

Tema 1. Objetivos

1. Diseño de sistemas digitales (**microprocesador, computador**)
2. Mediante lenguaje de descripción de hardware (**VHDL**)
3. Con dispositivos lógicos programables (**FPGA**)
4. Con el software de desarrollo (**Vivado de Xilinx**) + Simulador (**ISESimulator**)
5. Usando una placa de evaluación (**Nexys 4 - DDR**)

1. Sistemas Digitales

1. Sistema digital vs sistema analógico

Analógico

versus

Digital

-Toman valores en un rango continuo

-Ejemplos:

- Temperatura
- Voltaje
- Tiempo

- Correspondencia matemática

↓
Números reales

-Toman valores en un rango discreto

-Ejemplos:

- Número de personas
- Número de libros
- Etc.

- Correspondencia matemática

↓
Números enteros

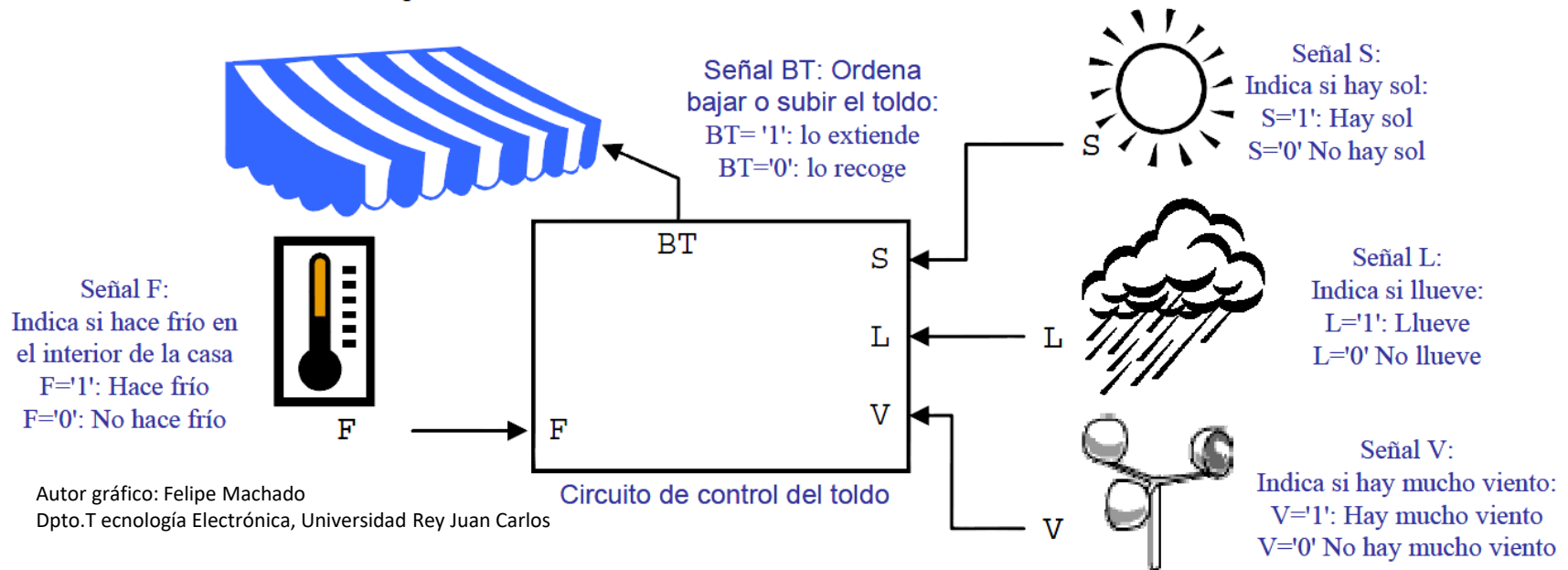


ÉXITO DE LOS SISTEMAS DIGITALES:

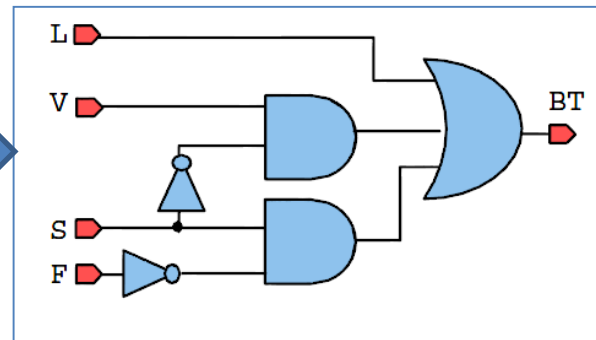
- **Programables**
- **Flexibles**
- **Mayor velocidad de funcionamiento**
- **Mayor inmunidad al ruido**
- **Menor tamaño**

1. Implementación de un diseño digital

1. Puertas lógicas ó Componentes discretos



- Tabla de verdad
- Mapa de Karnaugh
- Simplificación ecuaciones booleanas
- Implementar: puertas lógicas o MSI
- Test



No flexible

1. Implementación de un diseño digital

2. Lógica programable (uC, FPGAs, uP)

```
if ("No llueve" and  
    "Hay sol" and  
    "No viento")  
then  
    "Echar toldo"  
Else  
    "Recoger toldo"  
End if;
```

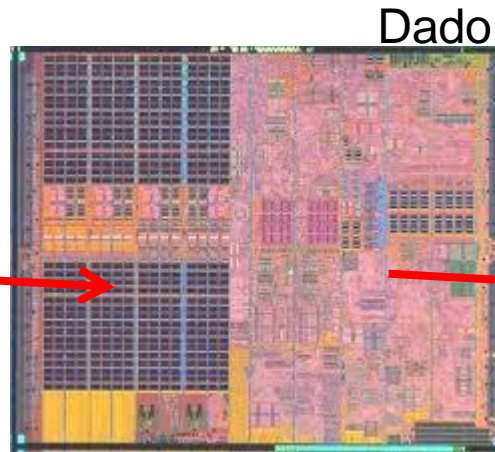
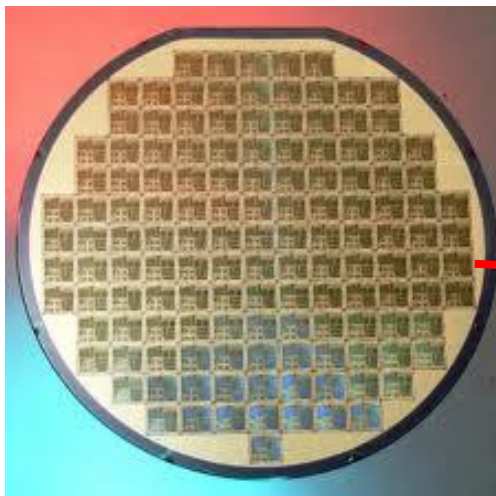


- Flexibles
- Diseño rápido
- Diseños con cientos y millones de puertas
- Reutilizable
- Producción baja y media (Menor \$)

1. Implementación de un diseño digital

3. ASIC (Circuito Integrado para Aplicaciones Específicas)

- Implementan en el dado la lógica específica diseñada. → El chip de un teléfono

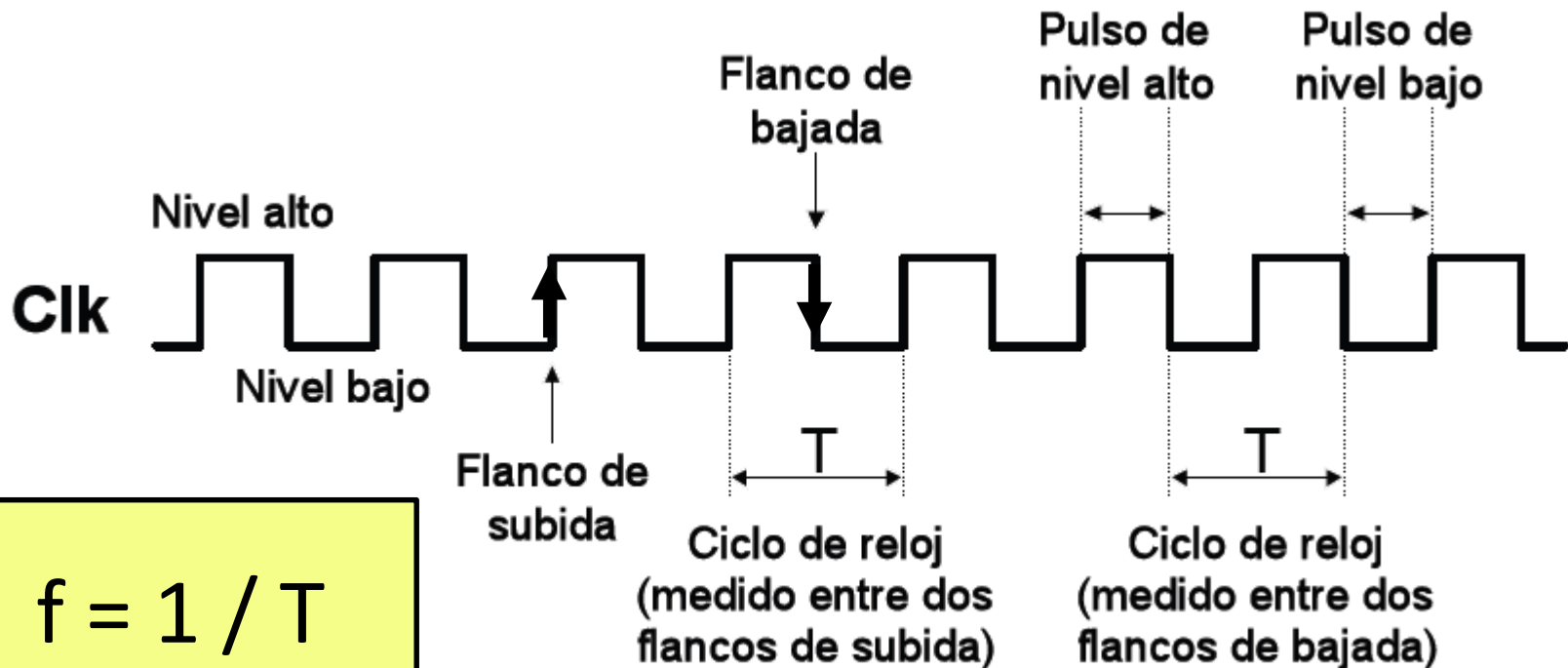


- No flexibles : **Error → Repetir máscara (mill. \$)**
- Mayor tiempo de diseño para no cometer errores
- Diseños con cientos y millones de puertas
- Coste alto
- Producción alta : **Menor precio/unidad**

1. Conceptos importantes: Reloj

Reloj (CLK)

- Señal periódica que varía entre un nivel alto (“1”) y un nivel bajo “0”
- Presente en todos los sistemas digitales
- Duracion infinita



$$f = 1 / T$$

1. Conceptos importantes: Reloj

Múltiplos del Sistema Internacional para segundo (s)

Submúltiplos			Múltiplos		
Valor	Símbolo	Nombre	Valor	Símbolo	Nombre
10^{-1} s	ds	decisegundo	10^1 s	das	decasegundo
10^{-2} s	cs	centisegundo	10^2 s	hs	hectosegundo
10^{-3} s	ms	milisegundo	10^3 s	ks	kilosegundo
10^{-6} s	μs	microsegundo	10^6 s	Ms	megasegundo
10^{-9} s	ns	nanosegundo	10^9 s	Gs	gigasegundo
10^{-12} s	ps	picosegundo	10^{12} s	Ts	terasegundo
10^{-15} s	fs	femtosegundo	10^{15} s	Ps	petasegundo
10^{-18} s	as	attosegundo	10^{18} s	Es	exasegundo
10^{-21} s	zs	zeptosegundo	10^{21} s	Zs	zettasegundo
10^{-24} s	ys	yoctosegundo	10^{24} s	Ys	yottasegundo
Prefijos comunes de unidades están en negrita.					

Fuente: Wikipedia. <http://es.wikipedia.org/wiki/Segundo>

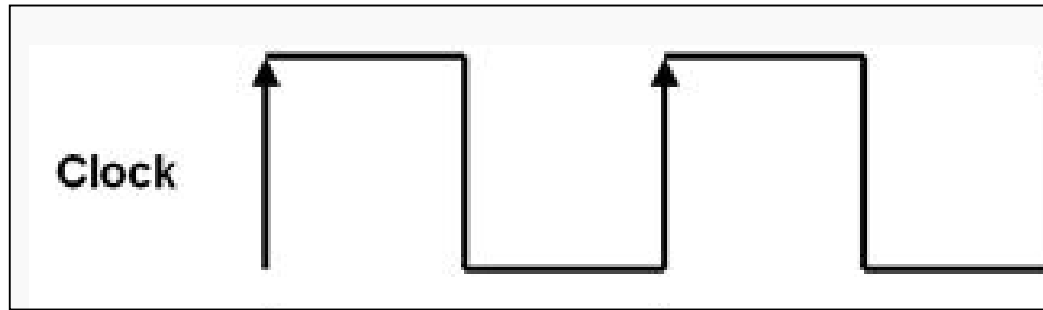
1. Conceptos importantes: Reloj

Múltiplos del Sistema Internacional para hercio (Hz)

Submúltiplos			Múltiplos		
Valor	Símbolo	Nombre	Valor	Símbolo	Nombre
10^{-1} Hz	dHz	decihercio	10^1 Hz	daHz	decahercio
10^{-2} Hz	cHz	centihercio	10^2 Hz	hHz	hectohercio
10^{-3} Hz	mHz	milihercio	10^3 Hz	kHz	kilohercio
10^{-6} Hz	μHz	microhercio	10^6 Hz	MHz	megahercio
10^{-9} Hz	nHz	nanohercio	10^9 Hz	GHz	gigahercio
10^{-12} Hz	pHz	picohercio	10^{12} Hz	THz	terahercio
10^{-15} Hz	fHz	femtohercio	10^{15} Hz	PHz	petahercio
10^{-18} Hz	aHz	attohercio	10^{18} Hz	EHz	exahercio
10^{-21} Hz	zHz	zeptohercio	10^{21} Hz	ZHz	zettahercio
10^{-24} Hz	yHz	yoctohercio	10^{24} Hz	YHz	yottahercio

Fuente: Wikipedia. <http://es.wikipedia.org/wiki/Hercio>

1. Conceptos importantes: Reloj

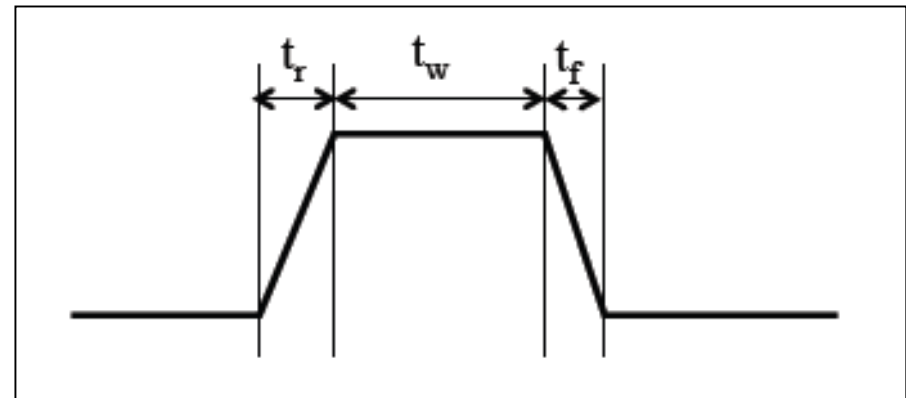


Representación *ideal* de una señal de reloj

Tiempo de subida (*rise-time*) (t_r)

Tiempo de bajada (*fall-time*) (t_f)

Ancho de pulso (*width*) (t_w)

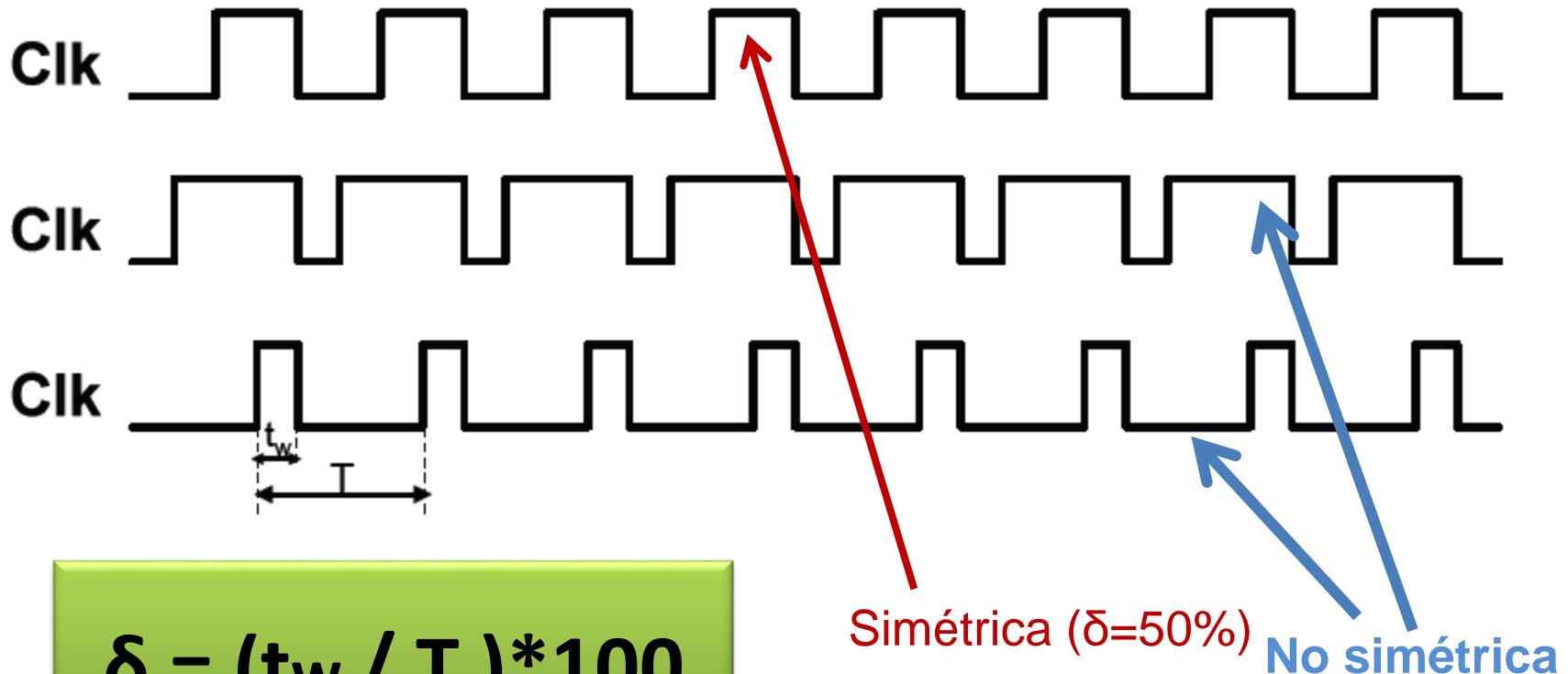


Representación *real* de una señal de reloj

1. Conceptos importantes: Reloj

Ciclo de trabajo de una señal de reloj (δ)

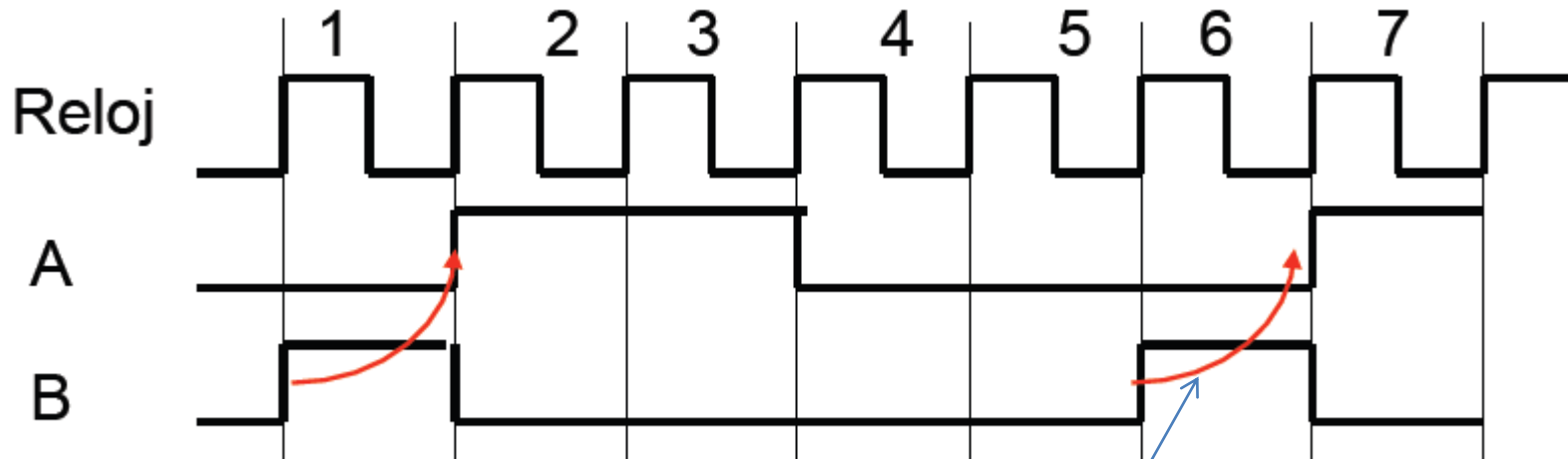
Razón entre el ancho del pulso (t_w) y el periodo (T)



1. Conceptos importantes: Cronograma

Cronograma ó diagrama de tiempo

Conjunto de formas de onda de las señales de un sistema digital (μP) relacionadas entre si.

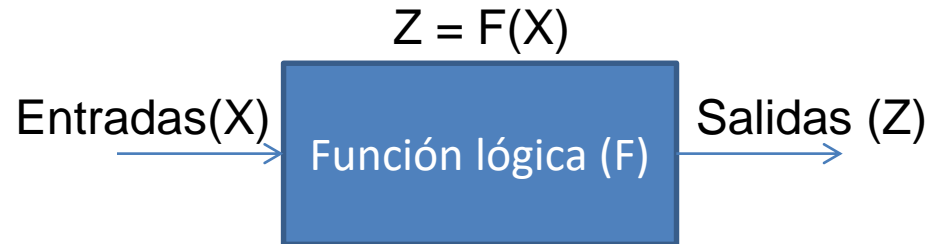


En ciclo 1: A='0' y B='1'

La transición de A es causa del flanco de subida de B

1. Sistemas digitales combinacionales

- Sus **salidas son función exclusiva del valor de sus entradas**, en un momento dado, sin que intervengan en ningún caso estados anteriores de las entradas o de las salidas.



- Su función se representa en una **tabla de la verdad**.
- **Carecen de memoria** y de retroalimentación.
- Ejemplos:

- **Lógicos:**

- ☐ Generador/Detector de paridad
- ☐ Multiplexor y Demultiplexor
- ☐ Codificador y Decodificador
- ☐ Conversor de código
- ☐ Comparador

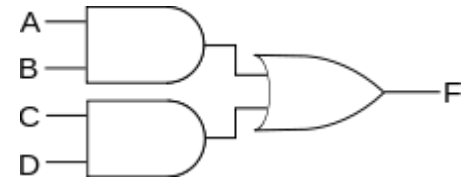
- **Aritméticos:**

- ☐ Sumador

- **Aritméticos y lógicos**

- ☐ Unidad aritmético lógica

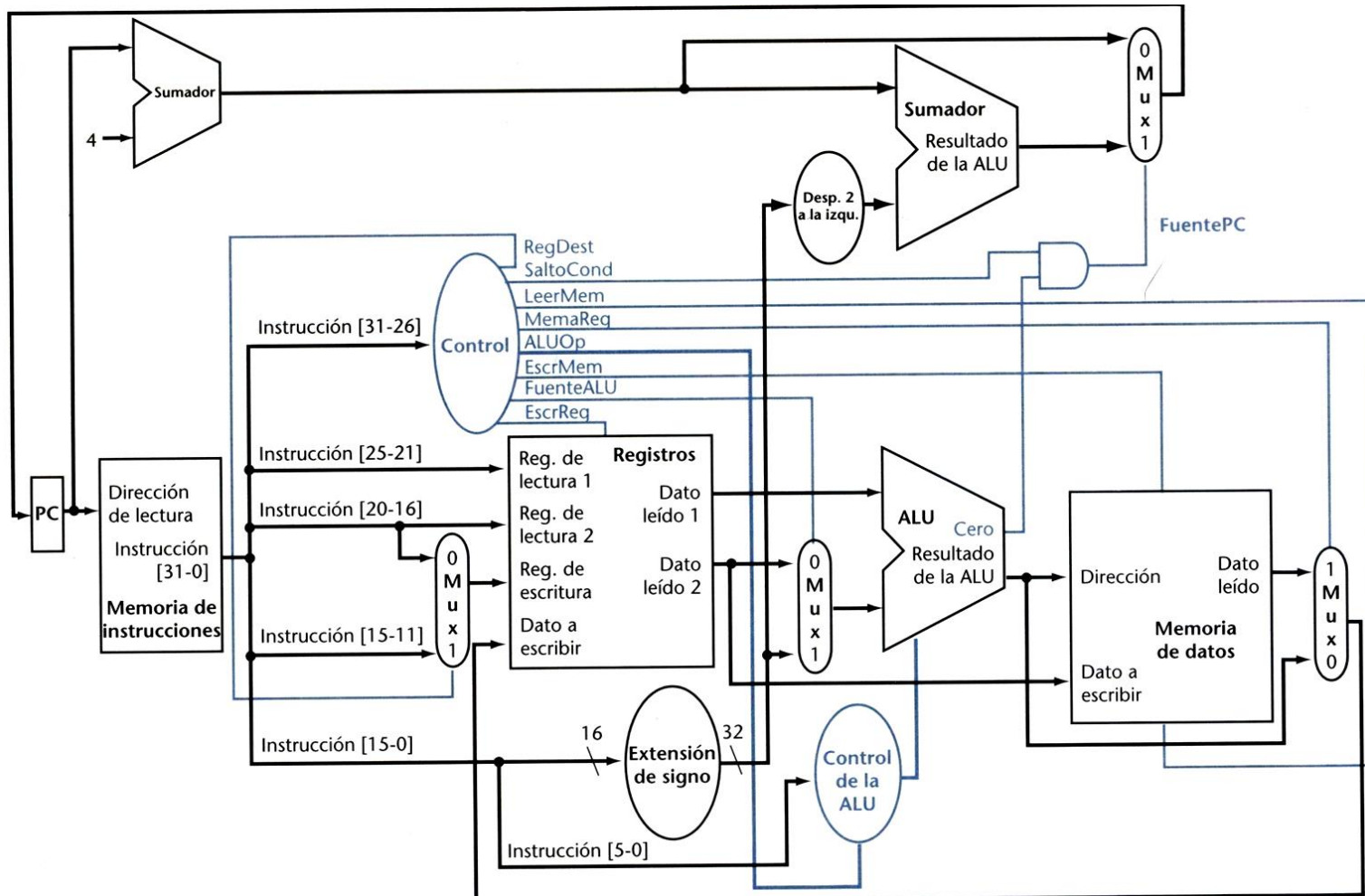
Están formados por puertas lógicas (AND, OR, XOR, etc.)



Sistema digital combinacional

1. Lógica combinacional

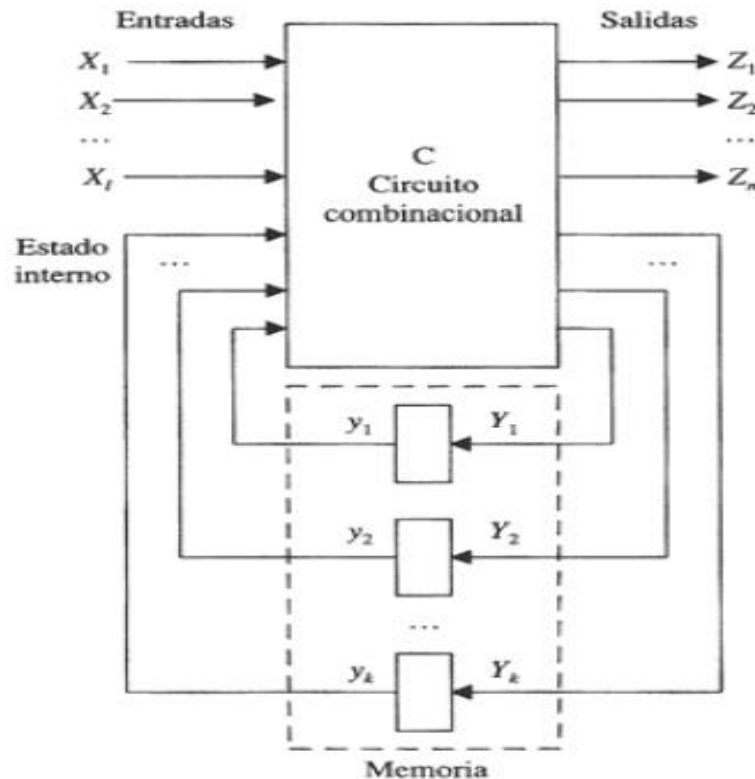
Procesador MIPS no segmentado (monociclo)



[Patt00]

1. Sistemas digitales secuenciales

- Los valores de las salidas, en un momento dado (t), no dependen exclusivamente de los valores de las entradas en dicho momento, sino que también dependen del estado anterior ($t-1$).



Modelo de Huffman para un circuito secuencial.

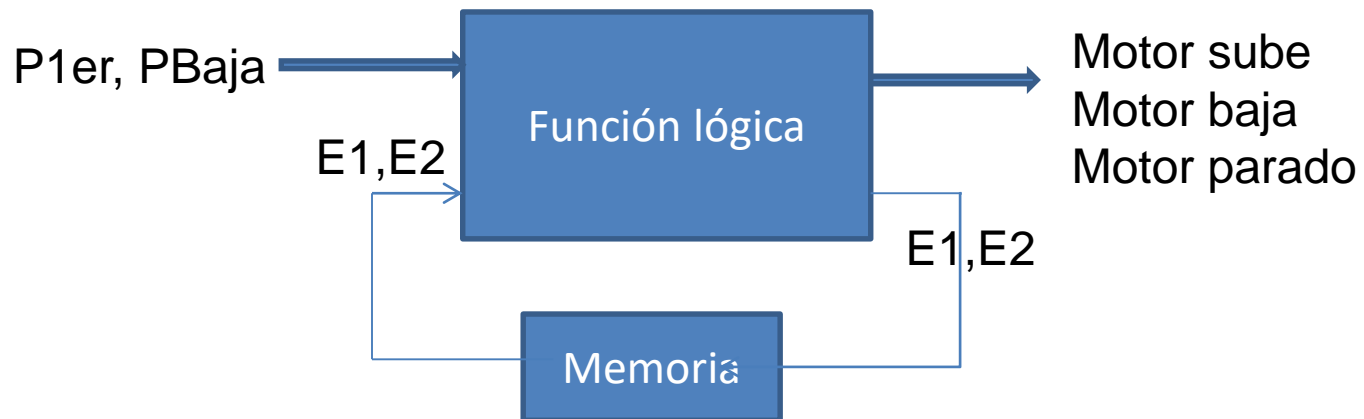
- Necesitan de un circuito de memoria que almacene el estado anterior

1. Sistemas digitales secuenciales

Ejemplo 1: Circuito de una única entrada y una única salida y que proporciona salida “1” cuando aparece en la entrada la secuencia “11”.

Ejemplo 2: Subir/bajar en Ascensor (Entradas botón “P1er” y “PBaja”)

Entradas (Pulsador)		Estado actual o interno	Salida (Motor)
P1er	PBaja		
0	1	Está abajo (E1)	Motor parado
0	1	Está arriba (E2)	Motor baja
1	0	Está arriba(E2)	Motor parado
1	0	Está abajo(E1)	Motor sube

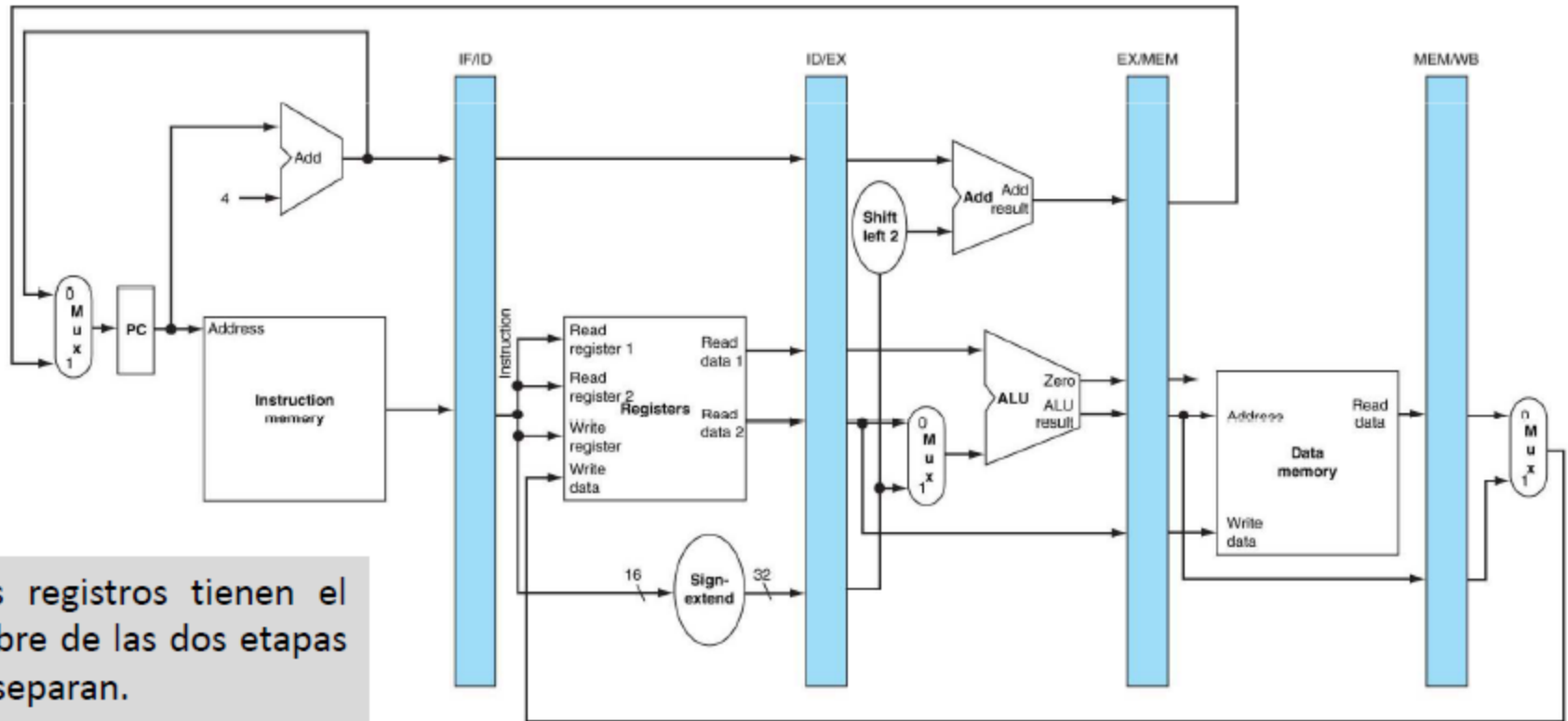


1. Sistemas digitales secuenciales

- Se clasifican en:
 - ❑ **Síncrono:** solo cambian ante un determinado evento de una señal de reloj
 - ✓ **Por nivel: alto o bajo**
 - ✓ **Por flanco: de subida de bajada**
 - ❑ **Asíncrono:** pueden cambiar en cualquier momento
- Entre los circuitos secuenciales clásicos tenemos:
 - ❑ **Biestables, Flip-Flops**
 - ❑ **Contadores**
 - ❑ **Registros**
- Suelen contar con una señal que inicia los elementos de memoria a un valor conocido (normalmente 0). Recibe el nombre de **RESET**.

1. Conceptos importantes: Lógica secuencial

Procesador MIPS segmentado (multiciclo)



-Figura obtenida de [PATT11]-

2. Lenguajes de descripción de hardware (HDL)

2. Lenguajes de descripción de hardware (HDL)

Sobre los lenguajes HDL...

- Lenguaje inventado para describir hardware → **Concurrente**
- C++, Java, etc. son lenguajes para describir algoritmos → **Secuencial**
- Usado para modelar circuitos y sistemas digitales¹
- Puede ser traducido para generar circuitos reales
→ **Crear sistemas complejos en poco tiempo**
- El programador escribe lo que el circuito final deberá hacer, el sintetizador² inferirá como deberá ser el circuito

¹ Circuito que procesa y/o almacena información

² Herramienta software que interpreta el código descrito, devuelve la definición de un circuito que será programado en un PLD (p.e. : FPGA)

2. Lenguajes de descripción de hardware (HDL)

Recuerda ...

- ... que cuando codificas en HDL estás diseñando hardware, **NO** programando.
- ... que debes tener una idea del “aspecto” del circuito final.
- ... que un diseño digital en HDL, al igual que los lenguajes algorítmicos, es posible dividirlo en bloques más simples.

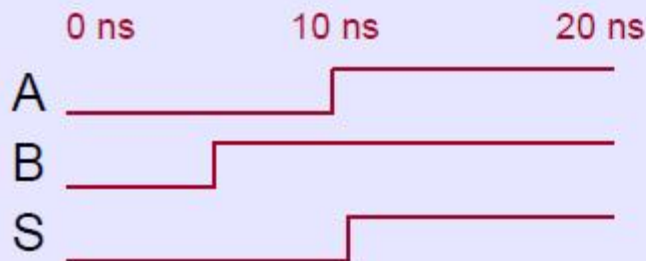
Historia del VHDL

- Lenguaje HDL estandarizado por IEEE desde 1987
- Tiene varias revisiones siendo la última la de 2008.
- Evolución en sus aplicaciones:
 - ✓ Documentación
 - ✓ Simulación
 - ✓ Modelado

2. VHDL

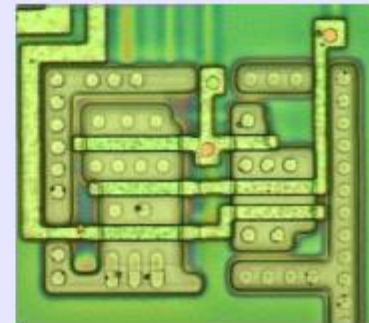
- El VHDL permite modelar **SISTEMAS DIGITALES**
- A partir de estos modelos podremos:

Simular



Comprobar que tienen la funcionalidad deseada

Sintetizar



Crear un circuito que funciona como el modelo

(C) Sergio López Buedo. UAM.

2. Lenguajes de descripción de hardware (HDL)

Ventajas del VHDL

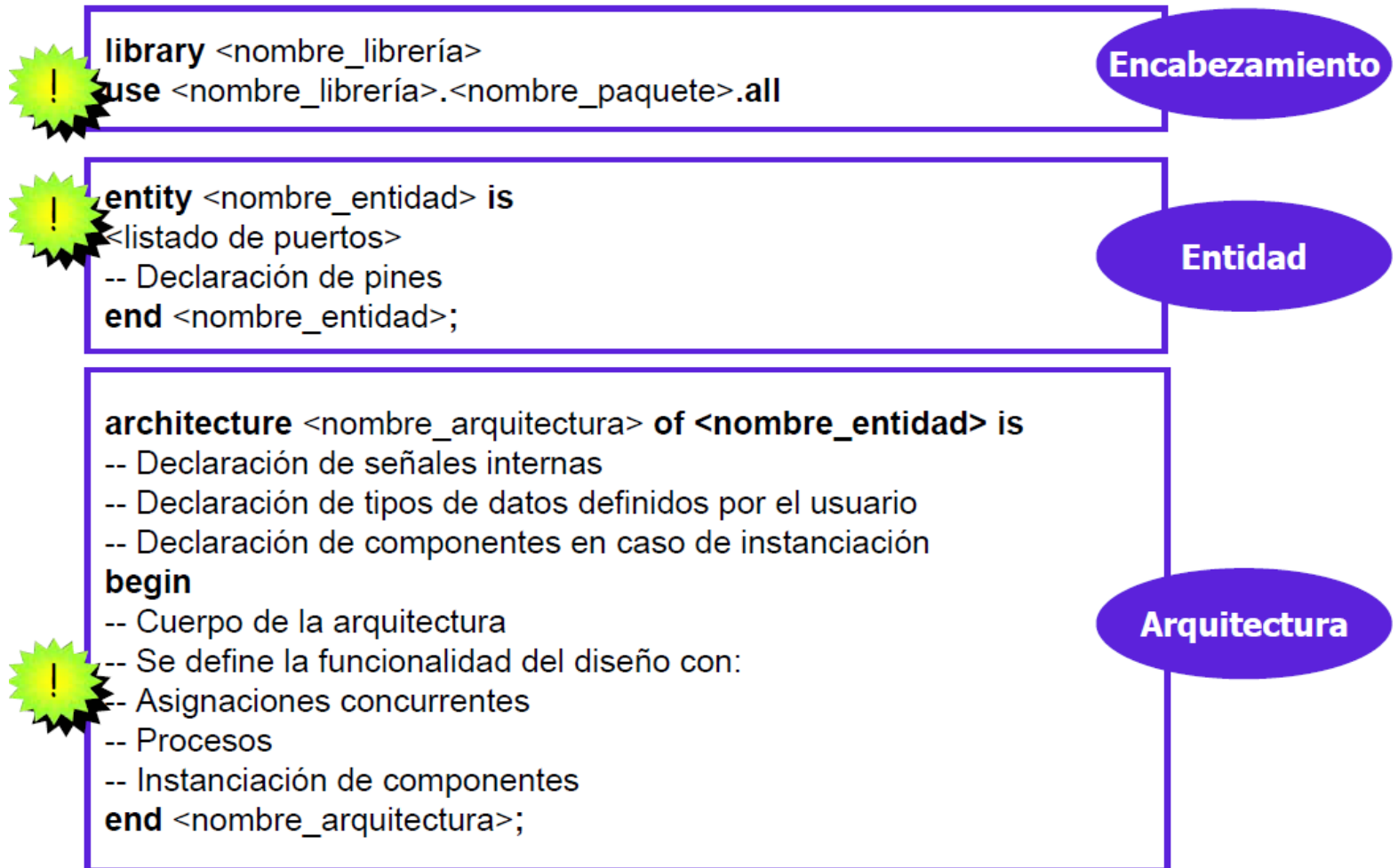
- Disponibilidad pública
- Independencia de dispositivos y fabricantes
- Reutilización
- Diseño jerárquico → Permite conectar componentes

¿Pero... qué significa VHDL?

VHDL – “**V**HSIC **H**ardware **D**escription **L**anguage”

VHSIC – “**V**ery **H**igh **S**peed **I**ntegrated **C**ircuit”

2. Estructura básica de un fichero/módulo VHDL

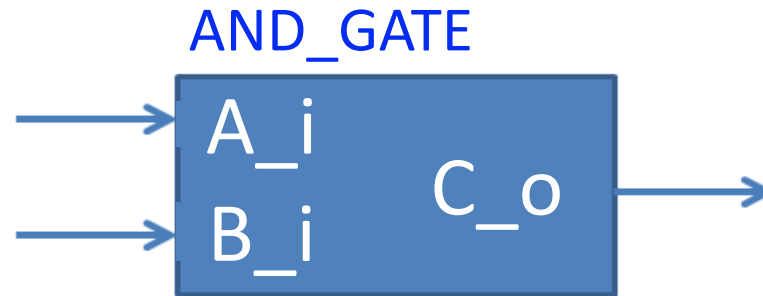


(C) Susana Borromeo. URJC.

2. Ejemplos VHDL

Proyecto01. Crear un módulo VHDL que describa el funcionamiento de una puerta lógica AND.

Primero, se describe la entidad, dándole un nombre y los puertos E/S de que dispondrá. Actúa como interfaz entre este módulo y el exterior (u otro módulo).



Debe incluir:

VHDL keywords: entity, is, Port, in, out, end

- Nombre de la Entity
- Nombre de los Port
- Modo del Port:
 - In
 - Out
 - Inout

```
entity and_gate is
  Port ( A_i : in  STD_LOGIC;
         B_i : in  STD_LOGIC;
         Z_o : out STD_LOGIC );
end and_gate;
```

- Tipo de datos: establece los valores posibles para los *ports*

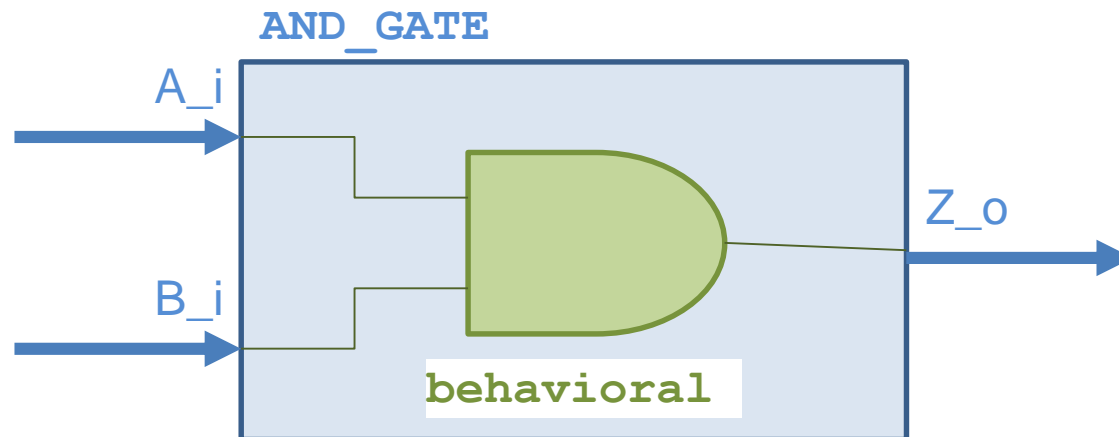
2. Ejemplos VHDL

Proyecto01. Crear un módulo VHDL que describa el funcionamiento de una puerta lógica AND.

Segundo, se describe la funcionalidad del circuito dentro de la “architecture”

```
Architecture behavioral of and_gate is  
begin  
    Z_o <= A_i and B_i;  
end behavioral ;
```

“and” operador lógico
incluido en IEEE_std_logic



3. Dispositivos lógicos programables (PLD)

3. Dispositivos lógicos programables (PLD)

PLDs (Programmable Logic Devices)

Entradas + Inversores

**Matriz
AND**

**Matriz
OR**

Biestables (opcional)

Inversores + Salidas

© Luis Entrena, Celia López, Mario García, Enrique Barrio, Universidad Carlos III de Madrid, 2008

3. Dispositivos lógicos programables (PLD)

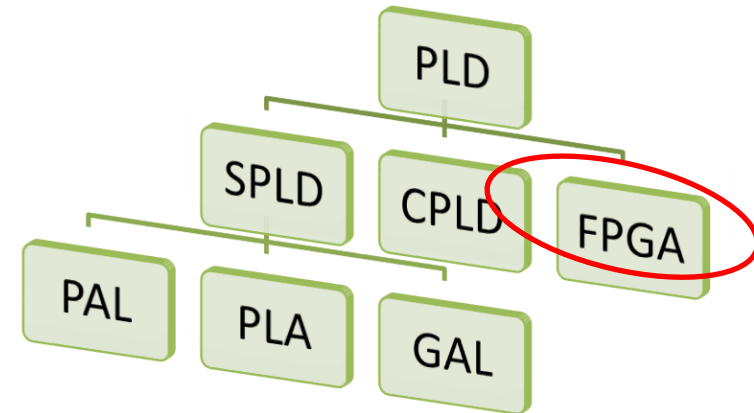
- Circuitos integrados formados por:

- ☐ puertas lógicas
- ☐ módulos básicos
- ☐ y/o biestables
- ☐ etc.

- Disponen de **conexiones flexibles** para ser programadas/conectados por el fabricante o usuario.

- Ventajas:

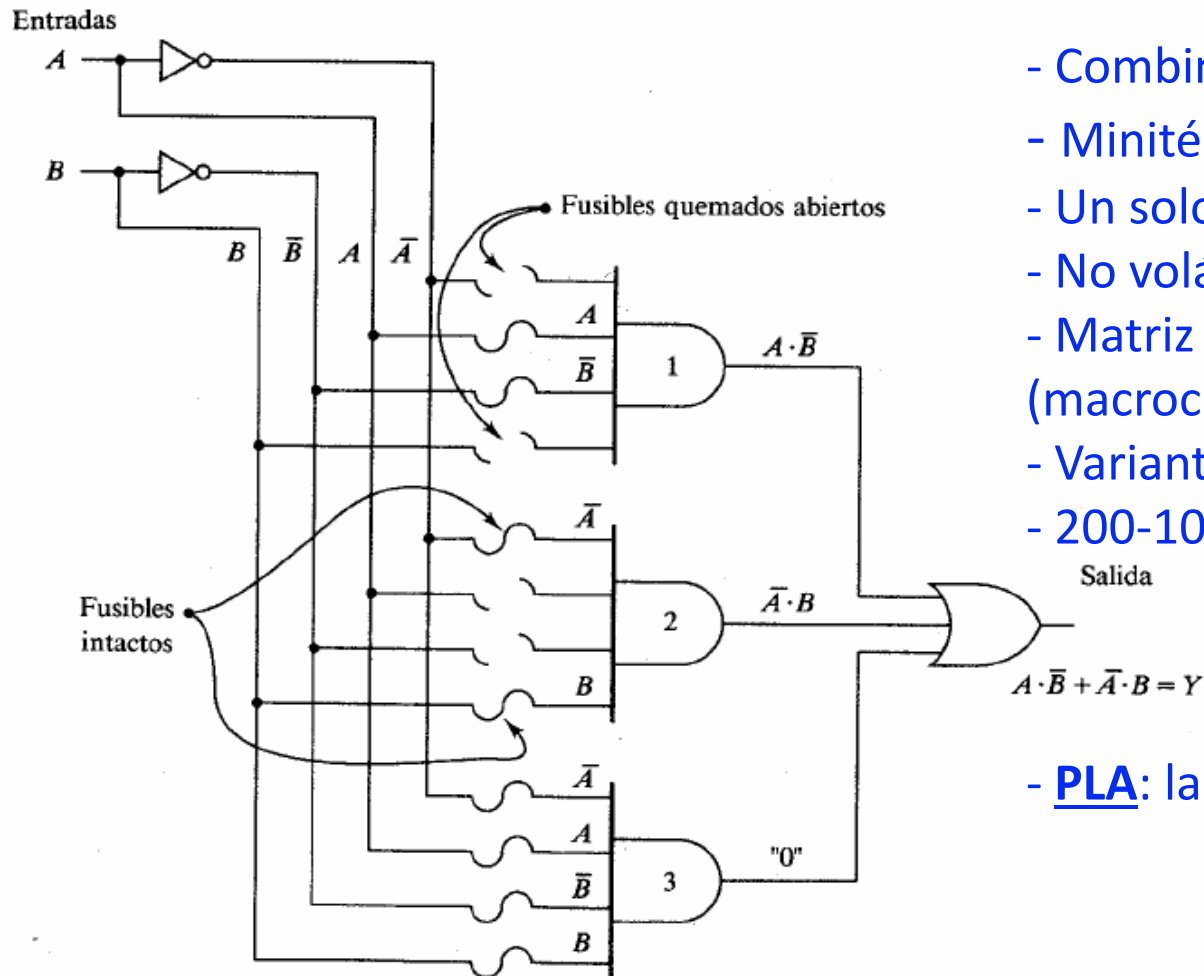
- ✓ Alto volumen de producción → Bajo coste
- ✓ Consumo medio
- ✓ Velocidad media
- ✓ Tiempo de desarrollo bajo y sencillo
- ✓ El diseñador preserva su trabajo



3. Dispositivos lógicos programables (PLD)

$$Y = \overline{A}\overline{B} + B\overline{A}$$

PAL (*Programmable Array Logic*)



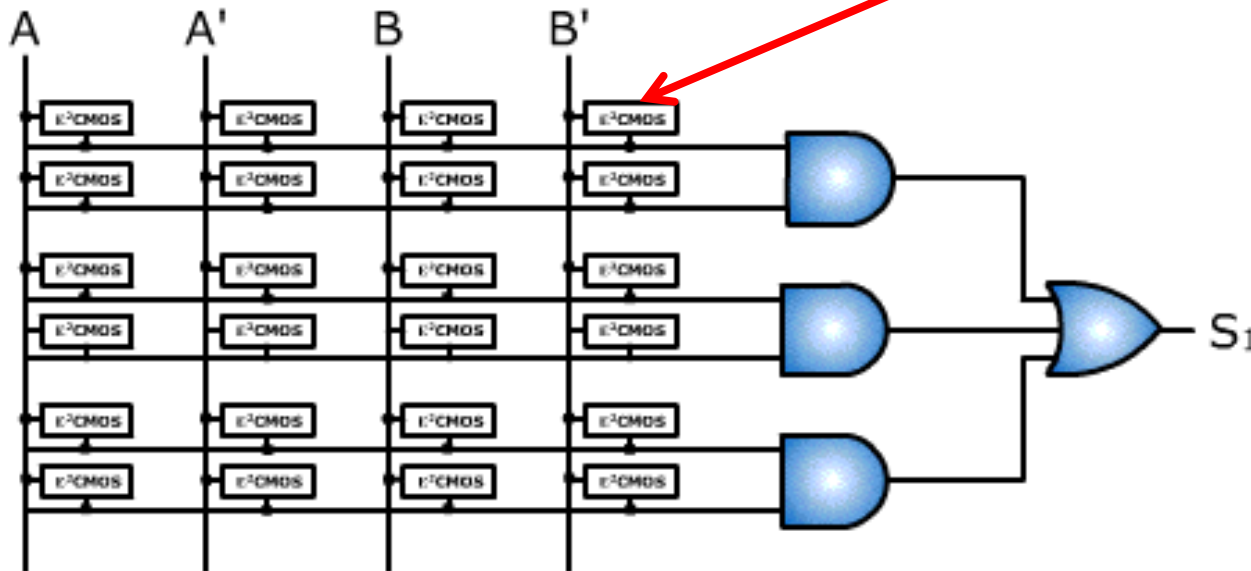
- Combinacionales
- Minitérminos → Suma de productos
- Un solo uso
- No volátil
- Matriz OR fija / AND programable (macrocelda)
- Variante con salidas registradas
- 200-1000 puertas
- PLA: la matriz OR también es programable

3. Dispositivos lógicos programables (PLD)

GAL (Generic Array Logic)

- Minitérminos → Suma de productos
- Reprogramable tecnología *ECMOS* (electricidad, ultravioleta)
- No volátil
- Matriz OR fija / AND reprogramable
- Salidas combinacionales o registradas
- 200-1000 puertas

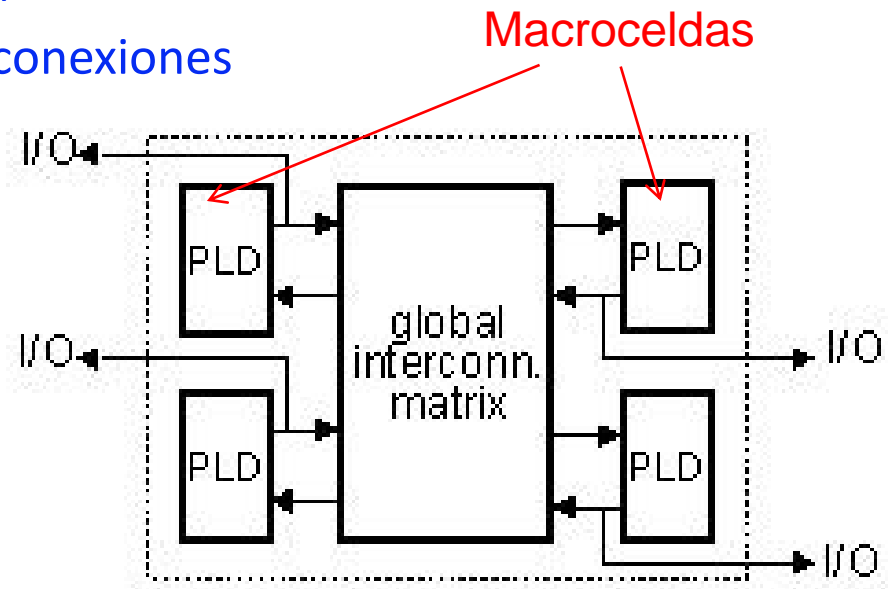
Sustituyen a los fusibles



3. Dispositivos lógicos programables (PLD)

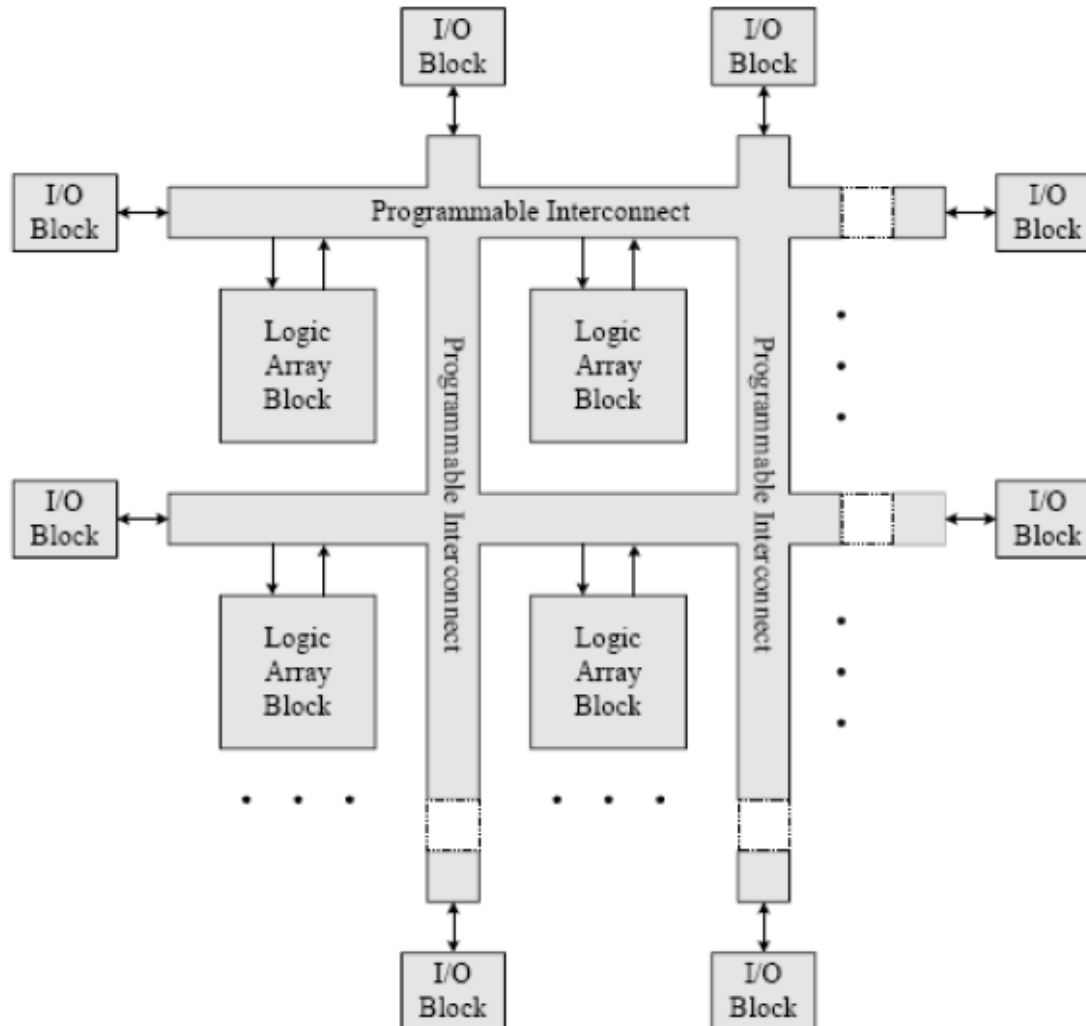
CPLD (*Complex Programmable Logic Device*)

- Programmable/Reprogrammable
- Volátil/No volátil
- Grupos bloques lógicos interconectados por una sola matriz de interconexiones
- 1 bloque lógico = varias macroceldas PAL/GAL
- Se programan los PLD (CMOS) y las interconexiones (fusible, EEPROM) → NO ISP (SRAM, flash) → ISP
- 1.000-10.000 puertas
- ISP (*In-System Programming*)
- Requiere entorno de desarrollo



3. Dispositivos lógicos programables (PLD)

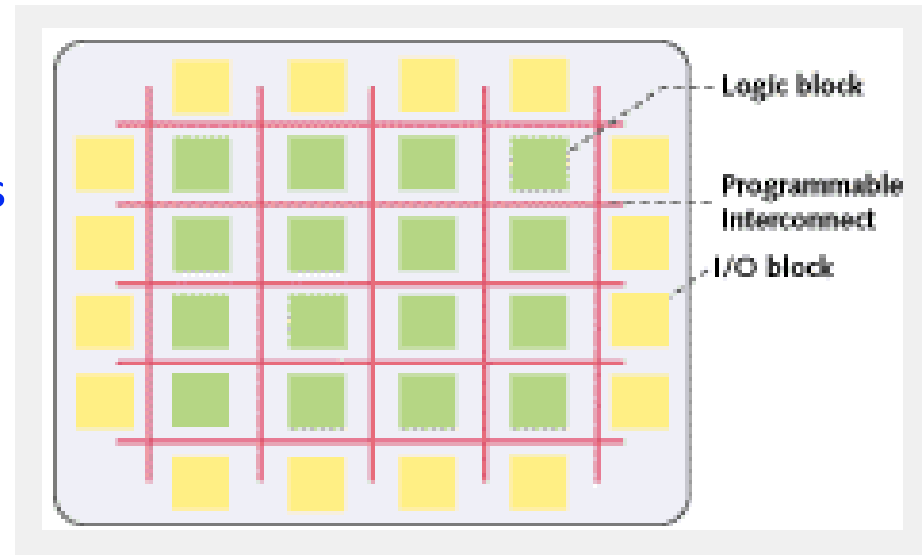
CPLD (Complex Programmable Logic Device)



3. Dispositivos lógicos programables (PLD)

FPGA (*Field Programmable Logic Array*)

- Programable/Reprogramable
- Volátil/No volátil
- Alterna bloques lógicos (más o menos complejos) con interruptores de interconexión
- Interconexiones (fusible, sram, flash)
- 10.000 – 10.000.000 puertas
- ISP (*In-System Programming*)
- Requiere entorno de desarrollo
- No restringido a suma de productos
- Pueden incluir otros recursos:
 - Multiplexores
 - Multiplicadores
 - Memorias,
 - etc.



3. Dispositivos lógicos programables (PLD)

FPGA (Field Programmable Logic Array)

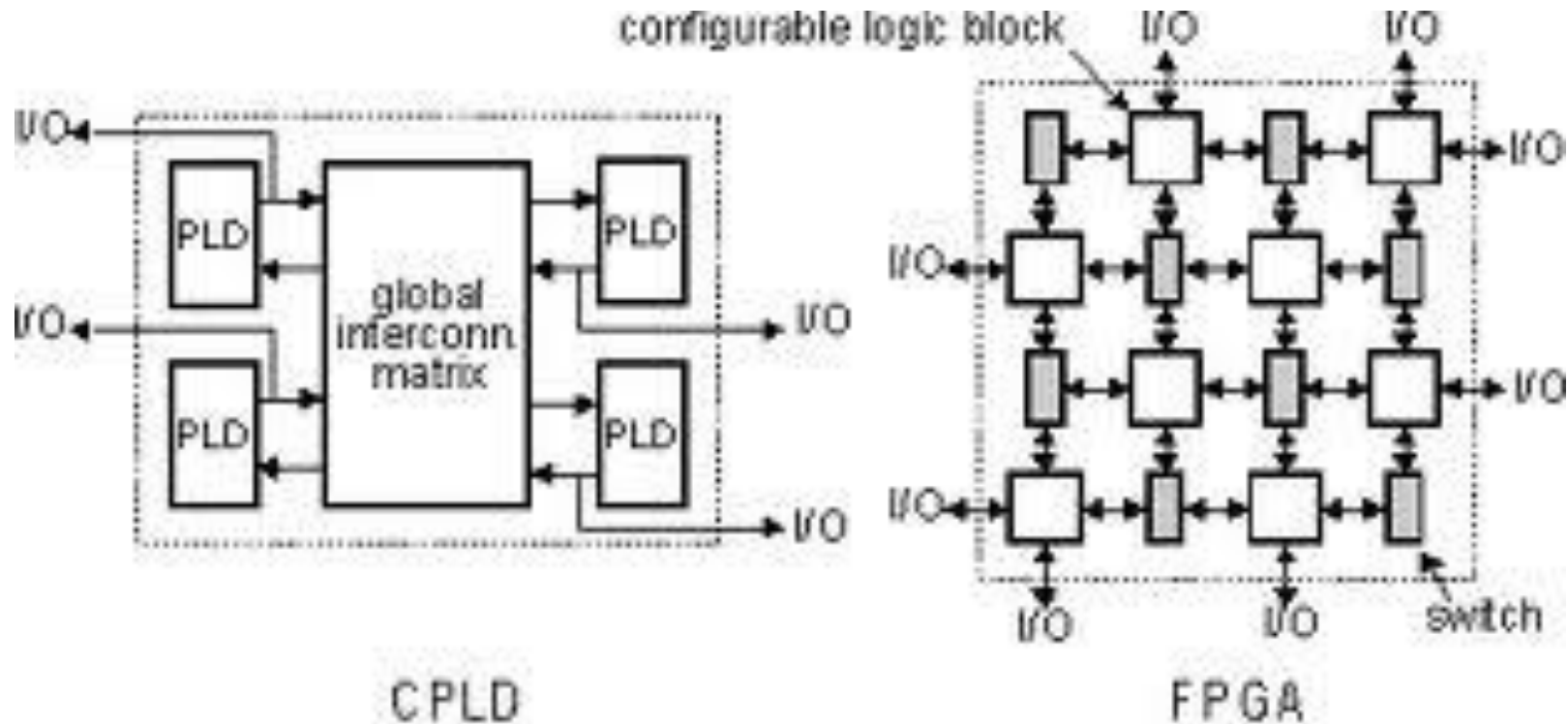
-Bloque lógico:

-Grano fino (puertas lógicas)

-Grano grueso (Mux, LUTs, etc)

CPLD	Bloque lógico	Las funciones se implementan utilizando AND/OR de dos niveles.	Puertas AND con muchas entradas
	Interconexiones	Todos con todos. Cada salida del BL es interconectable a cada entrada con 1 o 2 interruptores máximo.	Tiempo predecible
FPGA	Bloque lógico	Más niveles de lógica	Bloques más sencillos
	Interconexiones	Segmentadas.	Varios interruptores por interconexión

3. Dispositivos lógicos programables (PLD)



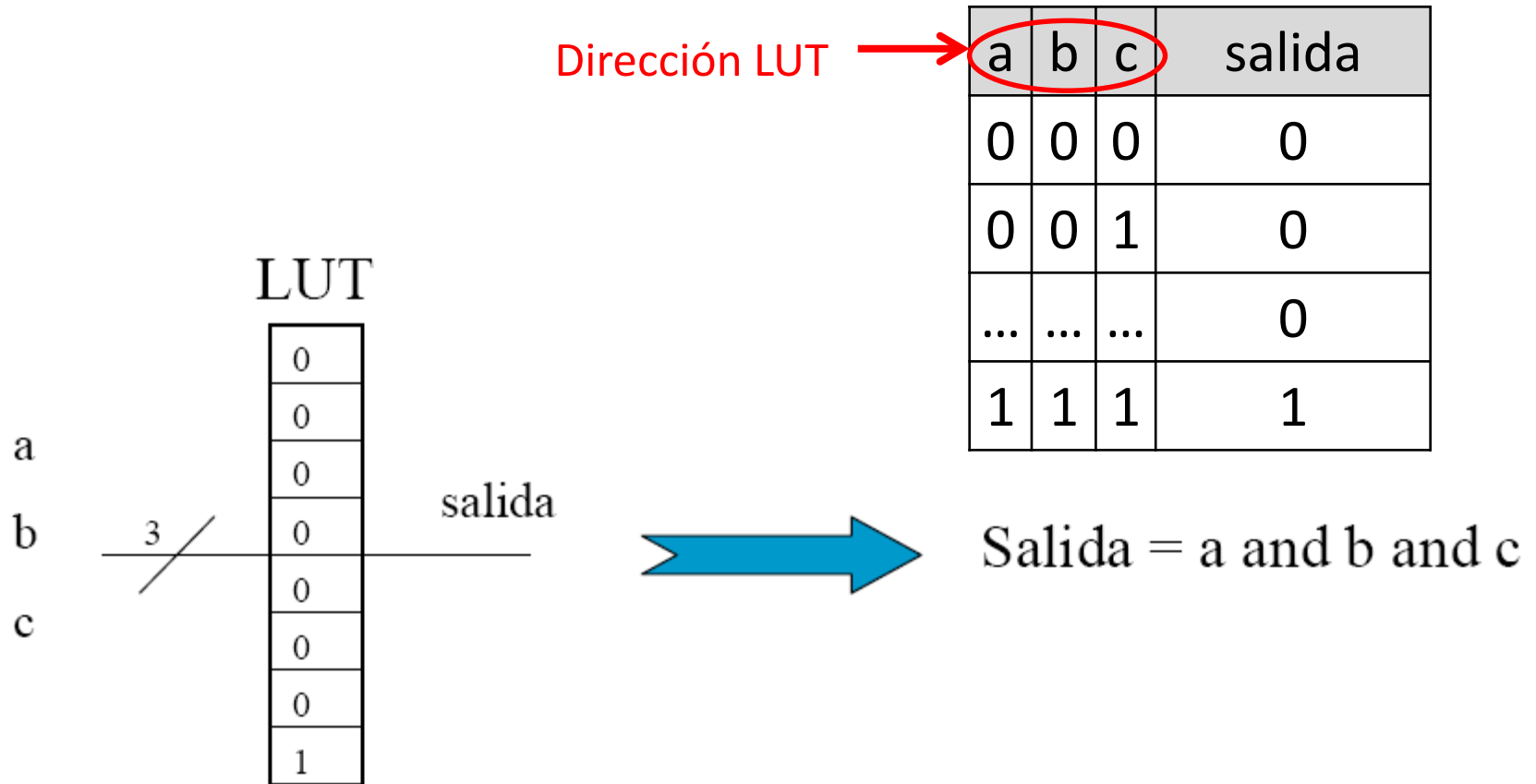
CPLD and FPGA Architectures

3. PLD versus CI estándar SSI/MSI

- Disminuye el **COSTE** del circuito
 - ✓ Se reduce el número de chips y por tanto el tamaño de las placas
 - ✓ aumento de la velocidad de funcionamiento del circuito.
 - ✓ aumento de la fiabilidad (disminución de interconexiones)
 - ✓ reducción del coste del circuito (área de PCB)
 - ✓ reducción del consumo
- **FLEXIBILIDAD**
 - ✓ se puede cambiar la funcionalidad del diseño sin modificar el PCB.
- Es más **DIFÍCIL DE COPIAR**(propiedad intelectual)
- Simplifica el trabajo del diseñador
 - ✓ Permite utilizar **HERRAMIENTAS DE DISEÑO (ECAD)**.

3. FPGA Artix7 - Xilinx

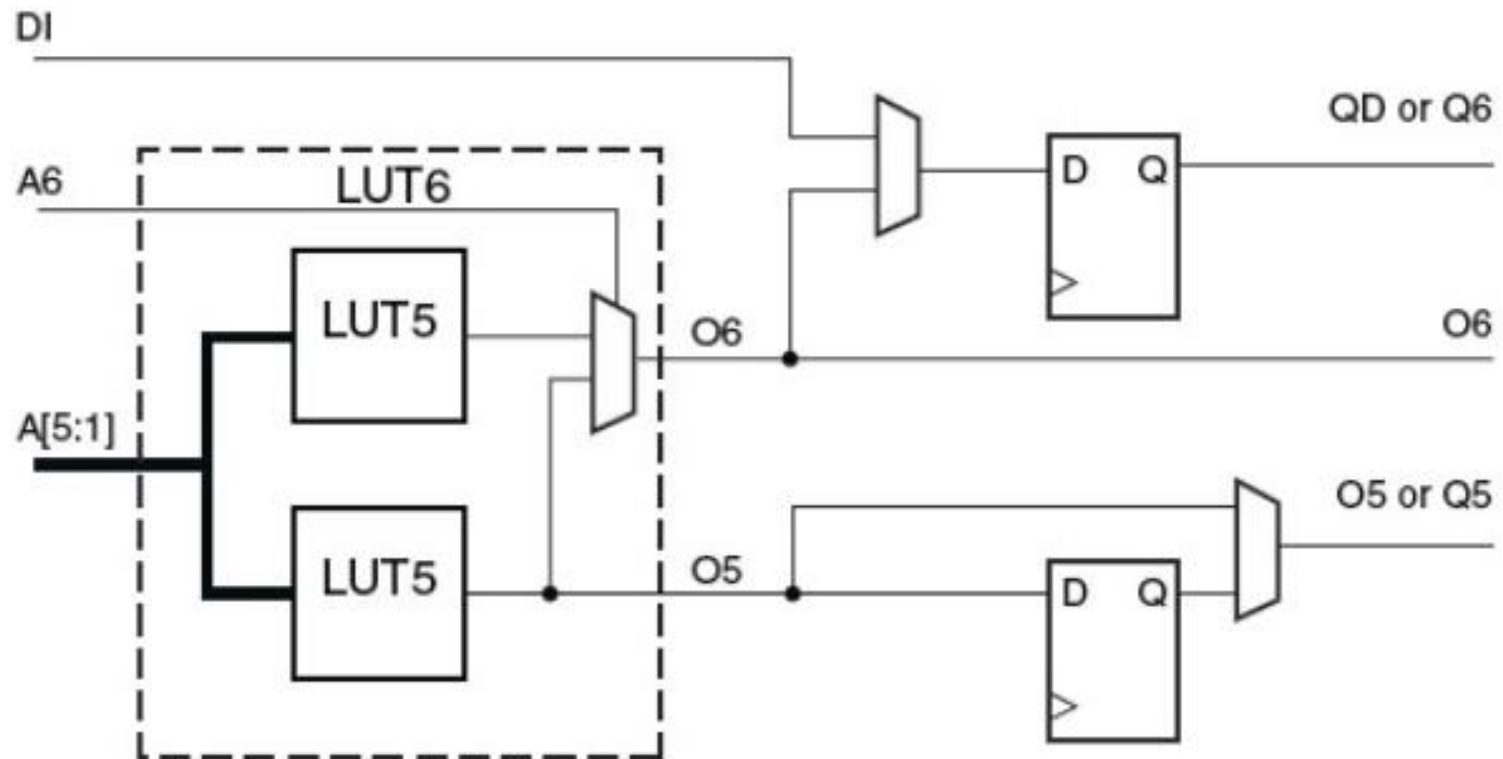
LOOK-UP TABLE (LUT). Tabla de valores preasignados



3. FPGA Artix7 - Xilinx

SLICE: Contiene

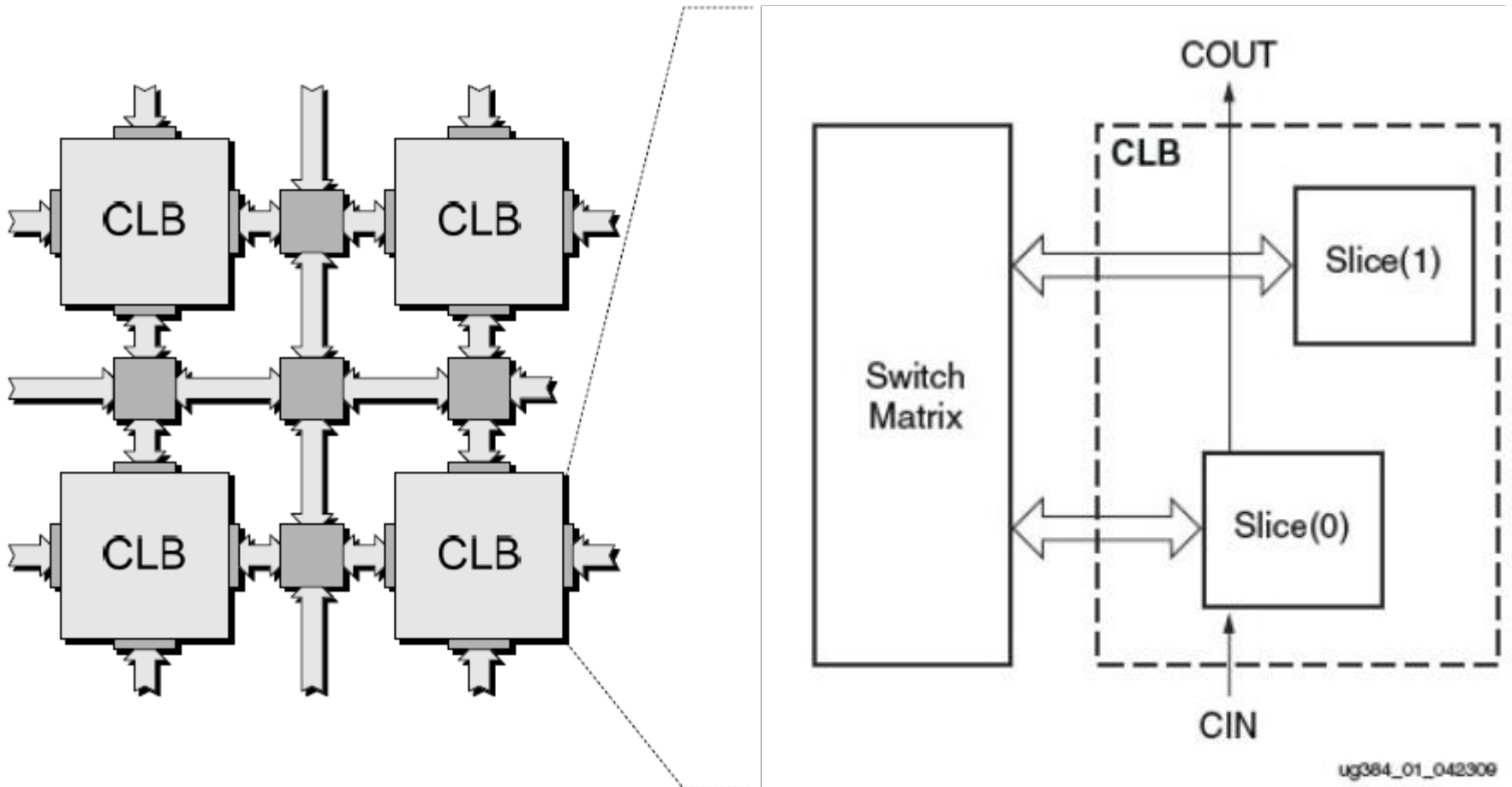
- 4 LUT (6-inputs)
- 8 FFs



UG384_06new_021210

3. FPGA Artix 7 - Xilinx

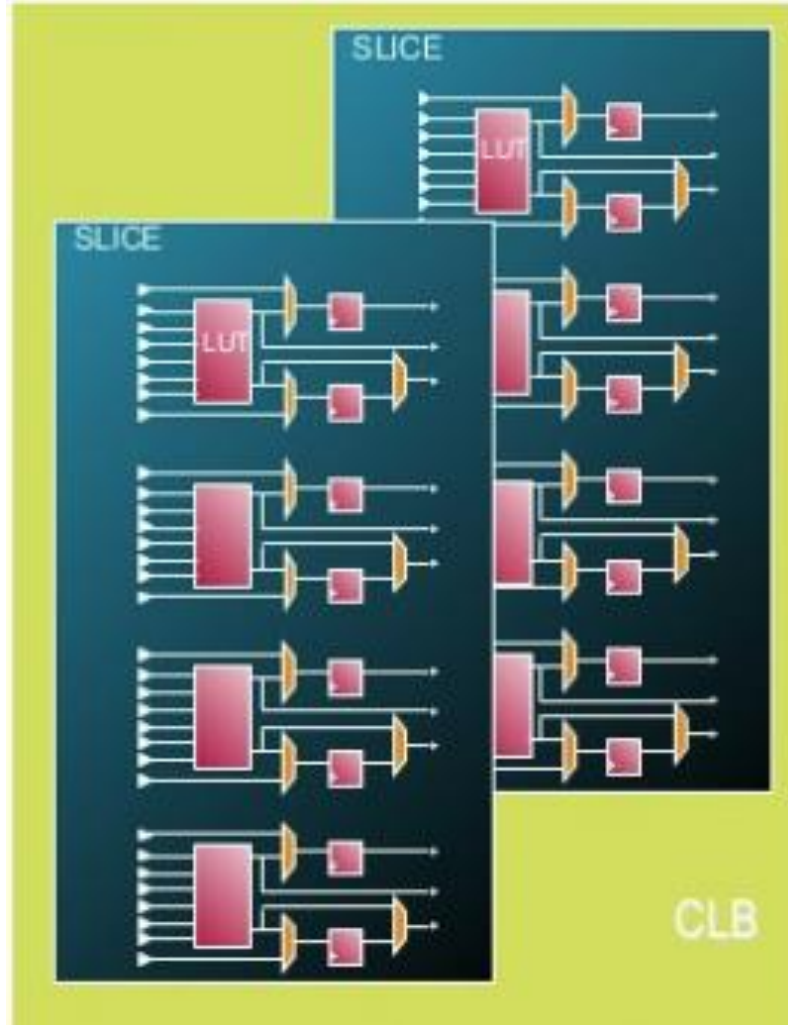
CLB: Bloque lógico configurable. Contiene 2 Slices



C. Maxfield

3. FPGA Spartan 3 - Xilinx

CLB: Bloque lógico configurable. Contiene 2 Slices



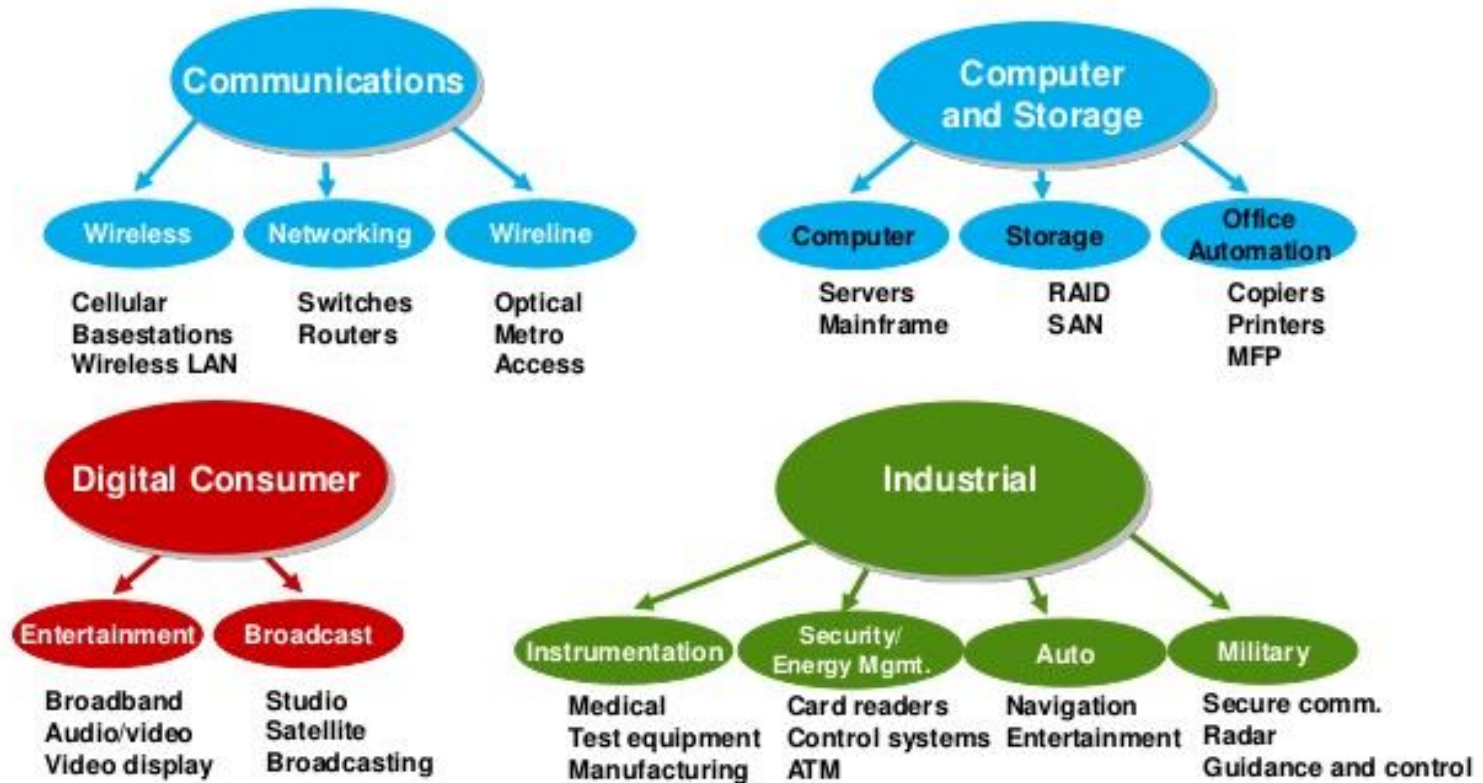
3. Fabricantes de FPGA

	2015		2016		
Vendor	FPGA Total	Market share	FPGA Total	Market share	Growth CY15-CY16
Xilinx	\$2,044	53%	\$2,167	53%	6%
Intel (Altera)	\$1,389	36%	\$1,486	36%	7%
Microsemi	\$301	8%	\$297	7%	-1%
Lattice	\$124	3%	\$144	3%	16%
QuickLogic	\$19	0%	\$11	0%	-40%
Others	\$2	0%	\$2	0%	0%
TOTAL	\$3,879	100%	\$4,112	100%	6%

Source: EE Times

3. Fabricantes de FPGA

FPGA End Markets



ALTERA

3. ¿Dónde hay FPGAs?

Perspectiva Ingeniería Electrónica



Rovers Mars Exploration

<http://www.xilinx.com/about/customer-innovation/aerospace-and-defense/mars-exploration-rovers.htm>



F1 BMW FW25





<http://me.queensu.ca/Courses/452/reference13.pdf>



3. ¿Dónde hay FPGAs?

Perspectiva Ingeniería Informática

FPGA versus GPU

Low Latency	Connectivity	Engineering cost	Energy Efficiency	
Autopilotaje de un jet. Guiado de misiles	Radio astronomía. Grandes cantidades de datos		Minado de criptomonedas	
Aprox. 1 us, uP aprox 50 us	Conexión directa sensor-pin-mayor ancho de banda	Difícil de programar (HDL)	En aplicaciones de coma fija bien.	
Más determinista al no depender de SO		Largos tiempos de compilación	No requiere de un host como las GPU (ahorro)	
FPGA 	FPGA 	FPGA 	FPGA 	

3. ¿Dónde hay FPGAs?

Ingeniería Software

Intel →

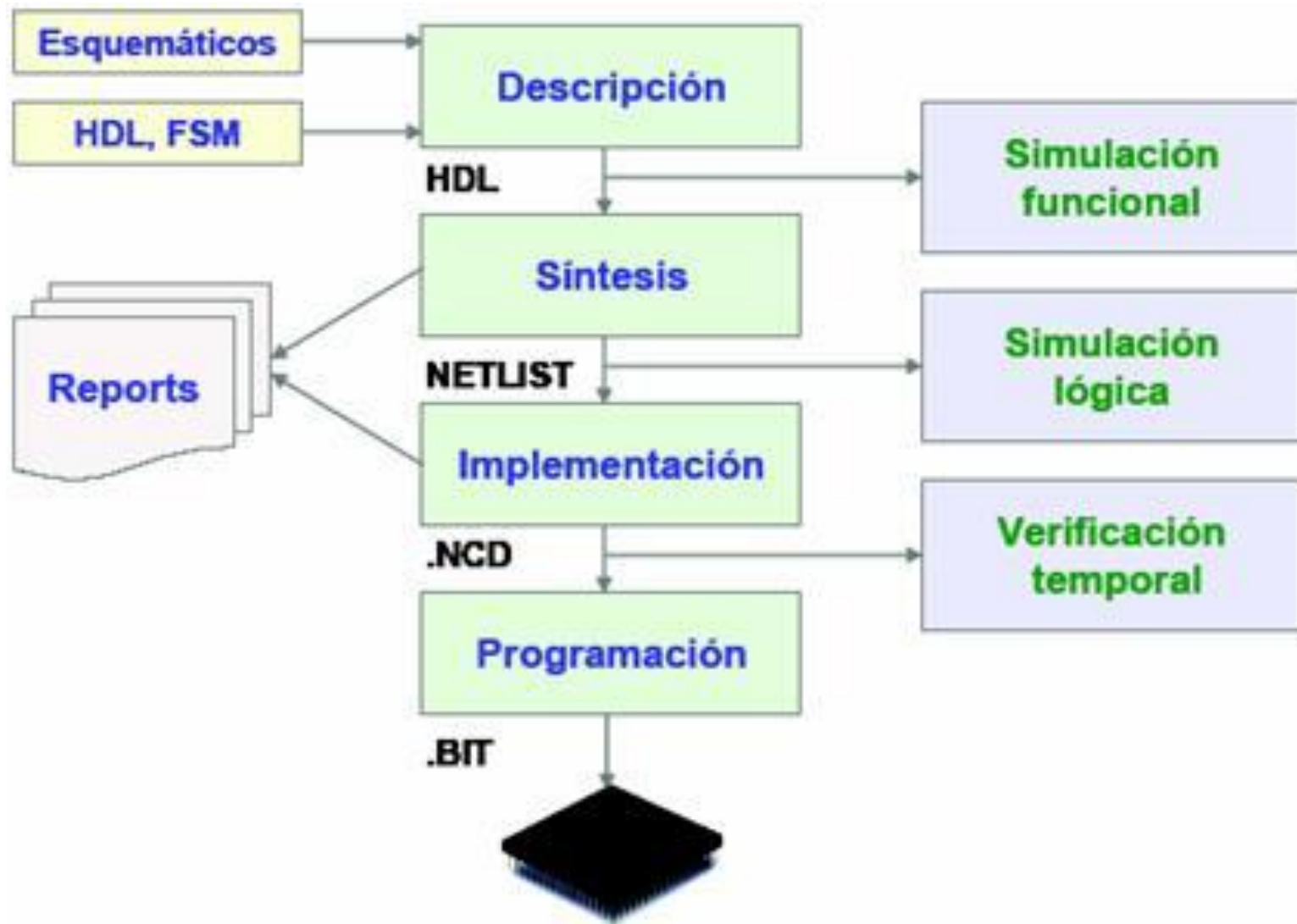
- Compra Altera 16 billones \$
- Objetivos:
 - 1.- High Performance computing (GPU)
 - 2.- Clouds providers (Descargar en FPGA ciertos trabajos en vez de en los servidores. Amazon EC2 instances)
- Colabora con Microsoft → 50% mejora en BING

Vivado HLS → OpenCL

<https://blog.esciencecenter.nl/why-use-an-fpga-instead-of-a-cpu-or-gpu-b234cd4f309c>

4. Entorno de desarrollo (Vivado Webpack)

4. Entorno de desarrollo (Flujo de diseño)



4. Entorno de desarrollo (Flujo de diseño)

Síntesis lógica

Descripción VHDL

```
architecture MLU_DATAFLOW of MLU is
  signal A1:STD_LOGIC;
  signal B1:STD_LOGIC;
  signal Y1:STD_LOGIC;
  signal MUX_0, MUX_1, MUX_2, MUX_3: STD_LOGIC;

begin
  A1<=A when (NEG_A='0') else
    not A;
  B1<=B when (NEG_B='0') else
    not B;
  Y1<=Y1 when (NEG_Y='0') else
    not Y1;

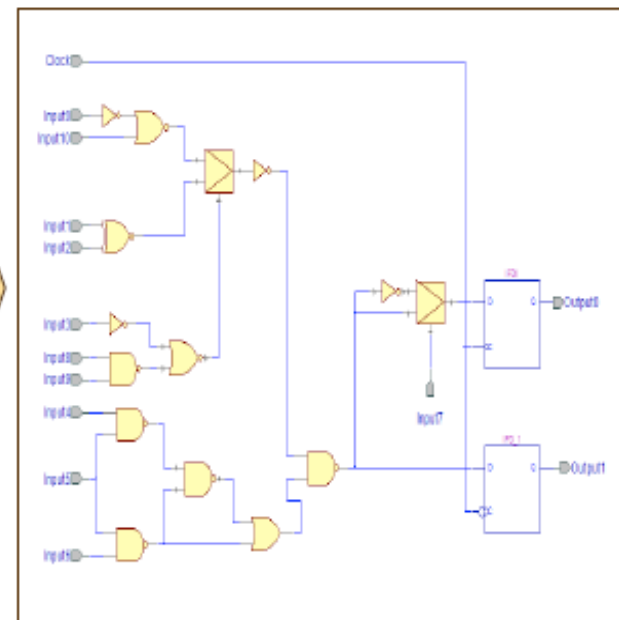
  MUX_0<=A1 and B1;
  MUX_1<=A1 or B1;
  MUX_2<=A1 xor B1;
  MUX_3<=A1 xnor B1;

  with (L1 & L0) select
    Y1<=MUX_0 when "00",
      MUX_1 when "01",
      MUX_2 when "10",
      MUX_3 when others;

end MLU_DATAFLOW;
```



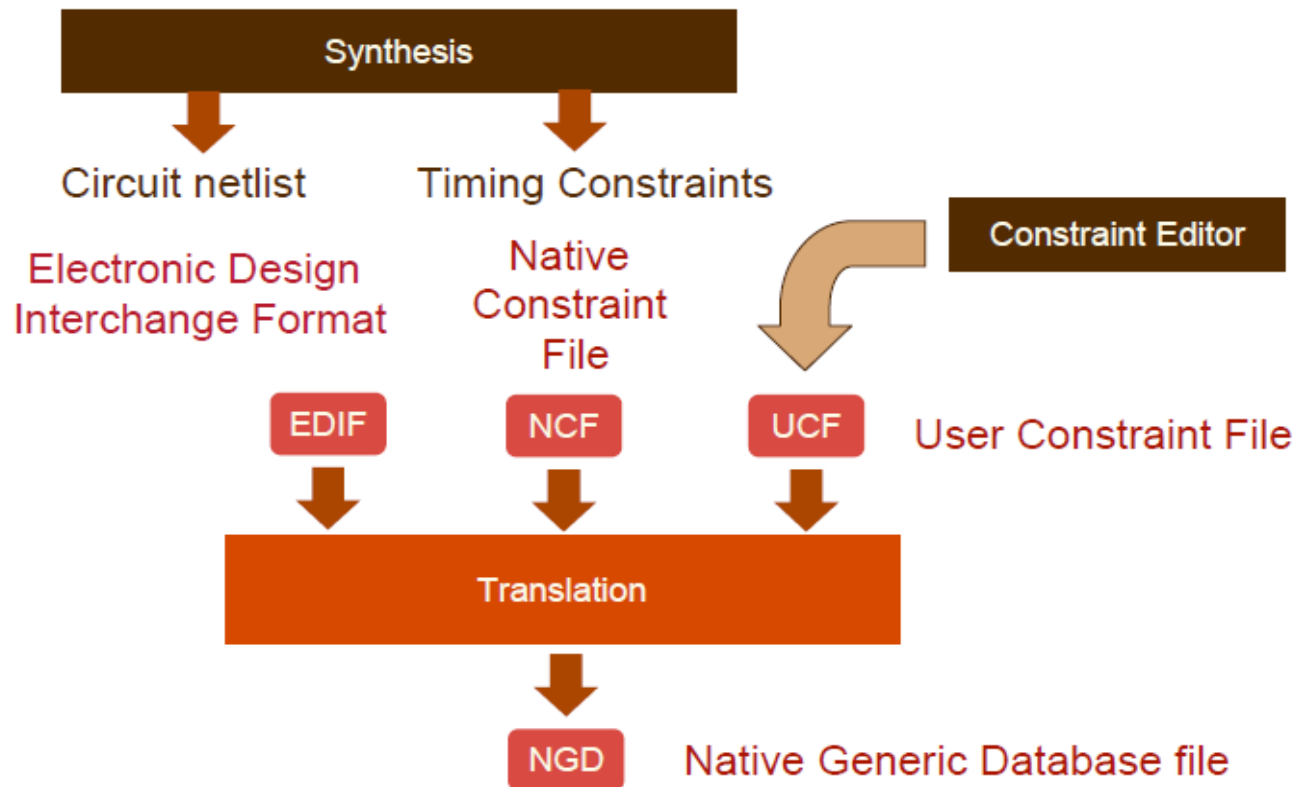
Netlist circuito



4. Entorno de desarrollo (ISE-WebPack)

Implementación: Etapa 1

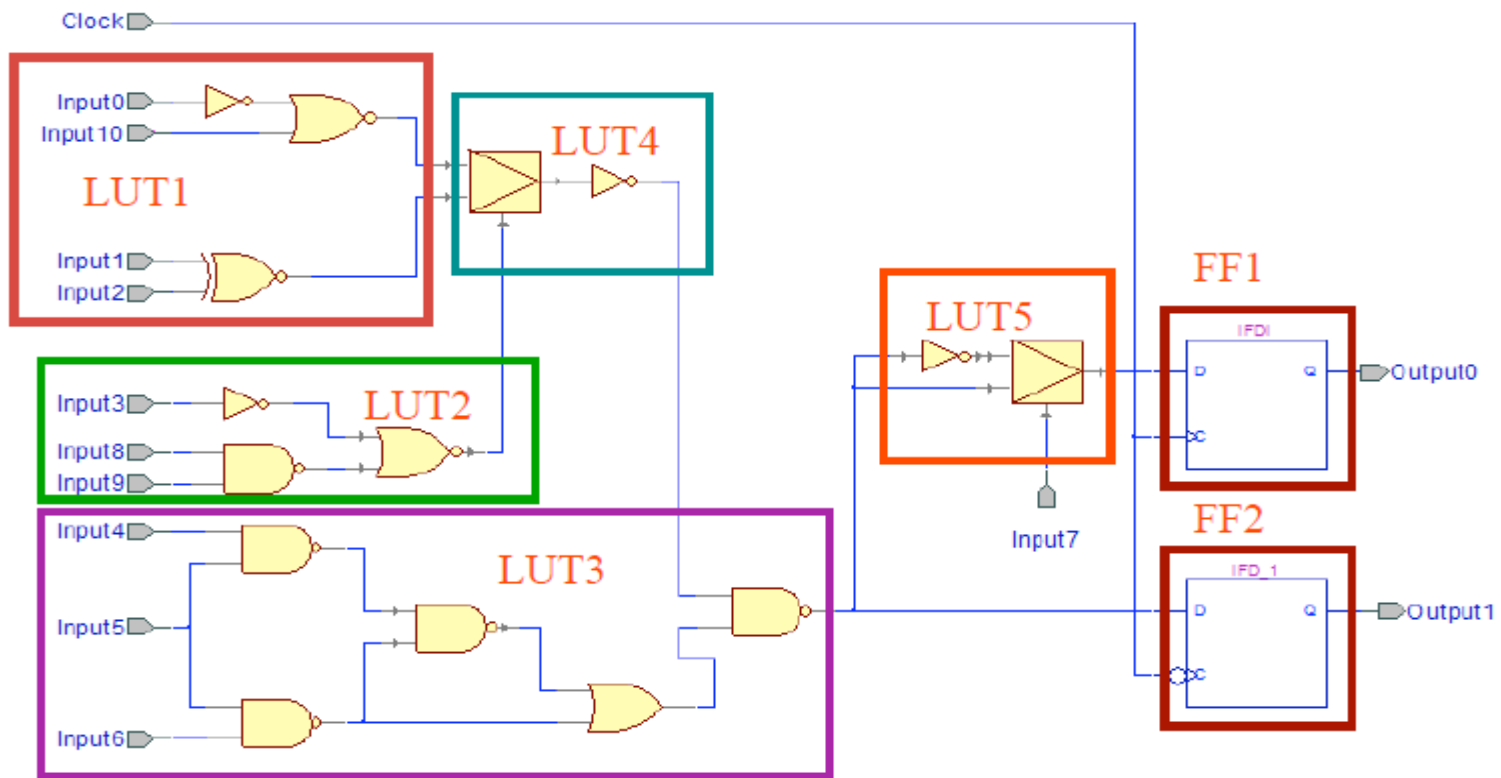
Translation



4. Entorno de desarrollo (ISE-WebPack)

Implementación: Etapa 2

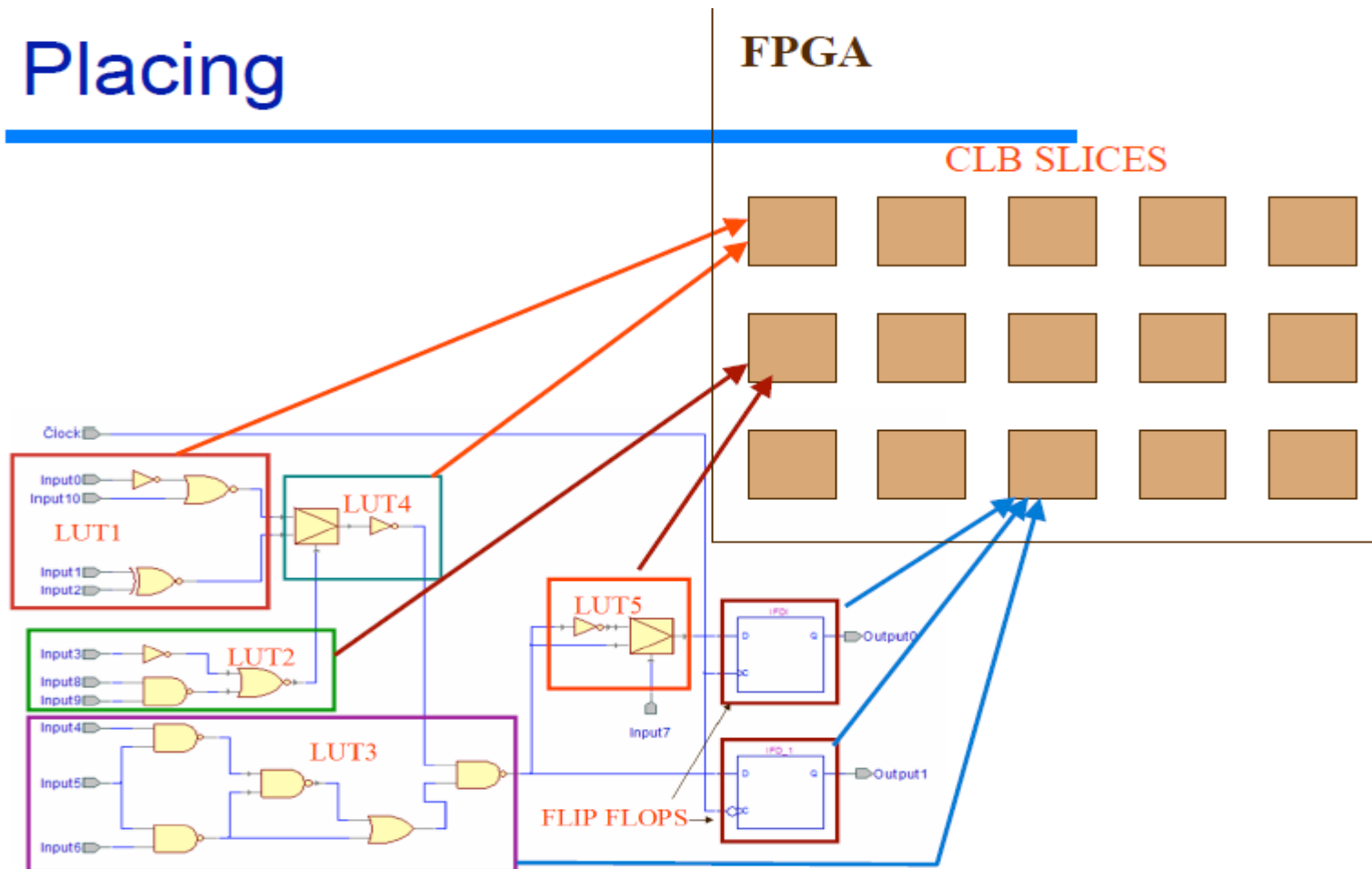
Mapping



4. Entorno de desarrollo (ISE-WebPack)

Implementación: Etapa 3

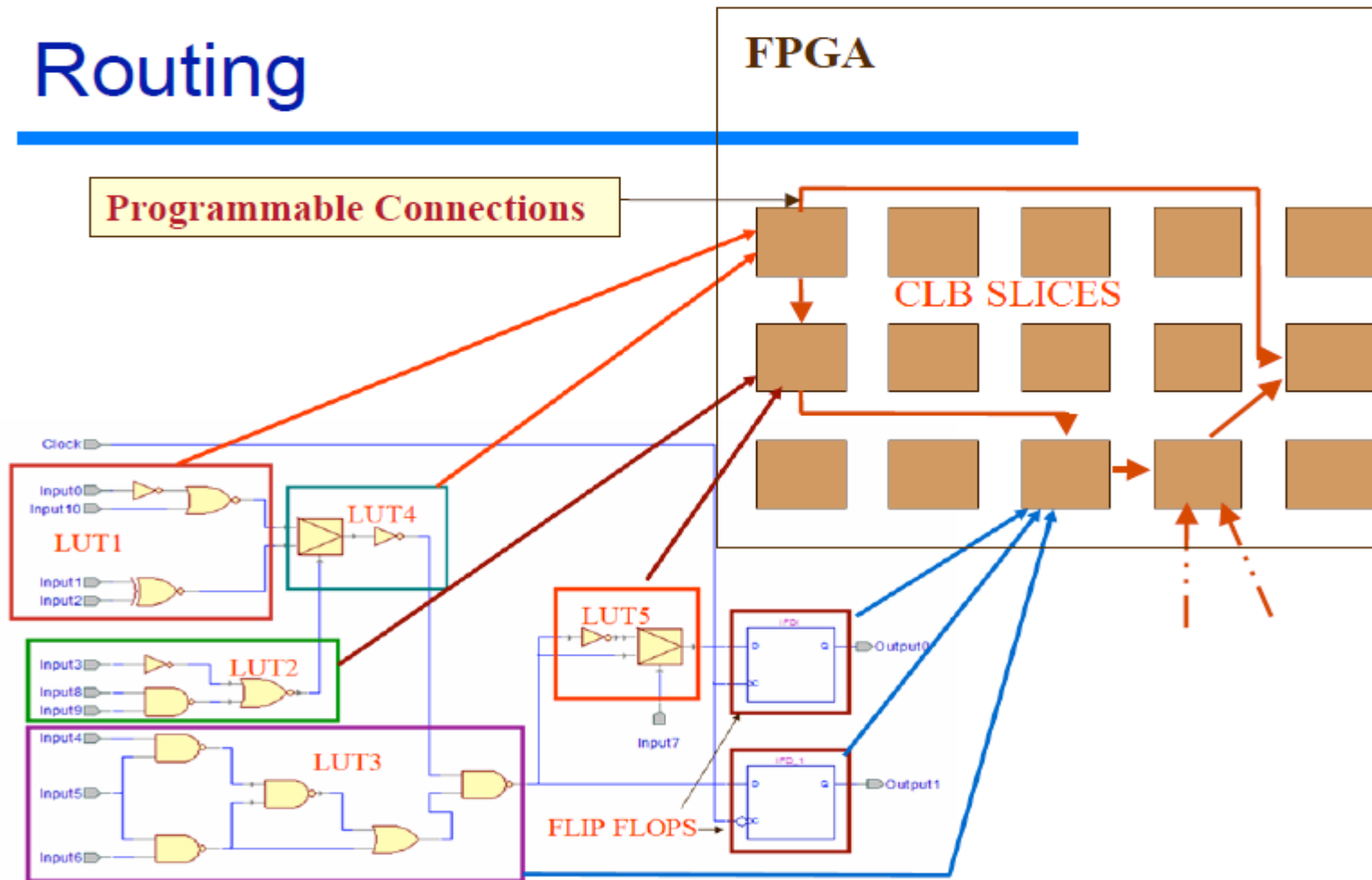
Placing



4. Entorno de desarrollo (ISE-WebPack)

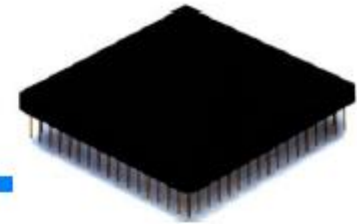
Implementación: Etapa 4

Routing



4. Entorno de desarrollo (ISE-WebPack)

Configuration



- Una vez implementado el diseño, se debe generar un fichero para configuración de la FPGA.
 - bit stream (.bit)
- El fichero BIT puede ser utilizado directamente con la FPGA o puede ser convertido a un fichero PROM para almacenar la información de programación.



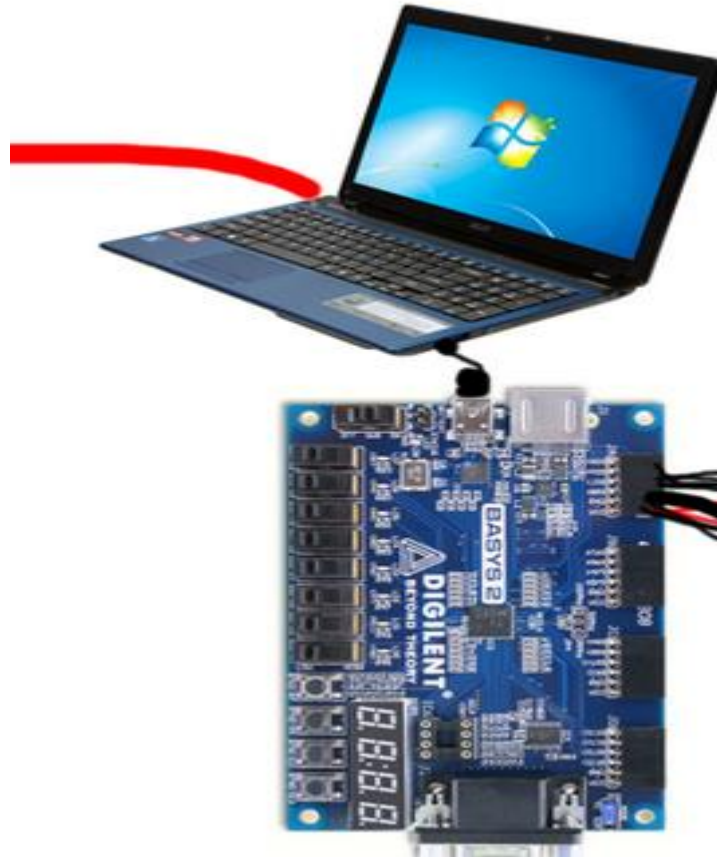
*Download desde el PC
a través del puerto paralelo*



**Lectura de la configuración
almacenada en una memoria
(EPROM serie o paralela)**

4. Entorno de desarrollo (ISE-WebPack)

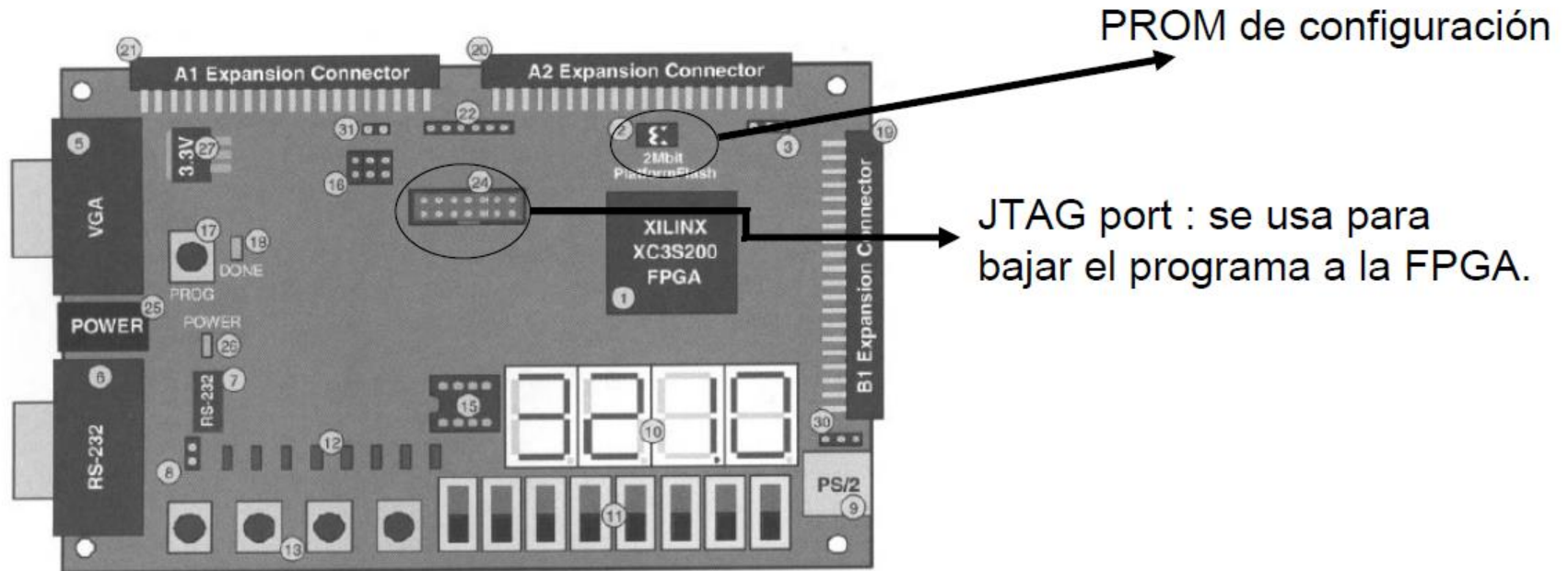
CONFIGURACIÓN



5. Placa de evaluación (Nexys4-DDR)

5. Placa de evaluación (Nexys4-DDR)

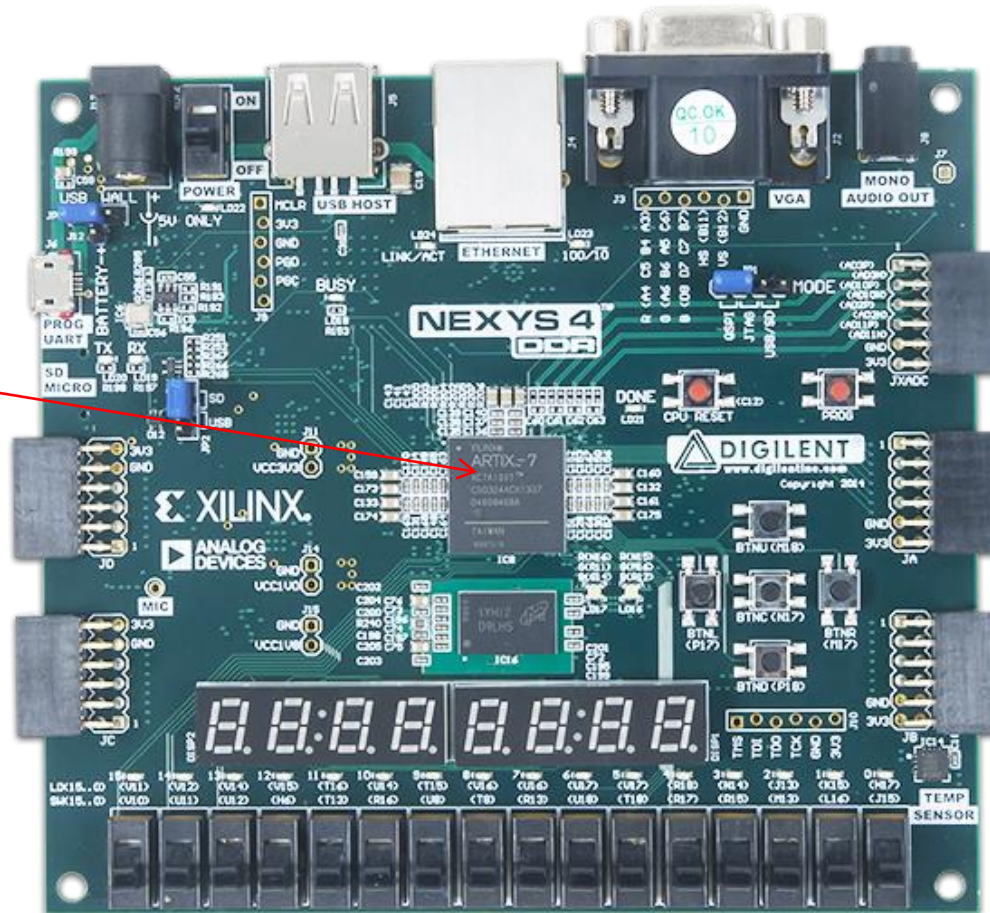
- Utilizamos una placa de prototipado que contiene un dispositivo FPGA.
- En nuestro caso, utilizamos una placa de la empresa Digilent que contiene una FPGA de Xilinx modelo Spartan 3.



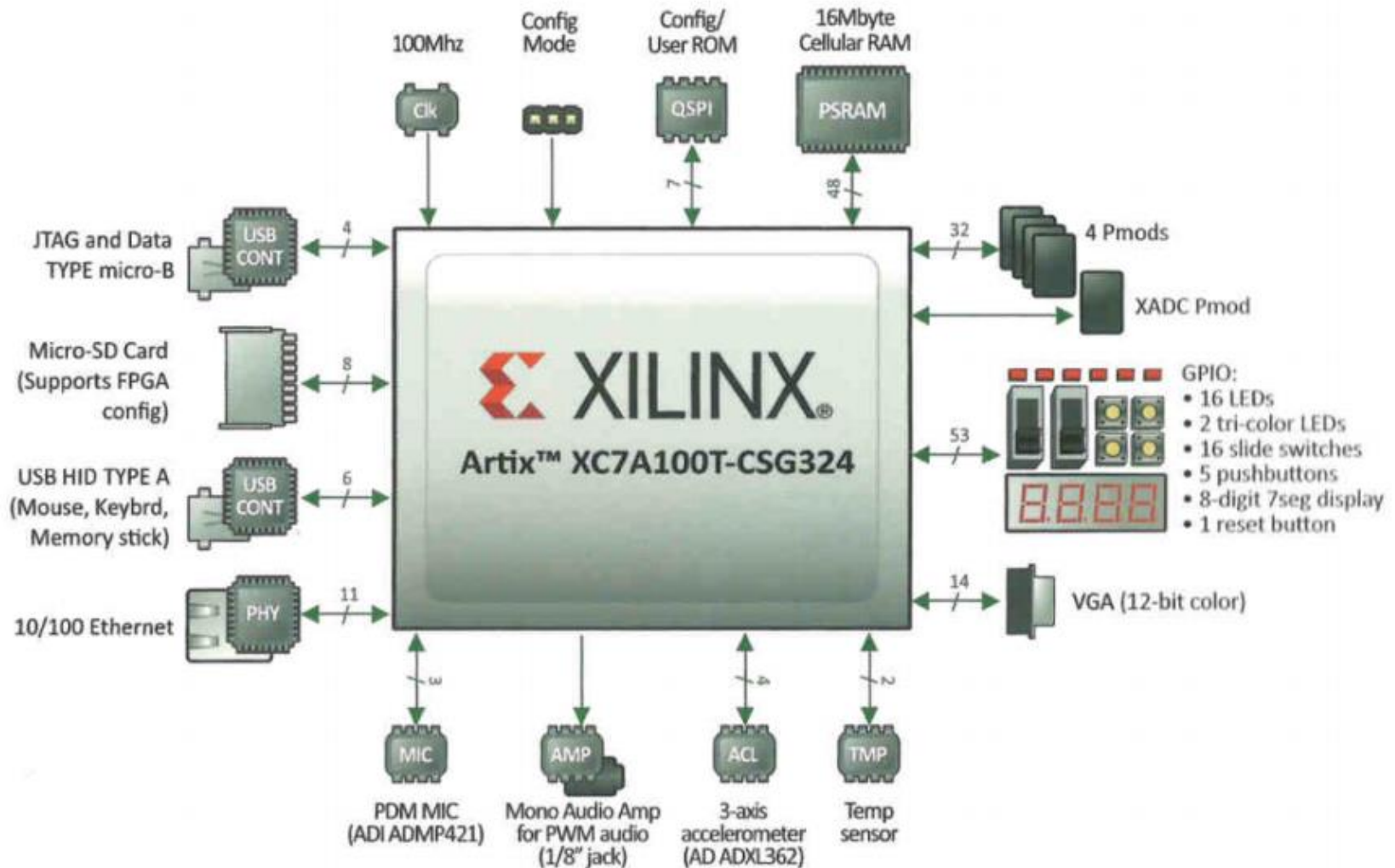
(a) Top view

5. Placa de evaluación (Nexys4-DDR)

FPGA



5. Placa de evaluación (Nexys4-DDR)



Bibliografía del tema

- The Design Warrior's Guide to FPGA. Clive Maxfield. Elsevier, 2004.
- FPGAs 101. Gina R. Smith. Ed. Newnes, 2010.
- www.digilent.com
- www.xilinx.com

Bibliografía del tema (Video)

- <https://www.xilinx.com/video/fpga/7-series-clb-architecture.html>