

Sistemas Digitales

Laboratorio

Semestre 2022-1

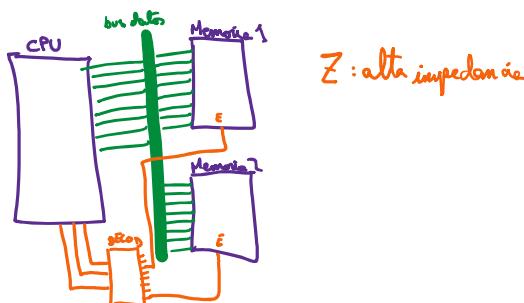
Sesión Laboratorio Semana 6

Profesor: Kalun José Lau Gan

1

Preguntas previas: (10minutos)

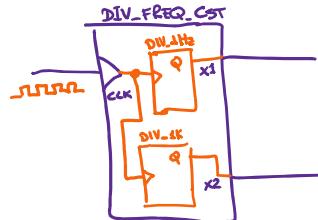
- La entrada de RESET en un circuito digital es síncrono o asíncrono?
 - Siempre es asíncrono por naturaleza
- En una máquina de estados podemos tener contemplado un solo estado?
 - No! Como mínimo dos estados.
- ¿Algún caso práctico para emplear los otros estados lógicos del IEEE1164?



2

Preguntas previas: (10minutos)

- Divisor con dos salidas de frecuencia distintas?



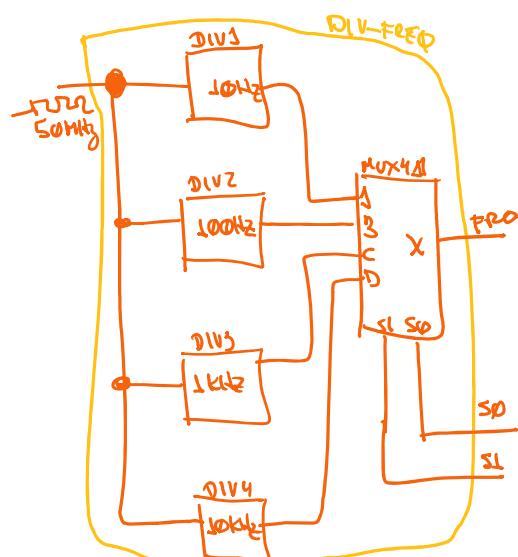
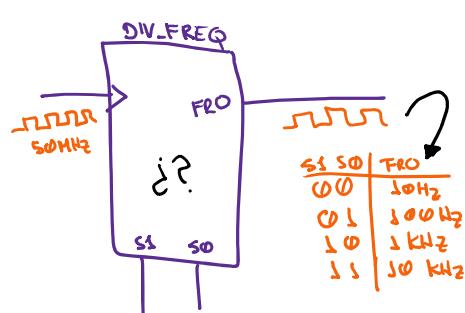
- Señal interna en VHDL":

PC1_REG						
1	0	1	0	1	1	0
bit7	bit6	bit5	bit4	bit3	bit2	bit1

signal PC1_REG : std_logic_vector(7 downTo 1) := "101000";

3

Diseño de un divisor de frecuencia con salida seleccionable de cuatro frecuencias diferentes



4

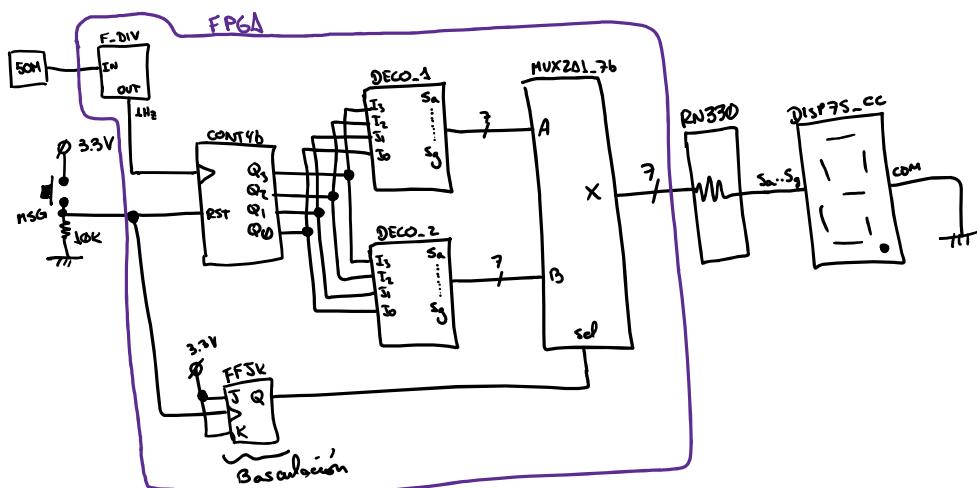
Aplicación práctica para un divisor de frecuencia?

#	1	2
1	LA	220
2	LA#	233.08
3	SI	246.94
4	DO	261.63
5	DO#	277.18
6	RE	293.66
7	RE#	311.13
8	MI	329.63
9	FA	349.23
10	FA#	369.99
11	SOL	392.00
12	SOL#	415.3
		440.0
		466.16
		493.88
		523.25
		554.37
		587.33
		622.25
		659.26
		698.46
		739.99
		783.99
		830.61

5

Preguntas previas:

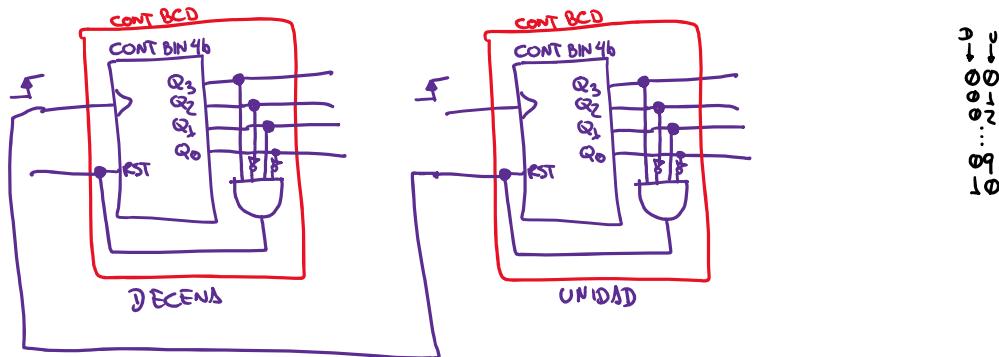
- ¿Cómo hago para que en el ejemplo de visualización de mensaje visto en sesiones anteriores que al cambiar de mensaje a visualizer se inicie la visualización desde la primera letra?



6

Preguntas previas:

- Contador BCD 00-99

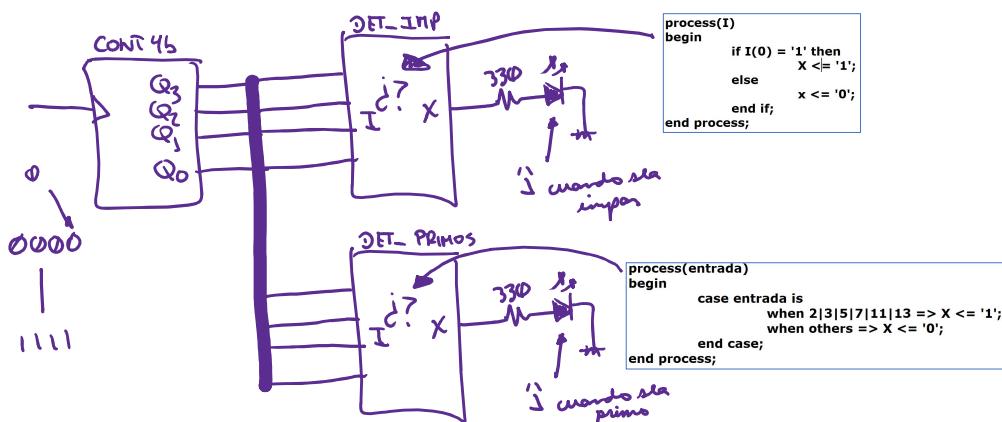


- Contador BCD 00-23

7

Preguntas previas:

- Declaración de relojes:
 - Flanco positivo: rising_edge(clk) / clk='1' and clk'event
 - Flanco negativo: falling_edge(clk) / clk='0' and clk'event
- ¿Cómo hago para detectar si un número es impar o primo?



8

Preguntas previas:

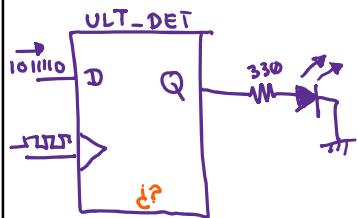
9

Agenda:

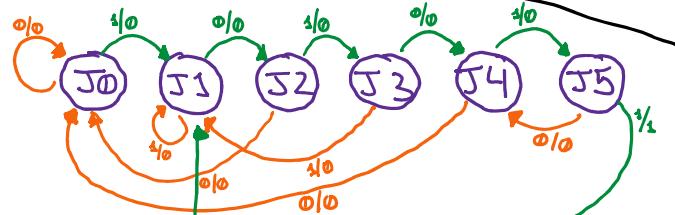
- Ejemplo de Máquinas de estado en VHDL (Mealy)
- Memorias

10

Ejemplo: Detector de secuencia \rightarrow 101011 en MEF Mealy



a) Diagrama de estados:

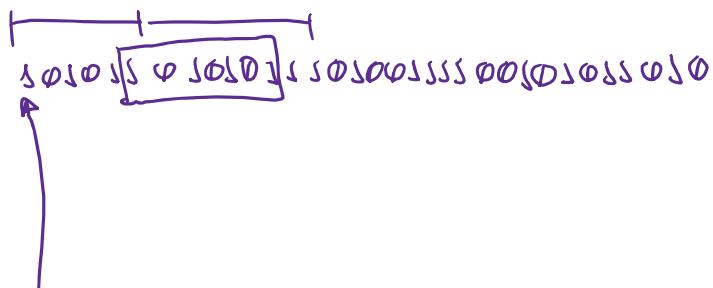


b) Tabla de transiciones:

D	E.P.	E.S.	Q
Θ	JØ	JØ	ΘΘ
₁		J₂	ΘΘ
₂	J₁	J₁	ΘΘ
₃	J₂	J₂	ΘΘ
₄	J₃	J₄	ΘΘΘΘ
₅	J₄	J₅	ΘΘΘΘΘΘ
₆	J₅	J₄	Θ₁

11

Sobre traslape o overlap:



12

Ejemplo: Detector de secuencia ->101011 en MEF Mealy

```

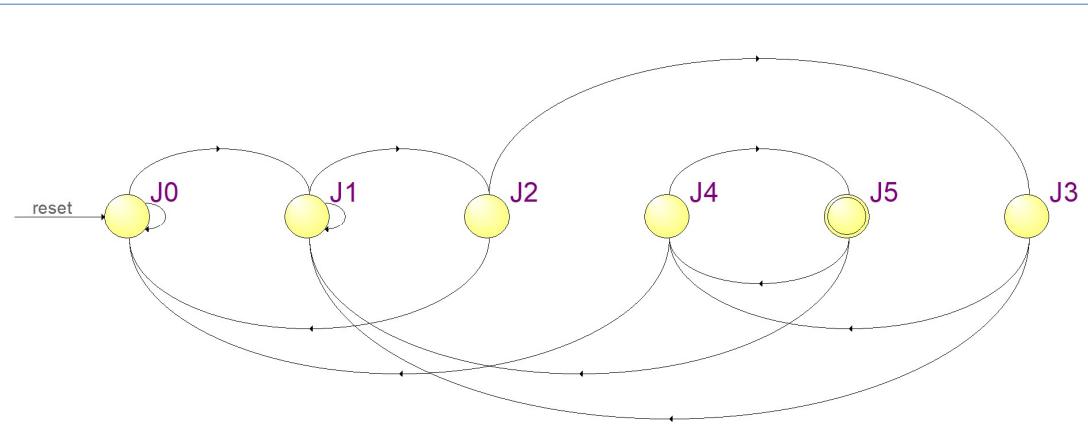
1 library IEEE;
2 use IEEE.std_logic_1164.all;
3 use IEEE.std_logic_arith.all;
4 use IEEE.std_logic_unsigned.all;
5
6 entity el_ultimo is
7 port ( clk, D: in std_logic;
8         Q: out std_logic);
9 end el_ultimo;
10
11 architecture flujo of el_ultimo is
12
13 TYPE estados is (J0, J1, J2, J3, J4, J5);
14 SIGNAL es_p: estados;
15
16 begin
17 process (clk)
18 begin
19 if rising_edge(clk) then
20 case es_p is
21 when J0 =>
22 if D = '0' then
23 es_p <= J0;
24 else
25 es_p <= J1;
26 end if;
27 when J1 =>
28 if D = '0' then
29 es_p <= J2;
30 else
31 es_p <= J1;
32 end if;
33 when J2 =>
34 if D = '0' then
35 es_p <= J0;
36 else
37 es_p <= J3;
38 end if;
39 when J3 =>
40 if D = '0' then
41 es_p <= J4;
42 else
43 es_p <= J1;
44 end if;
45 when J4 =>
46 if D = '0' then
47 es_p <= J0;
48 else
49 es_p <= J5;
50 end if;
51 when J5 =>
52 if D = '0' then
53 es_p <= J4;
54 else
55 es_p <= J1;
56 end if;
57 when others =>
58 es_p <= J0;
59 end case;
60 end if;
61 end process;
62
63 process(es_p, D)
64 begin
65 if es_p = J5 then
66 Q <= D;
67 else
68 Q <= '0';
69 end if;
70 end process;
71 end flujo;

```

D	E.P.	E.S.	Q
0	J0	J0	0
1	J0	J1	0
0	J1	J1	0
1	J1	J2	0
0	J2	J2	0
1	J2	J3	0
0	J3	J4	0
1	J3	J3	0
0	J4	J4	0
1	J4	J5	0
0	J5	J5	0

13

Ejemplo: Detector de secuencia ->101011 en MEF Mealy



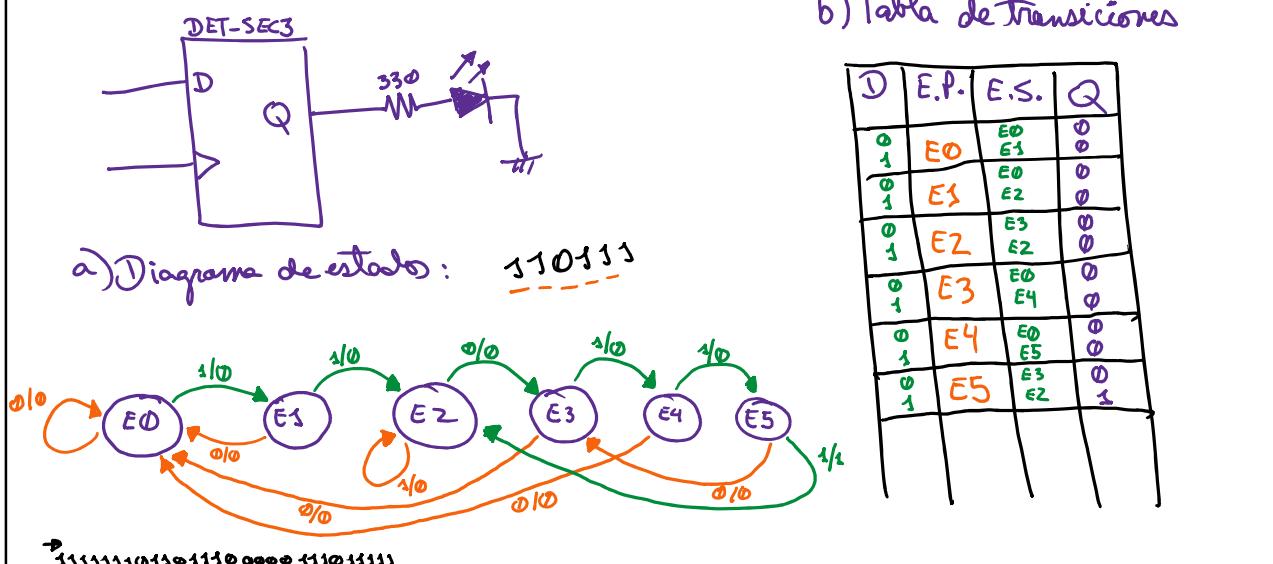
14

Ejemplo: Detector de secuencia $\rightarrow 101011$ en MEF Mealy



15

Ejemplo: Detector de secuencia $\rightarrow 110111$ con MEF Mealy



16

Ejemplo: Detector de secuencia ->110111 con MEF Mealy

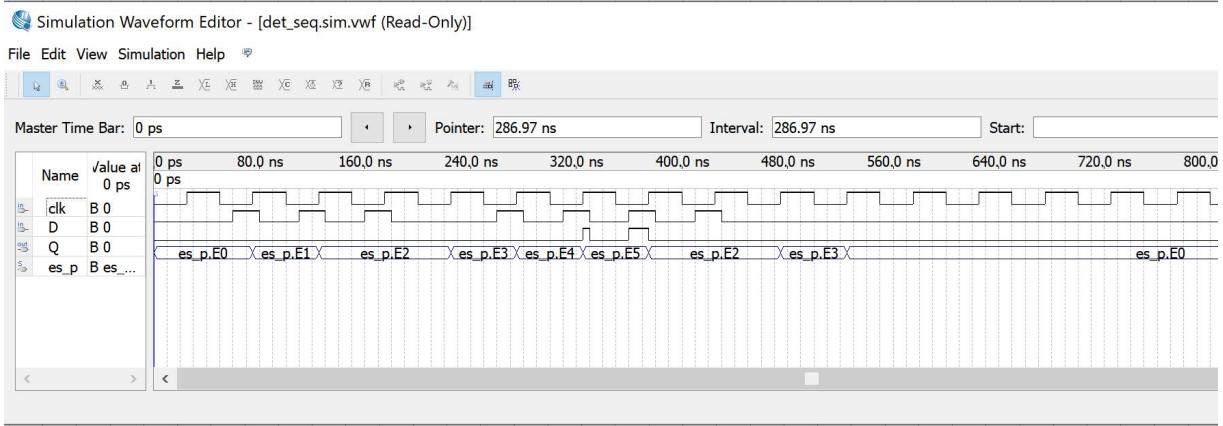
```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3 use IEEE.std_logic_arith.all;
4 use IEEE.std_logic_unsigned.all;
5
6 entity det_seq is
7   port ( clk, D: in std_logic;
8         Q: out std_logic);
9 end det_seq;
10
11 architecture flujo of det_seq is
12
13 TYPE estados is (E0, E1, E2, E3, E4, E5);
14 SIGNAL es_p: estados;
15
16 begin
17   process (clk)
18   begin
19     if rising_edge(clk) then
20       case es_p is
21         when E0 =>
22           if D = '0' then
23             es_p <= E0;
24           else
25             es_p <= E1;
26           end if;
27         when E1 =>
28           if D = '0' then
29             es_p <= E0;
30           else
31             es_p <= E2;
32           end if;
33         when E2 =>
34           if D = '0' then
35             es_p <= E3;
36           else
37             es_p <= E2;
38           end if;
39         when E3 =>
40           if D = '0' then
41             es_p <= E0;
42           else
43             es_p <= E4;
44           end if;
45         when E4 =>
46           if D = '0' then
47             es_p <= E0;
48           else
49             es_p <= E5;
50           end if;
51         when E5 =>
52           if D = '0' then
53             es_p <= E3;
54           else
55             es_p <= E2;
56           end if;
57         when others =>
58           es_p <= E0;
59       end case;
60     end if;
61   end process;
62
63   process(D)
64   begin
65     if es_p = E5 then
66       Q <= D;
67     else
68       Q <= '0';
69     end if;
70   end process;
71 end flujo;
72
73
74

```

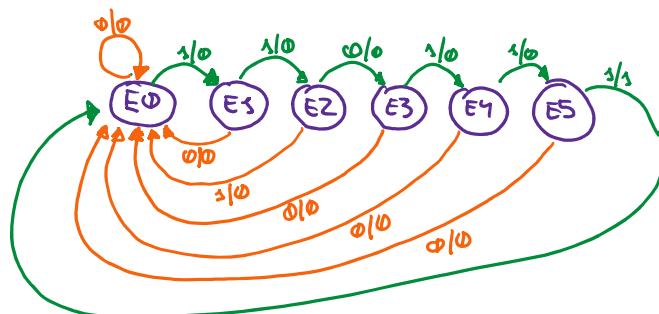
17

Resultado de la simulación VWF



18

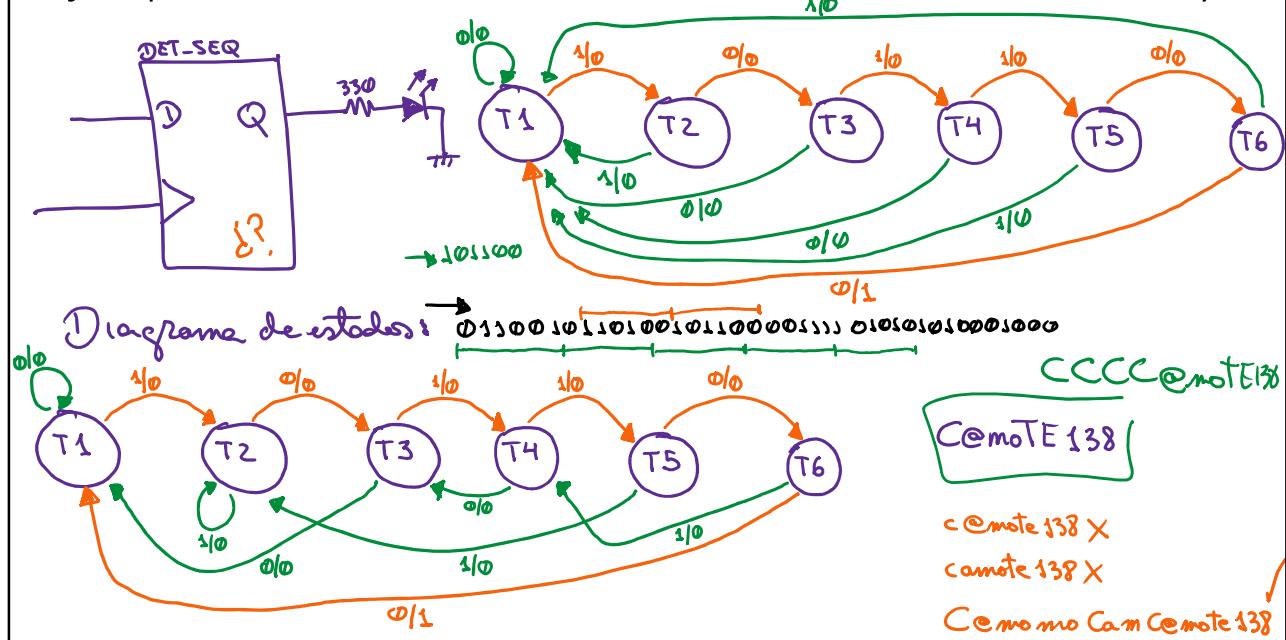
Ejemplo: Detector de clave ->110111 con MEF Mealy (nótese no hay traslape)



Ingrese password:

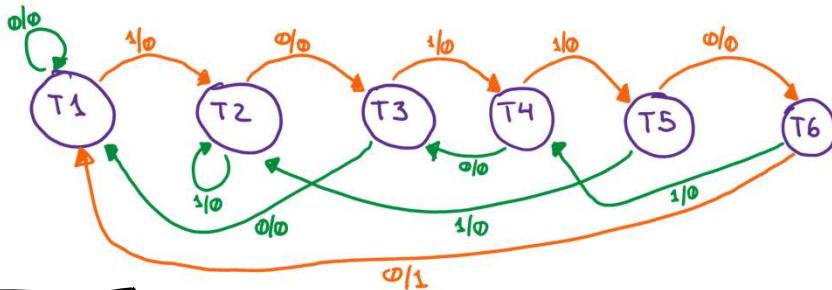
19

Ejemplo: Detector de secuencia ->101100 con MEF Mealy



20

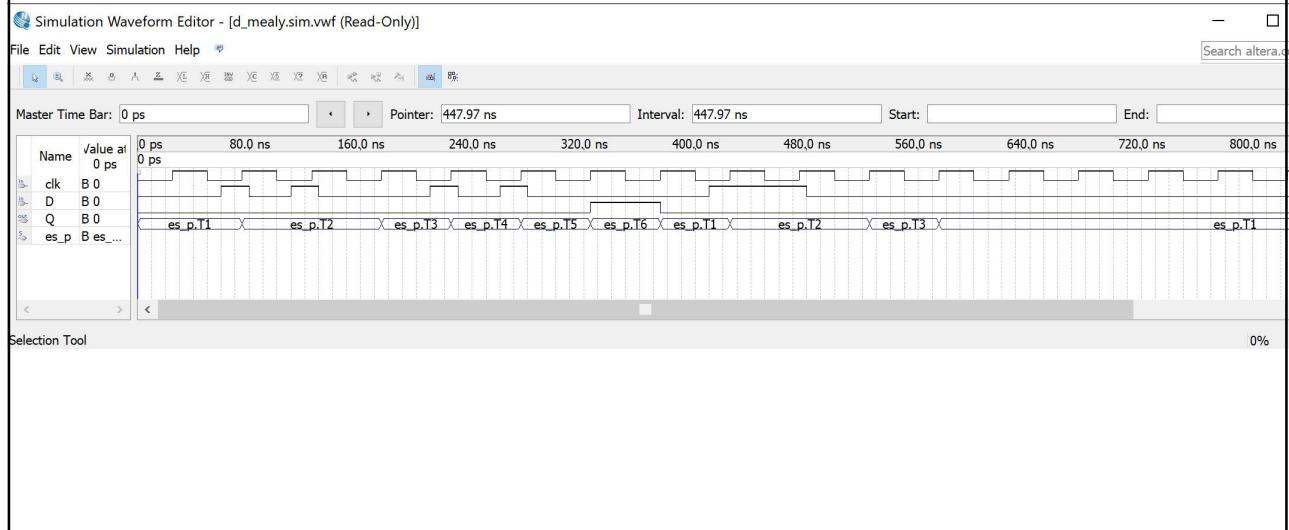
Desarrollo de la tabla de transiciones



D	E.P.	E.S.	Q
0	T1	T1	0
1		T2	0
0	T2	T3	0
1		T2	0
0	T3	T4	0
1		T4	0
0	T4	T5	0
1		T5	0
0	T5	T6	0
1		T2	0
0	T6	T1	1
1		T4	0

21

Resultado de la simulación VWF



22

Detector de secuencia $\rightarrow 011010$ empleando MEF Mealy

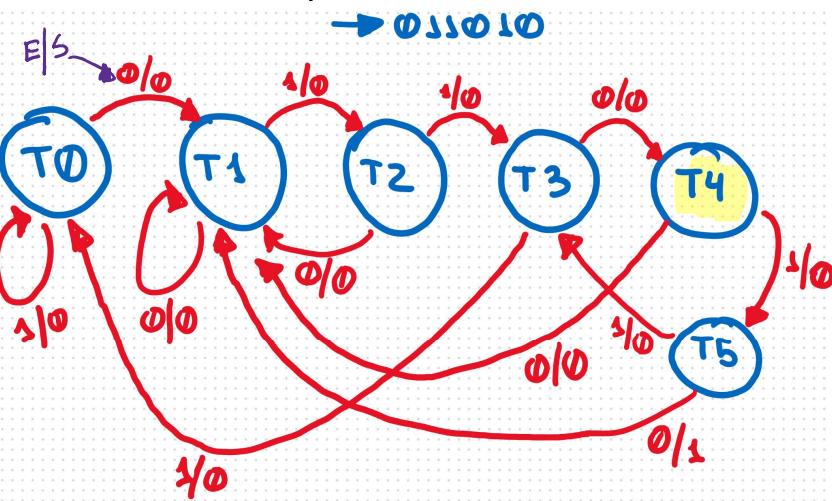
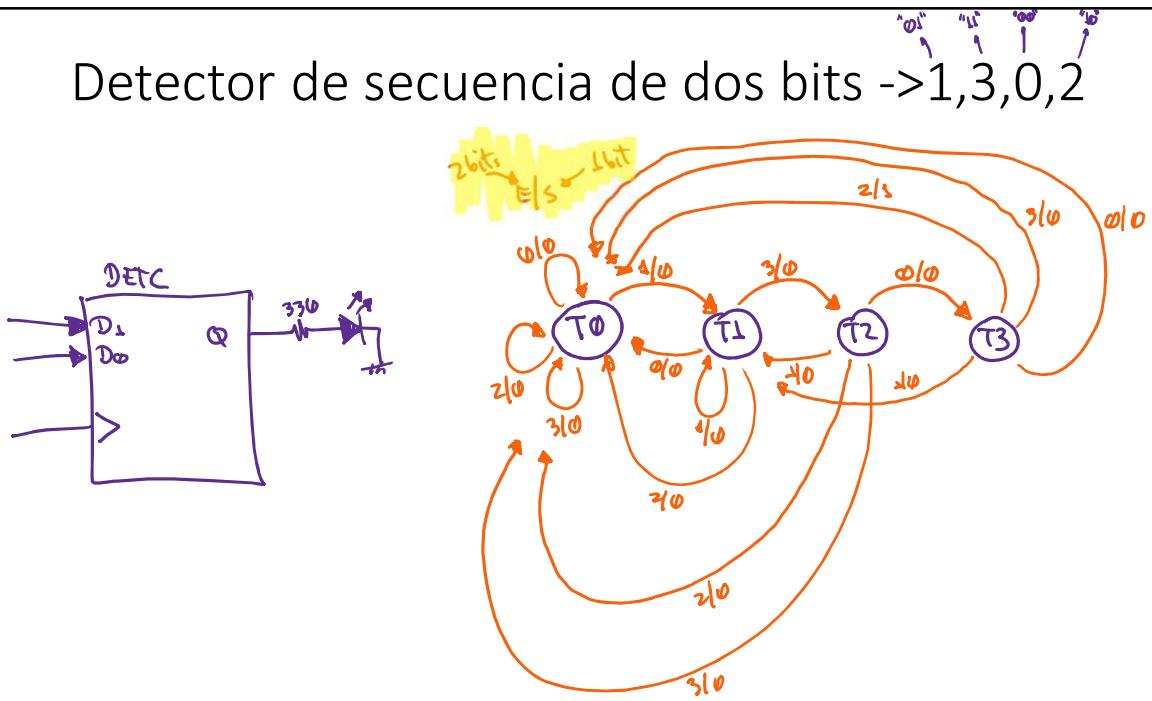


Tabla de transiciones

E	E.P.	E.S	S
0	T0	T1	0
1	T0	T2	0
0	T1	T2	0
1	T1	T3	0
0	T2	T3	0
1	T2	T4	0
0	T3	T4	0
1	T3	T5	0
0	T4	T5	0
1	T4	T0	1
0	T5	T0	1
1	T5	T1	0

23

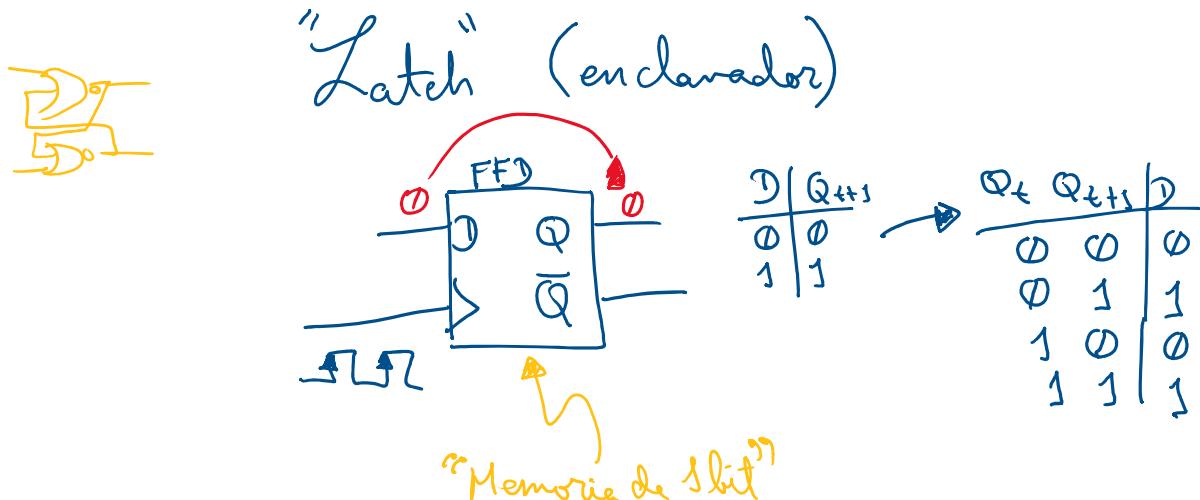
Detector de secuencia de dos bits $\rightarrow 1,3,0,2$



24

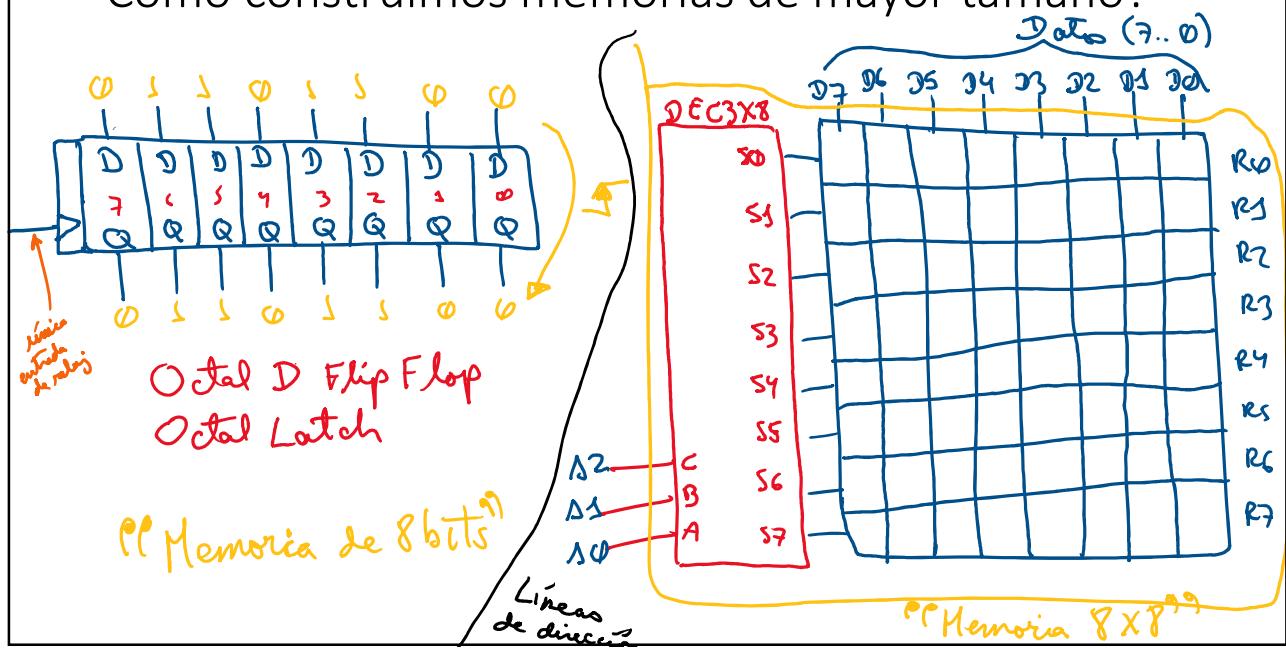
Memorias

- Las memorias son dispositivos que almacenan información



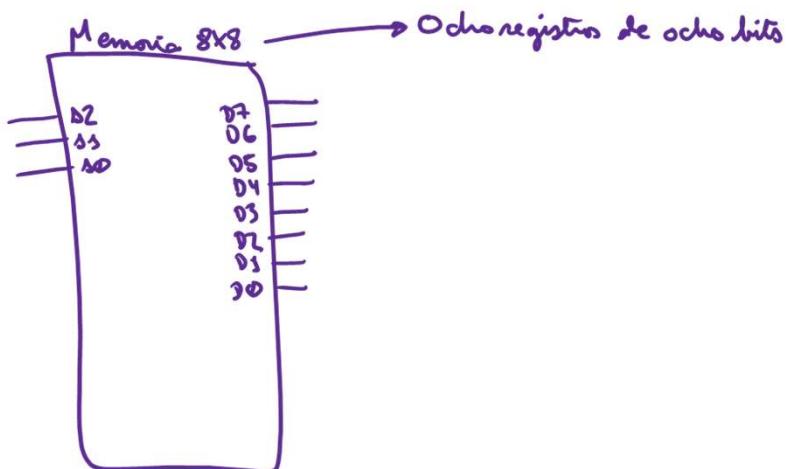
25

Cómo construimos memorias de mayor tamaño?



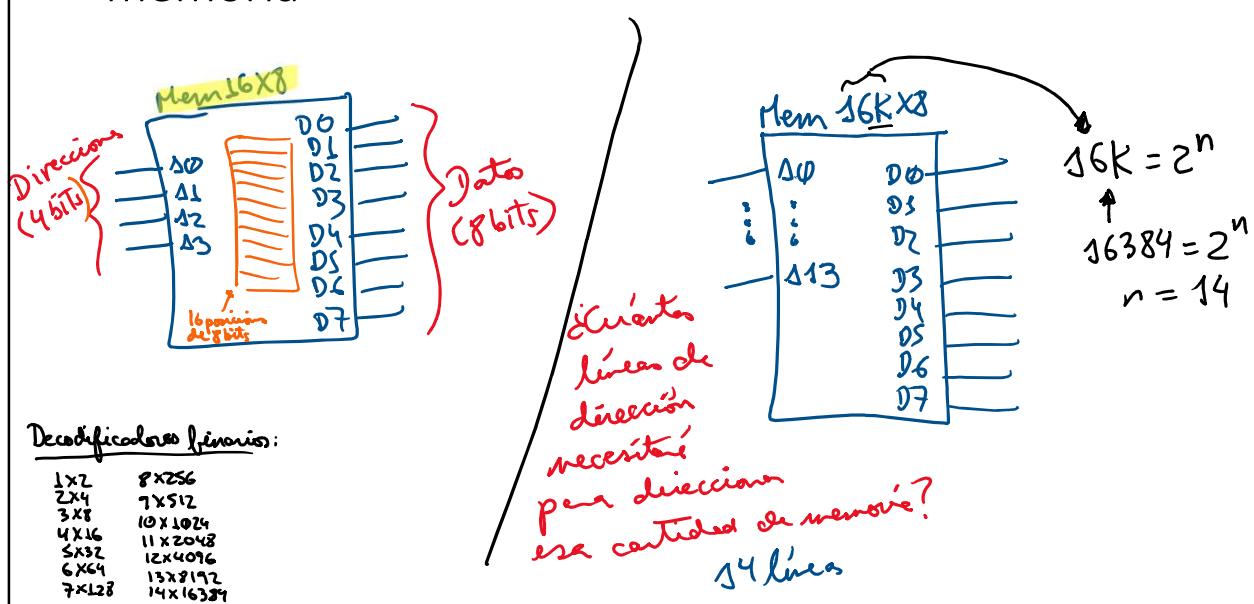
26

(cont...)



27

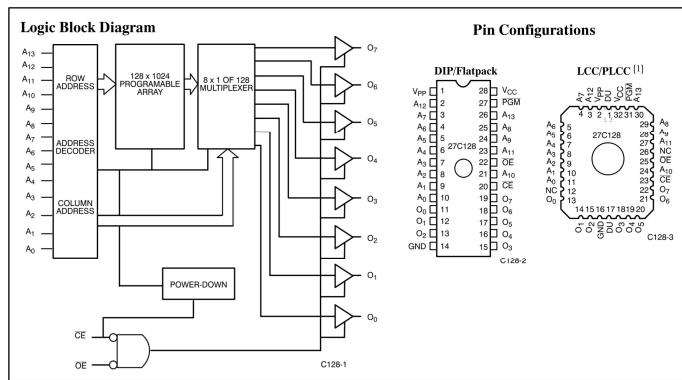
Relación entre líneas de dirección y tamaño de la memoria



28

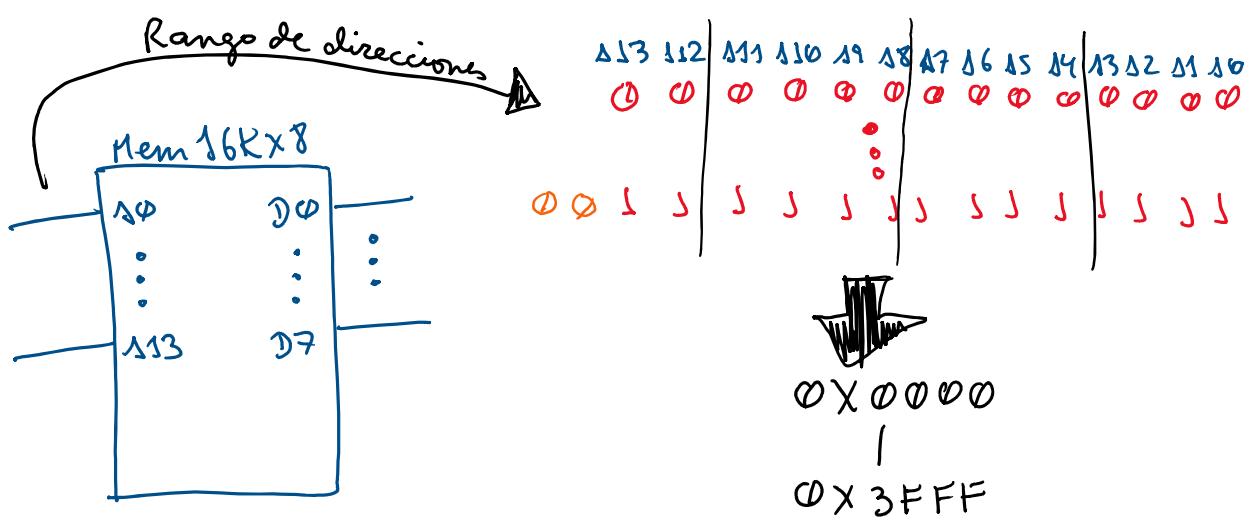
Memoria comercial EEPROM 27C128

- Es una memoria de 128Kbit (16K x 8bit)
- Tiene 14 líneas de dirección



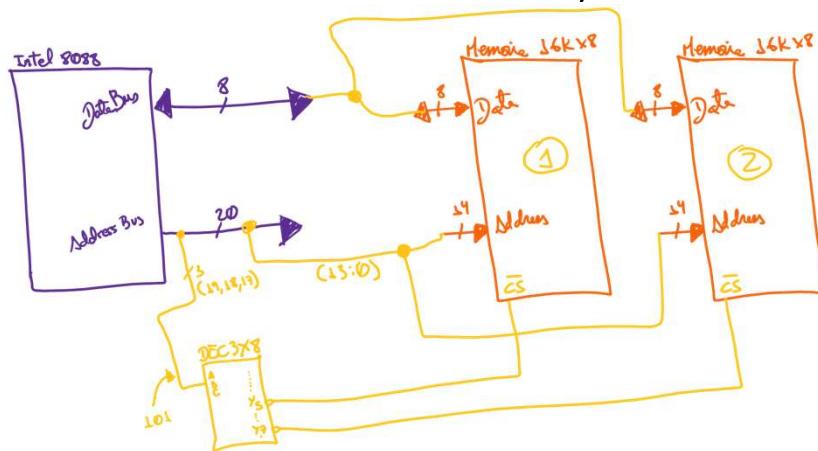
29

En la memoria anterior de 16Kx8:



30

Mapeo de direcciones con un 8086 y una memoria 16Kx8



	Líneas de dirección																Rango
1	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	0xA0000
1	1	0	1	x	x	x	0	0	0	0	0	0	0	0	0	0	0xA3FFF
2	1	0	1	x	x	x	1	1	1	1	1	1	1	1	1	1	0xE0000
2	1	1	1	x	x	x	0	0	0	0	0	0	0	0	0	0	0xE3FFF

31

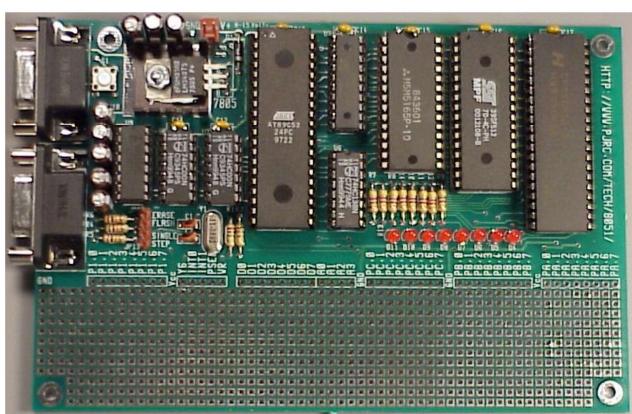
Analizando lo siguiente:

https://www.pjrc.com/tech/8051/board3/board_image.html
New Rev 4 Board

8051 Development Board: What It Does

The 8051 development board provides an easy-to-use and low-cost way to develop your 8051 based microcontroller projects, with equipment, such as IC programmers or emulators. See below for a [detailed list of the boards features](#).

Close-Up View of the Development Board



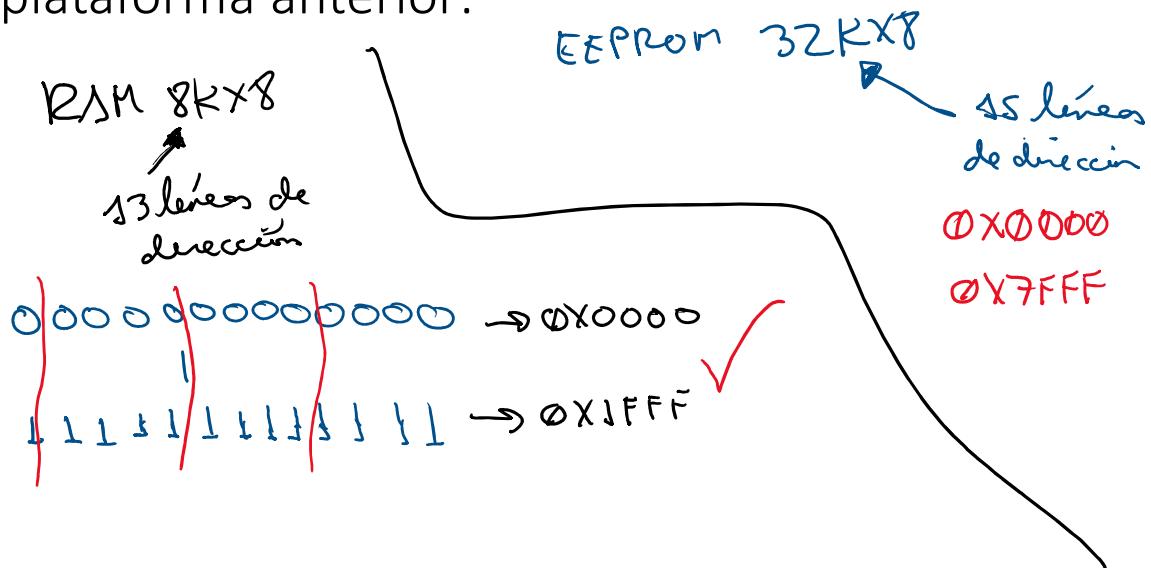
- Tarjeta embebida basada en el 8051.
- Tiene Memoria RAM M5M5165P (8Kx8)
- Tiene Memoria EEPROM (32Kx8)
- Tiene interface periférica programable (PPI) 82C55
- Precursor de los actuales microcontroladores "all-in-one"

Memory Map

0000 - 1FFF	PAULMON2 Monitor
2000 - 3FFF	SRAM
4000 - 5FFF	82C55 I/O Chip
4000: 4001	Port A (read or write)
4001: 4002	Port B (read or write)
4002: 4003	Port C (read or write)
4003: 4004	Configure (write only)
6000 - 7FFF	User Expansion (Y6 signal assert low)
8000 - FFFF	Flash ROM

32

Verificación de mapeo de memoria de la plataforma anterior:



33

Preguntas sobre memorias:

- Si la memoria EEPROM es no volátil. ¿Por qué se sigue usando memoria RAM si éste al quitarle la alimentación se borra lo que grabó?
 - La EEPROM es lenta (en los eventos de escritura) a comparación de la RAM que tiene tiempos de acceso más cortos.
- Hay que tener en cuenta la jerarquía de sistemas de almacenamiento

34

Tecnologías de Memorias

- ROM (Read Only Memory – Memoria de Sólo Lectura)
 - Éstas memorias vienen programadas de fábrica, no se altera el contenido en cortes de alimentación.
 - No es posible borrar ni modificar el contenido.

35

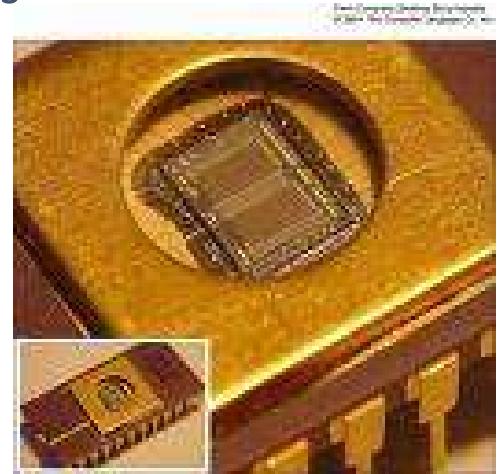
Tecnologías de Memorias

- OTP (One Time Programmable – Memoria programable una única vez)
 - También conocidas como PROM
 - Éstas memorias sólo pueden ser grabadas una sola vez.
 - No es posible borrar ni modificar el contenido después de haberla grabado.

36

Tecnologías de Memorias

- **EPROM (Erasable Programmable Read Only Memory)**
 - Para programarlas hay que borrarlas, generalmente son las memorias con una ventana en la cara superior en donde se aplica radiación UV para el proceso de borrado.
 - Luego de programadas, su contenido no se borra si se le corta la alimentación.



37

Tecnologías de Memorias

- **EEPROM (Electrically Erasable Programmable Read Only Memory)**
 - Éstas se borran de manera eléctrica, es decir, aplicando un voltaje más elevado que el voltaje del nivel lógico y en un pin especial.
 - Luego de programadas, su contenido no se borra si se le corta la alimentación.

38

Tecnologías de Memorias

- Flash EEPROM (Electrically Erasable Programmable Read Only Memory)
 - Los eventos de borrado/escritura son más lentos.
 - Son mucho más rápidas que las EEPROM convencionales.
 - De mucha mayor capacidad.

39

Tecnologías de Memorias

- RAM (Random Access Memory – Memoria de Acceso Aleatorio)
 - Son volátiles (al desconectarle la alimentación se pierde la información que pueda haber contenido).
 - Son sumamente rápidas tanto en eventos de lectura, borrado como en eventos de escritura.

40

Tecnologías de Memorias

- SRAM (Static Random Access Memory)
 - Son volátiles (al desconectarle la alimentación se pierde la información que pueda haber contenido).
 - Poseen un costo muy elevado
 - Empleado para memoria CACHÉ y registros temporales.

41

Tecnologías de Memorias

- DRAM (Dynamic Random Access Memory)
 - Son volátiles (al desconectarle la alimentación se pierde la información que pueda haber contenido).
 - Menor costo que las SRAM.
 - Requieren ser “refrescadas” para que puedan almacenar y retener información
 - De grandes capacidades.

42

Ubicación de la Memoria en una PC

- CPU
 - La que se encuentra dentro del procesador
 - La de acceso más rápido
- Memoria Interna
 - Almacenamiento de mayor capacidad que la del CPU (memoria principal)
- Memoria Externa
 - Almacenamiento de mucha mayor capacidad (secundaria)

43

Método de Acceso de las Memorias

- Acceso Secuencial
 - Para acceder a una posición debe de ir trasladándose desde la posición actual a la deseada, pasando y obviando cada registro intermedio.
 - Método empleado en los “Tape Backup”.
- Acceso Directo
 - Se organiza en bloques para buscar más rápido un registro, una vez hallado el bloque en donde se encuentra la información solicitada, se emplea el acceso secuencial para hallarla.
 - Método empleado en las unidades de disco.

44

Método de Acceso de las Memorias

- Acceso Aleatorio (random)
 - Cada posición tiene un mecanismo de acceso cableado físicamente, el tiempo de acceso a una posición es constante.
 - Método empleado en la memoria principal
- Asociativa
 - Similar como la de acceso aleatorio, posee un mecanismo de comparación según valores dados para recuperar la información basándose en una porción de su contenido en lugar de su dirección.
 - Método empleado en las memorias caché.

45

Desempeño de las Memorias

Tiempo de acceso:

- Es el tiempo entre la asignación de memoria y la obtención de un dato válido.

Ciclo de la Memoria:

- Es el tiempo que la memoria requiere para "recuperarse" antes de un nuevo evento de acceso.

Tasa de Transferencia:

- Cuanta información se pueda mover

46

Jerarquía de las Memorias en una PC



- Registros
- Caché Nivel 1
- Caché Nivel 2
- Memoria Principal
- Caché de Disco Duro
- Disco Duro
- Discos Ópticos
- Cintas Tape Backup

47

Jerarquía de las Memorias en una PC

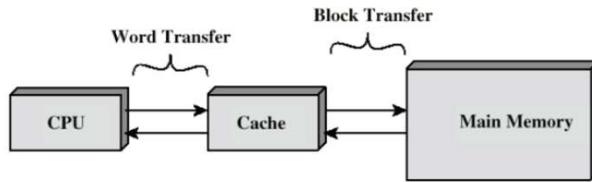
● Relaciones:

- A menor tiempo de acceso, mayor coste por bit.
- A mayor capacidad, menor coste por bit
- A mayor capacidad, mayor tiempo de acceso

48

Memoria CACHÉ

- Basado en memoria del tipo SRAM
- En pequeña cantidad
- Se encuentra entre la memoria principal y el µP
- Puede estar embebida en el mismo µP o de manera externa



49

Funcionamiento de la Memoria CACHÉ

- El µP pide el contenido de una posición de memoria
- Revisa la caché para ver si contiene esa información
- Si está presente, lo obtiene de la caché
- Si no está presente, lee el bloque requerido desde la memoria principal hacia la caché.
- Luego la información es entregada desde la caché hacia el µP.
- La caché incluye etiquetas para identificar cuál bloque de la memoria principal está en una casilla de la caché.

50

¿Porqué Memoria CACHE?

- Un sistema de memoria el cual almacena temporalmente información accedida frecuentemente y obtenida de dispositivos de almacenamiento con mayor tiempo de acceso.
- Con esto, se busca aminorar el tiempo de acceso, mientras la memoria CACHE es mayor, nuestra PC será mas rápida

51

¿Necesitamos velocidad?

- Es posible implementar una PC con sólamente memoria del tipo SRAM
- Sería sumamente rápido
- No se necesitaría memoria CACHE
- El costo sería extremadamente exorbitante!

52

Cuestionario

- Revisar modelos de ALU en VHDL

10 decimal
0x10 hexadecimel
10h DEH

53

Fin de la sesión

54