Diseño Automático de sistemas

Diseño de Sistemas Digitales (II)

Prof. Pablo Sarabia Ortiz

UNIVERSIDAD NEBRIJA

Objetivos Clase

- Lograr un entendimiento mejor de como debe organizarse el código VHDL y de estructuras básicas mediante ejemplos.
- Comprender que VHDL es descripción de HW
- Tener una serie de atajos y trucos que nos faciliten el trabajo.
- Entender los diagramas ASM para poder definir de forma precias un sistema.

Bibliografia

- **Capítulos 4 y 5**
  Digital Systems Design Using VHDL (Second Edition), Charles H. Roth, Jr and Lizy Kurian John.

- **Cápitulo 10**
  Pong P. Chu (2006), RTL Hardware Design Using VHDL: Coding for Efficiency, Portability, and Scalability, Willey, 1st Edition

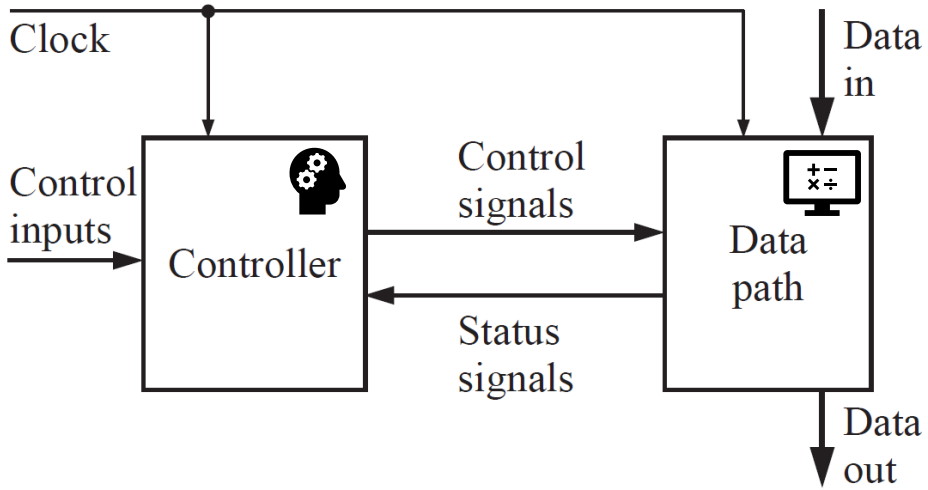
- **Repo** con el código:
  https://bitbucket.org/psarabiaortiz/fpgas/src/main/

Contenidos

1. Patrón diseño digital
2. Interpretación física del VHDL
3. Ejemplo BCD a 7 segmentos
4. Señales y generación de señales en VHDL
5. Marcador ejemplo
6. Diagrama de flujo (ASM)
7. Recursos útiles en VHDL

Patrón diseño digital

De forma genérica un diseño digital consta de dos partes diferenciadas: **Controlador** y **Datapath**



Patrón diseño digital: Controlador

- Controlador: Es la parte de HW encargada de mandar las señales de control y actuar en consecuencia de las señales de estado y las entradas del sistema.
  - Se implementa habitualmente mediante máquinas de estado o funciones secuenciales ej: if, case, loop, etc

Patrón diseño digital: Datapath

- Datapath: El Datapath es la parte del hardware que realiza el procesado. (No confundir con el databus)
  - Tiene bloques lógicos combinacionales y registros principalmente. Suele tener operadores bloques de procesado de señal, etc…

Patrón diseño digital: ¿Por qué?

- Fiable: El uso de este esquema limita los errores
- Sencillo: Es más fácil de realizar modificaciones.
- Modular y reusable: Los bloques son independientes y los podemos usar en diferentes diseños.
- Sistemático: Agiliza el diseño.
- Comprobable: Se pueden probar los bloques por separado, más sencillo debuguear y verificar el funcionamiento.

Interpretación física del código VHDL

Recuerda: VHDL **describe** HW

Al usar VHDL estamos realizando circuitos eléctricos por eso hay que prestar especial atención a que estemos planteando circuitos posibles.

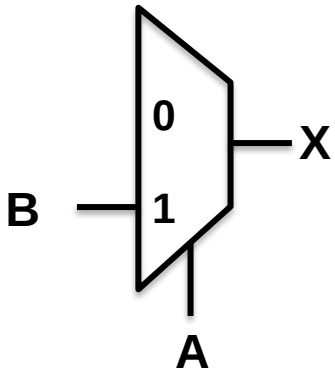
If else

Habitualmente cuando pensamos en una sentencia if lo vemos como una condición en hardware debemos pensar en un selector o mux.

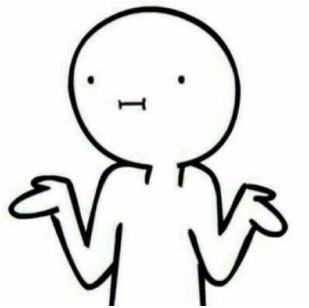
If else

```
3  if A = '1' then
4      X <= B;
5  end if;
```

¿problema?



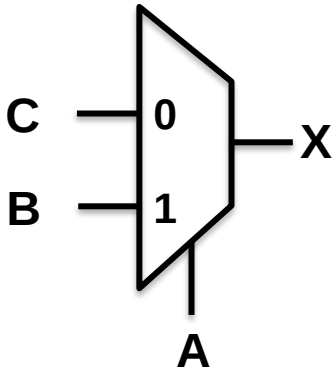
¿Qué pasa cuando A vale 0?



Solución:

Definir todos los casos

```
7  if A = '1' then
8      X <= B;
9  else
10     X <= C;
11 end if;
```



Signals (II)

Signals

Las signals **NO** son variables no tienen un espacio en memoria donde se almacena el valor. Son cables.

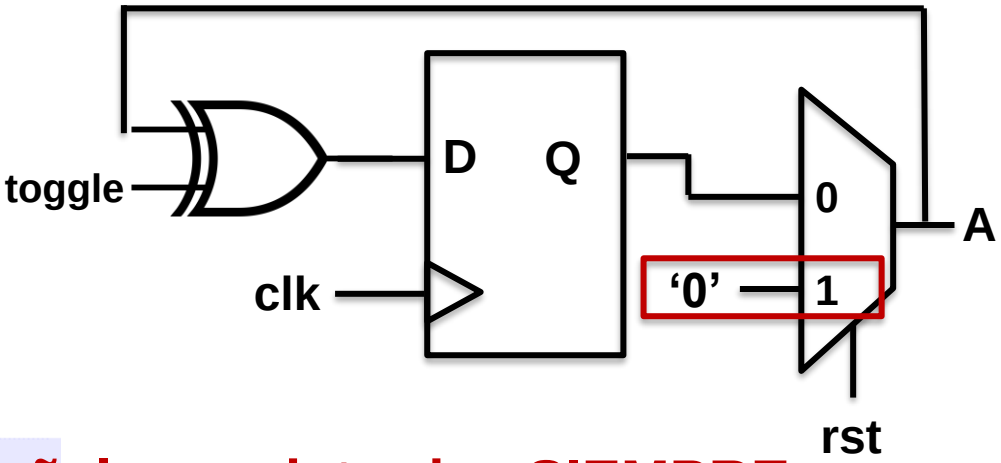
A <= B

Signals (II)

Solución

Si queremos que mantengan el valor debemos registrarlas

```
2  process (clk, rst)
3      if rst = '1'then
4          A <= '0';
5      elsif rising_edge(clk) then
6          A <= B;
7      end if;
8  end process;
9
10 process (A, toggle)
11     B <= A xor toggle;
12 end process;
13
```

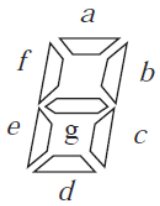


Inicializar las señales registradas SIEMPRE

Ejemplo: conversor BCD a 7 segmentos

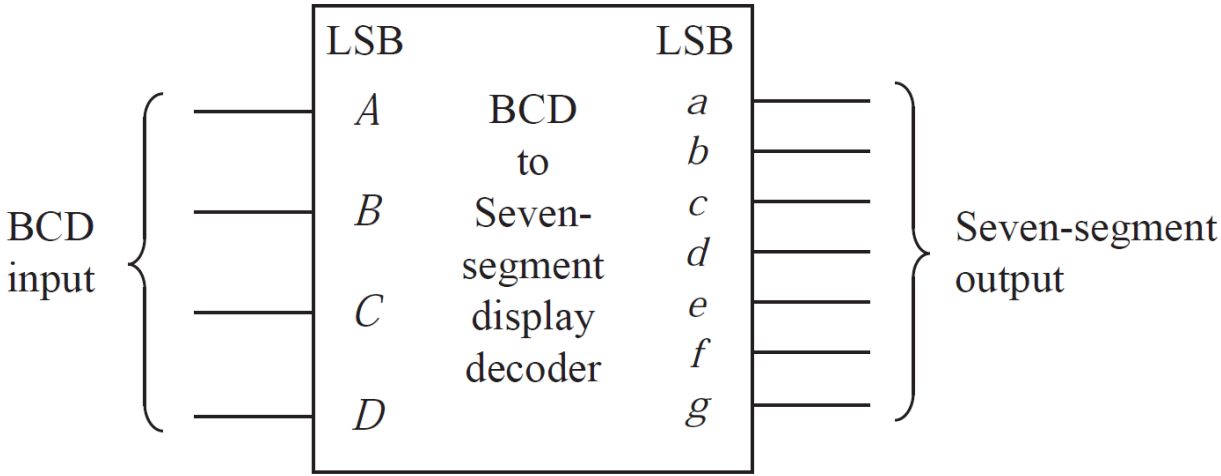
Problema:

Representar un número en notación BCD (Binary Coded Digit)



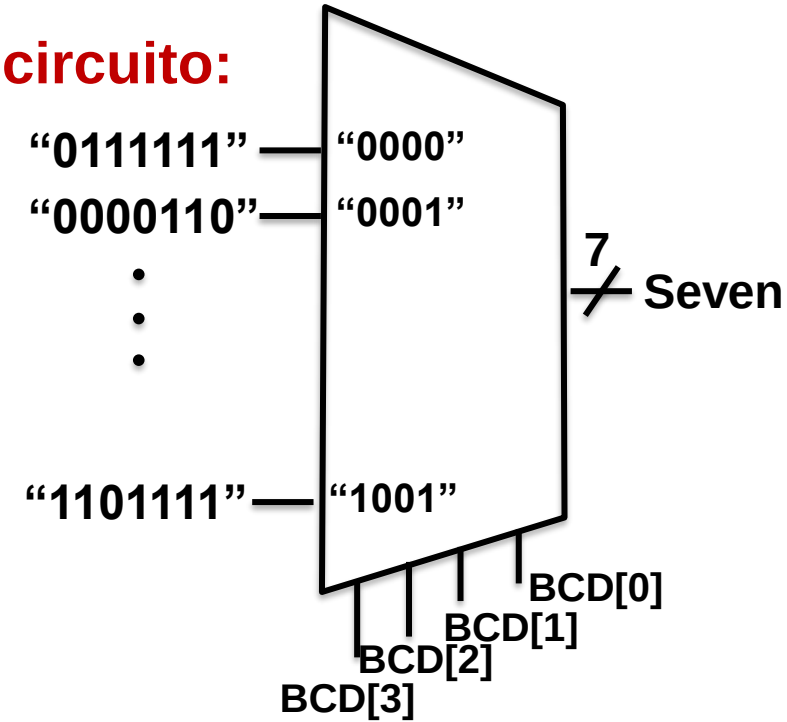
Ejemplo: conversor BCD a 7 segmentos (II)

Diagrama de bloques del sistema:



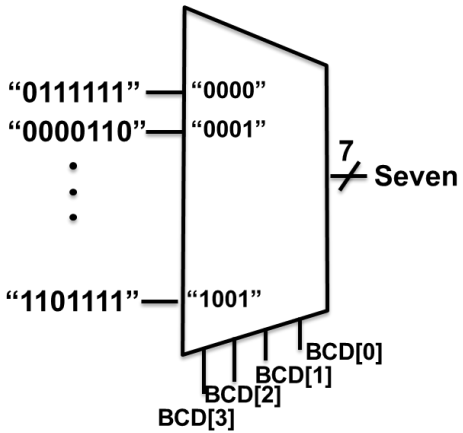
Ejemplo: conversor BCD a 7 segmentos (III)

Descripción del circuito:



Ejemplo: conversor BCD a 7 segmentos (III)

Descripción en VHDL:




```vhdl
entity bcd_seven is
port(   bcd: in std_logic_vector(3 downto 0);
        seven: out std_logic_vector(7 downto 1));
-- LSB is segment a of the display. MSB is segment g
end bcd_seven;
architecture behavioral of bcd_seven is
begin
    process (bcd)
    begin
        case bcd is
            when "0000" => seven <= "0111111";
            when "0001" => seven <= "0000110";
            when "0010" => seven <= "1011011";
            when "0011" => seven <= "1001111";
            when "0100" => seven <= "1100110";
            when "0101" => seven <= "1101101";
            when "0110" => seven <= "1111101";
            when "0111" => seven <= "0000111";
            when "1000" => seven <= "1111111";
            when "1001" => seven <= "1101111";
            when others => null;
        end case;
end process;
end behavioral;
```

Señales y generación de señales

¿Las signal son cables entonces, cómo les damos valor?

- Cada bloque (combinacional o secuencial) da valor a unas señales.
- Cada señal **SOLO** es modificada por un bloque.
- Una señal puede ser utilizada por varios bloques.
- Las señales registradas deben ser inicializadas **SIEMPRE**

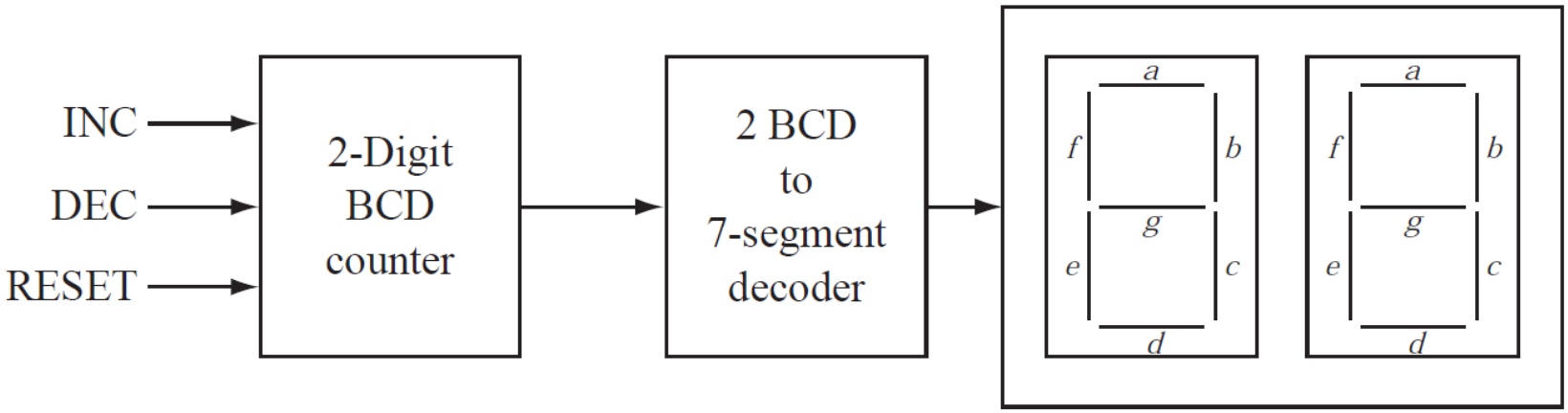
Ejemplo: Marcador

Objetivo: Diseñar un marcador que permita contar desde 0 a 99, incrementando o decrementar el valor.

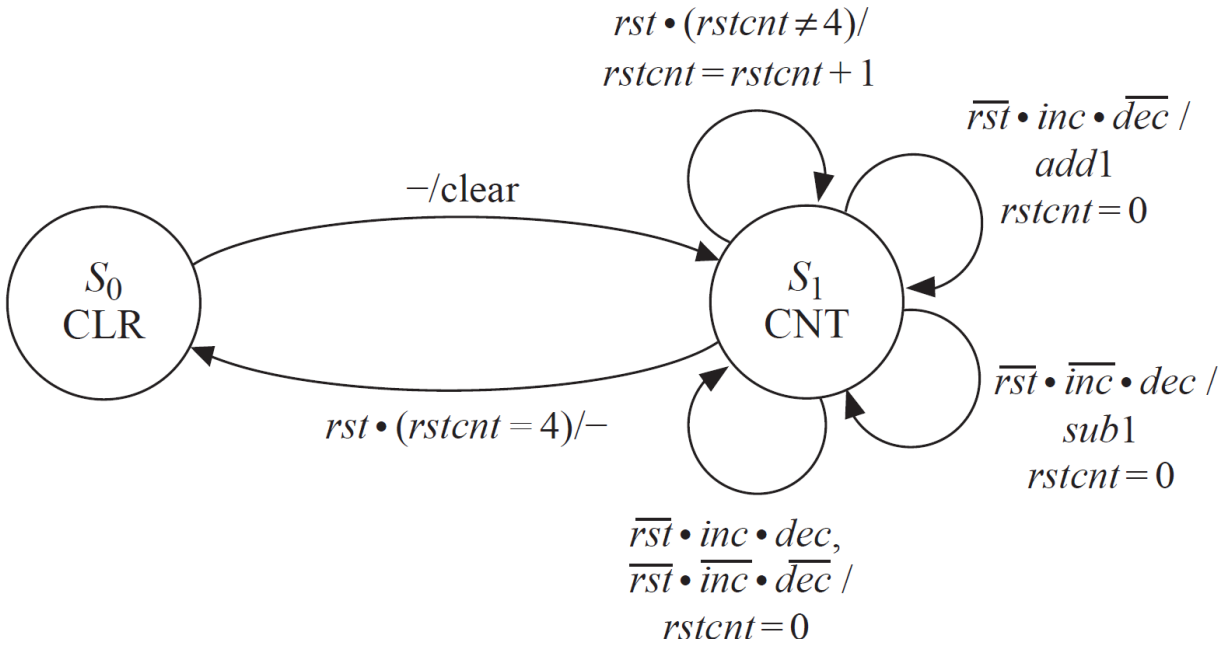
Detalles:

- Si la señal de INC está a 1 se aumenta en 1 el contador.
- Si la señal de DEC está a 1 se decrementa en 1 el contador.
- Si se pulsan ambas no se hace nada.
- Para borrar el valor mostrado en la pantalla el reset se debe pulsar durante 5 ciclos consecutivos.

Ejemplo: Marcador, datapath



Ejemplo: Marcador, controller



Ejemplo: Let’s Code

El código está en el repo.

Ejemplo Visualización del testbench

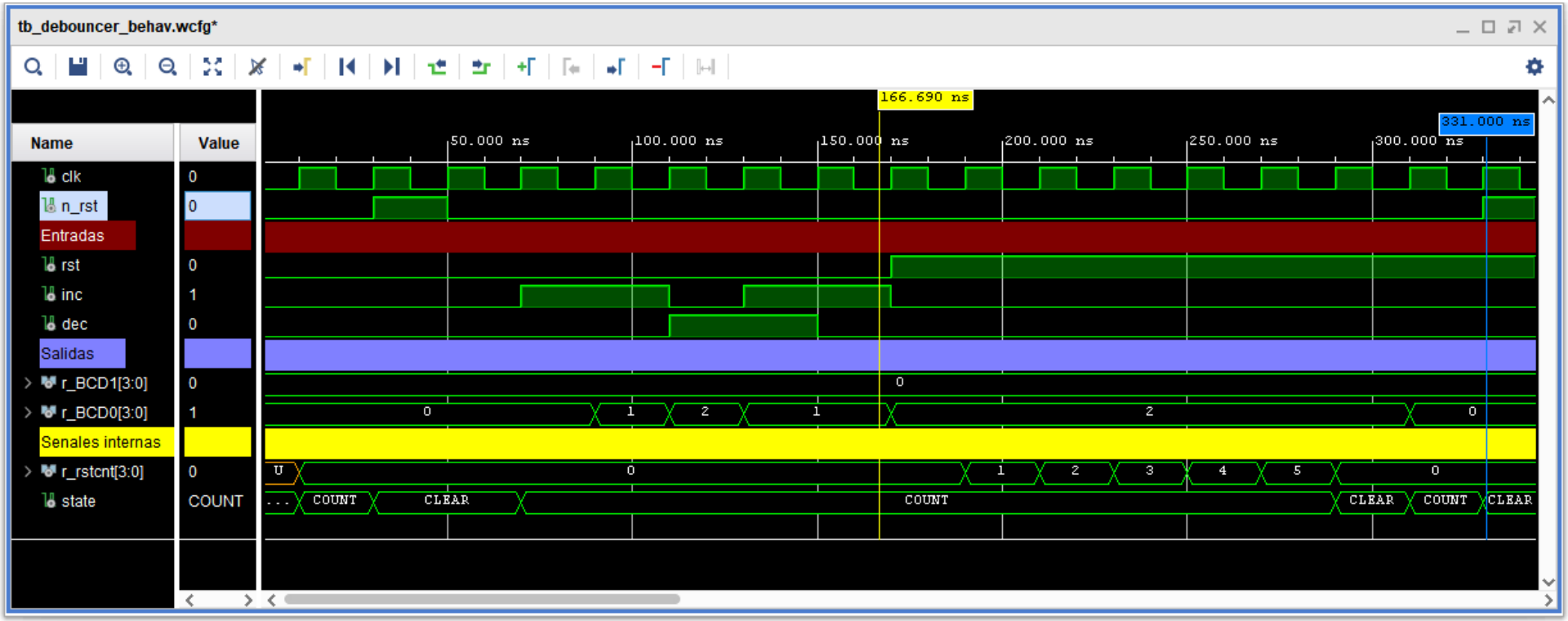
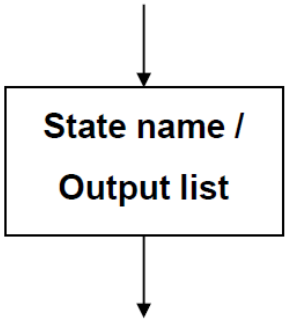


Diagrama de flujo (ASM)

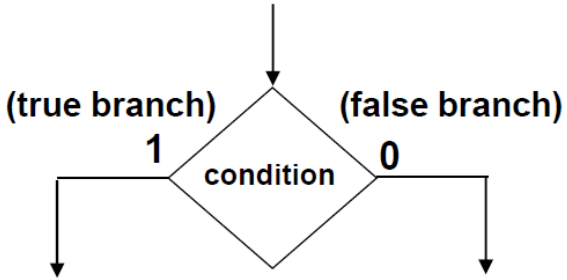
- **Alternativa** a los diagramas de estado
- Contiene la **misma información** que un diagrama de estados, pero es más **descriptivo**.
- Útiles para el **diseño** y **refinar** la implementación.
- Están compuestos por bloques ASM que pueden ser de tres tipos: **state box**, **decision box** y **condicional output box**.

Componentes de un ASM

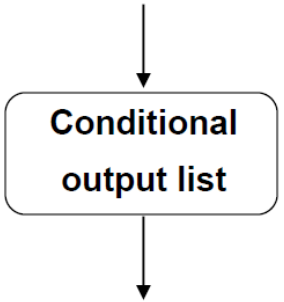
a) State box: Contiene el nombre del estado y opcionalmente la lista de salidas.
b) Decision box: Contiene una expresión booleana que determina la rama a seguir.
c) Condicional output box: Contiene una lista de salidas que se



(a) State box



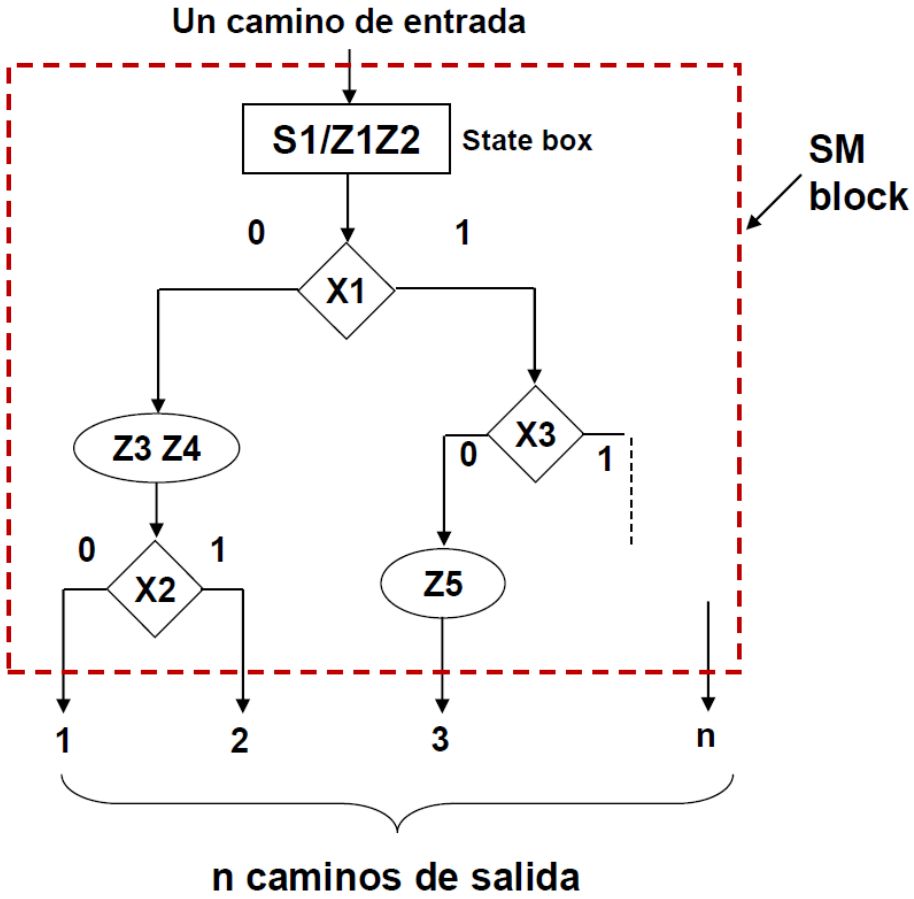
(b) Decision box



(c) Conditional Output box

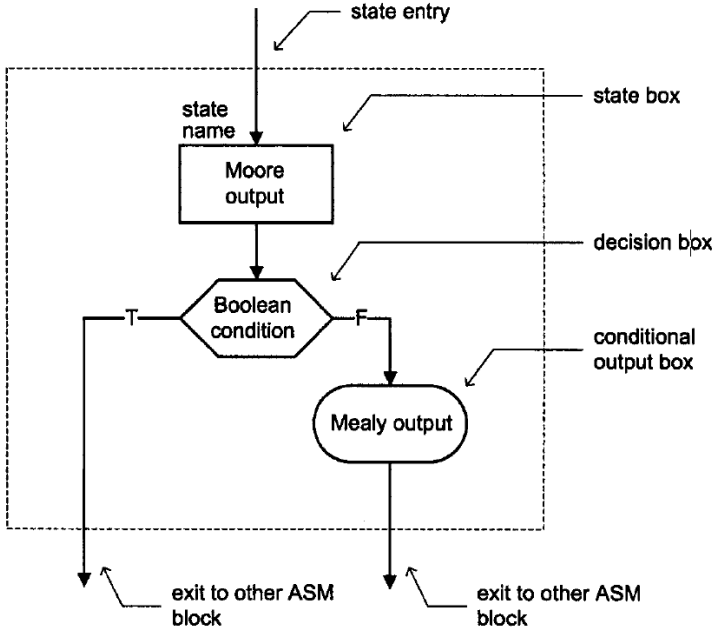
ASM Blocks

- Describe el funcionamiento de **un estado** de la FSM.
- Cada ASM Block contiene **solo un state box**
- Tiene **un camino de entrada** pero **“n” de salida**
- Un ASM chart tiene **varios** SM Blocks



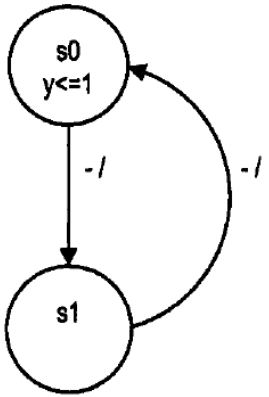
ASM Blocks para FSMs

- Un ASM Block puede describir FSMs de Mealy y Moore.
- Un único diagrama de estados puede dar lugar a varios ASM chart.
- Un ASM chart solo da lugar a un diagrama de estados.

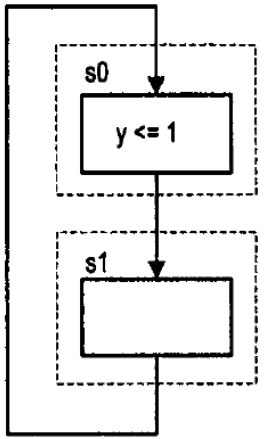


Ejemplos de conversión (I)

Solo escribimos las variables que cambian.



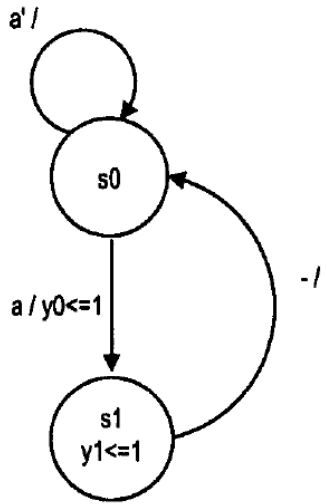
(a)



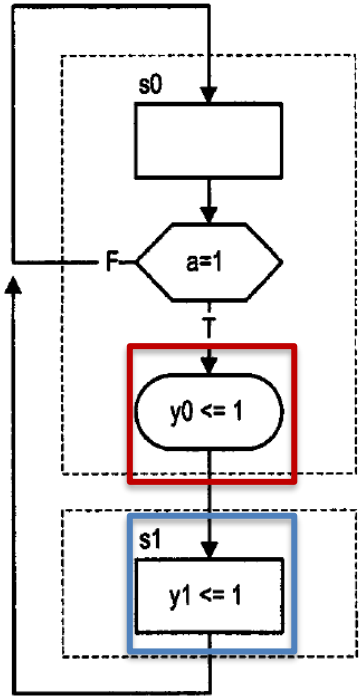
(b)

Ejemplos de conversión (II)

Podemos mezclar **Mealy** (rojo) con **Moore** (Azul)



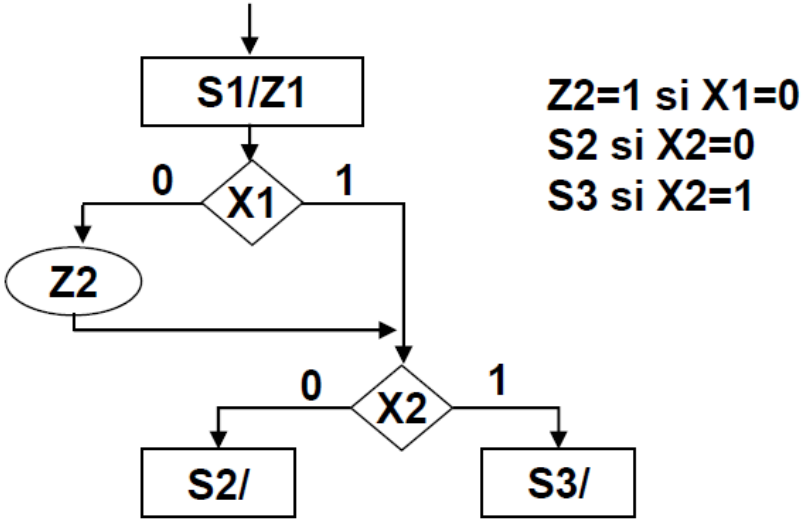
(a)



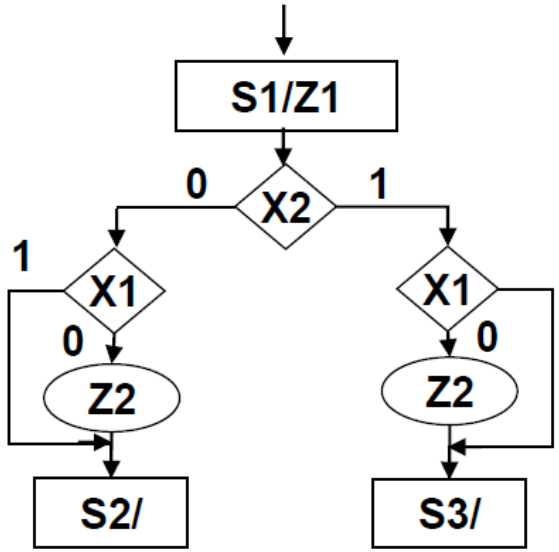
(b)

Ejemplos de ASM equivalentes

Dos diagramas ASM pueden ser equivalentes:



Z2=1 si X1=0
S2 si X2=0
S3 si X2=1



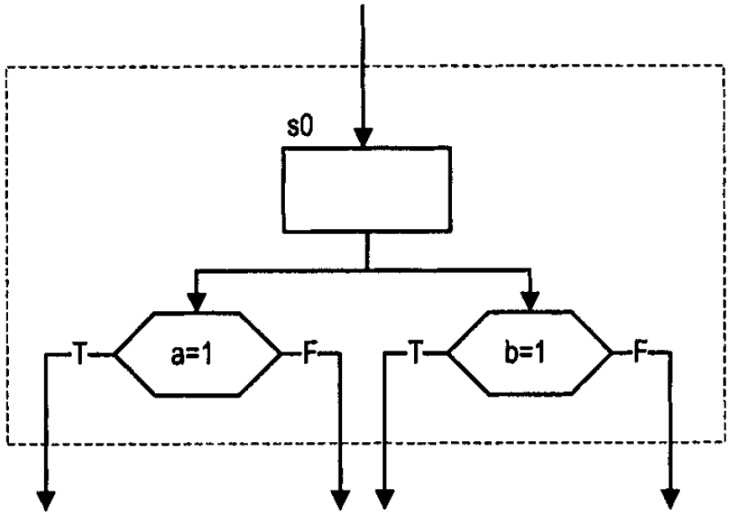
Z2=1 si X1=0
S2 si X2=0
S3 si X2=1

El orden en el que se evalúan las entradas puede afectar a la complejidad

Propiedades generales

- Remanencia: Sólo escribimos las señales que cambian.
- Unicidad: Dada una combinación de entradas solo existe una salida para cada ASM Block.

Errores comunes en ASM (I)

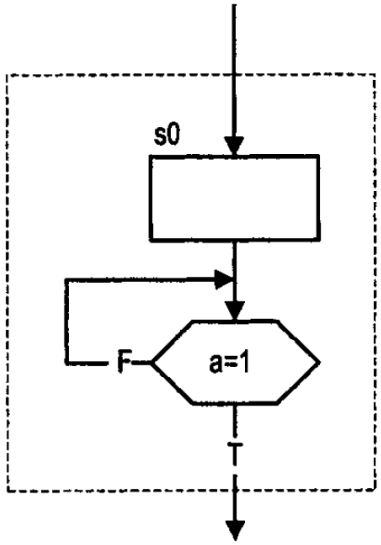


(a)

- Bifurcación sin una condición.
- No está definida la salida para un conjunto de entradas dado: **Condición de carrera** (según el orden de evaluación de las señales el resultado cambia)

Solución: Poner en cascada las condiciones.

Errores comunes en ASM (II)

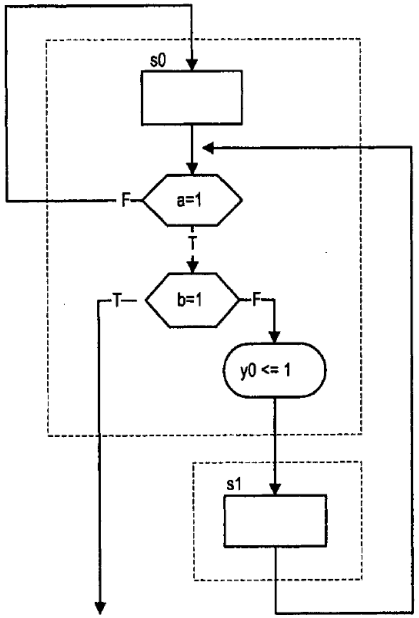


(b)

- Realimentación dentro del mismo bloque.
- Todas las salidas del bloque no llevan a otro bloque.

Solución: Si a=1 es false salir del bloque y pasar por el estado s0.

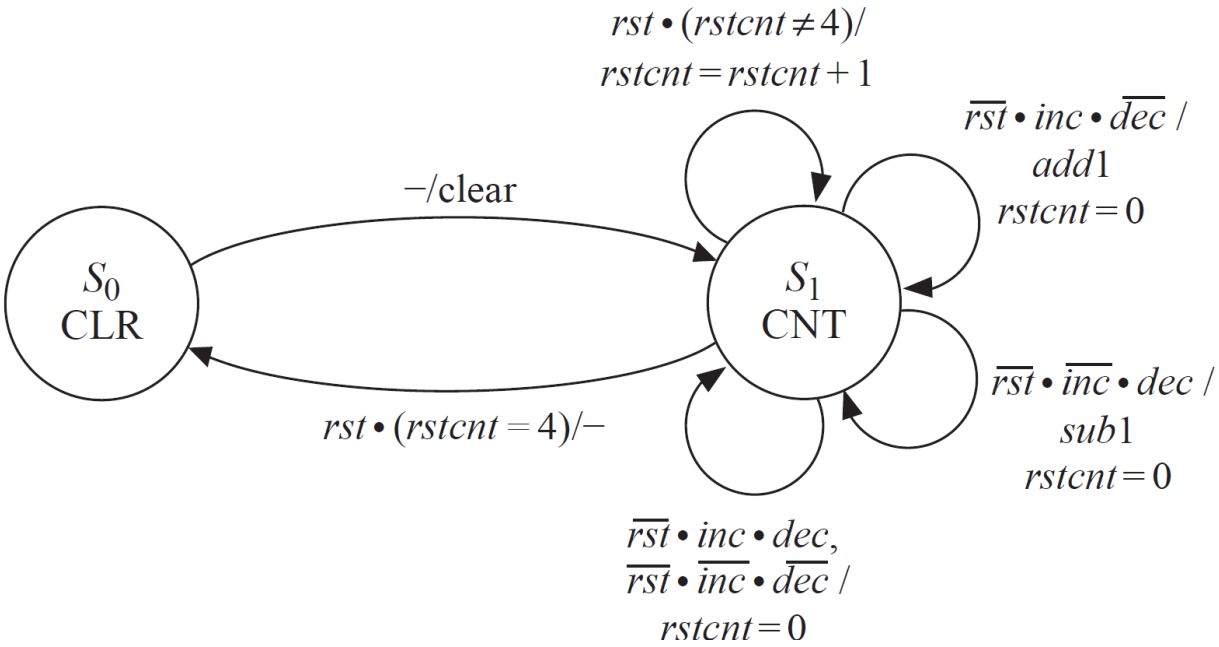
Errores comunes en ASM (III)



- El estado s1 no está conectado con otro estado.

Solución: Crear un estado intermedio para que no compartan lógica el estado s0 y s1.

Ej: Convertir a ASM el scoreboard controller



Recursos útiles para VHDL

- Plantillas: Reutilizar estructuras para: timers, fsms, contadores, memorias, etc.
- Repositorios Online: Ej: opencores.org
- Herramientas de generación automática: Vitis HLS, Matlab, IP de Xilinx, etc..

Resumen

- El patrón de diseño controlador datapath es muy útil en diseño digital.
- Describimos HW luego piensa en HW
- Las señales se generan en un único lugar del código.
- Los diagramas ASM son importantes y los utilizaremos en próximas clases.
- Existen recursos para acelerar el proceso de diseño.

En la próxima clase

- Como implementar un diagrama ASM.
- Ejemplo completo de implementación usando diagrama ASM.
- Introducción a VHDL avanzado.
- Proyecto, siguiente hito.

¿Preguntas?

UNIVERSIDAD
NEBRIJA