

MÁSTER OFICIAL EN INGENIERÍA ELECTRÓNICA

SISTEMAS INTEGRADOS

BLOQUE 3: Arquitectura de las FPGAs de Xilinx

Curso 2013-2014

José Torres

Raimundo García

Julio Martos

Jesús Soret

Adrián Suárez

Pedro A. Martínez

Abraham Menéndez



SISTEMAS INTEGRADOS

Tema 3.- Arquitectura de las FPGAs de Xilinx

**Máster Oficial en Ingeniería Electrónica
Curso 2013-14**



Arquitectura de las FPGAs de Xilinx

Índice

- ❑ Introducción a las FPGAs.
 - ❑ Arquitectura familia Virtex II.
 - ❑ Virtex II Pro frente a Virtex II.
 - ❑ Virtex 4-5 frente a Virtex II.
 - ❑ Actualidad y novedades.
 - ❑ Normas de selección.
-

Arquitectura de las FPGAs de Xilinx

Introducción FPGAs

- ❑ ¿Qué es una FPGA?
 - ❑ Es un tipo de circuito digital configurable con arquitecturas distribuidas.
 - ❑ Permiten la realización de cualquier sistema combinacional o secuencial.
 - ❑ Las FPGAs se realizan en muy gran (VLSI) o ultra gran (ULSI) escala de integración.
 - ❑ Con tecnología CMOS exclusivamente.
 - ❑ Las FPGAs se han desarrollado gracias a la combinación de la arquitectura de los circuitos semicustom con la programabilidad de los PLDs.
-

Arquitectura de las FPGAs de Xilinx

Introducción FPGAs

- ❑ Las FPGAs se caracterizan por poseer:
 - ❑ Recursos lógicos, constituidos por bloques lógicos configurables, repartidos por todo el circuito:
 - ❑ Bloques lógicos internos.
 - ❑ Bloques entrada/salida.
 - ❑ Recursos de interconexión formados por:
 - ❑ Líneas de interconexión.
 - ❑ Conexiones configurables.
 - ❑ Los bloques lógicos internos realizan las funciones principales de la aplicación.
 - ❑ Los bloques de entrada/salida nos permiten conectar nuestro dispositivo con el exterior.
 - ❑ Los recursos de interconexión conectan interiormente cada bloque lógico dando versatilidad a la FPGA.
-

Arquitectura de las FPGAs de Xilinx

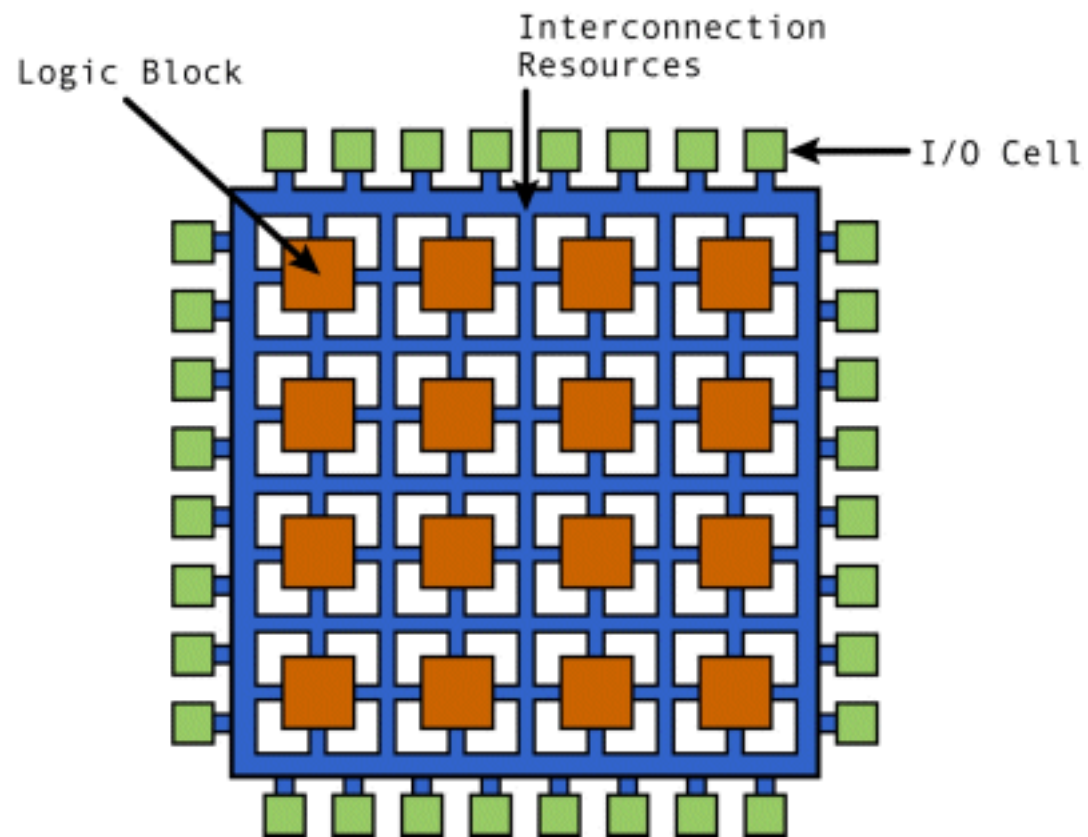
Introducción FPGAs

- ❑ Las características principales de las FPGAs son las siguientes:
 - ❑ Gran capacidad de integración, permite realizar sistemas digitales complejos en un único circuito integrado.
 - ❑ Arquitectura flexible que se adapta fácilmente a cada aplicación.
 - ❑ Poseen recursos específicos para la realización de circuitos aritméticos, reduciendo los retardos y haciéndolos más eficientes.
 - ❑ Presentan recursos lógicos específicos para realizar unidades de memoria internas.
 - ❑ Al ser reconfigurables, podemos cambiar casi en tiempo real la función que realiza el circuito.
 - ❑ Podemos, por tanto, realizar circuitos adaptativos.
 - ❑ Programación por descripción hardware, capacidad de trabajar varias aplicaciones en paralelo.
-

Arquitectura de las FPGAs de Xilinx

Introducción FPGAs

- ❑ Esquema de una FPGA clásica:



Arquitectura de las FPGAs de Xilinx

Introducción FPGAs

❑ Familias de FPGAs:

❑ Familias de altas prestaciones

- ❑ Virtex (220 nm)
- ❑ Virtex-E, Virtex-EM (180 nm)
- ❑ Virtex-II, Virtex-II PRO (130 nm)
- ❑ Virtex-4 (90 nm)
- ❑ Virtex-5 (65 nm)
- ❑ Virtex-6 (40 nm)
- ❑ Virtex-7 y Zynq (28 nm) - recién salidas

❑ Familias de bajo coste

- ❑ Spartan-3 (90 nm)
 - ❑ Spartan-3E (90 nm) - optimizadas para lógica
 - ❑ Spartan-3A (90 nm) - optimizadas para entradas/salidas
 - ❑ Spartan-3AN (90 nm) - memoria no volátil
 - ❑ Spartan-3A DSP (90 nm) - optimizadas para DSPs
 - ❑ Spartan-6 (45 nm)
 - ❑ Artix y Kintex (28 nm) - recién salidas
-

Arquitectura de las FPGAs de Xilinx

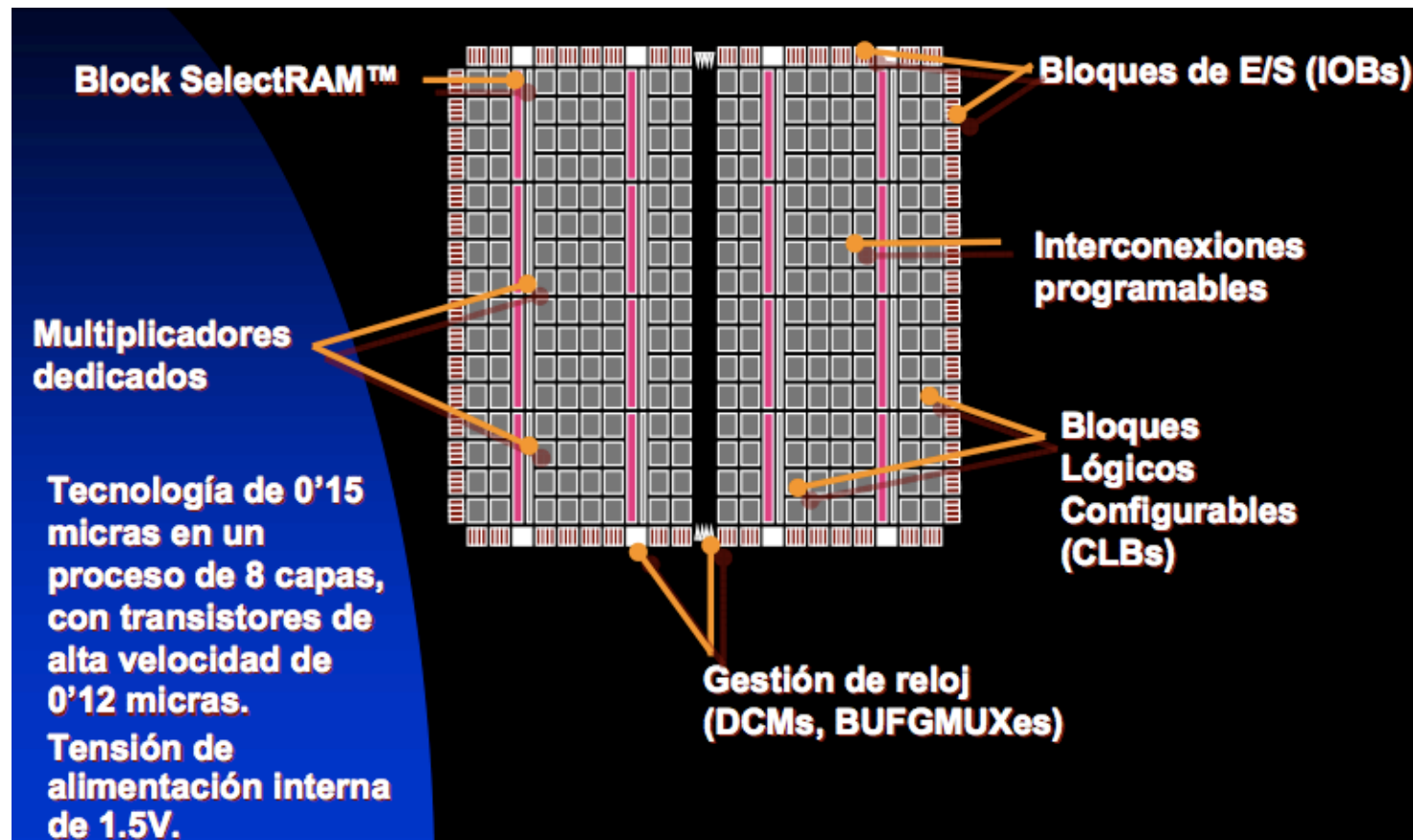
Arquitectura Virtex II

- ❑ Las FPGAs han dejado de tratarse como un dispositivo simple en cuanto a su arquitectura y forma de seleccionarlas para convertirse en verdaderos Sistemas Integrados con familias y características muy diversas.
 - ❑ A lo largo de los últimos años, la evolución ha sido espectacular. Desde el primer cambio que se produjo con la Familia Virtex II (año 2001) hasta las nuevas tecnologías de hoy en día.
 - ❑ Cuando queremos obtener un producto basado en FPGA que cumpla los 5 beneficios que nos aportan (Rendimiento, Time to Market, Precio, Fiabilidad y Mantenimiento), es necesario conocer a fondo la arquitectura de las mismas.
 - ❑ Vamos, en las siguientes transparencias, a analizar las arquitecturas de las familias Virtex y Spartan de Xilinx basadas en tablas de consulta (LUT).
-

Arquitectura de las FPGAs de Xilinx

Arquitectura Virtex II

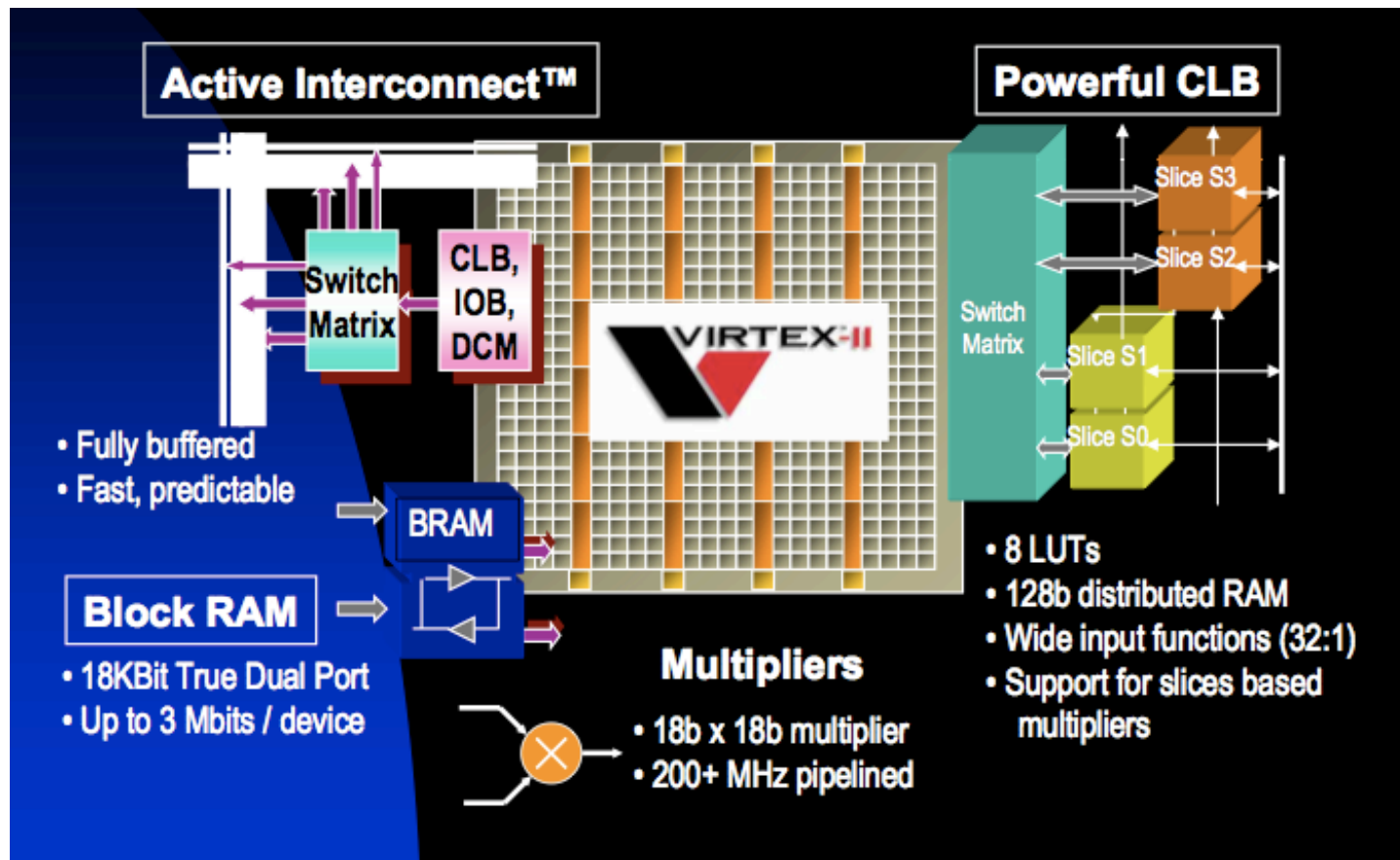
- ❑ Diagrama de bloques de una Virtex II



Arquitectura de las FPGAs de Xilinx

Arquitectura Virtex II

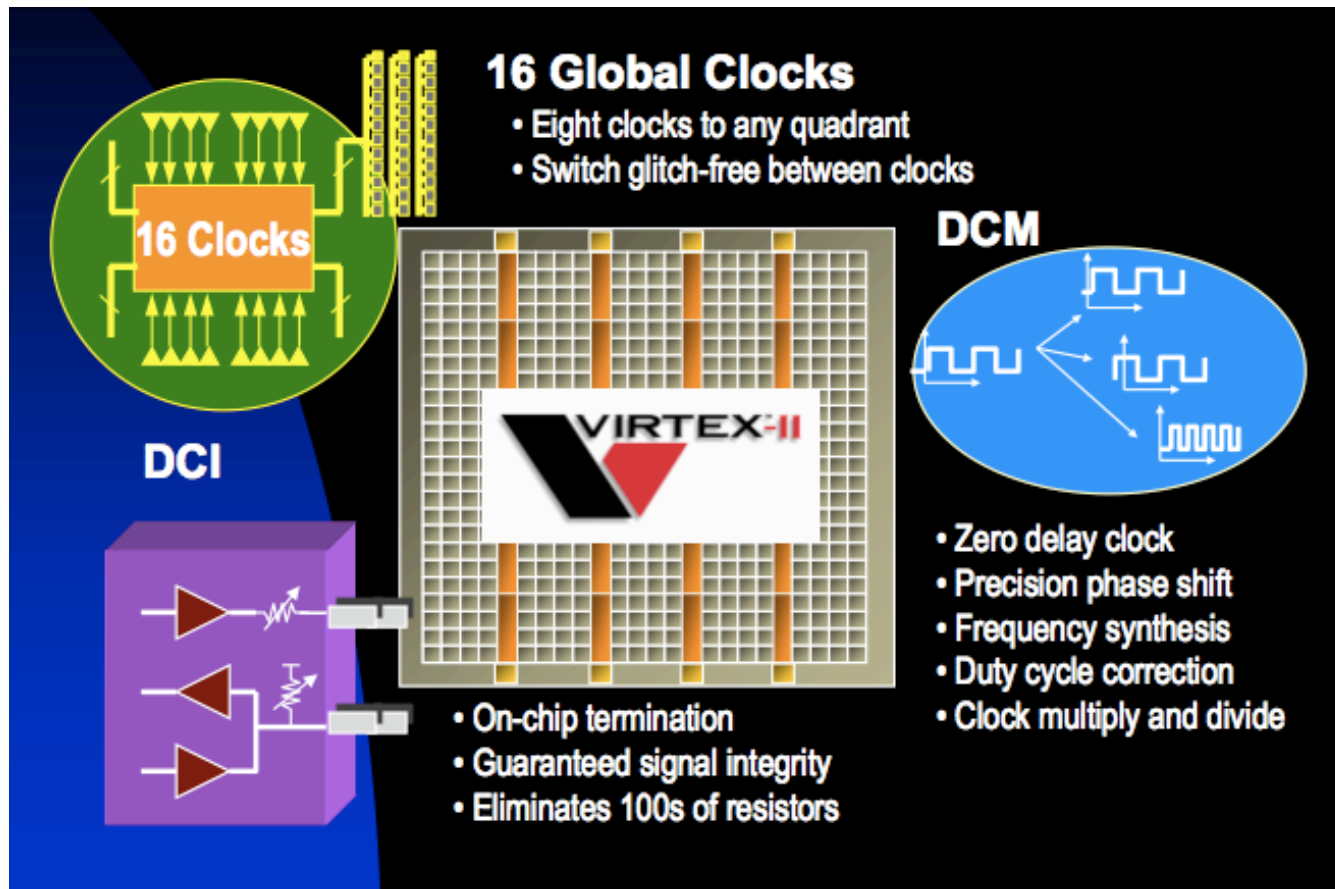
- ❑ Diagrama de bloques de una Virtex II



Arquitectura de las FPGAs de Xilinx

Arquitectura Virtex II

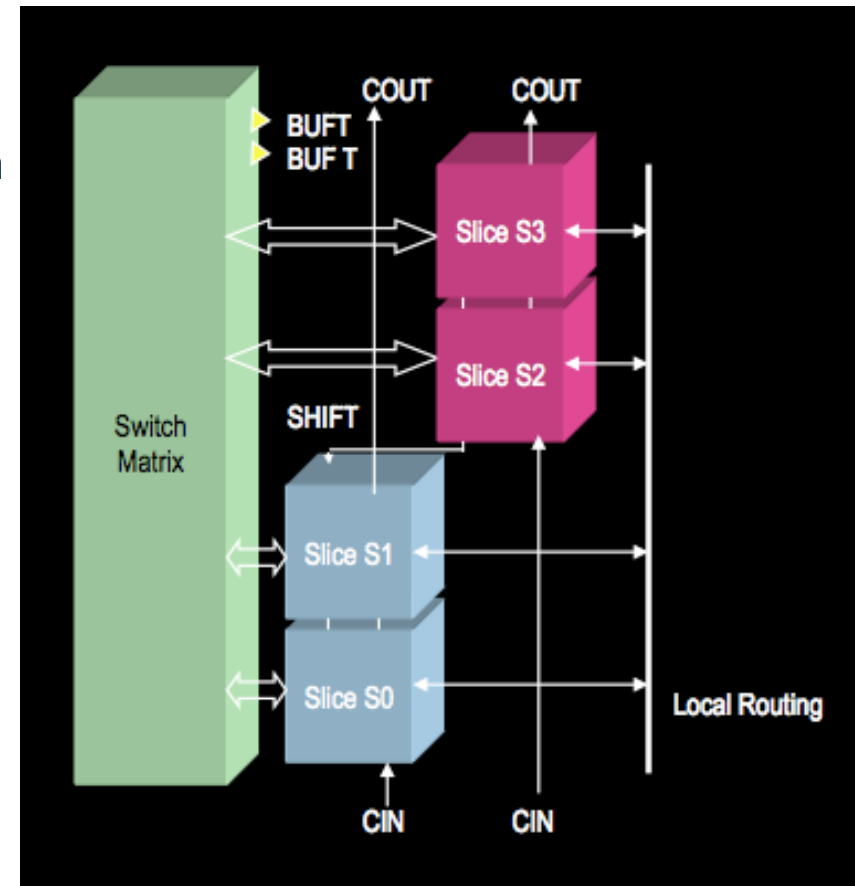
- ❑ Diagrama de bloques de una Virtex II



Arquitectura de las FPGAs de Xilinx

Arquitectura Virtex II

- ❑ Bloques Lógicos Configurables (CLBs)
 - ❑ Cada CLB contiene cuatro Slices (Bloques)
 - ❑ Hay líneas locales que interconectan los Slices de un CLB y éste con los CLBs vecinos.
 - ❑ La matriz de conmutación (Switch Matrix) permite acceder a los recursos de interconexión generales.



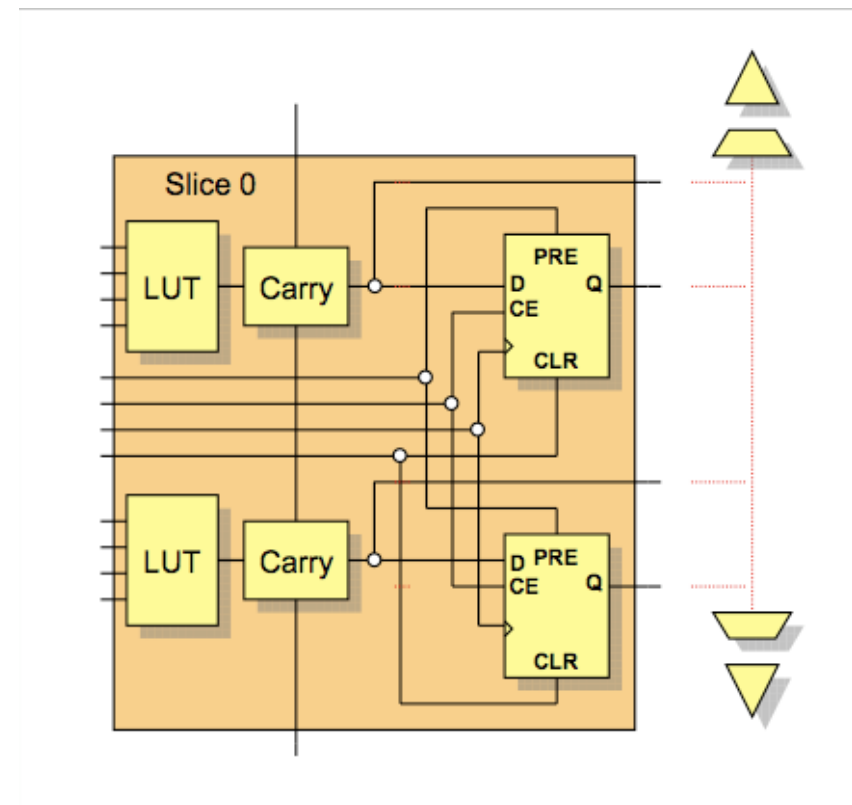
Arquitectura de las FPGAs de Xilinx

Arquitectura Virtex II

❑ Bloques Lógicos Configurables (CLBs)

❑ Cada Slice contiene:

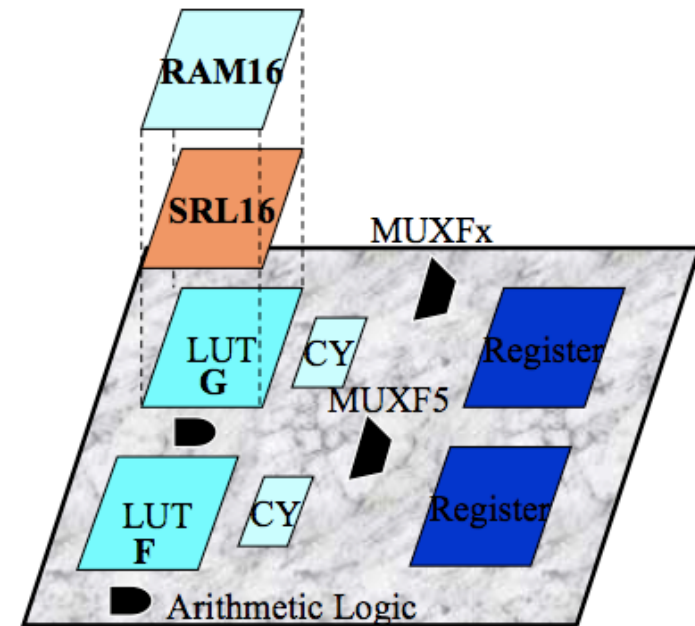
- ❑ 2 generadores de funciones de 4 entradas.
- ❑ 2 biestables que pueden activarse por flanco (flip-flop) o por nivel (latch).
- ❑ Multiplexores.
- ❑ Lógica Aritmética.
- ❑ 4 salidas (2 combinacionales y 2 secuenciales)



Arquitectura de las FPGAs de Xilinx

Arquitectura Virtex II

- ❑ Bloques Lógicos Configurables (CLBs)
 - ❑ Cada generador de funciones permite realizar:
 - ❑ 1 tabla de consulta (LUT) de 4 entradas.
 - ❑ 1 memoria RAM de 16 bits.
 - ❑ 1 registro de desplazamiento de 16 bits.



Arquitectura de las FPGAs de Xilinx

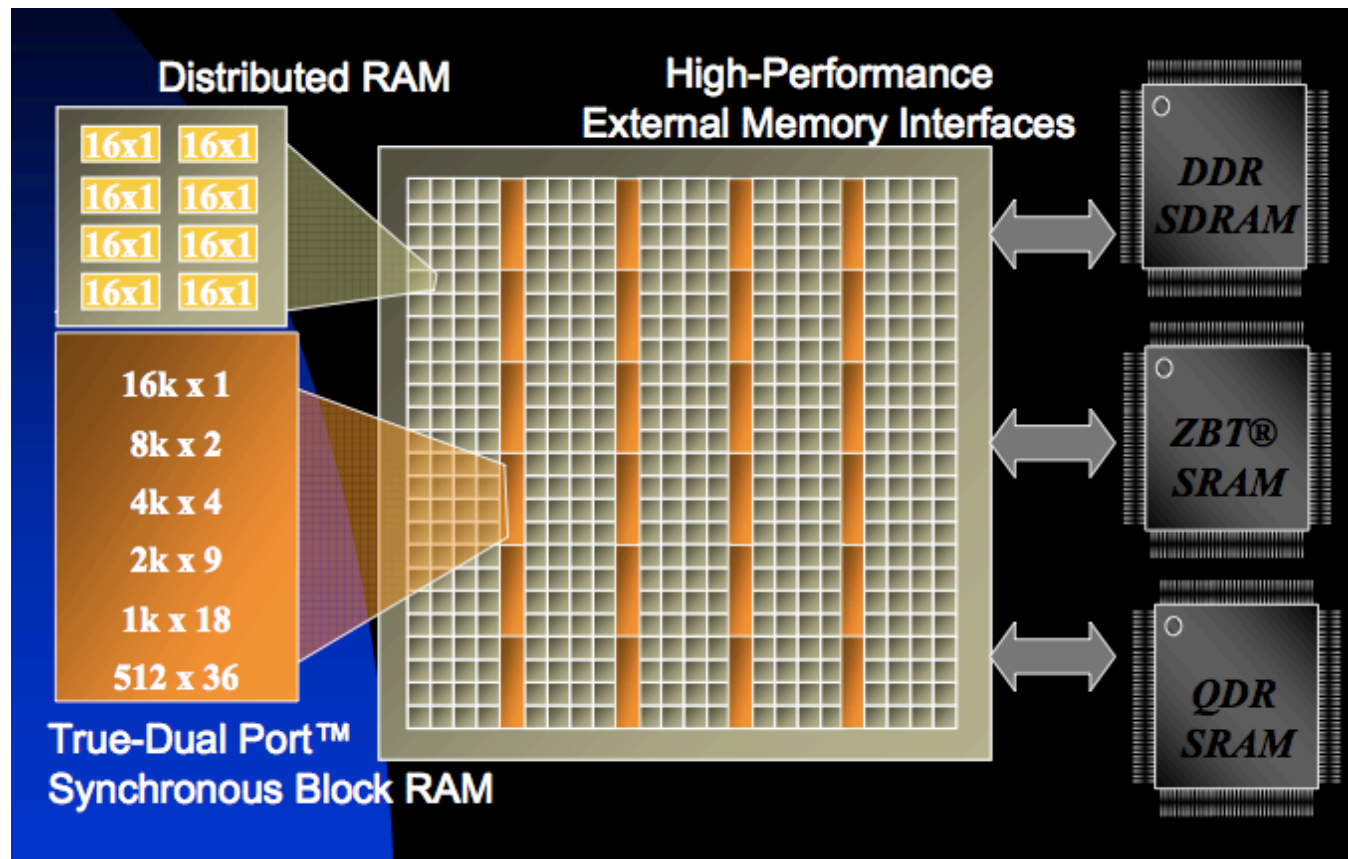
Arquitectura Virtex II

- ❑ Memorias en una Virtex II
 - ❑ Tenemos 3 tipos de memorias que podemos usar con una Virtex II.
 - ❑ Block Select RAM: Memoria interna de la FPGA y dedicada.
 - ❑ Memorias distribuidas: Memorias internas basadas en LUTs.
 - ❑ Memorias externas: Cualquier circuito externo que deseemos conectar.
 - ❑ El tipo de memoria que debemos usar depende de la capacidad que se necesite:
 - ❑ Memoria distribuida: menor de 1 Kbyte.
 - ❑ BRAM: menor de 10 Kbytes.
 - ❑ Memoria externa: mayor 10 Kbytes.
-

Arquitectura de las FPGAs de Xilinx

Arquitectura Virtex II

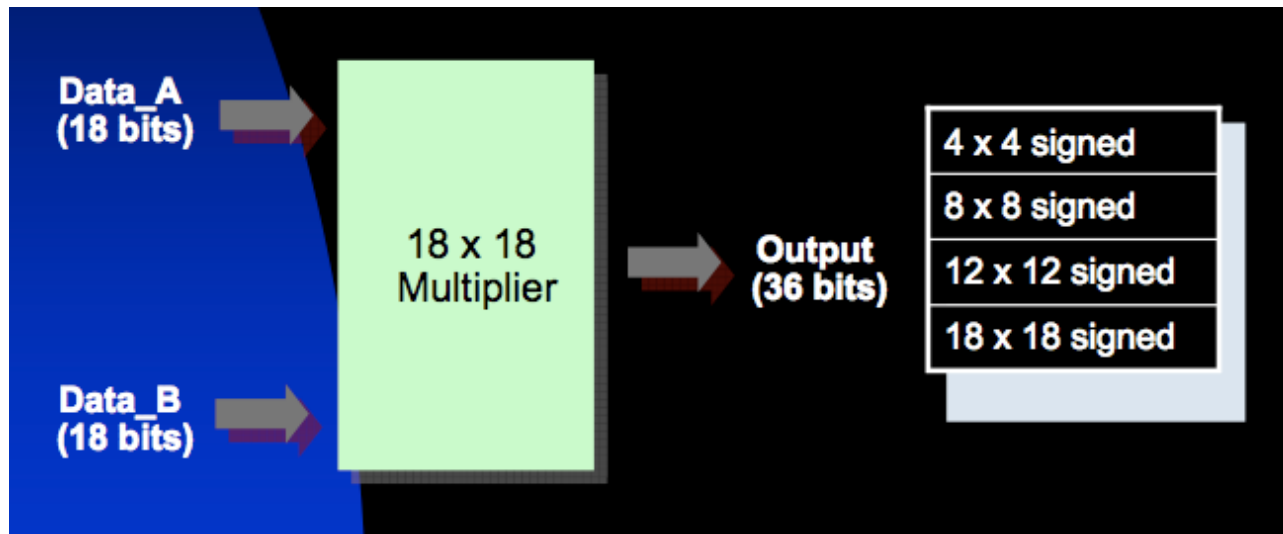
- Memorias en una Virtex II



Arquitectura de las FPGAs de Xilinx

Arquitectura Virtex II

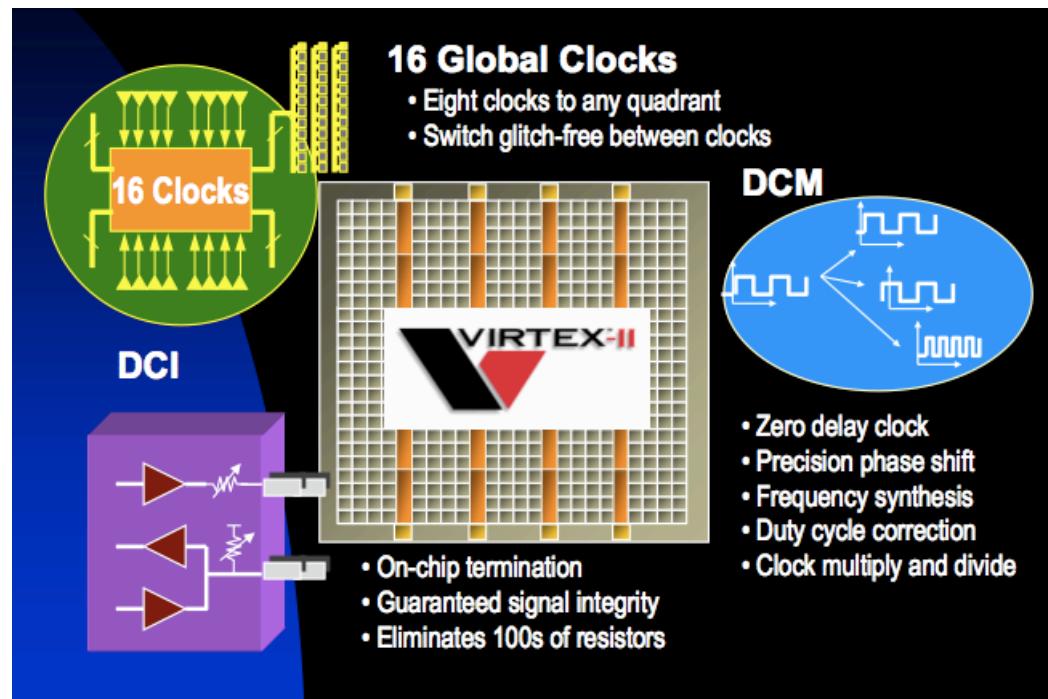
- ❑ Multiplicadores Hardware en una Virtex II
 - ❑ Multiplicadores dedicados de 18 bits x 18 bits.
 - ❑ Permiten implementar funciones MAC.
 - ❑ Permiten operaciones con signo en complemento a 2.
 - ❑ Cercanos a las memorias BRAM.



Arquitectura de las FPGAs de Xilinx

Arquitectura Virtex II

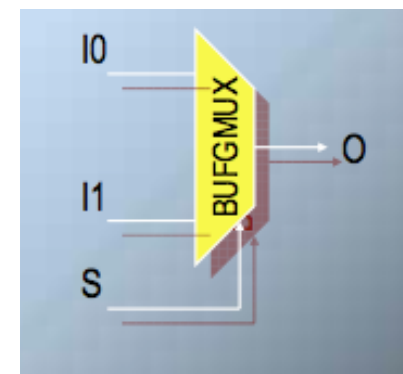
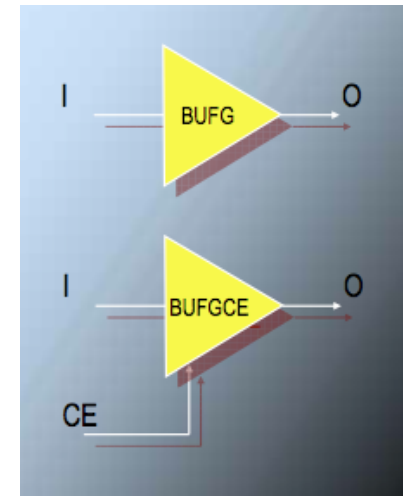
- ❑ Circuitos de reloj en una Virtex II
 - ❑ Cada FPGA tiene 16 multiplexores globales de reloj (BUFGMUX).
 - ❑ La señal de reloj puede proceder de un terminal de entrada de la FPGA, de un circuito gestor de reloj (DCM) o de una línea de interconexión local.



Arquitectura de las FPGAs de Xilinx

Arquitectura Virtex II

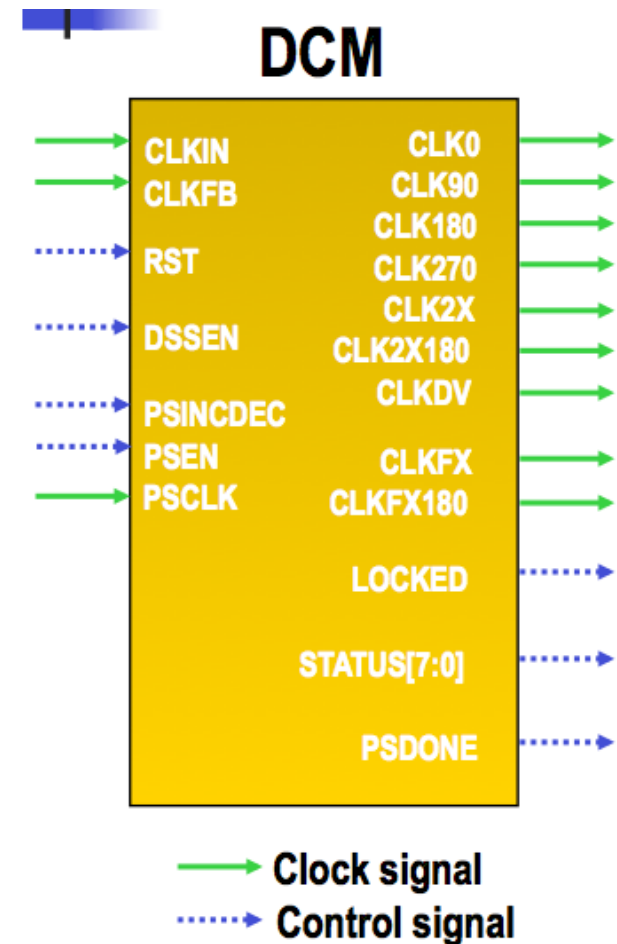
- ❑ Circuitos de reloj en una Virtex II
 - ❑ Los BUFGMUX permiten:
 - ❑ Distribuir la señal de reloj por toda la FPGA con mínimo desfase.
 - ❑ Habilitación de reloj.
 - ❑ Conmutación entre diferentes señales de reloj sin desfase.
 - ❑ Podemos usar hasta 8 señales de reloj en cada región de la FPGA.



Arquitectura de las FPGAs de Xilinx

Arquitectura Virtex II

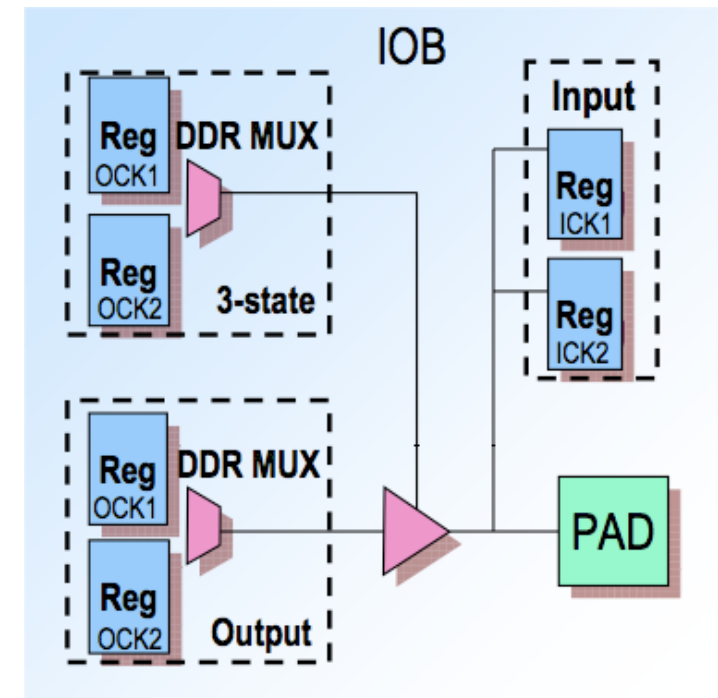
- ❑ Circuitos de reloj en una Virtex II
 - ❑ Los DCMs se caracterizan por:
 - ❑ Permitir el desfase de la señal de reloj.
 - ❑ Amplificar la misma.
 - ❑ Engancharse en fase y frecuencia.
 - ❑ Hasta 12 DCMs en una FPGA.
 - ❑ Su entrada procede de terminales de entrada de la FPGA.



Arquitectura de las FPGAs de Xilinx

Arquitectura Virtex II

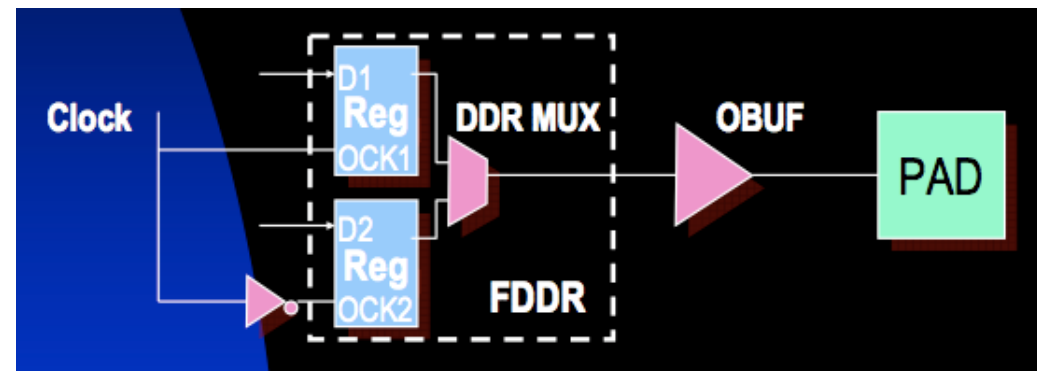
- ❑ Bloques de entrada/salida en una Virtex II
 - ❑ Los IOBs presentan:
 - ❑ 2 biestables DDR en la entrada.
 - ❑ 4 biestables DDR en la salida.
 - ❑ Relojes y habilitación separados para entrada y salida.
 - ❑ Señales de puesta a uno y a cero compartidas.



Arquitectura de las FPGAs de Xilinx

Arquitectura Virtex II

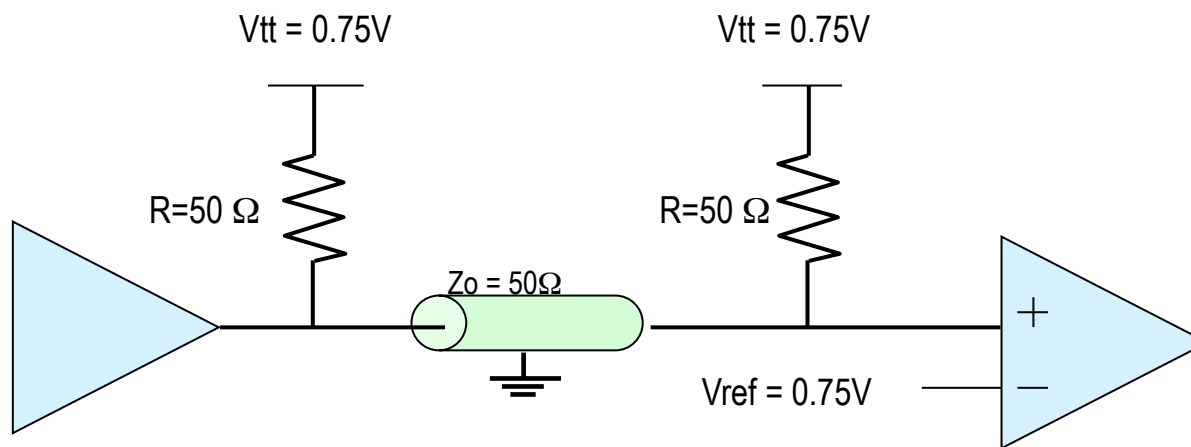
- ❑ Bloques de entrada/salida en una Virtex II
 - ❑ Los IOBs permiten:
 - ❑ Conexión directa de señales externas de diferentes tensiones.
 - ❑ Optimizan la relación velocidad/ruido.
 - ❑ Soportan los estándares:
 - ❑ LVTTTL, LVCMOS.
 - ❑ PCI-X.
 - ❑ GTL, GTPL.
 - ❑ LVDS, BLVDS, ULVDS.
 - ❑ LDT.
 - ❑ LVPECL.



Arquitectura de las FPGAs de Xilinx

Arquitectura Virtex II

- ❑ Bloques de entrada/salida en una Virtex II
 - ❑ Los IOBs contienen un Control Digital de Impedancia (DCI) que permite:
 - ❑ Salidas cuya impedancia se adapta a las pistas de la PCB.
 - ❑ Terminaciones para receptores y transmisores en la propia FPGA.
 - ❑ Mejora la integridad de las señales eliminando reflexiones.
 - ❑ Reduce la complejidad del rutado de la FPGA.



Arquitectura de las FPGAs de Xilinx

Arquitectura Virtex II

Modelos de la familia Virtex II

Virtex-II Part Number	XC2V 40	XC2V 80	XC2V 250	XC2V 500	XC2V 1000	XC2V 1500	XC2V 2000	XC2V 3000	XC2V 4000	XC2V 6000	XC2V 8000
LUTs + FFs	512	1,024	3,072	6,144	10,24	15,36	21,50	28,67	46,08	67,58	93,18
					0	0	4	2	0	4	4
BRAM (kb)	72	144	432	576	720	864	1,008	1,728	2,160	2,592	3,024
Multipliers	4	8	24	32	40	48	56	96	120	144	168
DCM Units	4	4	8	8	8	8	8	12	12	12	12
Package	Available SelectIO										
CS144	88	92	92								
FG256	88	120	172	172	172						
FG456			200	264	324						
FG676						392	456	484			
FF896					432	528	624				
FF1152								720	824	824	824
FF1517									912	1,104	1,108
BG575					328	392	408				
BG728							456	516			
BG957							624	684	684	684	684

Arquitectura de las FPGAs de Xilinx

Virtex II Pro frente a Virtex II

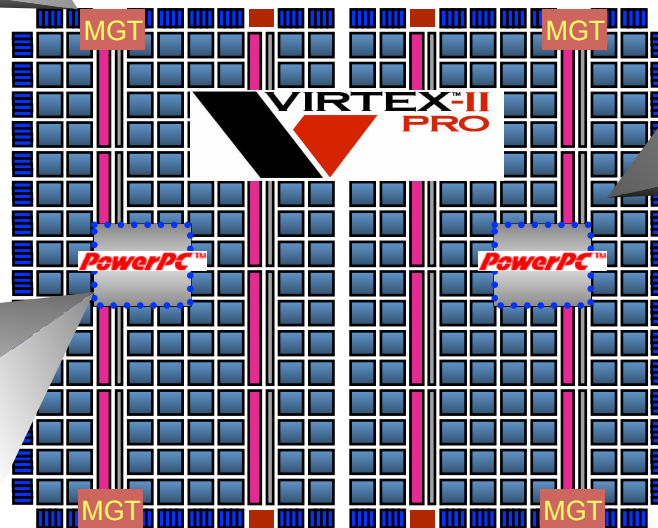
Novedades Virtex II Pro

RocketIO

3.125 Gbps Multi-Gigabit Transceivers (MGTs)
Supports 10 Gbps standards
Up to 24 per device

PowerPC™

PowerPC 405 Core
300MHz / 450DMIPS
Up to 4 per device



VIRTEX™ II
Fabric

Arquitectura de las FPGAs de Xilinx

Virtex II Pro frente a Virtex II

- ❑ Novedades Virtex II Pro
 - ❑ Bloques RocketIO Multi-Gigabit Transceivers (MGT):
 - ❑ Serializa-Deserializa (SERDES) señales de muy alta tasa de datos (2,488 Gbps hasta 10,312 Gbps) provenientes de fibra óptica, Gigabit Ethernet...
 - ❑ Permite trabajar con buses de datos seleccionables entre 8, 16 y 32 bits.
 - ❑ Incluía hasta 24 transceivers en la FPGA de mayor gama.
 - ❑ Procesador PowerPC:
 - ❑ Incluye por primera vez en una FPGA un Procesador hardware.
 - ❑ Existían FPGAs con hasta 4 PowerPCs a 300MHz.
 - ❑ Más multiplicadores hardware.
 - ❑ Más terminales I/O.
 - ❑ Más memoria interna.
-

Arquitectura de las FPGAs de Xilinx

Virtex II Pro frente a Virtex II

Novedades Virtex II Pro

Table 1: Virtex-II Pro / Virtex-II Pro X FPGA Family Members

Device ⁽¹⁾	RocketIO Transceiver Blocks	PowerPC Processor Blocks	Logic Cells ⁽²⁾	CLB (1 = 4 slices = max 128 bits)		18 X 18 Bit Multiplier Blocks	Block SelectRAM+		DCMs	Maximum User I/O Pads
				Slices	Max Distr RAM (Kb)		18 Kb Blocks	Max Block RAM (Kb)		
XC2VP2	4	0	3,168	1,408	44	12	12	216	4	204
XC2VP4	4	1	6,768	3,008	94	28	28	504	4	348
XC2VP7	8	1	11,088	4,928	154	44	44	792	4	396
XC2VP20	8	2	20,880	9,280	290	88	88	1,584	8	564
XC2VPX20	8 ⁽⁴⁾	1	22,032	9,792	306	88	88	1,584	8	552
XC2VP30	8	2	30,816	13,696	428	136	136	2,448	8	644
XC2VP40	0 ⁽³⁾ , 8, or 12	2	43,632	19,392	606	192	192	3,456	8	804
XC2VP50	0 ⁽³⁾ or 16	2	53,136	23,616	738	232	232	4,176	8	852
XC2VP70	16 or 20	2	74,448	33,088	1,034	328	328	5,904	8	996
XC2VPX70	20 ⁽⁴⁾	2	74,448	33,088	1,034	308	308	5,544	8	992
XC2VP100	0 ⁽³⁾ or 20	2	99,216	44,096	1,378	444	444	7,992	12	1,164

Arquitectura de las FPGAs de Xilinx

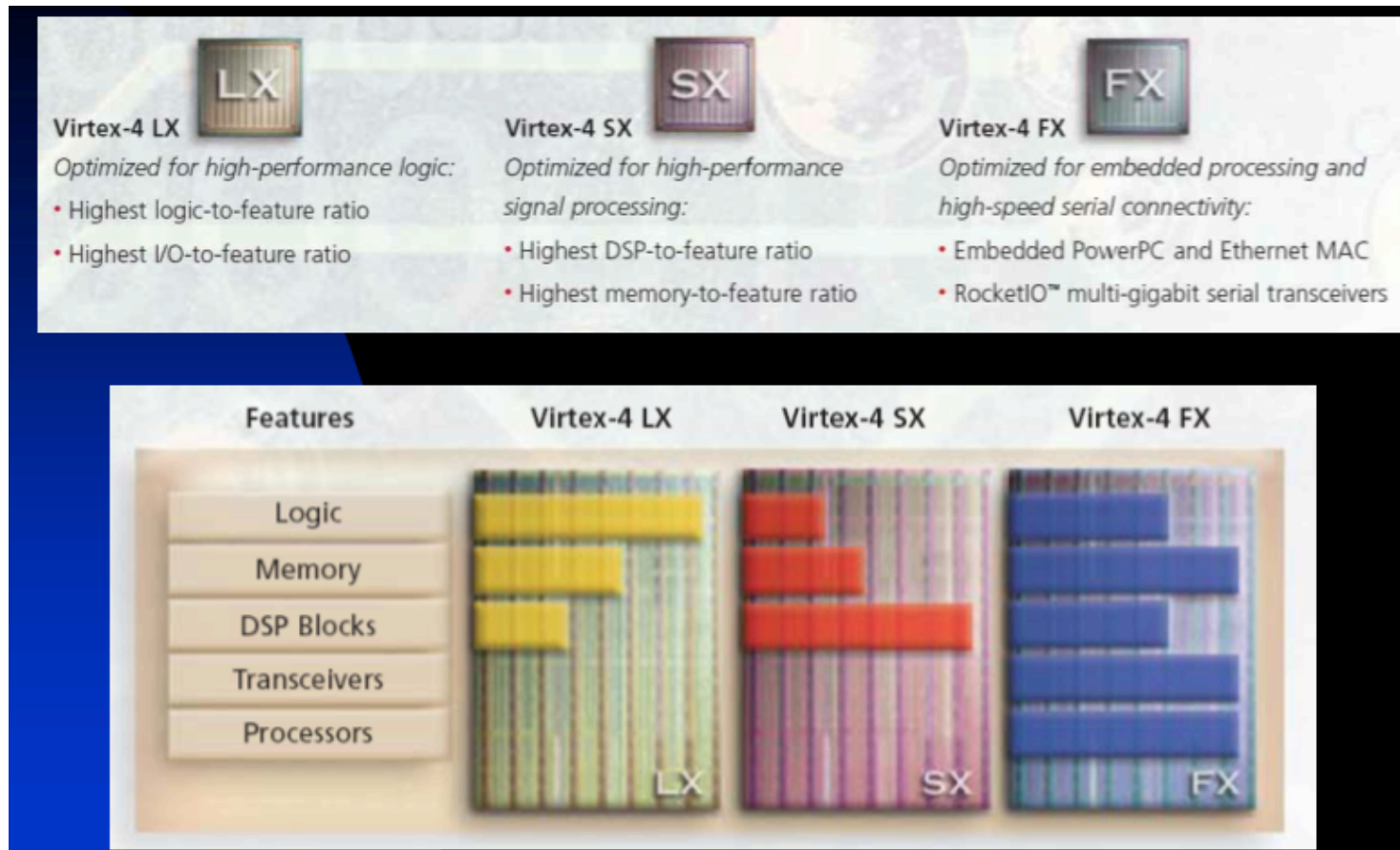
Virtex 4-5 frente a Virtex II

- ❑ Bloques dedicados a DSP.
 - ❑ Divisores de reloj (PMCD).
 - ❑ Conversores series/paralelo y paralelo/serie incorporados en los terminales de I/O.
 - ❑ Ethernet MACs (familia FX)
 - ❑ Las memorias BRAM pueden ser configuradas como FIFOs.
 - ❑ Redes de distribución de reloj más avanzadas (Regional Clock Buffers)
 - ❑ Transceptores Multi-Gigabit hasta 11,1 Gbps.
 - ❑ Monitorización de temperatura y tensión (Virtex 5)
 - ❑ Menor consumo.
-

Arquitectura de las FPGAs de Xilinx

Virtex 4-5 frente a Virtex II

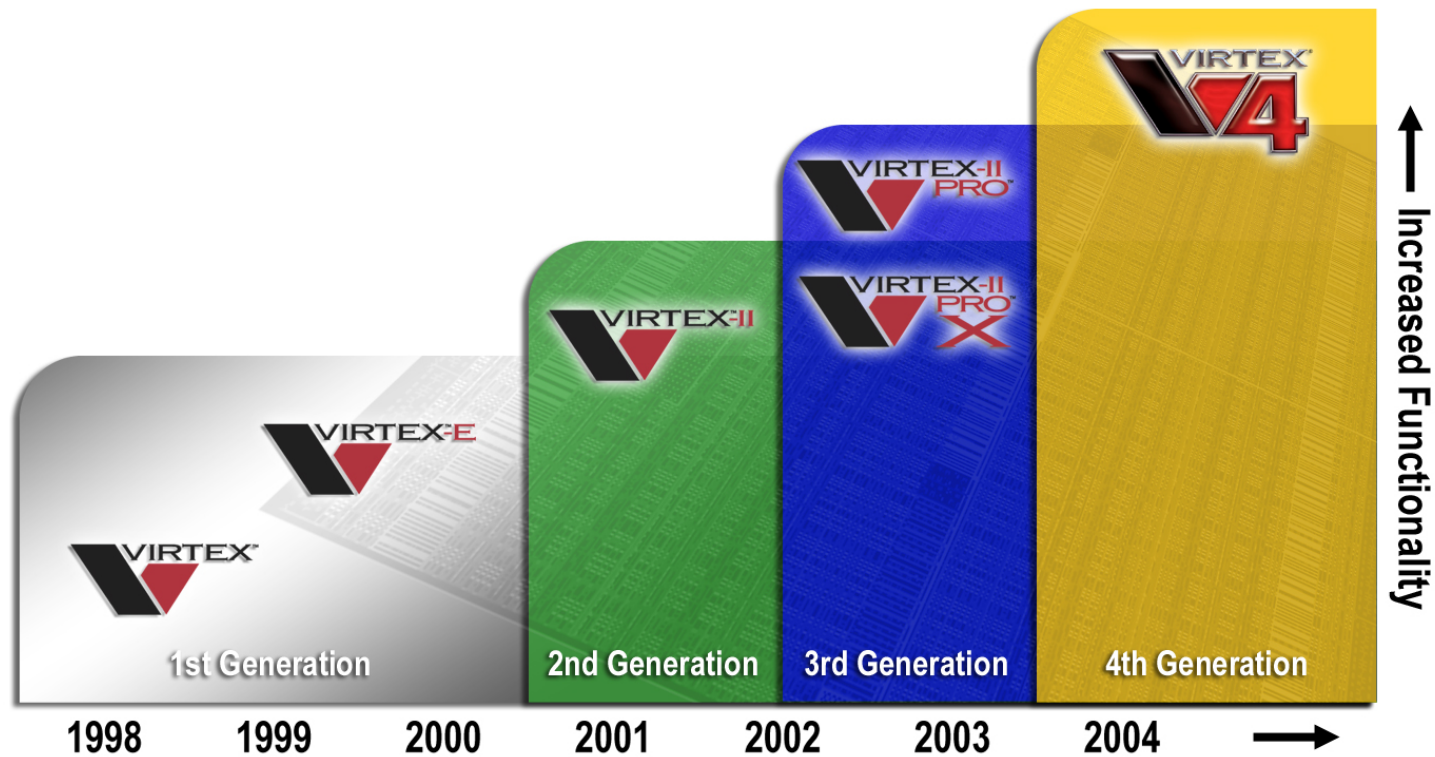
- Modelos de la familia Virtex 4.



Arquitectura de las FPGAs de Xilinx

Virtex 4-5 frente a Virtex II

- Modelos de la familia Virtex 4.



Arquitectura de las FPGAs de Xilinx

Virtex 4-5 frente a Virtex II

■ Incorporación bloques DSP

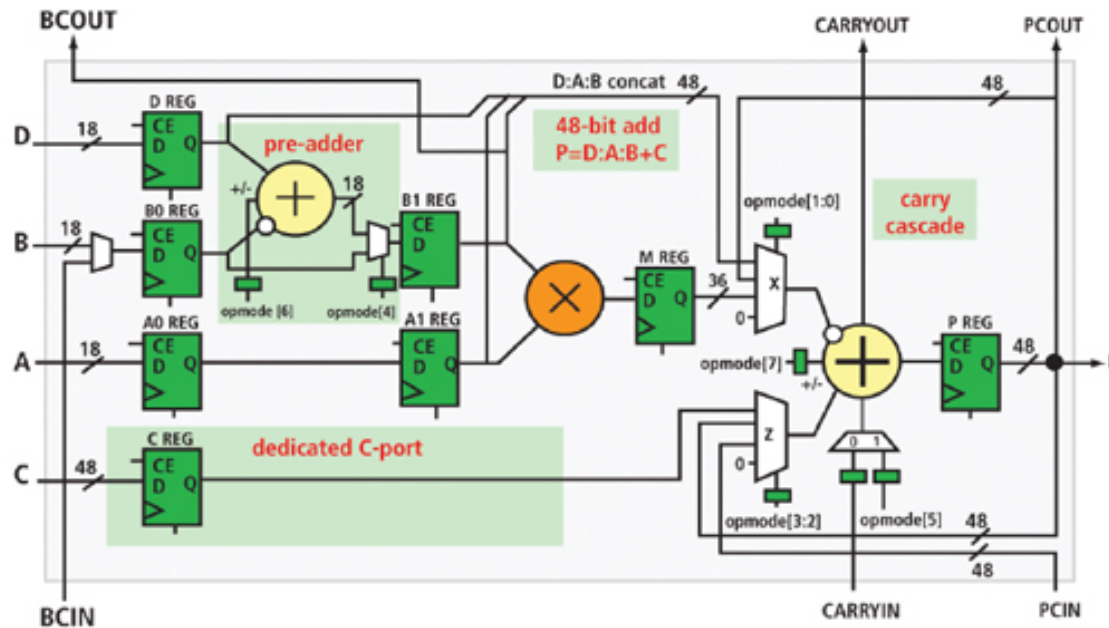


Figure 1 – The pre-adder saves 9 slices for every XtremeDSP48A used in a FIR filter design.

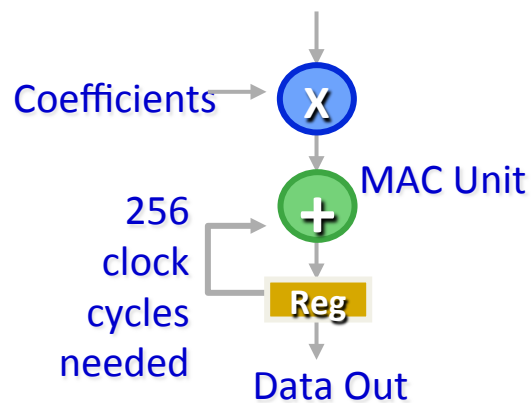
OpMode	Z			Y			X			Output
	6	5	4	3	2	1	0	6	5	
Zero	0	0	0	0	0	0	0	+/- Cin		
Hold P	0	0	0	0	0	1	0	+/- (P + Cin)		
A:B Select	0	0	0	0	0	1	1	+/- (A:B + Cin)		
Multiply	0	0	0	1	0	1	0	+/- (A * B + Cin)		
C Select	0	0	0	1	1	0	0	+/- (C + Cin)		
Feedback Add	0	0	0	1	1	1	0	+/- (C + P + Cin)		
36-Bit Adder	0	0	0	1	1	1	1	+/- (A:B + C + Cin)		
P Cascade Select	0	0	1	0	0	0	0	PCIN +/- Cin		
P Cascade Feedback Add	0	0	1	0	0	1	0	PCIN +/- (P + Cin)		
P Cascade Add	0	0	1	0	0	1	1	PCIN +/- (A:B + Cin)		
P Cascade Multiply Add	0	0	1	0	1	0	1	PCIN +/- (A * B + Cin)		
P Cascade Add	0	0	1	1	1	0	0	PCIN +/- (C + Cin)		
P Cascade Feedback Add Ad	0	0	1	1	1	1	0	PCIN +/- (C + P + Cin)		
P Cascade Add Add	0	0	1	1	1	1	1	PCIN +/- (A:B + C + Cin)		
Hold P	0	1	0	0	0	0	0	P +/- Cin		
Double Feedback Add	0	1	0	0	0	1	0	P +/- (P + Cin)		
Feedback Add	0	1	0	0	0	1	1	P +/- (A:B + Cin)		
Multiply-Accumulate	0	1	0	1	0	1	1	P +/- (A * B + Cin)		
Feedback Add	0	1	0	1	1	0	0	P +/- (C + Cin)		
Double Feedback Add	0	1	0	1	1	1	0	P +/- (C + P + Cin)		
Feedback Add Add	0	1	0	1	1	1	1	P +/- (A:B + C + Cin)		
C Select	0	1	1	0	0	0	0	C +/- Cin		
Feedback Add	0	1	1	0	0	1	0	C +/- (P + Cin)		
36-Bit Adder	0	1	1	0	0	1	1	C +/- (A:B + Cin)		
Multiply-Add	0	1	1	0	1	0	1	C +/- (A * B + Cin)		
Double	0	1	1	1	1	0	0	C +/- (C + Cin)		
Double Add Feedback Add	0	1	1	1	1	1	0	C +/- (C + P + Cin)		
Double Add	0	1	1	1	1	1	1	C +/- (A:B + C + Cin)		

Arquitectura de las FPGAs de Xilinx

Virtex 4-5 frente a Virtex II

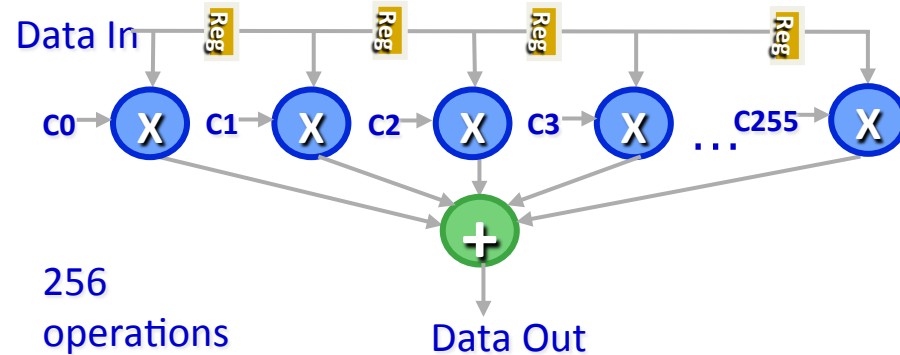
- ❑ Incorporación bloques DSP

Programmable DSP - Sequential



$$\frac{1 \text{ GHz}}{256 \text{ clock cycles}} = 4 \text{ MSPS}$$

FPGA - Fully Parallel Implementation



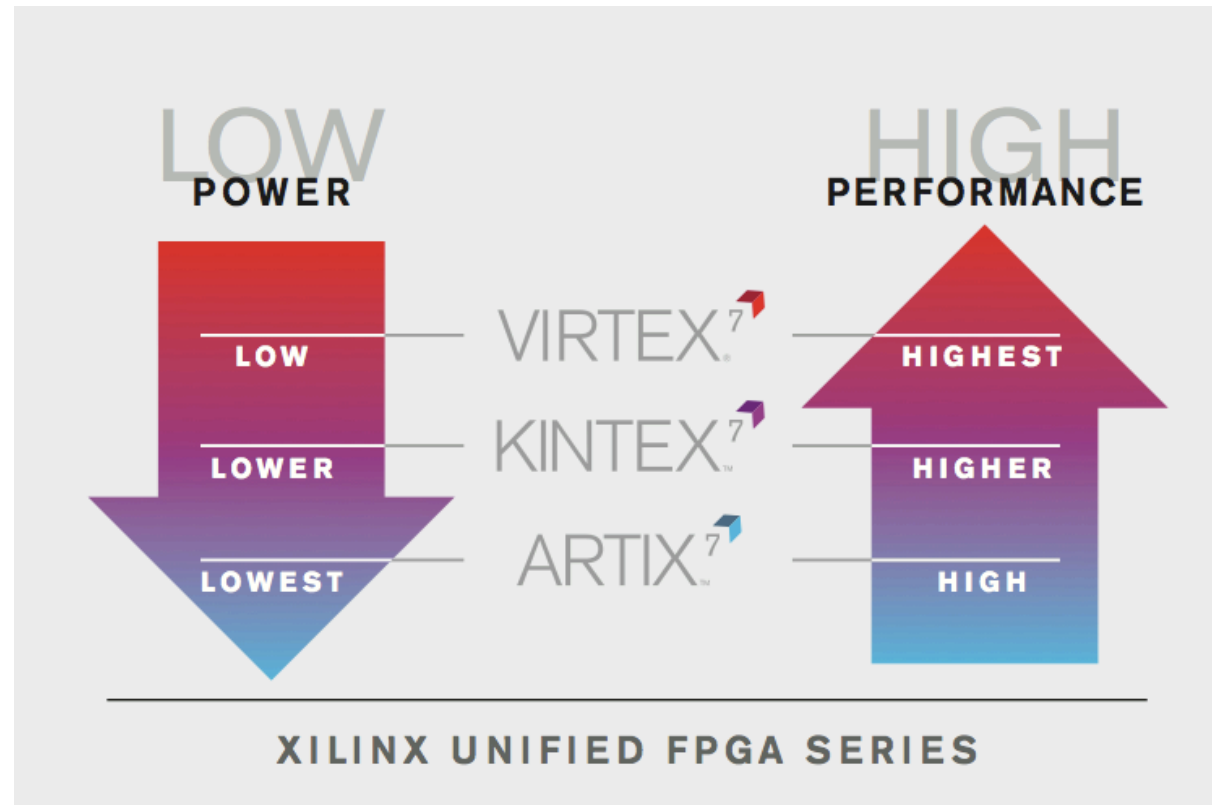
256 operations in 1 clock cycle

$$\frac{500 \text{ MHz}}{1 \text{ clock cycle}} = 500 \text{ MSPS}$$

Arquitectura de las FPGAs de Xilinx

Actualidad y novedades

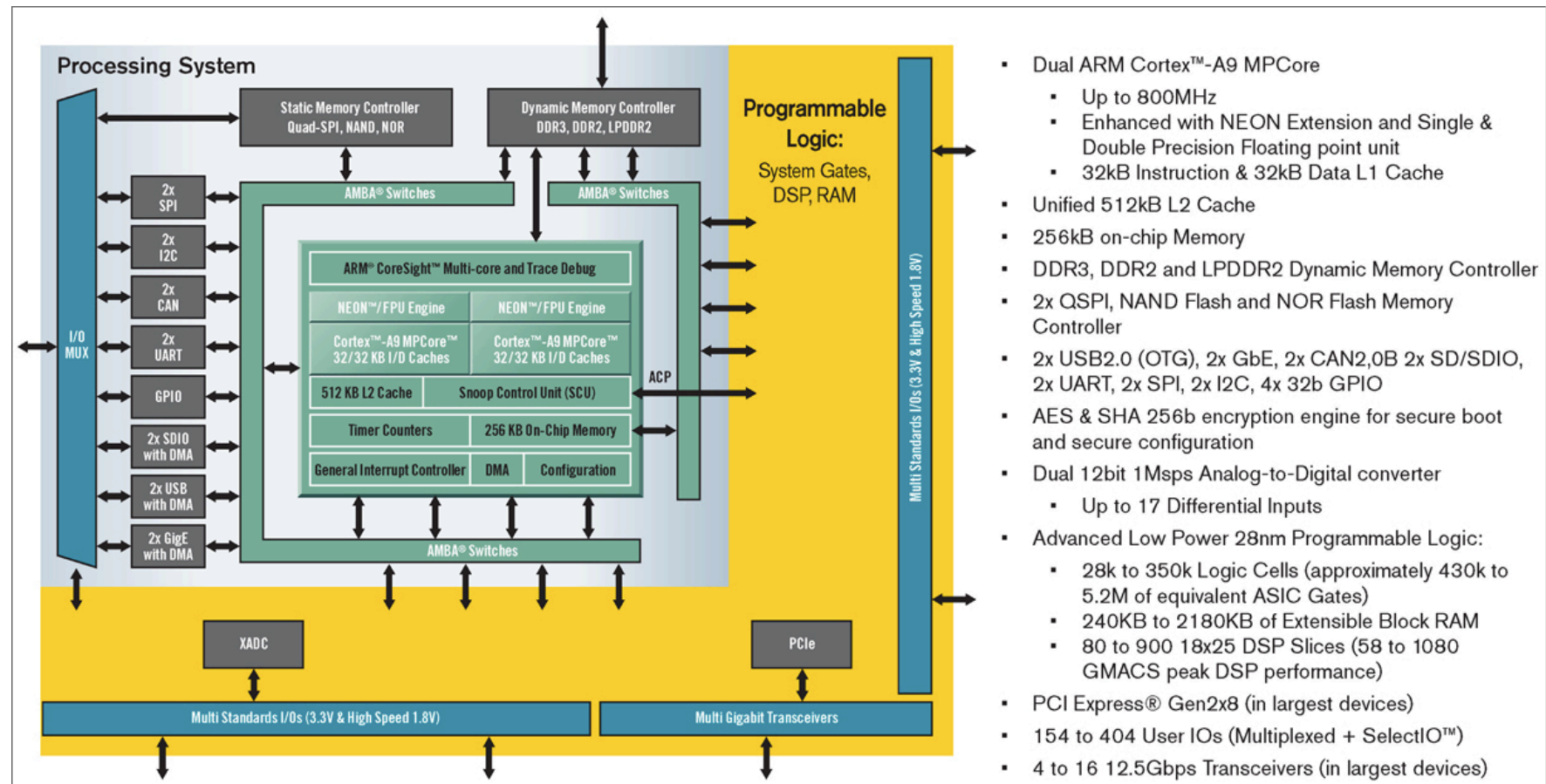
- Nuevas familias orientadas a bajo consumo o altas prestaciones. Desaparece el concepto familia de alto coste (Virtex) y familia de bajo coste (Spartan)



Arquitectura de las FPGAs de Xilinx

Actualidad y novedades

□ Aparece un nuevo concepto en Xilinx (SoCs)

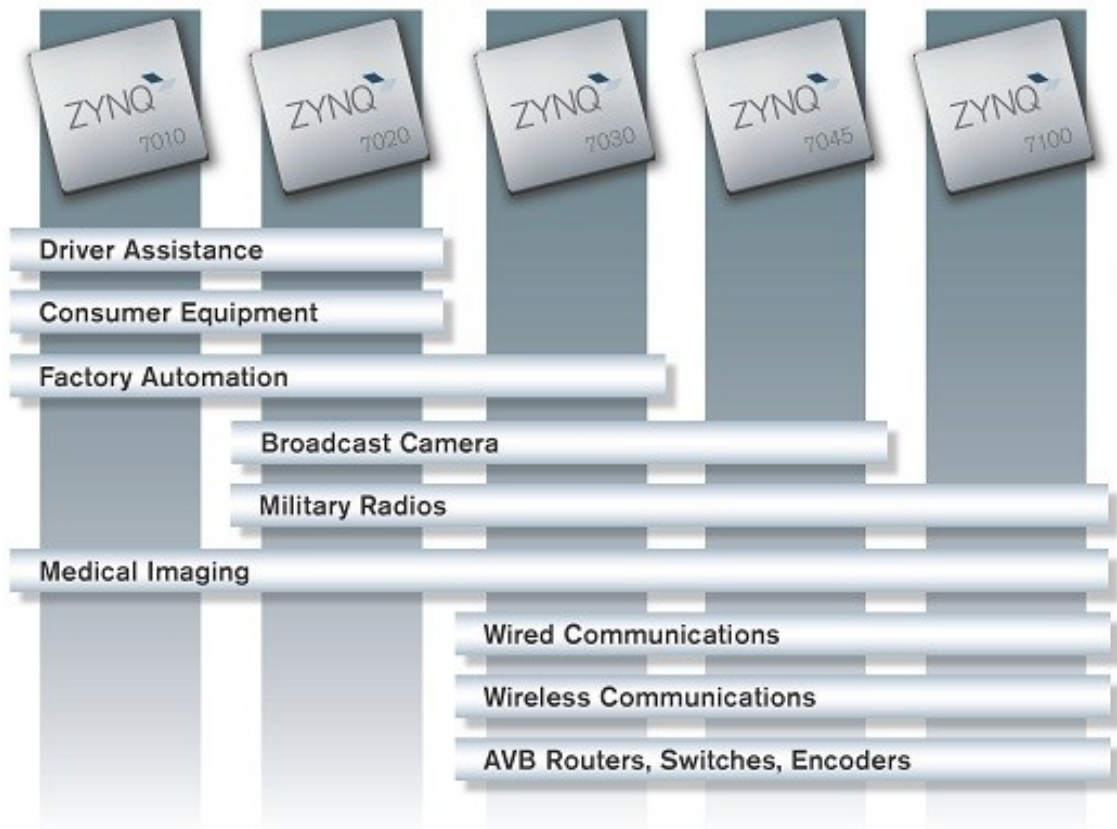


- Dual ARM Cortex™-A9 MPCore
 - Up to 800MHz
 - Enhanced with NEON Extension and Single & Double Precision Floating point unit
 - 32kB Instruction & 32kB Data L1 Cache
- Unified 512kB L2 Cache
- 256kB on-chip Memory
- DDR3, DDR2 and LPDDR2 Dynamic Memory Controller
- 2x QSPI, NAND Flash and NOR Flash Memory Controller
- 2x USB2.0 (OTG), 2x GbE, 2x CAN2.0B, 2x SD/SDIO, 2x UART, 2x SPI, 2x I2C, 4x 32b GPIO
- AES & SHA 256b encryption engine for secure boot and secure configuration
- Dual 12bit 1MSPs Analog-to-Digital converter
 - Up to 17 Differential Inputs
- Advanced Low Power 28nm Programmable Logic:
 - 28k to 350k Logic Cells (approximately 430k to 5.2M of equivalent ASIC Gates)
 - 240KB to 2180KB of Extensible Block RAM
 - 80 to 900 18x25 DSP Slices (58 to 1080 GMACS peak DSP performance)
- PCI Express® Gen2x8 (in largest devices)
- 154 to 404 User IOs (Multiplexed + SelectIO™)
- 4 to 16 12.5Gbps Transceivers (in largest devices)

Arquitectura de las FPGAs de Xilinx

Actualidad y novedades

□ <http://www.youtube.com/watch?v=X6MnmQ9iGEM>

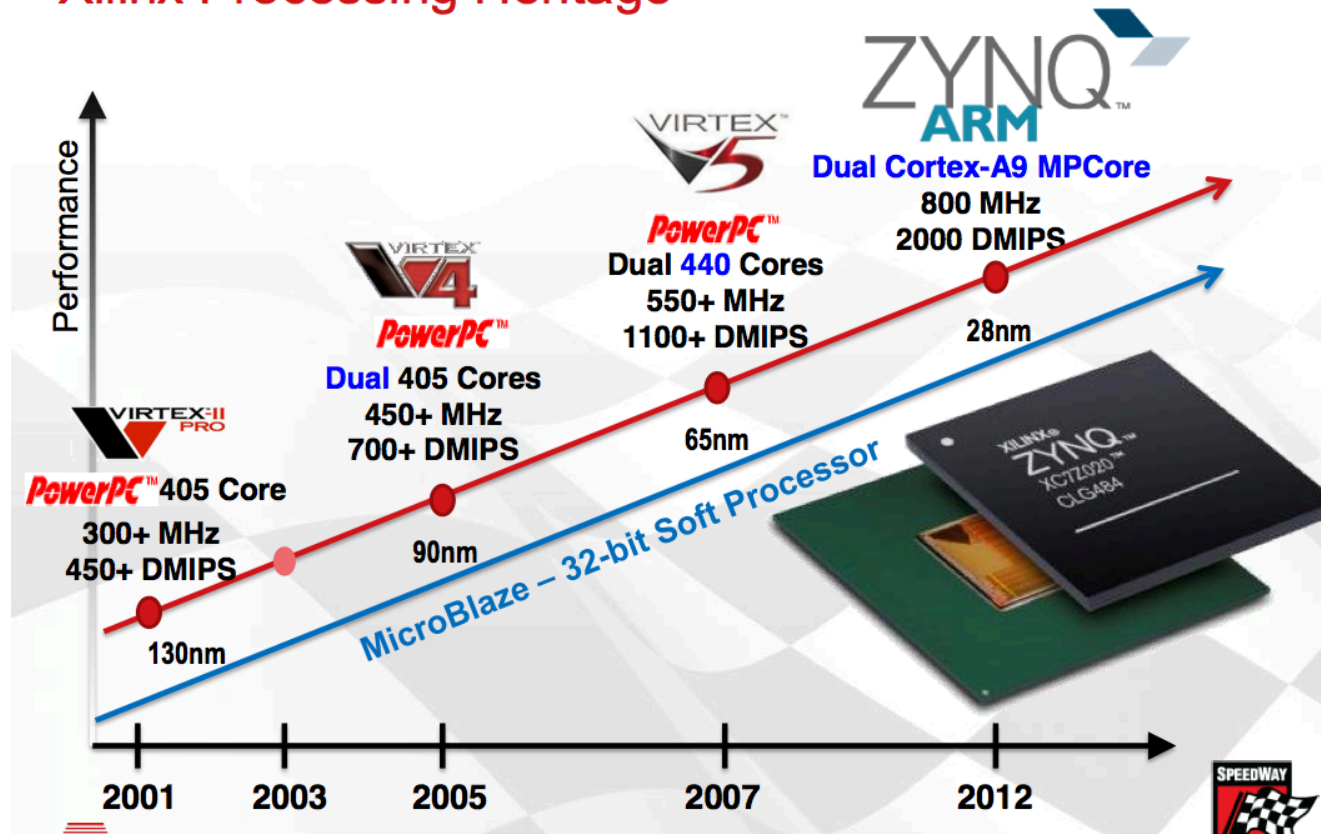


Arquitectura de las FPGAs de Xilinx

Actualidad y novedades

□ Evolución

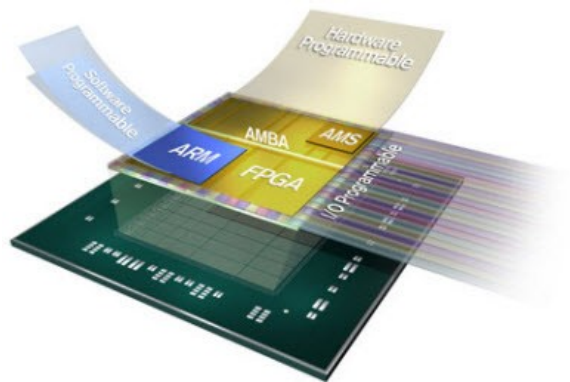
Xilinx Processing Heritage



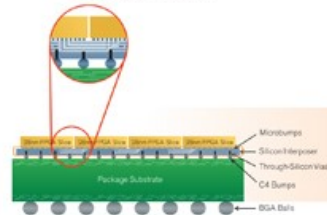
Arquitectura de las FPGAs de Xilinx

Actualidad y novedades

- Aparece una nueva tecnología en Xilinx (3D IC)

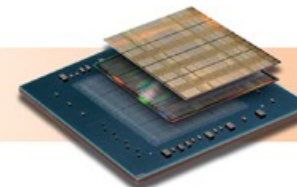


Stacked Silicon Interconnect



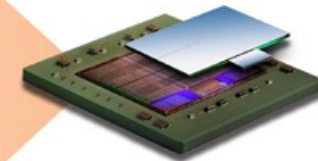
Stacked Silicon Interconnect provides massive quantities of interconnect enabling the highest bandwidth, lowest latency, and power

The Industry's First All Programmable 3D IC

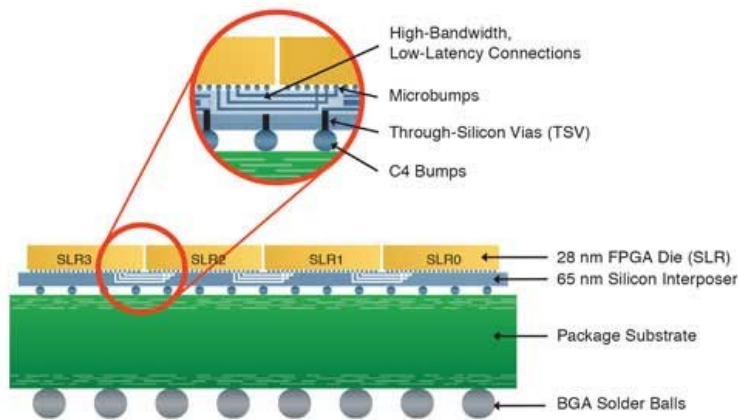


With 2M Logic Cells Delivering Breakthrough Capacity, Bandwidth and Power Efficiency

The Industry's First Heterogeneous 3D IC

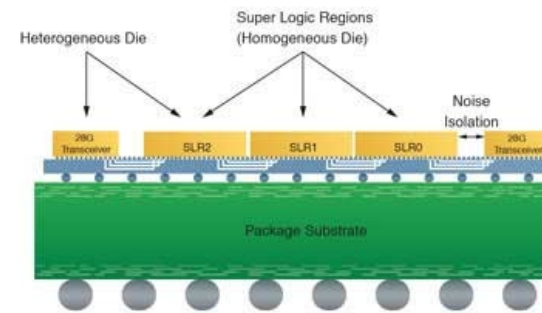


Highest Bandwidth with 278 Tb/s serial connectivity, electrically isolated 28Gbps for optimal signal integrity



Virtex®-7 2000T FPGA Enabled by SSI Technology

WP380_01_112812



Heterogeneous 3D FPGA with Integrated 28G Transceivers

WP380_06_112912

Arquitectura de las FPGAs de Xilinx

Actualidad y novedades

□ Nuevo panorama en FPGAs-SoCs-3D ICs...

Portfolio: All Programmable FPGAs, SoCs, and 3D ICs Available Today

All Programmable FPGA Family

- First shipped: Q1, 2011
- Delivering one extra node of power and performance

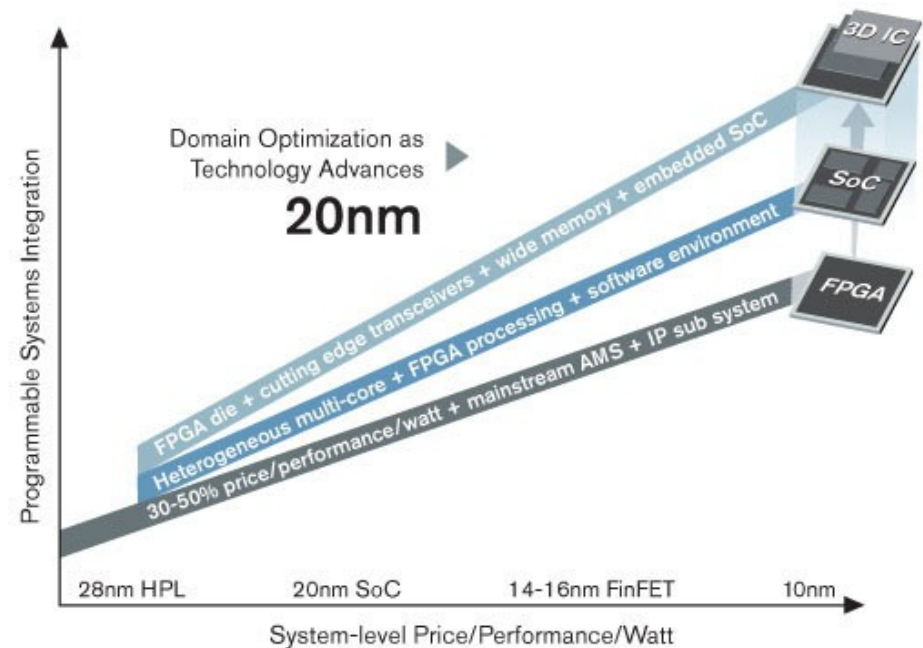


All Programmable SoC

- First shipped: Q4, 2011
- Integrating FPGA, CPU, DSP, AMS
- Competitor silicon not available

All Programmable 3D ICs

- First shipped: Q3, 2011
- Integrating 2x logic and SerDes bandwidth
- Competitor working only on test chips



Arquitectura de las FPGAs de Xilinx

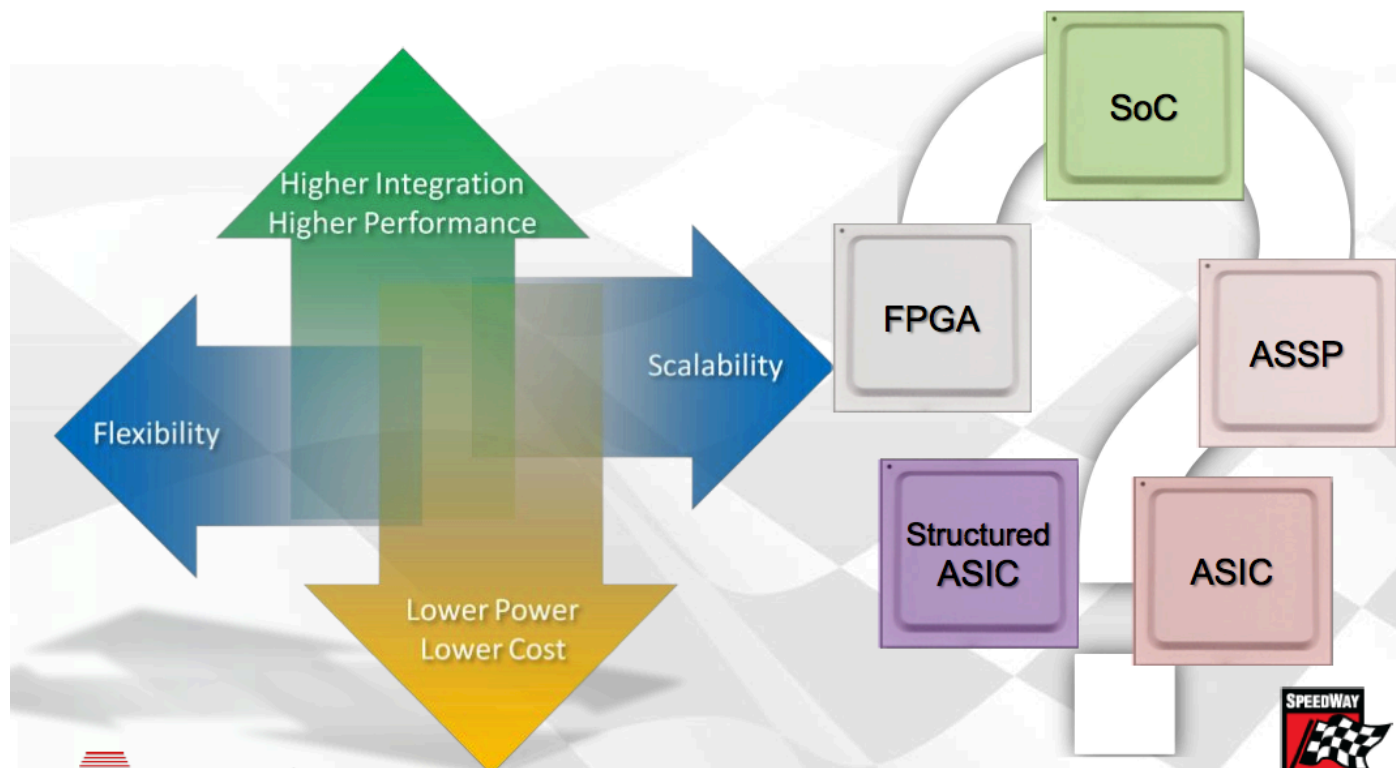
Normas de selección

- ❑ La aparición de una gran cantidad de componentes específicos en las FPGAs requiere, a la hora de su selección, de un estudio intenso del mercado. Tendremos que mirar:
 - ❑ Tecnología entrada/salida.
 - ❑ Señales de reloj.
 - ❑ Bloques DSP.
 - ❑ Microblaze o ARM.
 - ❑ RocketIO Transceivers.
 - ❑ Consumo.
 - ❑ Señales analógicas...
 - ❑ Y más aún, FPGA o Zynq (SoC) o ASIC Estructurado (3D IC)...
-

Arquitectura de las FPGAs de Xilinx

Normas de selección

- En los últimos años, se han vendido alrededor de 500 millones de microprocesadores para el mercado de los PCs y más de 10 mil millones para el mercado de los Sistemas Embebidos.



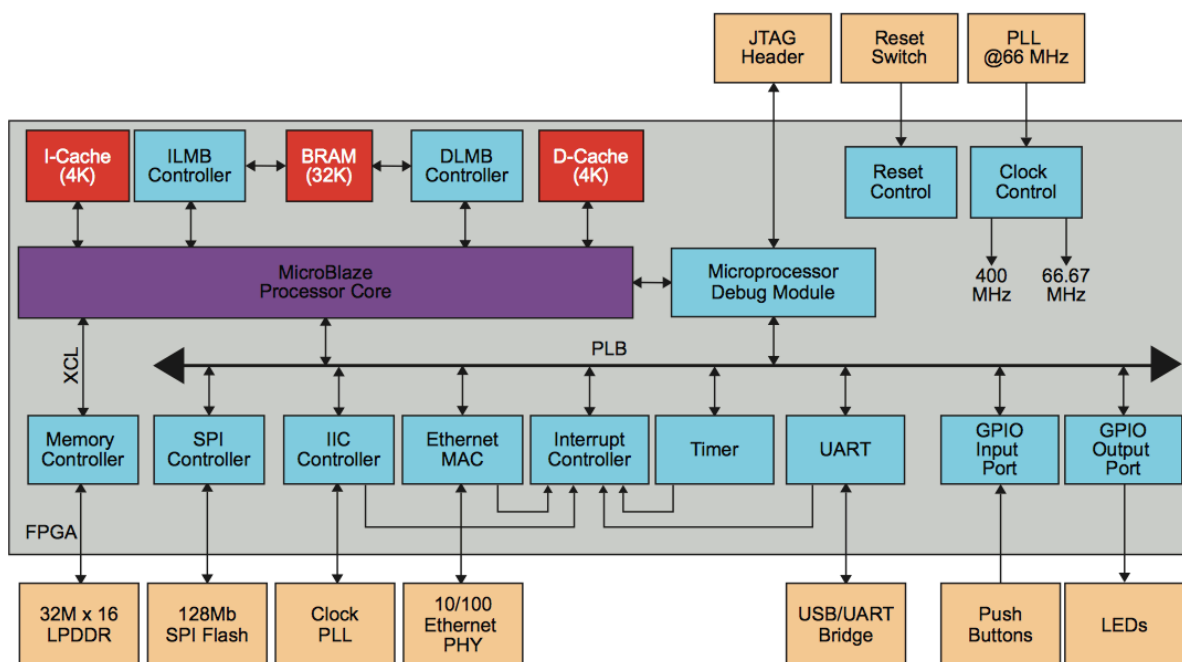
Manejo Puerto Serie

En este tercer diseño con MicroBlaze vamos a crear un sistema empotrado hardware configurando el Puerto Serie de nuestra placa.

El desarrollo de este ejercicio nos llevará a crear un diseño hardware con Microblaze utilizando de nuevo BSB pero trabajando con el puerto serie para la lectura y escritura del mismo y mostrando el valor leído en los LEDs de la placa de desarrollo LX9 Spartan-6.

Así mismo, crearemos una aplicación software para gestionar estos periféricos. También añadiremos los interruptores para trabajar con ellos.

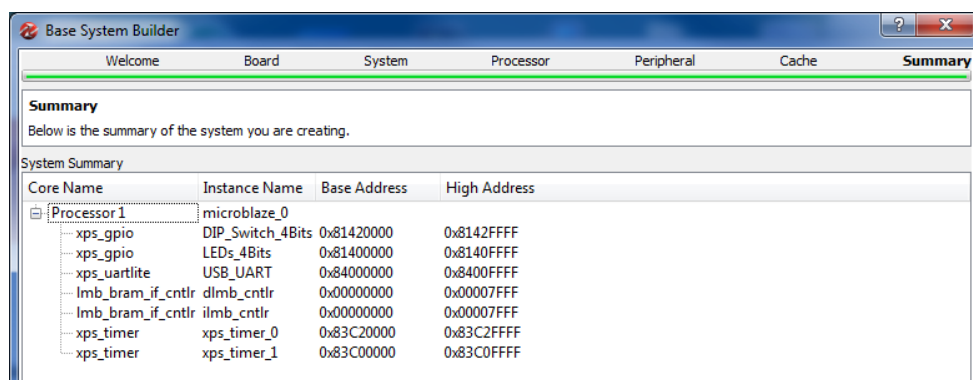
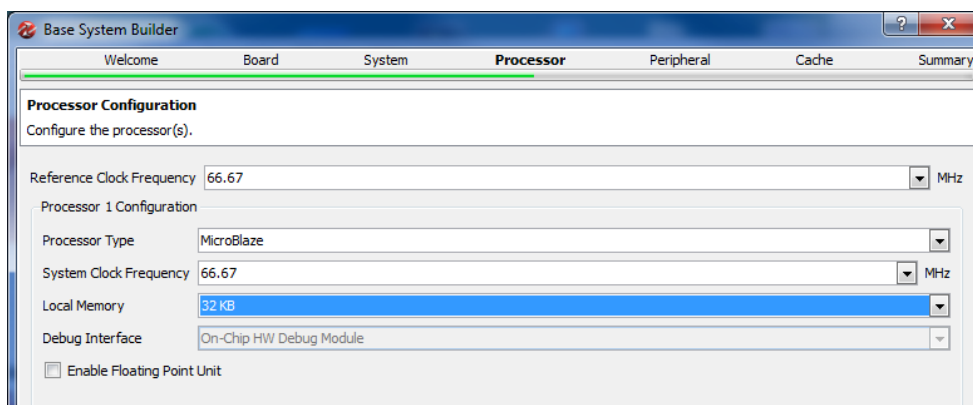
El diagrama de bloques que tendrá esta aplicación es el siguiente.



Creación sistema empotrado

Siguiendo los pasos del Ejercicio_1, creamos un nuevo proyecto con el nombre Ejercicio_3. Definiendo Microblaze como nuestro microprocesador empotrado, la misma placa de desarrollo usada anteriormente, la misma frecuencia y el mismo tamaño de memoria BRAM.

En este caso también usaremos el puerto USB_UART, los periféricos LEDs y Switches. Desactivaremos el uso de la Flash, de la DDR SDRAM y de Ethernet.



Recordar que podemos revisar los resultados de la implementación consultando el archivo system.par.

En primer lugar, generamos el bitstream desde EDK y comprobamos que el hardware generado no tiene errores.

Al igual que en los Ejercicios anteriores, abriendo SDK, vamos a añadir un nuevo proyecto software a nuestro sistema empotrado. Seguimos los mismos pasos del Ejercicio_2.

Queremos poder leer de Terminal un valor entre 0 y 8, el cual lo recibirá nuestro Microprocesador Empotrado (MicroBlaze) a través del Puerto Serie. Para ello, debemos usar la función específica del core xps_uartlite (buscarla en su librería correspondiente tal y como vimos en el Ejercicio_2).

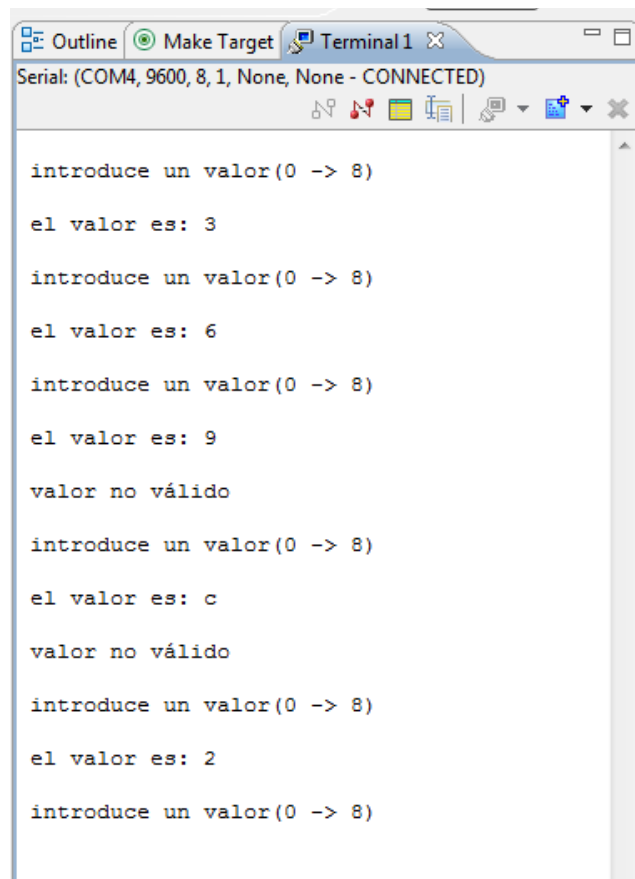
Asimismo, debe comprobar que el valor leído está entre esos dos números y luego mostrarlo tanto en Terminal como en los LEDs.

Si el valor introducido no es correcto, debe volver a preguntar en Terminal que introduzcamos el valor.

Una vez introducido el código C, compilamos la nueva aplicación (Software -> Build All) y programamos la FPGA.

Por último, abrimos Terminal para verificar el correcto funcionamiento seleccionando para su configuración la velocidad y el número de bits que vienen por defecto. Una vez conectado, podemos escribir un número para ver el comportamiento de la solución implementada.

Comprobar el correcto funcionamiento y guardar la solución.



```
Serial: (COM4, 9600, 8, 1, None, None - CONNECTED)

introduce un valor(0 -> 8)
el valor es: 3
introduce un valor(0 -> 8)
el valor es: 6
introduce un valor(0 -> 8)
el valor es: 9
valor no válido
introduce un valor(0 -> 8)
el valor es: c
valor no válido
introduce un valor(0 -> 8)
el valor es: 2
introduce un valor(0 -> 8)
```

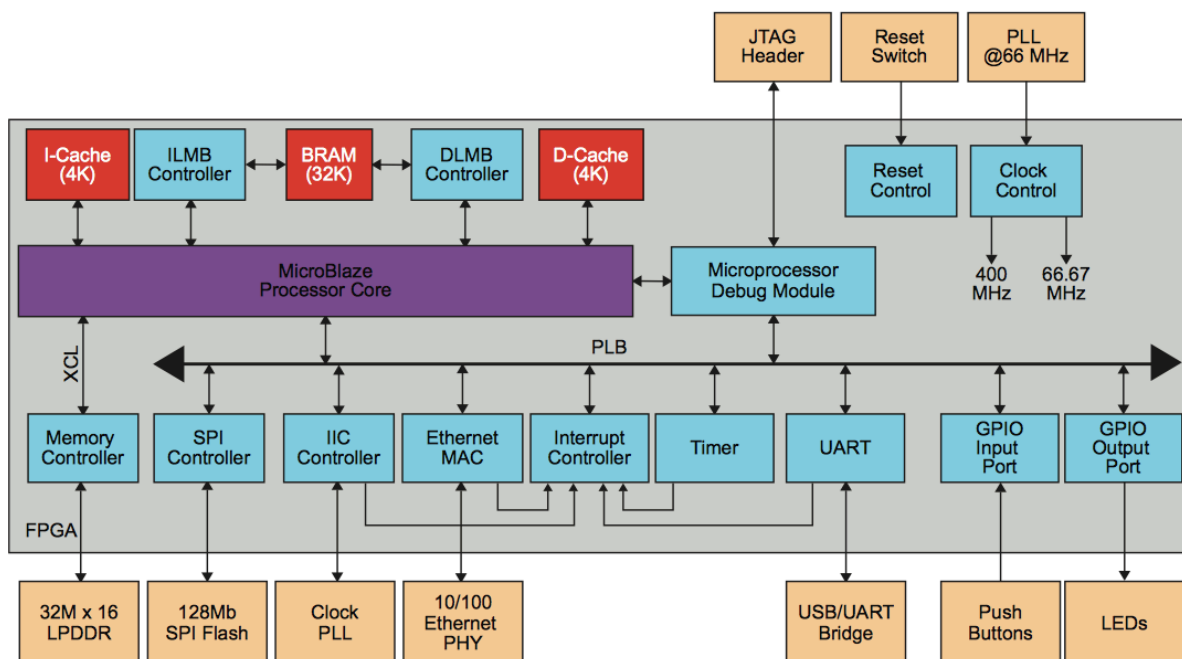
Como ejercicio adicional se pide hacer que MicroBlaze lea el valor de los interruptores y envíe ese valor por el puerto serie, no usando la función de escribir en pantalla sino usando la función de enviar por puerto serie.

Configuración periféricos UART

En este quinto diseño con MicroBlaze vamos a crear un sistema empotrado hardware configurando un periférico UART (Universal Asynchronous Receiver-Transmitter).

El desarrollo de este ejercicio nos llevará a crear un diseño hardware con Microblaze utilizando de nuevo BSB pero añadiendo y configurando un nuevo periférico UART para la comunicación con un módulo PMOD Bluetooth PmodBT2 de DigiLent. Con este módulo de comunicación Bluetooth se va a establecer la conexión con un dispositivo móvil mediante el cual se controlarán los LEDs de nuestra placa LX9.

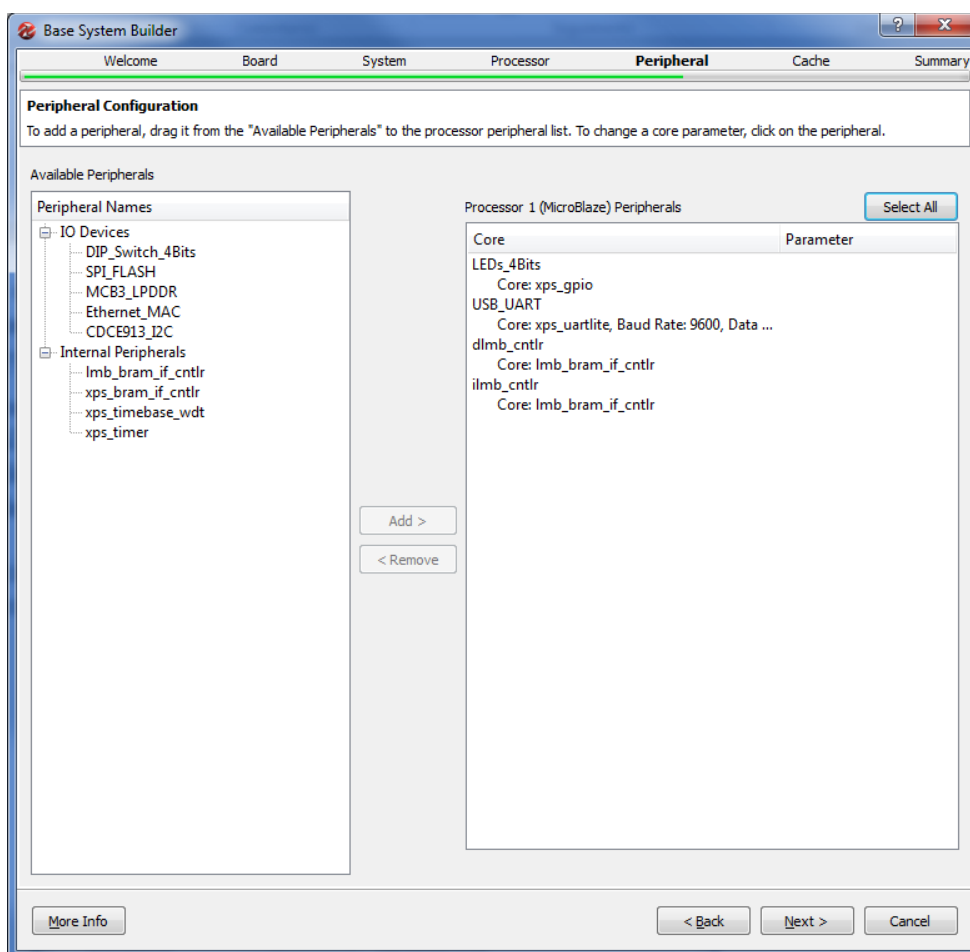
Así mismo, crearemos una aplicación software para gestionar estos periféricos. El diagrama de bloques que tendrá esta aplicación es el siguiente.



Creación sistema empotrado - XPS

Siguiendo los pasos del Ejercicio_1, creamos un nuevo proyecto con el nombre Ejercicio_5. Definiendo MicroBlaze como nuestro microprocesador empotrado, la misma placa de desarrollo usada anteriormente, la misma frecuencia y el mismo tamaño de memoria BRAM.

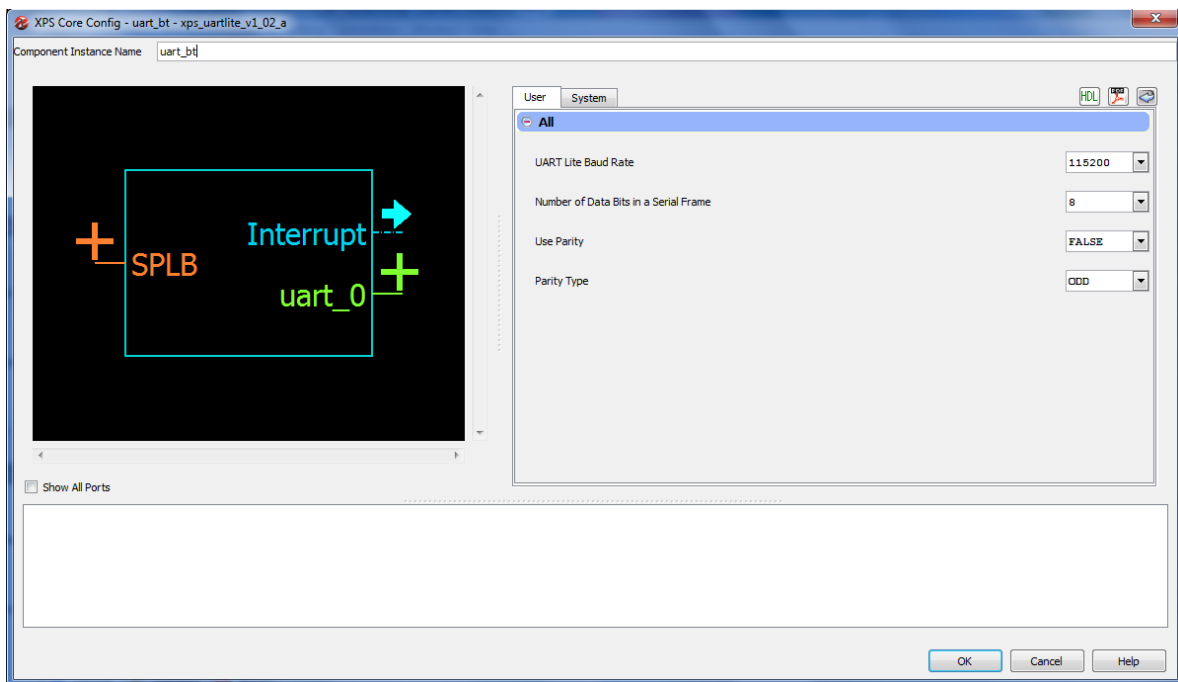
En este caso, usaremos el puerto USB_UART para enviar los comandos recibidos por el periférico Bluetooth creado (UART_BT) al Terminal del PC y poder visualizar el flujo de comunicación. También emplearemos los LEDs que serán controlados por el dispositivo móvil. El resto de periféricos serán desactivados (Switches, SPI Flash, DDR SDRAM y Ethernet). Debemos obtener lo siguiente.



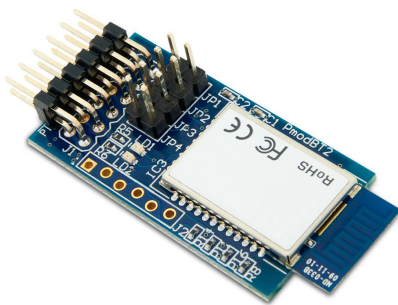
Una vez abierto el entorno XPS con el nuevo proyecto creado, implementar el hardware con la opción Hardware -> Generate Bitstream.

Al igual que en el Ejercicio_4, vamos a añadir un nuevo periférico a nuestro sistema empotrado. En este caso no será un GPIO si no un periférico UART, que será el encargado de comunicarse con el módulo Bluetooth. Para ello, seleccionando la pestaña de IP Catalog, expandimos la opción Communication Low-Speed y se añadimos XPS UART (Lite).

Se abrirá una ventana en la que se puede configurar el periférico UART. En primer lugar, se cambia el nombre en Component Instance Name y se introduce `uart_bt`.



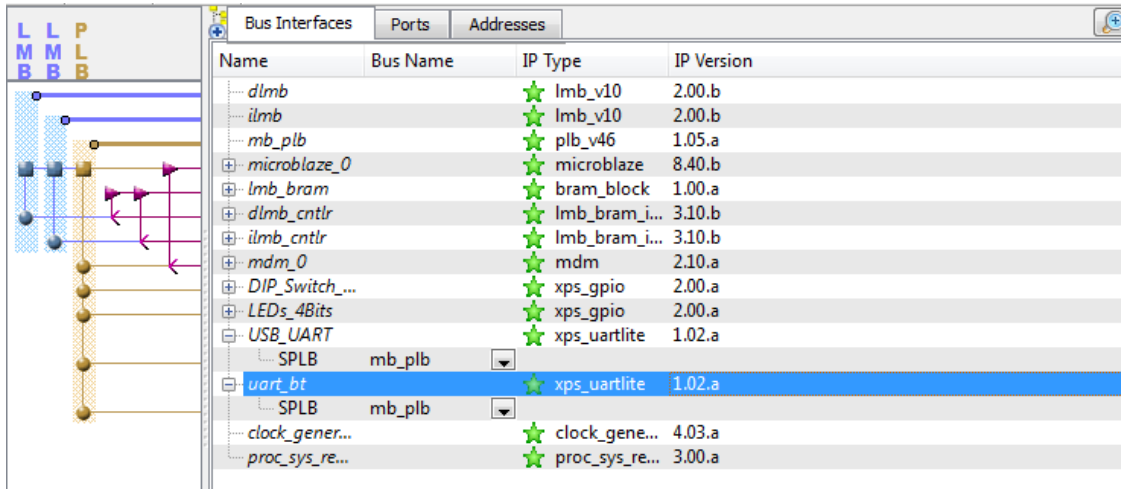
En la pestaña User se configura la comunicación serie con la configuración que emplea el módulo Bluetooth PmodBT2 de Digilent para comunicarse, por tanto, debemos abrir el datasheet y buscar dichos parámetros (PmodBT2_rm.pdf).



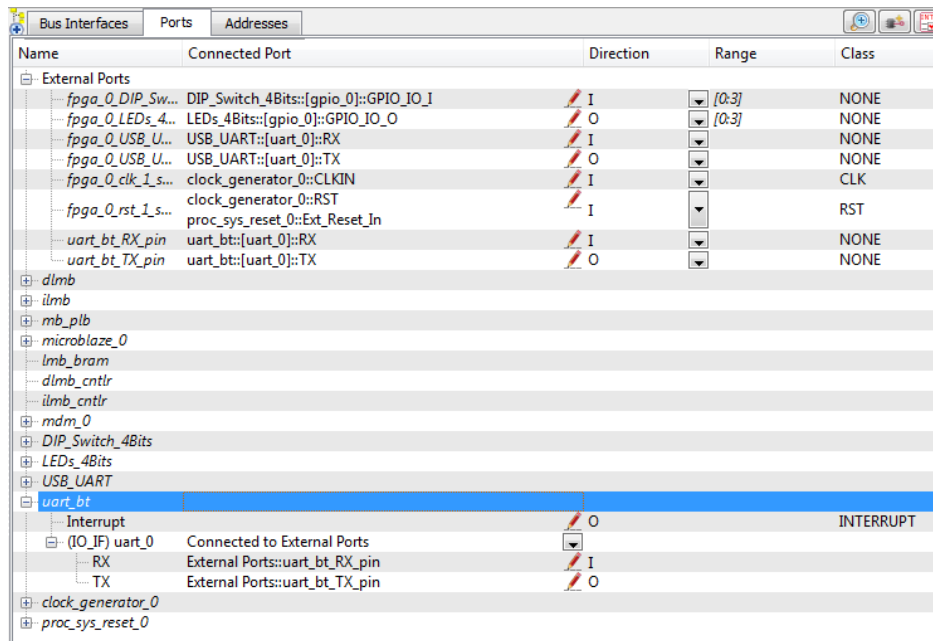
UART Interface

By default, the UART interface uses a baud rate of 115.2 kbps, 8 data bits, no parity, and a single stop bit. The startup baud rate may be customized to predefined rates or set to a specific user customized baud rate. Predefined baud rates range from 1200 to 921k.

Ahora conectamos el periférico creado al bus PLB.



Configuramos el periférico uart_bt generado con conexión externa y definimos el pin uart_bt_RX_pin como entrada y uart_bt_TX_pin como salida.



Este nuevo periférico no tiene asignada una dirección de memoria. Por tanto, debemos ir a la pestaña Addresses y bloquear las direcciones asignadas a los periféricos iniciales de nuestro sistema empotrado.

Pinchamos en Generate Addresses (icono que se encuentra en la pestaña Addresses en la parte superior derecha) y aparecerán las direcciones de nuestros nuevos periféricos.

Instance	Base Name	Base Address	High Address	Size	Bus Interface(s)	Bus Nam
microblaze_0's Address Map						
dlmb_cntlr	C_BASEADDR	0x00000000	0x00007FFF	32K	SLMB	dlmb
ilmb_cntlr	C_BASEADDR	0x00000000	0x00007FFF	32K	SLMB	ilmb
LEDs_4Bits	C_BASEADDR	0x81400000	0x8140FFFF	64K	SPLB	mb_plb
DIP_Switch_4Bits	C_BASEADDR	0x81420000	0x8142FFFF	64K	SPLB	mb_plb
uart_bt	C_BASEADDR	0x84000000	0x8400FFFF	64K	SPLB	mb_plb
USB_UART	C_BASEADDR	0x84020000	0x8402FFFF	64K	SPLB	mb_plb
mdm_0	C_BASEADDR	0x84400000	0x8440FFFF	64K	SPLB	mb_plb

Ahora debemos añadir los periféricos a nuestro archivo de restricciones para definir la asignación de los terminales externos. Abrimos el archivo system.ucf que se encuentra en la pestaña Project -> Project Files.

```

1 # Avnet Spartan-6 LX9 MicroBoard
2 Net fpga_0_USB_UART_RX_pin LOC=R7 | IOSTANDARD = LVCMOS33;
3 Net fpga_0_USB_UART_TX_pin LOC=T7 | IOSTANDARD = LVCMOS33;
4 Net fpga_0_LEDs_4Bits_GPIO_IO_0_pin<0> LOC=P4 | IOSTANDARD = LVCMOS18;
5 Net fpga_0_LEDs_4Bits_GPIO_IO_0_pin<1> LOC=L6 | IOSTANDARD = LVCMOS18;
6 Net fpga_0_LEDs_4Bits_GPIO_IO_0_pin<2> LOC=F5 | IOSTANDARD = LVCMOS18;
7 Net fpga_0_LEDs_4Bits_GPIO_IO_0_pin<3> LOC=C2 | IOSTANDARD = LVCMOS18;
8 Net fpga_0_DIP_Switch_4Bits_GPIO_IO_I_pin<0> LOC=B3 | IOSTANDARD = LVCMOS33 | PULLDOWN;
9 Net fpga_0_DIP_Switch_4Bits_GPIO_IO_I_pin<1> LOC=A3 | IOSTANDARD = LVCMOS33 | PULLDOWN;
10 Net fpga_0_DIP_Switch_4Bits_GPIO_IO_I_pin<2> LOC=B4 | IOSTANDARD = LVCMOS33 | PULLDOWN;
11 Net fpga_0_DIP_Switch_4Bits_GPIO_IO_I_pin<3> LOC=A4 | IOSTANDARD = LVCMOS33 | PULLDOWN;
12 Net fpga_0_clk_1_sys_clk_pin TNM_NET = sys_clk_pin;
13 TIMESPEC TS_sys_clk_pin = PERIOD sys_clk_pin 66666.7 kHz;
14 Net fpga_0_clk_1_sys_clk_pin TNM_NET = sys_clk_pin | LOC=K15 | IOSTANDARD = LVCMOS33;
15 Net fpga_0_rst_1_sys_rst_pin TIG;
16 Net fpga_0_rst_1_sys_rst_pin LOC=V4 | IOSTANDARD = LVCMOS33 | PULLDOWN;
17
18
19 ##### microblaze_0
20 ### Set Vccaux for S6LX9 MicroBoard to 3.3V ###
21 CONFIG VCCAUX = "3.3" ;
22
23

```

Añadiremos los terminales necesarios para interconectar el periférico `uart_bt` generado con el módulo PMOD de Bluetooth. Podéis obtenerlos del datasheet de la PCB de la LX9 (`xlx_s9_lx9_fpga_microboard-ug072711.pdf`) y del módulo BT de forma que los pines del conector de la placa LX9 seleccionados para la transmisión serie coincidan con el RX y TX del PMOD.

Connector J1 – UART Communications		
Pin	Signal	Description
1	RTS	Ready to Send
2	RX	Receive
3	TX	Transmit
4	CTS	Clear to Send
5	GND	Power Supply Ground
6	VCC	Power Supply (3.3V)
7	STATUS	Connection Status
8	~RST	Reset
9	NC	Not Connected
10	NC	Not Connected
11	GND	Power Supply Ground
12	VCC	Power Supply (3.3V)

2.5 User I/O and Expansion Connectors

2.5.1 Peripheral Module (PMOD)

Two 12-pin (2 x 6 female) Peripheral Module (PMOD) headers (J4, J5) are interfaced to the FPGA, with each header providing 3.3 V power, ground, and eight I/O's. These headers may be utilized as general-purpose I/Os or may be used to interface to PMODs. J4 and J5 are placed in close proximity (0.9"-centers) on the PCB in order to support dual PMODs. Table 9 and Table 10 provide the connector and FPGA pinout. For Digilent PMODs see: <http://www.digilentinc.com/pmods>

FPGA Pin #	I/O Signal	Connector Pin #	Connector Pin #	I/O Signal	FPGA pin #
H12	FPGA_PMOD2_P1	1	7	FPGA_PMOD2_P7	K12
G13	FPGA_PMOD2_P2	2	8	FPGA_PMOD2_P8	K13
E16	FPGA_PMOD2_P3	3	9	FPGA_PMOD2_P9	F17
E18	FPGA_PMOD2_P4	4	10	FPGA_PMOD2_P10	F18
-	GND	5	11	GND	-
-	+3.3V_LS1	6	12	+3.3V_LS1	-

Table 9 – Peripheral Module Connections – J4

El nombre que debemos poner a los terminales debe coincidir con el que aparece en la sección Ports del archivo `system.mhs`.

Ahora debemos implementar de nuevo el hardware para añadir a nuestro sistema empotrado los nuevos periféricos y comprobar que no hay errores (Hardware -> Generate Bitstream)

Programación del control de la UART generada - SDK

Para poder usar estos periféricos y comprobar que nuestra aplicación funciona correctamente, en lugar de usar las aplicaciones de test que genera SDK, vamos a crear una nueva aplicación software.

En SDK, como siempre, usamos la opción File->New->Application Project y le asignaremos el nombre uso_bluetooth a nuestro proyecto software. Teniendo cuidado con tener elegido el procesador correcto (microblaze_0) y seleccionando el hardware generado con EDK para este ejercicio (Ejercicio_5_hw_platform). En cuanto al soporte de la placa que estamos empleando (Board Support Package), se van a generar de nuevo los archivos de soporte de la tarjeta LX9 que se está empleando.

Añadimos el fichero fuente pulsando el botón derecho sobre el nombre de nuestro proyecto y seleccionamos New->Source File, se indica el nombre del proyecto (uso_bluetooth), el nombre del archivo (control_bluetooth.c) y el lenguaje que se va a emplear para su programación (C).

Queremos en esta aplicación que al pulsar el botón LED ON en la app de Android, reciba el '1' que se envía al Bluetooth PMOD y encienda todos los LEDs de la placa. Mientras que si se recibe un '0' es porque se ha pulsado el botón LED OFF y se deben apagar todos los LEDs.

Si no recibimos ninguno de esos valores pero existe comunicación, debemos recibir una 'E' de Error y encender una A en los LEDs.

Utilizando esta plantilla del código, completarla adecuadamente para que realice lo comentado anteriormente.

```
#include "xparameters.h"

#include "xuartlite.h"
#include "xuartlite_l.h"
#include "xil_exception.h"

#include "xutil.h"
#include "xgpio.h"

#define SEND_BUFFER_SIZE X

/*
 * Los siguientes buffers se usan para enviar y recibir datos por las UARTs.
 */

u8 SendBuffer[X];
u8 RecvBuffer[Y];

int main (void)
{
    // Declaración variables
    XUartLite X, Y;

    XGpio X;
```

```

/*
 * Inicializamos las UARTs
 */
X //Bluetooth
Y //USB-a-PC

/*
 * Inicializamos los LEDs.
 */
XGpio_Initialize(&LEDS4_Bit, XPAR_LEDS_4BITS_DEVICE_ID);
XGpio_SetDataDirection (&LEDS4_Bit, 1, 0x00);

while(1)
{
    // Se hace un polling para comprobar si el buffer de recepción está vacío
    // (TRUE) o si ha llegado un dato y no está vacío (FALSE)

    while(!XUartLite_IsReceiveEmpty(X)
    {
        XUartLite_Recv(Y); // Recibe los bytes desde Bluetooth

        Z; // Limpia Buffer de FIFOs reseteándolos para que el polling
funcione correctamente

        if(RecvBuffer[X]== 'Y')
        {
            XGpio_DiscreteWrite(&LEDS4_Bit, 1, HEX1);
        }
        else if(RecvBuffer[X]== 'I')
        {
            XGpio_DiscreteWrite(&LEDS4_Bit, 1, HEX2);
        }
        else
        {
            RecvBuffer[X]='I';
            XGpio_DiscreteWrite(&LEDS4_Bit, 1, HEX3);
        }

        XUartLite_Send(X); // Se hace un eco de lo que se recibe por BT y se
envía por comunicación serie al PC.
    }
}
}

```

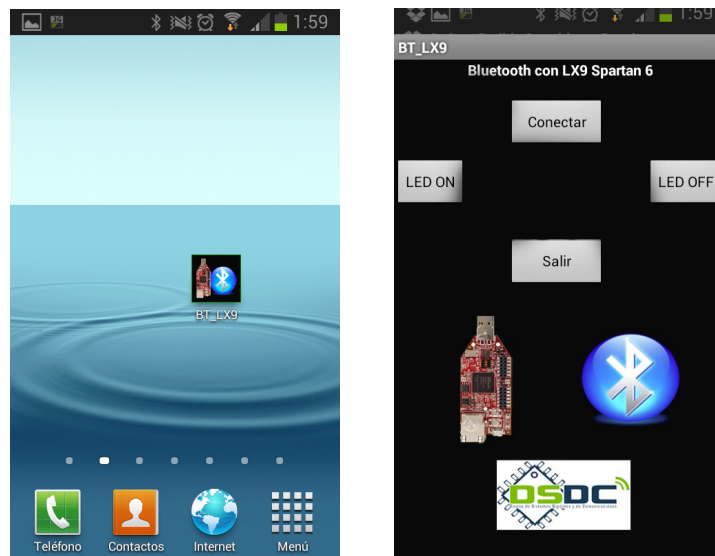
Ahora compilamos la nueva aplicación guardando lo escrito y programamos la FPGA.

Por último, abrimos Terminal y comprobamos el correcto funcionamiento del programa pulsando los botones LED ON y LED OFF de la aplicación Android para el dispositivo móvil.

Aplicación en Android

Se ha creado una aplicación simple para Android llamada BT_LX9 (con la extensión de la MAC de cada Bluetooth) con el fin de probar el correcto funcionamiento del periférico uart_bt generado con el PMOD Bluetooth. Dicha aplicación la tenéis disponible en el Aula Virtual.

De esta forma, se ha programado una aplicación que consta de cuatro botones:



- Conectar: Se emplea para realizar la conexión Bluetooth del dispositivo móvil Android con el módulo Bluetooth PmodBT2 de Digilent conectado a la FPGA mediante el conector J4 empleando comunicación serie.
- LED ON: Al pulsar este botón se encenderán los LEDs de la placa LX9.
- LED OFF: Al pulsar este botón se apagan los LEDs de la placa LX9.
- Salir: Cierra la aplicación BT_LX9 y elimina el proceso abierto en Android.

Para poder conectar correctamente con el módulo PmodBT2 es necesario antes de pulsar el botón Conectar, vincularlo de forma manual el dispositivo móvil. Para ello, se accede dentro de Ajustes, a las opciones de conexiones Bluetooth del dispositivo móvil, se selecciona como visible y se procede a realizar una búsqueda de nuevos dispositivos. Cuando aparezca un dispositivo RN42 seguido de cuatro caracteres, lo seleccionamos para proceder con la vinculación y le introducimos la contraseña 1234.



Notar que los cuatro caracteres que aparecen en el nombre del dispositivo Bluetooth corresponden con los últimos caracteres de la dirección MAC del módulo PmodBT2. La dirección MAC puede ser consultada, conectando al PC el módulo PmodBT2 y enviándole los comandos de configuración correspondientes (acceso a configuración “\$\$\$”).

Una vez vinculado el dispositivo móvil con el módulo PmodBT2, podemos abrir la aplicación de Android BT_LX9 de forma que ya es posible establecer la conexión y poder controlar los LEDs de la placa LX9.