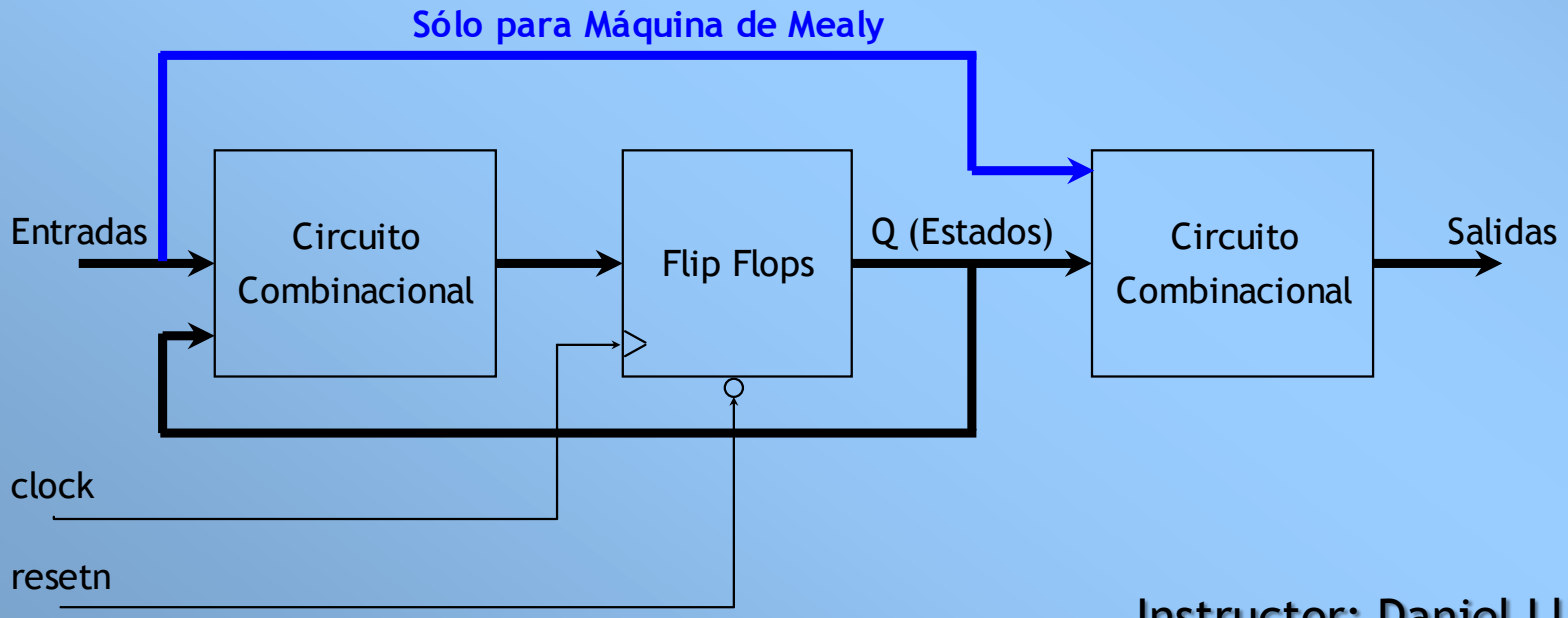
LABORATORIO DE CIRCUITOS DIGITALES
(2005-II)
QUINTA CLASE DE VHDL

- ✓ *MÁQUINAS DE ESTADO FINITAS (FSMs)*
  - Máquinas de Moore
  - Máquinas de Mealy

Instructor: Daniel Llamocca

✓ *MÁQUINAS DE ESTADOS FINITAS (FSMs)*

- Tipos:
  - Máquina de Moore: Las salidas dependen solamente del estado en el que se encuentra el circuito.
  - Máquina de Mealy: La salida depende tanto del estado presente así como de las entradas del circuito.
- Para colocar al circuito en un estado inicial, siempre debe existir una entrada asíncrona 'resetn', la que no se considera para decidir si una máquina es de Moore o de Mealy. Se muestra la forma genérica de una máquina de estados:



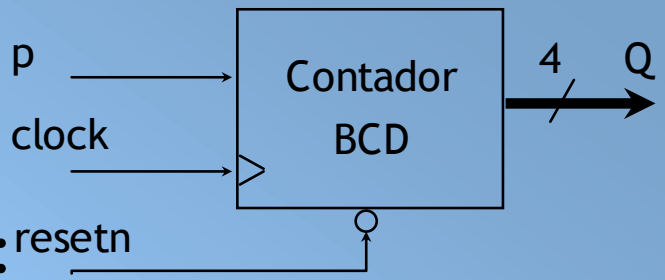
✓ *MÁQUINAS DE ESTADOS FINITAS (FSMs)*

- Codificación en VHDL: Hay varias formas de codificar máquinas de estados en VHDL. Se explicará sólo una forma muy utilizada debido a su simplicidad:
  - El código se realiza directamente a partir del diagrama de estados, por lo que sólo es necesario disponer del diagrama de estados.
  - Se debe definir un nuevo tipo de dato (p.e. 'estado') para representar los estados:

```
type estado is (S1, S2, ... S10) -- se tienen 10 estados
signal y: estado;
```

    La señal 'y' es ahora del tipo 'estado', y sólo puede tomar valores de S1 a S10, esta señal también puede apreciarse en la simulación.
  - Deberán hacerse dos procesos: uno en donde se definan los cambios de estados que ocurren en el flanco de reloj, y otro en donde se definan cómo es que las salidas cambian en base a los estados presentes.

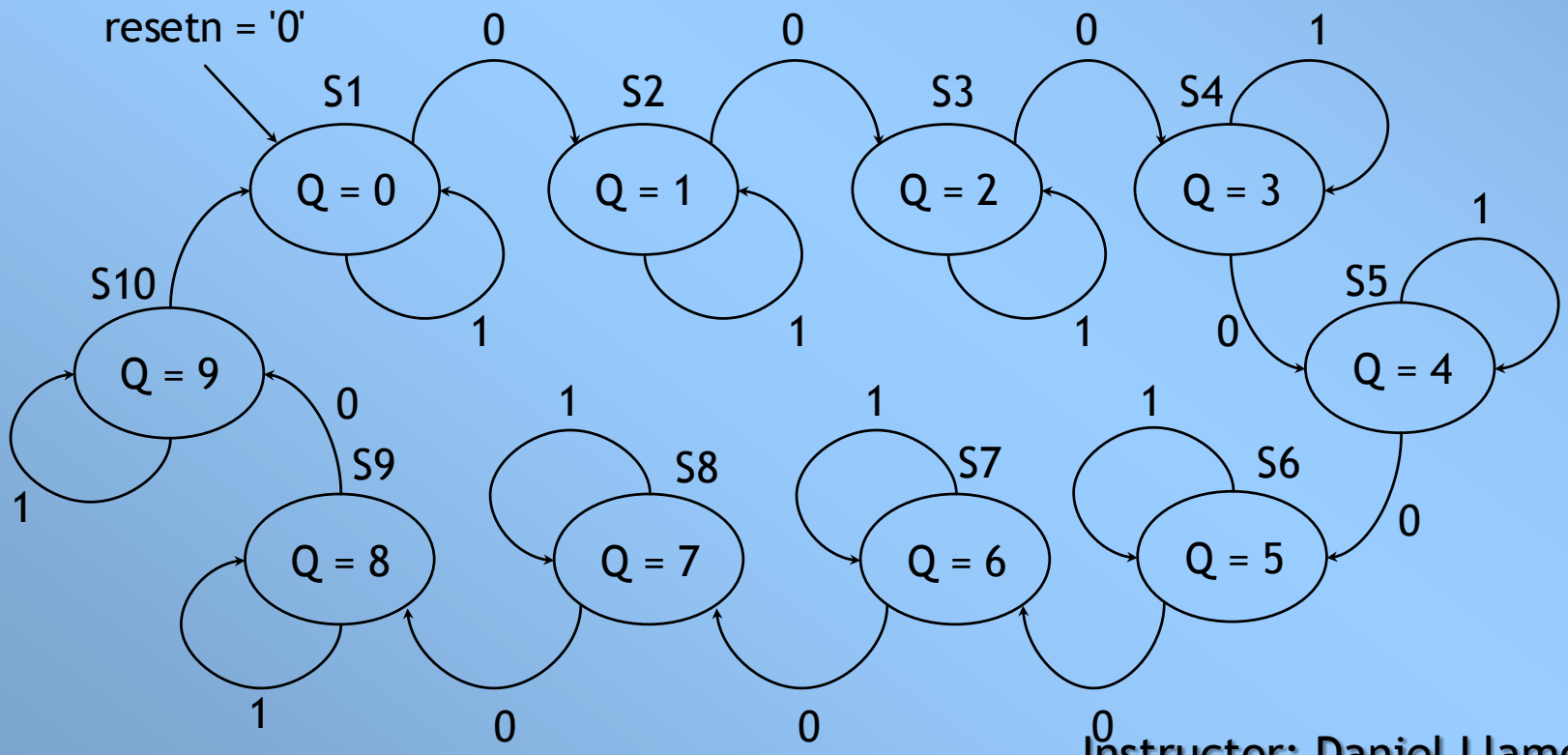
✓ *MÁQUINA DE MOORE*



- Ejemplo:

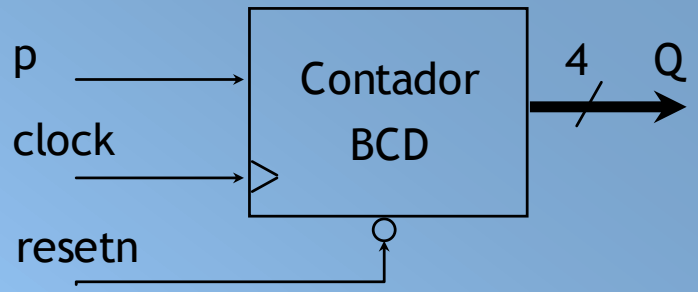
  Contador BCD con señal de parada:

Un contador BCD también puede definirse en base al método de la máquina de estados. Si se activa la señal de parada (´p´), la cuenta se detiene. Note que la señal 'p' es la que permite el cambio de estados:



✓ *MÁQUINA DE MOORE*

- Contador BCD con señal de parada:



- Código VHDL:

```
library ieee;
use ieee.std_logic_1164.all;

entity contbcd is
 port (clock, resetn: in std_logic;
    p: in std_logic;
    Q : out integer range 0 to 9);
end contbcd;

architecture behaviour of contbcd is
 type estado is (S1, S2, S3, S4, S5, S6, S7, S8, S9, S10);
 signal y: estado;

begin
 Cambios_de_Estado: process (resetn, clock)
 begin
  if resetn = '0' then -- señal asíncrona
   y <= S1; -- Si resetn = '0' ==> estado inicial: S1
...
```

Definición de nuevo tipo de dato 'estado' que tiene 10 posibles estados: S1 → S10

Definición de la señal 'y' de tipo 'estado'. A la señal 'y' puede asignarséle valores de S1 a S10

Proceso que define los cambios de estado

```
elsif (clock'event and clock = '1') then
  case y is
   when S1 =>
    if p = '0' then y <= S2; else y <= S1; end if;
   when S2 =>
    if p = '0' then y <= S3; else y <= S2; end if;
   when S3 =>
    if p = '0' then y <= S4; else y <= S3; end if;
   when S4 =>
    if p = '0' then y <= S5; else y <= S4; end if;
   when S5 =>
    if p = '0' then y <= S6; else y <= S5; end if;
   when S6 =>
    if p = '0' then y <= S7; else y <= S6; end if;
   when S7 =>
    if p = '0' then y <= S8; else y <= S7; end if;
   when S8 =>
    if p = '0' then y <= S9; else y <= S8; end if;
   when S9 =>
    if p = '0' then y <= S10; else y <= S9; end if;
   when S10 =>
    if p = '0' then y <= S1; else y <= S10; end if;
  end case;
 end if;
end process;
...
```

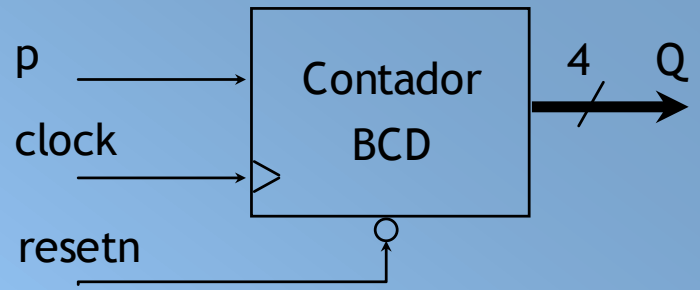
Note que los cambios de estados dependen de la entrada ‘p’

Proceso que define los cambios de estado

Note que los cambios de estados ocurren **sólo** en el flanco de subida

✓ *MÁQUINA DE MOORE*

- Contador BCD con señal de parada:
- Código VHDL:




```
...
 Salidas: process (y)
 begin
  case y is
   when S1 =>     Q <= 0;
   when S2 =>     Q <= 1;
   when S3 =>     Q <= 2;
   when S4 =>     Q <= 3;
   when S5 =>     Q <= 4;
   when S6 =>     Q <= 5;
   when S7 =>     Q <= 6;
   when S8 =>     Q <= 7;
   when S9 =>     Q <= 8;
   when S10 =>    Q <= 9;
  end case;
 end process;
end behaviour;
```

Note que las salidas sólo dependen del estado presente, lo que concuerda con la definición de la máquina de Moore

Proceso que define las salidas

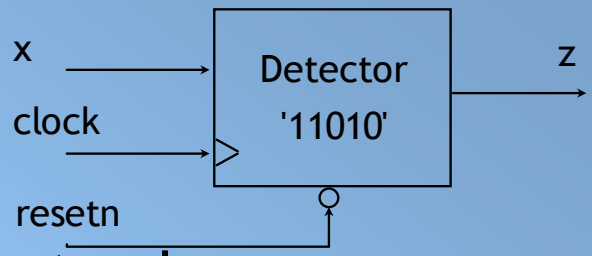
Note que la salida no está controlada por el flanco de subida, sólo por el estado presente

Código fuente: **contbcd.vhd**

Simulación: **contbcd.scf**

Instructor: Daniel Llamocca

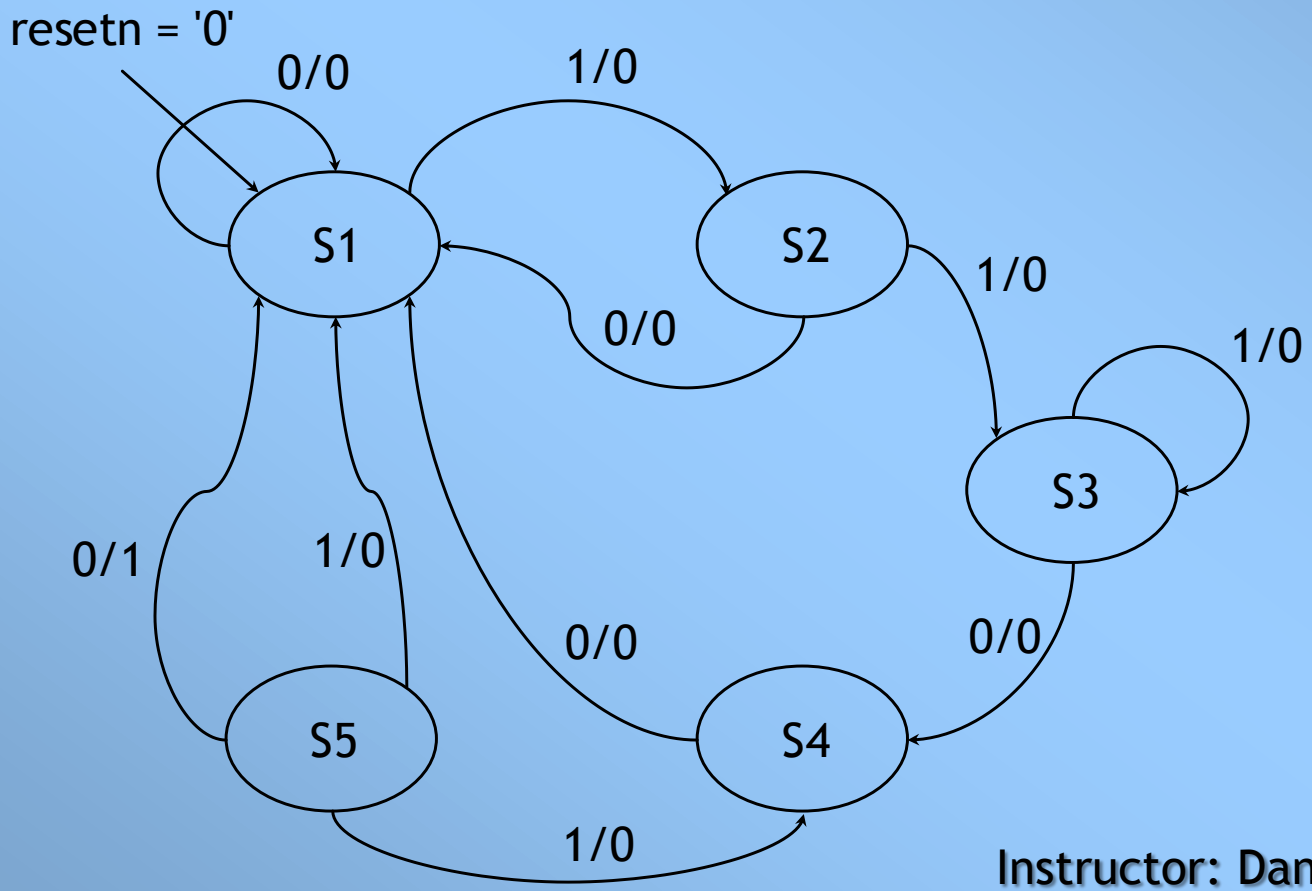
✓ *MÁQUINA DE MEALY*



- Ejemplo:

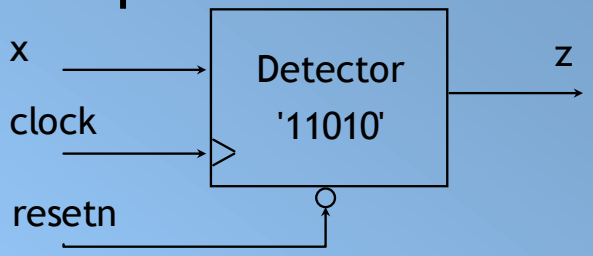
  Detector de secuencia ‘11010’ con traslape

Se muestra el diagrama de estados. Sólo son necesarios cinco estados, la entrada es ‘x’ y la salida es ‘z’.



✓ *MÁQUINA DE MEALY*

- Detector de secuencia ‘11010’ con traslape
- Código VHDL:




```
library ieee;
use ieee.std_logic_1164.all;

entity detector is
 port (clock, resetn: in std_logic;
    x: in std_logic;
    z : out std_logic);
end detector;

architecture behaviour of detector is
 type estado is (S1, S2, S3, S4, S5);
 signal y: estado;

begin
 Cambios_de_Estado: process (resetn, clock)
 begin
  if resetn = '0' then -- señal asíncrona
   y <= S1; -- Si resetn = '0' ==> estado inicial: S1
...
```

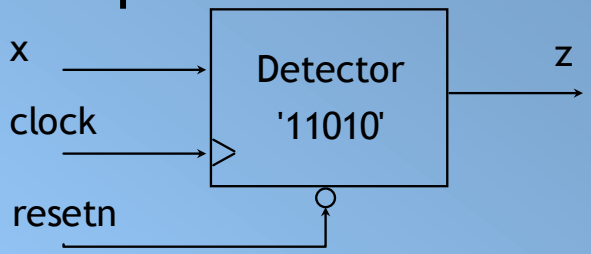
Definición de nuevo tipo de dato ‘estado’ que tiene 5 posibles estados: S1 → S5

Definición de la señal ‘y’ de tipo ‘estado’. A la señal ‘y’ puede asignarséle valores de S1 a S5

Proceso que define los cambios de estado

✓ *MÁQUINA DE MEALY*

- Detector de secuencia ‘11010’ con traslape
- Código VHDL:




```
...
  elsif (clock'event and clock = '1') then
   case y is
    when S1 =>
     if x = '1' then y <= S2; else y <= S1; end if;
    when S2 =>
     if x = '1' then y <= S3; else y <= S1; end if;
    when S3 =>
     if x = '1' then y <= S3; else y <= S4; end if;
    when S4 =>
     if x = '1' then y <= S5; else y <= S1; end if;
    when S5 =>
     y <= S1;
   end case;
  end if;
 end process;
...
```

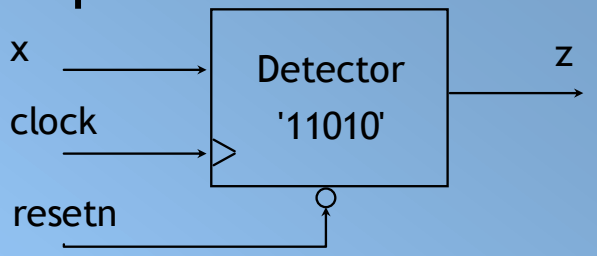
Note que los cambios de estados dependen de la entrada ‘x’

Proceso que define los cambios de estado

Note que los cambios de estados ocurren **sólo** en el flanco de subida

✓ *MÁQUINA DE MEALY*

- Detector de secuencia ‘11010’ con traslape
- Código VHDL:




```
...
 Salidas: process (y)
 begin
  z <= '0'; -- SIEMPRE debe dársele un valor
   -- por defecto a 'z', por si ninguna de
   -- las condiciones siguientes se cumple.
  case y is
   when S1 =>
   -- se puede dejar vacío, como 'z' ya se
   -- inicializó en '0' en S1 valdrá '0'.
   when S2 =>
   when S3 =>
   when S4 =>
   when S5 =>
    if x = '0' then z <= '1'; end if;
     -- Si x = '1' y aquí no se indica, 'z'
     -- toma su valor por defecto (z <='0')
   end case;
 end process;
end behaviour;
```

Note que la salida dependen del estado presente **y de la entrada ‘x’**

Proceso que define las salidas

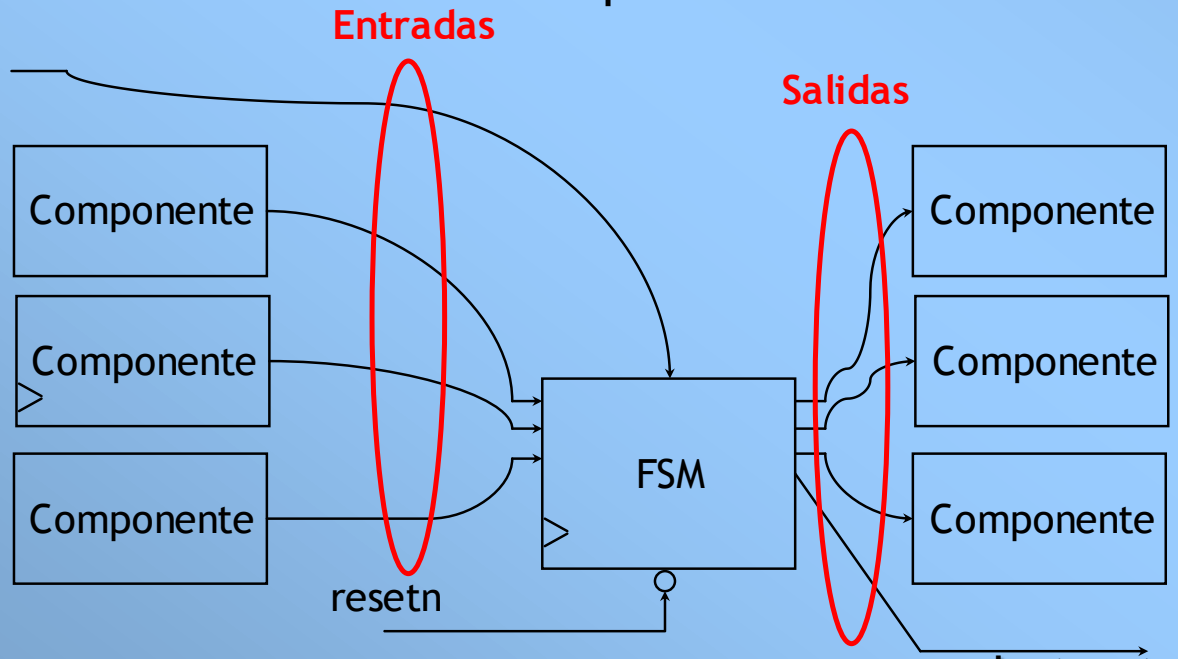
Código fuente: **detectpr.vhd**

Simulación: **detector.scf**

Instructor: Daniel Llamocca

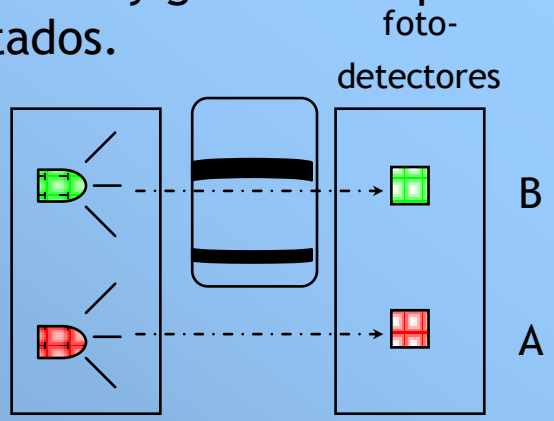
✓ *MÁQUINAS DE ESTADOS FINITAS (FSMs)*

- La mayoría de máquinas de estados en circuitos reales son de Mealy, pues este tipo ofrece mayor control para describir las salidas.
- **Observación**: En los ejemplos anteriores se ha visto que la máquina de estados era todo el circuito necesario para implementar un problema dado. Sin embargo, rara vez este es el caso. En un sistema real, la máquina de estados es sólo un circuito de control que maneja varios componentes (síncronos o asíncronos). Las entradas de la FSM pueden venir de estos componentes o del exterior, así como las salidas de la FSM pueden ir hacia estos componentes o al exterior.



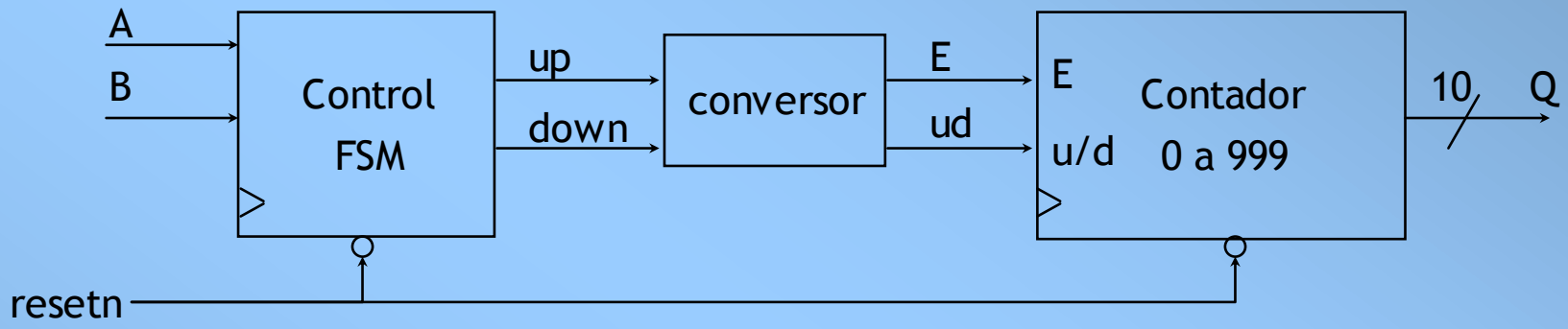
✓ *MÁQUINAS DE ESTADOS FINITAS (FSMs)*

- Ejemplo: Contador de número de carros en un estacionamiento.
- El estacionamiento tiene una puerta por la que sólo puede entrar o pasar un carro a la vez. Existen 2 señales A y B que provienen de fotodetectores alineados con LEDs. Cada detector produce un '1' cuando el carro obstruye el camino entre el LED y el respectivo fotodetector.
- Consideraciones:
- Si el carro ingresa , primero deberá obstruir el LED A, luego los dos LEDs (A y B) y seguidamente el LED B. Por último ningún LED deberá estar obstruido. Si se cumple esta secuencia se aumenta la cuenta del número de carros. Para los carros que salen el proceso es inverso. Notar que la secuencia debe cumplirse, porque un carro puede ingresar o salir sólo parcialmente y luego regresar; en este caso no se debe aumentar ni disminuir la cuenta.
- También se debe notar que el carro puede quedarse en un estado por varios ciclos, ya que la velocidad del carro es muy grande comparada con la frecuencia del reloj que controla el cambio de estados.



✓ *MÁQUINAS DE ESTADOS FINITAS (FSMs)*

- Consideraciones de Diseño:
- Primero debe hacerse el Diagrama de Bloques:



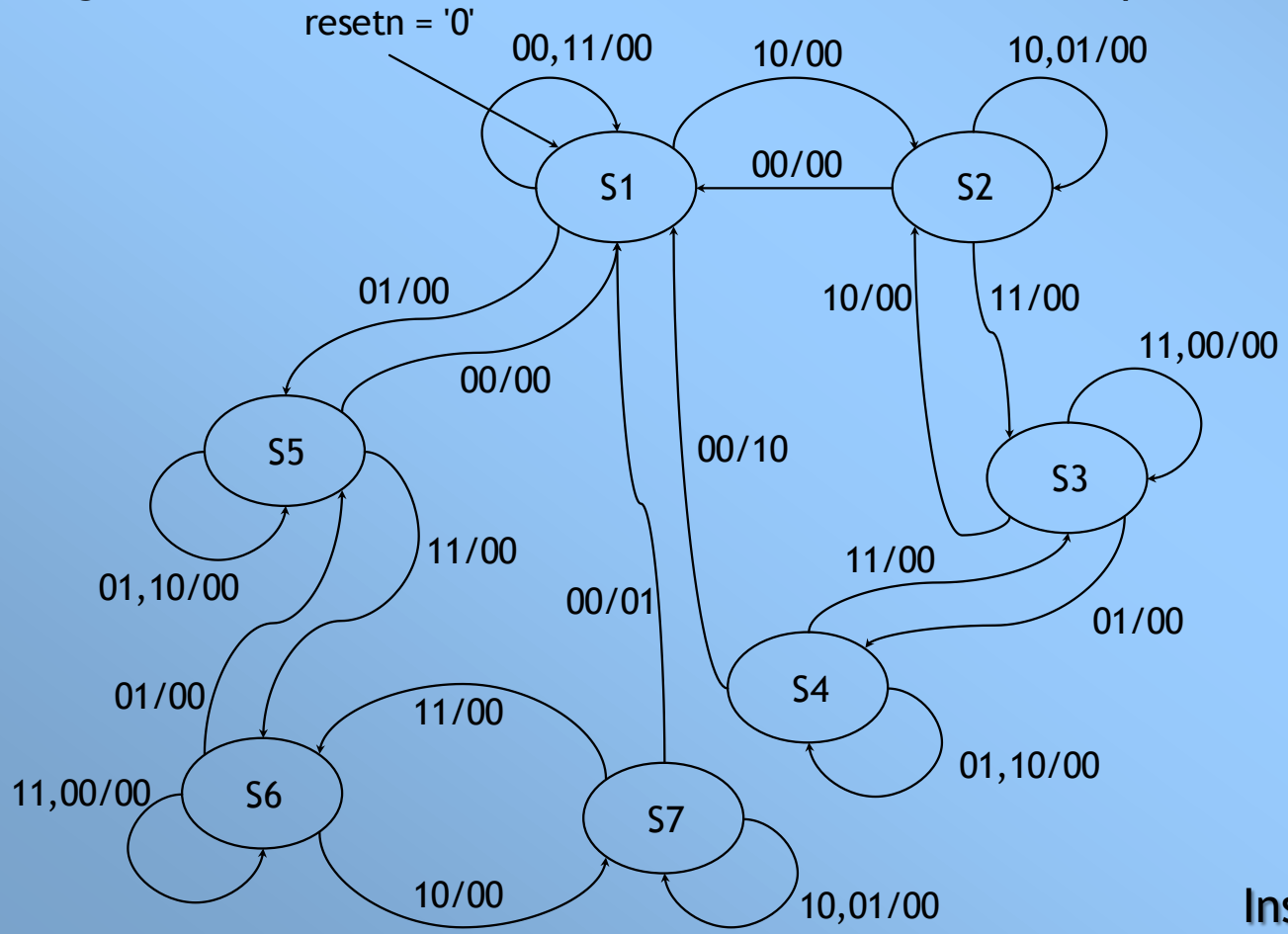
- El bloque de Control es la FSM, la que pondrá la señal ‘up’ en ‘1’ cuando se necesite aumentar la cuenta, y pondrá la señal ‘down’ en ‘1’ cuando se necesita disminuir la cuenta
- El conversor se encarga de transformar las señales ‘up’ y ‘down’ en señales ‘E’ (habilitador) y ‘ud’ (control de ascendente/descendente), mediante la siguiente tabla de verdad:

up	down	E	ud
0	0	0	X
0	1	1	0
1	0	1	1
1	1	0	X

- El contador 0 a 999 es un contador de 10 bits con habilitador y control ascendente/descendente.

✓ *MÁQUINAS DE ESTADOS FINITAS (FSMs)*

- Para que up = ‘1’ debe ocurrir la siguiente secuencia:
- AB = “10”, “11”, “01”, “00”
- Para que down = ‘1’ debe ocurrir la siguiente secuencia:
- AB = “01”, “11”, “10”, “00”
- Diagrama de Estados: Entradas: A, B Salidas: up, down



Código fuente:

carros.vhd (principal)

control_carros.vhd (FSM)

cont0_999.vhd (contador)

Simulación: **carros.scf**

Instructor: Daniel Llamocca