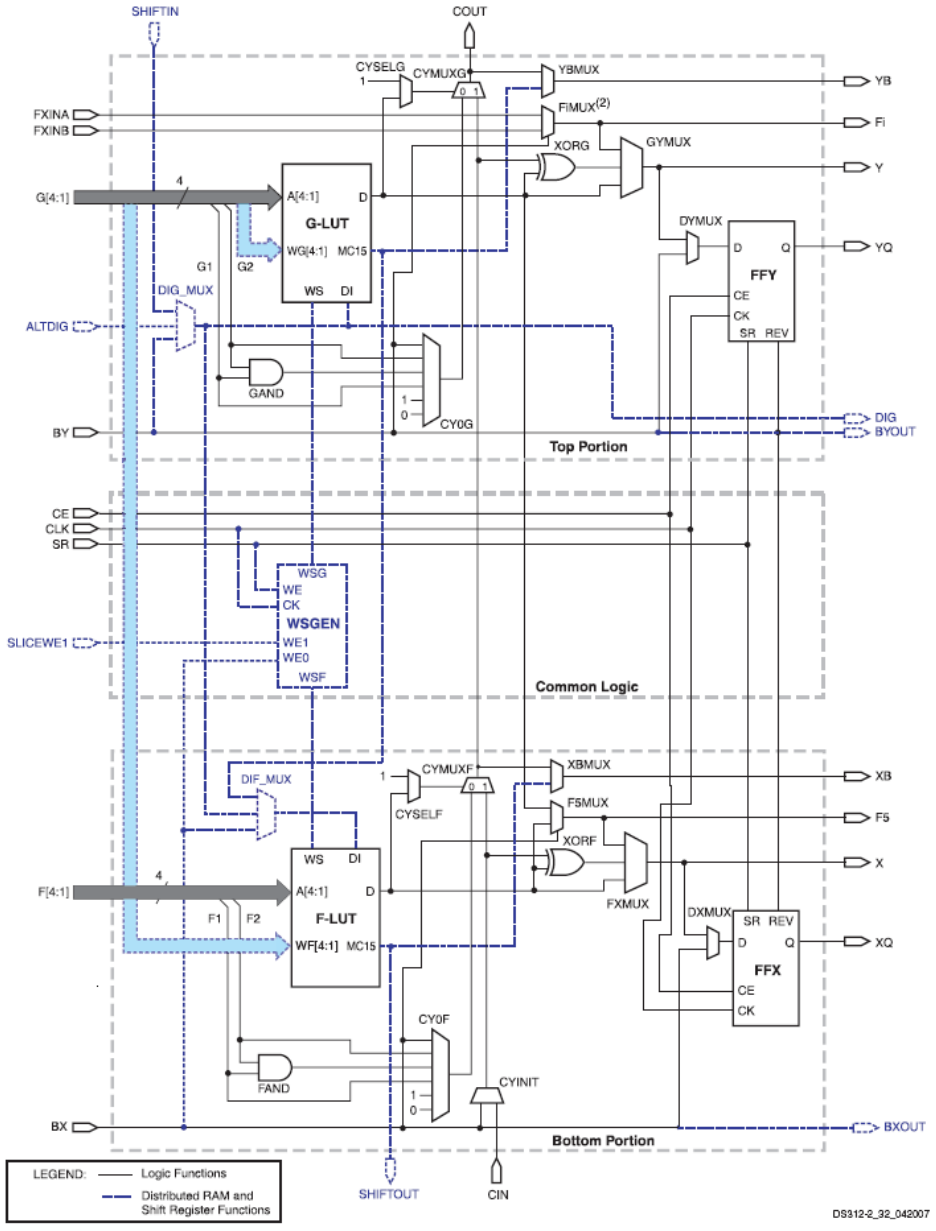


Figura 16 - Detalle de un SLICEM

Cuatro líneas de señal de entradas, F(4:1), entran directamente a la LUT o Generador de Funciones F. Tal como se detalló antes, en la LUT o GF se genera la lógica combinacional necesaria. La salida de la LUT o GF, llamada D, tiene cinco caminos posibles:

- Salir en forma directa o negada (XORF) por la salida X, pasando por el multiplexor FXMUX.
- Entrar por la entrada de datos D al elemento de almacenamiento FFX, cuya salida es XQ.
- Controlar el mutliplexor CYMUXF de la cadena de acarreo.
- Entrada de datos al multiplexor F5Mux para implementar funciones combinacionales de más de 4 entradas.

Otras entradas al SLICE son BY y BX, conocidas como Bypass Y y Bypass X. Estas pueden tener una de las siguientes funciones:

- Eludir (bypass) la LUT y entrar a la entrada D del elemento de almacenamiento. De este modo, en una mitad de un SLICE se puede tener al mismo tiempo una función combinacional y otra secuencial.
- Controlar el multiplexor F5Mux.
- Entrar a la cadena de acarreo.
- BY (solamente) controla la entrada REV de FFY y FFX (ver la Figura 12).

Se debe resaltar el hecho de que los dos elementos de almacenamiento del SLICE son controlados por el mismo reloj y las mismas señales de habilitación de reloj y de reset. También que la salida de una mitad del SLICE, por ejemplo XQ, puede entrar en la otra mitad del SLICE, por la entrada BY por ejemplo, utilizando rutas de conexión local. Esto es muy usado para sincronizar una señal asincrónica utilizando doble flip-flop, con un mínimo retardo entre ellos.

4.2. Bloque de Entrada/Salida (Input/Output Block, IOB)

Para poder recibir y transmitir señales digitales, los FPGAs disponen de un bloque de E/S bastante elaborado que posibilita usarlos con muy diversos rangos de tensiones, frecuencias de trabajo, estándares de señales digitales, etc, lo que los hace muy adaptables a las necesidades del sistema del que forman parte.

Existe un bloque E/S por cada terminal de E/S del FPGA. Así cada terminal puede ser configurado como entrada, como salida o bidireccional. En cada bloque E/S existe un buffer que tiene diversas funciones configurables por el diseñador que permiten adaptar el FPGA en un sistema complejo trabajando con diferentes tensiones y corrientes, en un circuito impreso con muchos problemas de integridad de señal (signal integrity).

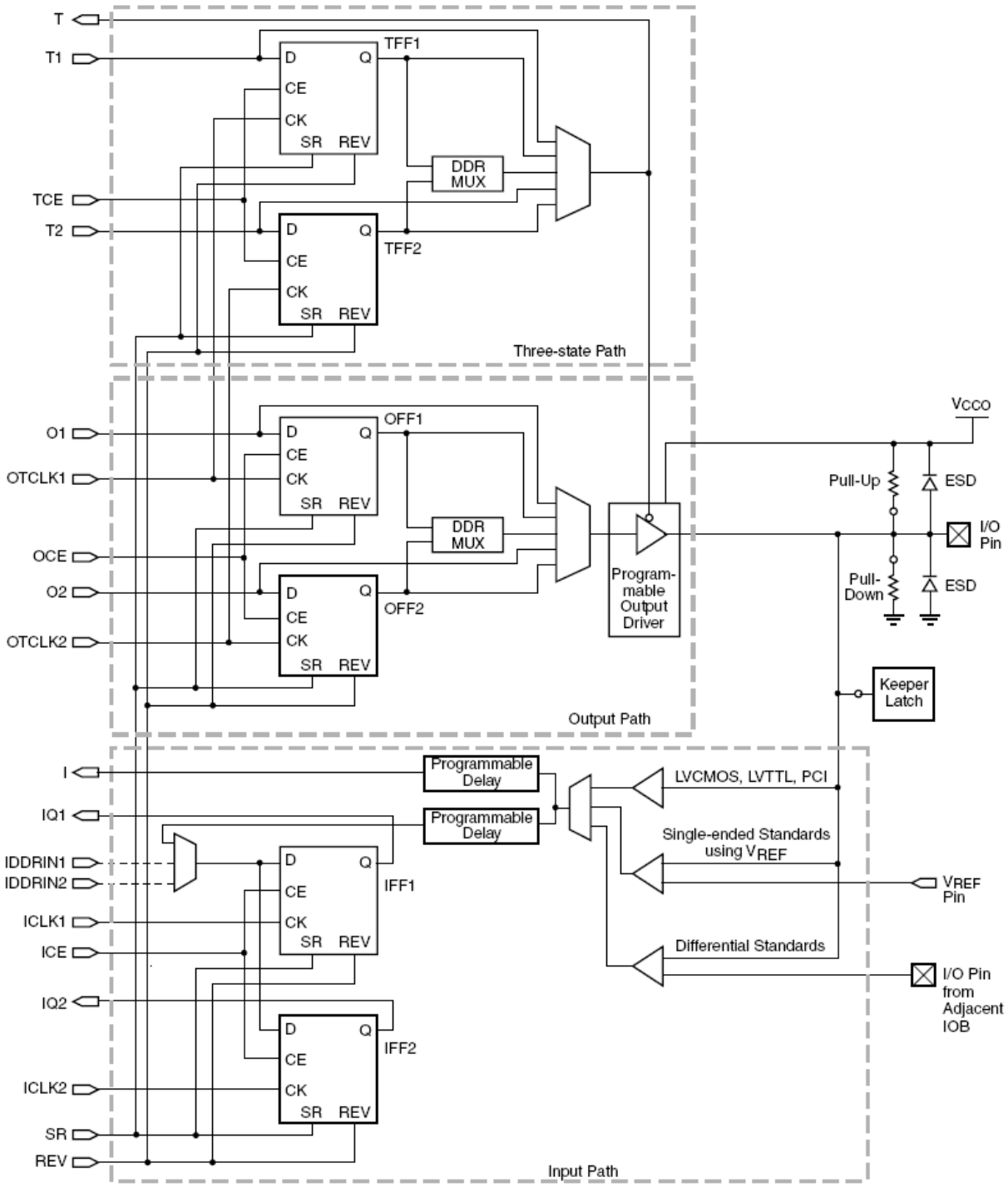


Figura 17 - IOB de un Xilinx Spartan FPGA

En la Figura 17 se observa un diagrama de la estructura interna de un IOB. Se puede observar que hay tres caminos posibles para una señal:

- **Camino de entrada**, que conecta el IO Pad (terminal del circuito integrado) a la lógica interna del FPGA. Esta conexión puede ser:
  - Directa, a través de un elemento de retardo (opcional).
  - Con registro usando el flip-flop IFF1
  - Con registro usando el flip-flop IFF2

Las tres salidas del IOB, I, IQ1 e IQ2, se conectan al ruteo general interno del FPGA. El par de flip-flops IFF1 e IFF2 se usan para interfaces tipo DDR (dual data rate). De modo que el dato de entrada sea capturado en ambos flancos con distintos flip-flops. El flip-flop IFF1 suele ser usado como registro de entrada para reducir al mínimo el tiempo de establecimiento de la señal de entrada. Se observa también que cada entrada tiene un par equivalente (IO Pin from adjacent IOB) para el caso que se utilice una configuración de entrada diferencial, tales como LVDS.

- **Camino de salida**, comunica la lógica interna con el IO Pad del FPGA. Entra al IOB por medio de O1 u O2, pasa a través de un multiplexor y del buffer de salida llegando al IO Pad. El par de flip-flops OFF1 y OFF2 se usan para transmitir datos en ambos flancos del reloj (DDR). También estos flip-flops se usan para poner registros en las señales de salida del FPGA, de modo de reducir al mínimo el tiempo de retardo de flip-flop (Clock to Output, Tco). Además, si se los IOB flip-flops en todas las señales de salida de un sistema, se reducen los diferentes tiempos de propagación entre las distintas salidas, generando una transición de las salidas "casi" al mismo tiempo.
- **Camino de alta impedancia**, comunica la señal de control de alta impedancia con el buffer de salida. Esta ruta también dispone de un par de flip-flops para interfaces tipo DDR o para registrar en este bloque la señal de control de alta impedancia. Se observa también una realimentación de la señal que controla la alta impedancia del buffer de salida.

4.2.1. Buffer de E/S

El buffer de Entrada/Salida del IOB es un buffer configurable de acuerdo al estándar de la Entrada/Salida que se desea. La Tabla 2 detalla los diferentes estándares de E/S disponibles en un FPGA, con sus respectivas tensiones de entrada y de salida, como así también, si se necesita alguna tensión de referencia o de terminación.

En el caso del buffer de entrada, se puede configurar para trabajar con cualquiera de los estándares de E/S detallados en la Tabla 2. En algunos de estos estándares el buffer de entrada utiliza una tensión de referencia o disparo (threshold), V_{REF}, que se suministra por un terminal determinado del FPGA. V_{REF} normalmente se usa para generar las tensiones de referencia de estándares como diferencial HSTL (High Speed Transceiver Logic) y SSTL (Stub Series Terminal Logic). Este ultimo usado en las interfaces de memorias tipo DDR y DDR2.

En el caso del buffer se salida, también se puede configurar para un amplio rango de estándares de E/S, tal como detalla Tabla 2. Pero también se puede configurar su corriente de salida (drive strength) y la pendiente de la señal de salida (slew control). Con estos dos controles se pueden minimizar los efectos de rebote de señal (ringing) y las emisiones de alta frecuencia.

También se puede observar en la Figura 17que cada IOB tiene un diodo de protección a Vcco y otro a GND. Dispone de resistencias tipo pull-up y pull-down configurables, como así también un circuito de retención de bus (bus keeper o bus holder).

Tabla 2 - Estándares de E/S soportados por el Bloque E/S (IOB)

Estandard	Descripción	Uso	Buffer Entrada	Buffer Salida
LVTTL	Low-Voltage TTL	Propósito general 3.3V	LVTTL	Push-pull
LVCMOS	Low-Voltage CMOS	Propósito general 3.3V, 2.5V, 1.8V, 1.5V	CMOS	Push-pull
PCI	Peripheral Component Interconnect	Bus PCI	LVTTL	Push-pull
GTL	Gunning Transceiver Logic	Bus alta velocidad, backplane	V_{REF}	Open Drain
GTL+	GTL Plus	Intel Pentium Pro	V_{REF}	Open Drain
HSTL	High Speed Transceiver Logic	Interface con SRAM	V_{REF}	Push-pull
SSTL3	Stub Series Terminated Logic 3.3V	SRAM/SDRAM	V_{REF}	Push-pull
SSTL2	Stub Series Terminated Logic 2.5V	SRAM/SDRAM	V_{REF}	Push-pull
SSTL18	Stub Series Terminated Logic 1.8V	SRAM/SDRAM	V_{REF}	Push-pull
Estandares Diferenciales				
LVDS	Low-Voltage Differential Signaling	High speed interface	Diferencial	Diferencial
BLVDS	Bus LVDS	Multipoint LVDS	Diferencial	Diferencial
LVPECL	Low Voltage Positive ECL	High-speed clocks	Diferencial	Diferencial
LDT	Lightning Data Transport	Bidireccional serie/paralelo (Hyper Transport)	Diferencial	Diferencial
Mini-LVDS	Mini-LVDS	Flat Panel Displays	Diferencial	Diferencial
LVDSExt	Extensión de LVDS	Hard Drive interface	Diferencial	Diferencial
RSDS	Reduced Swing Differencial Signaling	DVI/HDMI	Diferencial	Diferencial

4.2.2. Bancos de E/S

Los bloques de E/S (IOBs) están agrupados en lo que se llaman bancos de bloques de E/S (grupos de IOBs). Dependiendo del FPGA, cada banco puede tener entre 20 y 40 IOBs. Cada banco de IOBs tiene su propia tensión de alimentación (V_{CCO}) y su propia tensión de referencia V_{REF} que son comunes a todos los IOBs. Es por esta disposición de tensiones de V_{CCO} y V_{REF} por banco, que si se quieren usar diferentes estándares de E/S con diferente V_{CCO}, se deberán usar diferentes bancos de E/S.

La Figura 18 muestra la división de los IOBs en 8 bancos de E/S en un FPGA Virtex 2.

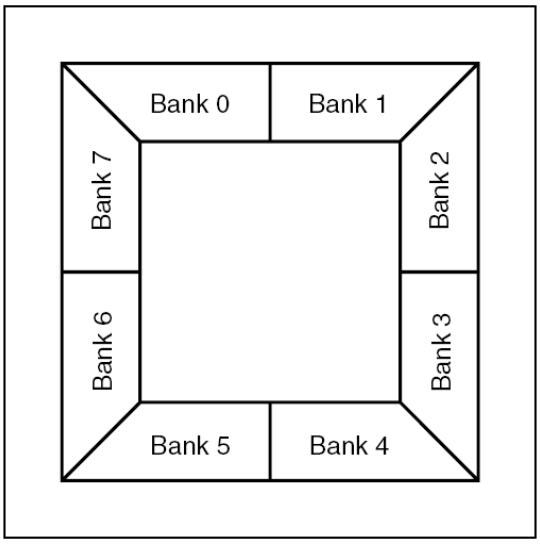


Figura 18 - Bancos de E/S

Las herramientas usadas cuando se implementa un diseño en un FPGA automáticamente asignan terminales de diferentes bancos cuando los estándares E/S usados necesitan diferentes V_{CCO} o $V_{REF.}$ De todos modos, el diseñador también tiene control suficiente para asignar terminales específicos de E/S a señales específicas de su diseño.

Otra característica disponible en los IOBs es la de *control digital de impedancia* (no representada en la Figura 17), que permite controlar la impedancia de entrada o la de salida de los terminales del IOB. Una aplicación típica es, por ejemplo, adecuar la impedancia de salida del FPGA a la impedancia de la traza del circuito impreso (típicamente 50 ohms), para evitar o reducir al mínimo las posibles reflexiones de la señal digital. Esta función es configurable en un esquema banco por banco de IO, es decir cada banco puede tener su propia adaptación de impedancia.

4.3. Bloques de Memoria RAM (BRAM)

Para aplicaciones que requieren acceso a memoria, ya sea para escritura y lectura (tipo RAM) o sólo lectura (tipo ROM), los FPGAs tienen bloques de memoria RAM (BRAM) disponibles para usarlos de acuerdo a la necesidad. Aún cuando no se usen estos bloques, siguen estando dentro del FPGA. La cantidad de bloques disponibles depende del tamaño del FPGA. En la familia Spartan, por ejemplo, el rango de bloques de RAM disponibles va desde 4 hasta 100 bloques. Cada bloque de RAM contiene 18.432 bits de RAM estática rápida, de los cuales 16K son dedicados para datos, y los

otros 2K para bits de paridad o para algunos bits extras de los datos almacenados. Se pueden conectar diferentes BRAMs en cascada, ya sea para tener un mayor ancho de la palabra de datos, para tener un mayor tamaño de la memoria o bien para ambas cosas (ancho + tamaño). En este caso el retardo es mínimo gracias a las conexione, ya que se usan recursos dedicados de ruteo. La Figura 19 muestra ejemplos de distintos tamaños de bloques de memoria que se pueden implementar usando un simple BRAM.

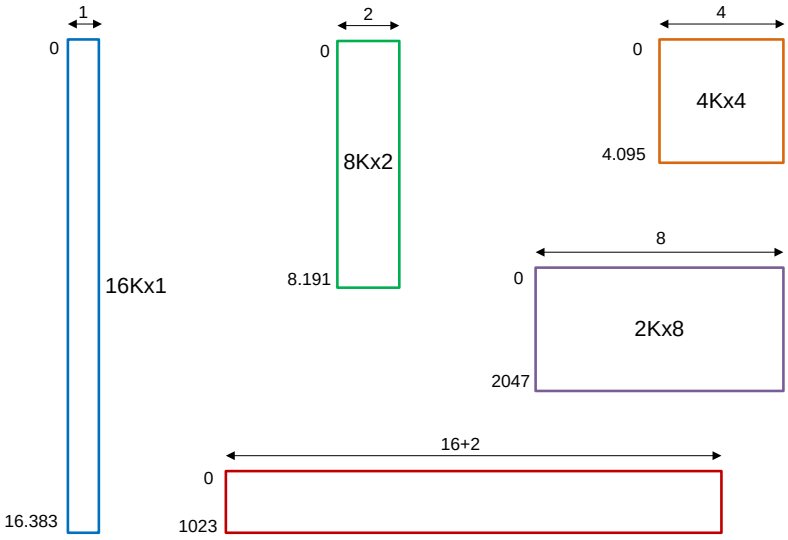


Figura 19- Diferentes configuraciones de relación datos/direcciones que se pueden implementar en los BRAMs

Los BRAM son bloques configurables de acuerdo a las necesidades del diseño, es decir el mismo bloque puede ser configurado para que funciones como RAM, ROM, FIFO (First Input First Output), convertidor de ancho de palabra, buffers circulares y registros de desplazamientos. A su vez, cada una de estas configuraciones soporta diferentes anchos de la palabra de datos y diferentes tamaños del bus de direcciones.

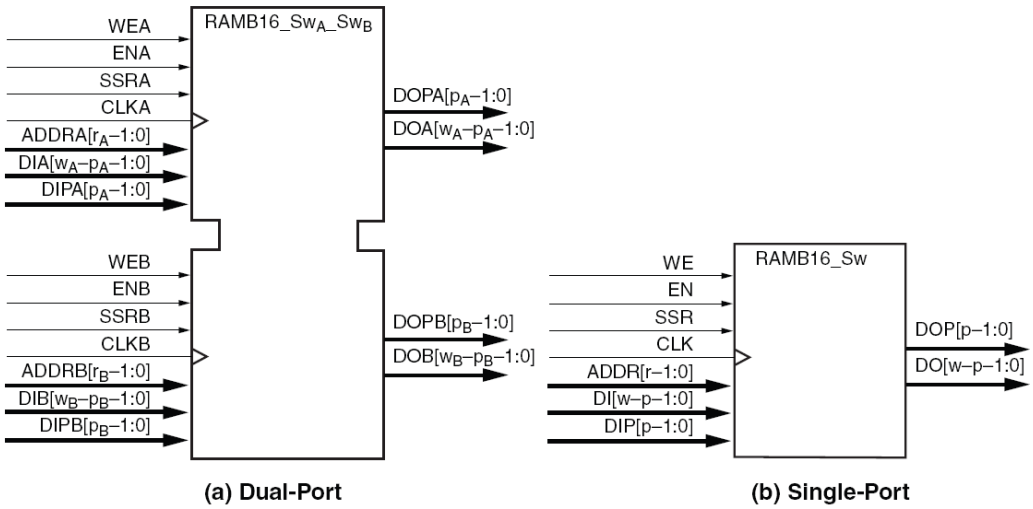


Figura 20 - BRAM de 18Kb como a) dual port y b) single port

Tal como muestra la Figura 20, físicamente el bloque RAM (BRAM), tiene dos puertos de acceso completamente independientes, llamados Puerto A y Puerto B. La estructura es totalmente simétrica. Cada puerto de memoria tiene su propia señal de reloj, habilitación de reloj y habilitación de escritura. Por ello, cada puerto de memoria es sincrónico con su propio reloj, habilitación de reloj y habilitación de escritura. La lectura de la memoria es sincrónica y requiere un reloj y una habilitación de reloj.

Entre las principales aplicaciones de los BRAM se pueden destacar las siguientes:

- Almacenamiento de programas para procesadores embebidos en el FPGA.
- Rd/Wr variables durante cálculos matemáticos, por ej. Coeficientes para filtros FIR.
- Buffers circulares.
- Registros de desplazamiento o muy largos o muy anchos.
- Líneas de retardo.
- Realización de MEF usando técnicas de microprogramación.
- Contadores muy largos.
- Memorias Direccionables por Contenido (CAM) de alto rendimiento de Rd/Wr
- Almacenamiento de formas de onda o tablas de funciones trigonométricas para generar salidas tipo Direct Digital Synthesis (DDS) (Síntesis Digital Directa).

Cabe aclarar que lo detallado con respecto al tamaño y cantidad de los BRAMs corresponde a la familia Spartan-3. Por otro lado, el Virtex-5 tiene algunas diferencias al respecto. Por ejemplo, cada bloque BRAM es de 36Kb, se puede configurar para disponer de código corrector de error tipo Hamming (usando 8 bits), las FIFOs tienen construidas la lógica de banderas dentro del BRAM, y así, otras características que por algo hacen a la V-5 más cara, pero de mayor rendimiento.

Finalmente es necesario recordar algo ya analizado en el tema Rebanadas (SLICE) de un CLB en el punto 4.1.1, para implementar pequeñas memorias se hace uso de los SLICEM, que tienen la opción de ser configurados como bloques de memoria de 16x1. Concatenando varios SLICEM se pueden tener memorias de tamaño pequeño y de rápido acceso.

4.4. Bloques de Multiplicación – Bloques DSP

Las aplicaciones de procesamiento digital de señales (DSP, Digital Signal Processing) basan sus cálculos básicamente en dos elementos: multiplicadores y sumadores. Para cálculos complejos se requiere un gran número de estos elementos, cuya implementación en LUTs resultaría muy compleja, consumiendo gran cantidad de la lógica disponible en el FPGA. Por ello, las generaciones nuevas de FPGAs incluyen en su arquitectura elementos lógicos configurables dedicados a la multiplicación (Spartan 3E) o los FPGAs más avanzados, que permiten hacer la multiplicación y la suma en paralelo (Virtex 4/5/6).

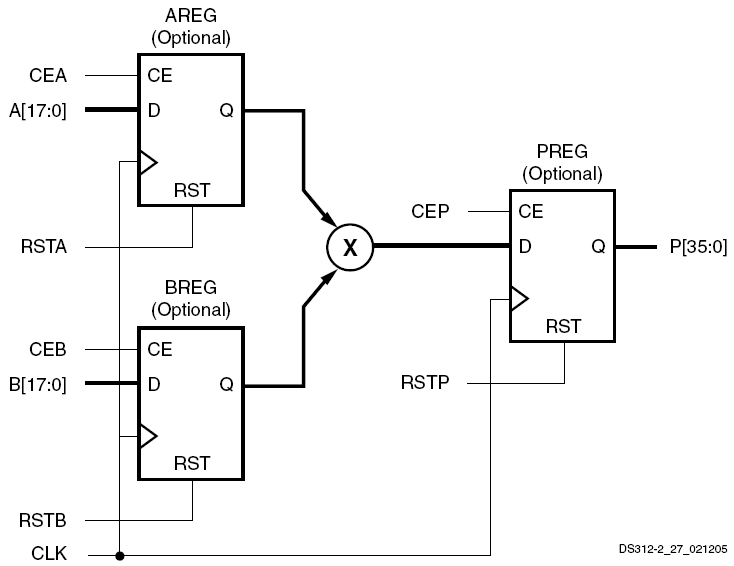


Figura 21 - Bloque de Multiplicación en el Spartan 3E

La familia de FPGAs Spartan 3 provee en cada FPGA de 4 a 36 multiplicadores dedicados, detallados en la Figura 21, Estos multiplicadores están localizados cerca de los bloques de memoria RAM a fin de tener un mínimo retardo para la lectura y/o escritura de los datos. Cada multiplicador ejecuta la multiplicación P = A x B, donde A y B son datos de 18 bits en complemento a dos, y P es el resultado de 36 bits, también en C2. Los registros AREG, BREG y PREG son opcionales, pero su uso en una configuración tipo pipelining beneficia el rendimiento total del sistema.

En el rango de los FPGAs de más alto rendimiento, y por ello mucho más caros, se encuentran disponibles bloques más complicados llamados bloques DSP, o más precisamente DSP48 (ya se entenderá porque). La Figura 22 detalla la lógica interna del DSP48 de un Virtex-5 (similar al disponible en el Virtex-6).

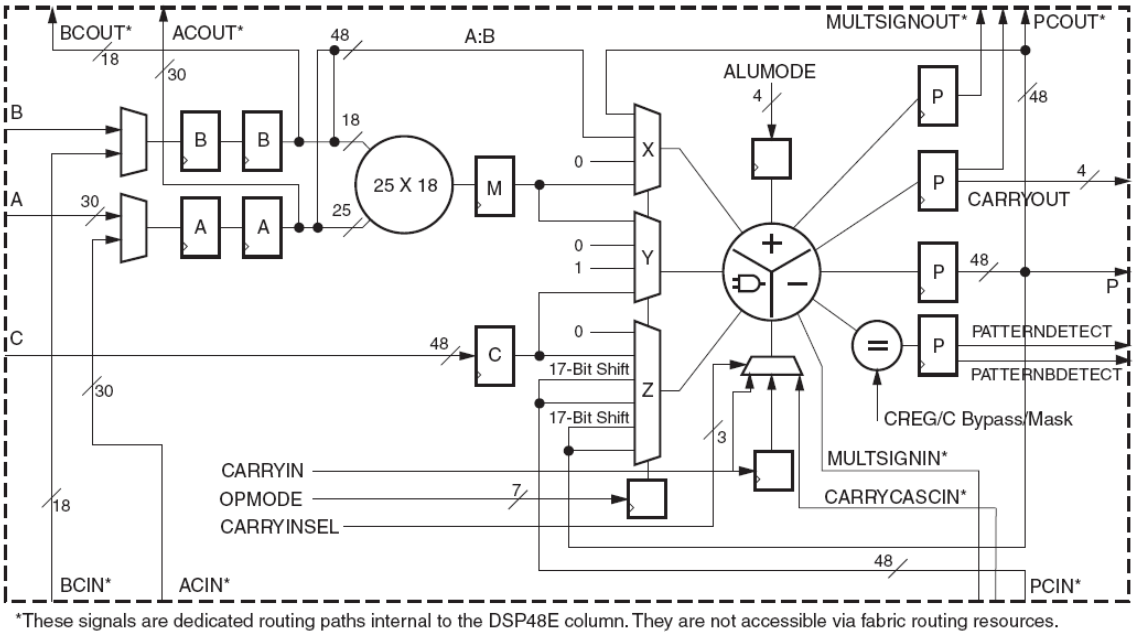


Figura 22 - Bloque DSP en un Virtex-5/6

Viendo en detalle la Figura 22, se observa que consiste en un multiplicador de 25x18 bits en C2 y un acumulador de 48 bits. En conjunto pueden llegar trabajar a frecuencias de hasta 550MHz, lo que hace al FPGA un dispositivo muy adecuado para el cálculo en tiempo real de variables aritméticas complejas. Si se tiene en cuenta que un FPGA tipo Virtex-5 puede tener hasta 1056 bloques DSP y, más importante aún, es que pueden trabajar todos al mismo tiempo en paralelo. En muchas aplicaciones el FPGA a desplazados a los procesadores dedicados DSP por su paralelismo y alto rendimiento. Cuando se necesitan hacer cálculos complejos, por ejemplo Correlaciones, Filtros FIR, Convoluciones, etc.; se usa un flujo de diseño que comienza en MatLab-SimuLink y termina con la herramienta del fabricante del FPGA. Esto facilita enormemente el proceso de implementación de cálculos matemáticos complejos y aprovecha toda la capacidad del FPGA.

4.5. Interconexiones en los FPGAs

Además de las celdas de lógica programable, los FPGAs tienen celdas de interconexión programables. Estas definen el camino (ruta) a seguir por cada señal interna del FPGA. La tecnológica de configuración y la arquitectura de la celda lógica determinan la estructura y complejidad de la interconexión. La Figura 23 muestra cómo se distribuyen las rutas de interconexión entre los CLBs. A la izquierda se muestra más en detalle cómo es la conexión interna entre las diferentes rutas de conexión, programable a través de transistores,

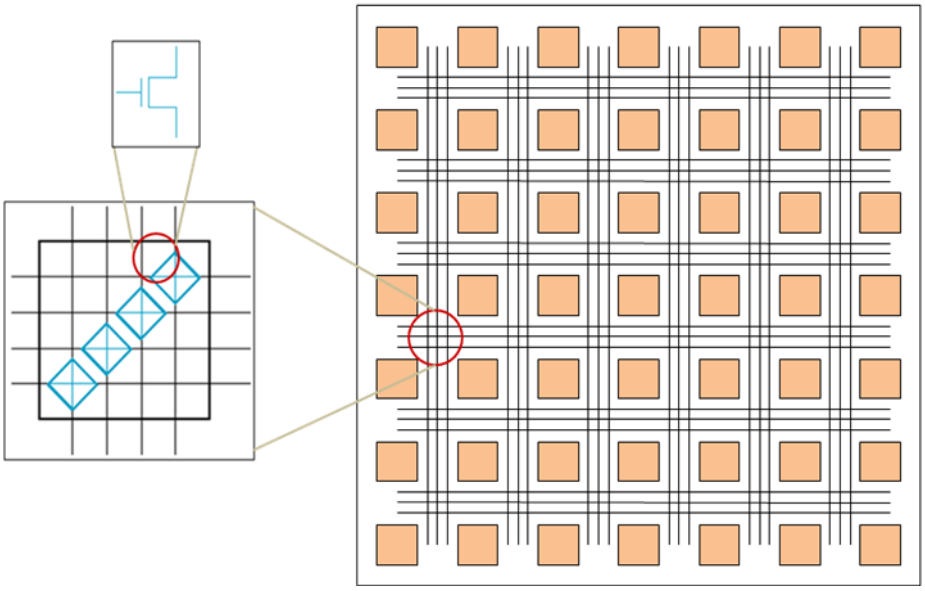


Figura 23 - Interconexión entre distintos caminos de ruteo

La programabilidad de la interconexión, si bien es una gran ventaja, agrega retardos a la señal que pasa por el transistor. Así, si una señal debe pasar por varios elementos de interconexión la suma total de los retardos puede ser considerable y debe ser tenida en cuenta en diseños de alta frecuencia. Por ello, a fin de no ir sumando demasiados retardos en cada interconexión, existe lo que se llaman *rutas largas*. De este modo, si una señal tiene que pasar por más de medio dispositivo puede hacerlo usando estas rutas largas, evitando las interconexiones intermedias. Del mismo modo, existen también las rutas cortas, llamadas *conexiones directas*, que comunican un bloque lógico con sus bloques vecinos.

Dentro del FPGA también existen lo que se llaman *rutas dedicadas*, que se usan para las señales que tienen mucha cargabilidad de salida (fan-out) en el sistema que se está diseñando. Las señales que más comúnmente usan las rutas dedicadas son el reloj y la habilitación del reloj (clock enable). Estas rutas tienen buffers especiales que hacen que la señal no se distorsione con la carga. Por ejemplo, en diseños grandes es fácil encontrar una señal de reloj que llegue a 2000 flip-flops.

Hay una relación directa entre la cantidad de interconexiones y el tamaño físico del FPGA. Por ello es que no se implementan físicamente todas las interconexiones posibles, así como, no todos los bloques lógicos están interconectados entre sí. A raíz de esto, para poder conectar la salida de un bloque lógico con la entrada de otro a veces se pasa por diversas interconexiones que conectan diferentes rutas para poder llegar a destino. Por suerte para el diseñador, el software del fabricante del FPGA realiza automáticamente todas las interconexiones. Es una tarea totalmente transparente al diseñador, siempre y cuando se cumplan los requisitos de tiempo estipulados por las especificaciones del sistema implementado. De lo contrario, el diseñador deberá optimizar algunos parámetros de la herramienta de ruteo (llamada Place and Route) para que pueda cumplir con las especificaciones de tiempo requeridas.

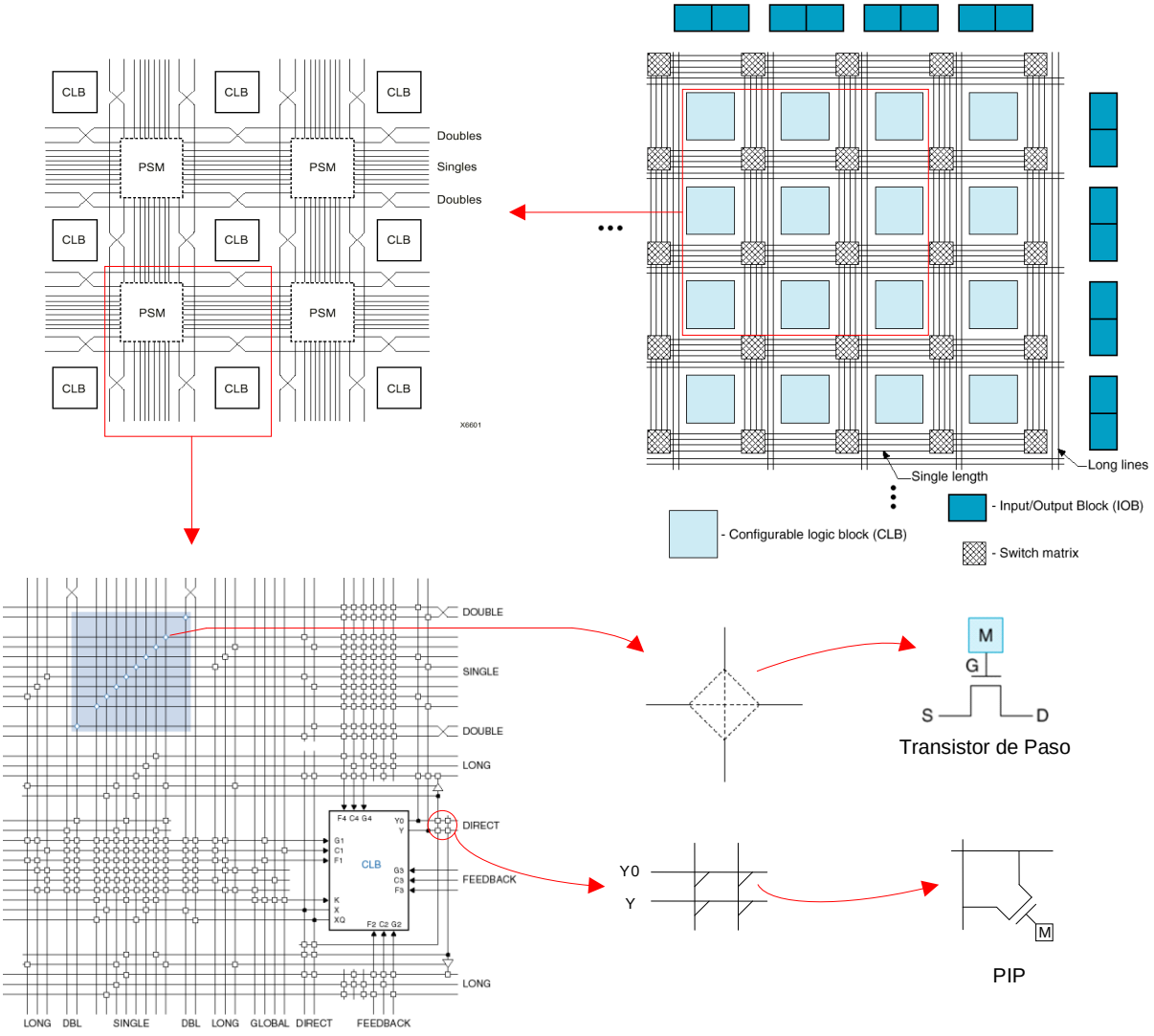


Figura 24 - Vista macroscópica a vista microscópica de las interconexiones de un FPGA de Xilinx

La Figura 24 detalla las interconexiones en un FPGA de Xilinx. Se destacan las líneas verticales y horizontales que corren entres los bloques lógicos. La matriz de interconexiones programable (Programmable Switching Matrix, PSM) conecta las diferentes rutas dentro del FPGA a través de los transistores de paso. Se observan también las rutas largas (long lines) que cruzan todo el FPGA. Las rutas de conexión directa saltan sobre las matrices de interconexión para conectar directamente bloques lógicos adyacentes. Se destacan también los transistores que conectan las entadas y salidas de los bloques lógicos a las rutas generales de interconexiones, llamados puntos de interconexión programables (Programmable Interconnection Points, PIP).

La Figura 25 detalla cómo se conectan dos bloques lógicos por medio de las matrices de conexión programables (PSM) y los puntos de interconexión programables (PIP).

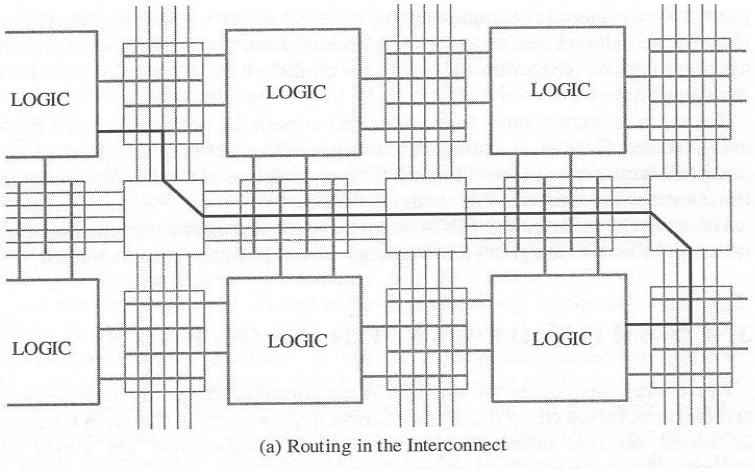


Figura 25 - Ejemplo de ruteo entre dos bloques lógicos a través de PIPs y PSMs

Finalmente, la Figura 26 resalta la importancia de las interconexiones de ruteo en un diseño. Es decir, del total de los recursos de un FPGA usados en este diseño, el 31% corresponde a recursos de interconexión, que es el porcentaje más elevado de recursos usados en este caso. Es por ello que este tema de las interconexiones y el ruteo son de gran importancia, realizándose estudios complejos para obtener algoritmos de optimización del ruteo, a fin de obtener resultados óptimos y evitar el *congestionamiento de las rutas de conexión* que derivan en una reducción del rendimiento general del FPGA.

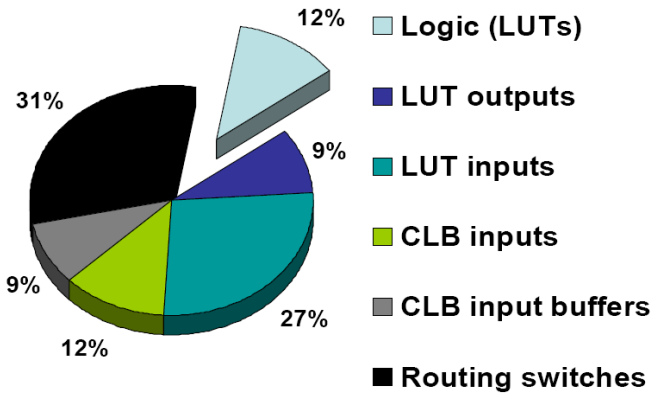


Figura 26 - Este gráfico resalta la importancia del retardo en los conectores de ruteo (Dr. G. Lemieux, UBC)

4.6. Generación de Reloj y su Distribución

Los FPGA tienen unos bloques de lógica dedicados exclusivamente a funciones de control y generación de señales de reloj. En los FPGAs de Xilinx a estos bloques genéricamente se los llama Digital Clock Managers (DCMs) (Gestores de Reloj Digitales). La cantidad de estos bloques disponibles en un FPGA depende del tamaño del mismo, puede haber desde 2 DCMs en los FPGAs más pequeños hasta 12 DCMs en los FPGA grandes.

Los DCMs integran capacidades avanzadas del reloj, dentro de la red de distribución dedicada del reloj del FPGA. Las principales funciones del DCM se pueden resumir en:

- Eliminar el sesgo del reloj (clock skew), ya sea dentro del FPGA o con componentes externos. De este modo se mejora el rendimiento del sistema y se eliminan los retardos de ruteo del reloj.
- Producir corrimiento de fase (Phase shifting) de una señal de reloj, ya sea por una fracción del periodo de reloj o por incrementos fijos.
- Multiplicar o dividir la frecuencia de entrada del reloj, generando una frecuencia completamente nueva.
- Acondicionar la señal de entrada del reloj, asegurando un reloj limpio, con un ciclo de trabajo del 50%.
- Amplificar de nuevo (rebuffer) una señal de reloj, normalmente para eliminar el sesgo (deskew) y convertir la señal de entrada a un estándar diferente, por ejemplo, de LVDS a LVTTL.

La Figura 27 detalla gráficamente las distintas funcionalidades del DCM explicadas en los puntos anteriores (excepto el corrimiento de fase).

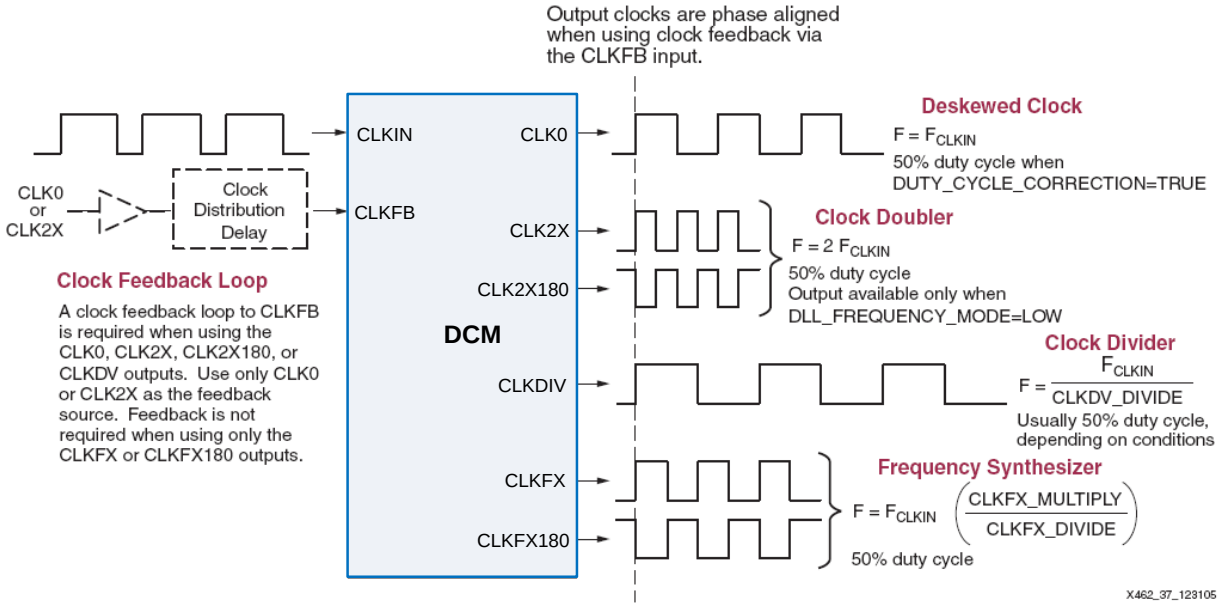


Figura 27 - Diferentes opciones de uso del DCM

Tal como se explicó anteriormente, la señal de reloj usa rutas de conexión dedicadas, cuya distribución dentro del FPGA es bastante particular, tal como se puede ver en la Figura 28. El FPGA

se divide en cuatro cuadrantes, los cuales pueden tener como máximo 8 relojes. Multiplexers dedicados seleccionan las señales de reloj que se van a usar en cada cuadrante. Las líneas de ruteo de reloj son líneas dedicadas de muy bajo retardo, de casi nula distorsión del ciclo de trabajo, con jitter (oscilación) mínimo y llegan a todos los elementos sincrónicos del FPGA.

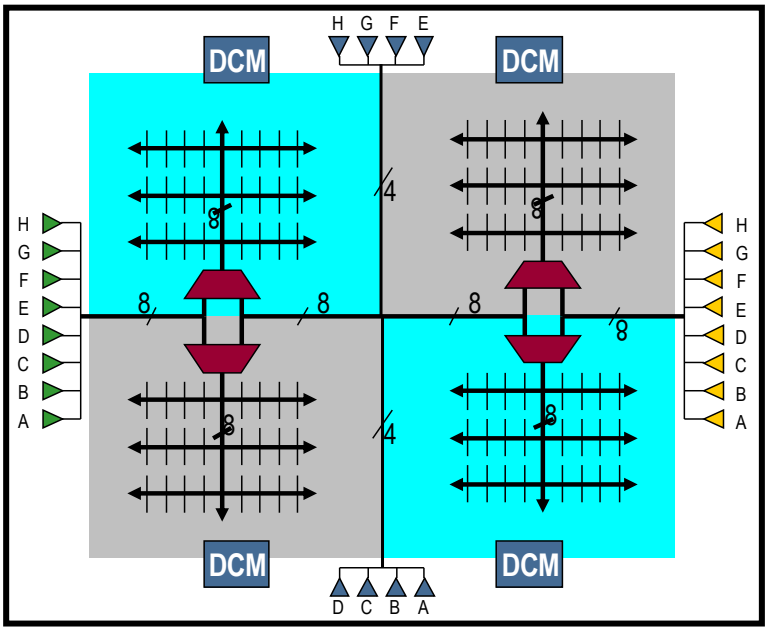


Figura 28 - Distribución de los DCMs y rutas de reloj en un FPGA Spartan

Los buffers de entrada (A-H) son buffers con características especiales, como, por ejemplo, conexión directa con el DCM (para evitar retardos de ruteo). De todos modos, una señal de reloj externa puede entrar a través del buffer de entrada y ser ruteado a los flip-flops del FPGA sin pasar por el DCM.

Para configurar un DCM se usa un software del fabricante del FPGA. En el caso de Xilinx se llama CoreGen (Core Generator) el cual tiene una interfaz grafica que facilita la configuración del DCM de acuerdo a las necesidades del diseño.

4.6.1. Ejemplo de uso del DCM- Eliminando el sesgo del reloj

Se detallará un ejemplo de uso del DCM como supresor del sesgo del reloj del sistema.

En todo sistema sincrónico existe inherentemente sesgo de reloj. Aún la señal de reloj más precisa llega en tiempos diferentes a los diferentes puntos del sistema, ya sea en un mismo dispositivo o a los diferentes dispositivos conectados a la fuente de reloj. Esa diferencia en el tiempo de llegada se conoce como sesgo de reloj (clock skew).

La Figura 29 detalla un ejemplo de un sistema simple que consta de un FPGA y otro dispositivo, que puede ser, por ejemplo, un microcontrolador, un ASIC, o lógica discreta. El reloj entra al FPGA (punto A) a través de un buffer de entrada, por ende sufre un cierto retardo. Usando ruteo dedicado (mínimo retardo, pero retardo al fin) llega a los flip-flops del FPGA (punto B). Luego de un cierto retardo, llamado Δb en la Figura 29, sale a través de un buffer de salida (más retardo). Además, se le debería sumar también el retardo que sufre la señal a través de la línea de conexión del circuito

impreso. Así, cuando el reloj llega al otro dispositivo (punto C) tiene un cierto retardo, creando un cierto desfasaje (sesgo) con el reloj original y con el reloj interno del FPGA (llamado ΔC en la Figura 29). El retardo es tal, que si se quieren transmitir datos desde el FPGA al dispositivo (punto D) se corre el riesgo de violar los tiempos del establecimiento o sostenimiento en el flip-flop del dispositivo receptor.

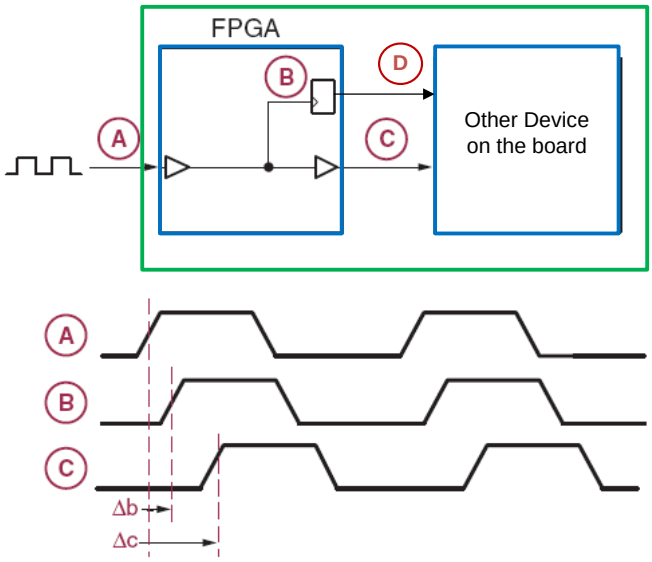


Figura 29 - Sistema con sesgo de reloj

Este problema del sesgo del reloj en la jerga técnica se conoce como 'el ladrón del rendimiento del sistema'. Como consecuencia de todos estos retardos, se incrementan los tiempos de establecimientos y los tiempos de retardo de los flip-flops (clock to output) y por lo tanto, se incrementa el periodo de reloj del sistema. De forma análoga, en algunos casos también puede que se requieran tiempos de mantenimiento más largos. Todos esto lleva a que necesariamente se deba buscar una solución a este problema.

Afortunadamente los DCMs internos de un FPGA ofrecen una manera sencilla de eliminar el sesgo de reloj de un sistema. La Figura 30 muestra un ejemplo de uso de dos DCMs del FPGA, un DCM tiene como función eliminar el sesgo del reloj dentro del FPGA, y el otro DCM elimina al sesgo del sistema, resultando prácticamente un alineamiento ideal del reloj en los distintos puntos (A, B, C, D) del sistema.

Cómo es elimina el sesgo del reloj?. Se debe recordar primero que el sesgo es provocado por los retardos en el camino del reloj (buffers, ruteo, etc). En la Figura 30 el reloj en el punto B está retardado por ΔB y el reloj en el punto C está retardado por ΔC. Qué pasaría si existiera un modo de proveer en el punto B una versión 'adelantada' por el tiempo ΔB del reloj?, y otro modo de proveer al punto C una versión 'adelantada' por un tiempo ΔC del reloj? El resultado sería que *todos* los relojes arribarían a su destino perfectamente alineados tal como muestran las formas de ondas de la Figura 30-B. En realidad, el DCM lo que hace es retardar la señal de entrada de modo que parece estar adelantada en el tiempo. Por ejemplo en la Figura 30-C, el reloj en el punto B parece estar adelantado en el tiempo por el retardo ΔB. Sin embargo, la realidad es que el reloj está retardado por T (periodo del reloj)- ΔB. Del mismo modo para el reloj en punto C, el cual es retardado por T-ΔC.

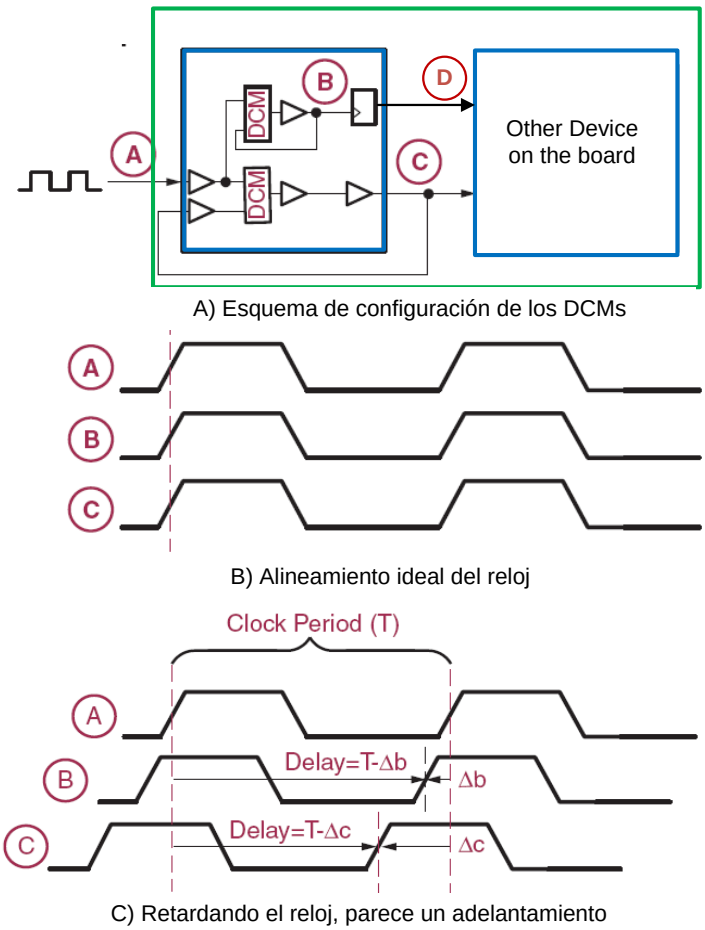


Figura 30 - Uso de DCMs para eliminar sesgo de reloj dentro del FPGA y del sistema

Una ventaja aún mayor al usar el DCM es que no es necesario conocer los valores de ΔB y ΔC, es decir no hace falta ni calcularlos ni medirlos, sino que el mismo DLL dentro del DCM constantemente monitorea el retardo por medio del lazo de realimentación (feedback loop) tal como se muestra en la Figura 30-A.

Al usar los DCMs la performance total del sistema es óptima, anulando cualquier inherencia del sesgo del reloj en la misma.

5. Configuración de un FPGA

Dispositivos electrónicos como microprocesadores, microcontroladores, procesadores DSPs y periféricos son totalmente programables a través de una serie de unos y ceros empaquetados en lo que se llama programa. Por eso se dice que un procesador se programa cuando se carga en la memoria de programa un archivo binario que el procesador ejecutará. Por otro lado, los FPGAs son dispositivos lógicos reprogramables, pero en un FPGA el proceso de carga o programación es llamado configuración (no programación), usando un archivo de configuración llamado 'bits de configuración' ('configuration bitstream') que define la funcionalidad del FPGA.

El FPGA puede opcionalmente auto-cargar el archivo de configuración, por ejemplo desde una memoria externa no-volátil. Un FPGA puede también ser configurado por un dispositivo tipo

microprocesador, DSP procesador, PC o algún otro dispositivo que pueda comunicarse inteligentemente con el FPGA. Es decir, un FPGA puede, a) auto-cargar el archivo de configuración residente en un componente externo, en este caso se dice que el FPGA opera en Modo Maestro, o b) otro dispositivo externo puede cargar el archivo de configuración en el FPGA, en cuyo caso el FPGA opera en Modo Esclavo.

A continuación se detallarán los métodos más comunes de configurar un FPGA.

5.1. Modo Maestro

Cuando el FPGA auto-carga el archivo de configuración desde un dispositivo externo, se dice que el FPGA usa el Modo Maestro (Master Modo) para cargar su configuración. Esta carga se puede efectuar en un modo de transmisión de datos serie o modo paralelo. Normalmente en este modo el archivo de configuración reside en una memoria no-volátil externa al FPGA, y el FPGA genera la señal de reloj, CCLK, y controla la comunicación con la memoria.

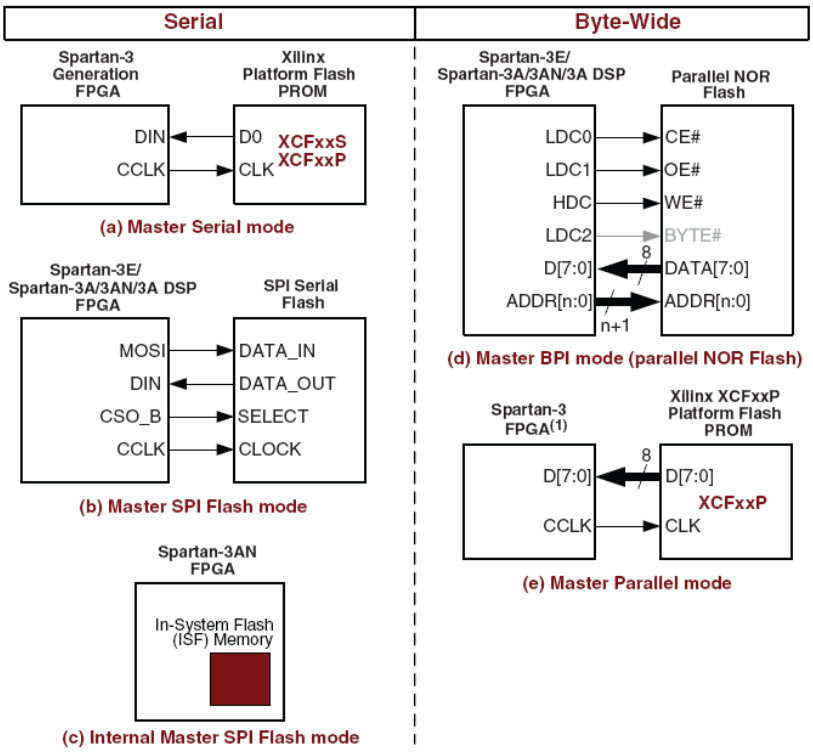


Figura 31 - Distintas opciones de configuración del FPGA en Modo Maestro (Master Mode)

La Figura 31 detalla diferentes opciones de configuración del FPGA en modo Master. La Figura 31.a y Figura 31.b son situaciones similares donde el FPGA extrae de la memoria externa su configuración. La Figura 31.b detalla una interface SPI muy común para este tipo de configuración pues la memoria es barata, y ocupa muy poco lugar en el PCB. Una configuración de un FPGA tamaño medio usando este método Master-Serial puede tardar unos 500ms entre la comunicación y hasta que el FPGA esté listo para funcionar. Dentro del modo Master también está la opción de comunicarse en modo paralelo con una memoria externa tipo NOR Flash o PROM Flash. Las Figura 31.d y Figura 31.e detallan éste tipo de interface. En este caso la gran ventaja es la disminución del tiempo de carga del archivo de configuración, mientras que la principal desventaja es la cantidad de líneas necesarias para rutear en el PCB.

5.2. Modo Esclavo

Otra opción de configurar el FPGA es el modo en que el FPGA recibe los datos, el reloj y las señales de control del dispositivo que controla la comunicación entre ambos. En este caso el FPGA actúa como esclavo del otro dispositivo, por ello el modo se llama Modo Esclavo. La Figura 32 detalla las distintas opciones para este método. Tal como ocurre en el Modo Master, en éste modo también existe una comunicación serie y una paralela. Por lo general el dispositivo maestro es un dispositivo inteligente tipo microcontrolador/microprocesador que se encarga de toda la comunicación. Una de las grandes ventajas de este método es que el archivo de configuración puede residir en cualquier parte del sistema, puede ser en un disco duro, en una memoria conectada a Internet, o en la memoria que también contiene el archivo de programación del micro.

El modo de configuración conocido como JTAG, Figura 32.b, es el modo más usado durante la etapa de debug o prototipo del sistema a implementar en el FPGA. Una vez seguidos todos los pasos para implementar el sistema en el FPGA, el último paso es generar el archivo de configuración del FPGA. Para probar si el diseño sintetizado funciona correctamente se debe configurar el FPGA y comprobar su funcionamiento. Durante la etapa de prueba o prototipo, el software de desarrollo del fabricante del FPGA provee una herramienta para configurar el FPGA a través de un cable tipo USB que se conecta por un lado a la PC y por otro a esta conexión estándar llamada JTAG. De este modo si el FPGA no funciona como se esperaba, se corrigen los problemas en el código descriptivo del funcionamiento, se sintetiza, se genera un nuevo archivo de configuración y se configura de nuevo el FPGA.

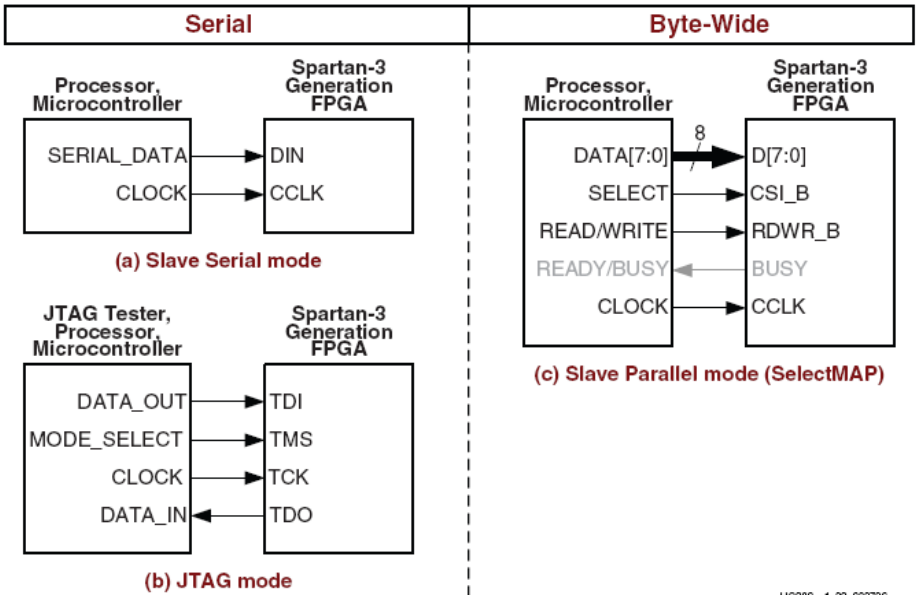


Figura 32 - Distintas opciones de configuración del FPGA en Modo Esclavo (Slave Mode)

6. Acrónimos

ASIC	Application Specific Integrated Circuit
BRAM	Block RAM
CLB	Configurable Logic Block
DCM	Digital Clock Manager
DDR	Dual Data Rate
DLL	Delay Lock Loop
DSP	Digital Signal Processing
EDA	Electronics Design Automation
E/S	Entrada - Salida
FIFO	First Input First Output
FPGA	Field Programmable Gate Array
HSTL	High Speed Transceiver Logic
IP	Intellectual Property
I/O IO	Input – Output
ISP	In-Circuit Programmable
IOB	Input Output Block
JTAG	Joint Test Action group
MIM	Metal-Insulator-Metal
ONO	Oxide-Nitride-Oxide
OTP	One Time Programmable
PCB	Printed Circuit Board
PLICE	Programmable Logic Interconnect Circuit Element
PIP	Programmable Interconnect Point
PLL	Phase Lock Loop
PSM	Programmable Switch Matrix
SiO_2	Silicon Dioxide
SPI	Serial Protocol Interface
SRAM	Static Random Access Memory
SSTL	Stub Series Terminal Logic
TCK	Test Clock
TDI	Test Data In
TDO	Test Data Out
TMS	Test Mode Select
VHDL	Very High Speed Hardware Description Language

Bibliografía

- "*Spartan-3 Generation FPGA User Guide*", UG331, Xilinx Inc.
- "*Spartan-3 Generation Configuration User Guide*", UG332, Xilinx Inc.
- "*Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet*", DS083, Xilinx Inc.
- "*Virtex-5 User Guide*", UG190, Xilinx Inc.
- "*High-Speed Digital Design, A Handbook of Black Magic*", Howard Johnson and Martin Graham, Prentice Hall, 1995.
- *"ProASIC 3 Fabric User's Guide",* Actel Corporation.
- *"Radiation Tolerant ProASIC3 Single Even Latch-Up"*, Actel Corporation.
- "*Digital Design, Principles & Practices*", Forth Edition, John Wakerly, Prentice Hall, 2004.
- "*A VHDL Synthesis and Modeling Methodology for FPGAs*", Cristian Sisterna, Arizona State University, 1998.
- "*Digital Design Using Field Programmable Gate Arrays*", Pak Chan and Semiha Mourad, Prentice Hall, 1994.
- *"A Lab Guide for Synthesizing a Digital Design into an FPGA with Synplify VHDL Synthesis Tool".* Cristian Sisterna and Charles Lipari. Department of Electronics and Computer Engineering Technology. Arizona State University, 1997.
- "*Digital System Design and Prototyping Using Field Programmable Logic"*, Zoran Salcic and Asim Smailagic, Kluwer Academic Publisher, 1999.