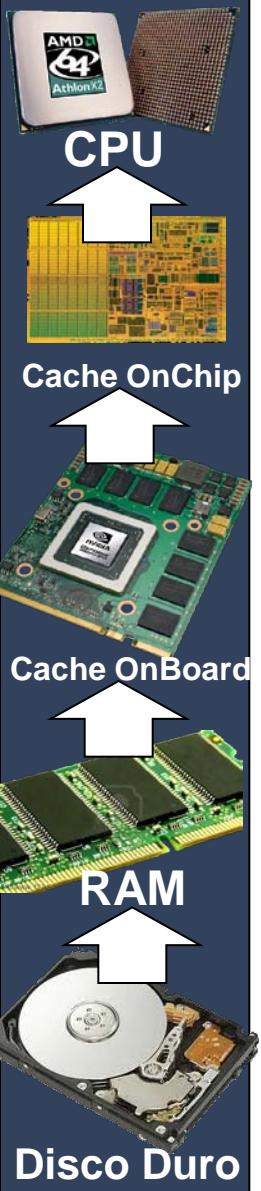




# DISEÑO AUTOMÁTICO DE SISTEMAS

## Curso 2013-2014



# Introducción

**Profesor teoría y laboratorio  
José Miguel Montaña Aliaga.**

e-mail: [jmontanana@fdi.ucm.es](mailto:jmontanana@fdi.ucm.es)

**Tutorías: 1er cuatrimestre Miércoles : 10:00-13:00**

**2ndo cuatrimestre Miércoles 14:00-17:00**

**Fac Informática dpch 421, Fac. Física dpcho 225**

**Apuntes: Campus Virtual**

**D E P A R T A M E N T O D E  
A R Q U I T E C T U R A D E C O M P U T A D O R E S  
Y A U T O M Á T I C A**

Dpto. de Arquitectura de Computadores y Automática

Facultad de Informática.

Universidad Complutense de Madrid.



# DISEÑO AUTOMÁTICO DE SISTEMAS

## Curso 2013-2014



## LABORATORIO LIAM



Ricerca e sviluppo per l'innovazione nel campo  
dell'automazione di macchine automatiche per il packaging

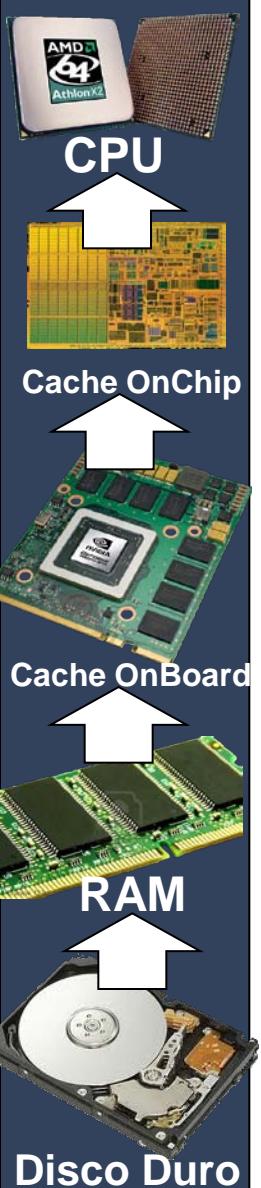


Innovation Days 3 - Open Innovation



# DISEÑO AUTOMÁTICO DE SISTEMAS

## Curso 2013-2014



## PROFILO AZIENDALE SCOPO E COMPAGINE SOCIALE

Fornire supporto alle imprese per  
l'innovazione e il trasferimento di tecnologia



Analisi e produzione di  
informazione tecnologica

Pratica della  
“innovazione collaborativa”

18/04/2013

Innovation Days 3 - Open Innovation

### Competencias de la asignatura:

Capacidad de conocer, comprender y evaluar la estructura y arquitectura de los computadores.

Capacidad de diseñar y construir sistemas digitales como son sistemas basados en **microprocesador** y **sistemas de comunicaciones**.

Capacidad de desarrollar **procesadores** específicos y sistemas empotrados, así como desarrollar y optimizar el **software** de dichos sistemas.

Capacidad de analizar, evaluar y seleccionar las plataformas hardware y software más adecuadas para:  
el soporte de aplicaciones **empotradas** y de **tiempo real**.

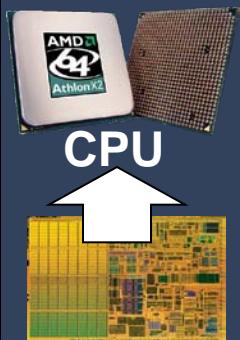
#### Método:

**FPGAs y software de Xilinx ISE.**

**Diseño a nivel lógico-RT de sistemas usando VHDL.**

**Análisis y síntesis de sistemas digitales a nivel lógico-RT**

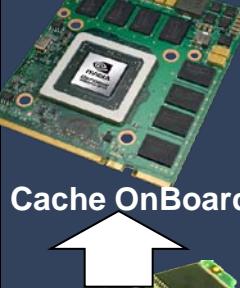




CPU



## Cache OnChip



# RAM



# Disco Duro

LTC (sesiones de clase)			
2 Octubre	0, Introducción	11-Noviembre	7. Pendiente de definir
4 Octubre	1, Diseño de sistemas	13-Noviembre	8.- Memorias estáticas y dinámicas
9 Octubre	2, Temporización y maquinas de estados (intro VHDL)	20-Noviembre	9.- Gráficos VGA
16 Octubre	3, Especificación nivel lógico, (repaso práctica 1)	25-Noviembre	9. Arquitectura de procesador
23 Octubre	4. Entrada salida E/S, ejemplo ps/2	27-Noviembre	10. Circuito Asíncrono (FPGA)
30 Octubre	5. Técnicas de diseño	4-Diciembre	Plan proyectos fin de curso
6-Noviembre	6. Modulación de sonido	18-Diciembre	X- examen evaluación continua teoría 2013-14

### MÉTODO DE EVALUACIÓN:

#### Primer filtro:

Asistencia a los turnos de prácticas (3 faltas máximo)

Control de la realización de las mismas por el profesor

#### Segundo filtro:

Los que cumplan los requisitos anteriores realizarán una PRACTICA FINAL

La **NOTA FINAL** de cada parcial se compondrá de:

**Exámenes sobre la materia: 0-60%**

- EXAMEN

**Otras actividades: 40-100%**

- prácticas en el laboratorio (3 puntos)
- PRACTICA FINAL en junio (7 puntos).

Para aprobar el parcial hay que tener un 40% en cada parte.

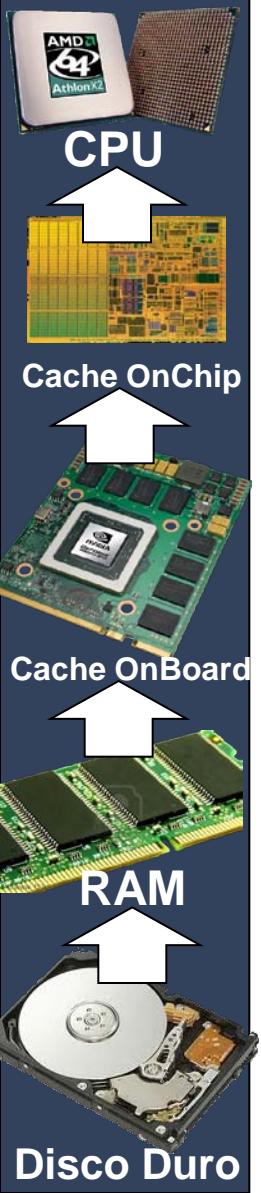
**En otro caso:** (más de 3 faltas ó que no hayan entregado todas las prácticas) deberán realizar un **EXAMEN Teórico-Práctico** en Feb/Sept.

La nota del parcial **SOLO** se conservará hasta septiembre

La convocatoria de septiembre también implicará la realización de una **práctica final Ó examen**.



### Antes del inicio de las prácticas:



**Los estudiantes deben descargarse (legal)**

- **SOFTWARE:**

Para la implementación y el diseño: “**Xilinx WebPACK**” de la página:

**<http://www.xilinx.com/tools/webpack.htm>**

- **Manuales y documentación de la web de XILINX:**

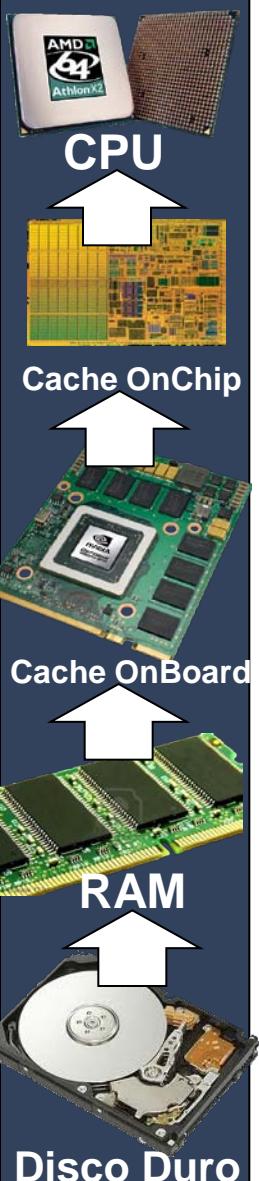
Documentación en particular de la **Spartan 3**

Manual de la herramienta de Xilinx (**ISE software**).

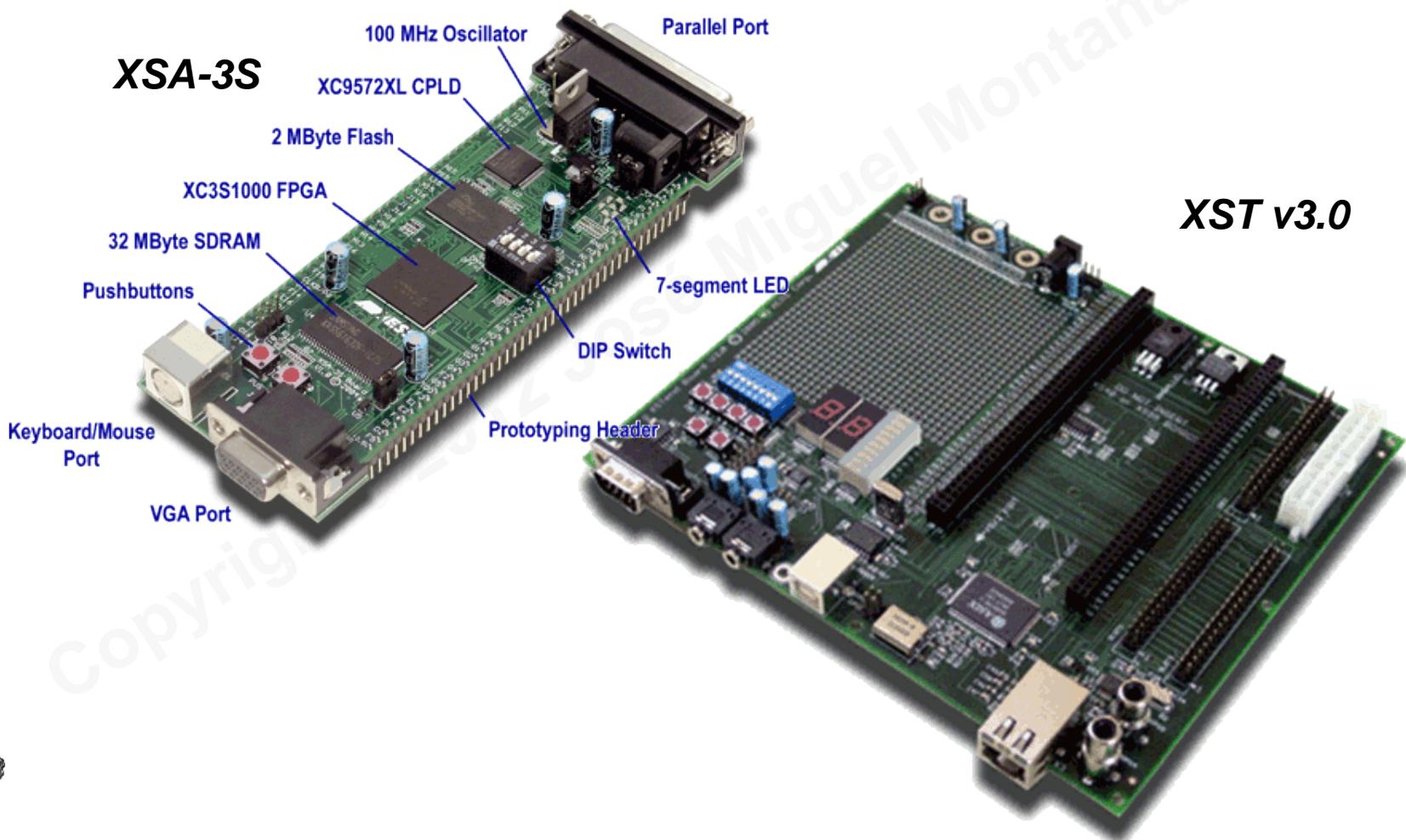
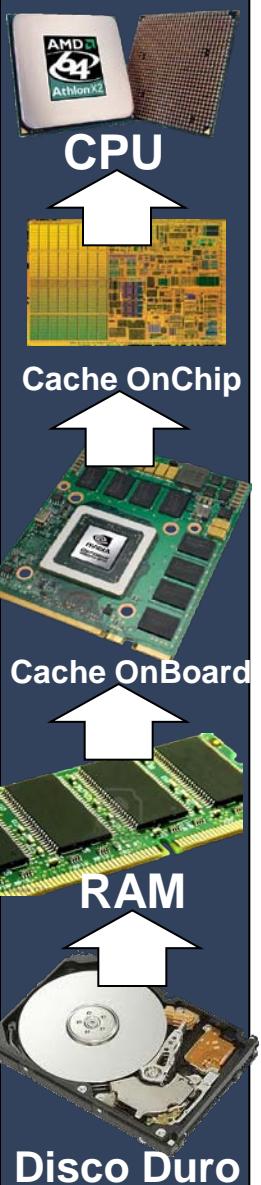
Nota: En la web de Xilinx, hay que registrarse para bajar algunos archivos

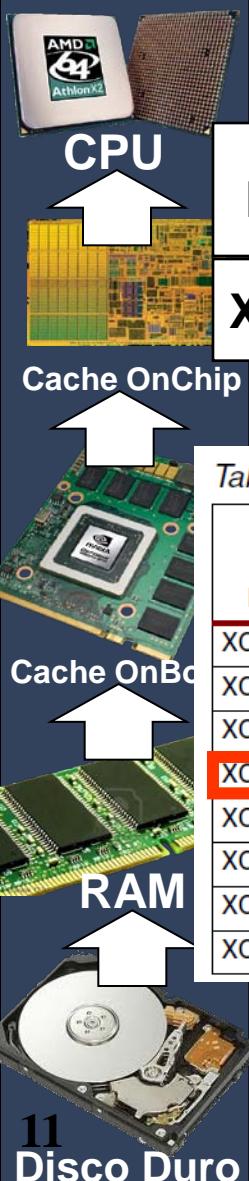
### BIBLIOGRAFÍA:

- G. De Micheli; **Synthesis and Optimization of Digital Circuits**; McGraw Hill, 1994
- VHDL: Lenguaje estándar de diseño electrónico; L. Terés, Y. Torroja, S. Olcoz, E. Villar; McGraw Hill, 1998
- Digital Systems Engineering; W.J. Dally, J.W. Poulton; Cambridge University Press
- Digital Integrated Circuits; Jan M. Rabaey; Prentice Hall
- **Tecnología de Computadores. Técnicas analógicas y digitales**  
M. Fernández, V. Sánchez, I. Martín. Síntesis 1996
- **The Practical Xilinx Lab Book, versión 1.5**  
David Van den Bout, Prentice Hall 1999
- **Electrónica Digital (Aplicaciones y Problemas con VHDL).**  
Artigas, Barragán, Orrite, Urriza, Ed. Prentice Hall 2002



## Material docente en los laboratorios de la Facultad de Informática

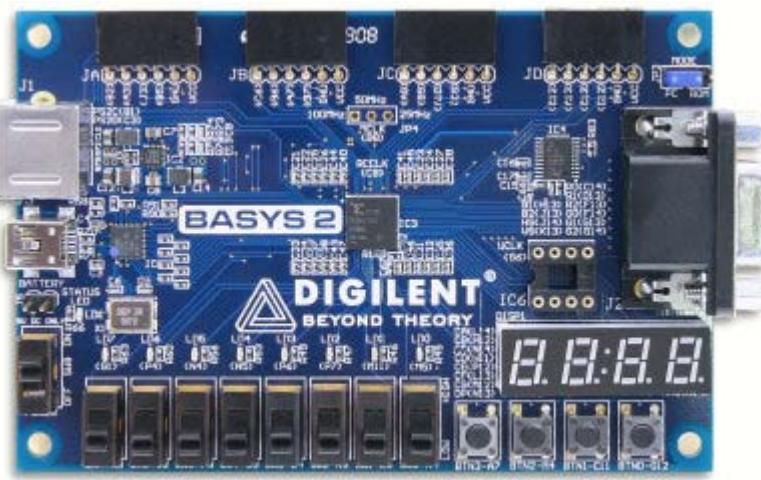




placa base	placa expansión	FPGA	memoria	sw	pb	leds	seggs	audio	video	VGA	PS2	USB	IDE	RS232	ether
XSA-3S v1.1	XST v3.0	XC3S1000	32 MB SDRAM	12	8	10	3	SI	SI	SI	SI	SI	SI	SI	SI

Table 1: Summary of Spartan-3 FPGA Attributes

Device	System Gates	Logic Cells	CLB Array (One CLB = Four Slices)			Distributed RAM (bits <sup>1</sup> )	Block RAM (bits <sup>1</sup> )	Dedicated Multipliers	DCMs	Maximum User I/O	Maximum Differential I/O Pairs
			Rows	Columns	Total CLBs						
XC3S50	50K	1,728	16	12	192	12K	72K	4	2	124	56
XC3S200	200K	4,320	24	20	480	30K	216K	12	4	173	76
XC3S400	400K	8,064	32	28	896	56K	288K	16	4	264	116
XC3S1000	1M	17,280	48	40	1,920	120K	432K	24	4	391	175
XC3S1500	1.5M	29,952	64	52	3,328	208K	576K	32	4	487	221
XC3S2000	2M	46,080	80	64	5,120	320K	720K	40	4	565	270
XC3S4000	4M	62,208	96	72	6,912	432K	1,728K	96	4	712	312
XC3S5000	5M	74,880	104	80	8,320	520K	1,872K	104	4	784	344



## Basys™2 Spartan-3E FPGA Board

100K gates: 79\$ para estudiantes 59\$

250K gates : 99\$ para estudiantes 79\$

<http://www.digilentinc.com>



## Xilinx FPGA Devices

Technology	Low-cost	High-performance
120/150 nm		Virtex 2, 2 Pro
90 nm	Spartan 3	Virtex 4
65 nm		Virtex 5
45 nm	Spartan 6	
40 nm		Virtex 6
28 nm (2013)	Artix 7	Kintex 7
		Virtex 7

## Altera FPGA Devices

Technology	Low-cost	Mid-range	High-performance
130 nm	Cyclone		Stratix
90 nm	Cyclone II		Stratix II
65 nm	Cyclone III	Arria I	Stratix III
40 nm	Cyclone IV	Arria II	Stratix IV



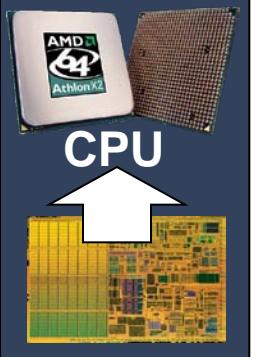
### METODOLOGÍA PARA LA REALIZACIÓN DE LAS PRÁCTICAS

La realización de una práctica puede dividirse en las siguientes fases:

1. OBTENCIÓN DE LAS ECUACIONES LÓGICAS del circuito a diseñar.
2. DISEÑO DEL CIRCUITO A NIVEL LÓGICO, utilizando puertas, biestables, multiplexores, sumadores, etc.
3. SIMULACIÓN del circuito, utilizando las herramientas [Xilinx ISE Project Navigator](#) y [ModelSim SE](#), (instalados en el laboratorio)

**Los 3 primeros puntos se realizarán ANTES de ir al laboratorio presentándolos al profesor antes de comenzar el montaje.**

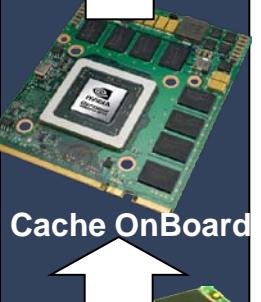
4. DISEÑO DEL CIRCUITO A NIVEL FÍSICO, utilizando las puertas disponibles.
5. MONTAJE.
  - Polarización individual de cada chip y de la placa de E/S.
  - Realización sobre el panel de montaje del entrenador de todas las conexiones que aparecen en el diagrama físico realizado.
6. COMPROBACIÓN de las salidas se corresponden con las respectivas entradas. (Utilizar todo el rango de entradas si es posible). Los problemas típicos son:
  - Ausencia de polarización.
  - Polarización incorrecta.
  - Salidas forzadas.
  - Entradas no conectadas.



CPU



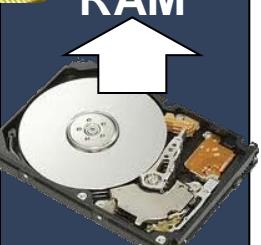
Cache OnChip



Cache OnBoard



RAM

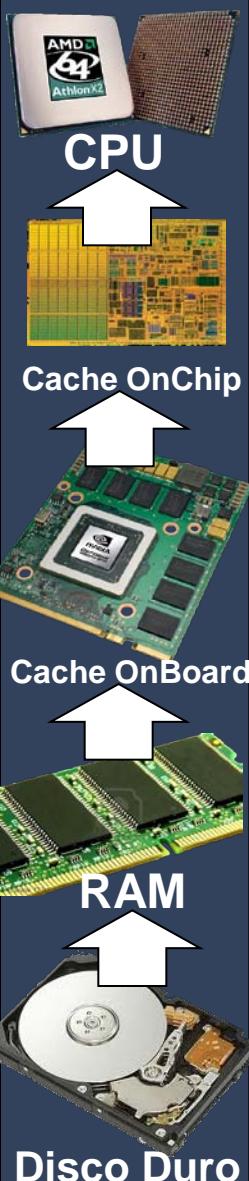


Disco Duro

### DEPURACIÓN DE CIRCUITOS COMBINACIONALES

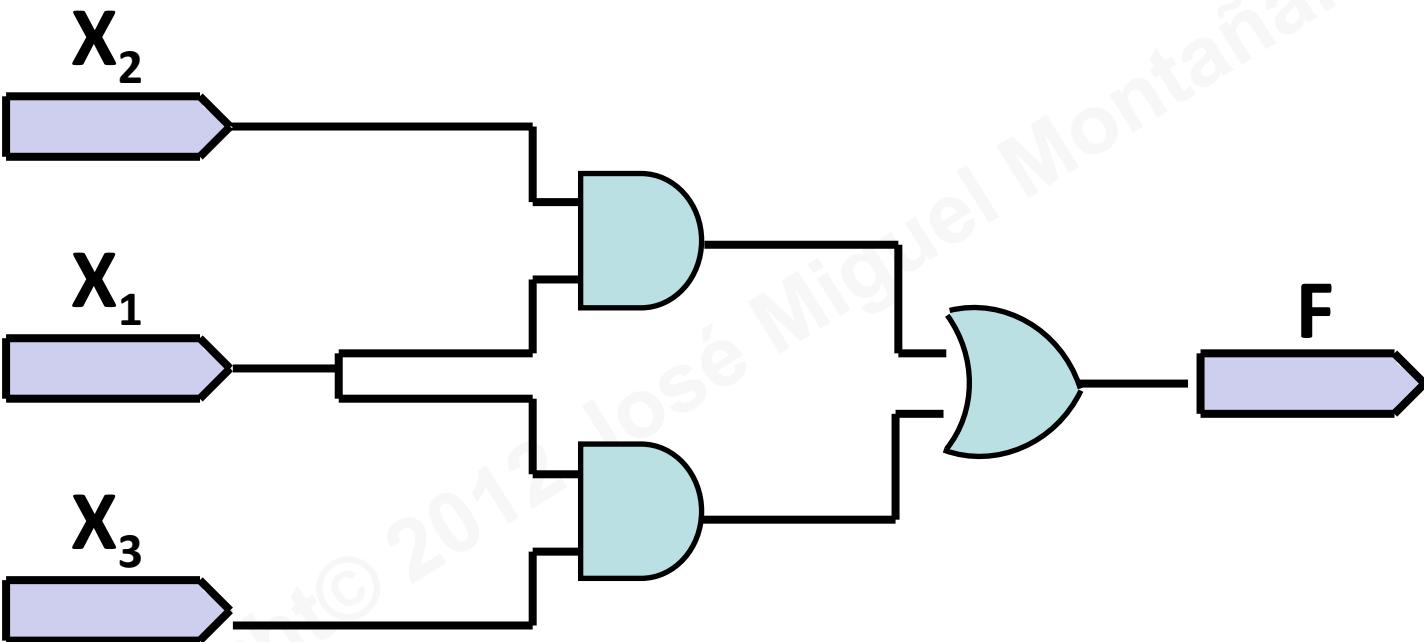
Búsqueda del error cuando una salida no se corresponde con la entrada:

1. **Repasar el diseño sobre el PAPEL.** Corregir los errores de diseño. Si esta bien ir a 2.
2. **Aislar la SALIDA incorrecta Y.** Para ello desconectar Y de todos los puntos a los que esté conectada.
  - a. Si Y toma su valor correcto ir a 3.
  - b. Si Y no toma su valor correcto ir a 4.
3. **La salida Y estaba conectada a un punto que forzaba su estado.** Esto ocurre cuando se conecta a tierra, a polarización o a otra salida.
  - a. Examinar los cables de salida de la puerta para comprobar cuál está mal conectado. Si no se detecta pasar a b.
  - b. Ir conectando, uno por uno, los cables de salida de la puerta, hasta que Y pase a ser incorrecta, (conexión errónea).
4. **La salida Y está generada por una puerta lógica P.**
  - a. Si Y se corresponde con las entradas de P, entonces, alguna/s de la/s entrada/s de P es/son incorrectas, ir a 5.
  - b. Si Y no se corresponde con las entradas de P, P no funciona correctamente, posibles causas: Chip mal polarizado o estropeado.
5. **Hallar la entrada de P que está tomando un valor incorrecto.** Repetir todo el procedimiento para la puerta lógica Q que genera esa entrada



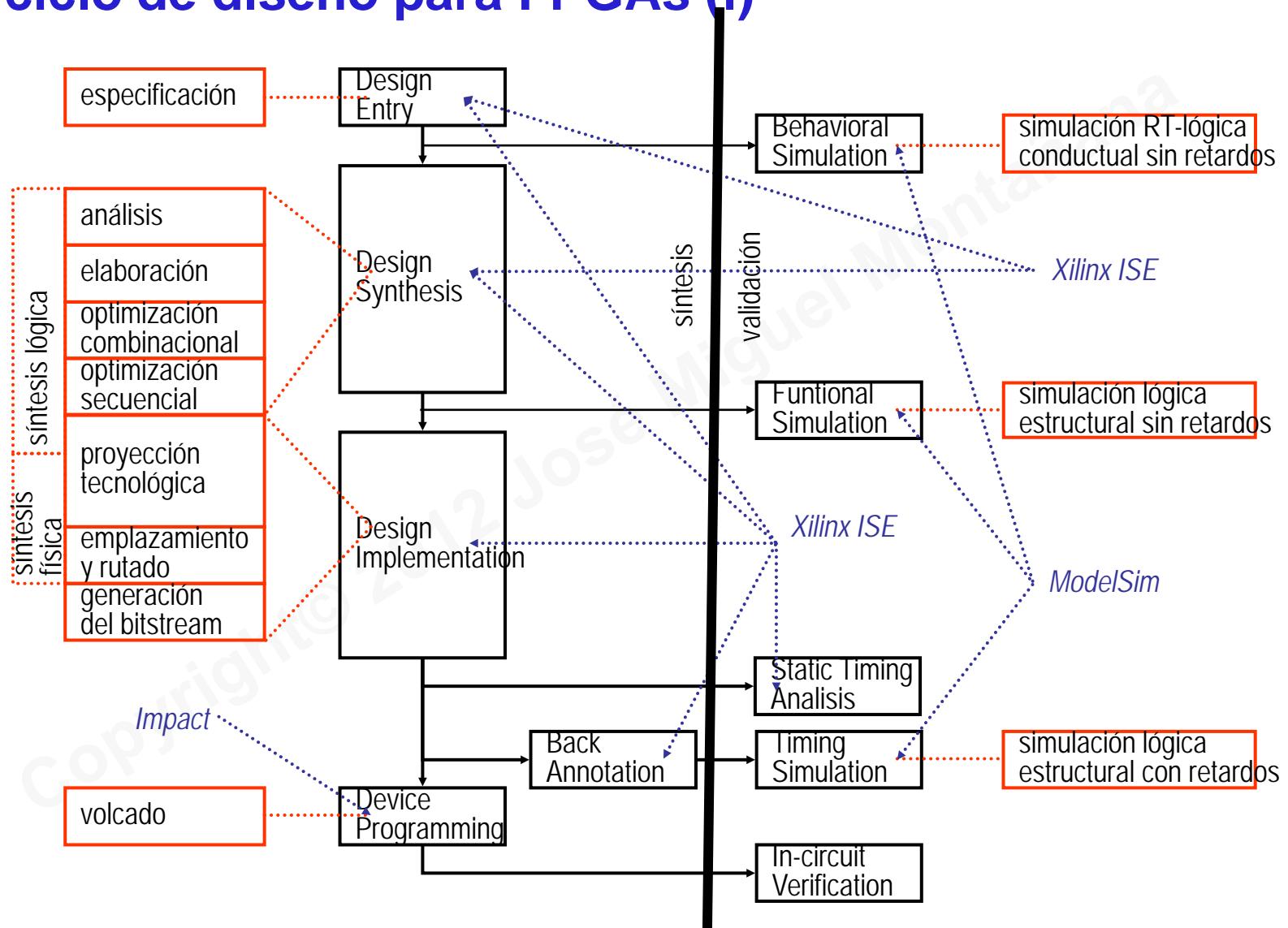
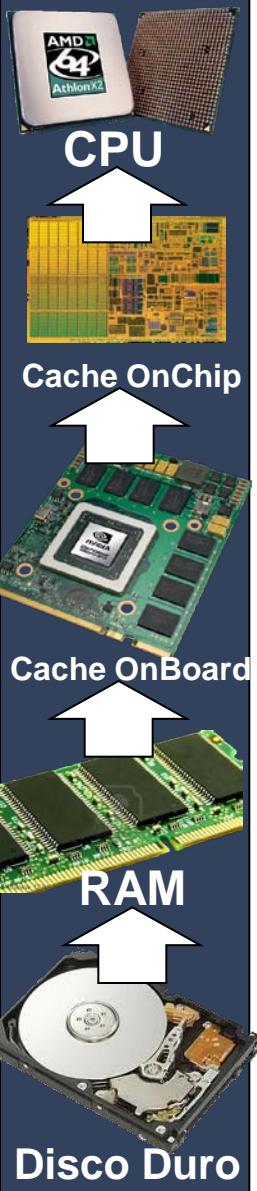


## Ejemplo:



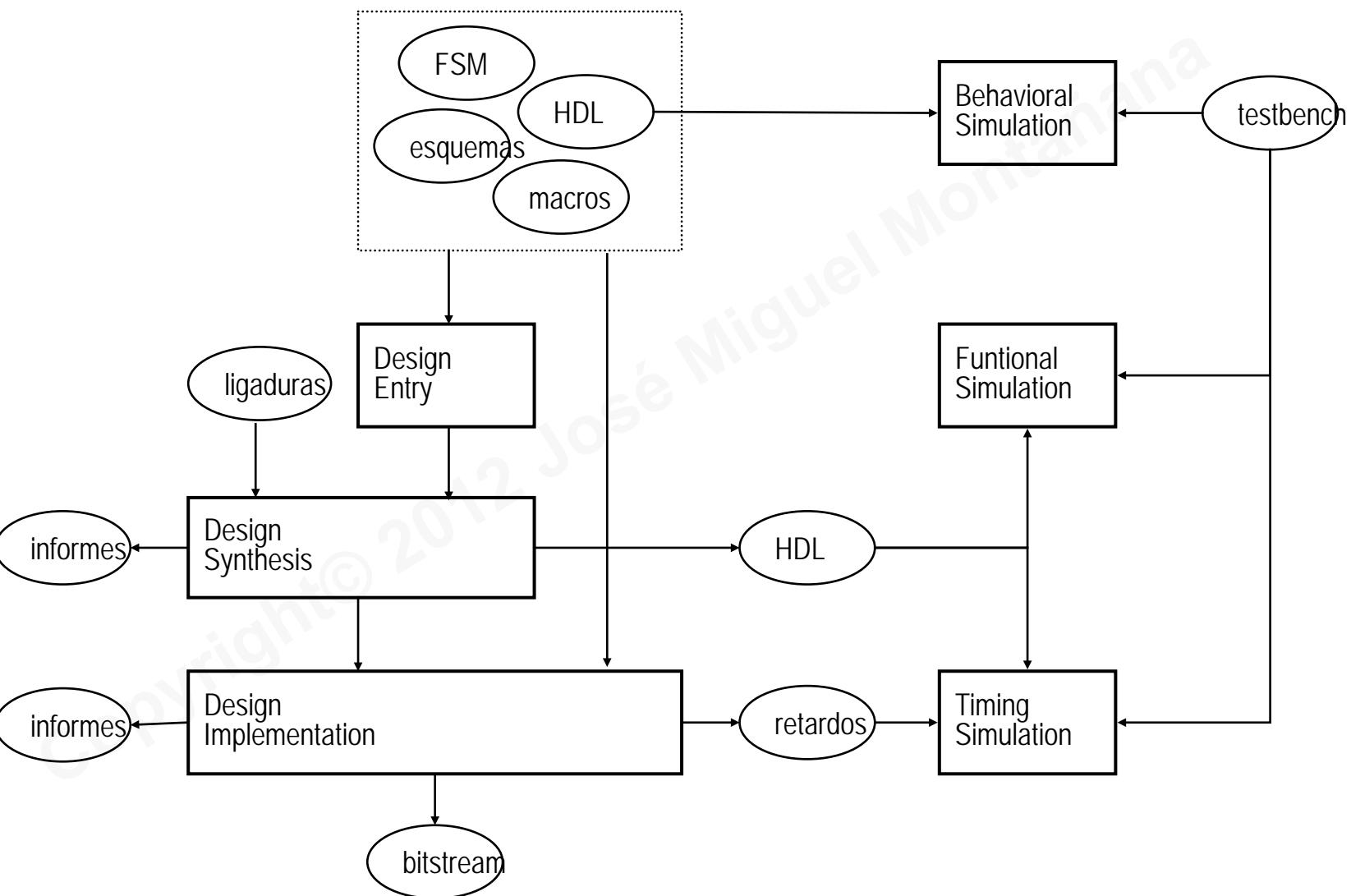
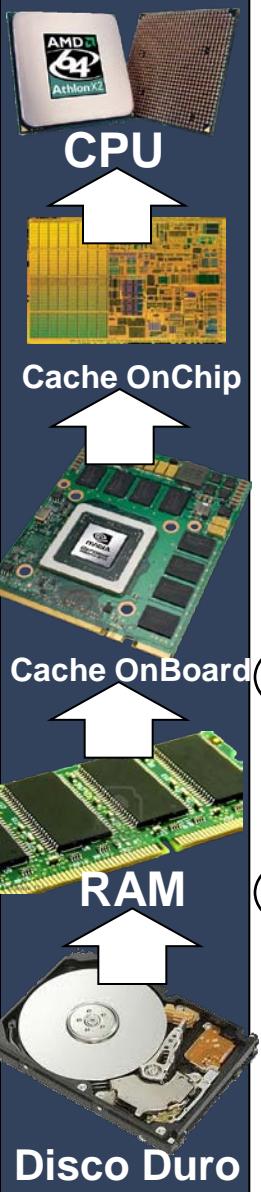
$$F = X_1 X_2 + X_1 X_3$$

### ciclo de diseño para FPGAs (i)





# ciclo de diseño para FPGAs (ii)



### Ciclo de diseño para FPGAs (iii)

Los informes (reports) son ficheros que contienen información sobre el diseño

- Avisos y errores encontrados durante el proceso de diseño
- Herramientas usadas y opciones utilizadas
- Decisiones de diseño tomadas (inferencias)
- Estadísticas de caracterización de la implementación

La visualización de informes puede ser:

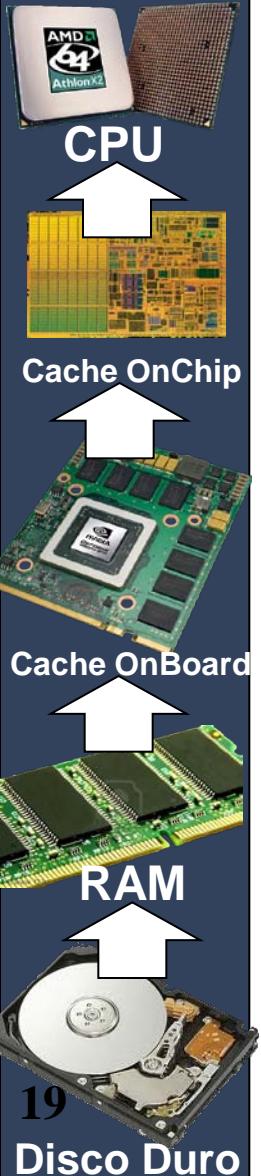
- Directa (fichero texto)
- A través de GUI

Las ligaduras (constraints) permiten controlar el ciclo de diseño y fijar los niveles mínimos de calidad aceptables de una implementación

- Técnicas de diseño a aplicar (locales a módulo):
  - Diseño jerárquico, codificación de FSM, estilo de RAM (block o distribuida) ...
- Estrategias de optimización (aplicables a todo el diseño)
  - Objetivo de la optimización (área o velocidad), esfuerzo (normal o alto) ...
- Ligaduras físicas
  - Tiempo de ciclo, retardos, localización (celdas y pines) ...

La definición de ligaduras puede ser

- A través de GUI: método simple
- A través de atributos de HDL: método asociable a HDL
- A través de un fichero de ligaduras: método más completo





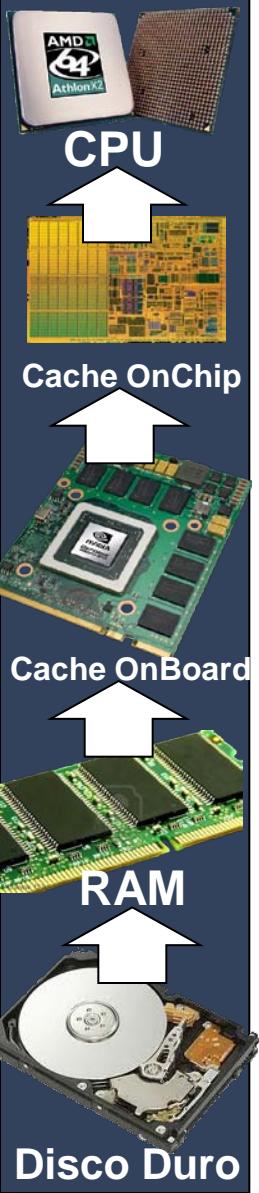
### *Herramientas de diseño:*

- **ISE v14.1 (Xilinx):** conjunto de herramientas para la síntesis lógica y física de sistemas digitales sobre FPGAs/CPLDs
  - Soporte a familias Xilinx: Spartan2, Spartan3, Virtex2, Virtex2-Pro
  - Xilinx ISE Webpack: versión gratuita descargable [www.xilinx.com/support/download](http://www.xilinx.com/support/download)

### *Herramientas de simulación:*

- **ModelSim v10.1c (Mentor Graphics):** simulador de VHDL/Verilog completo

### síntesis con Xilinx ISE en 5 pasos (i)



1. Crear proyecto HDL en el directorio de usuario : **File – New Project**
  - Family: **Spartan3**; Device: **XC3S1000**; Package: **FT256**; Speed: **4**; Language: **VHDL**
  - Usar el Wizard para crear la entidad top:
    - *Usar el mismo nombre para el proyecto, archivo VHDL, entidad top*
2. Editar código fuente VHDL (**fichero de especificación de conducta**)
  - Usar plantillas cuando haya dudas: **Edit – Language Templates**
  - Chequear sintáxis y salvar: **Processes – Synthesize – Check Syntax**
  - En cualquier momento se pueden crear/añadir nuevos ficheros:
    - **Processes – Create New Source / Add existing Source**
3. Crear un fichero UCF del mismo nombre (**fichero de ligaduras**)
  - Asignar individualmente a cada puerto VHDL un pin y salvar
    - **NET puertoVHDL LOC=id**
  - Añadir el fichero al proyecto: **Processes – Add Existing Source**
4. Implementar: **Processes – Generate Programming File**
5. Volcar fichero BIT (**fichero de bitstream**) resultante usando **Impact**
  - **Processes – Configure Target Device**

# DISEÑO AUTOMÁTICO DE SISTEMAS

## Curso 2013-2014

### síntesis con Xilinx ISE en 5 pasos (ii)



Xilinx - ISE - C:\Users\Mendi\cronometro\cronometro.ise - [Design Summary]

File Edit View Project Source Process Window Help

Sources for: Implementation

- crono - estructural
- eliminador - debu
- start - pulsador -
- timer - contador -
- decimas - contac

Processes for crono - estructural

- Add Existing Source
- Create New Source
- View Design Summary
- Design Utilities
- User Constraints
- Synthesize - XST
- Implement Design
- Generate Programming
- Configure Target Dev

Project Properties

- Enable Enhanced Design Summ
- Enable Message Filtering
- Display Incremental Messages
- Show Partition Data
- Show Errors

FPGA Design Summary

Design Overview

- Summary
- IOB Properties
- Module Level Utilization
- Timing Constraints
- Pinout Report
- Clock Report
- Errors and Warnings

  - Synthesis Messages
  - Translation Messages
  - Map Messages

Project File: cronometro.ise

Module Name: crono

Target Device: xc3s1000-4ft256

Product Version: ISE 10.1 - WebPACK

Design Goal: Balanced

Design Strategy: Xilinx Default (unlocked)

Current State: Programming File Generated

- Errors: No Errors
- Warnings: 10 Warnings
- Routing Results: All Signals Completely Routed
- Timing Constraints: All Constraints Met
- Final Timing Score: 0 (Timing Report)

cronometro Project Status (10/11/2011 - 12:06:11)

No partition information was found.

Device Utilization Summary

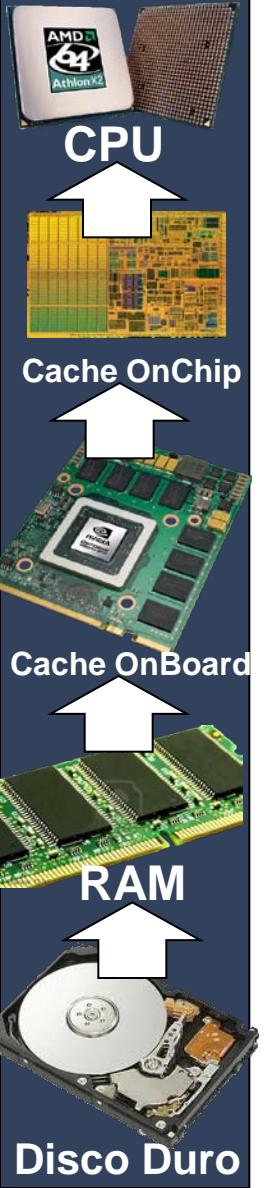
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	58	15,360	1%	
Number of 4 input LUTs	112	15,360	1%	

Logic Distribution				
Number of occupied Slices	72	7,680	1%	
Number of Slices containing only related logic	72	72	100%	

Design Summary p4.vhd

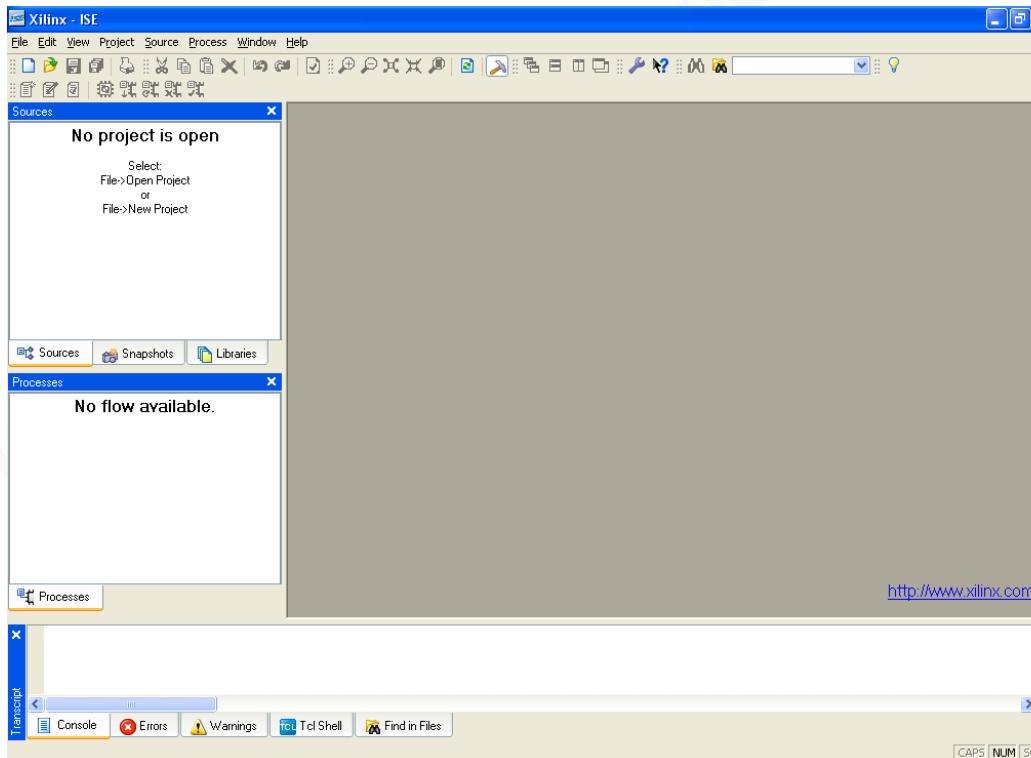
Console Errors Warnings Tcl Shell Find in Files

### PRIMEROS PASOS CON EL XILINX ISE



Xilinx ISE 10.1

**Inicio**  
**Programas →**  
**→Electronica->**  
**→Xilinx ISE 10.1 →**  
**→Project Navigator**



# DISEÑO AUTOMÁTICO DE SISTEMAS

## Curso 2013-2014

### 0. Introducción



**New Project Wizard - Create New Project**

Enter a Name and Location for the Project

Project Name:  Project Location:

Select the Type of Top-Level Source for the Project

Top-Level Source Type:

**New Project Wizard - Device Properties**

Select the Device and Design Flow for the Project

Property Name	Value
Product Category	All
Family	Spartan3
Device	XC3S1000
Package	FT256
Speed	-5
Top-Level Source Type	<input type="button" value="Schematic"/>
Synthesis Tool	<input type="button" value="XST (VHDL/Verilog)"/>
Simulator	<input type="button" value="Modelsim-XE VHDL"/>
Preferred Language	<input type="button" value="VHDL"/>
Enable Enhanced Design Summary	<input checked="" type="checkbox"/>
Enable Message Filtering	<input type="checkbox"/>
Display Incremental Messages	<input type="checkbox"/>



New Source Wizard - Select Source Type

- IP (Coregen & Architecture Wizard)
- Schematic
- State Diagram
- Test Bench WaveForm
- User Document
- Verilog Module
- Verilog Test Fixture
- VHDL Module
- VHDL Library
- VHDL Package
- VHDL Test Bench

File name: prueba1

Location: C:\hlocal\practica1

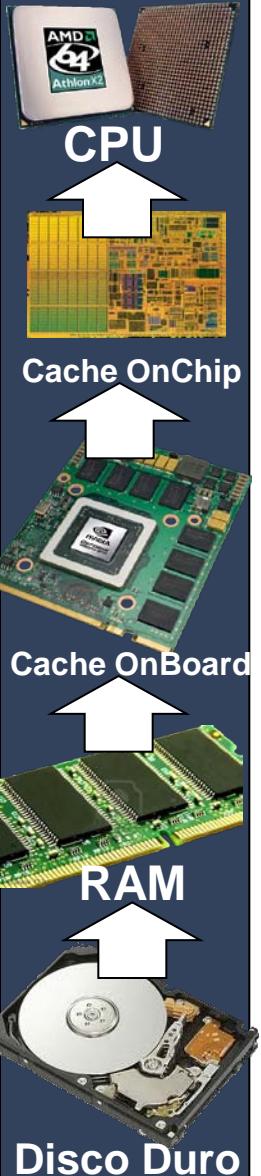
Add to project

More Info Back Next Cancel

# DISEÑO AUTOMÁTICO DE SISTEMAS

## Curso 2013-2014

### 0. Introducción



Xilinx - ISE - C:\hlocal\practica1\practica1.ise - [prueba1.ucf]

File Edit View Project Source Process Window Help

Sources Sources for: Synthesis/Implementation

practica1 xc3s1000-5ft256 prueba1 (prueba1.sch) XLXI\_30 - divisor - divisor\_arch (divisor) prueba1.ucf (prueba1.ucf)

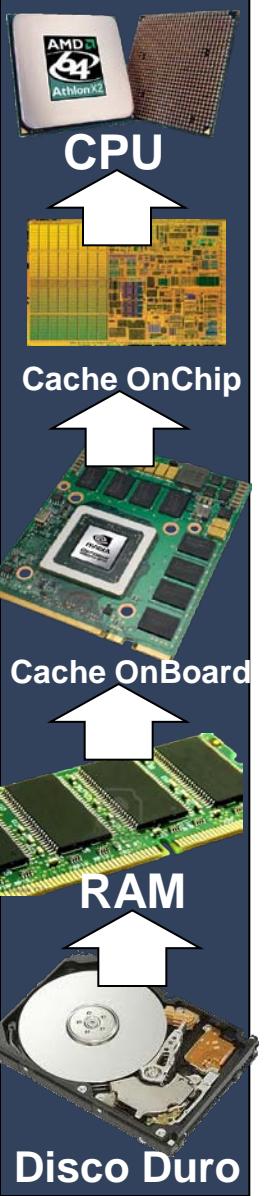
```
1 #salida del oscilador a 100MHz
2 #
3 NET ent4 LOC=T9;
4 #
5 #
6 #switches placa extendida
7
8 NET ent1 LOC=P12;
9 NET ent2 LOC=J1;
10 NET ent3 LOC=H1;
11 #NET DIPSW<4> LOC=H3;
12 #NET DIPSW<5> LOC=G2;
13 #NET DIPSW<6> LOC=K15;
14 #NET DIPSW<7> LOC=K16;
15 #NET DIPSW<8> LOC=F15;
16
17 #barra de leds placa extendida
18
19 NET salida LOC=L5;
20 #NET leds<2> LOC=N2;
21 #NET leds<3> LOC=M3;
22 #NET leds<4> LOC=N1;
23 #NET leds<5> LOC=T13;
24 #NET leds<6> LOC=L15;
25 #NET leds<7> LOC=J13;
26 #NET leds<8> LOC=H15;
27 #NET leds<9> LOC=J16;
28 #NET leds<10> LOC=J14;
29
30
```

Processes Processes for: prueba1

- Add Existing Source
- Create New Source
- View Design Summary
- Design Utilities
- User Constraints
  - Create Timing Constraints
  - Assign Package Pins
  - Create Area Constraints
  - Edit Constraints (Text)

prueba1.sch prueba1.ucf

Ln 12 Col 22 CAPS NUM SCRL UCF



# DESCARGA DEL DISEÑO EN LA PLACA XSA-3S1000

Desde el menú de Windows seleccionar:

Programas → Electrónica → XStools → GXSLoad

Seleccionar el tipo de tarjeta: XSA-3S1000

Arrastrar el \*.bit a la ventana FPGA y pulsar Load.

Una vez volcado el mapa de bits, colocar las entradas ent1, ent2 a 1 y ent3 a 0 (switches 1, 2 y 3 de la placa extendida) y comprobar que un parpadeo aparece representado el led 1.

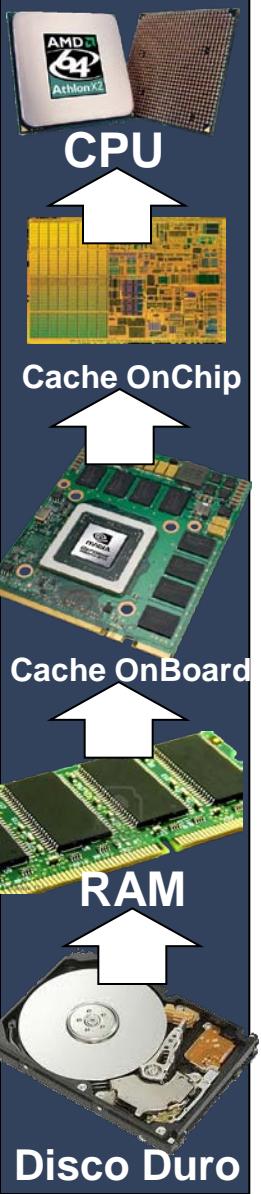
En las placas XSA-3S1000 debe realizarse un paso adicional para que el puerto paralelo no fuerce la salida:

Programas → Electrónica → XStools → GXSPort

Poner todos los datos a 1 y pulsar SET

### MERCADO (MUNDO REAL)

Modern Designing:  
Natural Progression of Tools



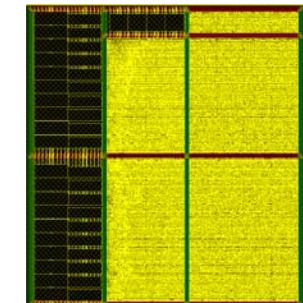
**Software**

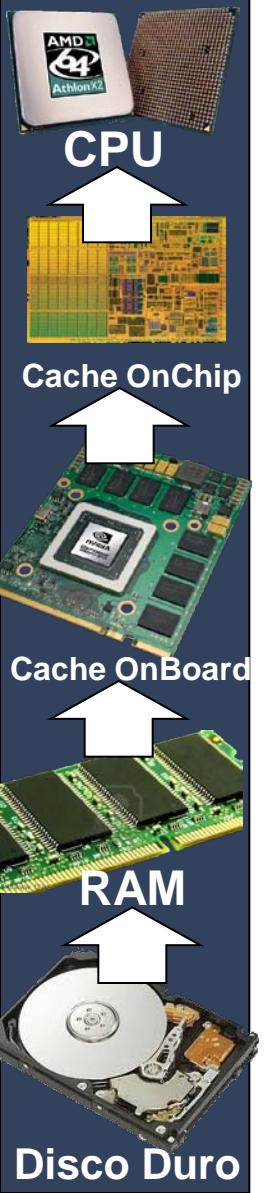


**FPGAs**



**ASICs**





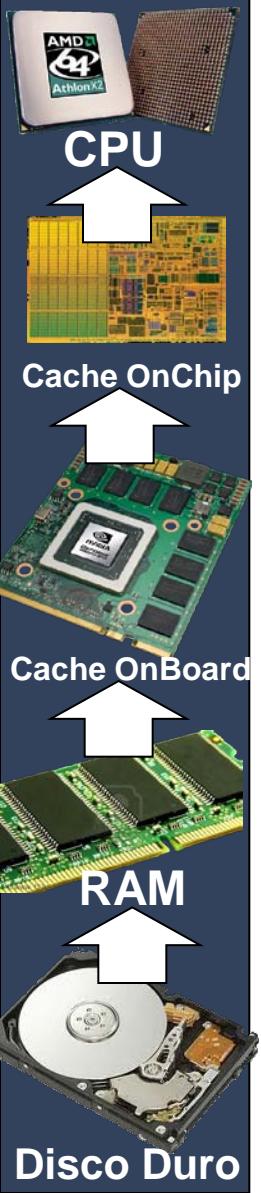
### Video de un sistema con la Spartan 3

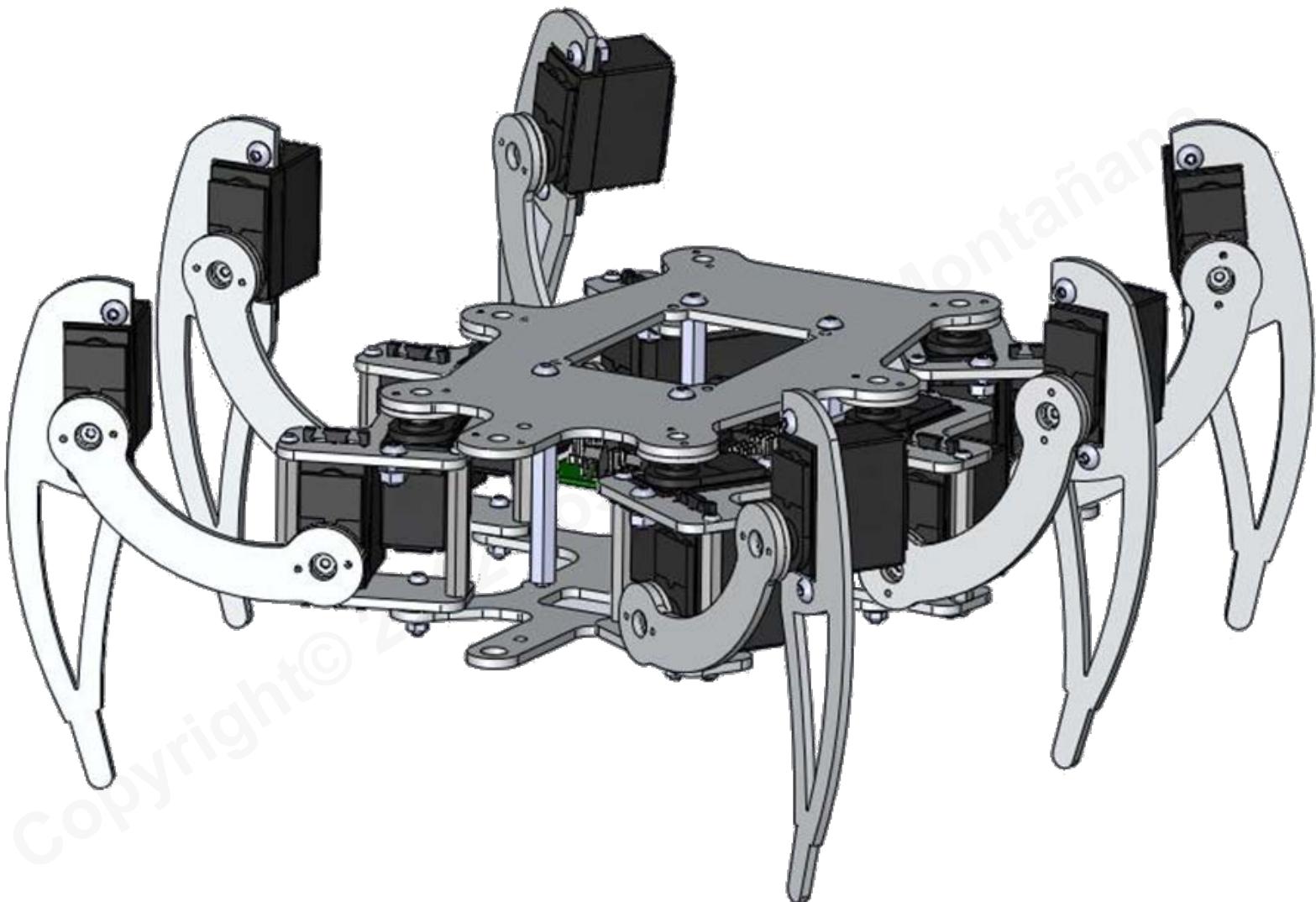
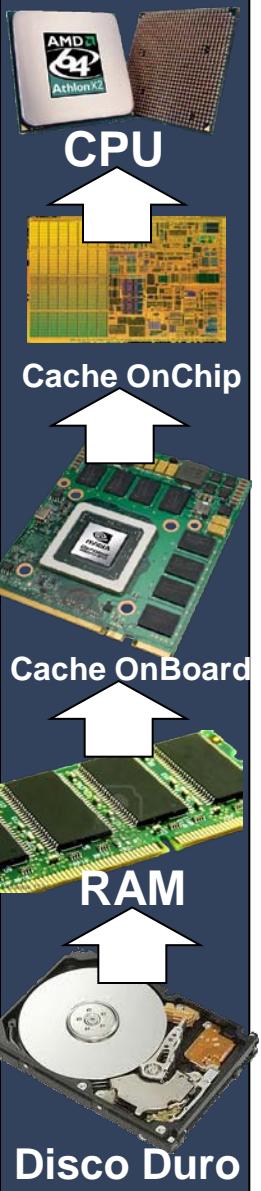
Es un ejemplo, pero **NO** se realiza este curso

# DISEÑO AUTOMÁTICO DE SISTEMAS

## Curso 2013-2014

### 0. Introducción

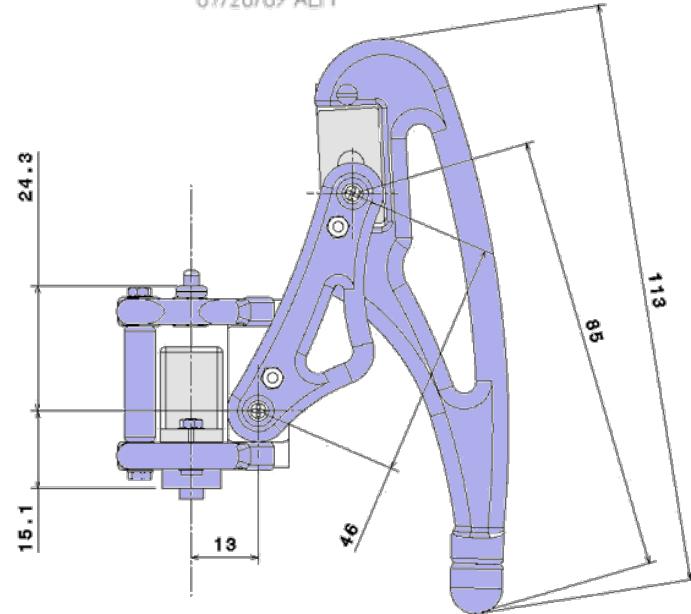
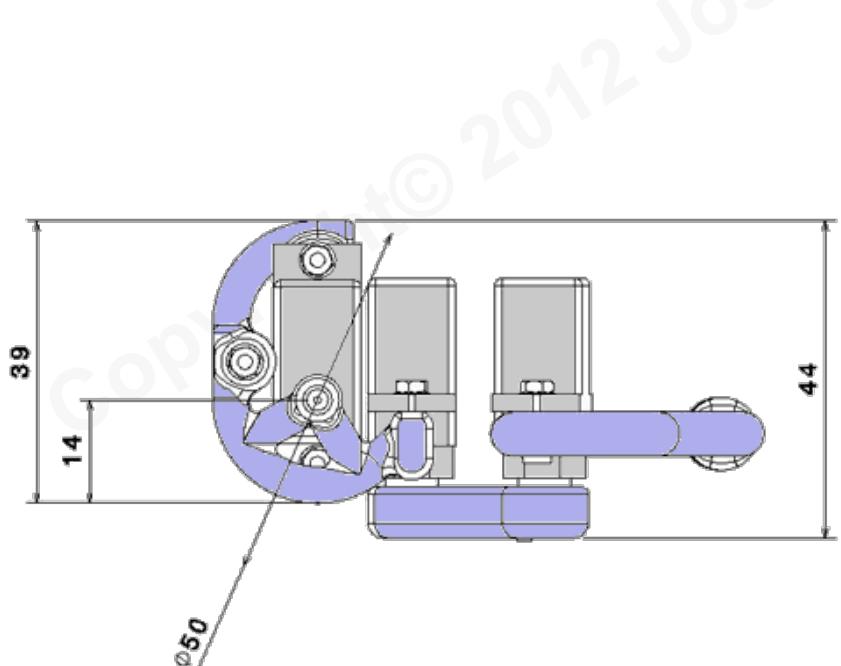
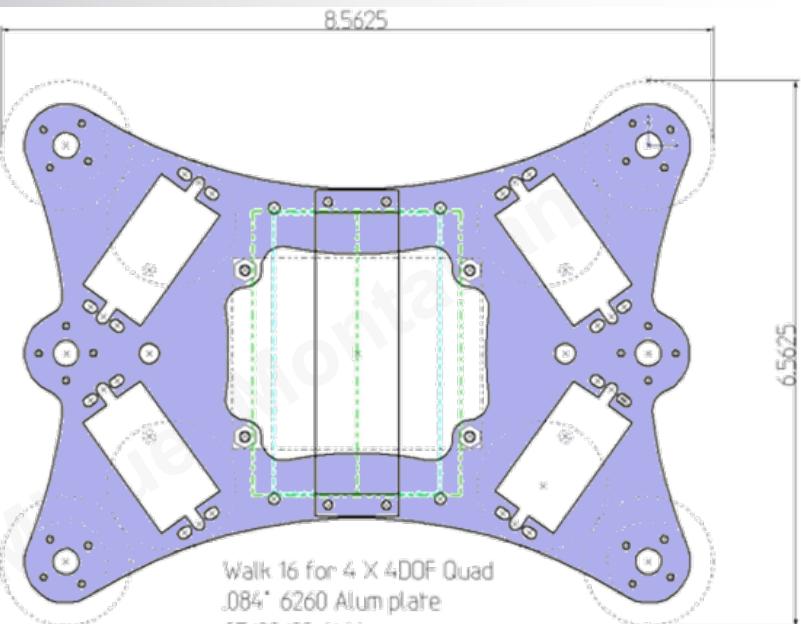
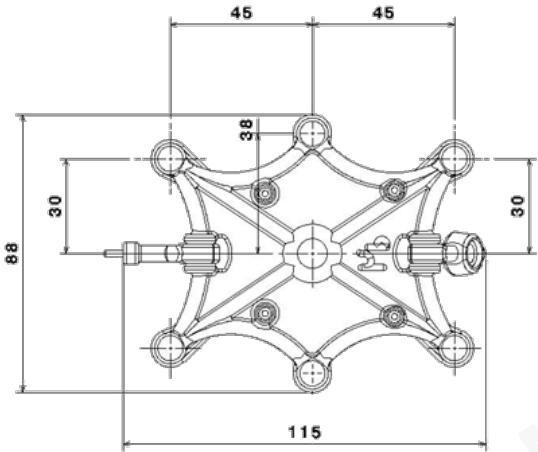
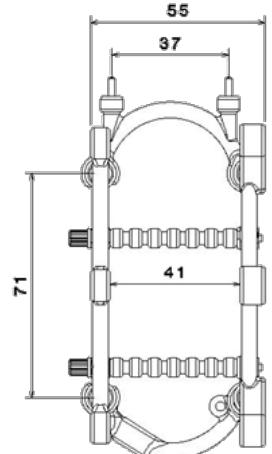
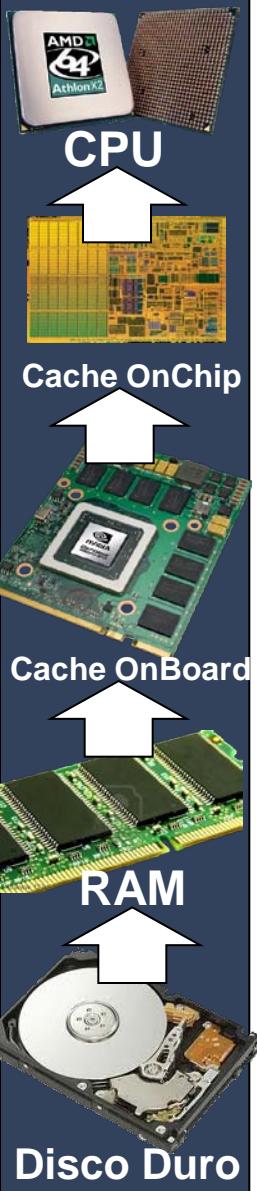


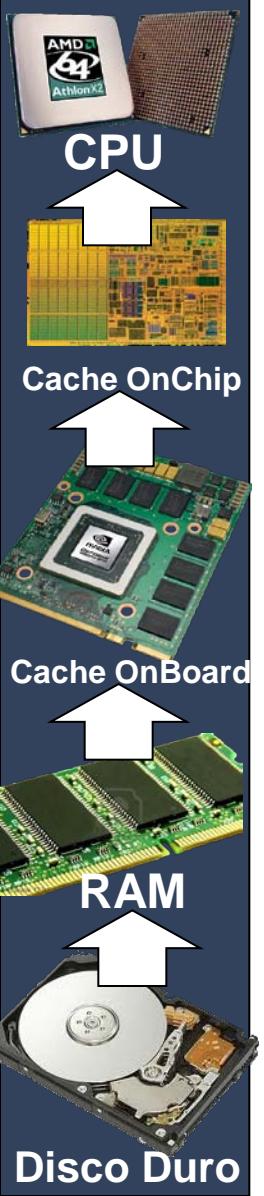


# DISEÑO AUTOMÁTICO DE SISTEMAS

## Curso 2013-2014

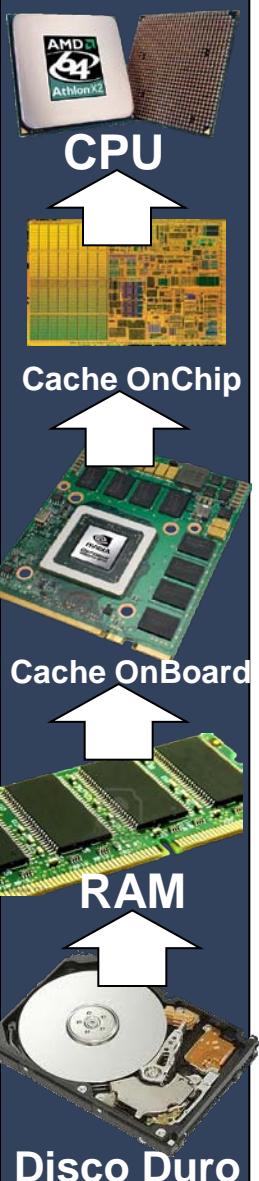
### 0. Introducción





# Introducción a VHDL

1. “If”
  - Multiplexor
  - Registros
  - Eliminador de rebotes
  - Contadores
  - Divisor de frecuencia
2. Maquinas de estados
3. “Case”
  - Decodificador de 7 segmentos
  - ALU
4. “For”
  - Multiplicador
  - Divisor
5. Sistema Algorítmico



```
IF expresión THEN  
....  
ELSIF expresión THEN  
....  
ELSE  
....  
END IF;
```

```
PROCESS (lista de sensibilidad)  
declaraciones  
BEGIN  
....  
END PROCESS;
```

## DISEÑO DE SISTEMAS SINCRONOS.

```
IF expresión THEN  
....  
ELSIF expresión THEN  
....  
ELSE  
....  
END IF;
```

```
PROCESS (clk, lista de sensibilidad)  
declaraciones  
BEGIN  
IF clk'event AND clk='1' THEN  
....  
END IF;  
END PROCESS;
```

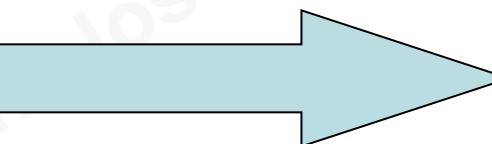
### Ejemplo: Multiplexor

#### 2- Que es un multiplexor?

#### *MULTIPLEXORES: Concepto General*

I1	I0	A	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$Z = \text{NOT}(A) I_0 + A I_1$$



A	Z
0	I0
1	I1

Descripción lógica

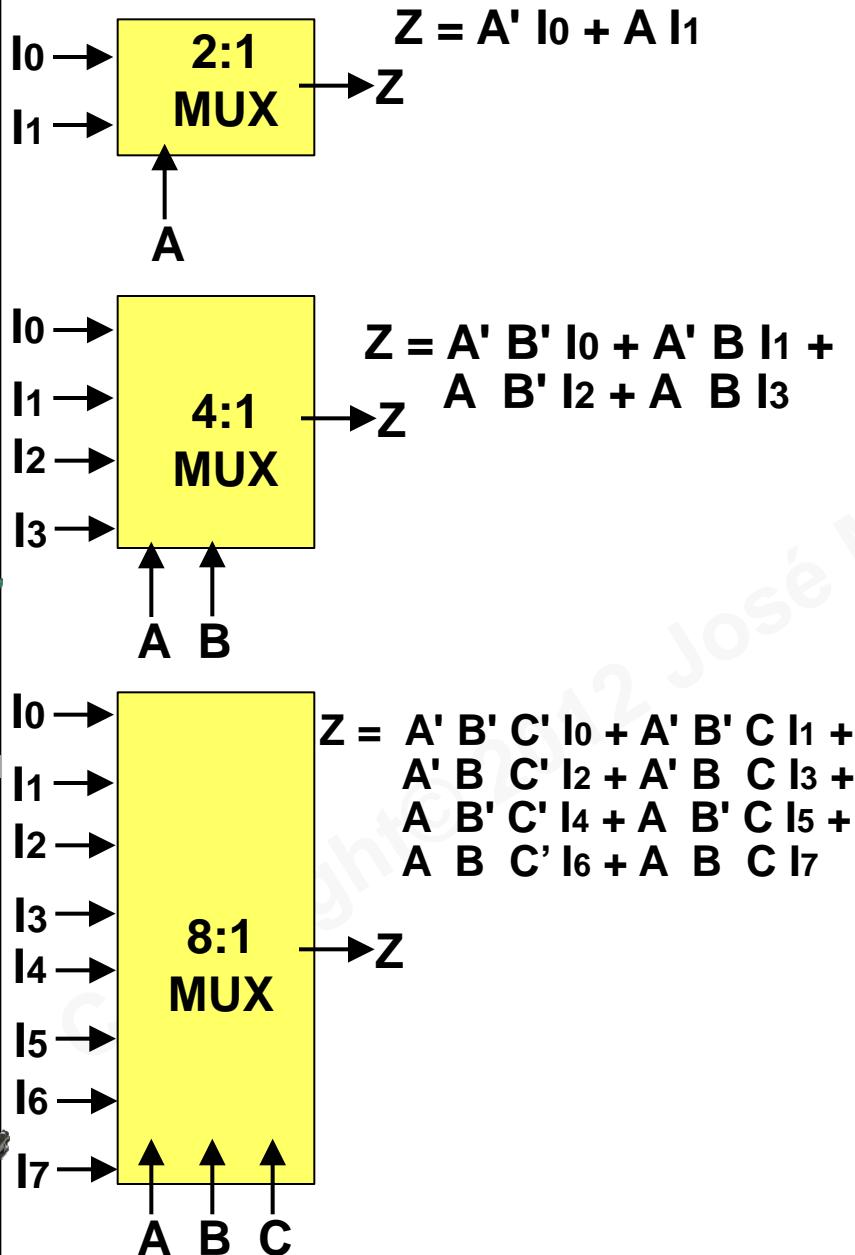
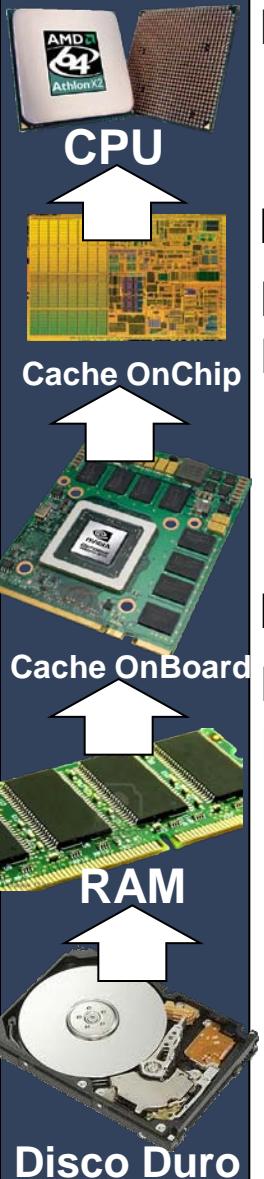
Descripción funcional



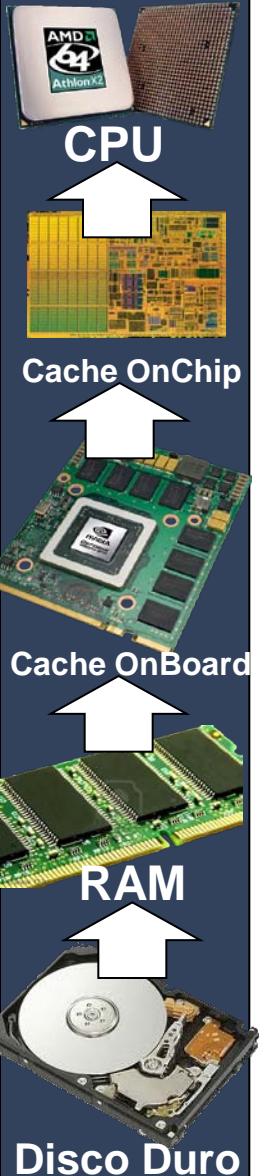
# DISEÑO AUTOMÁTICO DE SISTEMAS

## Curso 2013-2014

Introducción a VHDL  
**MULTIPLEXOR**



--MODULO MULTIPLEXOR 2:1  
 Library IEEE;  
 Use IEEE.SD\_LOGIC\_1164.all;  
**ENTITY mux IS**  
 port (I0: IN STD\_LOGIC;  
 I1: IN STD\_LOGIC;  
 A: IN STD\_LOGIC;  
 z: OUT STD\_LOGIC);  
**ARCHITECTURE mux OF mux IS**  
**BEGIN**  
 PROCESS(I0, I1, A)  
**BEGIN**  
 IF A='0' THEN  
 Z <=I0;  
 ELSE  
 Z <=I1;  
 END IF;  
 END PROCESS;  
**END mux;**



### Como usamos el modulo mux?

#### --MODULO PRINCIPAL

```
Library IEEE;
Use IEEE.SD_LOGIC_1164.all;
ENTITY algo IS
```

```
port (I0: IN STD_LOGIC;
      I1: IN STD_LOGIC;
      I2: IN STD_LOGIC;
      A: IN STD_LOGIC;
      z: out STD_LOGIC );
```

```
ARCHITECTURE algo OF algo IS
```

```
COMPONENT mux IS
```

```
port (I0: IN STD_LOGIC;
      I1: IN STD_LOGIC;
      A: IN STD_LOGIC;
      z: OUT STD_LOGIC);
```

```
END COMPONENT;
```

```
BEGIN
```

```
mi_mux1: mux port map(
      I0 => I0, I1 => I1, A => A,
      z =>z);
```

```
END algo;
```

#### --MODULO MULTIPLEXOR

```
Library IEEE;
```

```
Use IEEE.SD_LOGIC_1164.all;
```

```
ENTITY mux IS
```

```
port (I0: IN STD_LOGIC;
      I1: IN STD_LOGIC;
      A: IN STD_LOGIC;
      z: OUT STD_LOGIC);
```

```
ARCHITECTURE mux OF mux IS
BEGIN
```

```
PROCESS( A)
```

```
BEGIN
```

```
IF A='0' THEN
```

```
  Z <=I0;
```

```
ELSE
```

```
  Z <=I1;
```

```
END IF;
```

```
END PROCESS;
```

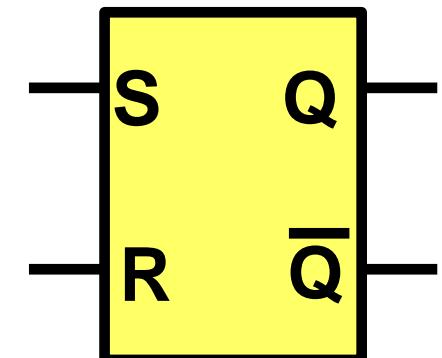
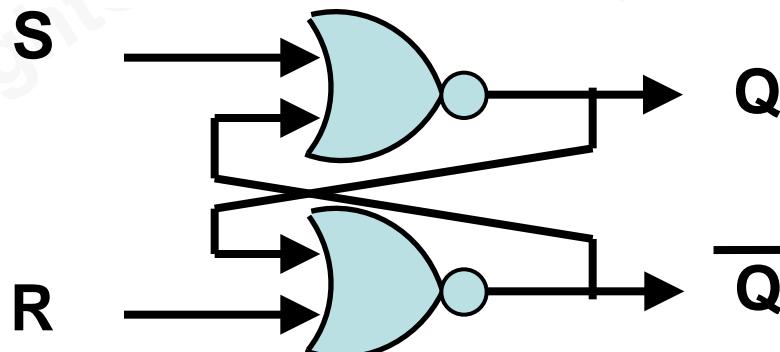
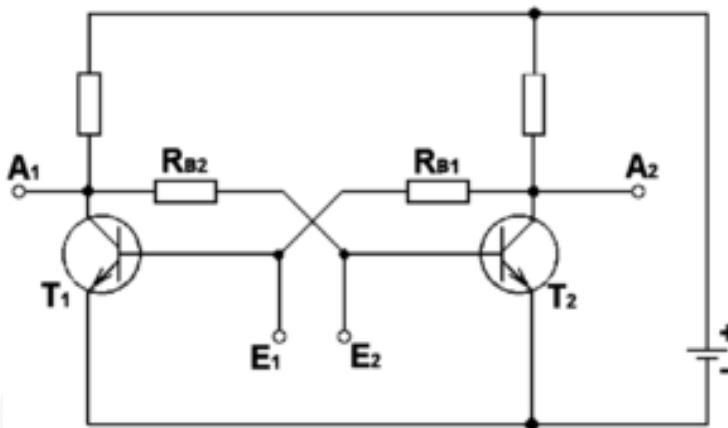
```
END mux;
```

#### Biestables (Flip-Flops):

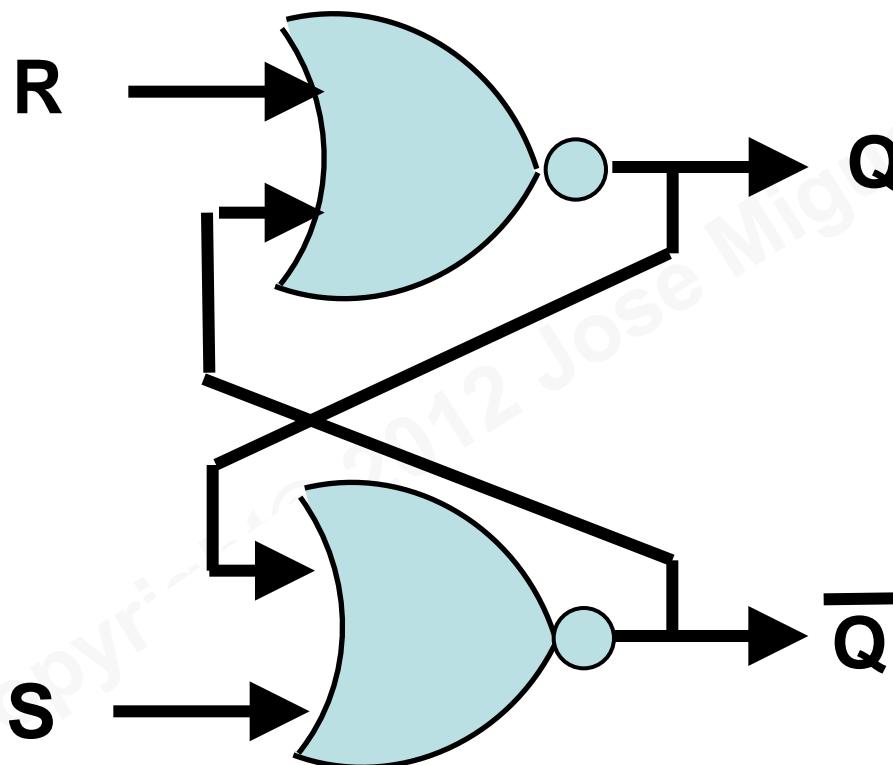
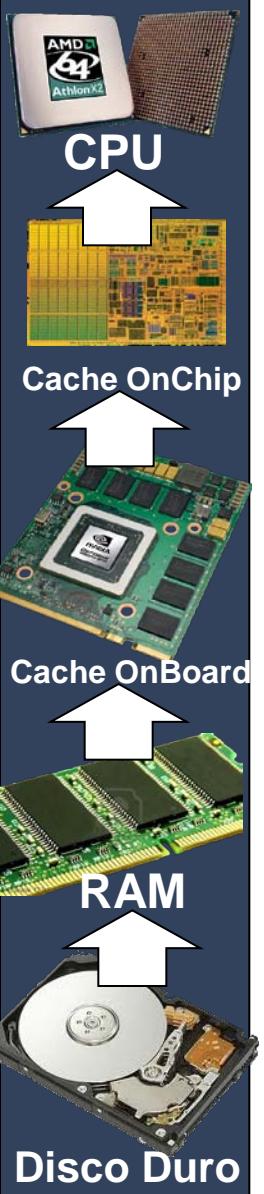
Tiene dos estados estables.

Tiene una o dos salidas ( $Q$ ,  $\bar{Q}$ )

El estado de la salidas, depende del estado actual de la las entradas,  
( y del estado anterior)

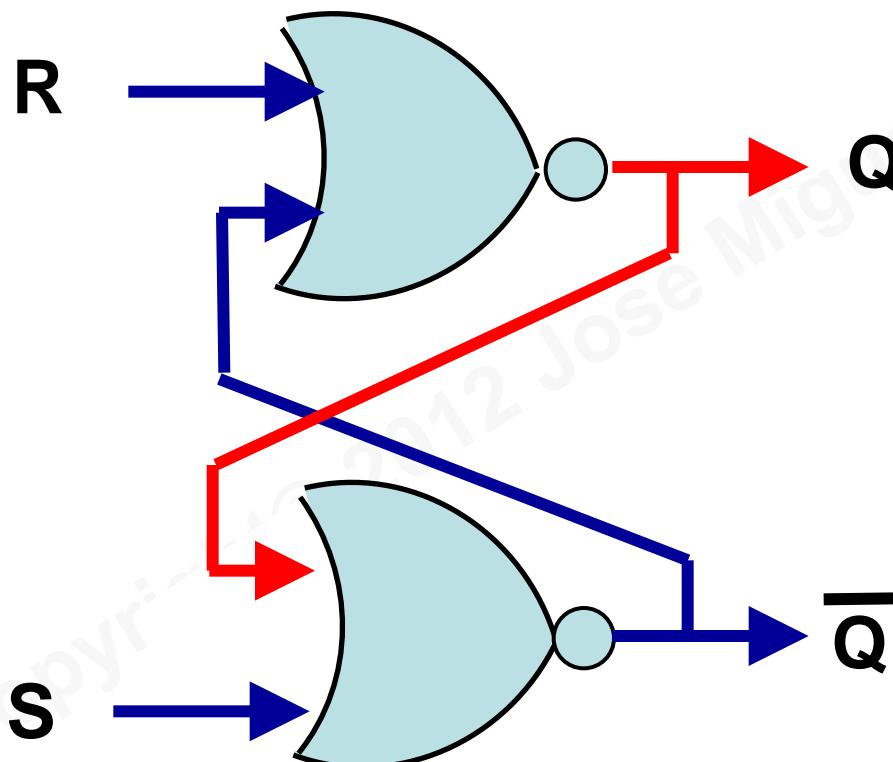
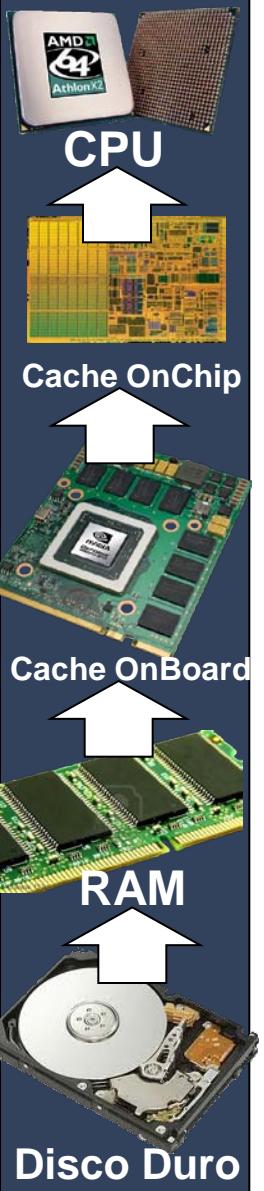


#### REALIZANDO SET

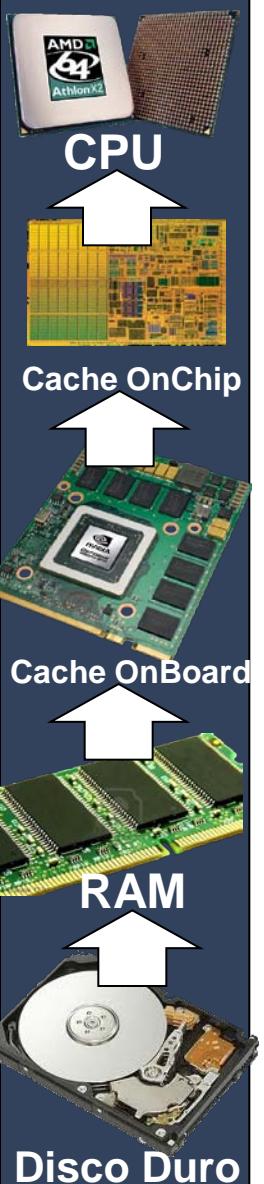
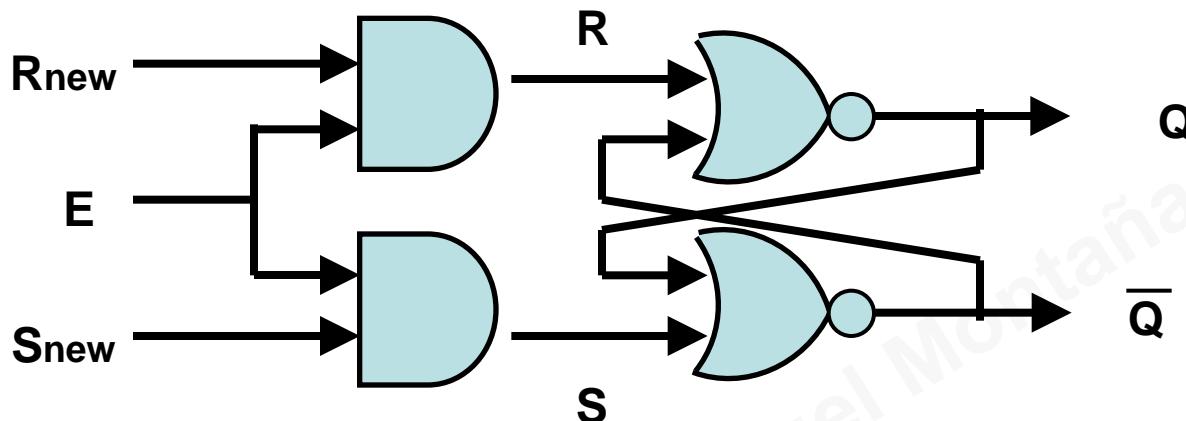


S	R	ESTADO
0	0	SIN CAMBIO
0	1	$Q=0$
1	0	$Q=1$
1	1	PROHIBIDO

#### REALIZANDO RESET



S	R	ESTADO
0	0	SIN CAMBIO
0	1	$Q=0$
1	0	$Q=1$
1	1	PROHIBIDO



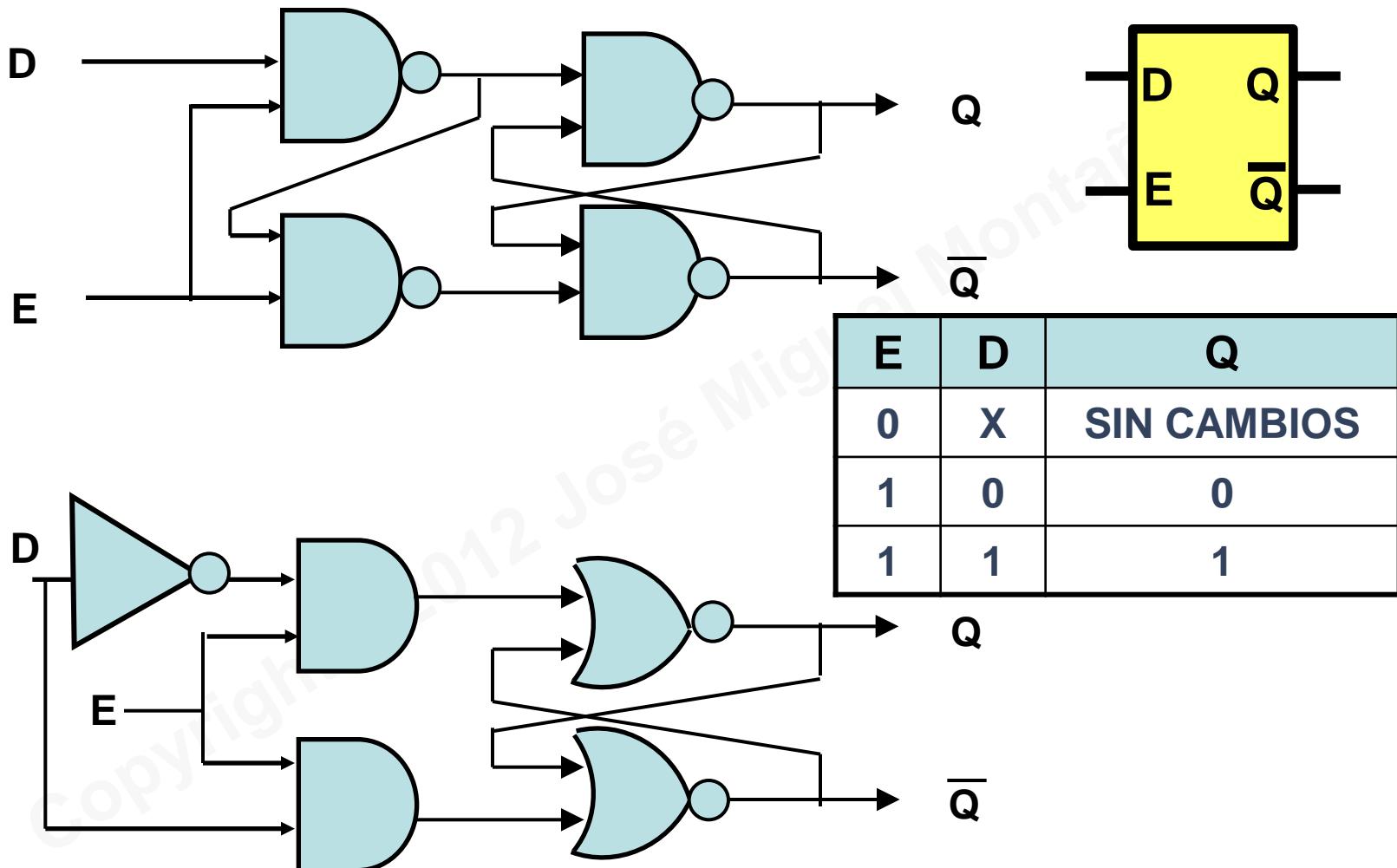


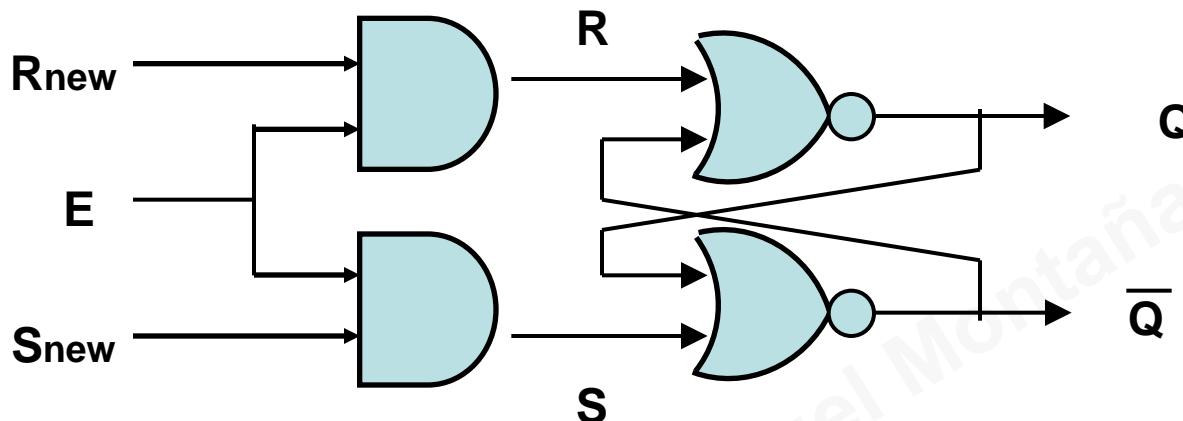
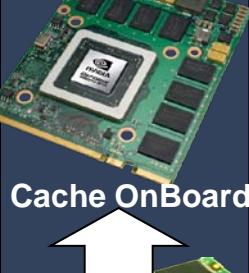
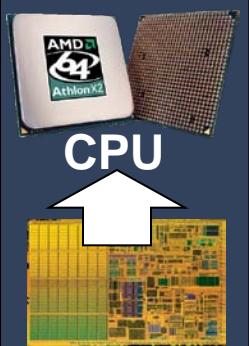
# DISEÑO AUTOMÁTICO DE SISTEMAS

## Curso 2013-2014

### Introducción a VHDL

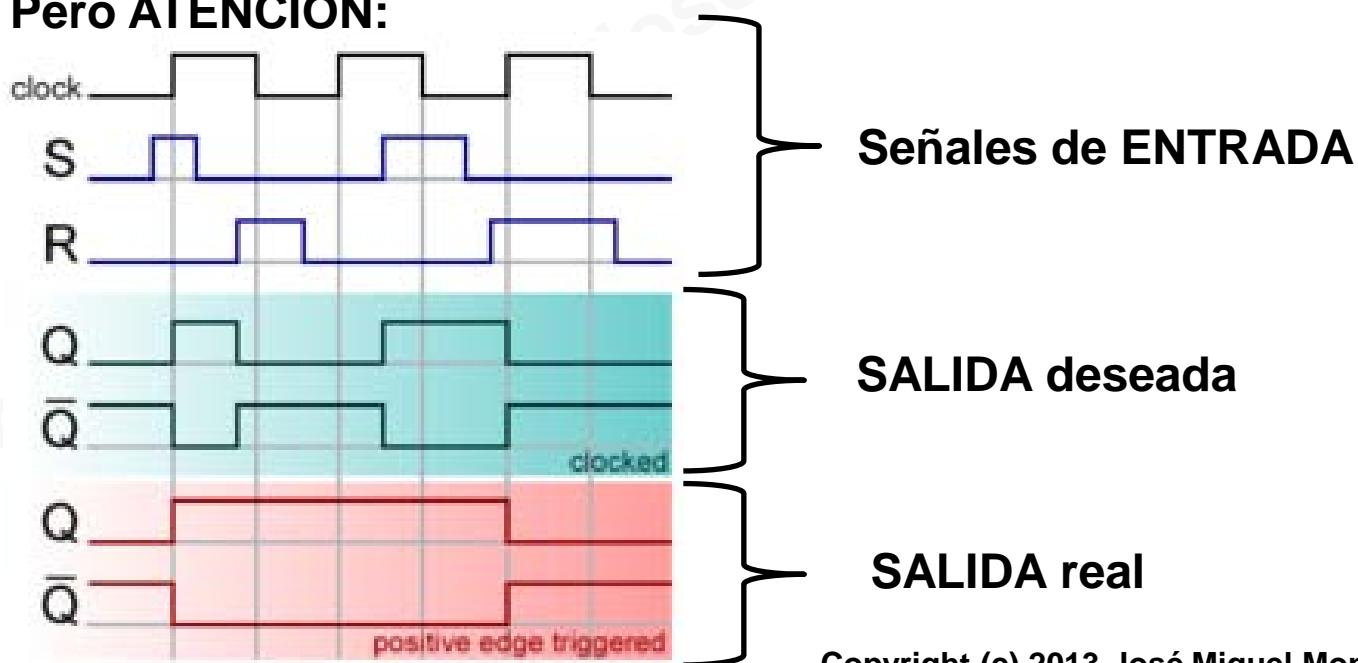
#### 1. BIESTABLES



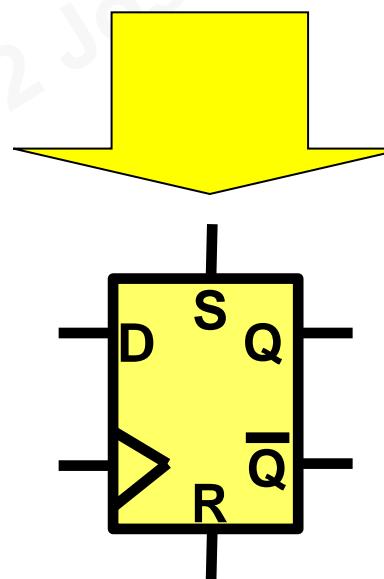
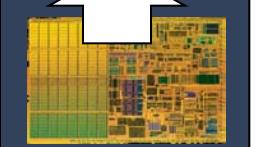
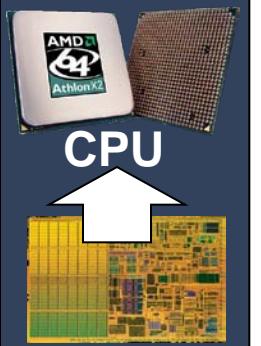
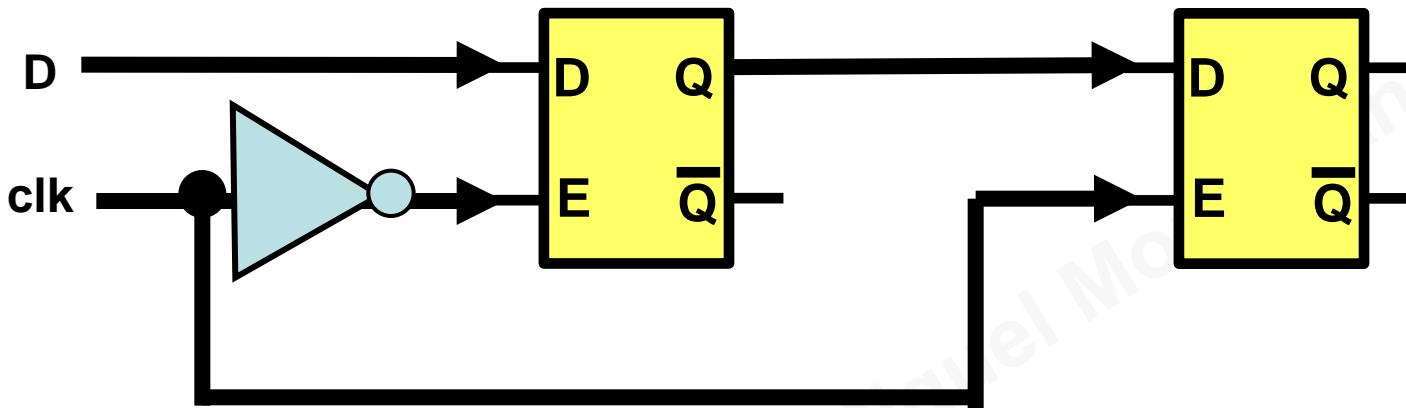


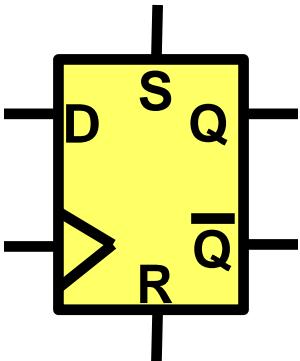
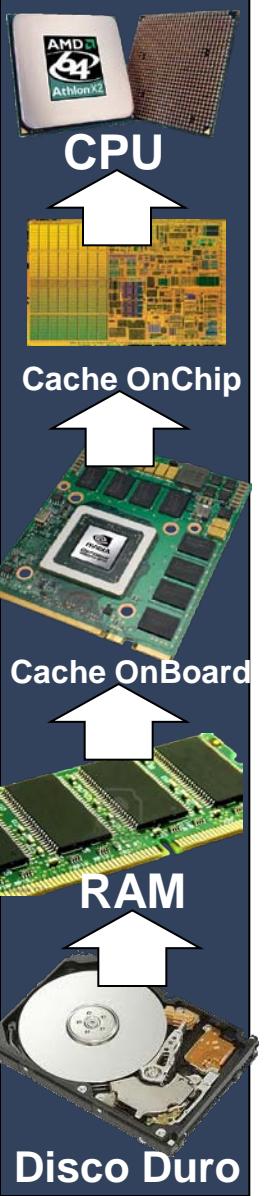
Queremos poner una señal de reloj en la entrada ENABLE

Pero ATENCION:



#### CON DETECCIÓN DE FLANCOS DE SUBIDA DE LA SEÑAL DE RELOJ





#### --MODULO BIESTABLE

```
Library IEEE;  
Use IEEE.STD_LOGIC_1164.all;
```

```
ENTITY bies IS
```

```
port (D: IN STD_LOGIC;  
      clk: IN STD_LOGIC;  
      R: IN STD_LOGIC;  
      q: OUT STD_LOGIC);
```

```
ARCHITECTURE bies OF bies IS  
BEGIN
```

```
PROCESS(clk,R)  
BEGIN
```

```
IF R='1' THEN
```

```
  q <='0';
```

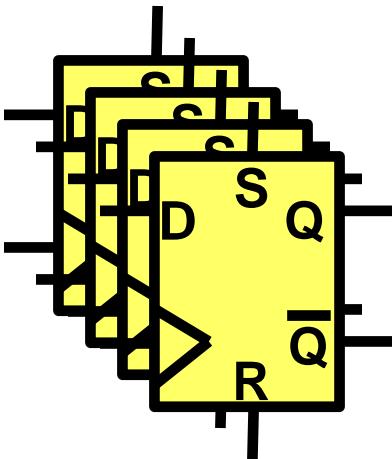
```
ELSIF clk'event AND clk='1' THEN
```

```
  q <=D;
```

```
END IF;
```

```
END PROCESS;
```

```
END bies;
```



--MODULO BIESTABLE

Library IEEE;

Use IEEE.STD\_LOGIC\_1164.all;

ENTITY bies IS

```
port (D: IN STD_LOGIC_VECTOR(3 downto 0);
      clk: IN STD_LOGIC;
      R: IN STD_LOGIC;
      q: OUT STD_LOGIC_VECTOR(3 downto 0));
```

ARCHITECTURE bies OF bies IS

BEGIN

PROCESS(clk,R)

BEGIN

IF R='1' THEN

q <= "0000";

ELSIF clk'event AND clk='1' THEN

q(3 downto 0) < =D(3 downto 0);

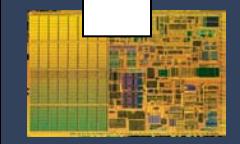
END IF;

END PROCESS;

END bies;



CPU



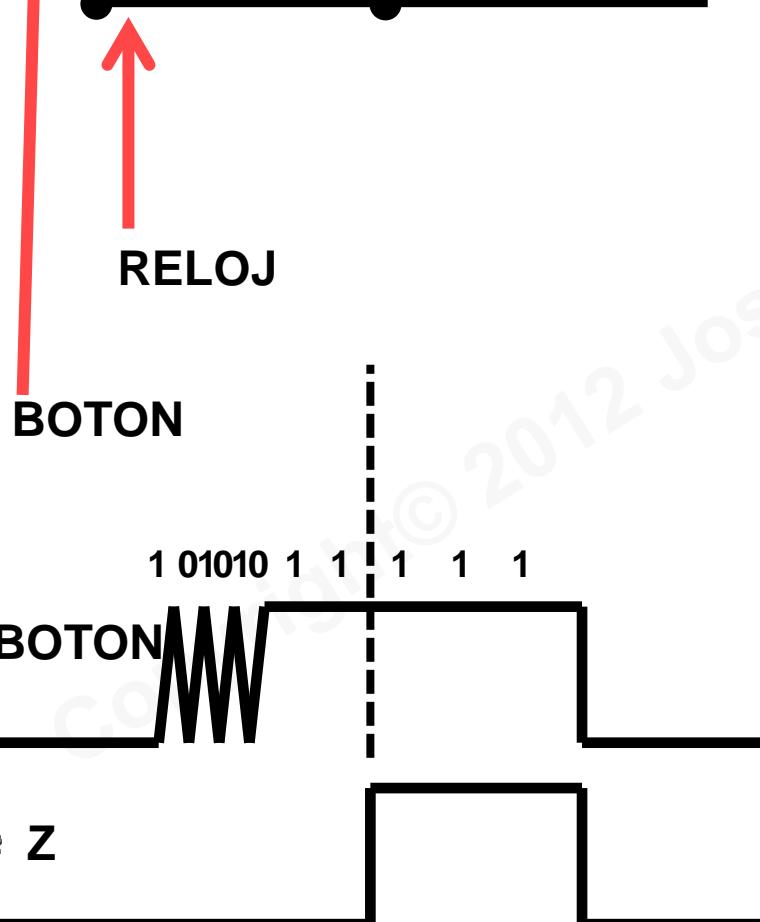
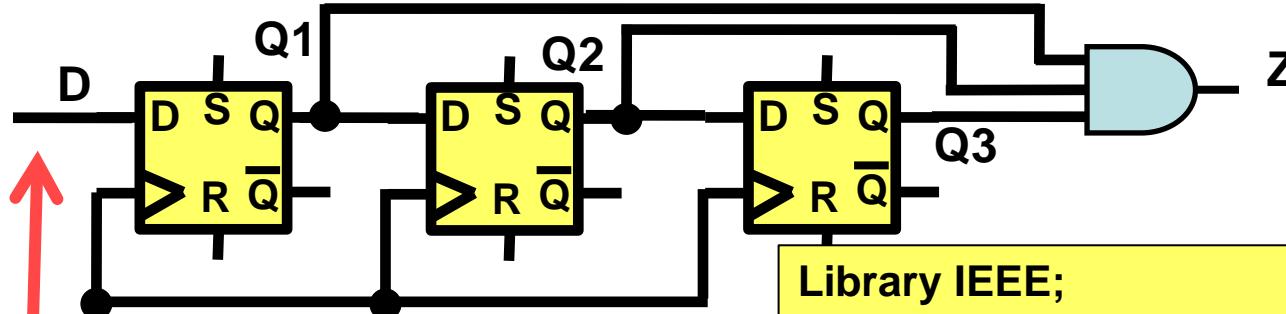
Cache OnChip



RAM

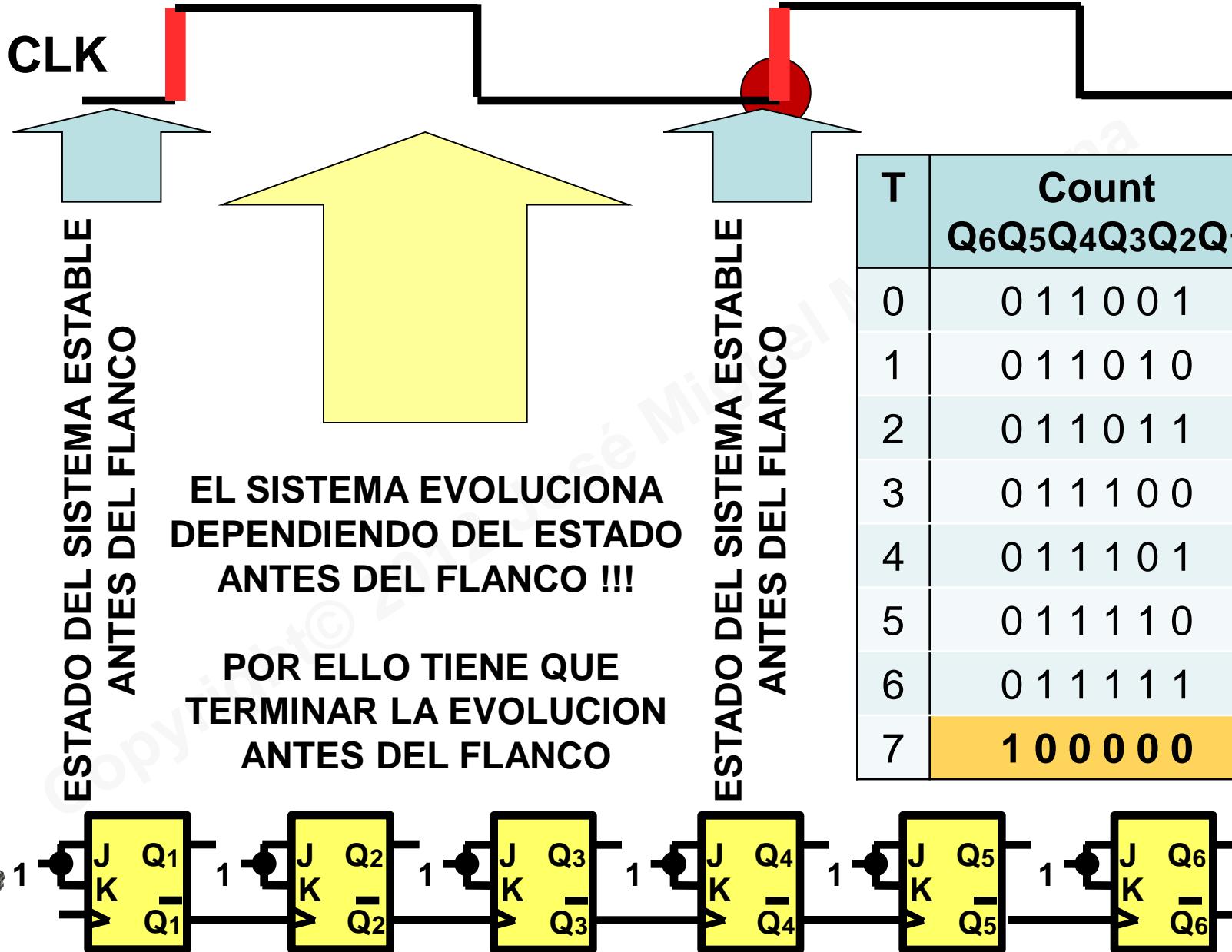
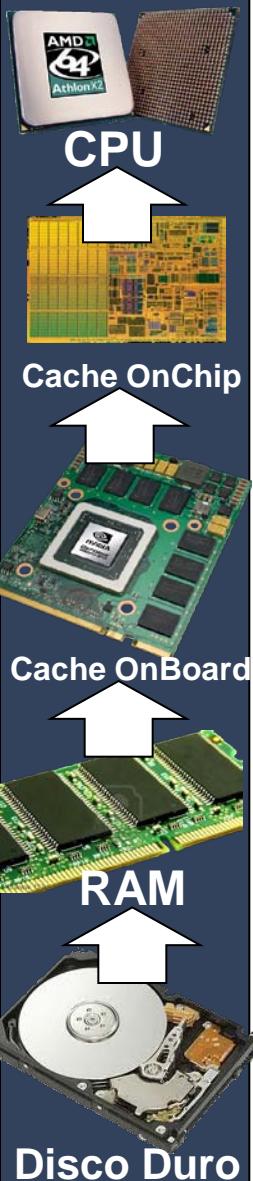


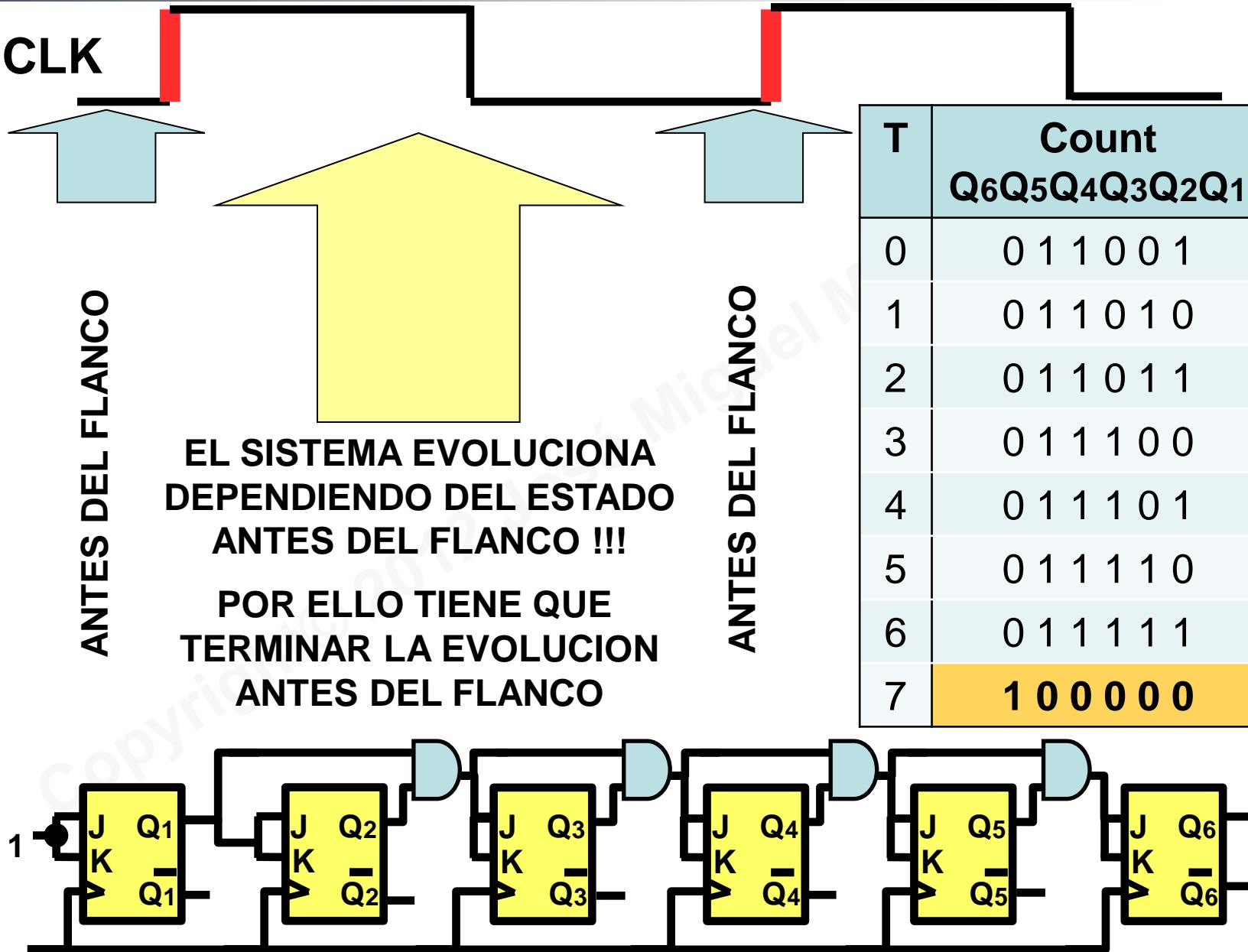
Disco Duro

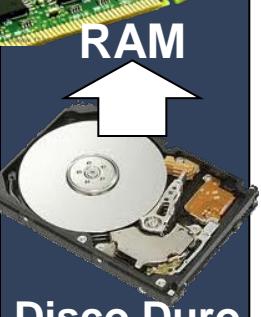
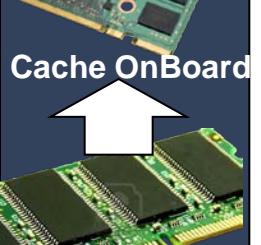
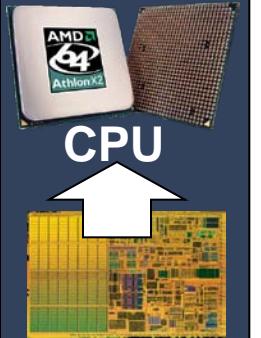


```

Library IEEE;
Use IEEE.STD_LOGIC_1164.all;
ENTITY detector IS
  port ( clk: IN STD_LOGIC;
         D: IN STD_LOGIC;
         Z: OUT STD_LOGIC );
ARCHITECTURE detector OF detector IS
  SIGNAL Q1, Q2, Q3: STD_LOGIC;
BEGIN
  PROCESS(clk)
  BEGIN
    IF clk'event AND clk='1' THEN
      Q1 <= D;
      Q2 <= Q1;
      Q3 <= Q2;
    END IF;
    Z<= Q1 and Q2 and Q3;
  END PROCESS;
END detector;
  
```







### SOLUCION A

```

process(clk)
begin
  if clk'event and clk='1' then
    count <=count+'1';
  end if;
end process;

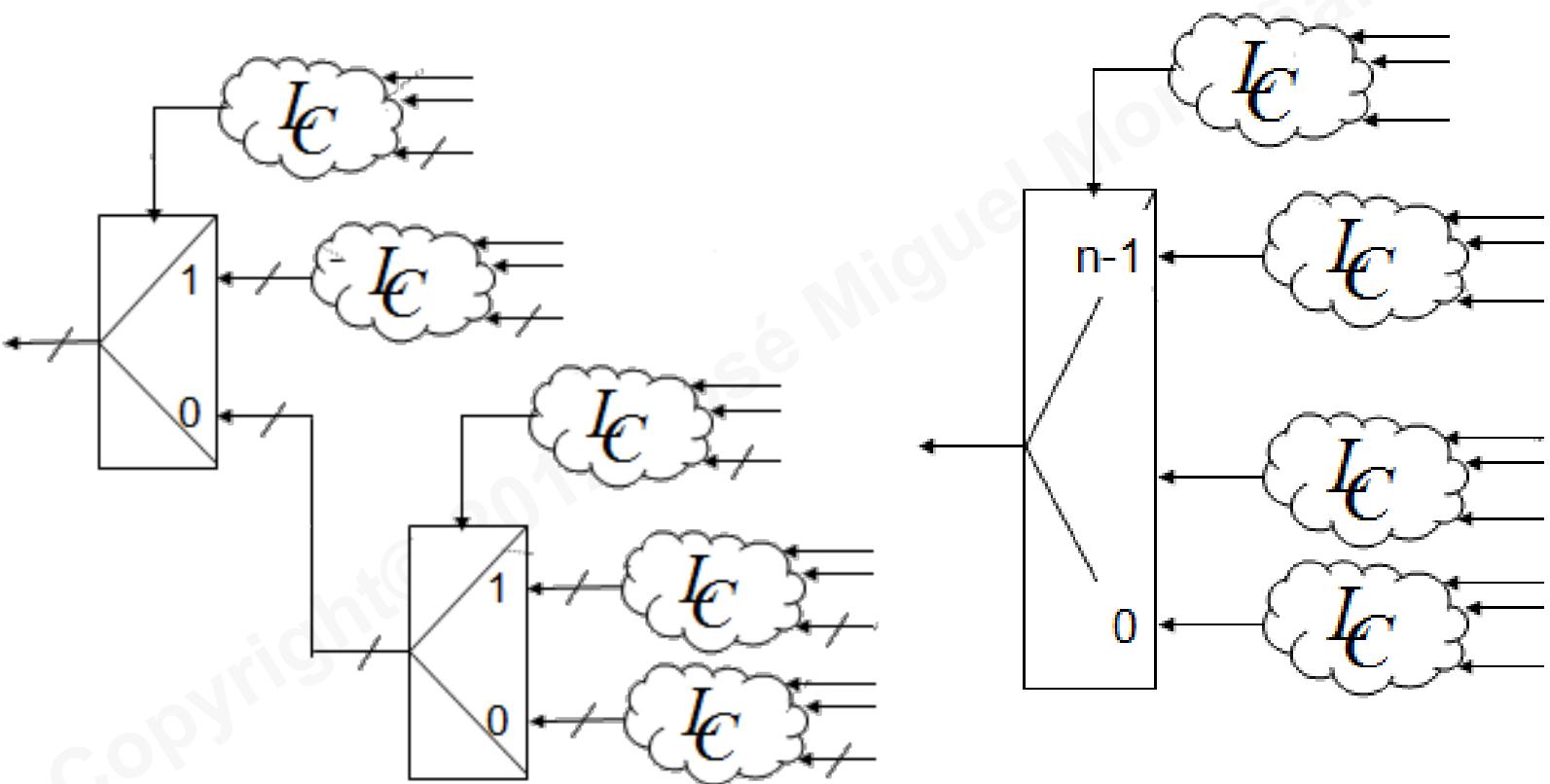
process(count )
  if count ="0000" then
    output1 <= "0010";
  elsif count ="0001" then
    output1 <= "0011";
  elsif count ="0010" then
    output1 <= "0101";
  elsif count ="0011" then
    output1 <= "0110";
  else
    output1<="0111";
  end if;
end process;
  
```

### SOLUCION B

```

process(clk)
begin
  if clk'event and clk='1' then
    count <=count+'1';
  end if;
end process;

process(count )
case count is
  when "0000" => output1 <="0010";
  when "0001" => output1 <="0011";
  when "0010" => output1 <="0101";
  when "0011" => output1 <="0110";
  when others => output1 <="0111";
end case;
end process;
  
```

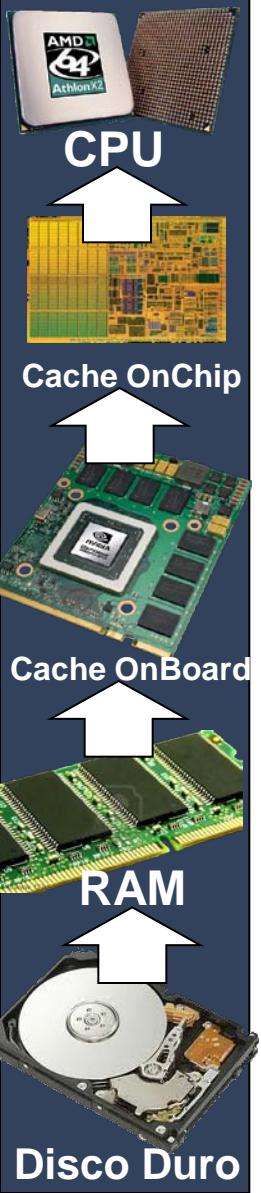


Figuras del prof. Mendias.



# DISEÑO AUTOMÁTICO DE SISTEMAS

## Curso 2013-2014



# Práctica 1

## MAQUINA DE ESTADOS

**Profesor teoría y laboratorio  
José Miguel Montaña Aliaga.**

e-mail: [jmontanana@fdi.ucm.es](mailto:jmontanana@fdi.ucm.es)

**Tutorías: 1er cuatrimestre Miércoles : 10:00-13:00  
2ndo cuatrimestre Miércoles 14:00-17:00**

**Fac Informática dpch 421, Fac. Física dpcho 225  
Apuntes: Campus Virtual**

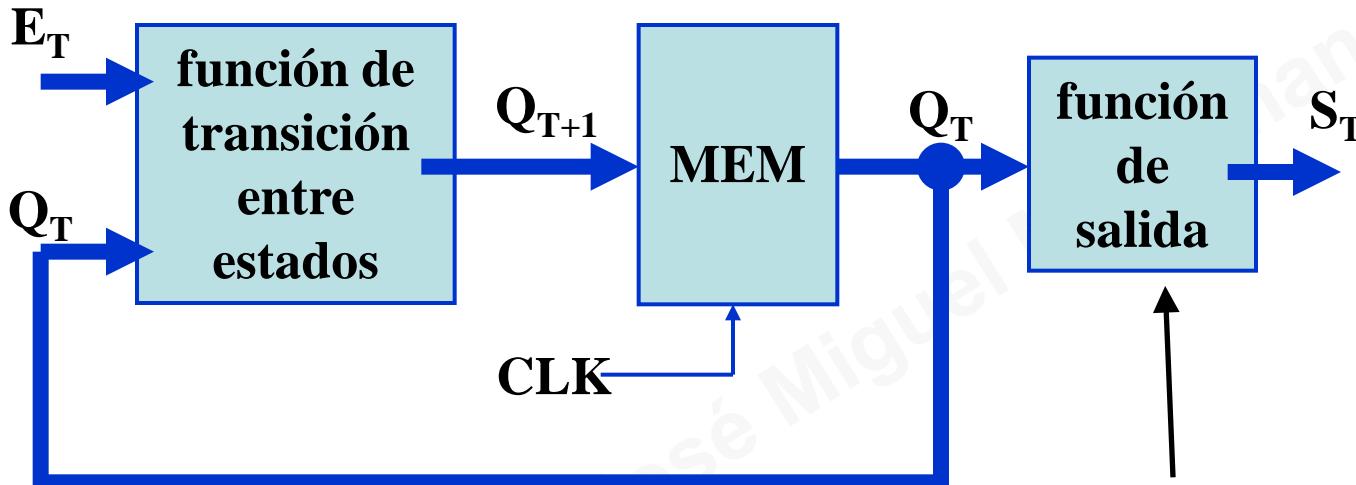
**D E P A R T A M E N T O D E  
A R Q U I T E C T U R A D E C O M P U T A D O R E S  
Y A U T O M Á T I C A**

Dpto. de Arquitectura de Computadores y Automática  
Facultad de Informática.  
Universidad Complutense de Madrid.





## Autómata de Moore



la salida solo depende de las variables de estado

$$S_T = f(Q_T)$$

$$Q_{T+1} = f(E_T, Q_T)$$

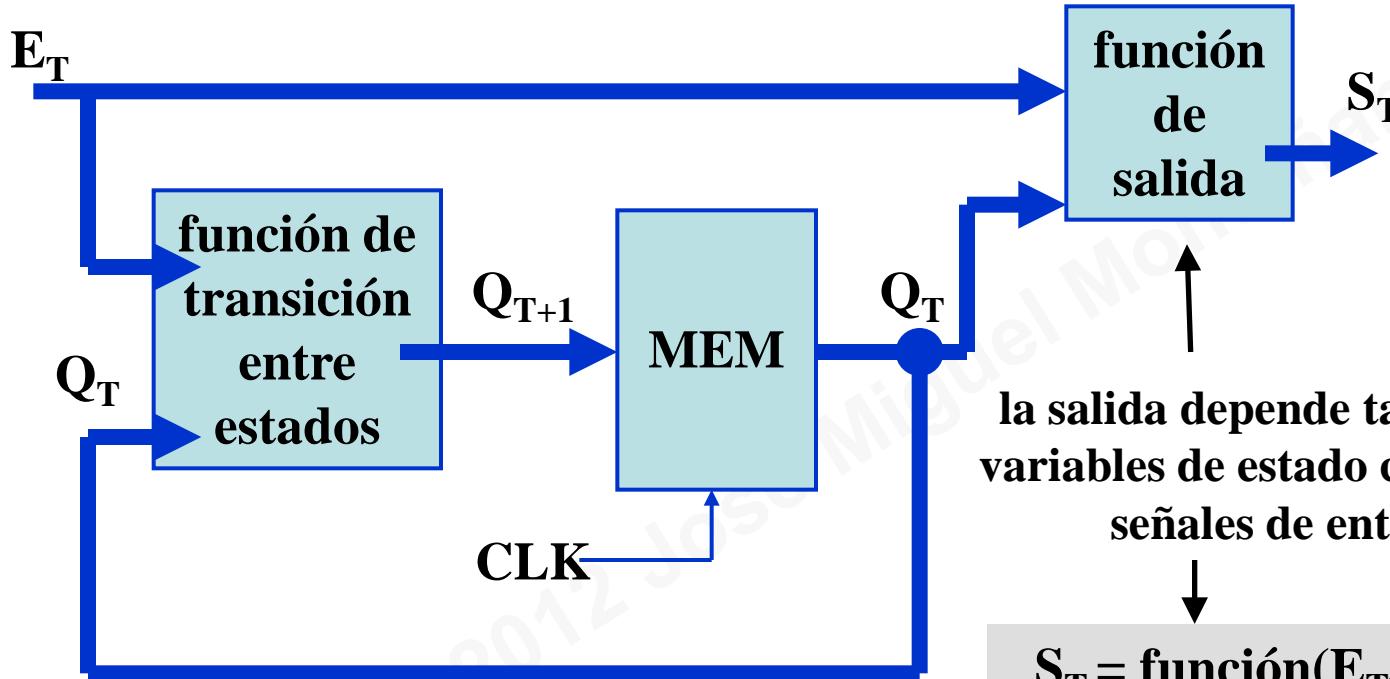
**LA SALIDA SÓLO CAMBIA EN LOS FLANCOS ACTIVOS DE LA SEÑAL DE RELOJ**

Función de salida más sencilla

autómata: Máquina de estados finitos

Copyright (c) 2013-José Miguel Montañana Aliaga

## Autómata de Mealy



**LA SALIDA PUEDE CAMBIAR EN EL INSTANTE EN QUE CAMBIEN LAS ENTRADAS, INDEPENDIENTEMENTE DE LA SEÑAL DE RELOJ**

**Menos estados; menos circuitos de memoria**

autómata: Máquina de estados finitos

Copyright (c) 2013-José Miguel Montañana Aliaga





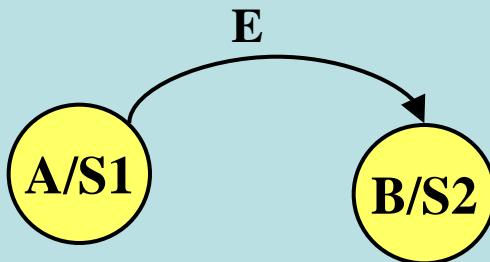
## Descripción de sistemas: diagramas de estado

Cada estado se representa por un círculo

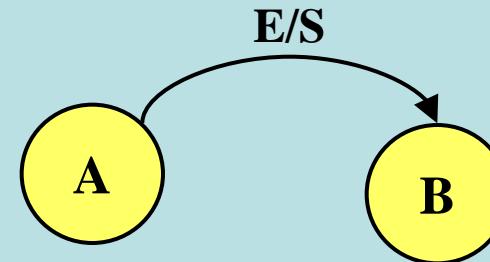
Cada transición se representa por flechas entre estados

Moore: la salida se define únicamente por el estado

Mealy: la salida se define por el estado y las entradas



Transición entre el estado A y el B  
para un autómata de Moore



Transición entre el estado A y el B  
para un autómata de Mealy

### Diseño de sistemas secuenciales

- Descripción del funcionamiento del sistema
- Tabla y diagrama de estados
- Minimización de estados
  - Moore: mismas salidas y mismo estado
  - Mealy: mismo estado
- Asignación de variables de estado
- Tabla de transiciones
- Tabla de excitación (tabla de verdad)
  - Moore: una para las variables de estado y otra para las salidas
  - Mealy: única

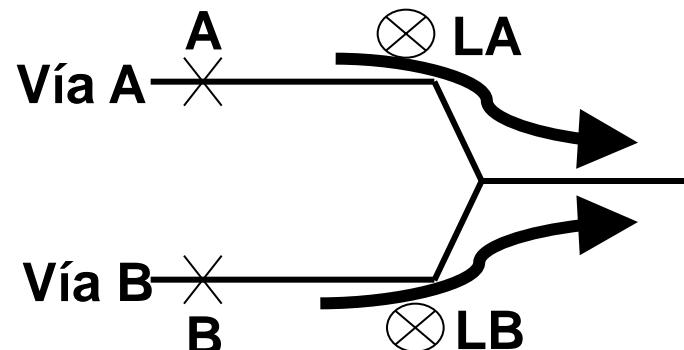


## Diseño de un sistema secuencial: Enunciado



Se tiene una confluencia de dos vías férreas con el mismo sentido de circulación. Cada vía está dotada de un dispositivo que detecta si hay un vehículo en ella (A y B) y de un semáforo (LA y LB), ambos a cierta distancia del punto de unión.

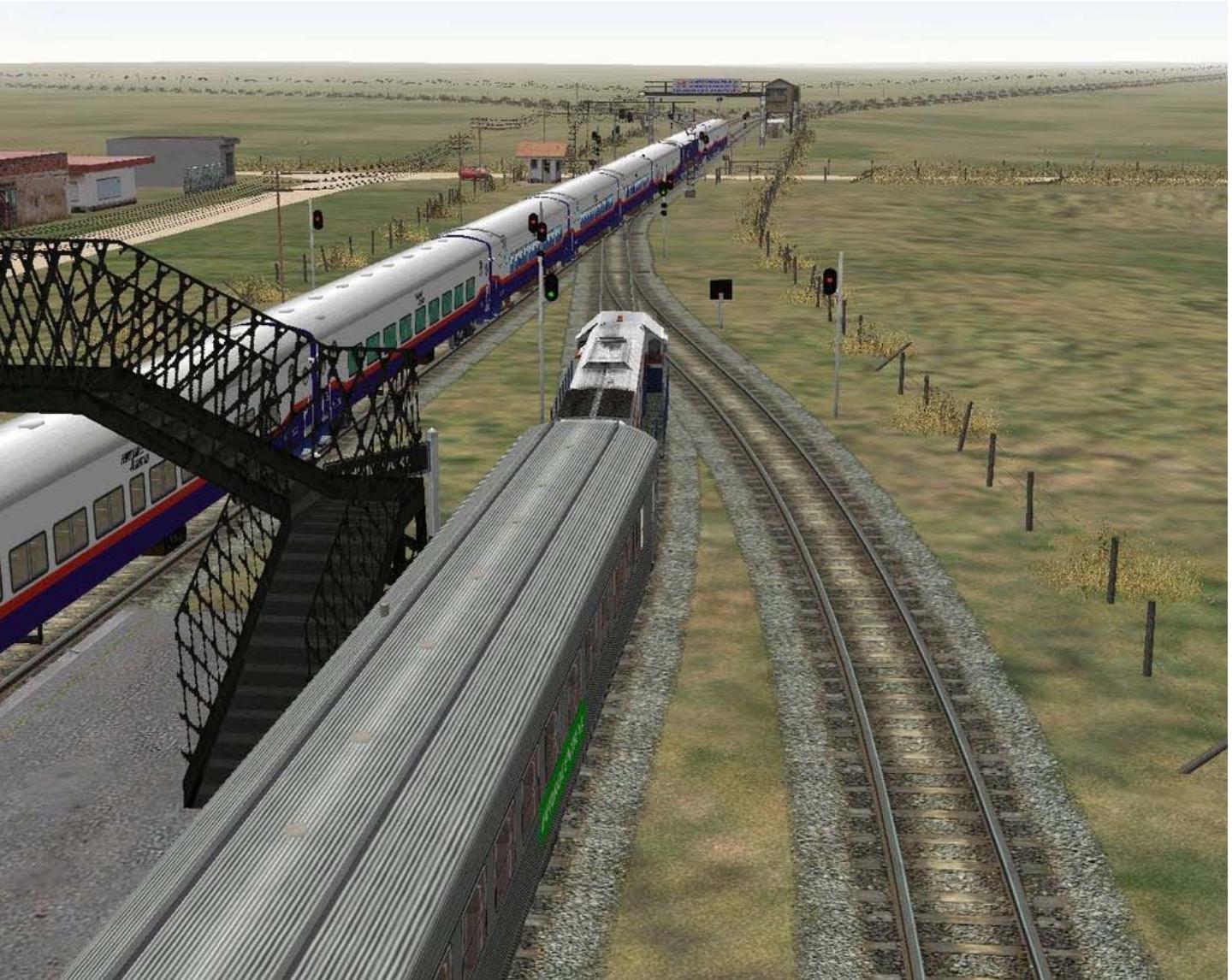
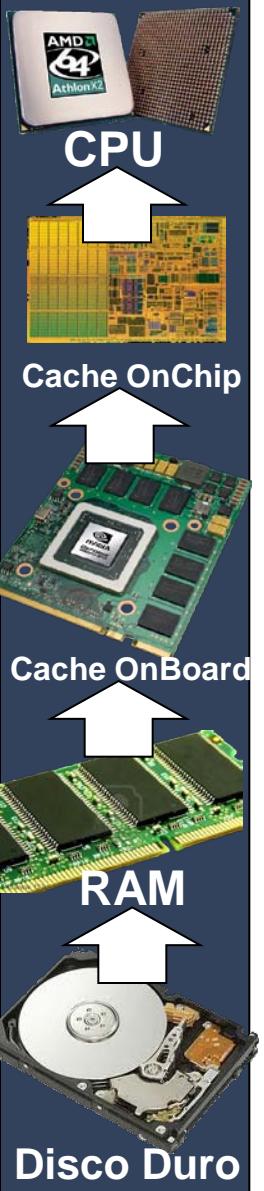
Cuando llega un vehículo a cualquiera de los detectores A o B debe activarse la luz de detención de la otra vía, LB o LA respectivamente, con el fin de detener a cualquier vehículo que llegue por esa vía hasta que el primero haya abandonado su detector.



# DISEÑO AUTOMÁTICO DE SISTEMAS

## Curso 2013-2014

Práctica 1



## Diseño de sistemas secuenciales: Mealey

Las salidas cambian cuando lo hacen las entradas

- 1: Ningún vehículo
- 2: Llega vehículo por B
- 3: Llega vehículo por A
- 4: Entró vehículo por A, otro espera en B
- 5: Entró vehículo por B, otro espera en A

Diagrama

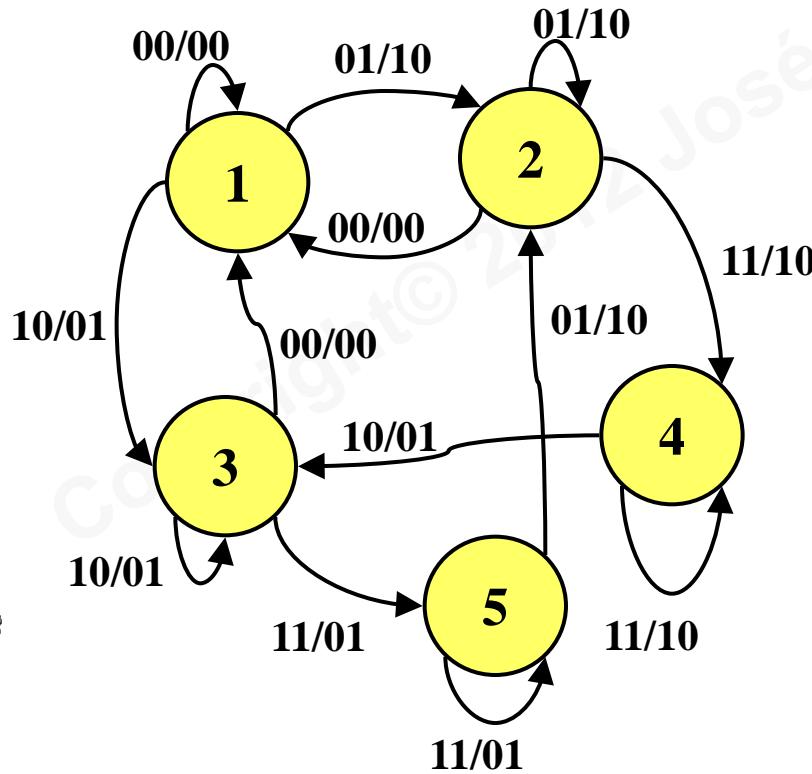
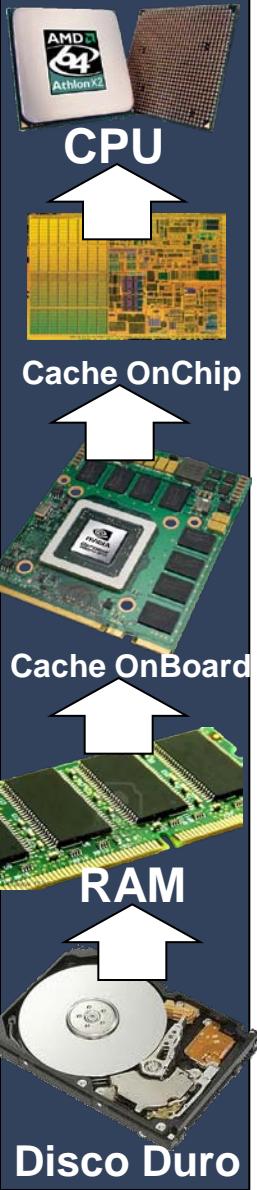


Tabla de estados

ESTADO	AB				LUZ A	LUZ B
	00	01	10	11		
1	1	2	3	-	0	0
2	1	2	-	4	1	0
3	-	-	3	5	0	1
4	-	-	3	4	1	0
5	-	2	-	5	0	1



## Diseño de sistemas secuenciales: Moore

Las salidas sólo dependen de las variables de estado

- 1: Ningún vehículo
- 2: Llega vehículo por B
- 3: Llega vehículo por A
- 4: Entró vehículo por A, otro espera en B
- 5: Entró vehículo por B, otro espera en A

Diagrama

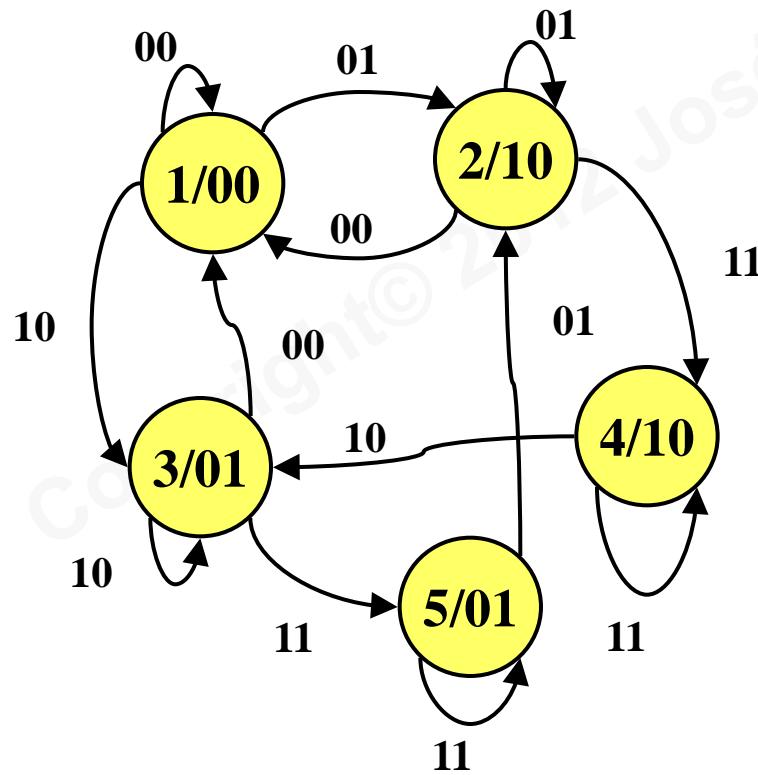
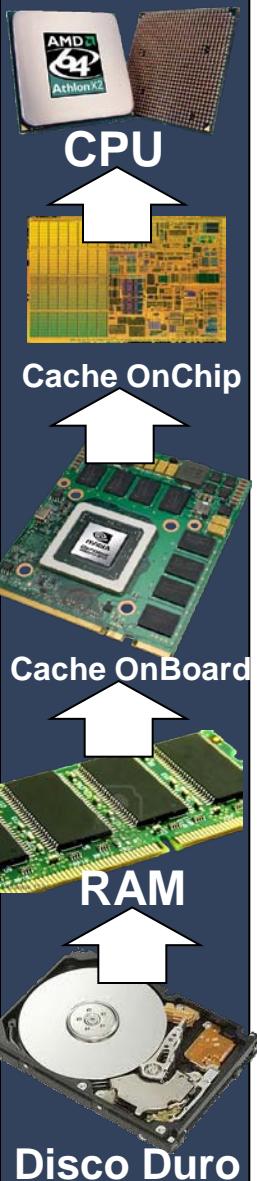


Tabla de estados

ESTADO	AB (entrada)				SALIDA	
	00	01	10	11	LUZ A	LUZ B
1	1	2	3	-	0	0
2	1	2	-	4	1	0
3	-	-	3	5	0	1
4	-	-	3	4	1	0
5	-	2	-	5	0	1



## Diseño de sistemas secuenciales: Minimización de estados



ESTADO	AB (entrada)				SALIDA	
	00	01	10	11	LUZ A	LUZ B
1	1	2	3	-	0	0
2	1	2	-	4	1	0
3	-	-	3	5	0	1
4	-	-	3	4	1	0
5	-	2	-	5	0	1

Por Moore podemos fusionar los estados 2/4 y el 3/5, (misma transición Estado\_sig=f(entrada) y misma salida)

ESTADO	AB (entrada)				SALIDA	
	00	01	10	11	LUZ A	LUZ B
1	1	2	3	-	0	0
2,4	1	2	3	4	1	0
3,5	-	2	3	5	0	1

Por Mealy podemos fusionar los estados 1/2/4 y el 3/5, (mismas transiciones Estado\_sig=f(entrada))

ESTADO	AB (entrada)			
	00	01	10	11
1,2,4	1/00	2/10	3/01	4/10
3,5	-	2/10	3/01	5/01

## Diseño de un sistema secuencial: Asignación de variables de estado

Codificamos cada estado con un código diferente a cada estado, podemos asignarles valores con cualquier criterio.

Máquina de Moore tiene tres estados.

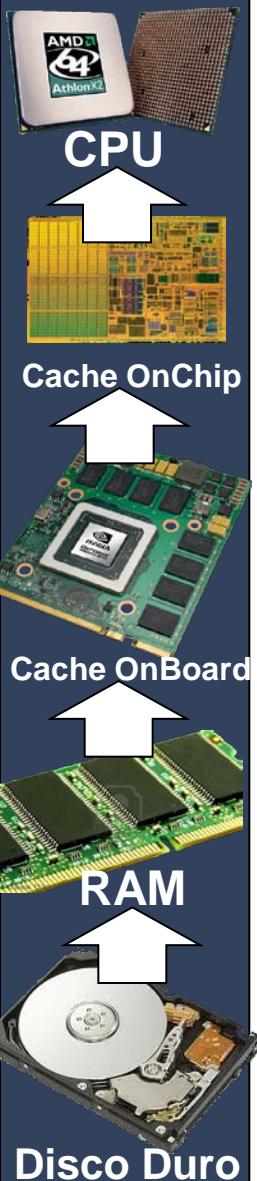
Q1	Q2	ESTADO	AB (entrada)				SALIDA	
			00	01	10	11	LUZ A	LUZ B
0	0	1	1	2	3	-	0	0
1	0	2,4	1	2	3	4	1	0
0	1	3,5	-	2	3	5	0	1

Asignar las variables de estado con las salidas

Máquina Mealy tiene dos estados.

Q	ESTADO	AB (entrada)			
		00	01	10	11
0	1,2,4	1/00	2/10	3/01	4/10
1	3,5	-	2/10	3/01	5/01

## Diseño de un sistema secuencial: Tabla de transiciones



Sustituimos cada estado por el valor de las variables de estado que le hemos asignado en la codificación.

Máquina de Moore tiene tres estados.

Q1	Q2	AB (entrada)				SALIDA	
		00	01	10	11	LUZ A	LUZ B
0	0	00	10	01	-	0	0
1	0	00	10	01	10	1	0
0	1	-	10	01	01	0	1

Asignar las variables de estado con las salidas simplifica la generación de la salida

Máquina Mealy tiene dos estados.

Q	AB (entrada)			
	00	01	10	11
0	0/00	0/10	1/01	0/10
1	-	0/10	1/01	1/01

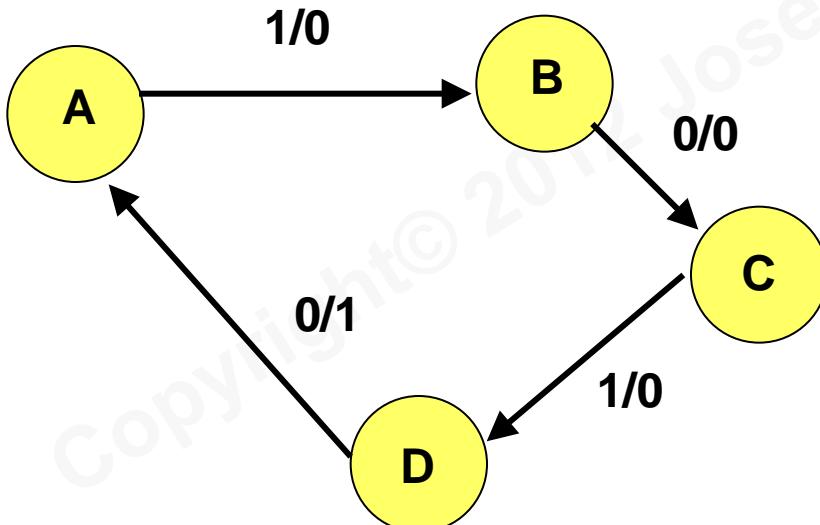
Valor de las variables de estado en el instante T+1

## 5. EJEMPLO DE UN DETECTOR DE SECUENCIA

## EJEMPLO DE UN DETECTOR DE SECUENCIA(simplificado/Mealy):

- 1.- Dibujar diagrama de estados
- 2.- Contar el número de estados (N)
- 3.- Calcular cuantos biestables =  $\log_2(N)$
- 4.- Asignar un único identificador a cada estado
- 5.- Escribir las ecuaciones

**Secuencia 1010:**



Estado Presente	Siguiente Estado/SALIDA	
	Entrada 0	Entrada 1
A		B/0
B	C/0	
C		D/0
D	A/1	

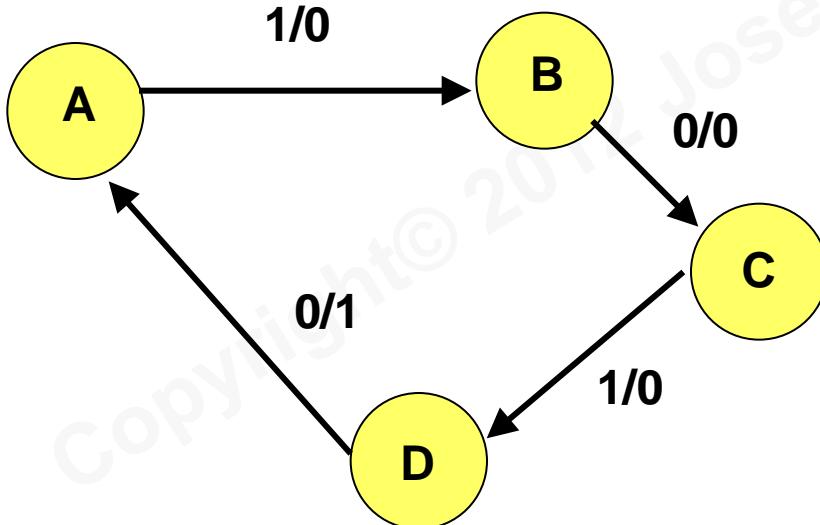


## 5. EJEMPLO DE UN DETECTOR DE SECUENCIA

## EJEMPLO DE UN DETECTOR DE SECUENCIA(simplificado/Mealy):

- 1.- Dibujar diagrama de estados
- 2.- Contar el número de estados (N)
- 3.- Calcular cuantos biestables =  $\log_2(N)$
- 4.- Asignar un único identificador a cada estado
- 5.- Escribir las ecuaciones

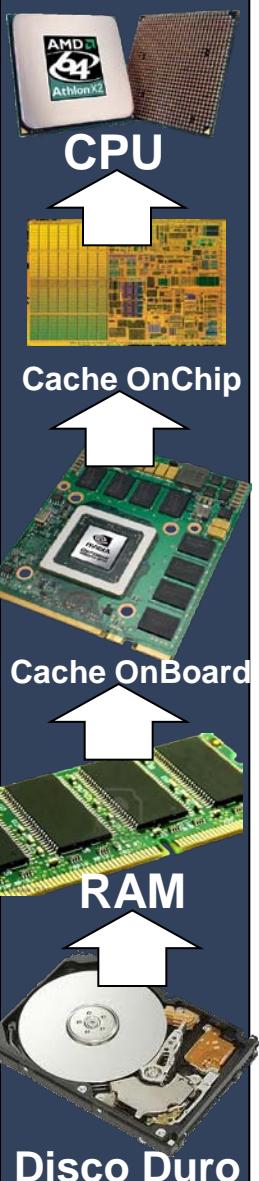
**Secuencia 1010:**



Estado Presente	Siguiente Estado/ SALIDA	
	Entrada 0	Entrada 1
A	A/0	B/0
B	C/0	B/0
C	C/0	D/0
D	A/1	D/0

SUPONIENDO PROCESO EN 4 CICLOS

4 ESTADOS → 2 BIESTABLES

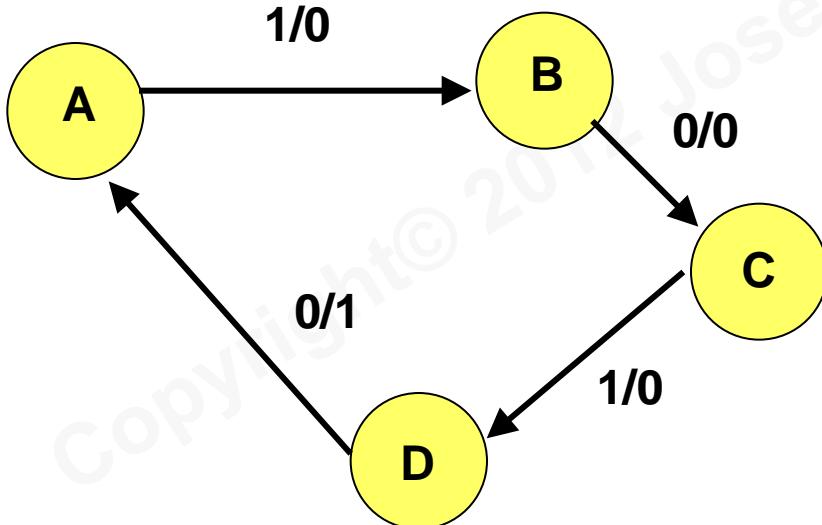


### 5. EJEMPLO DE UN DETECTOR DE SECUENCIA

#### EJEMPLO DE UN DETECTOR DE SECUENCIA(simplificado/Mealy):

- 1.- Dibujar diagrama de estados
- 2.- Contar el número de estados (N)
- 3.- Calcular cuantos biestables =  $\log_2(N)$
- 4.- Asignar un único identificador a cada estado
- 5.- Escribir las ecuaciones

**Secuencia 1010:**



Estado Presente	Siguiente Estado/SALIDA	
	Entrada 0	Entrada 1
A=00	00/0	01/0
B=01	10/0	01/0
C=10	10/0	11/0
D=11	00/1	11/0

**SUPONIENDO PROCESO EN 4 CICLOS**

**4 ESTADOS → 2 BIESTABLES**



## 5. EJEMPLO DE UN DETECTOR DE SECUENCIA

## EJEMPLO DE UN DETECTOR DE SECUENCIA(simplificado/Mealy):

Estado Presente	SALIDA	
	Entrada 0	Entrada 1
F1 F2		
A=00	0	0
B=01	0	0
C=10	0	0
D=11	1	0

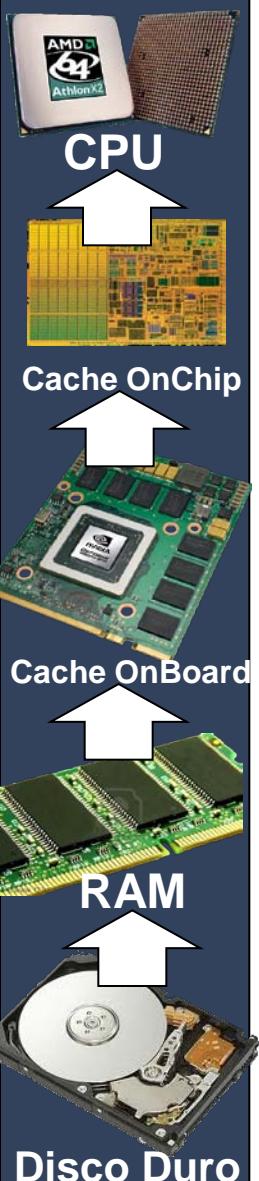
SALIDA=F1 AND F2 AND NOT ENTRADA

Estado Presente	Siguiente Estado F1	
	Entrada 0	Entrada 1
F1 F2		
A=00	0	0
B=01	1	0
C=10	1	1
D=11	0	1

Estado Presente	Siguiente Estado F2	
	Entrada 0	Entrada 1
F1 F2		
A=00	0	1
B=01	0	1
C=10	0	1
D=11	0	1

$F1 = ((F1 \text{ XOR } F2) \text{ AND NOT ENTRADA})$   
 OR (F1 and ENTRADA)

F2= ENTRADA





#### --MODULO BIESTABLE

Library IEEE;

Use IEEE.STD\_LOGIC\_1164.all;

ENTITY bies IS port (

```
D: IN STD_LOGIC_VECTOR(3 downto 0);
clk: IN STD_LOGIC;
R: IN STD_LOGIC; --linea de reset
q: OUT STD_LOGIC_VECTOR(3 downto 0));
```

ARCHITECTURE bies OF bies IS

BEGIN

PROCESS(clk, present\_state )

BEGIN

CASE present\_state is

when "001" =>

next\_state<="101";

when "101" =>

next\_state<="011";

when "011" =>

next\_state<="001";

when others =>

next\_state<="001";

END CASE;

END PROCESS;

END bies;

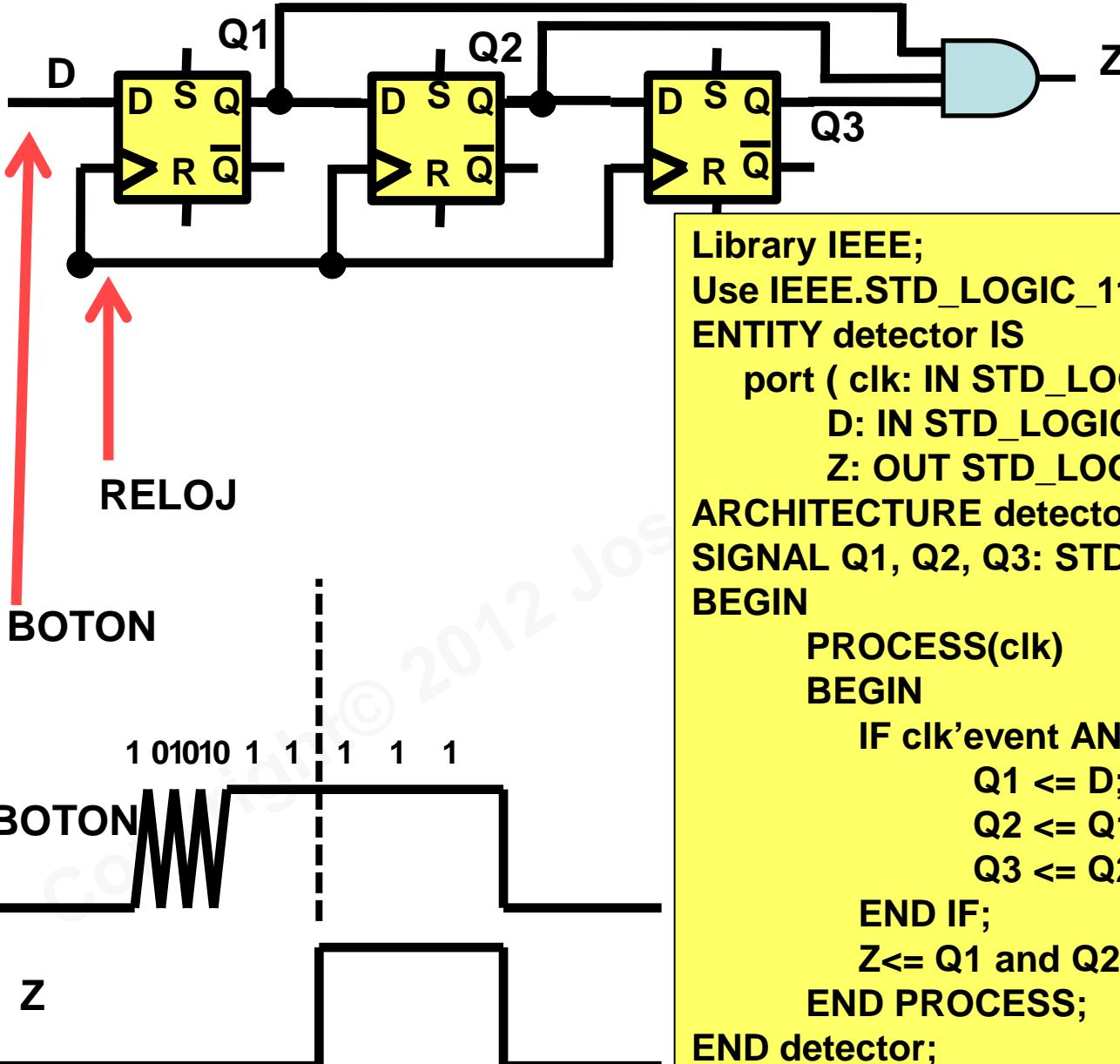
PROCESS(clk, R)

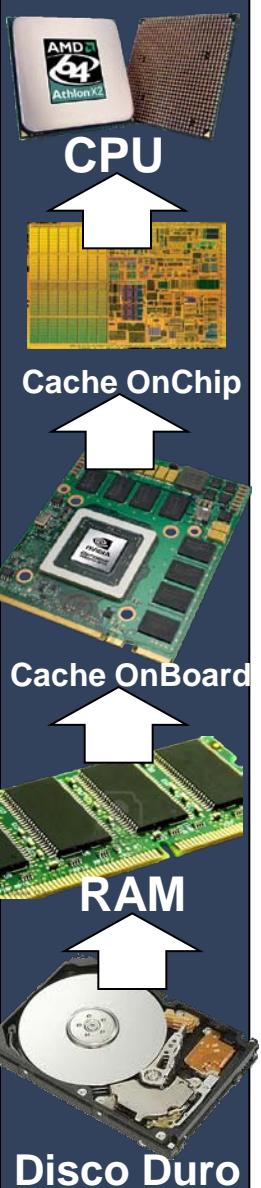
BEGIN

```
IF clk'event and clk='1' THEN
    present_state <= next_state;
```

END IF;

END PROCESS;





## Levels of Abstraction

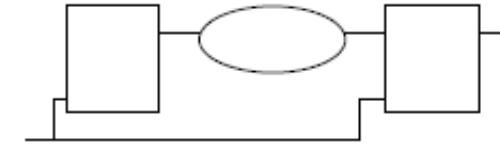
Fewer details,  
Faster design  
entry and  
simulation

Behavioral

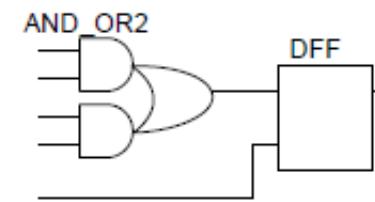


f

RTL

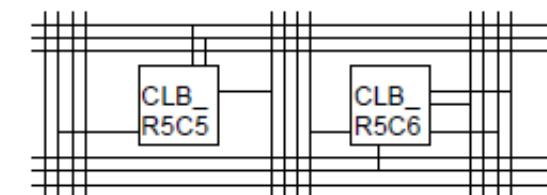


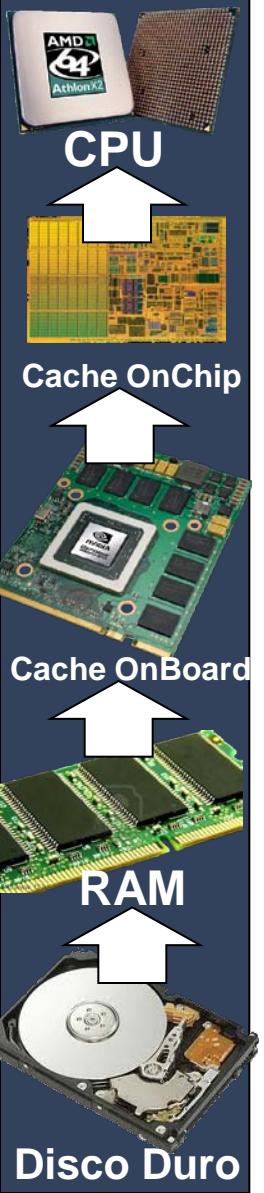
Logic



Technology  
specific details,  
slower design  
entry and  
simulation

Layout

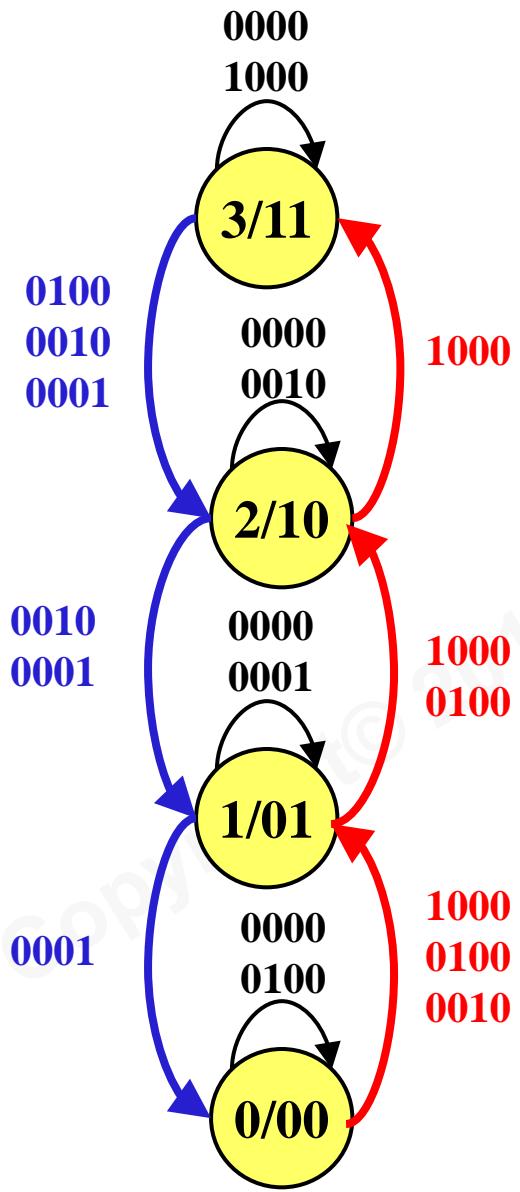




# Ayuda para la Práctica 1.



## Diagrama del ascensor sin memoria: Moore



**Entradas:** 4 bits, uno por cada botón de llamada.

**Nota:** Solo se puede pulsar un botón en cada momento.

**ESTADO:** 2 bits para indicar el piso en el que está.

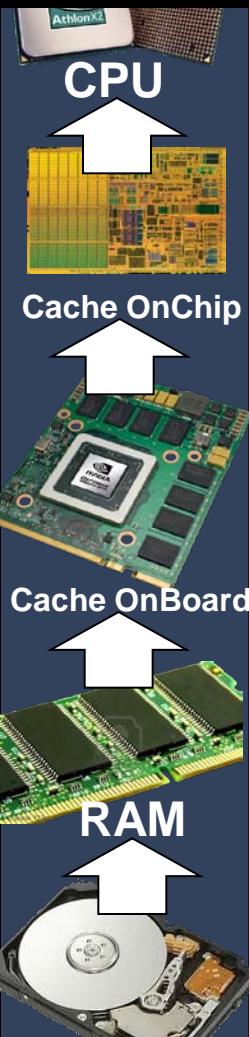


# DISEÑO AUTOMÁTICO DE SISTEMAS

Curso 2013-2014

Práctica 1

## MAQUINA DE ESTADOS ASC SIN MEMORIA



-- Company: UCM  
-- Engineer: J.M. Montañana

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
library UNISIM;
use UNISIM.VComponents.all;
ENTITY semaforo IS
    Port ( clk,reset : in STD_LOGIC;
           b : in STD_LOGIC_VECTOR (3 downto 0);
           leds : out STD_LOGIC_VECTOR (1 downto 0));
END semaforo;
ARCHITECTURE Behavioral of semaforo IS
    SIGNAL CURRENT_STATE,NEXT_STATE :
        STD_LOGIC_VECTOR(1 downto 0);
BEGIN
    PROCESS( clk, reset)
        BEGIN
            IF reset='1' THEN
                CURRENT_STATE<= "00";
            ELSIF clk'event and clk='1' THEN
                CURRENT_STATE<=NEXT_STATE;
            END IF;
        END PROCESS;
        leds<=CURRENT_STATE;
    END IF;
END CASE;
END IF;
END PROCESS.
```

```
PROCESS( CURRENT_STATE, reset, b)
BEGIN
    IF reset='1' THEN
        NEXT_STATE <= "00";
    ELSE
        CASE CURRENT_STATE IS
            WHEN "00" =>
                IF b(1)='1' OR b(2)='1' OR b(3)='1' THEN
                    NEXT_STATE <= "01";
                END IF;
            WHEN "01" =>
                IF b(2)='1' OR b(3)='1' THEN
                    NEXT_STATE <= "10";
                ELSIF b(0)='1' THEN
                    NEXT_STATE <= "00";
                END IF;
            WHEN "10" =>
                IF b(3)='1' THEN
                    NEXT_STATE <= "11";
                ELSIF b(0)='1' OR b(1)='1' THEN
                    NEXT_STATE <= "01";
                END IF;
            WHEN others =>
                IF b(0)='1' OR b(1)='1' OR b(2)='1' THEN
                    NEXT_STATE <= "10";
                END IF;
        END CASE;
    END IF;
END PROCESS.
```

**ALERTA: SOLO FUNCIONA SI SE MANTIENE b hasta llegar al piso.**

Juan Montañana Aliaga



## BOTONES



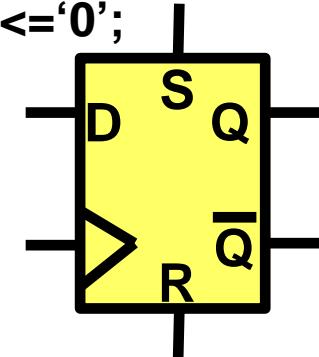
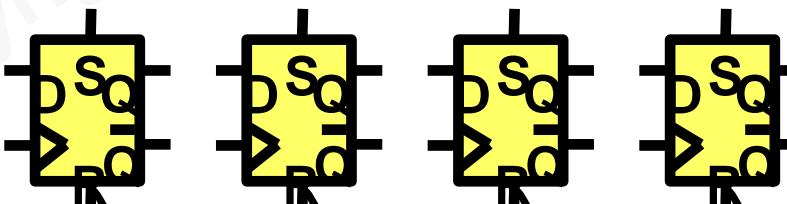
## LEDS



El led se enciende al pulsar el botón → If boton(1)='1' THEN  
led(1)<='1'

El led se apaga cuando llegamos al piso → ELSIF piso=1 THEN  
led(1)<='0';  
END IF;

El estado del led se almacena en un Biestable →

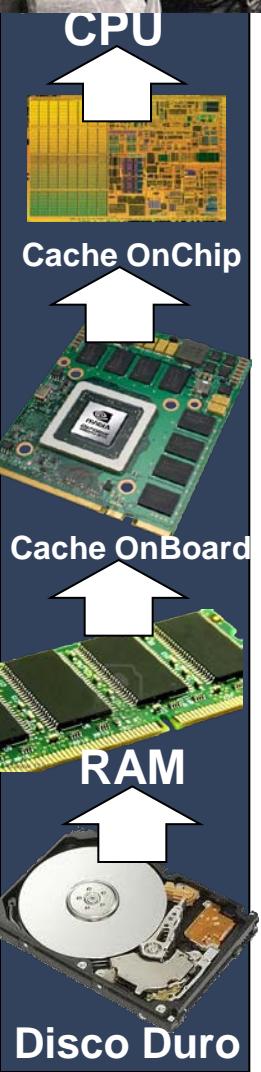


**NOTA:** en el ascensor sin memoria, se atiende si se pulsa un botón  
Solamente cuando todos los led(3 downto 0)="0000"

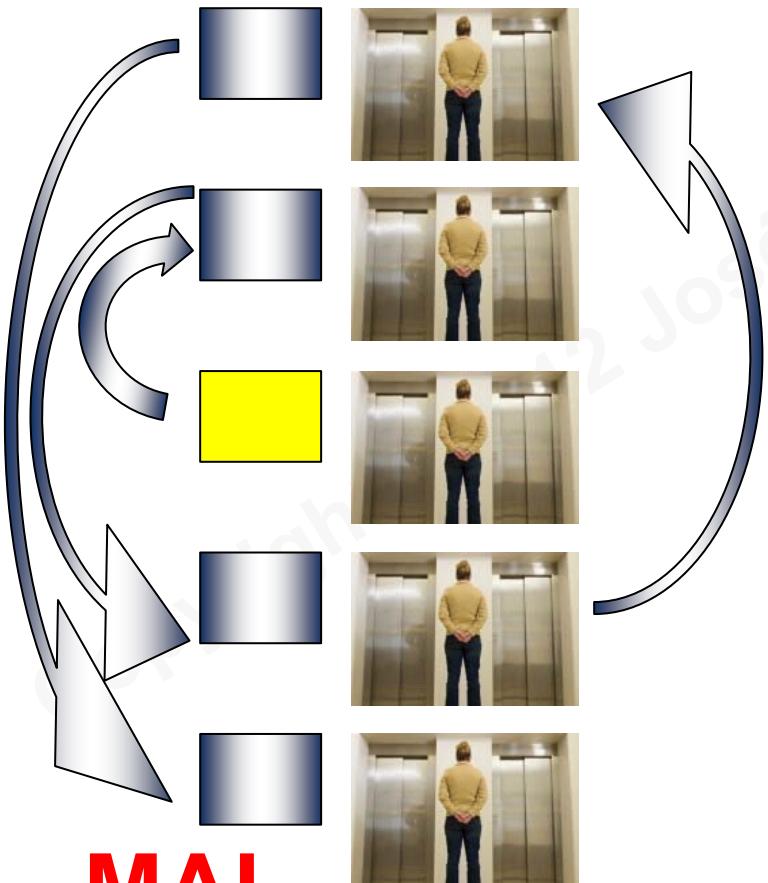


## Ascensor con memoria:

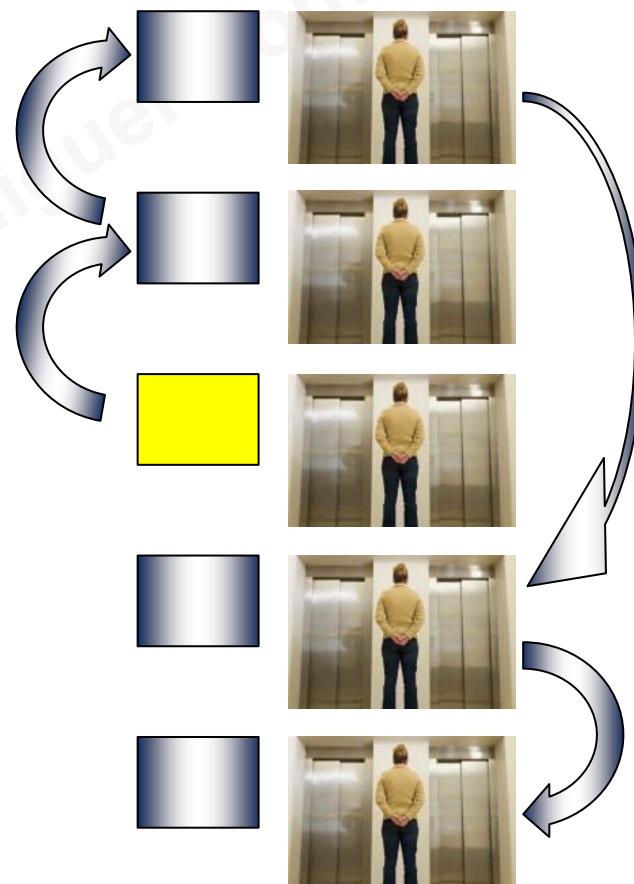
Suponiendo que hay llamadas pendientes de todos los pisos.  
Comenzamos en el piso marcado en amarillo.



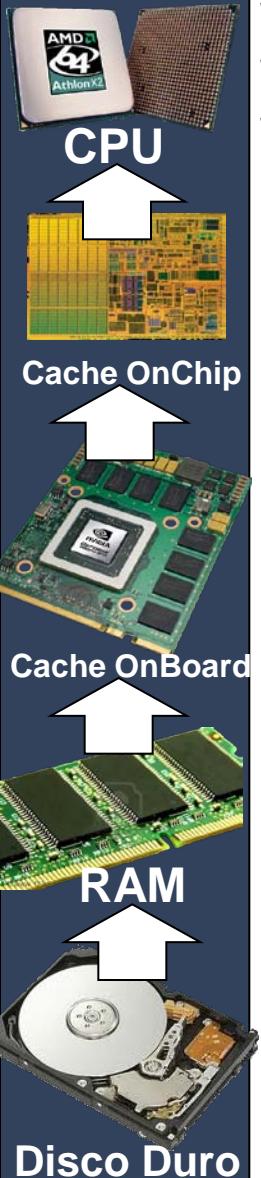
### SOLUCION A



### SOLUCION B



PRIMERO EN UN SENTIDO  
LUEGO EN EL OTRO



**Entradas:** 4 bits B(3 downto 0), uno por cada botón

**SIGNAL Q(3 downto 0)** memoria de las llamadas

**SIGNAL P(1 downto 0)** para indicar el PISO en el que está

**SIGNAL D(1 downto 0)** para indicar DIRECCIÓN

IF  $D(1)='0' \text{ AND } (Q(0)='1' \text{ OR } Q(1)='1' \text{ OR } Q(2)='1')$   
 $D \leqslant "01"$

$D="00"$  STOP

$D="01"$  bajando

$D="10"$  subiendo

IF  $D(1)='0' \text{ AND } (Q(0)='1' \text{ OR } Q(1)='1')$   
 $D \leqslant "01"$

$D(0)='0' \rightarrow$  no esta bajando

$D(1)='0' \rightarrow$  no esta subiendo

IF  $D(1)='0' \text{ AND } Q(0)='1'$   
 $D \leqslant "01"$

IF  $D(0)='0' \text{ AND } (Q(1)='1' \text{ OR } Q(2)='1' \text{ OR } Q(3)='1')$   
 $D \leqslant "10"$

IF  $Q(2,1,0)='000'$

Copyright (c) 2013-José Miguel Montañana Aliaga

IF  $Q(2,1,0)='000'$

$Q(3) \leqslant '0';$   
 $D(1) \leqslant '0';$

IF  $D(0)='0' \text{ AND } Q(3)='1'$   
 $D \leqslant "10"$

$Q(2) \leqslant '0'$   
IF  $Q(3)='00'$  THEN  $D(1) \leqslant '0'$   
IF  $Q(1,2)='00'$  THEN  
 $D(0) \leqslant '0'$

IF  $D(0)='0' \text{ AND } (Q(2)='1' \text{ OR } Q(3)='1')$   
 $D \leqslant "10"$

$Q(1) \leqslant '0';$   
IF  $Q(2,3)='00'$  THEN  $D(1) \leqslant '0';$   
IF  $Q(0)='0'$  THEN  $D(0) \leqslant '0';$

0/00

$Q(0) \leqslant '0'; \quad D(0) \leqslant '0'$



# DISEÑO AUTOMÁTICO

## Curso 2013-2014

-- Company: UCM  
-- Engineer: J.M. Montañana

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
library UNISIM;
use UNISIM.VComponents.all;
ENTITY semaforo IS
    Port ( clk,reset : in STD_LOGIC;
            b : in STD_LOGIC_VECTOR (3 downto 0);
            leds : out STD_LOGIC_VECTOR (1 downto 0));
END semaforo;
ARCHITECTURE Behavioral of semaforo IS
    SIGNAL CURRENT_STATE,NEXT_STATE :
        STD_LOGIC_VECTOR(1 downto 0);
    SIGNAL Q: STD_LOGIC_VECTOR(3 DOWNTO 0);
    SIGNAL D: STD_LOGIC_VECTOR(1 DOWNTO 0);--direcciones, arriba=10, abajo=01 , stop=00
BEGIN
    PROCESS( clk, reset)
        BEGIN
            IF reset='1' THEN
                CURRENT_STATE<= "00";
            ELSIF clk'event and clk='1' THEN
                CURRENT_STATE<=NEXT_STATE;
            END IF;
        END PROCESS;
        leds<=CURRENT_STATE;
```

```
PROCESS( CURRENT_STATE, reset, b)
BEGIN
    IF reset='1' THEN
        NEXT_STATE <= "00"; D<="00"; Q<="0000";
    ELSE
CASE CURRENT_STATE IS
WHEN "00" =>
    IF b(0)='1' THEN
        Q(0) <='1';
    ELSE
        Q(0) <='0';
    END IF;
    IF b(1)='1' THEN Q(1)<='1';END IF;
    IF b(2)='1' THEN Q(2)<='1';END IF;
    IF b(3)='1' THEN Q(3)<='1';END IF;
    D(0)<='0';
    IF D(0)='0' AND (Q(1)=1' OR Q(2)=1' OR Q(3)=1')THEN--subiendo
        NEXT_STATE <= "01"; D(1)<='1';
    END IF;
WHEN "01" =>
    Q(0)<=Q(0) OR b(1);
    Q(1) <=b(1);
    Q(3 downto 2)<=Q(3 downto 2) OR b(3 downto 2);
    IF D(0)='0' AND (Q(2)=1' OR Q(3)=1')THEN --subiendo
        NEXT_STATE <= "10"; D<="10";
    ELSIF D(1)='0' AND Q(0)=1' THEN--Bajando
        NEXT_STATE <= "00"; D<="01";
    ELSE
        D<="00";
    END IF;
WHEN "10" =>
    Q(1 downto 0)<=Q(1 downto 0) OR b(1 downto 0);
    Q(2) <=b(2);
    Q(3)<= Q(3) OR b(3);
    IF D(0)='0' AND Q(3)=1' THEN --subiendo
        NEXT_STATE <= "11"; D<="10";
    ELSIF D(1)='0' AND (Q(0)=1' OR Q(1)=1') THEN--Bajando
        NEXT_STATE <= "01"; D<="01";
    END IF;
```

$Q(0) \leq b(0);$   
 $Q(1) \leq Q(1) \text{ or } b(1)$



# DISEÑO AUTOMÁTICO

## Curso 2013-2014

-- Company: UCM  
-- Engineer: J.M. Montañana

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
library UNISIM;
use UNISIM.VComponents.all;

ENTITY semaforo IS
    Port ( clk,reset : in STD_LOGIC;
            b : in STD_LOGIC_VECTOR (3 downto 0);
            leds : out STD_LOGIC_VECTOR (1 downto 0));
END semaforo;
ARCHITECTURE Behavioral of semaforo IS
    SIGNAL CURRENT_STATE,NEXT_STATE :
        STD_LOGIC_VECTOR(1 downto 0);
    SIGNAL Q: STD_LOGIC_VECTOR(3 DOWNTO 0);
    SIGNAL D: STD_LOGIC_VECTOR(1 DOWNTO 0);--direcciones, arriba=10, abajo=01 , stop=00
BEGIN
    PROCESS( clk, reset)
        BEGIN
            IF reset='1' THEN
                CURRENT_STATE<= "00";
            ELSIF clk'event and clk='1' THEN
                CURRENT_STATE<=NEXT_STATE;
            END IF;
        END PROCESS;
        leds<=CURRENT_STATE;
    END;
```

```
PROCESS( CURRENT_STATE, reset, b)
BEGIN
    IF reset='1' THEN
        NEXT_STATE <= "00"; D<="00"; Q<="0000";
    ELSE
CASE CURRENT_STATE IS
WHEN "00" =>
    Q(0) <=b(0);
    Q(3 downto 1)<=Q(3 downto 1) OR b(3 downto 1);
    D(0)<='0';
    IF D(0)='0' AND (Q(1)='1' OR Q(2)='1' OR Q(3)='1')THEN--subiendo
        NEXT_STATE <= "01"; D(1)<='1';
    END IF;
WHEN "01" =>
    Q(0)<=Q(0) OR b(1);
    Q(1) <=b(1);
    Q(3 downto 2)<=Q(3 downto 2) OR b(3 downto 2);
    IF D(0)='0' AND (Q(2)='1' OR Q(3)='1')THEN --subiendo
        NEXT_STATE <= "10"; D<="10";
    ELSIF D(1)='0' AND Q(0)='1' THEN--Bajando
        NEXT_STATE <= "00"; D<="01";
    ELSE
        D<="00";
    END IF;
WHEN "10" =>
    Q(1 downto 0)<=Q(1 downto 0) OR b(1 downto 0);
    Q(2) <=b(2);
    Q(3)<= Q(3) OR b(3);
    IF D(0)='0' AND Q(3)='1' THEN --subiendo
        NEXT_STATE <= "11"; D<="10";
    ELSIF D(1)='0' AND (Q(0)='1' OR Q(1)='1') THEN--Bajando
        NEXT_STATE <= "01"; D<="01";
    END IF;
WHEN others =>
    Q(3 downto 1)<=Q(3 downto 1) OR b(3 downto 1);
    Q(3) <=b(3);
    IF D(1)='0' AND (Q(0)='1' OR Q(1)='1' OR Q(2)='1') THEN-bajando
        NEXT_STATE <= "10"; D<="01";
    END IF;
```



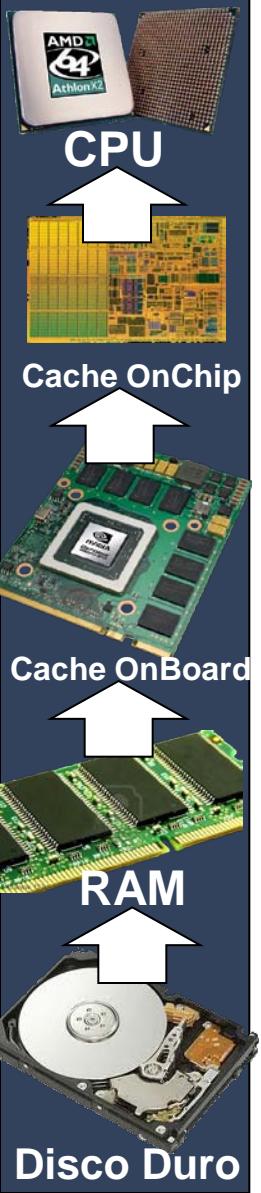
# DISEÑO AUTOMÁTICO

## Curso 2013-2014

-- Company: UCM  
-- Engineer: J.M. Montañana

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
library UNISIM;
use UNISIM.VComponents.all;
ENTITY semaforo IS
    Port ( clk,reset : in STD_LOGIC;
            b : in STD_LOGIC_VECTOR (3 downto 0);
            leds : out STD_LOGIC_VECTOR (1 downto 0));
END semaforo;
ARCHITECTURE Behavioral of semaforo IS
    SIGNAL CURRENT_STATE,NEXT_STATE :
        STD_LOGIC_VECTOR(1 downto 0);
    SIGNAL Q: STD_LOGIC_VECTOR(3 DOWNTO 0);
    SIGNAL D: STD_LOGIC_VECTOR(1 DOWNTO 0);--direcciones, arriba=10, abajo=01 , stop=00
BEGIN
    PROCESS( clk, reset)
        BEGIN
            IF reset='1' THEN
                CURRENT_STATE<= "00";
            ELSIF clk'event and clk='1' THEN
                CURRENT_STATE<=NEXT_STATE;
            END IF;
        END PROCESS;
        leds<=CURRENT_STATE;
```

```
PROCESS( CURRENT_STATE, reset, b)
BEGIN
    IF reset='1' THEN
        NEXT_STATE <= "00" ; D<="00"; Q<="0000";
    ELSE
CASE CURRENT_STATE IS
WHEN "00" =>
    Q(0) <=b(0);
    Q(3 downto 1)<=Q(3 downto 1) OR b(3 downto 1);
    IF Q(1)='1' OR Q(2)='1' OR Q(3)='1' THEN--subiendo
        NEXT_STATE <= "01"; D<="10";
    END IF;
WHEN "01" =>
    Q(0) <= Q(0) OR b(1);
    Q(1) <= b(1);
    Q(3 downto 2)<=Q(3 downto 2) OR b(3 downto 2);
    IF D(0)='0' AND (Q(2)='1' OR Q(3)='1')THEN --subiendo
        NEXT_STATE <= "10"; D<="10";
    ELSIF D(1)='0' AND Q(0)='1' THEN--Bajando
        NEXT_STATE <= "00"; D<="01";
    ELSE
        D<="00";
    END IF;
WHEN "10" =>
    Q(1 downto 0)<=Q(1 downto 0) OR b(1 downto 0);
    Q(2) <= b(2);
    Q(3) <= Q(3) OR b(3);
    IF D(0)='0' AND Q(3)='1' THEN --subiendo
        NEXT_STATE <= "11"; D<="10";
    ELSIF D(1)='0' AND (Q(0)='1' OR Q(1)='1') THEN--Bajando
        NEXT_STATE <= "01"; D<="01";
    END IF;
WHEN others =>
    Q(3 downto 1)<=Q(3 downto 1) OR b(3 downto 1);
    Q(3) <= b(3);
IF D(1)='0' AND (Q(0)='1' OR Q(1)='1' OR Q(2)='1') THEN-bajando
    NEXT_STATE <= "10"; D<="01";
    END IF;
END CASE;      END IF;  END PROCESS;
```



# Práctica 2

## CRONÓMETRO

**Profesor teoría y laboratorio  
José Miguel Montaña Aliaga.**

e-mail: [jmontanana@fdi.ucm.es](mailto:jmontanana@fdi.ucm.es)

**Tutorías: 1er cuatrimestre Miércoles : 10:00-13:00**

**2ndo cuatrimestre Miércoles 14:00-17:00**

**Fac Informática dpch 421, Fac. Física dpcho 225**

**Apuntes: Campus Virtual**

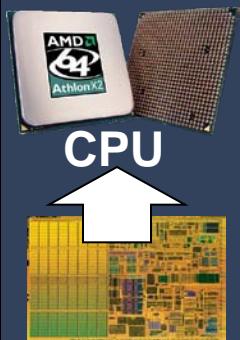
**D E P A R T A M E N T O D E  
A R Q U I T E C T U R A D E C O M P U T A D O R E S  
Y A U T O M Á T I C A**

Dpto. de Arquitectura de Computadores y Automática

Facultad de Informática.

Universidad Complutense de Madrid.

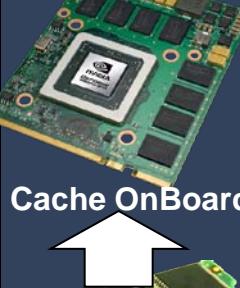




## CPU



## Cache OnChip



## RAM



# Disco Duro

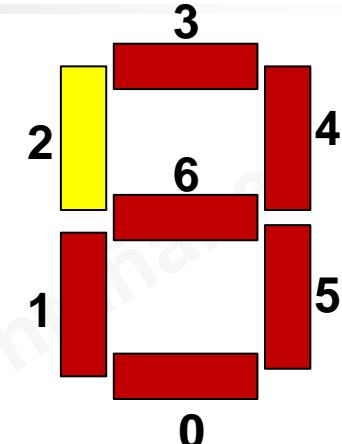
LTC (sesiones de clase)			
2 Octubre	0, Introducción	11-Noviembre	7. Pendiente de definir
4 Octubre	1, Diseño de sistemas	13-Noviembre	8.- Memorias estáticas y dinámicas
9 Octubre	2, Temporización y maquinas de estados (intro VHDL)	20-Noviembre	9.- Gráficos VGA
16 Octubre	3, Especificación nivel lógico, (repaso práctica 1)	25-Noviembre	9. Arquitectura de procesador
23 Octubre	4. Entrada salida E/S, ejemplo ps/2	27-Noviembre	10. Circuito Asíncrono (FPGA)
30 Octubre	5. Técnicas de diseño	4-Diciembre	Plan proyectos fin de curso
6-Noviembre	6. Modulación de sonido	18-Diciembre	X- examen evaluación continua teoria 2013-14

# DISEÑO AUTOMÁTICO DE SISTEMAS

## Curso 2013-2014

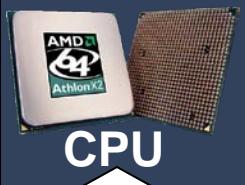
Práctica 2

### CONVERSOR A BCD

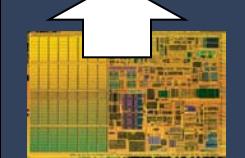


```

library ieee;
use ieee.std_logic_1164.all;
entity bcd_decoder is
  port (code : in std_logic_vector(3 downto 0);
        leds : out std_logic_vector(6 downto 0));
end bcd_decoder;
architecture prototype of bcd_decoder is
BEGIN
  PROCESS (code)
  BEGIN
    CASE code IS
      when "0000" => leds <= "0111111";
      when "0001" => leds <= "0110000";
      when "0010" => leds <= "1011011";
      when "0011" => leds <= "1111000";
      when "0100" => leds <= "1110100";
      when "0101" => leds <= "1101101";
      when "0110" => leds <= "1101111";
      when "0111" => leds <= "0111000";
      when "1000" => leds <= "1111111";
      when "1001" => leds <= "1111100";
      when others => leds <= null ; --the default case.
    END CASE;
  END process;
END prototype;
  
```



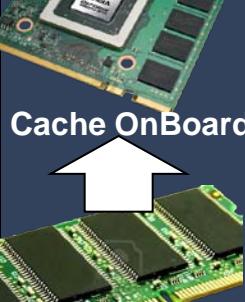
CPU



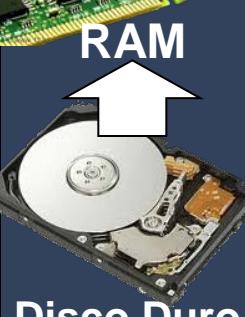
Cache OnChip



Cache OnBoard



RAM

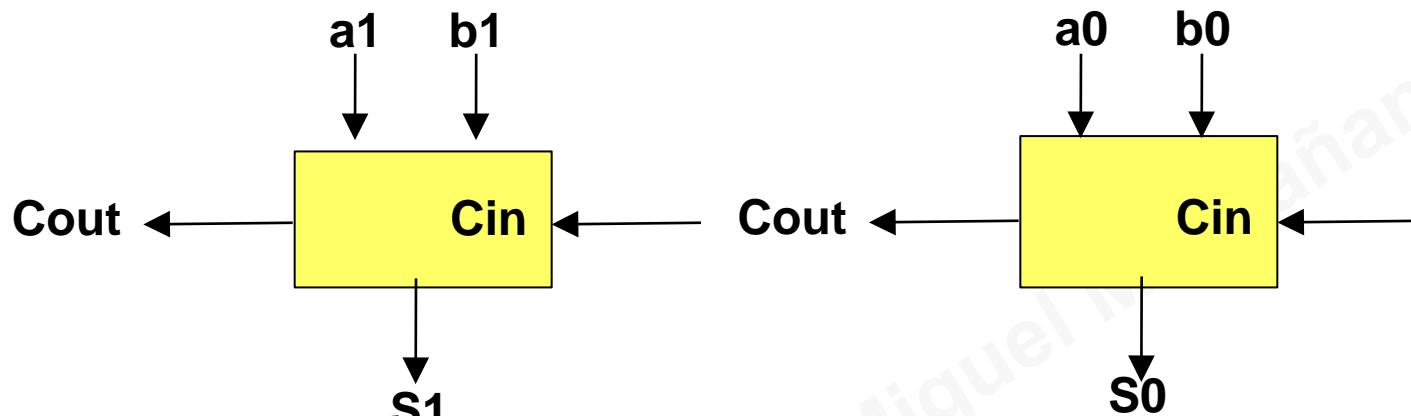


Disco Duro

inputs				decimal value	outputs							
c <sub>0</sub>	c <sub>1</sub>	c <sub>2</sub>	c <sub>3</sub>		s <sub>0</sub>	s <sub>1</sub>	s <sub>2</sub>	s <sub>3</sub>	s <sub>4</sub>	s <sub>5</sub>	s <sub>6</sub>	
0	0	0	0	0	1	1	1	1	1	1	1	0
0	0	0	1	1	0	0	0	0	1	1	1	0
0	0	1	0	2	1	1	0	1	1	0	1	1
0	0	1	1	3	0	0	0	1	1	1	1	1
0	1	0	0	4	0	0	1	0	1	1	1	1
0	1	0	1	5	1	0	1	1	0	1	1	1
0	1	1	0	6	1	1	1	1	0	1	1	1
0	1	1	1	7	0	0	0	1	1	1	1	0
1	0	0	0	8	1	1	1	1	1	1	1	1
1	0	0	1	9	0	0	1	1	1	1	1	1

Curso 2013-2014

a) Diseñar una red iterativa combinacional que realice la SUMA de 2 números de 2 bits en binario

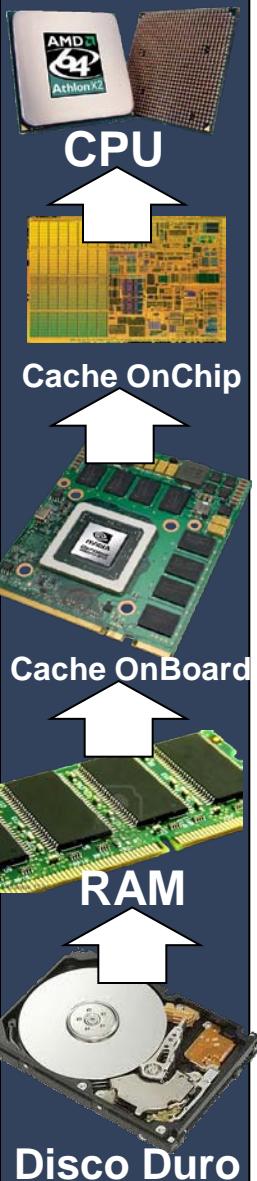


Diseñamos en primer lugar una celda:

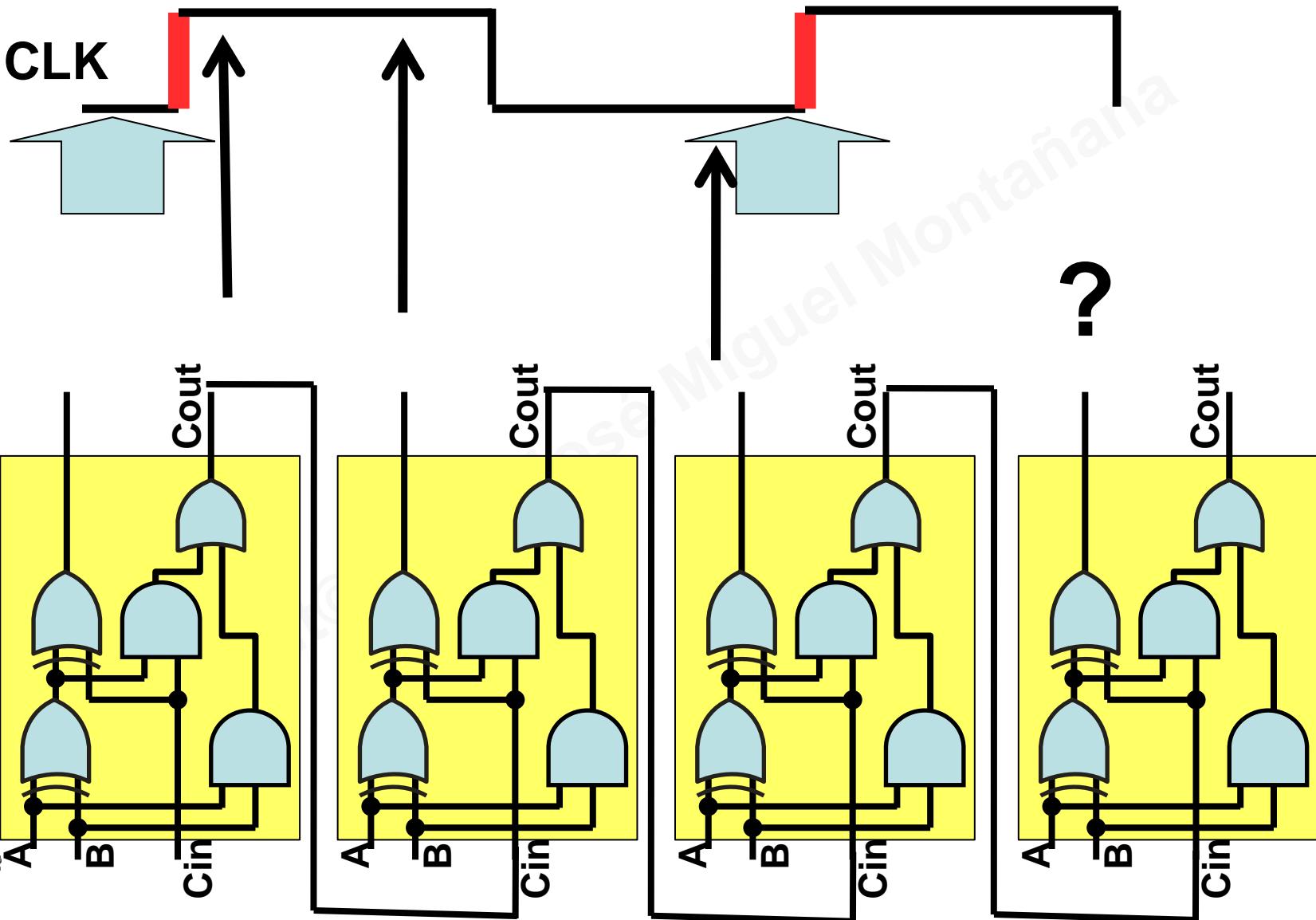
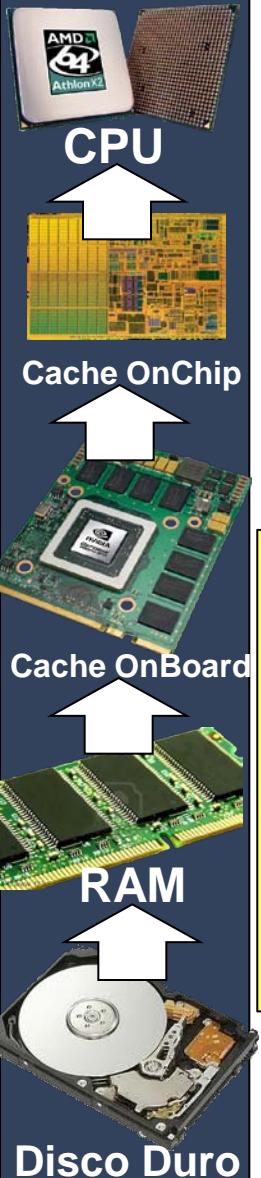
→ A continuación lo vemos en la PIZARRA:

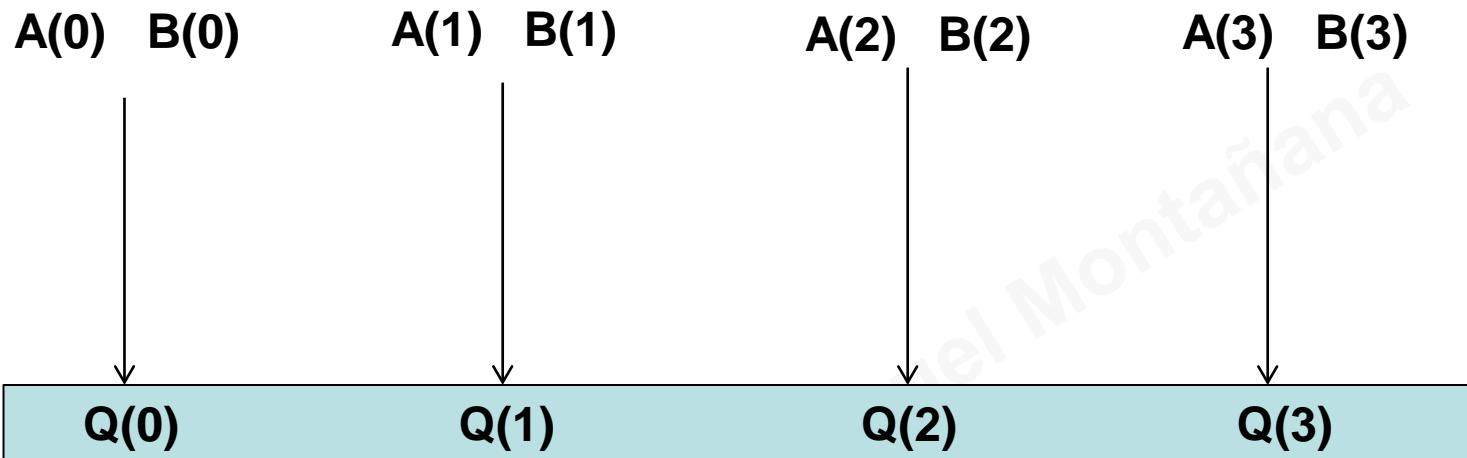
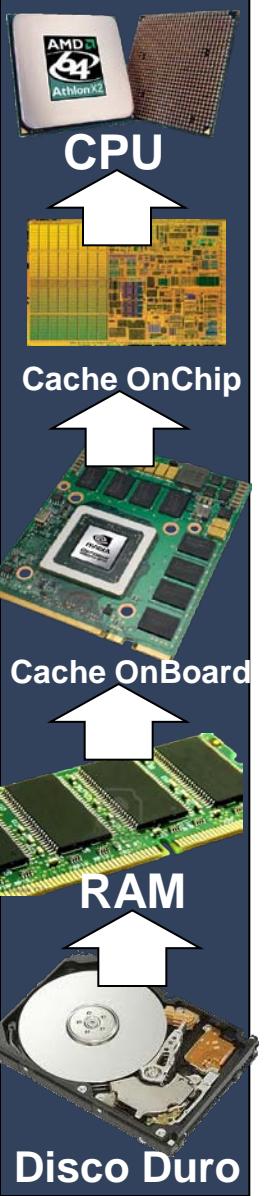
XOR de  
tres entradas:  
 $A \oplus B \oplus C$

Entrada A	Entrada B	Entrada C	Salida
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

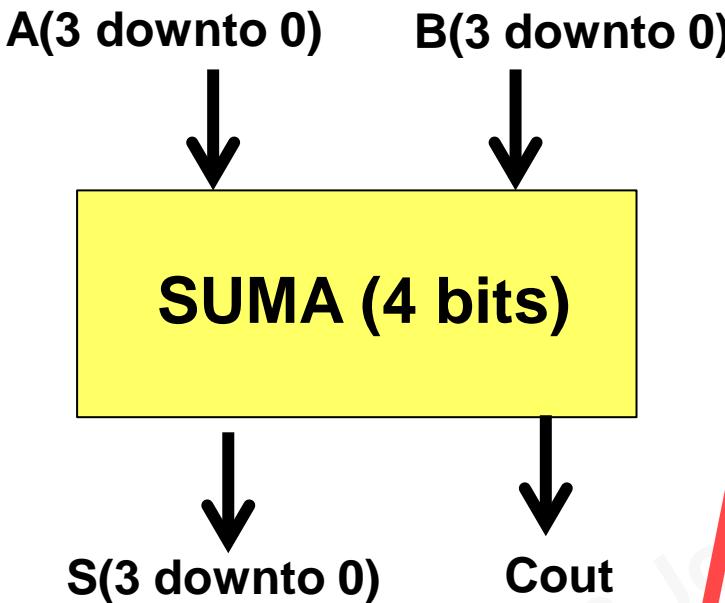
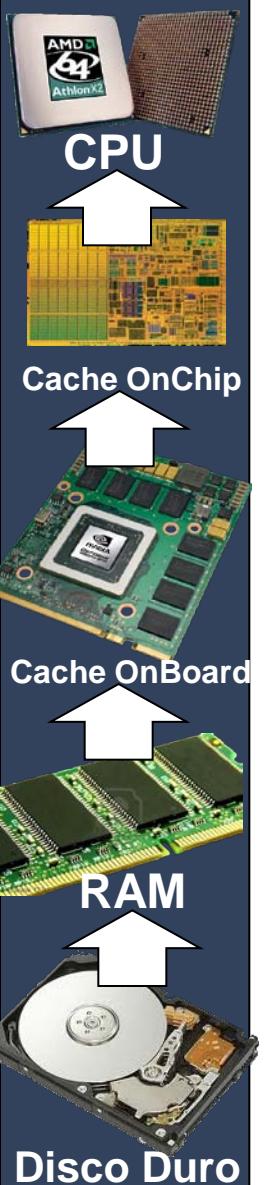


## SUMADOR DE 4 BITS





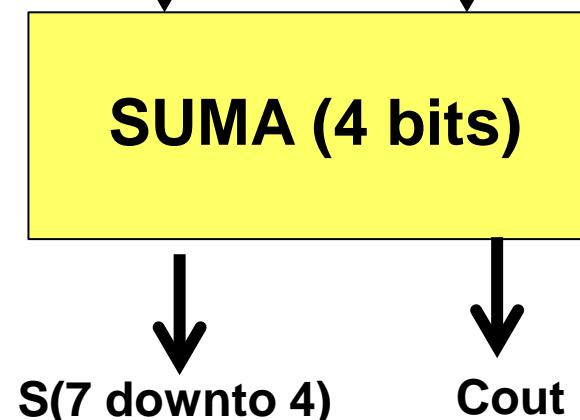
```
IF Q(3 downto 0)= "1101" THEN
    Z <= "00";
ELSE
    Z <= "11";
END IF;
```

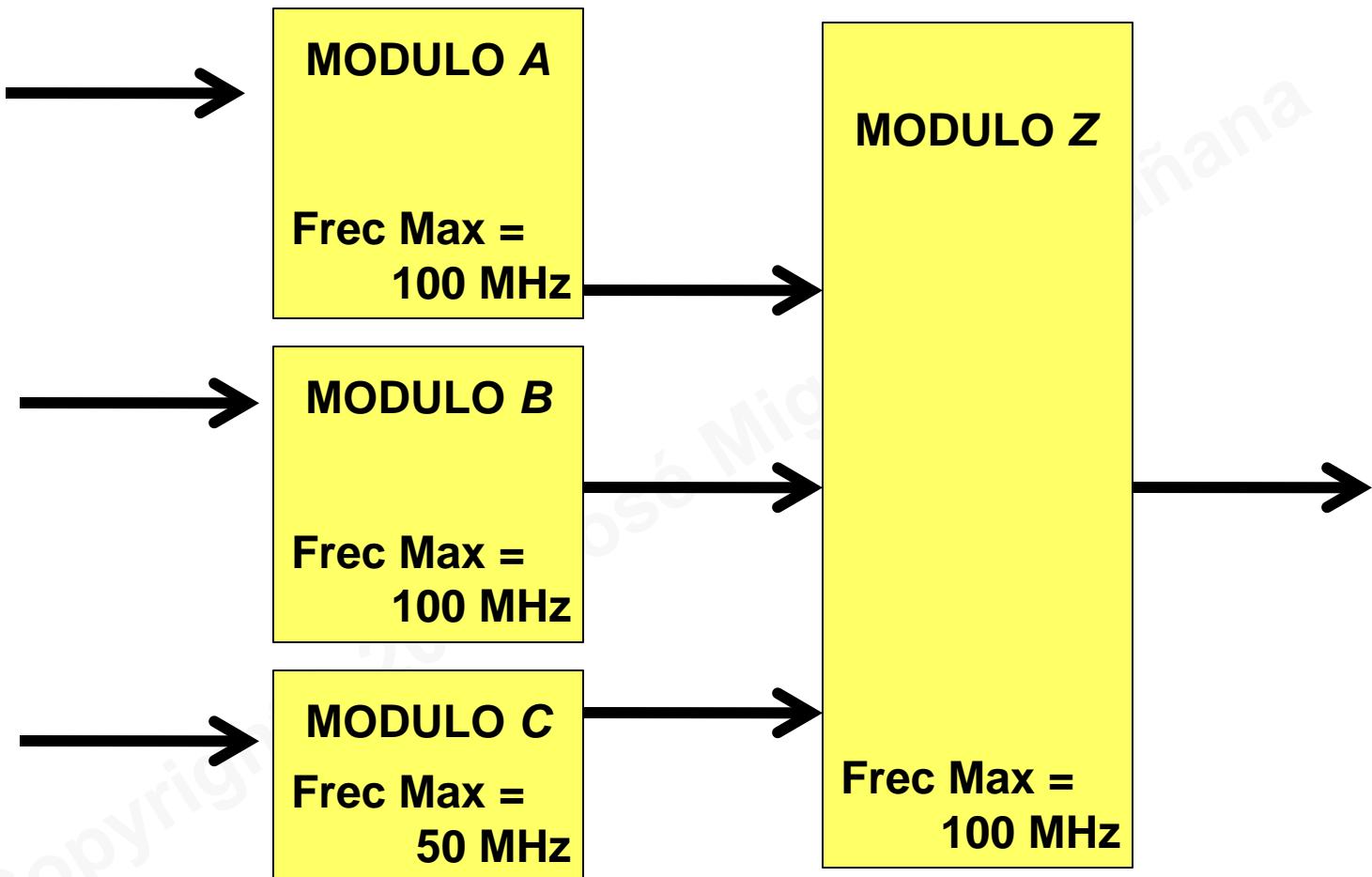


1er ciclo  
de reloj

2ndo ciclo  
de reloj

A(7 downto 4)      B(7 downto 4)

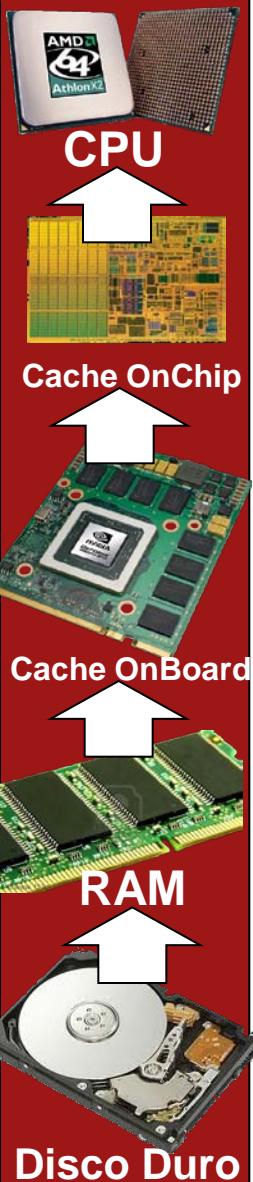




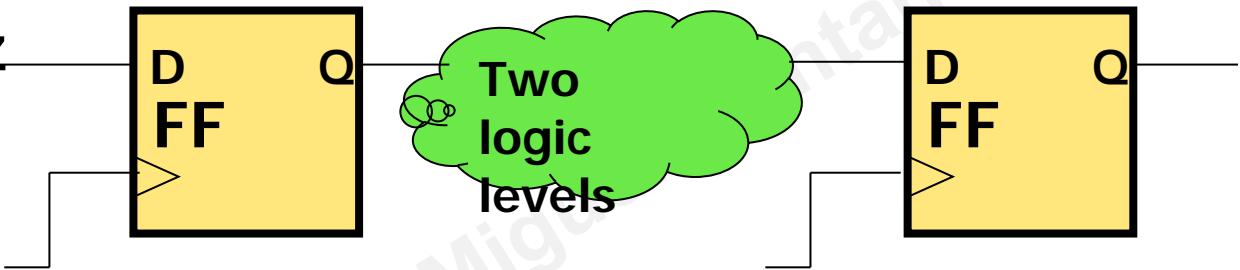
EL MODULO C LIMITA LA FRECUENCIA MAXIMA DEL SISTEMA

# Timing constraints

## Pipelining

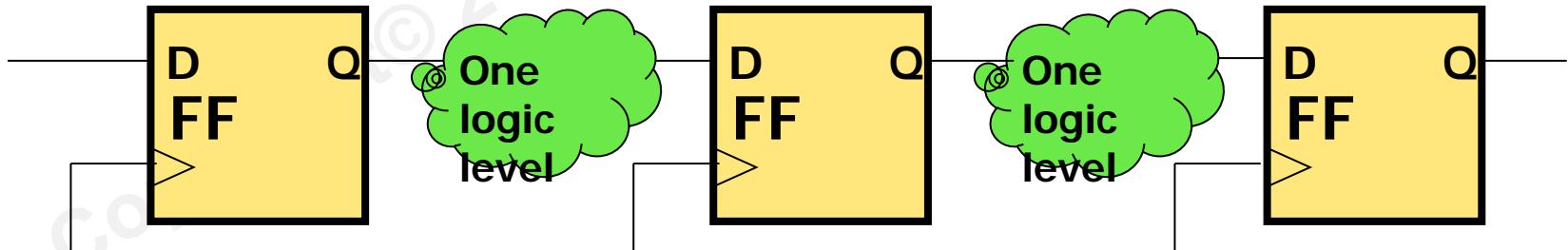


$$f_{\max} = n \text{ MHz}$$



DATA ARRIVES AT N-MHz

$$f_{\max} \approx 2n \text{ MHz}$$

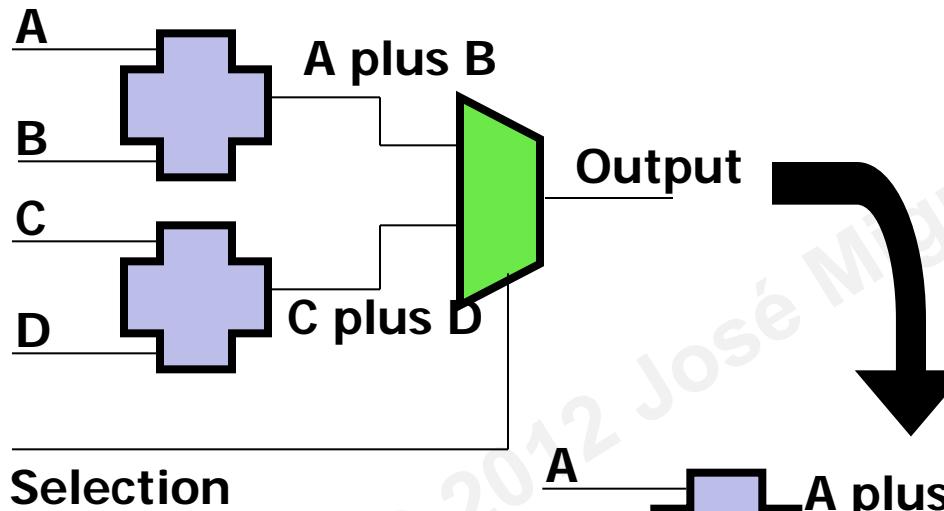
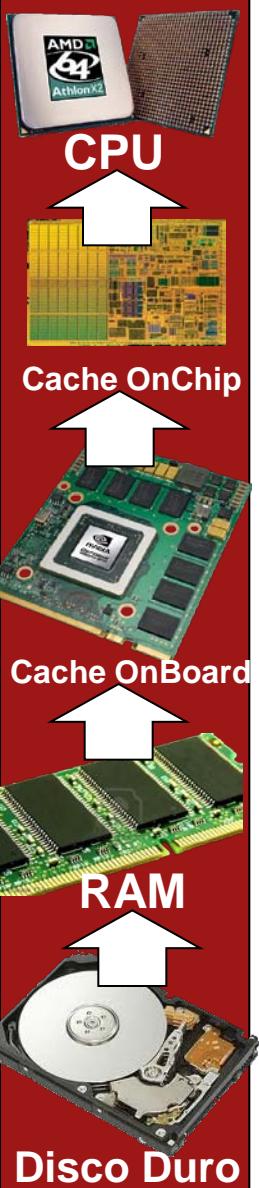


DATA ARRIVES AT 2N-MHz

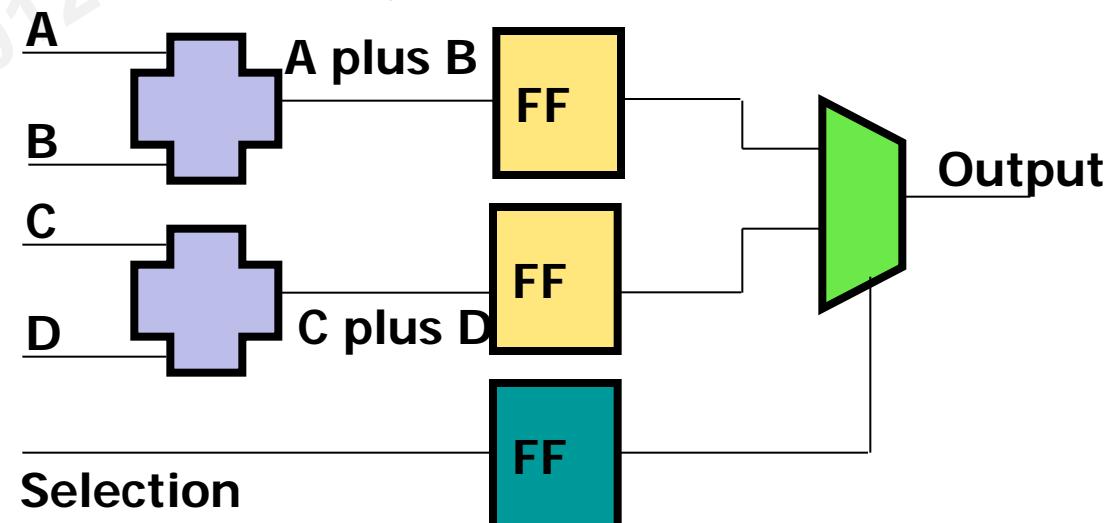
RESULT LEAVES AT N-MHz  
1 CLOCK CYCLE DELAYED

RESULT LEAVES AT 2N-MHz  
2 CLOCK CYCLES DELAYED

# Timing constraints Pipelining: Example

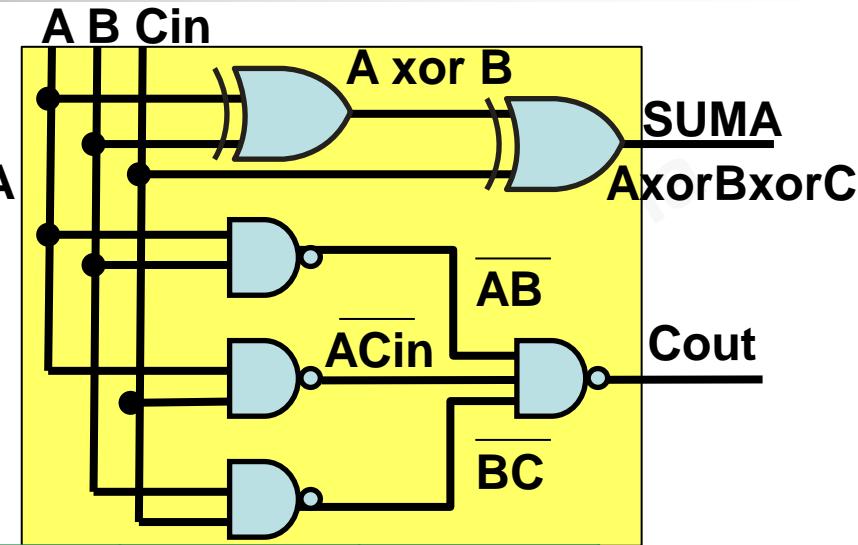
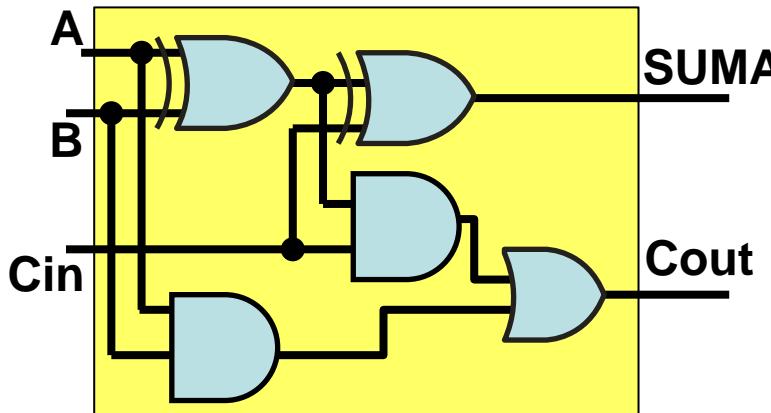


Selection

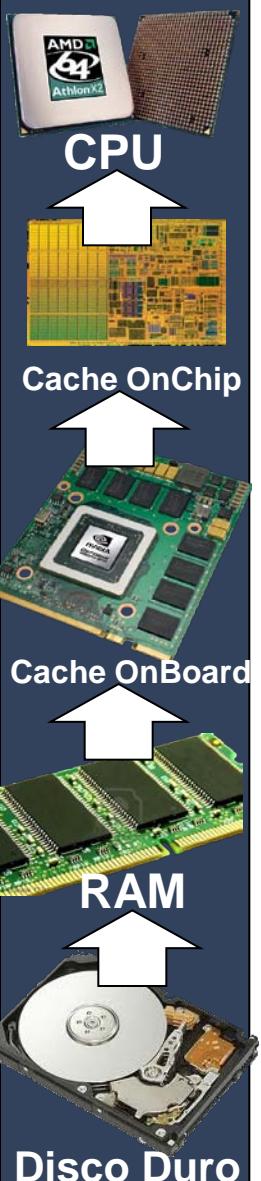


Selection

### APARTADO A:



A	B	Cin	Cout	Suma
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

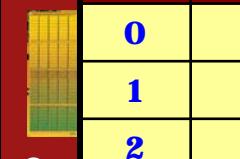


# Encoding Finite State Machines (FSM)

## Timing constraints



CPU



Cach



Cach

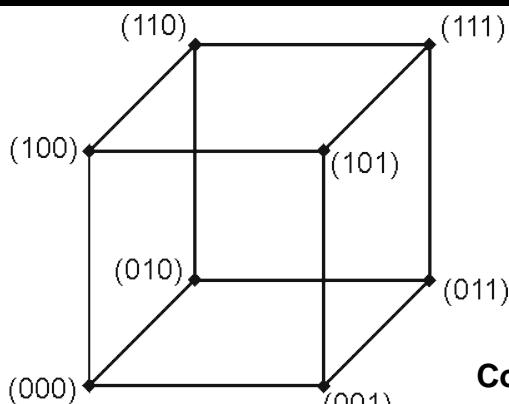


RAM



Disco Duro

E.A.	One-Hot	Compact	Sequential	Gray	Johnson	Speed
0	0000001	000	000	000	0000	000000
1	0000010	001	001	001	0001	010010
2	0000100	011	010	011	0011	001010
3	0001000	111	011	010	0111	000110
4	0010000	010	100	110	1111	000001
5	0100000	110	101	111	1110	100001
6	1000000	100	110	101	1100	100000
	7 FFs	3 FFs	3 FFs	3 FFs	4 FFs	6 FFs
	<b>187.161</b> <b>MHz</b>	<b>230.150</b> <b>MHz</b>	<b>181.653</b> <b>MHz</b>	<b>223.664</b> <b>MHz</b>	<b>181.653</b> <b>MHz</b>	<b>272.331</b> <b>MHz</b>



**ENTITY suma8 is**

```
Port ( a : in std_logic_vector(7 downto 0);
      b : in std_logic_vector(7 downto 0);
      cin : in std_logic;
      s : out std_logic_vector(7 downto 0);
      cout : out std_logic);
```

end suma8;

**ARCHITECTURE Behavioral of suma8 is**

**COMPONENT suma1 is**

```
Port ( a : in std_logic;
      b : in std_logic;
      cin : in std_logic;
      s : out std_logic;
      cout : out std_logic);
```

end COMPONENT;

**SIGNAL temp: std\_logic\_vector(6 downto 0);**

**begin**

```
mod1: suma1( a(0), b(0), cin      , s(0), temp(0));
mod2: suma1( a(1), b(1), temp(0), s(1) , temp(1));
mod3: suma1( a(2), b(2), temp(1), s(2) , temp(2));
mod4: suma1( a(3), b(3), temp(2), s(3) , temp(3));
mod5: suma1( a(4), b(4), temp(3), s(4) , temp(4));
mod6: suma1( a(5), b(5), temp(4), s(5) , temp(5));
mod7: suma1( a(6), b(6), temp(5), s(6) , temp(6));
mod8: suma1( a(7), b(7), temp(6), s(7) ,cout);
end Behavioral;
```

**ENTITY suma1 is**

```
Port ( a : in std_logic;
      b : in std_logic;
      cin : in std_logic;
      s : out std_logic;
      cout : out std_logic);
```

end suma1;

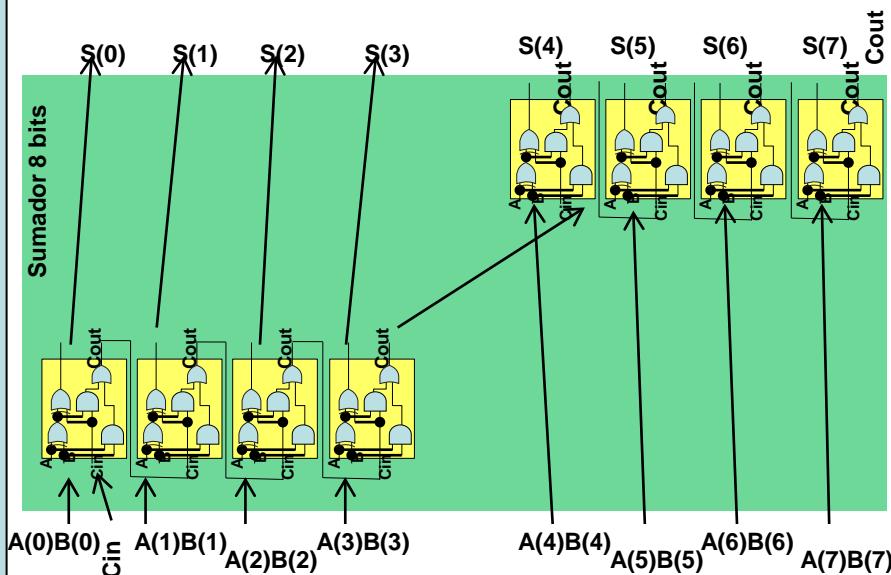
**ARCHITECTURE Behavioral of suma1 is**

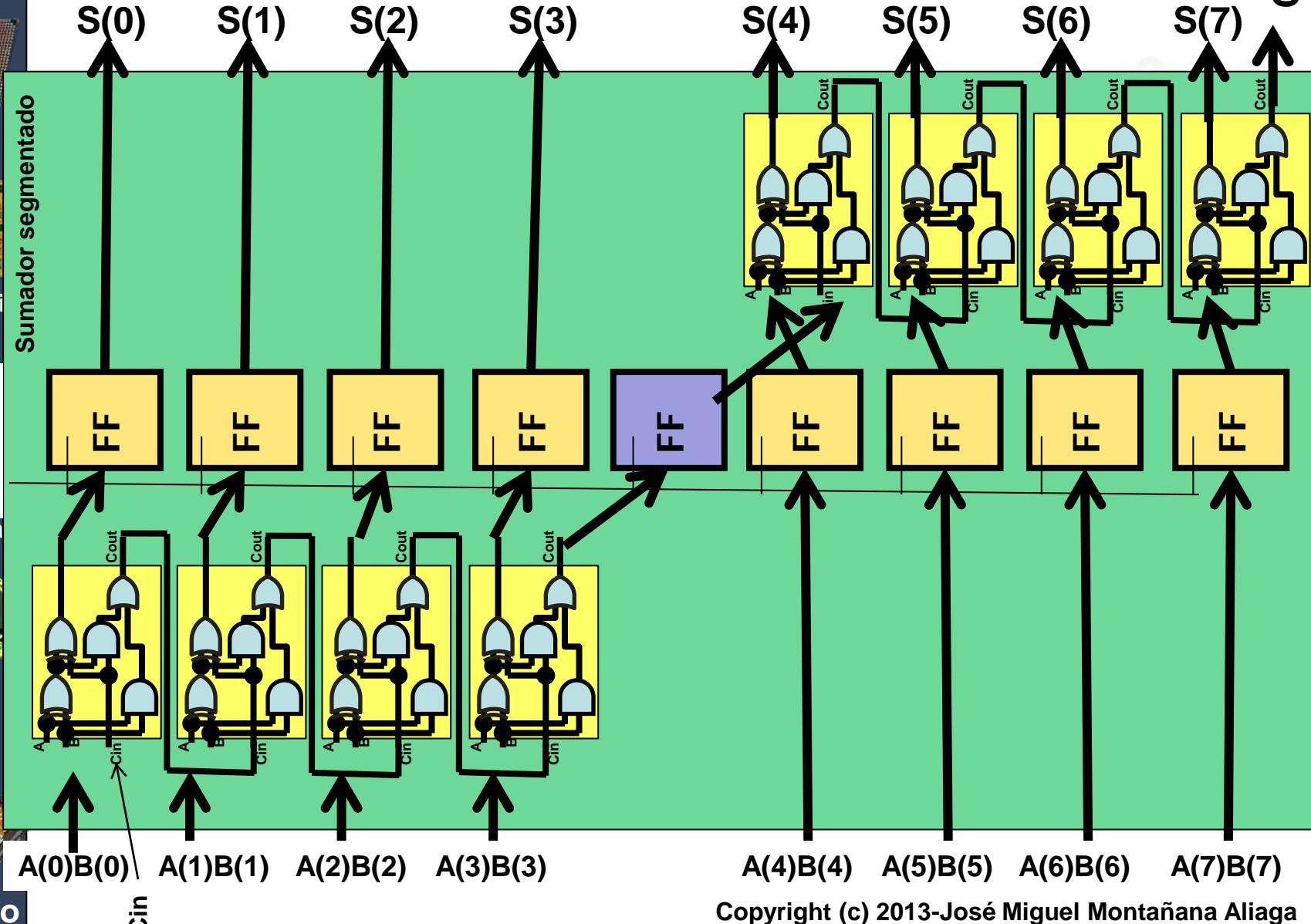
**begin**

$s<=(a \text{ xor } (b \text{ xor } \text{cin}))$ ;

$\text{cout}<=(\text{b and } \text{cin}) \text{ or } (\text{a and } \text{cin}) \text{ or } (\text{a and } \text{b})$ ;

**end Behavioral;**





**ENTITY** suma8\_seg **is**

Port ( clk: in std\_logic;

a: in std\_logic\_vector(7 downto 0);

b: in std\_logic\_vector(7 downto 0);

cin: in std\_logic;

s: out std\_logic\_vector(7 downto 0);

cout: out std\_logic);

end suma8\_seg ;

**ARCHITECTURE** Behavioral

of suma8\_seg **is**

**COMPONENT** suma1 **is**

Port ( a : in std\_logic;

b : in std\_logic;

cin : in std\_logic;

s : out std\_logic;

cout : out std\_logic);

end COMPONENT;

**SIGNAL** temp:std\_logic\_vector(3 downto 0);

**SIGNAL** ff\_s:std\_logic\_vector(3 downto 0);

**SIGNAL** ff\_a:std\_logic\_vector(7 downto 4);

**SIGNAL** ff\_b:std\_logic\_vector(7 downto 4);

**SIGNAL** ff\_c:std\_logic;

**SIGNAL** tempB: std\_logic\_vector(2 downto 0);

**BEGIN**

mod1a: suma1 port map(a(0),b(0),cin , ff\_s(0), temp(0));

mod2a: suma1 port map(a(1),b(1),temp(0),ff\_s(1),temp(1));

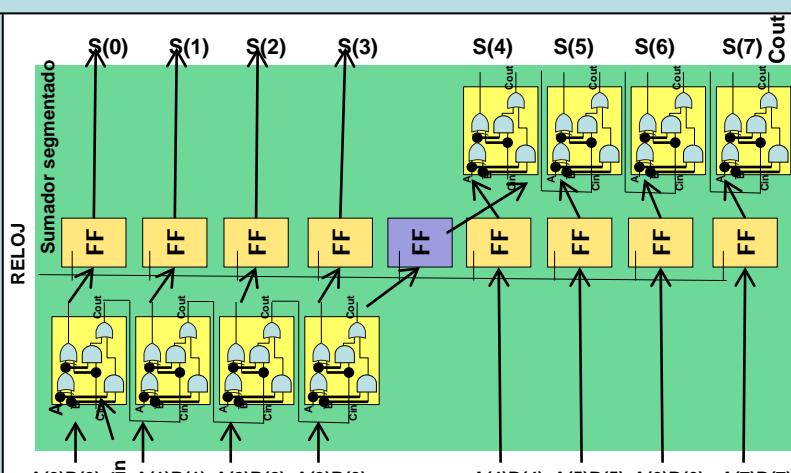
mod3a: suma1 port map(a(2),b(2),temp(1),ff\_s(2),temp(2));

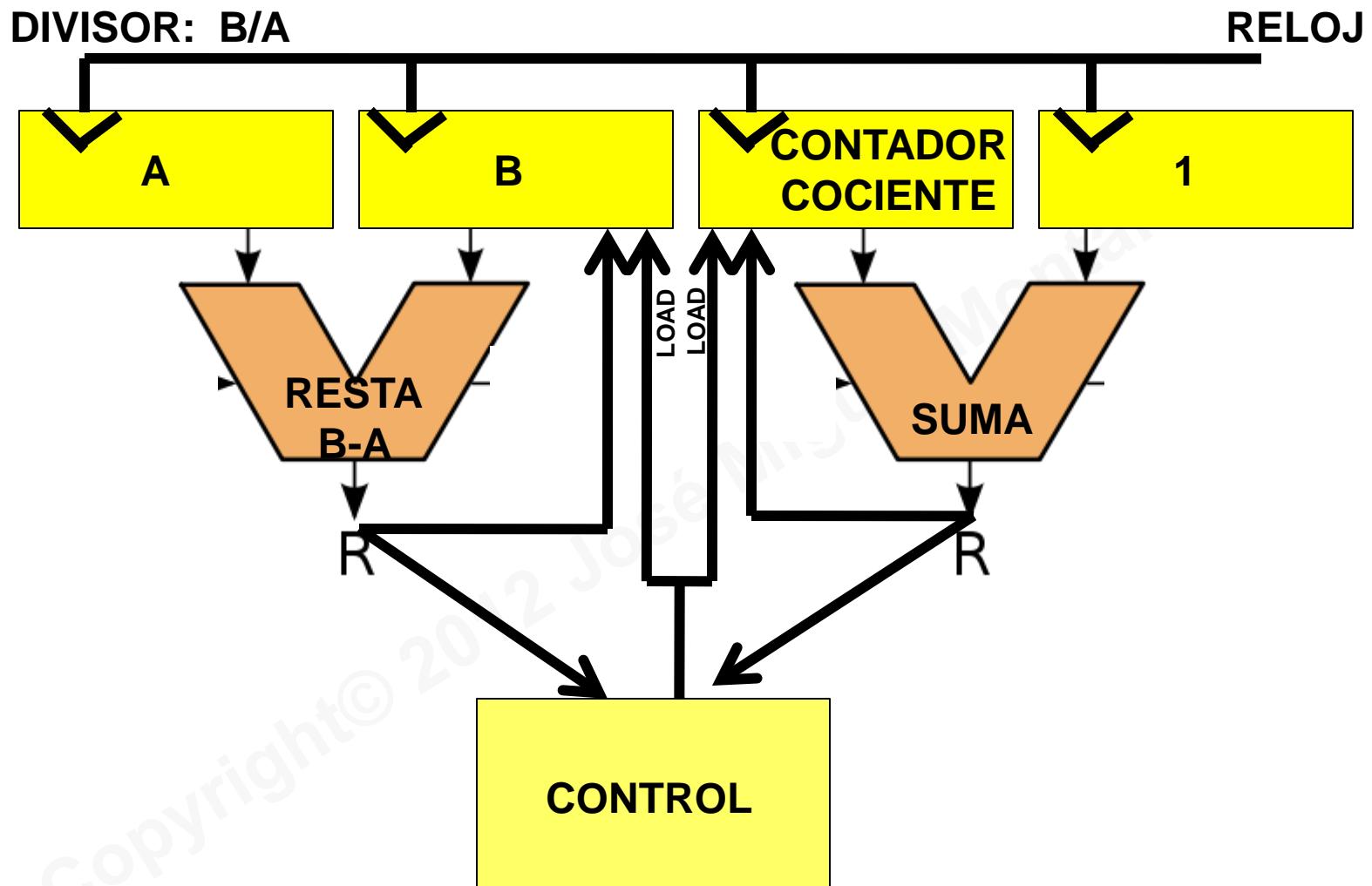
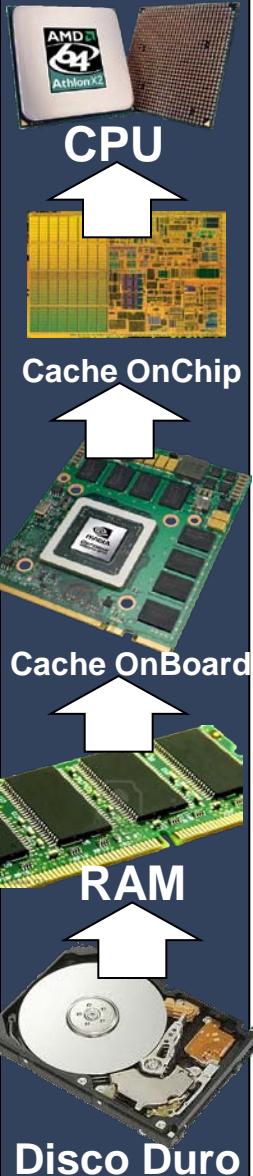
mod4a: suma1 port map(a(3),b(3),temp(2),ff\_s(3),temp(3));

```

mod1b:suma1 port map(ff_a(4),ff_b(4),ff_c,      s(4),tempB(0));
mod2b:suma1 port map(ff_a(5),ff_b(5),tempB(0),s(5),tempB(1));
mod3b:suma1 port map(ff_a(6),ff_b(6),tempB(1),s(6),tempB(2));
mod8b:suma1 port map(ff_a(7),ff_b(7),tempB(2),s(7),cout);
PROCESS (clk)
BEGIN
If clk'event and clk='1' THEN
    s(3 downto 0)<= ff_s(3 downto 0);
    ff_a(7 downto 4)<= a(7 downto 4);
    ff_b(7 downto 4)<= b(7 downto 4);
    ff_c<=temp(3);
END IF;
END PROCESS;
end Behavioral;

```



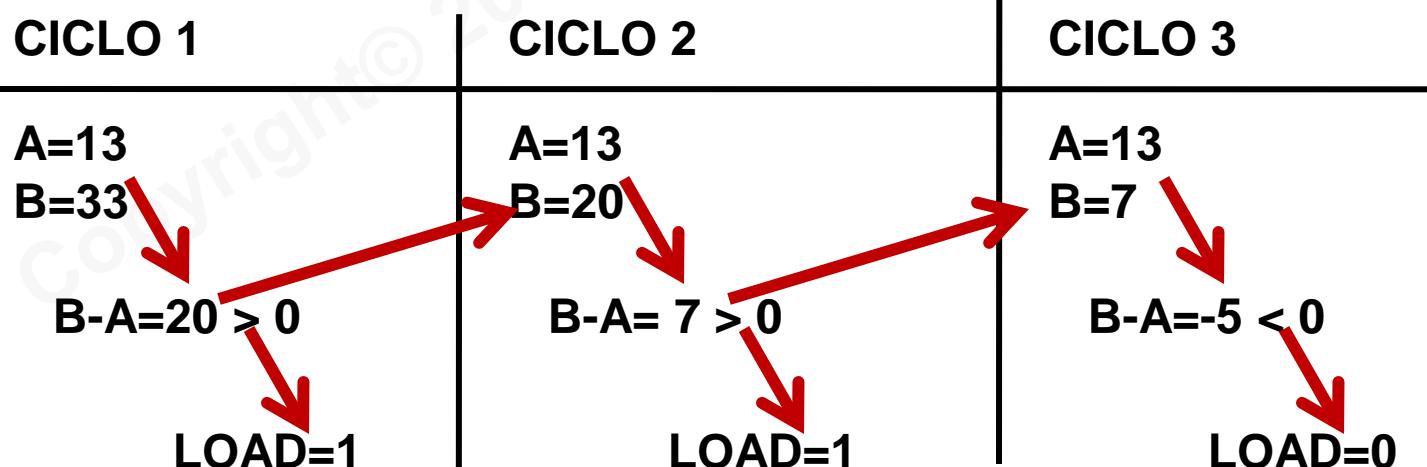
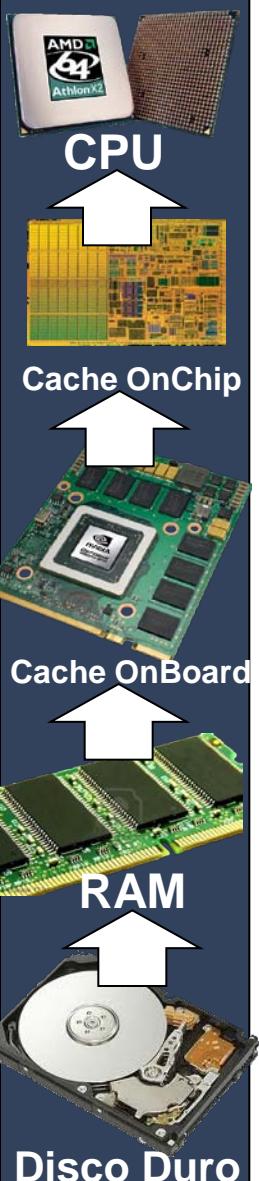
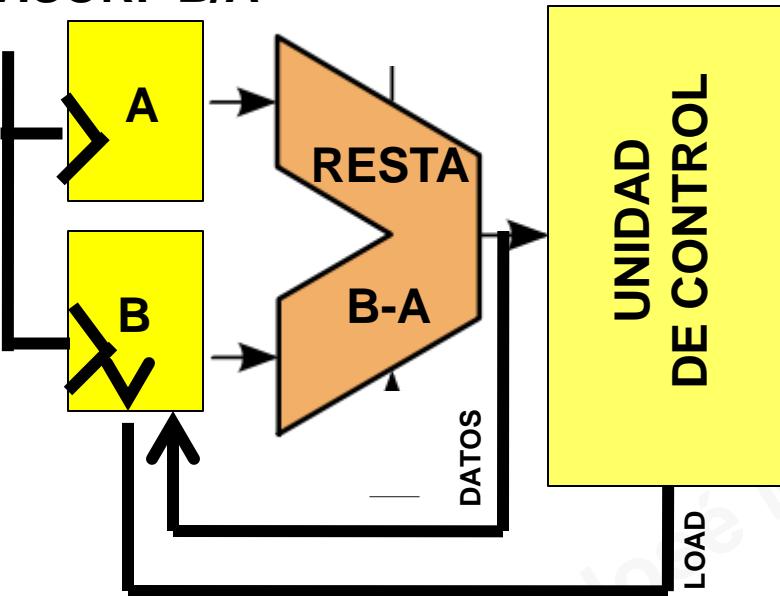


Cuando  $B-A$  sea negativo no se actualizará B

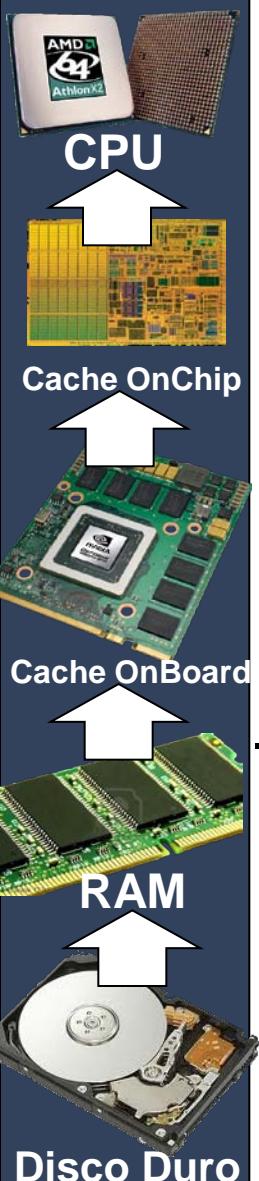
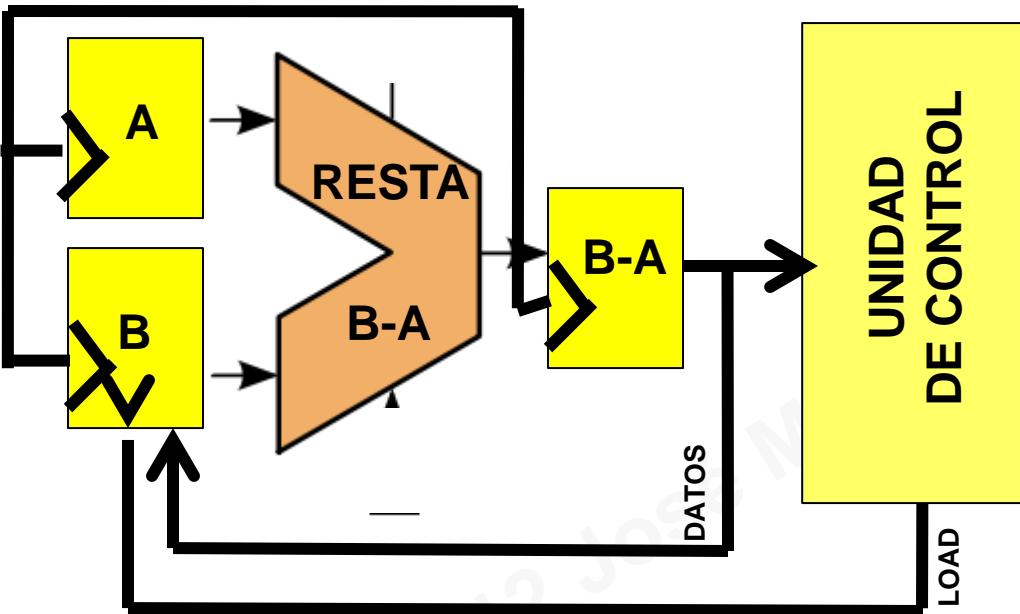
En ese momento B contendrá el resto de la división

Debido a que la frecuencia de reloj es muy alta, no da tiempo a que la unidad de control de la salida correcta a partir del resultado de las ALUs

### DIVISOR: B/A

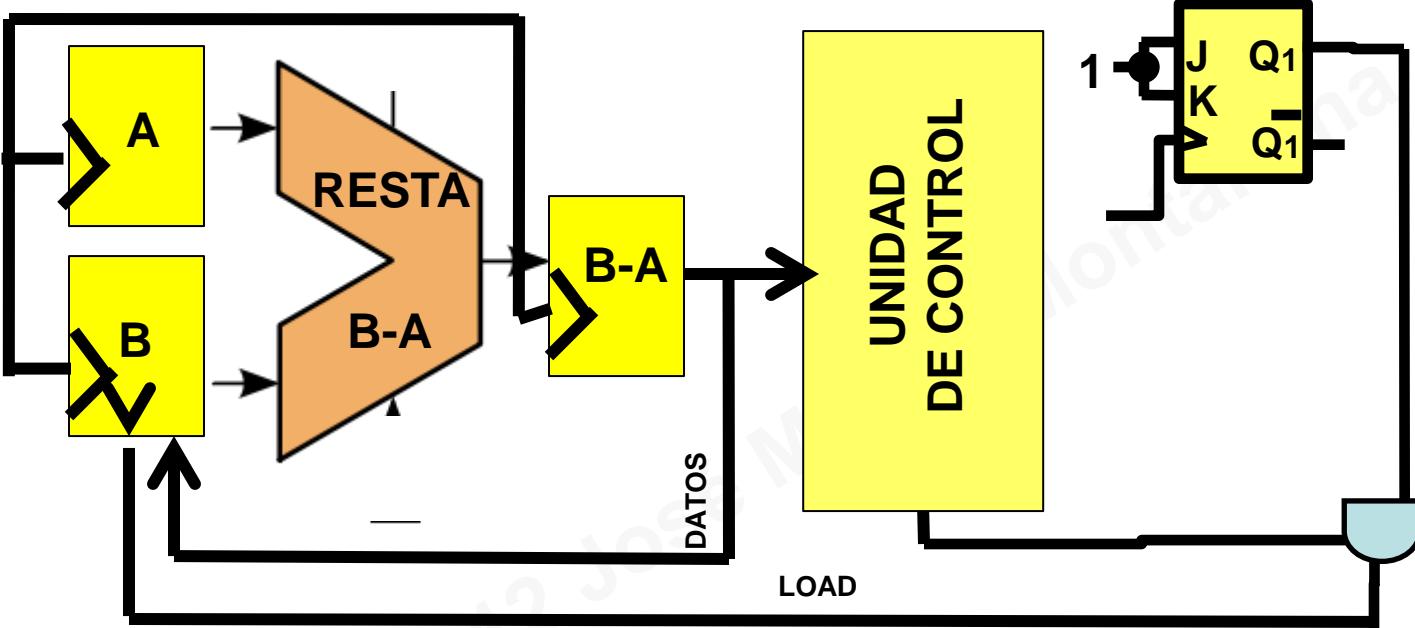
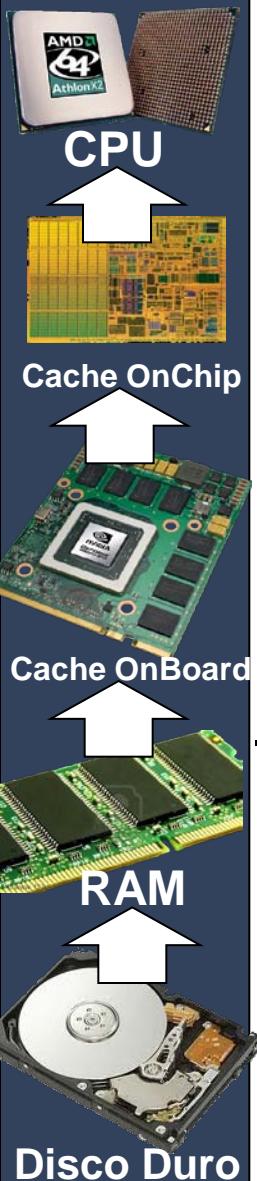


#### DIVISOR: B/A

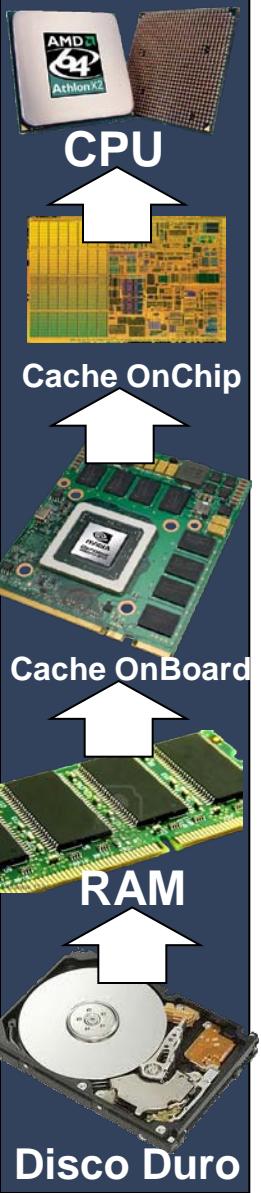


CICLO 1	CICLO 2	CICLO 3	CICLO 4	CICLO 5	CICLO 6
A=13 B=33	A=13 B=33	A=13 B=20	A=13 B=20	A=13 B=7	A=13 B=7
B-A=??	B-A= 20 > 0	B-A= 20 > 0	B-A= 7 > 0	B-A= 7 > 0	B-A = -5 < 0
LOAD=0	LOAD=1	LOAD=0	LOAD=1	LOAD=0	LOAD=1

#### DIVISOR: B/A



CICLO 1	CICLO 2	CICLO 3	CICLO 4	CICLO 5	CICLO 6
A=13 B=33	A=13 B=33	A=13 B=20	A=13 B=20	A=13 B=7	A=13 B=7
B-A=??	B-A= 20 > 0	B-A= 20 > 0	B-A= 7 > 0	B-A= 7 > 0	B-A = -5 < 0
LOAD=0	LOAD=1	LOAD=0	LOAD=1	LOAD=0	LOAD=1



- 1.- Interés por el ejercicio?  
**(Attitude)**
- 2.- Habilidad para comunicar sus ideas o diseño?  
**(Communication Skills)**
- 3.- Tiene la creatividad para encontrar una solución al problema  
**(Problem-Solving Skills)**
- 4.- Uso eficiente del tiempo, priorización de las partes de la tarea más importante  
**(Time Management Abilities)**
- 5.- Trabaja bien bajo presión  
**(Working Well Under Pressure)**



# Práctica 3

## Modulos E/S, teclado y ratón PS/2

**Profesor teoría y laboratorio**  
**José Miguel Montaña Aliaga.**

e-mail: [jmontanana@fdi.ucm.es](mailto:jmontanana@fdi.ucm.es)

**Tutorías: 1er cuatrimestre Miércoles : 10:00-13:00**

**2ndo cuatrimestre Miércoles 14:00-17:00**

**Fac Informática dpch 421, Fac. Física dpcho 225**

**Apuntes: Campus Virtual**

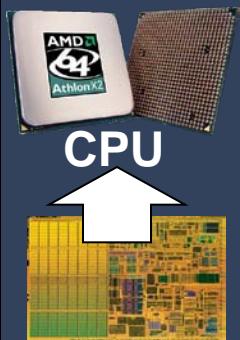
**D E P A R T A M E N T O D E  
A R Q U I T E C T U R A D E C O M P U T A D O R E S  
Y A U T O M Á T I C A**

Dpto. de Arquitectura de Computadores y Automática

Facultad de Informática.

Universidad Complutense de Madrid.

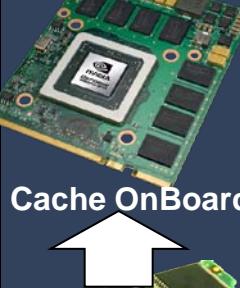




CPU



## Cache OnChip



## RAM



# Disco Duro

LTC (sesiones de clase)			
2 Octubre	0, Introducción	11-Noviembre	7. Pendiente de definir
4 Octubre	1, Diseño de sistemas	13-Noviembre	8.- Memorias estáticas y dinámicas
9 Octubre	2, Temporización y maquinas de estados (intro VHDL)	20-Noviembre	9.- Gráficos VGA
16 Octubre	3, Especificación nivel lógico, (repaso práctica 1)	25-Noviembre	9. Arquitectura de procesador
23 Octubre	4. Entrada salida E/S, ejemplo ps/2	27-Noviembre	10. Circuito Asíncrono (FPGA)
30 Octubre	5. Técnicas de diseño	4-Diciembre	Plan proyectos fin de curso
6-Noviembre	6. Modulación de sonido	18-Diciembre	X- examen evaluación continua teoría 2013-14

## Objetivo

El objetivo es implementar un modulo para comunicarse con un teclado y un ratón mediante el puerto PS2.

El modulo identificará la tecla pulsada, o el movimiento del ratón. Este modulo será reutilizado en la práctica con microprocesador.

La figura siguiente muestra el código de tecla generado en hexadecimal por cada una de las teclas:

ESC 76	F1 05	F2 06	F3 04	F4 0C	F5 03	F6 0B	F7 83	F8 0A	F9 01	F10 09	F11 78	F12 07
-----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	-----------	-----------	-----------

~ 0E	! 1 16	@ 2 1E	# 3 26	\$ 4 25	% 5 2E	^ 6 36	& 7 3D	* 8 3E	( 9 46	) 0 45	- + 4E	= 5 55	! 6 5D
---------	-----------	-----------	-----------	------------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------

TAB 0D	Q 15	W 1D	E 24	R 2D	T 2C	Y 35	U 3C	I 43	O 44	P 4D	{ 54	} 5B	← 66
-----------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------

Caps 58	A 1C	S 1B	D 23	F 2B	G 34	H 33	J 3B	K 42	L 4B	: ; 4C	" " 52	← 5A
------------	---------	---------	---------	---------	---------	---------	---------	---------	---------	-----------	-----------	---------

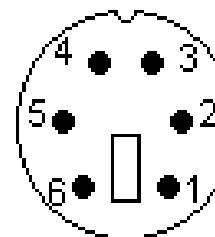
Shift 12	Z 1A	X 22	C 21	V 2A	B 32	N 31	M 3A	, < 41	, > 49	? / 4A	Shift 59
-------------	---------	---------	---------	---------	---------	---------	---------	-----------	-----------	-----------	-------------

Ctrl 14	Alt 11	SPACE 29	Alt E0 11	Ctrl E0 14
------------	-----------	-------------	--------------	---------------



## El conector del teclado/ratón:

En la figura a continuación se muestran los 5 pinos del conector PS/2:



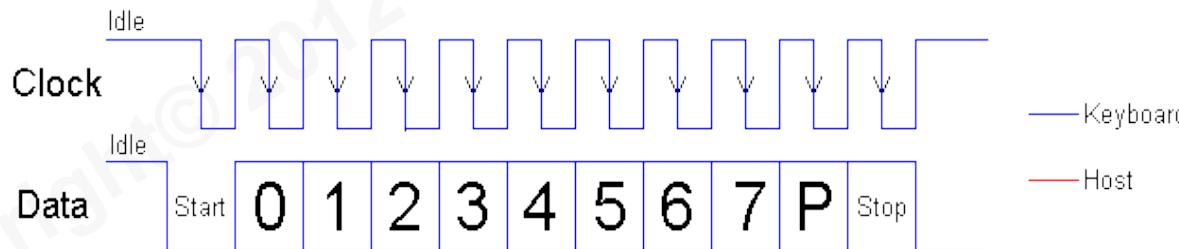
1. Reloj
2. GND
3. Datos
4. No conectado
5. +5V (VCC)
6. No conectado

## Protocolo de comunicación del Teclado

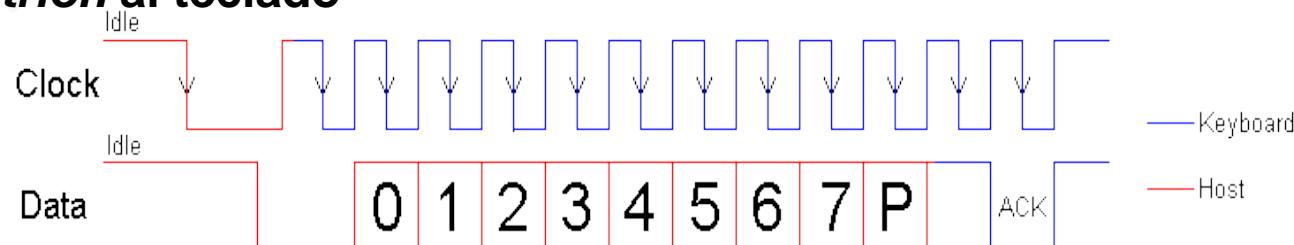
La comunicación es bidireccional.

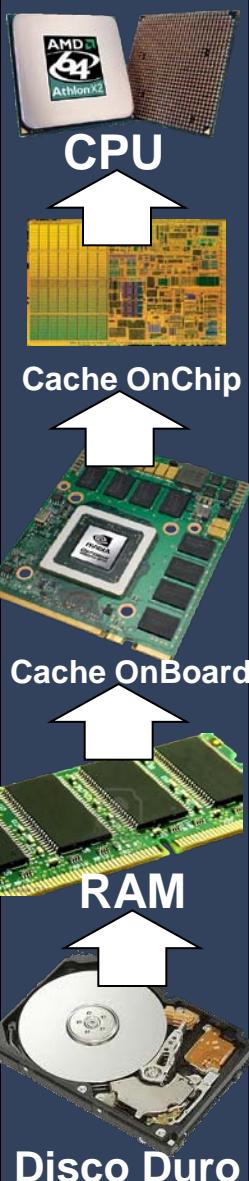
Cuando no hay comunicación la línea de **Datos** y la de **Reloj** quedan a valor alto, en ese caso se deben dejar esas líneas a 'Z' en vuestro módulo.

### Del teclado al anfitrión



### Del anfitrión al teclado



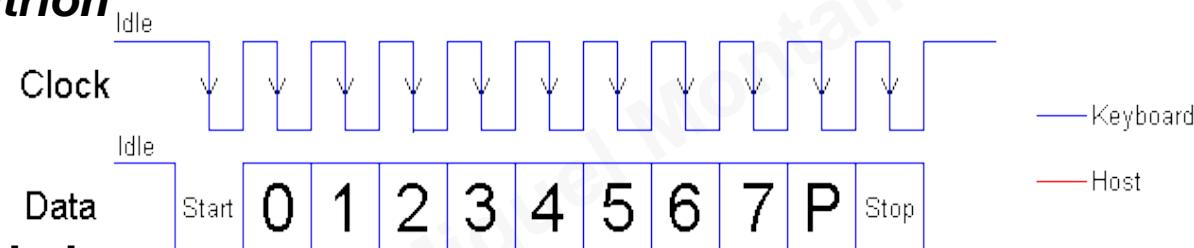


## Protocolo de comunicación del Teclado

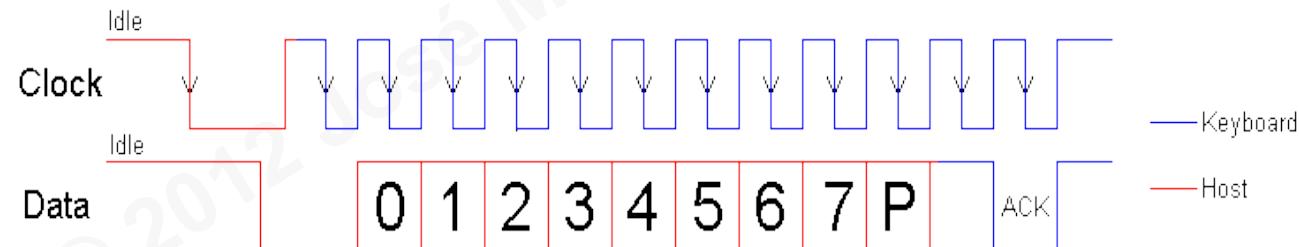
La comunicación es bidireccional.

Cuando no hay comunicación la línea de **Datos** y la de **Reloj** quedan a valor alto, en ese caso se deben dejar esas líneas a 'Z' en vuestro módulo.

### Del teclado al anfitrión

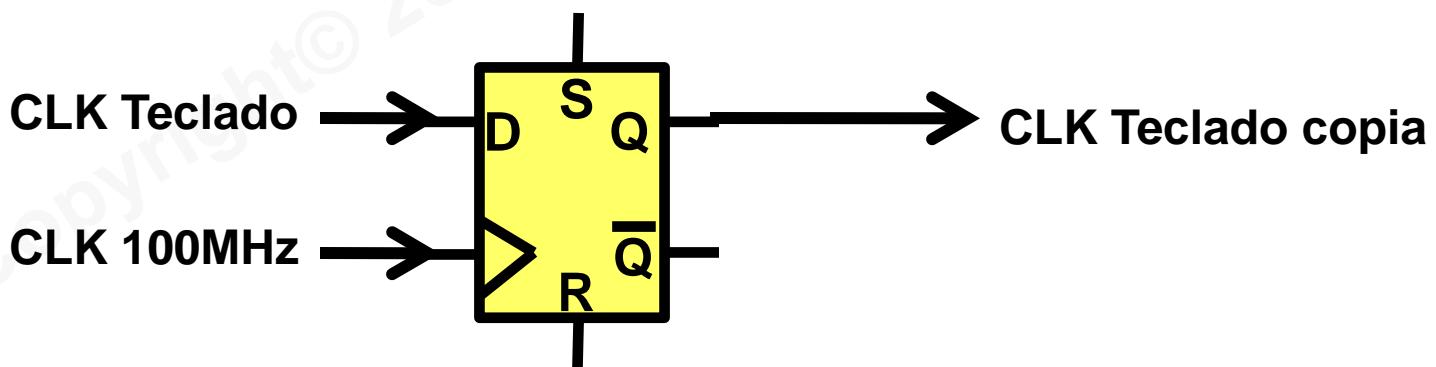
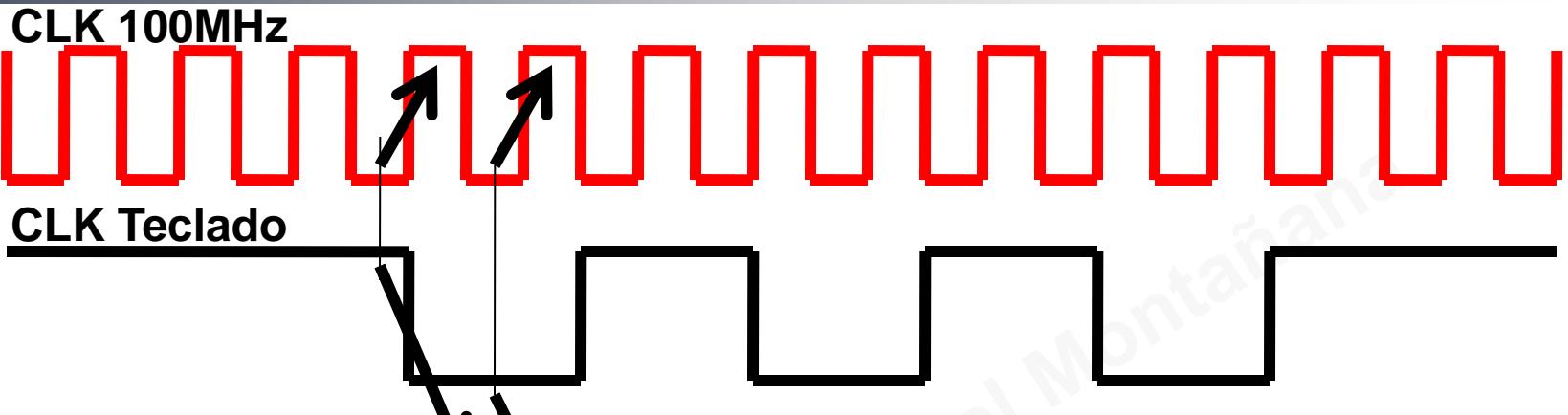
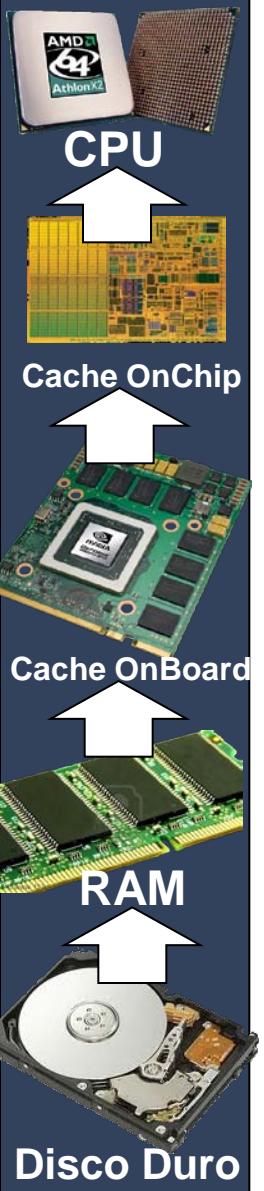


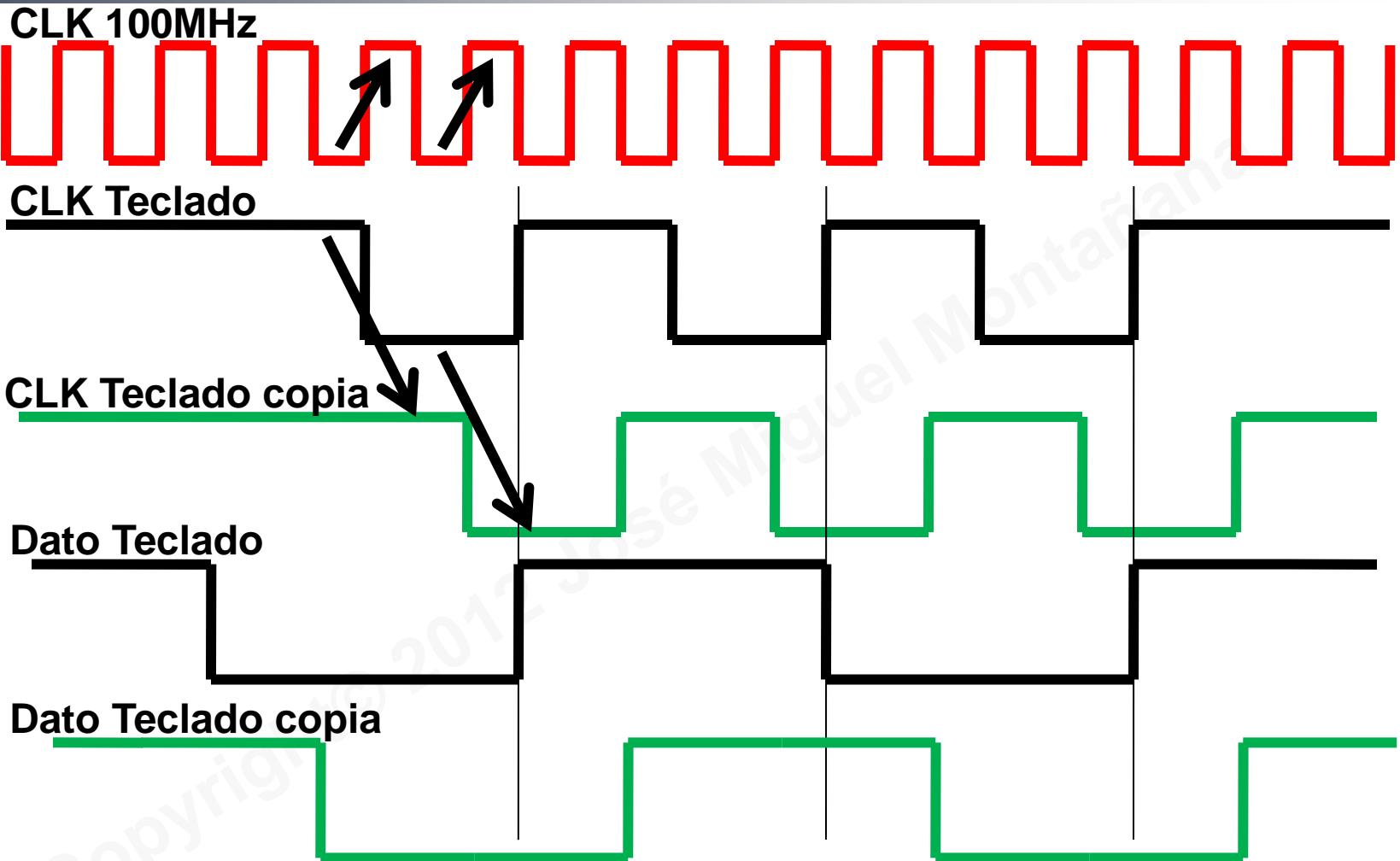
### Del anfitrión al teclado



## Cálculo del bit de paridad:

7 bits of data	(count of 1 bits)	8 bits including parity	
		even	odd
0000000	0	00000000	00000001
1010001	3	10100011	10100010
1101001	4	11010010	11010011
1111111	7	11111111	11111110





```

IF CLK_100MHz'event and CLK_100MHz='1' THEN
  IF CLK Teclado='1' AND CLK Teclado_copia='1'
    AND DatoTeclado='0' and DatoTecladoCopia='1' THEN
      START <='1';
    ELSE START <='0'; END IF;
  END IF;
END IF;
  
```

Pregunta:

Uso de lógica combinacional para generar una señal de reloj?

Respuesta:

Mejor NO hacerlo. Por ejemplo:

AB C

00 0

01 0

10 0

11 1

$C = A \text{ and } B$

SEÑAL AB:

00	01	10	11	00	01	10	11	00
----	----	----	----	----	----	----	----	----

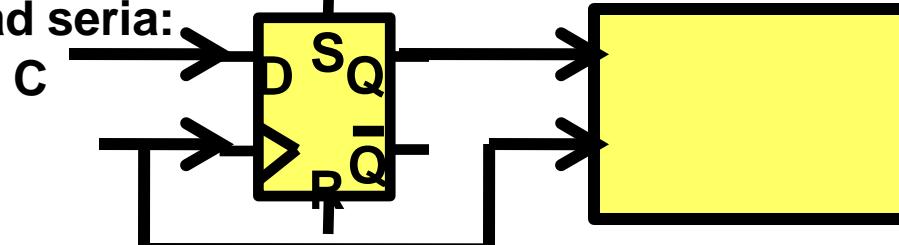
SEÑAL C ESPERADA (simulador):

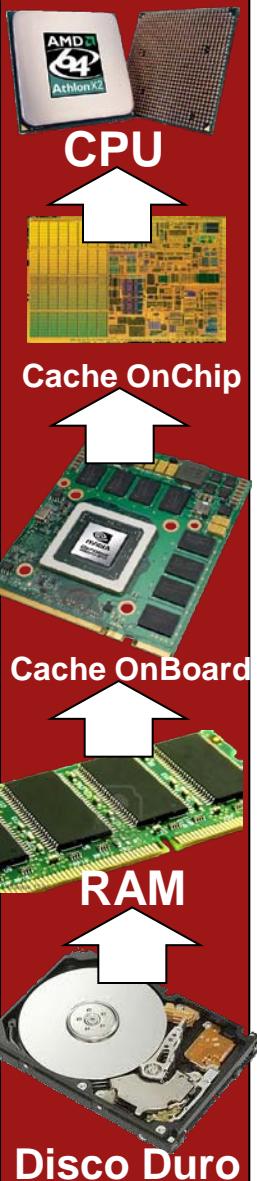
0	0	0	1	0	0	0	1	0
---	---	---	---	---	---	---	---	---

SEÑAL C REAL (placa):

0	0	0	1	0	0	0	1	0
---	---	---	---	---	---	---	---	---

PERO una posibilidad seria:





- **Los tiempos de propagación de las señales a través de elementos combinacionales:**
  - No son nulos.
  - No tienen valores completamente precisos debido a variaciones del proceso de fabricación, envejecimiento de dispositivos, temperatura, etc.
- **Consecuencias funcionales:**
  - En respuesta a una transición de entrada, una salida puede hacer varias transiciones funcionalmente imprevistas (glitches) en cualquier momento de su intervalo de incertidumbre.
  - Nunca se puede asegurar la simultaneidad en las transiciones de un conjunto de salidas, aunque sean provocadas por una misma transición de entrada (es decir, las líneas de una salida vectorial no conmutan simultáneamente).
- **Si las salidas sólo se muestran cuando han alcanzado el régimen permanente (diseño síncrono), el efecto de los glitches casi siempre puede ignorarse, no obstante:**
  - Si son muy frecuentes aumenta considerablemente el consumo del circuito.
  - Deben tenerse en cuenta cuando se comunica con sistemas asíncronos (lógicas de reset, memorias, etc.)
- **Los glitches aparecen cuando un circuito tiene riesgos.**
- **Se dice que un diseño tiene riesgos cuando las diferencias de retardo entre los caminos que una señal puede seguir conducen a una inconsistencia en la interpretación de su valor.**
- **Riesgos estáticos** provocan glitches en una señal que debiera ser constante.
  - típicamente se originan porque dos señales complementarias se hacen iguales durante un intervalo de tiempo corto.
- **Riesgos dinámicos** provocan glitches en una señal que debiera hacer una única transición.
  - típicamente se originan porque dos señales que debieran ser equivalentes se hacen diferentes durante un intervalo de tiempo corto.

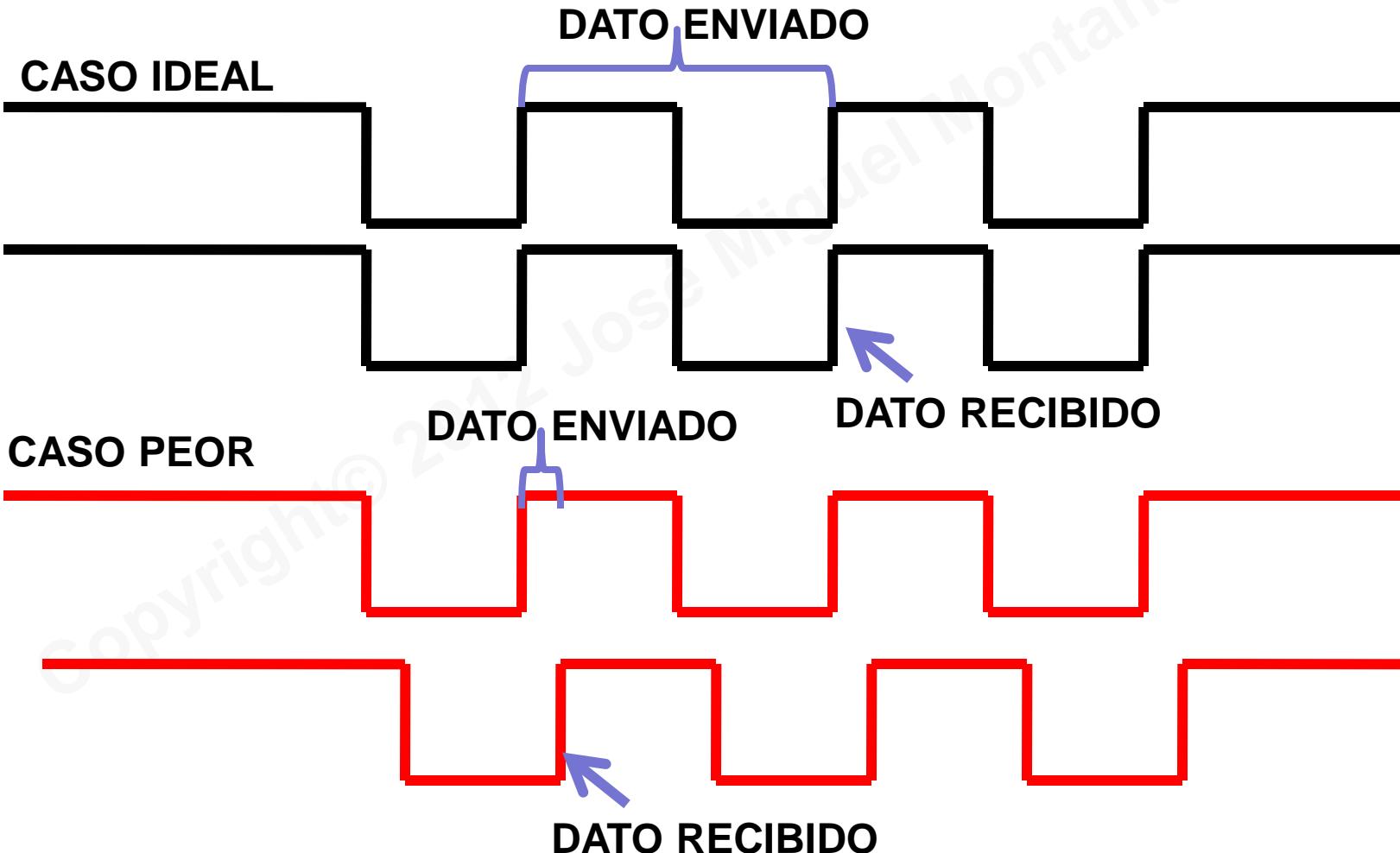


Pregunta:

No carga el proyecto en la placa del laboratorio, porque?

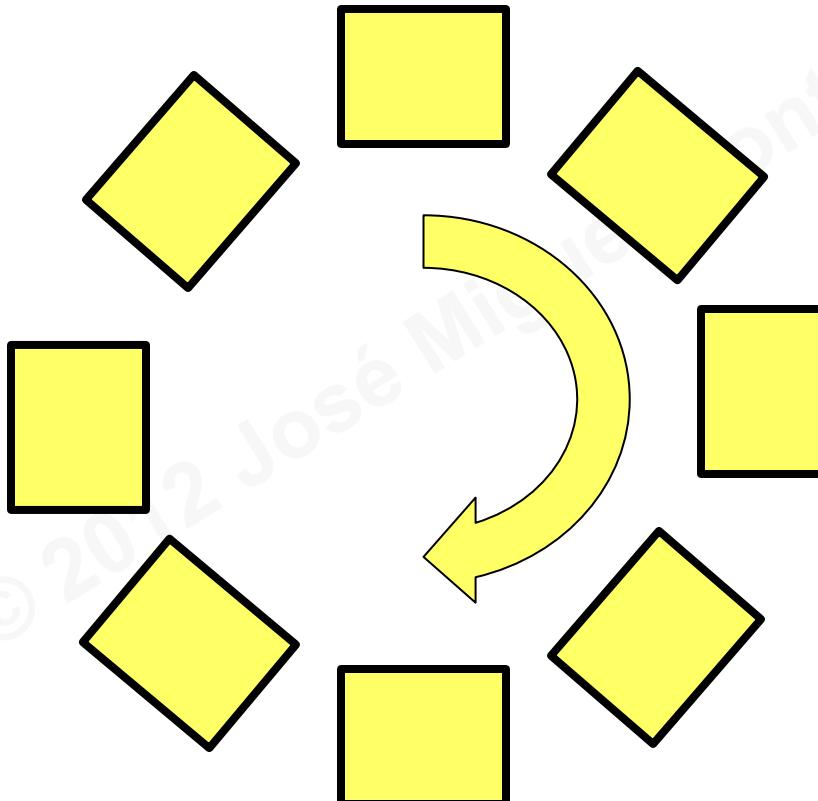
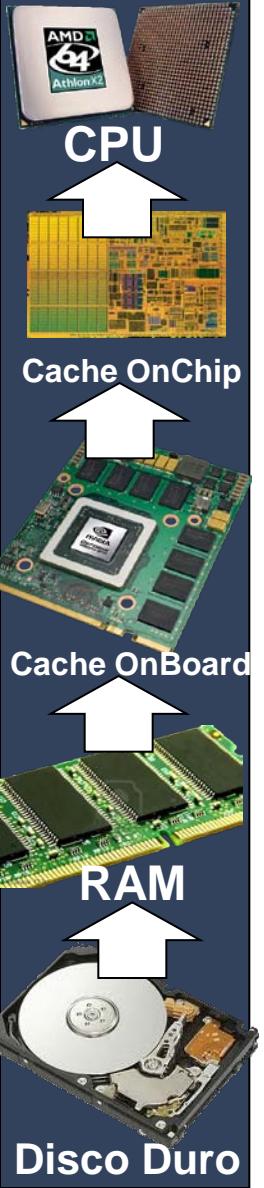
Respuesta:

Parece que lo mas probable es un problema de sincronismo.

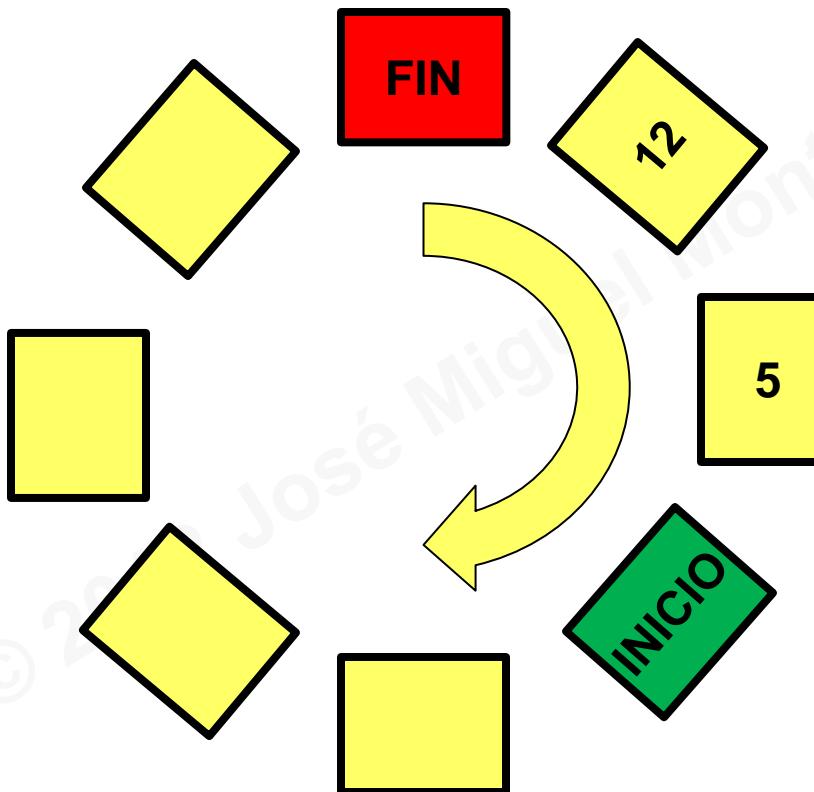


## Pregunta:

### Que es una memoria FIFO?



Pregunta:  
Que es una memoria FIFO?



Cuál es la condición de memoria LLENA?

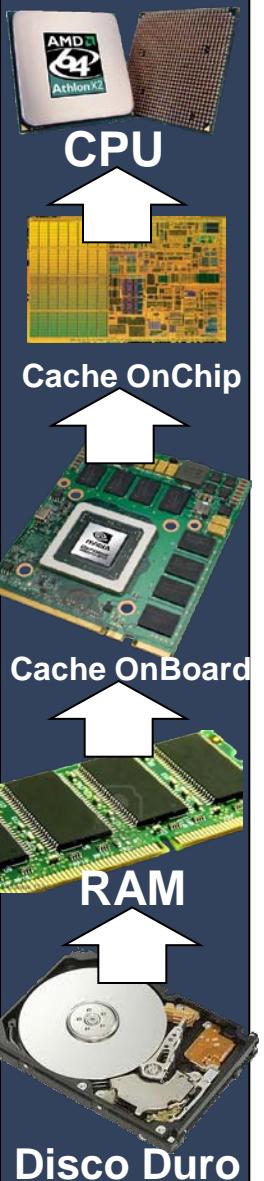
.....

Cuál es la condición de memoria VACIA?

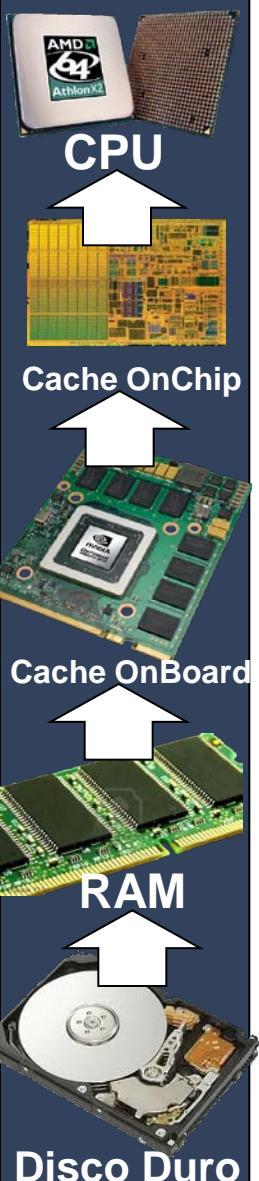
.....

# DISEÑO AUTOMÁTICO DE SISTEMAS

## Curso 2013-2014



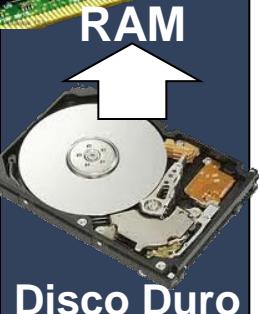
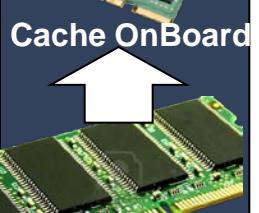
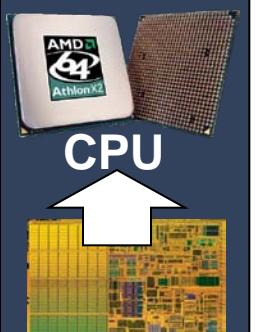
KEY	Press	Release	KEY	Press	Release	KEY	Press	Release
A	1C	F0,1C	'	0E	F0,0E	CAPS	58	F0,58
B	32	F0,32	\	4E	F0,4E	NUM	77	F0,77
C	21	F0,21	[	5D	F0,5D	SCROLL	7E	F0,7E
D	23	F0,23	]	54	F0,54	HOME	E0,6C	E0,F0,6C
E	24	F0,24	;	5B	F0,5B	END	E0,69	E0,F0,69
F	2B	F0,2B	,	4C	F0,4C	DELETE	E0,71	E0,F0,71
G	34	F0,34	.	41	F0,41	INSERT	E0,70	E0,F0,70
H	33	F0,33	/	49	F0,49	PG DN	E0,7A	E0,F0,7A
I	43	F0,43	/	4A	F0,4A	PG UP	E0,7D	E0,F0,7D
J	3B	F0,3B						
K	42	F0,42	L SHFT	12	F0,12	U ARROW	E0,75	E0,F0,75
L	4B	F0,4B	L CTRL	14	F0,14	L ARROW	E0,6B	E0,F0,6B
M	3A	F0,3A	L GUI	E0,1F	E0,F0,1F	D ARROW	E0,72	E0,F0,72
N	31	F0,31	L ALT	11	F0,11	R ARROW	E0,74	E0,F0,74
O	44	F0,44	R SHFT	59	F0,59			
P	4D	F0,4D	R CTRL	E0,14	E0,F0,14	NUMPAD /	E0,4A	E0,F0,4A
Q	15	F0,15	R GUI	E0,27	E0,F0,27	NUMPAD *	7C	F0,7C
R	2D	F0,2D	R ALT	E0,11	E0,F0,11	NUMPAD -	7B	F0,7B
S	1B	F0,1B				NUMPAD +	79	F0,79
T	2C	F0,2C	ENTER	5A	F0,5A	NUMPAD .	71	F0,71
U	3C	F0,3C	ESC	76	F0,76	NUMPAD 0	70	F0,70
V	2A	F0,2A				NUMPAD 1	69	F0,69
W	1D	F0,1D	F1	5	F0,05	NUMPAD 2	72	F0,72
X	22	F0,22	F2	6	F0,06	NUMPAD 3	7A	F0,7A
Y	35	F0,35	F3	4	F0,04	NUMPAD 4	6B	F0,6B
Z	1A	F0,1A	F4	0C	F0,0C	NUMPAD 5	73	F0,73
0	45	F0,45	F5	3	F0,03	NUMPAD 6	74	F0,74
1	16	F0,16	F6	0B	F0,0B	NUMPAD 7	6C	F0,6C
2	1E	F0,1E	F7	83	F0,83	NUMPAD 8	75	F0,75
3	26	F0,26	F8	0A	F0,0A	NUMPAD 9	7D	F0,7D
4	25	F0,25	F9	1	F0,01			
5	2E	F0,2E	F10	9	F0,09			
6	36	F0,36	F11	78	F0,78	BKSP	66	F0,66
7	3D	F0,3D	F12	7	F0,07	SPACE	29	F0,29
8	3E	F0,3E				TAB	0D	F0,0D
9	46	F0,46						



<b>KEY</b>	<b>Press</b>	<b>Release</b>
PRNT SCRN	E0,12, E0,7C	E0,F0, 7C,E0, F0,12
PAUSE	E1,14,77, E1,F0,14, F0,77	NONE
POWER	E0, 37	E0, F0, 37
SLEEP	E0, 3F	E0, F0, 3F
WAKE	E0, 5E	E0, F0, 5E
NEXT TRACK	E0, 4D	E0, F0, 4D
PREVIOUS TRACK	E0, 15	E0, F0, 15
STOP	E0, 3B	E0, F0, 3B
PLAY/PAUSE	E0, 34	E0, F0, 34
MUTE	E0, 23	E0, F0, 23
VOLUME UP	E0, 32	E0, F0, 32
VOLUME DOWN	E0, 21	E0, F0, 21
MEDIA SELECT	E0, 50	E0, F0, 50
E-MAIL	E0, 48	E0, F0, 48
CALCULATOR	E0, 2B	E0, F0, 2B
MY COMPUTER	E0, 40	E0, F0, 40
SEARCH	E0, 10	E0, F0, 10
HOME	E0, 3A	E0, F0, 3A
BACK	E0, 38	E0, F0, 38
FORWARD	E0, 30	E0, F0, 30
STOP	E0, 28	E0, F0, 28
REFRESH	E0, 20	E0, F0, 20
FAVOURITES	E0, 18	E0, F0, 18

#### Ratón PS/2

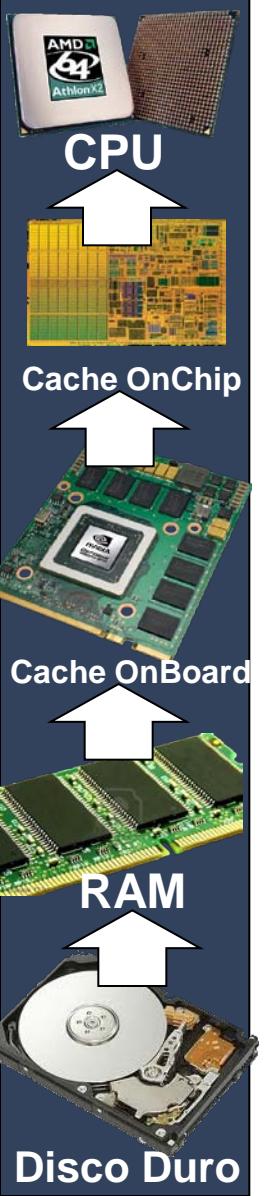
	7	6	5	4	3	2	1	0
Byte 1	YOVF	XOVF	YSIGN	XSIGN	1	CENTR	DER	IZQ
Byte 2					MOVIMIENTO EN X			
Byte 3					MOVIMIENTO EN Y			



Si el modo de scroll esta activado

	7	6	5	4	3	2	1	0
Byte 1	YOVF	XOVF	YSIGN	XSIGN	1	CENTR	DER	IZQ
Byte 2					MOVIMIENTO EN X			
Byte 3					MOVIMIENTO EN Y			
Byte 4					MOVIMIENTO EN LA RUEDA DE SCROLL			

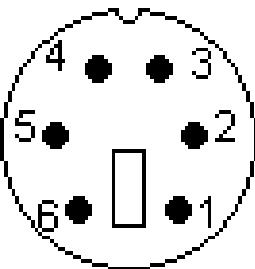
## INICIALIZACION DEL RATON

- 
1. At power-on, a mouse performs a power-on test internally. The mouse sends 1-byte data AA, which indicates that the test is passed, and then 1-byte data 00, which is the id of a standard PS2 mouse.
  2. The FPGA host sends the command, F4, to enable the stream mode. The mouse will respond with FE to acknowledge acceptance of the command.
  3. The mouse now enters the stream mode and sends normal data packets.

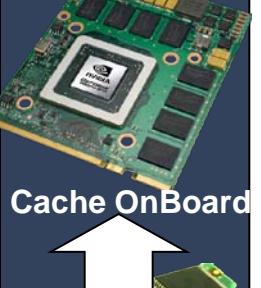
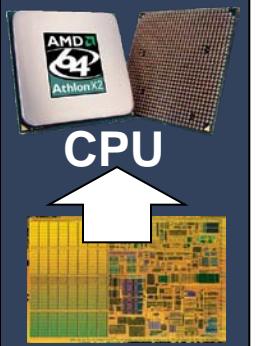
We can force the mouse to return to the initial state by sending the reset command:

- 1.- The FPGA host sends the command, FF, to reset the mouse. The Mouse will respond with FE to acknowledge acceptance of the command.
- 2 The mouse performs a power-on test internally and then sends AA 00. The stream mode be disabled during the process.

### El conector del teclado/ratón PS/2:

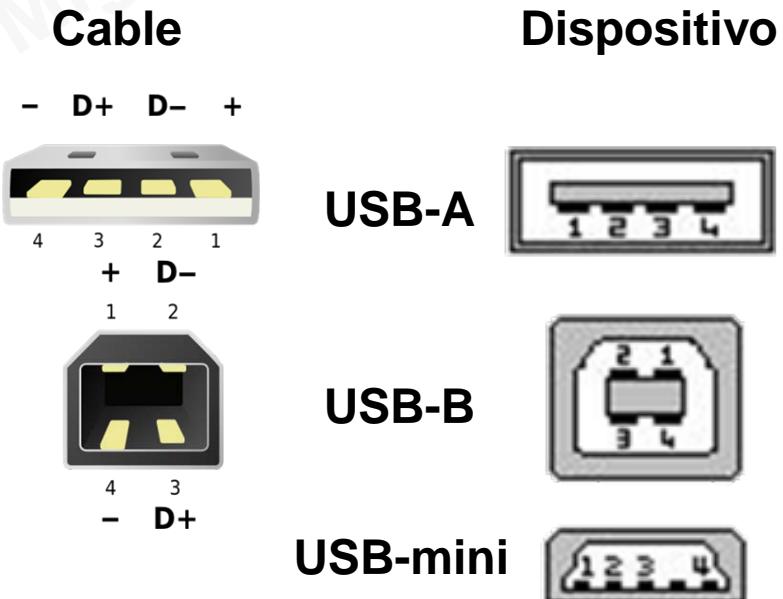


1. Reloj
2. GND
3. Datos
4. No conectado
5. +5V (VCC)
6. No conectado



### El conector del teclado/ratón USB:

Pin	Señal
1	Vcc 5V
2	Data-
3	Data+
4	Ground



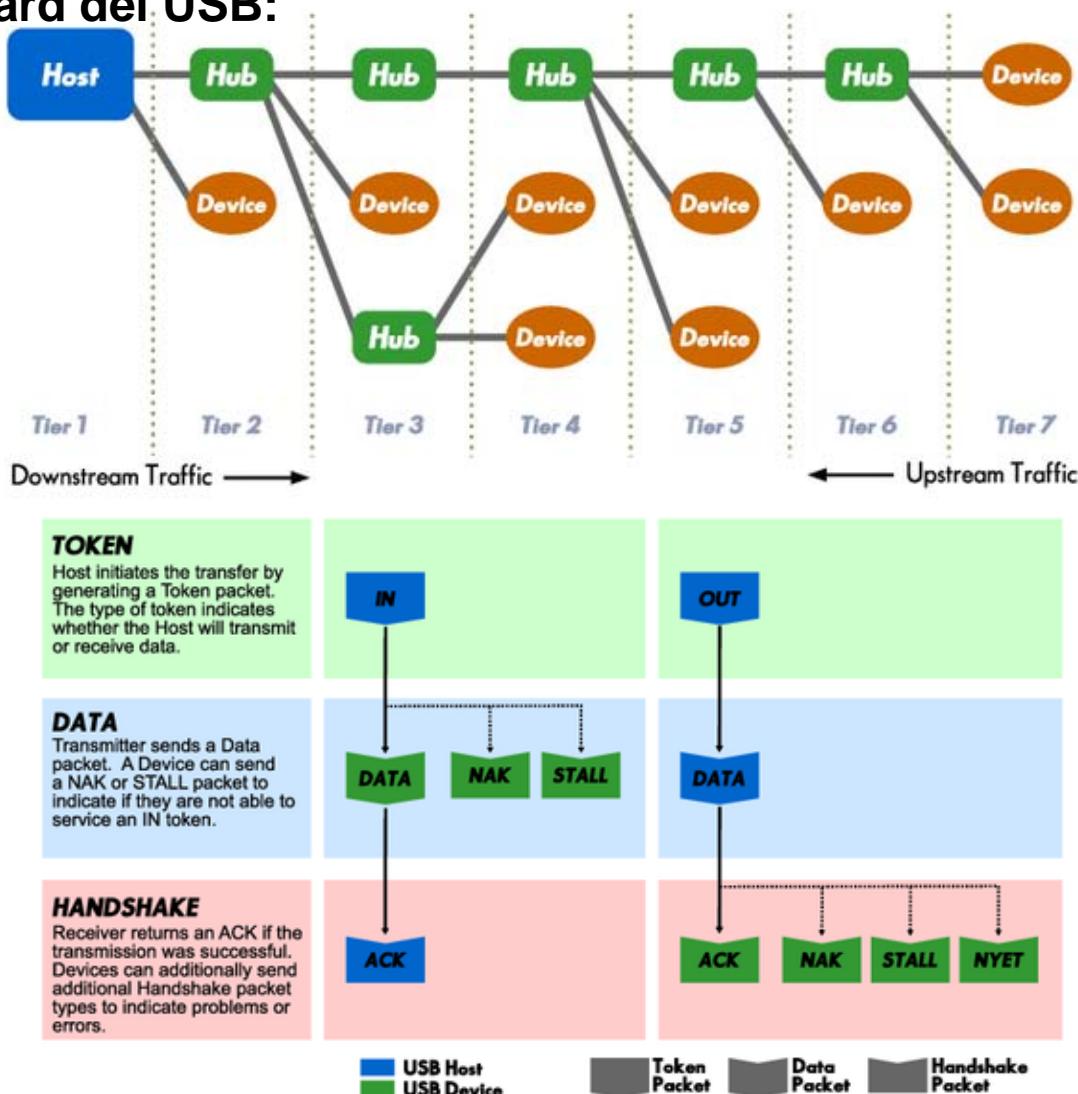
# DISEÑO AUTOMÁTICO DE SISTEMAS

## Curso 2013-2014

otros



### Standard del USB:



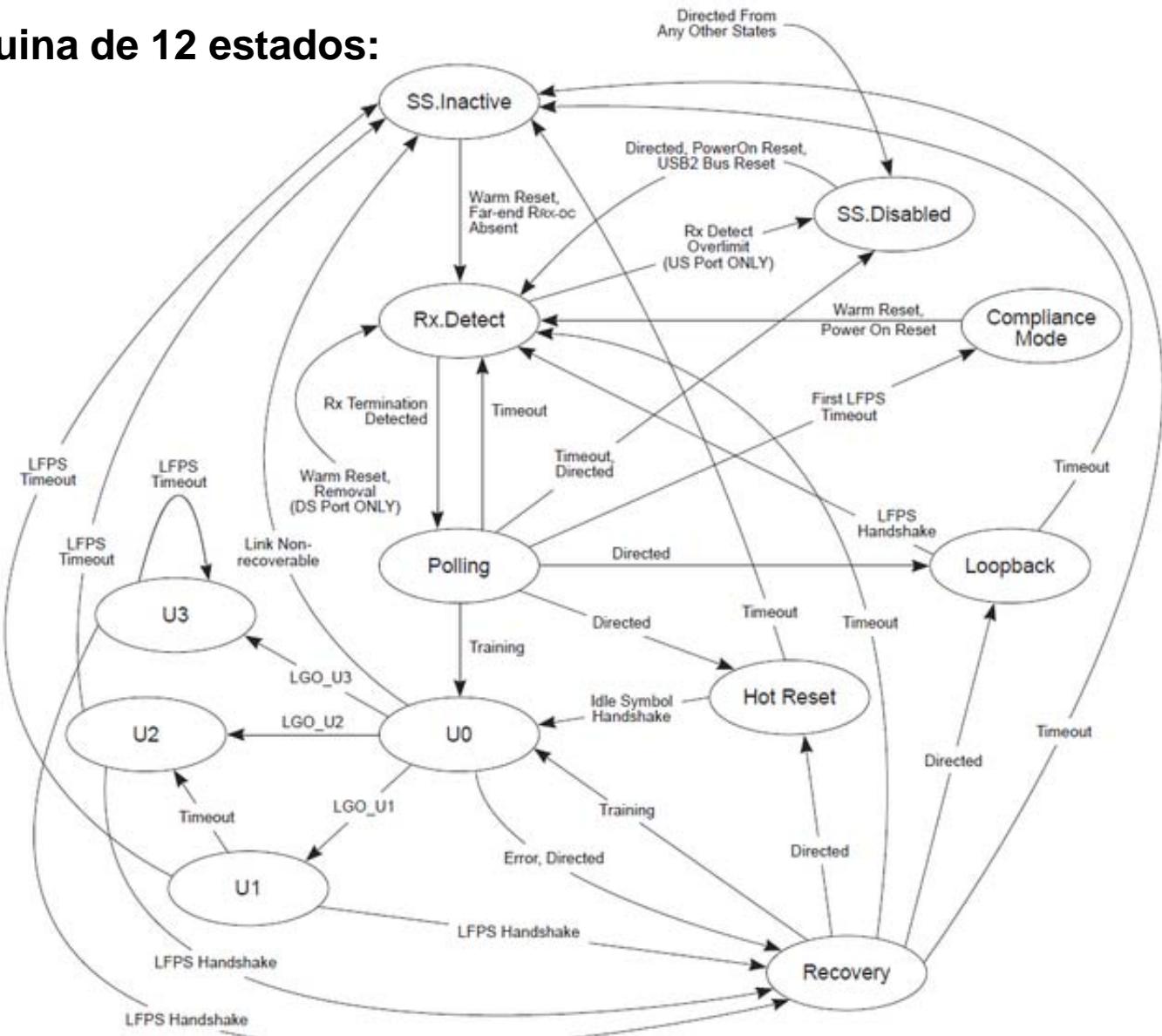
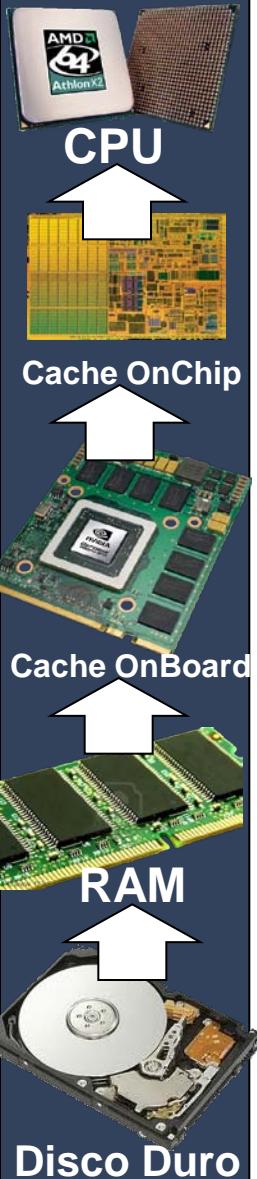
### Especificaciones del USB:

<http://www.usb.org/developers/docs/>

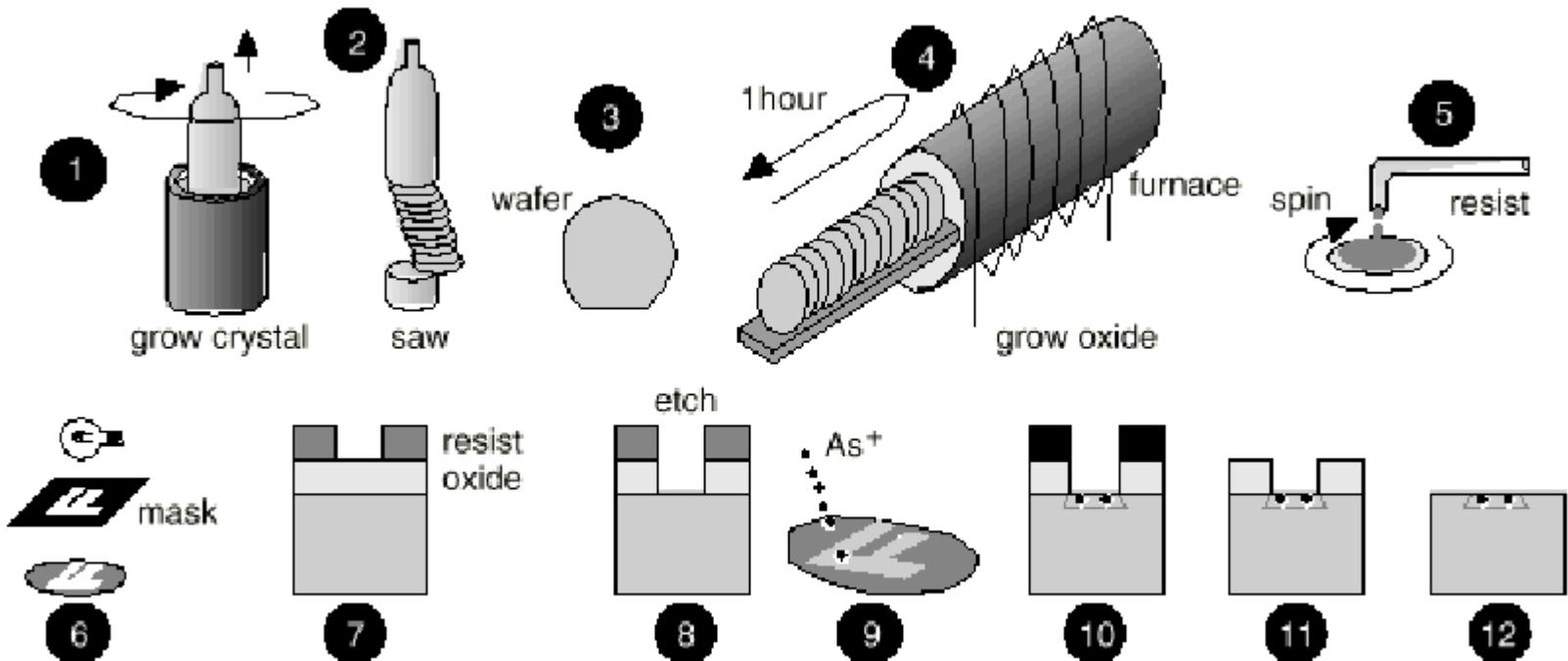
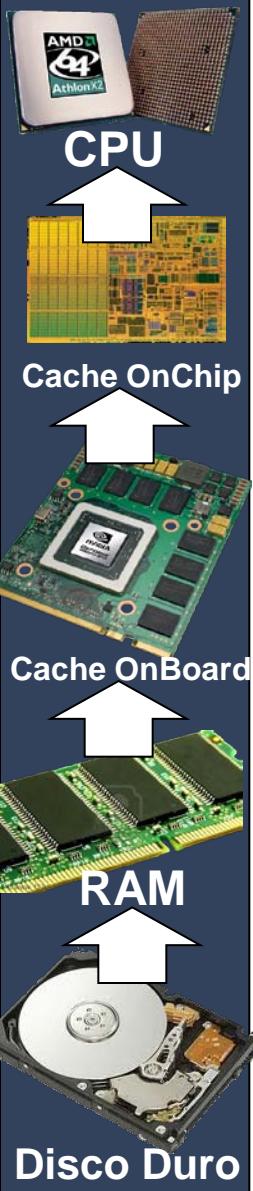
# DISEÑO AUTOMÁTICO DE SISTEMAS

## Curso 2013-2014

### Máquina de 12 estados:

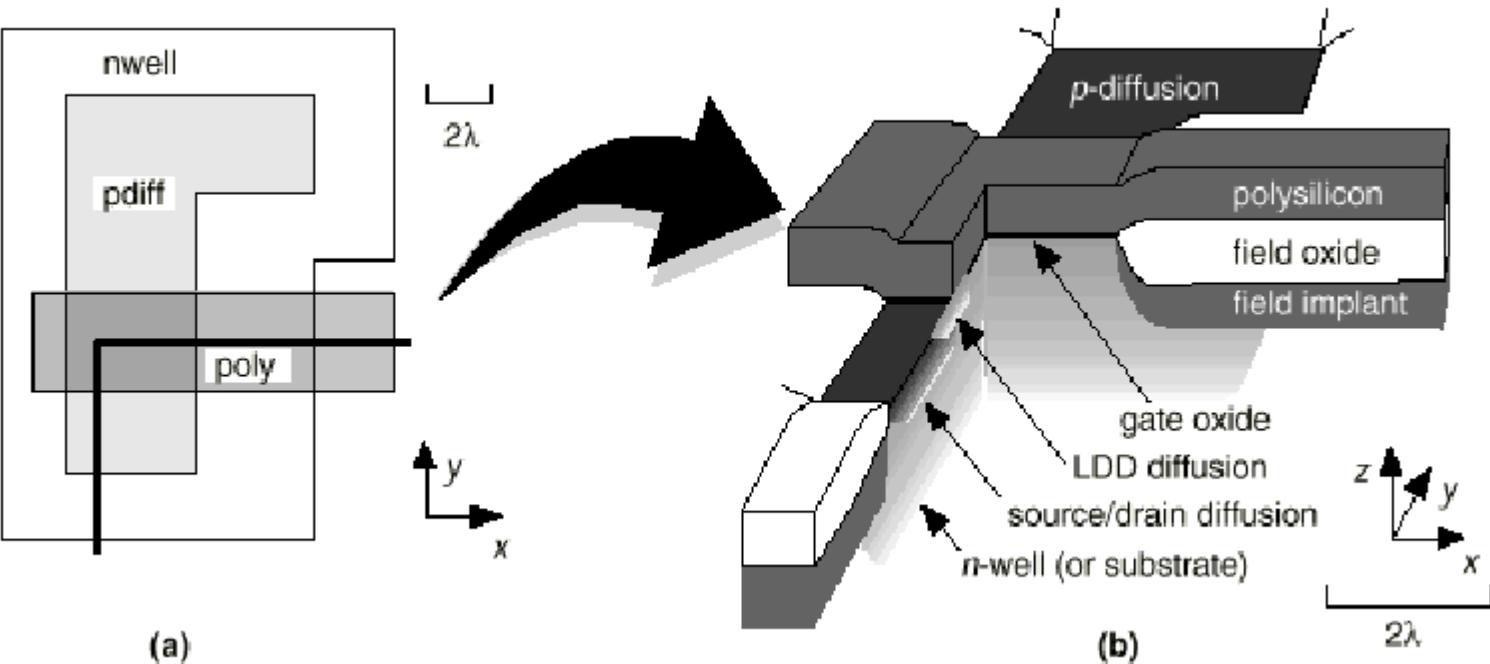


## proceso de fabricación



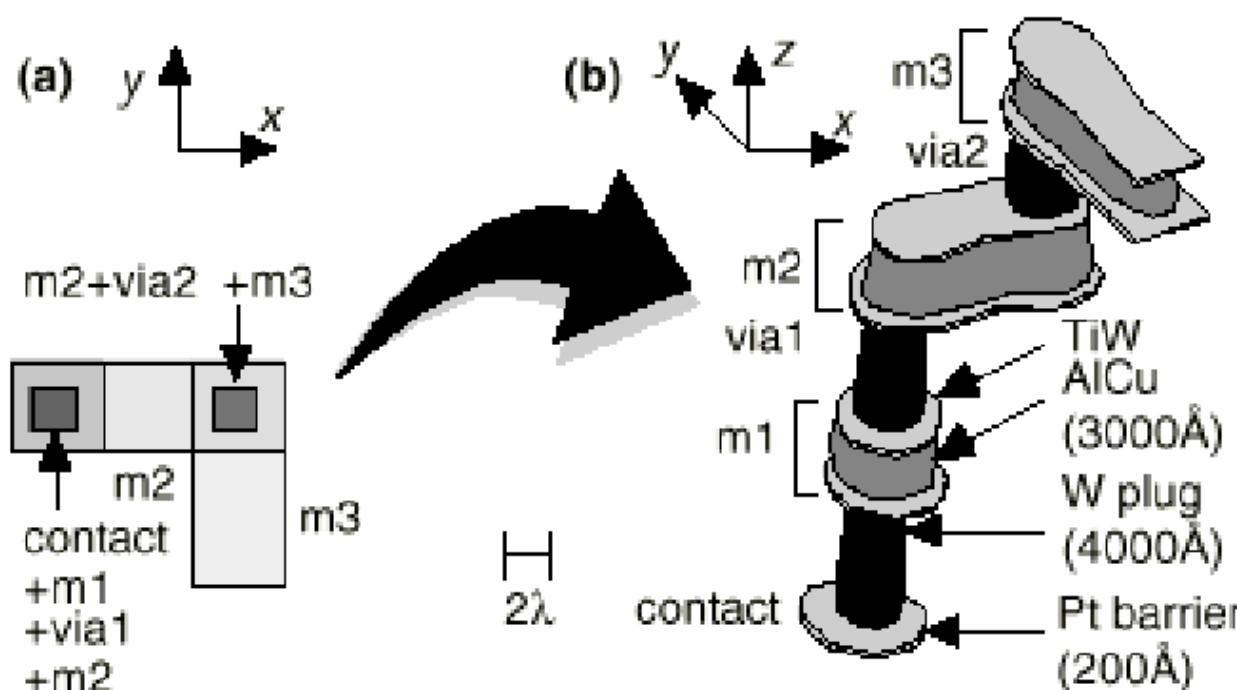
*Fabricación de un circuito integrado*

### proceso de fabricación



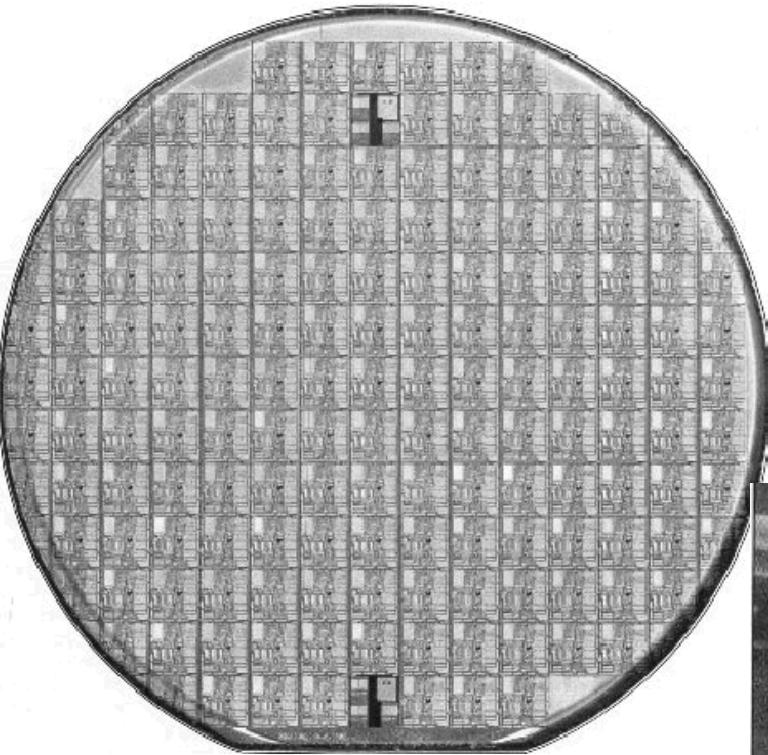
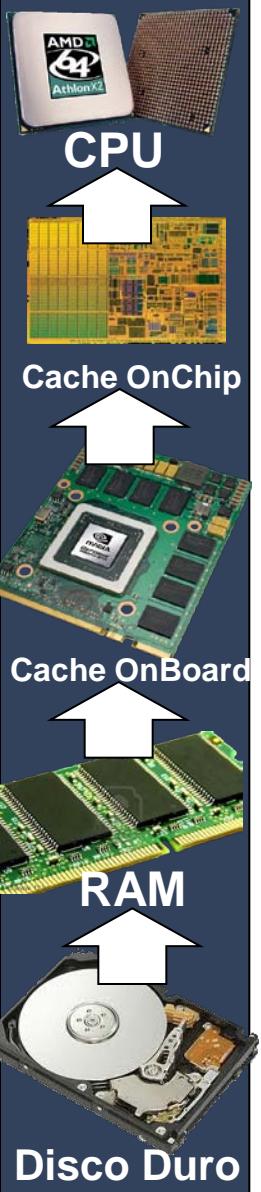
**Sección de un transistor**

### proceso de fabricación



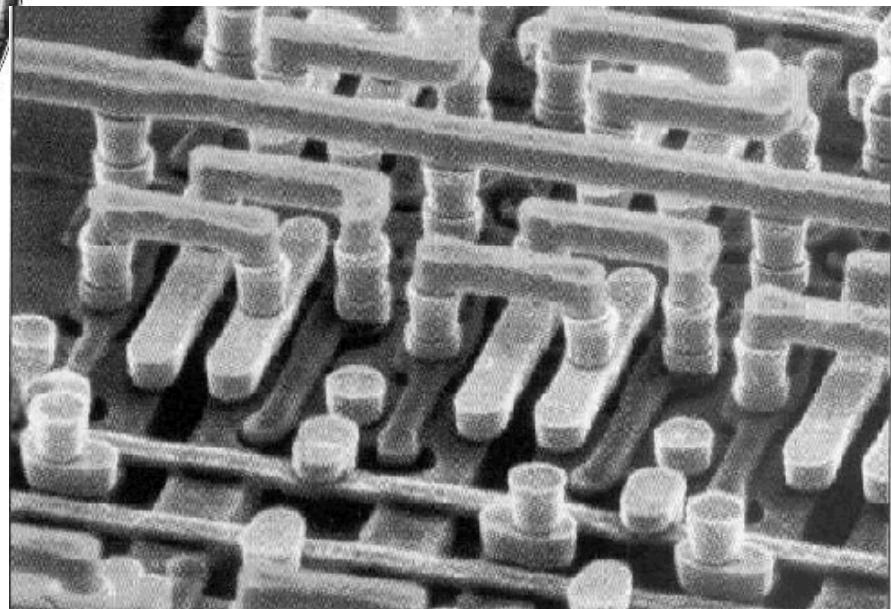
**Sección de una interconexión**

### proceso de fabricación



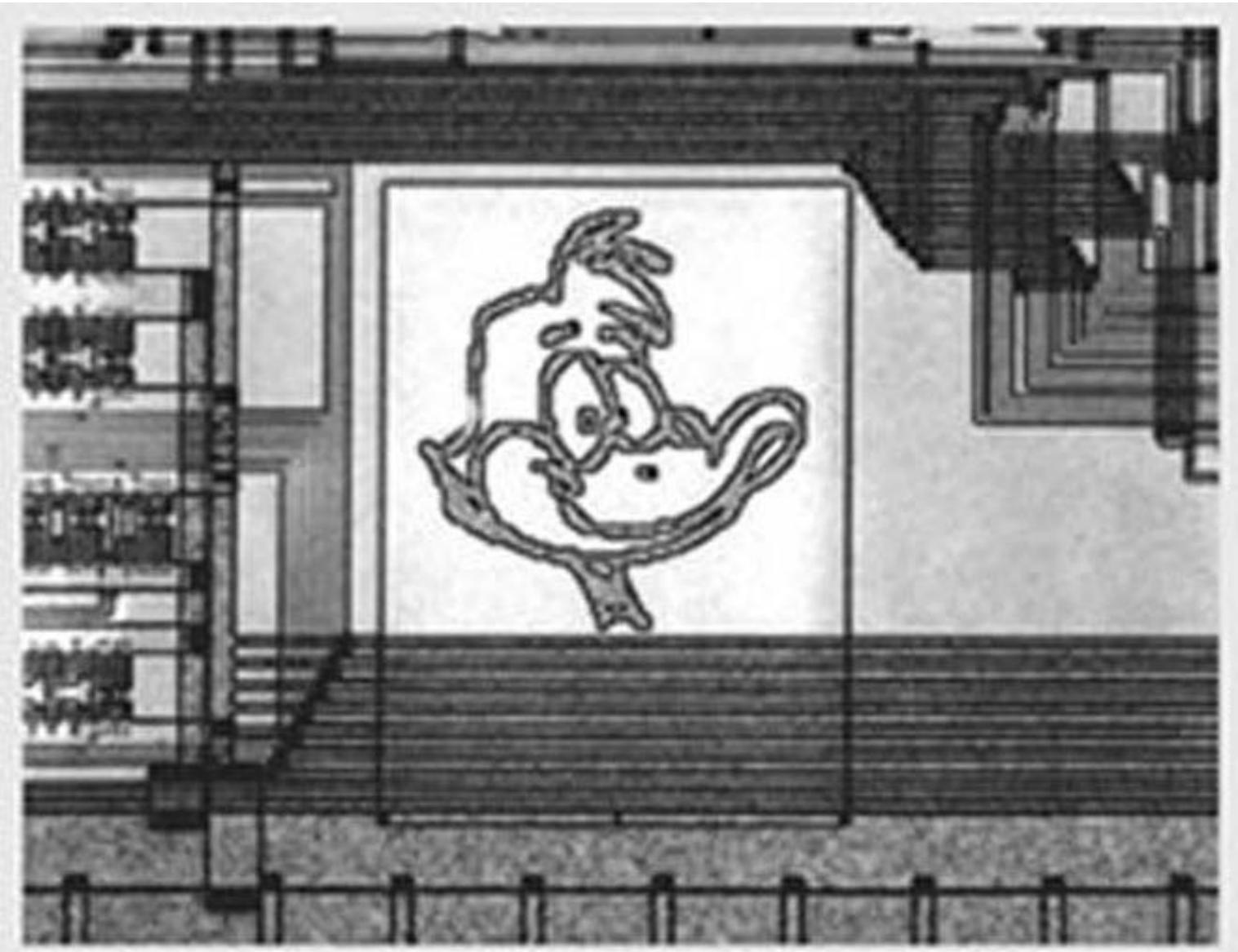
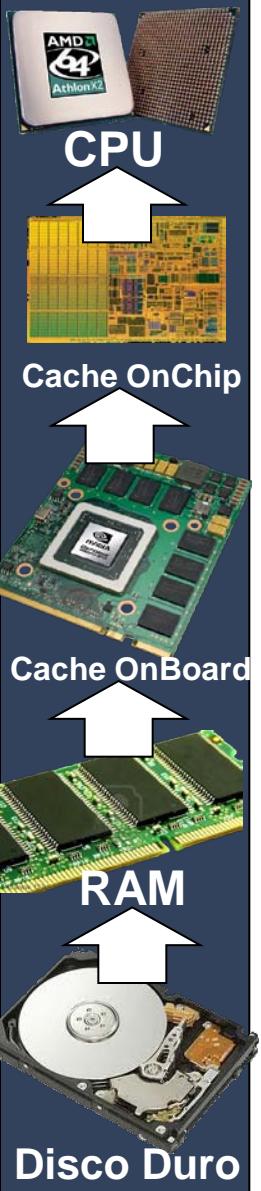
*Oblea fabricada*

*Vista al microscopio electrónico  
de una porción de circuito integrado*



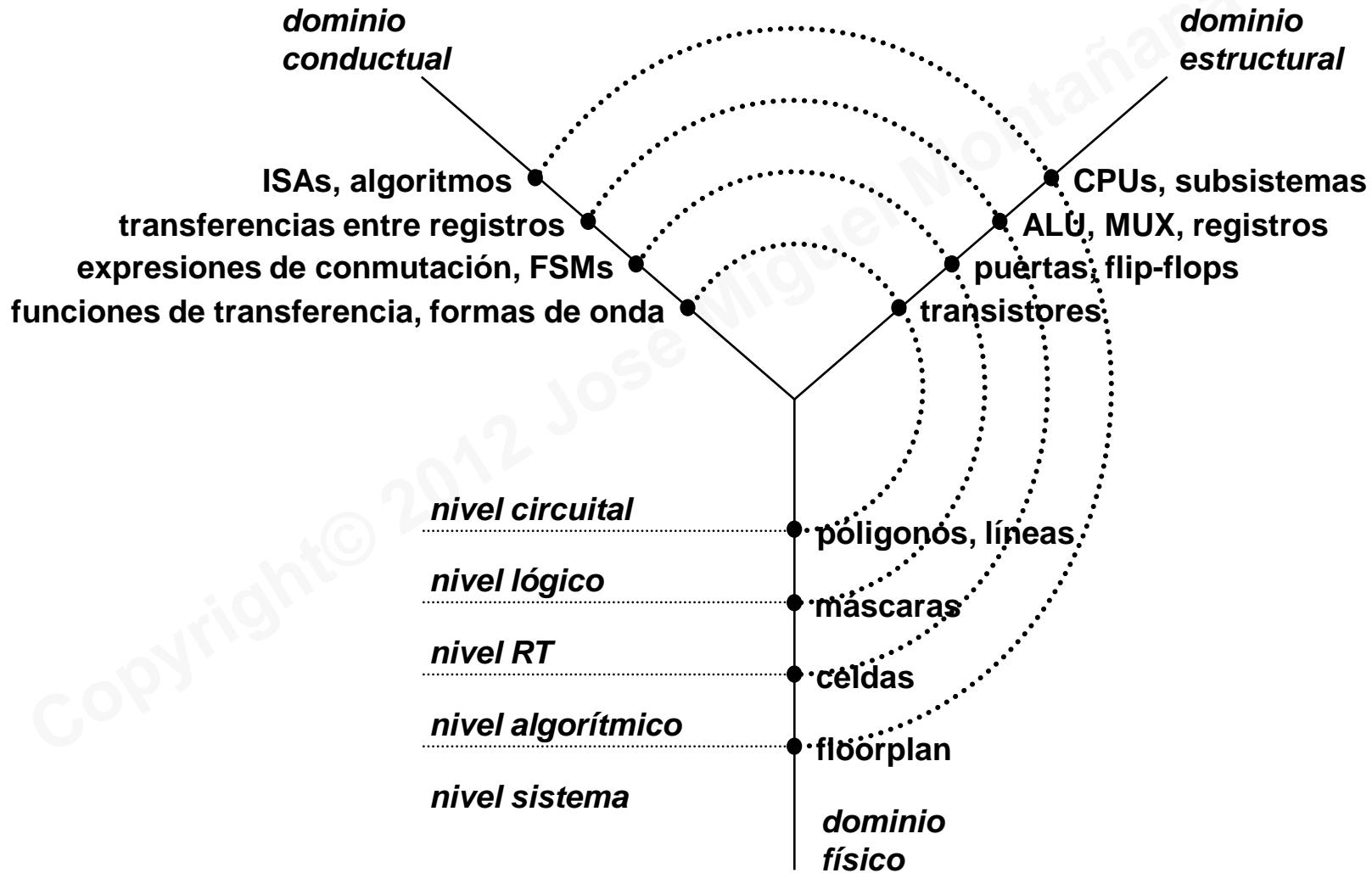
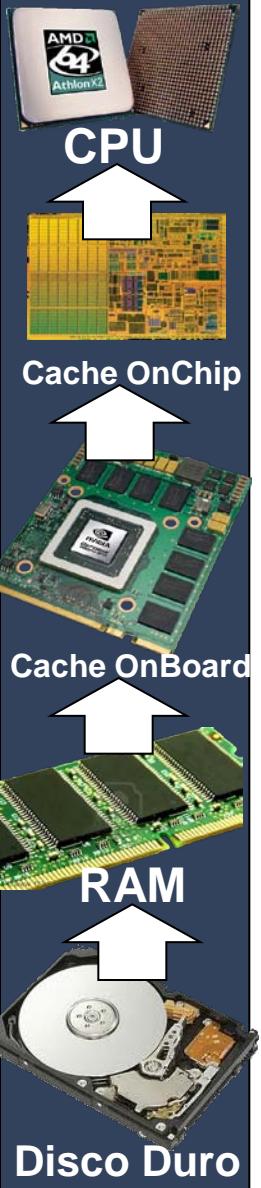
# DISEÑO AUTOMÁTICO DE SISTEMAS

## Curso 2013-2014



<http://dylan.tweney.com/tag/semiconductors/>

## niveles y dominios de descripción

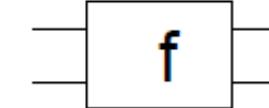




## Levels of Abstraction

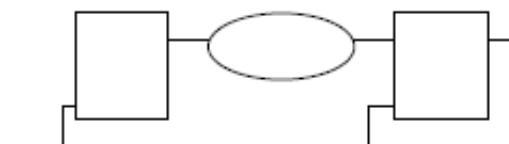
Fewer details,  
Faster design  
entry and  
simulation

Behavioral

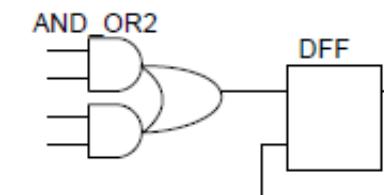


f

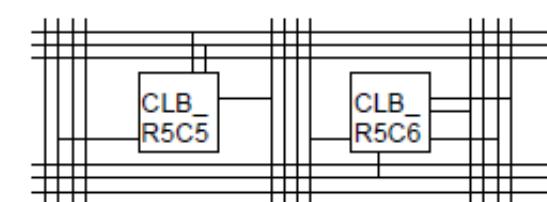
RTL



Logic



Layout



Technology  
specific details,  
slower design  
entry and  
simulation

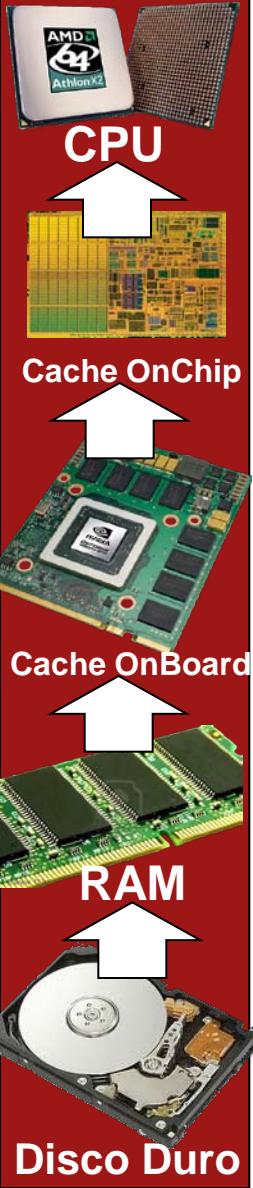
Behavioral to RTL - Slide 3

© 1999 TSI, Inc.  
All Rights Reserved

No part of this document may be reproduced or transmitted without the  
express written permission of the Director of Xilinx Customer Education.

 XILINX®

## síntesis y validación (i)



- **El objetivo global de todo proceso de síntesis es obtener una descripción física a nivel circuital a partir de una descripción conductual lo más abstracta posible.**
  - La descripción de partida se denomina especificación y la transformada, implementación.
  - En el caso del diseño automático se trata que este proceso se realice con la mínima intervención humana.
- **Conseguir este objetivo en un sólo paso es imposible, por lo que se realiza encauzando los resultados a través de una colección de procesos especializados:**
  - Síntesis de alto nivel.
  - Síntesis RT.
  - Síntesis lógica.
  - Síntesis física.
- **Consustancial a síntesis está la validación, que chequea si un sistema se ajusta a unos ciertos requisitos, que pueden ser:**
  - Funcionales: el sistema se comporta adecuadamente.
    - Simulación, testeo, chequeo de reglas, verificación formal.
  - No funcional: el sistema tiene cierto nivel de calidad
    - Simulación, testeo, análisis.

## métodos de síntesis (i)

### Síntesis de alto nivel o algorítmica

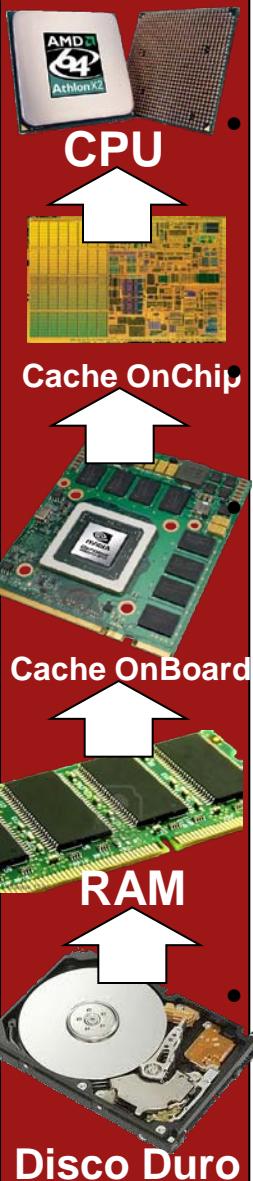
**Entrada:** una descripción conductual de nivel algorítmico de una computación y una colección de ligaduras.

- **Ligaduras típicas:** latencia, tiempo de ciclo, número y tipo de recursos, cadencia de admisión de datos.

**Salida:** una ruta de datos (netlist de nivel RT) y una descripción conductual de nivel RT del controlador (secuencia de transferencias entre registros).

#### Tareas:

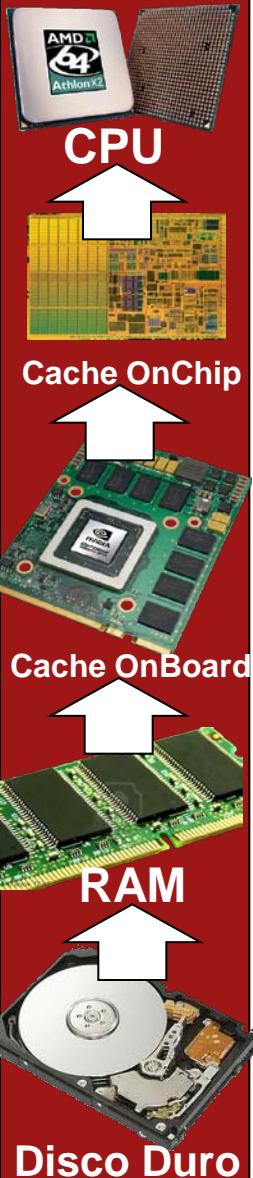
- **Planificación de operaciones (scheduling):** fija el ciclo en que se ejecuta cada una de las operaciones que forman la computación.
  - **Selección de recursos (resource binding):** determina el número y tipo de recursos funcionales (FUs, regs, mux/buses) requeridos para realizar la computación
  - **Asignación de recursos (resource allocation):** fija el recurso en el que se ejecuta, almacena o transmite cada una de las operaciones y objetos de datos que la forman.
  - **Otras tareas:** transformación de operaciones, transformación de bucles, segmentación
- Cuando se parte de una de una descripción conductual explícitamente planificada, o de una secuencia de transferencias entre registros sin referencia a recursos, se conoce como síntesis RT.**



## métodos de síntesis (i)

### Síntesis lógica

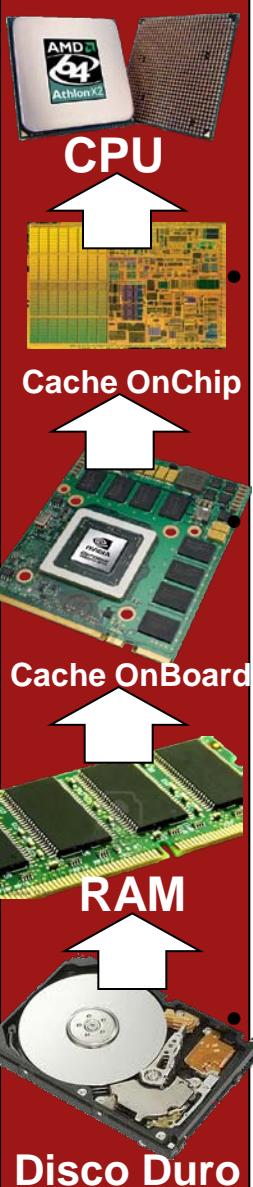
- Entrada: una descripción conductual formada por una o varias funciones combinacionales, secuenciales simples y máquinas de estados finitos y una colección de ligaduras.
- Ligaduras típicas: retardo/longitud del camino crítico, frecuencia de reloj, área equivalente, consumo de potencia
- Salida: una netlist de puertas y biestables.
- Tareas:
  - **Minimización combinacional:** construye una netlist reducida (a dos niveles o multinivel) capaz de implementar una cierta función lógica.
  - **Reestructuración combinacional:** modifica la ubicación relativa y número de las puertas de una netlist combinacional para optimizar su rendimiento global.
  - **Minimización de estados:** reduce el número de estados de una FSM
  - **Codificación de estados:** elige una codificación binaria de estados que minimice las funciones de estado y salida de una FSM.
  - **Reestructuración secuencial:** modifica la ubicación relativa y número de los biestables de una netlist secuencial para optimizar su rendimiento global
  - **Proyección tecnológica:** transforma una netlist general de puertas y biestables en otra equivalente cuyos componentes están limitados en cantidad o tipo.
  - **Generación de test:** crea una colección de estímulos capaces de detectar un cierto porcentaje de errores posibles provocados por fallos del proceso de fabricación.





## Síntesis física

- **Entrada:** una netlist de celdas.
- **Salida:** dependiendo del estilo de diseño microelectrónico, desde una colección de máscaras para la fabricación del circuito, hasta la configuración a ser descargada sobre una FPGA.
- **Tareas (dependientes del estilo de diseño microelectrónico):**
  - **Proyección tecnológica:** transforma una netlist de puertas y biestables en otra equivalente de las celdas pertenecientes a una biblioteca tecnológica.
  - **Particionamiento:** división de una netlist de celdas en varias particiones que se procesarán por separado.
  - **Floorplaning:** selección de la ubicación aproximada de cada una de las particiones o macroceldas que componen un circuito y de las regiones de interconexión.
  - **Ubicación de celdas (placement):** selección de la ubicación dentro de una cierta región de cada una de las celdas que componen una partición.
  - **Rutado (routing) global:** asignación de interconexiones a regiones de interconexión.
  - **Rutado detallado:** trazado físico de las interconexiones.
  - **Compactado:** eliminación de los espacios libres de un layout.
- **Cuando la síntesis física es full-custom comprende otras tareas (no automatizadas) como la obtención de un esquemático de transistores, dimensionado de los mismos, trazado de máscaras, etc.**

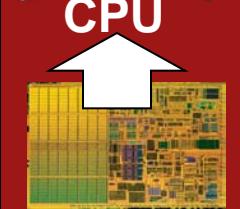


## Validación por simulación

- **Un simulador es una herramienta que permite reproducir el comportamiento (típicamente temporal) de un circuito a un cierto nivel de abstracción, bajo varios escenarios de operación**
- El diseñador facilita a la herramienta un modelo ejecutable del circuito, una colección de estímulos de entrada y evalúa las formas de onda que genera el circuito tras la simulación.
- **Problemas:**
  - **Incompletitud:** cualquier circuito requiere para su validación de una colección de estímulos que crece exponencialmente con su complejidad.
  - **Complejidad:** cuanto más precisa sea una simulación más tiempo demanda.
- **Soluciones:**
  - **Métricas de cobertura:** miden la calidad de un subconjunto de estímulos.
    - Cobertura de código (para modelos textuales)
    - Cobertura de datos.
  - **Aceleración hardware:** uso de computadores específicos para ejecutar modelo.
  - **Emulación hardware:** proyección de un modelo sobre hardware reconfigurable
    - Emulación de vectores: se usa un generador de patrones y un analizador lógico
    - Emulación sobre el circuito: el prototipo se integra en el sistema destino
- **Según la precisión con que se trate el tiempo y los dominios de datos encontramos**
  - Simuladores eléctricos, Simuladores lógicos, Simuladores RT



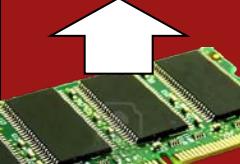
CPU



Cache OnChip



Cache OnBoard



RAM



Disco Duro

#### Simuladores eléctricos (o a nivel de transistor):

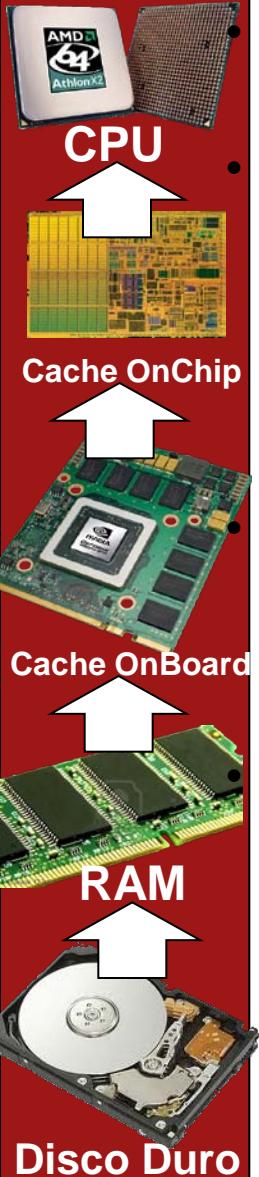
- Hacen un tratamiento continuo del tiempo y de todas las magnitudes físicas del circuito.
- Típicamente toma como entrada una netlist de elementos microelectrónicos (transistores, resistencias, capacidades)
- Está basada en la resolución de matrices de ecuaciones que relacionan los voltajes e intensidades que circulan por cada uno de los elementos del circuito.
- Pueden usar modelos de diferente nivel de complejidad que llevan aparejados diferentes márgenes de error (nunca es cero).
- Las simulaciones son lentas y por tanto sólo son aplicables a pequeñas partes del circuito y a pequeñas ventanas temporales.

#### Simuladores lógicos (o a nivel de puerta):

- Hacen un tratamiento continuo del tiempo, pero abstraen el comportamiento eléctrico del circuito discretizando las magnitudes físicas.
- Toma como entrada una netlist de puertas y biestables.
- Utilizan elaborados modelos de puertas y retardos RC para dar una visión detallada del timing de un circuito, mostrando tanto los estados estables como transitorios de los elementos del sistema.
- Pueden retroanotarse con información extraída de un layout

#### Simuladores RT (o cycle-based):

- Hacen un tratamiento discreto del tiempo y de las magnitudes físicas.
- Toma como entrada una netlist de puertas y biestables.
- Utilizan modelos funcionales de puertas, abstrayendo estados transitorios del sistema
- Sólo pueden usarse para validación funcional



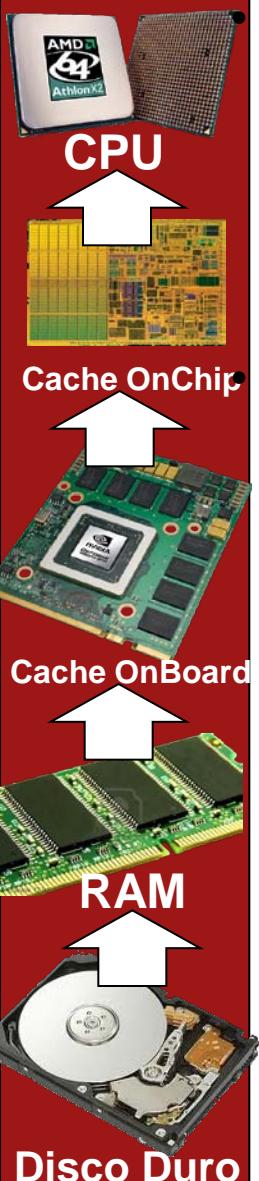
## Testeo

- Una vez fabricado un circuito, y antes de su lanzamiento, debe validarse que funciona adecuadamente (asumiendo que el diseño ha sido correcto).
- Se entiende por testeo al proceso de detección de errores en un circuito resultado de los defectos físicos y de las variaciones del proceso.
- Es similar a la validación por simulación, excepto en que se utiliza un circuito real en lugar de un modelo ejecutable del mismo.
- El diseñador aplica una colección de estímulos de entrada y evalúa las formas de onda que genera el circuito en funcionamiento.

- Presenta problemas de incompletitud y complejidad por lo que para evaluar la calidad de un conjunto de tests, se usan modelos y métricas de cobertura de fallos.
- Simulador de fallos: simulador que intencionadamente inserta fallos en el modelo de un circuito para evaluar la cantidad de ellos detectables por un conjunto de tests.

### Existen dos tipos de testeo

- Funcional: chequea que la conducta del circuito se ajusta a su diseño.
- Paramétrico: chequea aspectos no funcionales tales como: márgenes de ruido, frecuencias de reloj, retardos de propagación, etc. bajo diferentes condiciones de funcionamiento



### Chequeadores de reglas

**Siempre que exista un modelo de un circuito procesable por máquina, es posible chequear que satisface ciertas reglas constructivas**

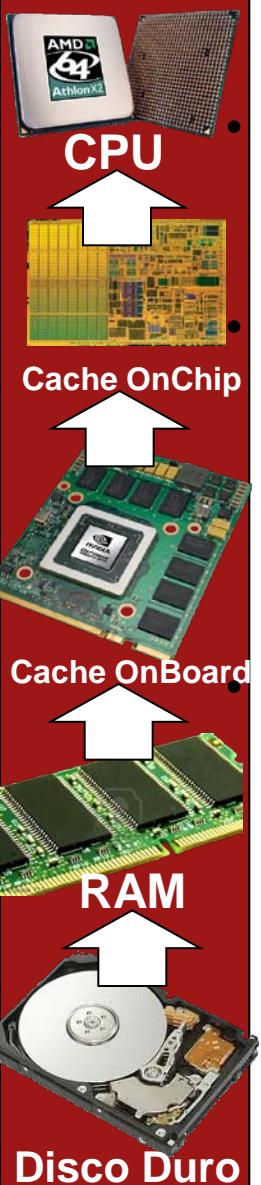
- La conformidad de dichas reglas no asegura la corrección, pero una violación de las mismas es síntoma de error.
- Son especialmente útiles para chequear diseños editados a mano, las herramientas automáticas los llevan implícitamente incorporados (corrección por construcción).

### Ejemplos:

- DRC (Design Rule Checkers): chequeadores de geometría y topología.
- Chequeadores de conectividad en esquemáticos.
- Chequeadores de estilo (para modelos textuales).

### Analizadores

- ☒ **Evalúan la calidad de un circuito mediante la extracción de medidas de rendimiento.**
  - **Estáticos:** mediante la observación de su estructura.
  - **Dinámicos:** mediante el estudio de la respuesta a una colección de estímulos
- ☒ **Analizadores de timing:** chequean que un circuito satisface las restricciones derivadas de una cierta estrategia de temporización (por flanco, por nivel, etc.)
- **Analizadores de potencia:** chequean que un circuito satisface ciertas restricciones de consumo.



## Verificación formal

**Son métodos matemáticos de validación funcional que establecen sus resultados de una manera independiente a estímulos.**

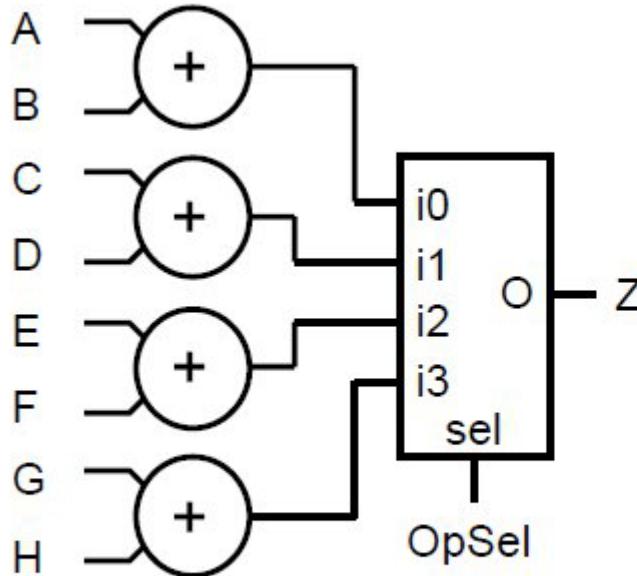
- Resuelven el problema de la incompletitud de los métodos basados en simulación.
- **Chequeadores de equivalencia:** demuestran que dos modelos (de diferentes dominios o nivel de abstracción) son funcionalmente equivalentes.
- Equivalencia estricta: tienen igual pinout y responderían igual a la misma secuencia de estímulos
- Consistencia: son equivalentes para un subconjunto del pinout, o son equivalentes para un subconjunto de los posibles estímulos y/o de las respuestas.

**Chequeadores de propiedades:** demuestran que un modelo posee una cierta característica funcional.

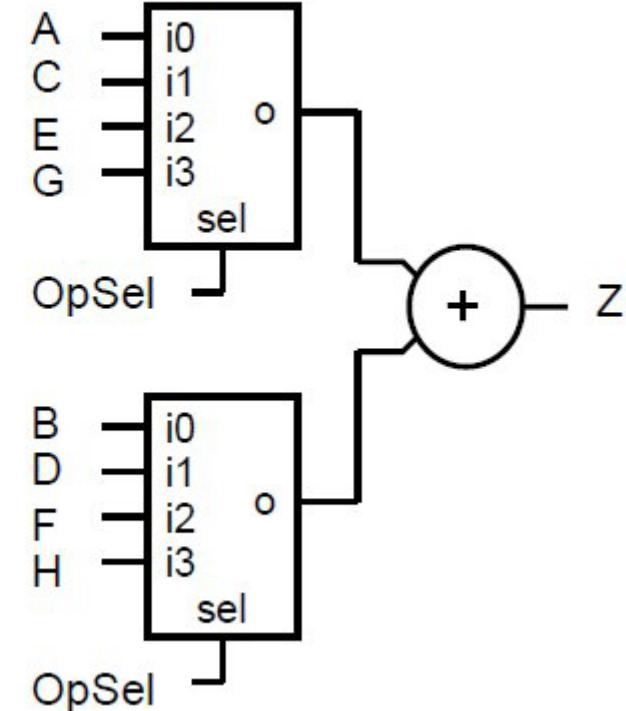


## DISEÑO de circuitos a nivel RT:

As Specified:



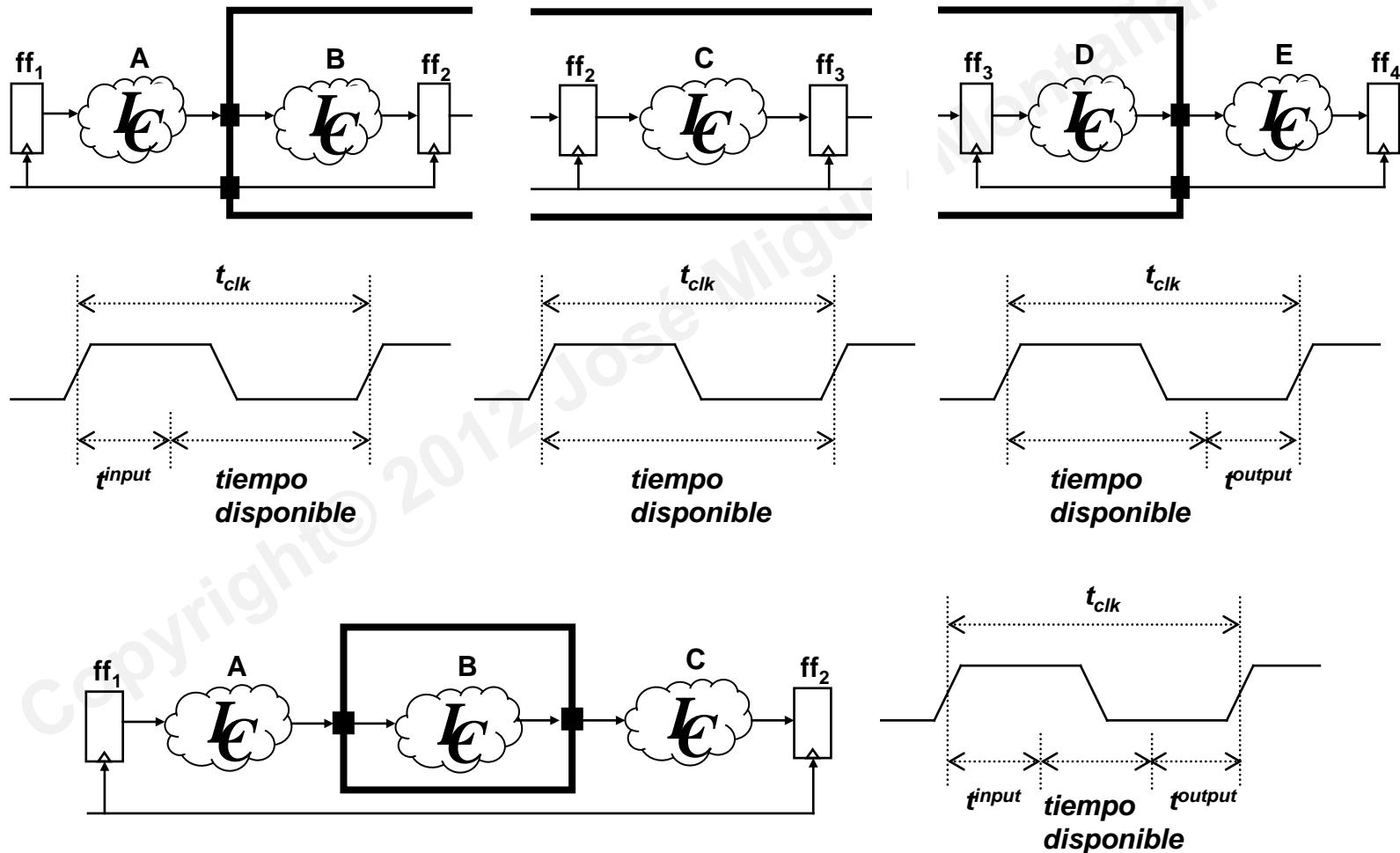
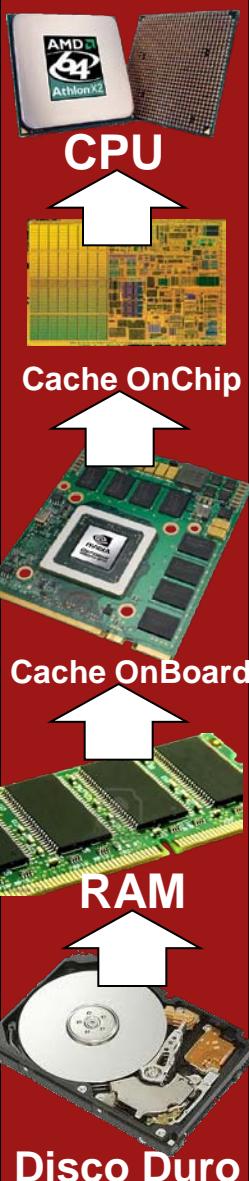
Optimal results:

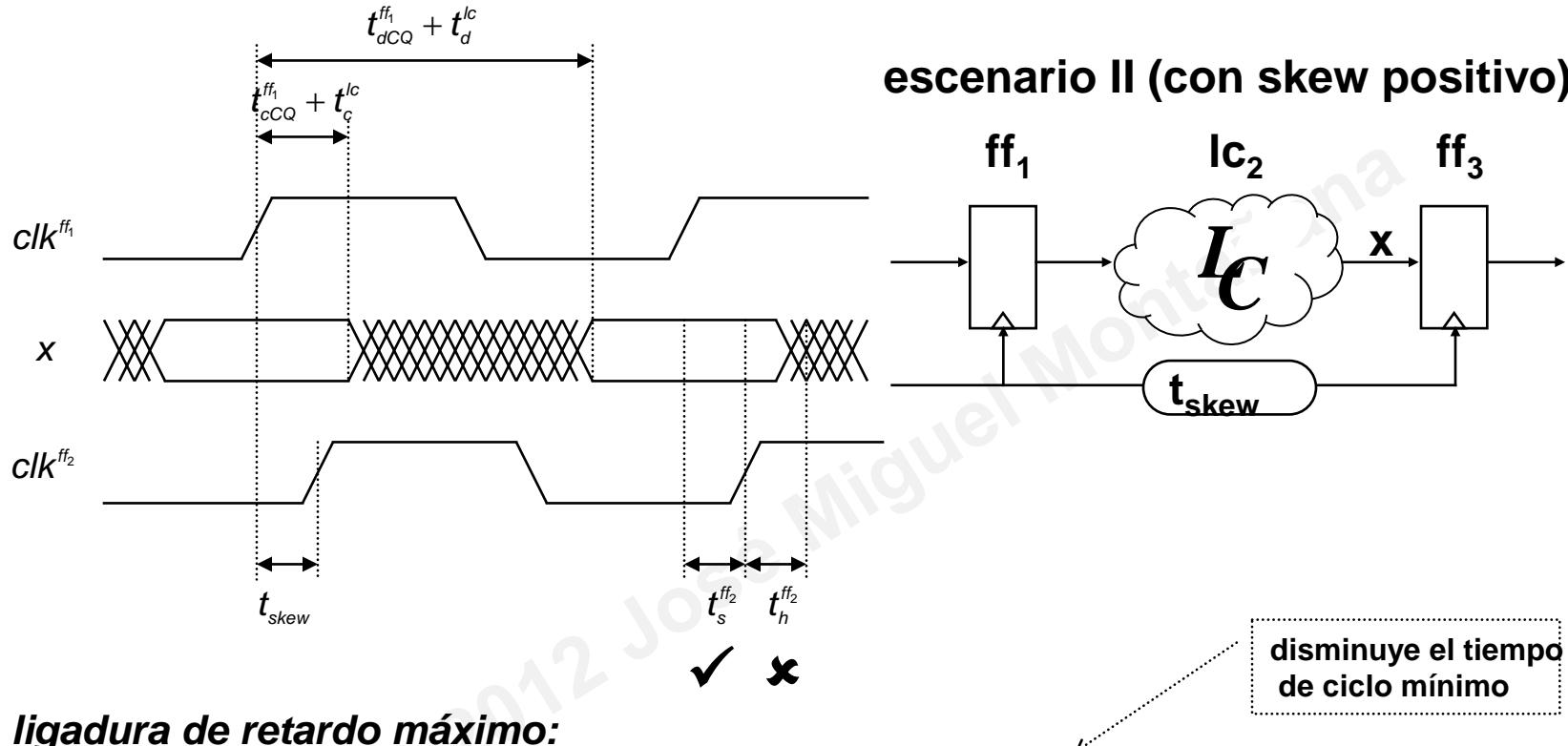
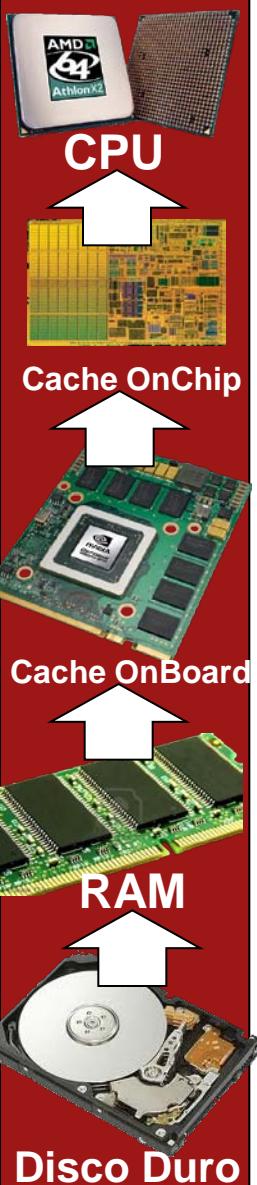


# DISEÑO AUTOMÁTICO DE SISTEMAS

## Curso 2013-2014

Diseño  
a nivel RT





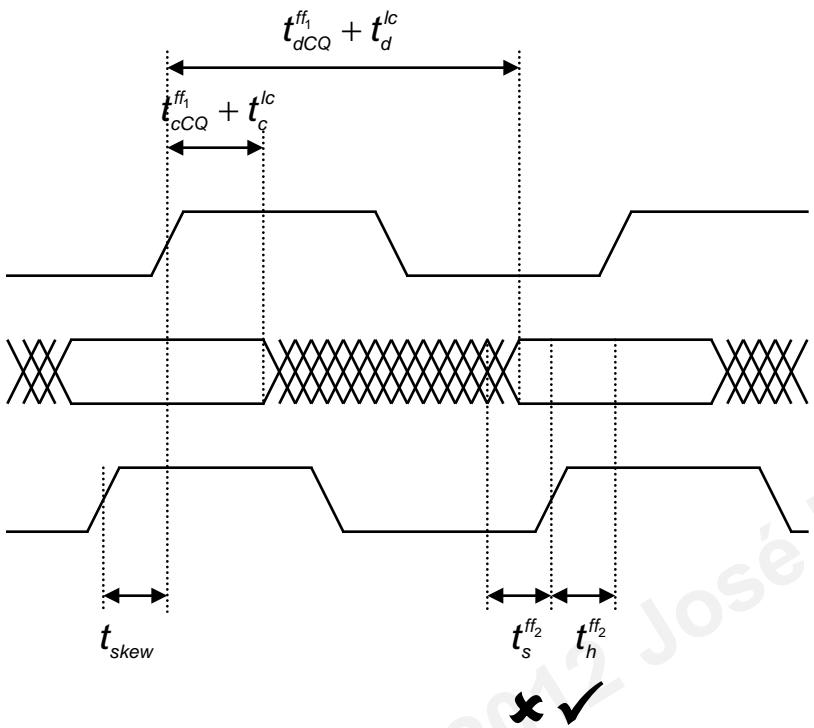
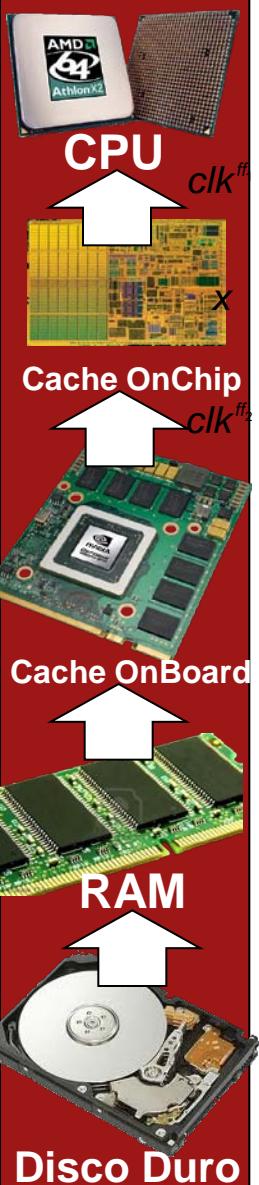
**ligadura de retardo máximo:**

$$t_{CLK} \geq (t_{dCQ}^{ff_1} + t_d^{ff_1} + t_c^{ff_2} + t_s^{ff_2} - t_{skew}) \Rightarrow (t_{CLK} - t_d^{ff_1}) \geq (t_{dCQ}^{ff_1} + t_s^{ff_2} - t_{skew})$$

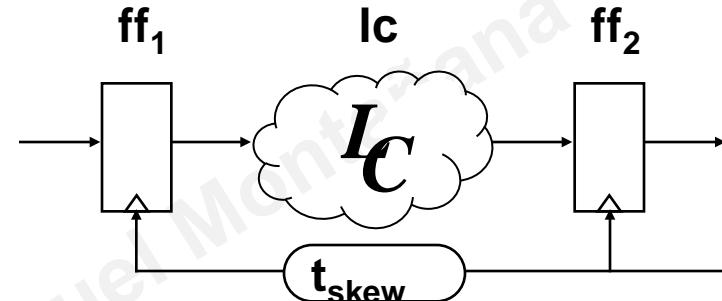
**ligadura de retardo mínimo:**

$$(t_{dCQ}^{ff_1} + t_d^{ff_1} + t_c^{ff_2}) \geq t_h^{ff_2} + t_{skew} \Rightarrow t_c^{ff_2} \geq (t_h^{ff_2} - t_{dCQ}^{ff_1} + t_{skew})$$

aparecen problemas, además  
 si  $t_{skew} > (t_{dCQ}^{ff_1} + t_d^{ff_1} + t_c^{ff_2})$  el sistema se desincroniza



escenario III (con skew negativo)



aumenta el tiempo de ciclo mínimo

**ligadura de retardo máximo:**

$$t_{CLK} \geq (t_{dCQ}^{ff_1} + t_c^{lc} + t_s^{ff_2} + t_{skew}) \Rightarrow (t_{CLK} - t_c^{lc}) \geq (t_{dCQ}^{ff_1} + t_s^{ff_2} + t_{skew})$$

**ligadura de retardo mínimo:**

$$(t_{cCQ}^{ff_1} + t_c^{lc}) \geq t_h^{ff_2} - t_{skew} \Rightarrow t_c^{lc} \geq (t_h^{ff_2} - t_{cCQ}^{ff_1} - t_{skew})$$

no aparecen problemas, sin embargo si el skew es muy grande, el sistema también se desincroniza



### Cuando el skew es positivo:

- puede aprovecharse para aumentar la frecuencia de funcionamiento.
- el circuito puede funcionar mal independientemente a la frecuencia de funcionamiento, y en función de las condiciones del entorno (temperatura).

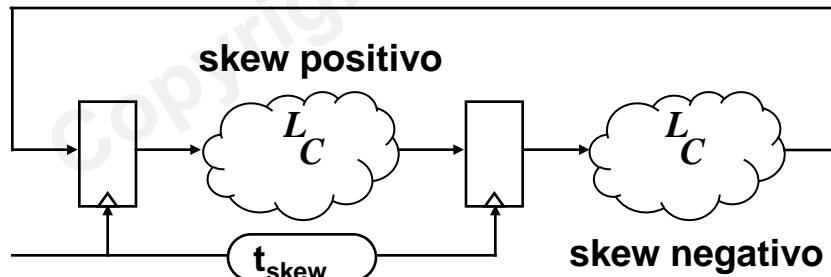
### Cuando el skew es negativo:

- el circuito podrá funcionar correctamente pero a una frecuencia de reloj menor.

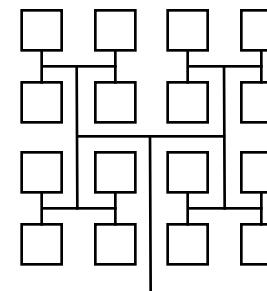
**En un circuito general conviven zonas con skew negativo y zonas con skew positivo, por lo que se debe diseñar según las reglas más restrictivas de cada caso.**

### Soluciones del skew:

- Utilizarlo en propio provecho (sólo aplicable durante la síntesis física).
- Diseñar redes especiales de distribución con skew acotado (árbol en H, árbol jerárquico)
- Usar circuitos no lineales de compensación (temporización de lazo cerrado)



**árbol en H**



**árbol jerárquico**

