+

Máquinas de Estado Finitas

Las máquinas de estado pueden ser:
SÍNCRONAS: Necesitan de la intervención de un pulso de reloj. Si la entrada participa también en la salida se denomina Máquina de estado de Mealy, y si no participa se denomina de Moore.
ASÍNCRONAS: No necesitan de la intervención de un pulso de reloj. Estos circuitos evolucionan cuando cambian las entradas.

Una máquina de estado finita o FSM representa un sistema como un conjunto de estados, transiciones entre estos estados, que dependen de las entradas, conjuntamente con las salidas y las entradas asociadas.
De modo tal que una máquina de estado es una representación, de un circuito secuencial particular.

Cualquier circuito con memoria puede ser considerado como una FSM. Aún una computadora puede ser considerada como una gran FSM.

El diseño de una FSM involucra lo siguiente:

- Definición de estados.
- Definición de transición entre estados (dependientes de las entradas de la máquina).
- Optimización/minimización.

Definición de términos

Diagrama de estado: ilustra la forma y funcionamiento de la máquina de estado. Usualmente se dibuja como un diagrama de burbujas y flechas.

Estado: un conjunto identificable y único de valores medidos en diversos puntos de un sistema digital.

Ramificación: El cambio del estado presente al estado siguiente.

Estado siguiente: es el estado hacia el cual la máquina de estado realiza la siguiente transición, determinada por las entradas presentes cuando el dispositivo es secuenciado por un clock.

Máquina de Moore: es una máquina de estado que determina sus salidas solamente dependiendo de los estados presentes de la máquina.

Máquina de Mealy: es una máquina de estado que determina sus salidas dependiendo de los estados presentes de la máquina y de las entradas.

Estado presente y estado siguiente:

Para un estado determinado existe un número finito de posibles estados siguientes. Para cada ciclo de reloj la máquina de estado se ramifica al estado siguiente. Uno de los posibles estados siguientes se convierte en el nuevo estado presente. Dependiendo de las entradas presentes en el ciclo de reloj.

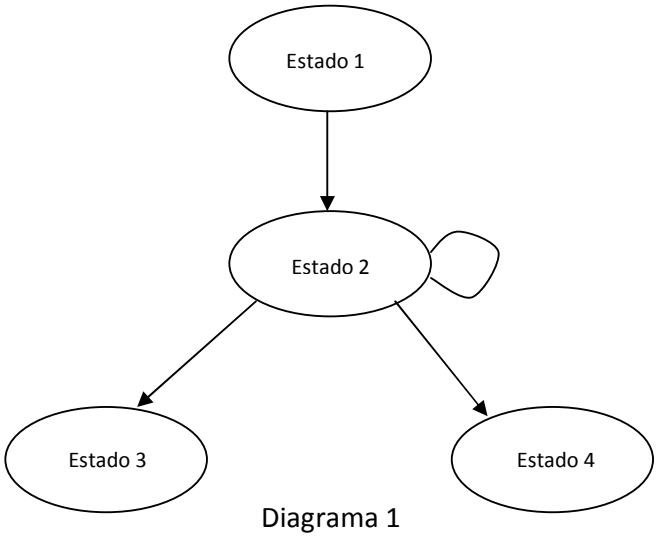


Diagrama 1

En un diagrama correcto, todas las posibles transiciones serán visibles incluyendo lazos realimentados en el mismo estado. En el diagrama 1 se puede deducir que si el estado presente es el Estado 2, el estado previo es el Estado 1 o el Estado 2 y el próximo estado debe ser 2, 3 o 4.

Máquinas de Moore y Mealy

Ambos tipos de máquinas siguen las características de las máquinas de estado pero difieren en la forma en que las salidas son producidas.

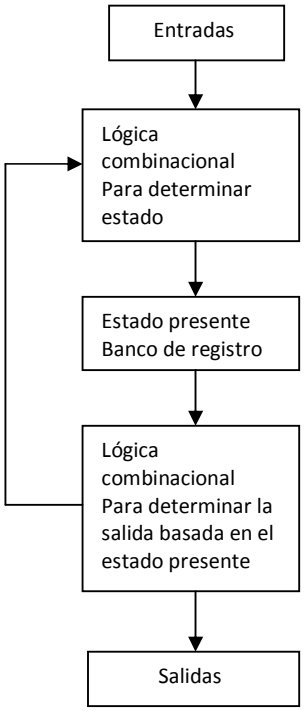
Máquina de Moore:
Las salidas son independientes de las entradas. Las salidas se producen efectivamente desde dentro del estado de la máquina. Se define como maquina tipo Moore si sus salidas solo dependen del estado de la máquina.

Máquina de Mealy: las salidas pueden ser determinadas por el estado presente solamente, o por el estado presente y las entradas presentes, es decir las salidas se producen dependiendo de cómo la máquina realiza una transición de un estado a otro.

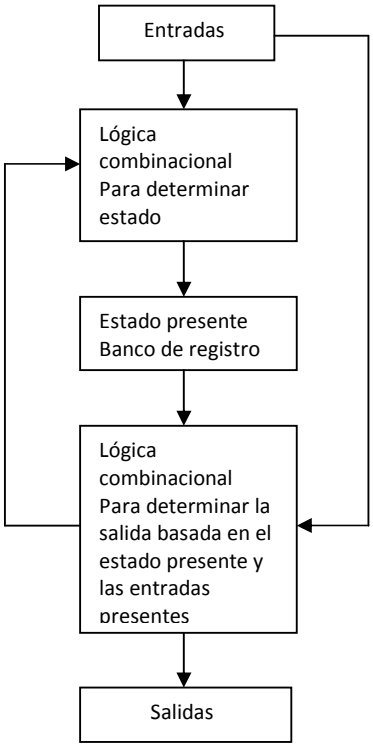
MODELO DE MÁQUINAS

Diagrama de Flujo

Máquina de Moore



Máquina de Mealy



Máquina de estado de Moore

Diagrama general

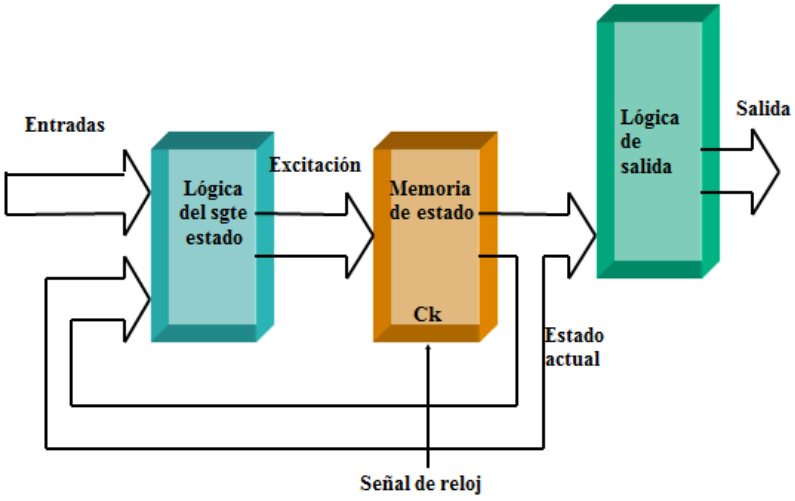
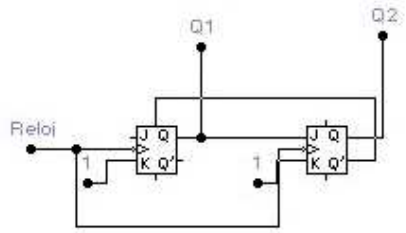


Diagrama a nivel Flip Flop y compuertas



Máquina de estado de Mealy

Diagrama general

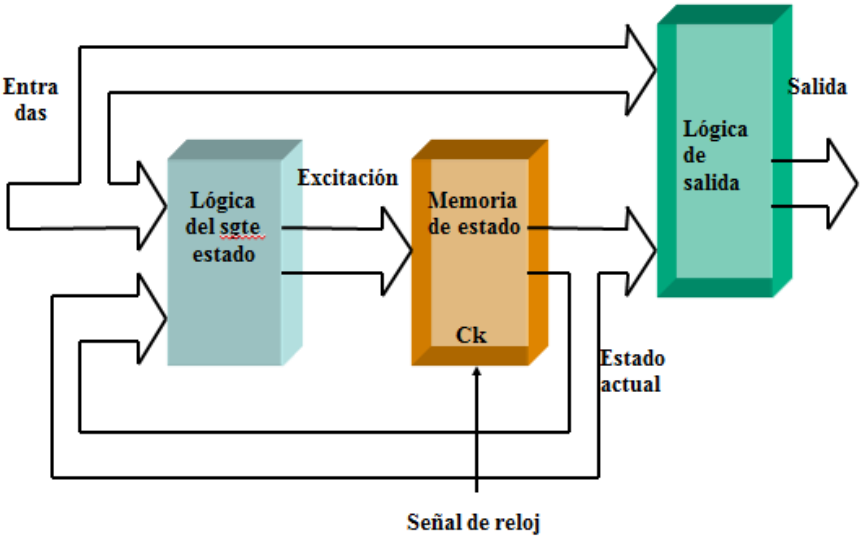


Diagrama a nivel Flip Flop y compuertas

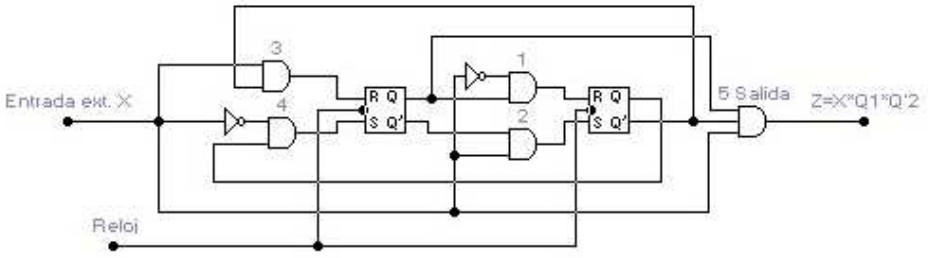
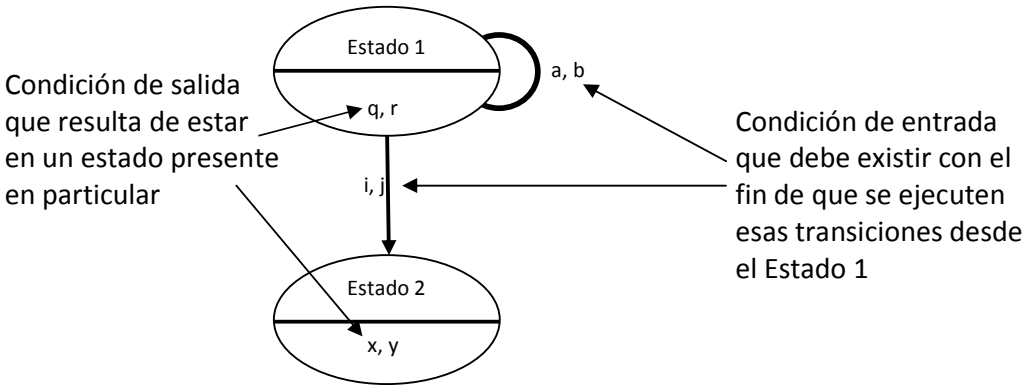


Diagrama de Máquina de estado de Moore



La salida en una Máquina de estado de Moore se muestra dentro de la burbuja de estado, debido a que la salida se mantiene igual, mientras la máquina se mantenga en ese estado. La salida puede ser arbitrariamente compleja pero debe ser la misma cada vez que la máquina entra a ese estado.

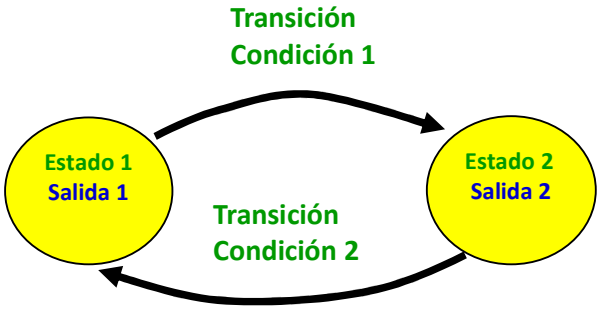
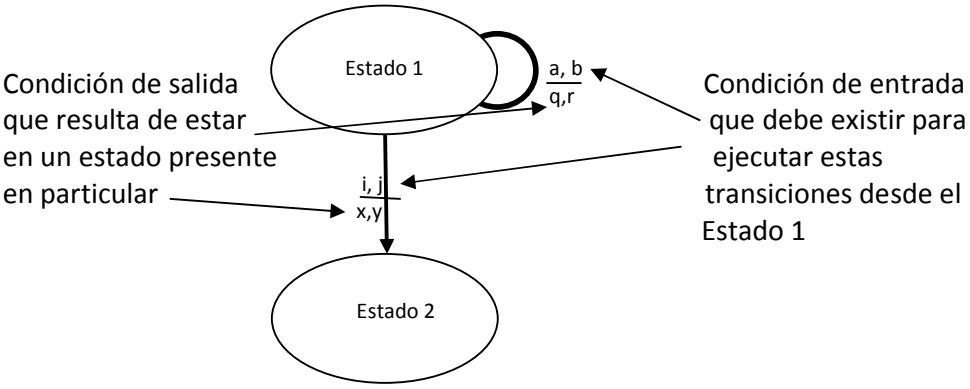


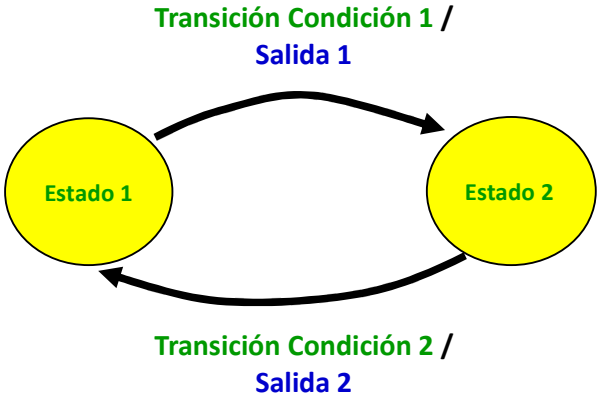
Diagrama de Máquina de estado de Mealy



La máquina de Mealy genera las salidas basada en:

- El estado presente.
- Las entradas a la máquina.

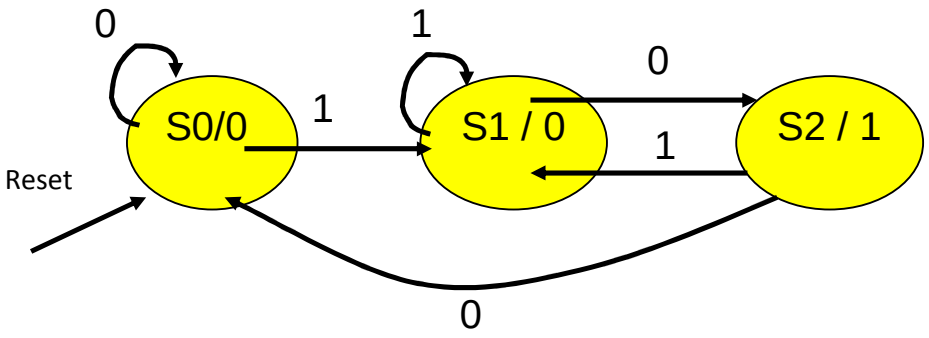
De modo de ser capaz de generar diversos patrones diferentes de señales de salida para el mismo estado, dependiendo de las entradas presentes en el ciclo de reloj.
Las salidas son mostradas sobre las transiciones del momento que están determinadas de la misma manera que el estado siguiente.



Diferencias entre Mealy y Moore

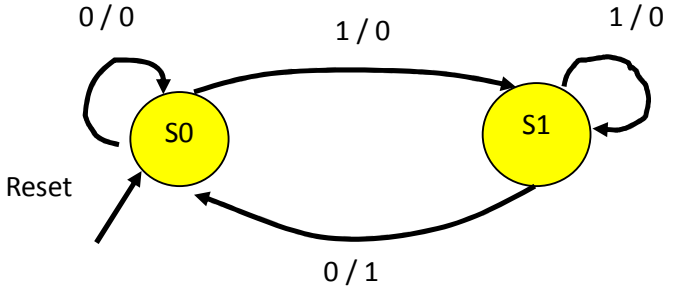
Las FSM Mealy y Moore pueden ser funcionalmente equivalentes.
Las FSM Mealy tienen una mejor descripción y usualmente requieren de un número menor de estados y menor área de circuito.
Mealy calcula las salidas tan pronto como se produce un cambio en las entradas.
Mealy responde un ciclo de reloj antes que el equivalente Moore.
Las FSM Moore no posee un camino de lógica combinacional entre entradas y salidas.

Ejemplo de FSM Moore
Detección de la secuencia 10.



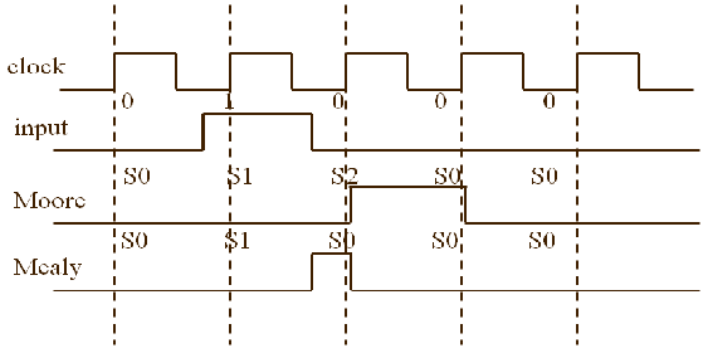
Significado de los estados:
S0: No se encuentra ningún elemento de la secuencia.
S1: aparece un “1”.
S2 Se detecta la secuencia “10”.

Ejemplo de FSM Mealy
Detección de la secuencia 10.



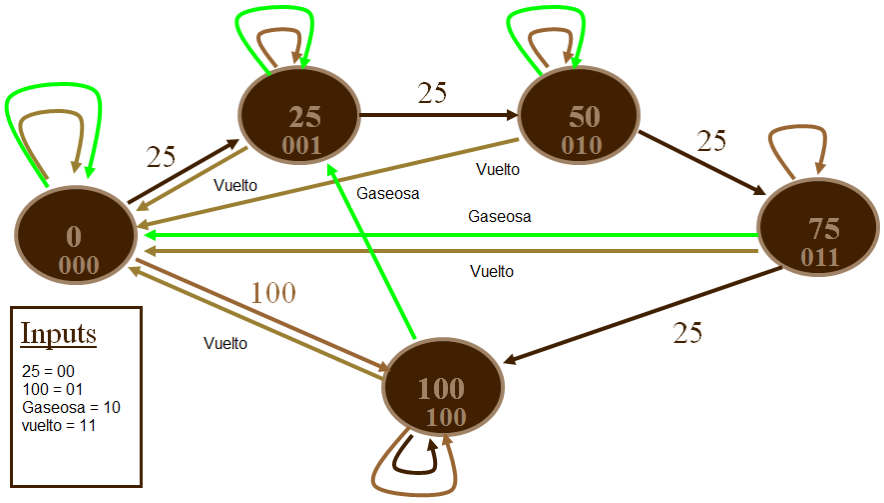
Significado de los estados:
S0: No se encuentra ningún elemento de la secuencia.
S1 Se detecta la secuencia “10”.

Diagrama de tiempos del ejemplo 1



Ejemplo de máquina expendedora de gaseosa

- Toma solamente monedas de 25 centavos y $1.
- No retiene más de $1.
- La gaseosa cuesta $0.75.
- Posibles acciones(entradas):
  - Depositar $0.25 (25)
  - Depositar $1 ($).
  - Presionar el botón para pedir gaseosa (gaseosa).
  - Presionar el botón para pedir devolución del dinero (vuelto).
- Estado: descripción del seteo interno de la máquina, por ej. cuánto dinero ha sido depositado y no gastado.
- Estados finitos: 0, 25, 50, 75, 100.
- Reglas: determinar cuántas entradas pueden cambiar el estado.



Estado			Entrada		Nuevo Estado		
S2	S1	S0	I0	I1	S2	S1	S0
0	0	0	0	0	0	0	1
0	0	1	0	0	0	1	0
0	1	0	0	0	0	1	1
0	1	1	0	0	1	0	0
1	0	0	0	0	1	0	0
0	0	0	0	1	1	0	0
0	0	1	0	1	0	0	1
0	1	0	0	1	0	1	0
0	1	1	0	1	0	1	1
1	0	0	0	1	1	0	0

Estado			Entrada		Nuevo Estado		
S2	S1	S0	I0	I1	S2	S1	S0
0	0	0	1	0	0	0	0
0	0	1	1	0	0	0	1
0	1	0	1	0	0	1	0
0	1	1	1	0	0	0	0
1	0	0	1	0	0	0	1
0	0	0	1	1	0	0	0
0	0	1	1	1	0	0	0
0	1	0	1	1	0	0	0
0	1	1	1	1	0	0	0
1	0	0	1	1	0	0	0

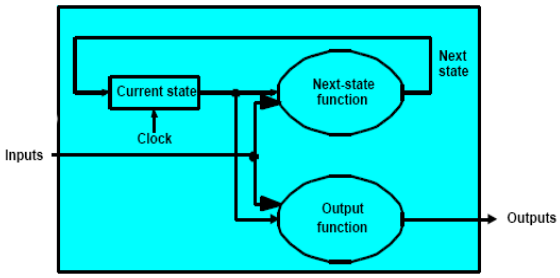
Estado			Entradas		Nuevo estado		
S2	S1	S0	I0	I1	S2	S1	S0
0	0	0	0	0	0	0	1
0	0	1	0	0	0	1	0
0	1	0	0	0	0	1	1
0	1	1	0	0	1	0	0
1	0	0	0	X	1	0	0
0	0	0	0	1	1	0	0
0	0	1	0	1	0	0	1
0	1	0	0	1	0	1	0
0	1	1	0	1	0	1	1

Estado			Entradas		Nuevo Estado		
S2	S1	S0	I0	I1	S2	S1	S0
0	0	0	1	0	0	0	0
0	0	1	1	0	0	0	1
0	1	0	1	0	0	1	0
0	1	1	1	0	0	0	0
1	0	0	1	0	0	0	1
X	X	X	1	1	0	0	0

MÁQUINAS DE ESTADO EN VHDL

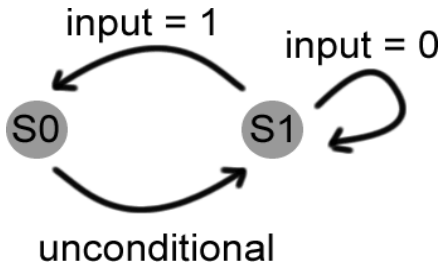
Como se explicó anteriormente existen dos tipos:

- Moore.
- Mealy.



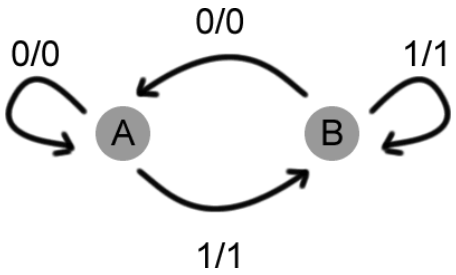
FSM Moore:

- La salida depende solamente del estado actual.
- Las salidas asociadas con cada estado, son establecidas en la transición del clock.



FSM Mealy:

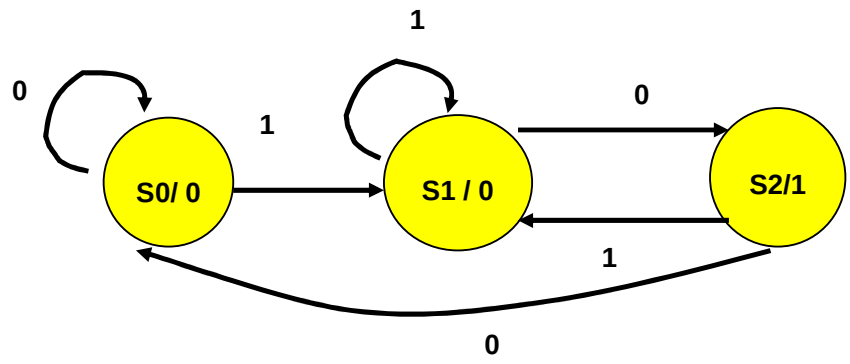
- La salida depende de las entradas y del estado actual.
- Las salidas son establecidas durante las transiciones.



Código VHDL para una FSM

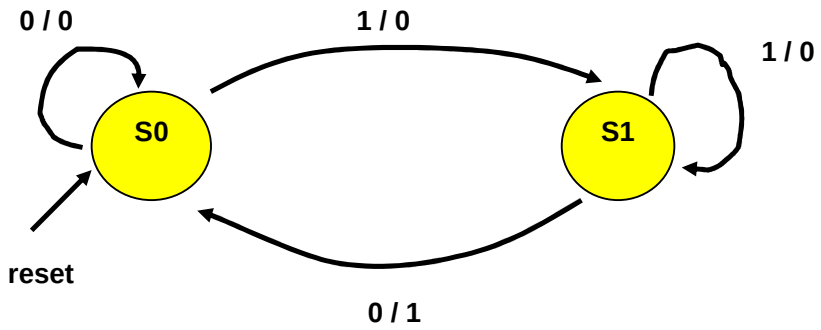
- Una FSM puede ser fácilmente descripta por medio del PROCESS.
- Las herramientas de síntesis interpretan una la descripción de unas FSM si se siguen ciertas reglas:
  - Las transiciones de estados deben ser descriptas en un PROCESS sensible solamente a las señales de CLOCK y RESET asíncrono.
  - Las salidas descriptas como sentencias concurrentes fuera del PROCESS.

FSM Moore reconocimiento de secuencia 10



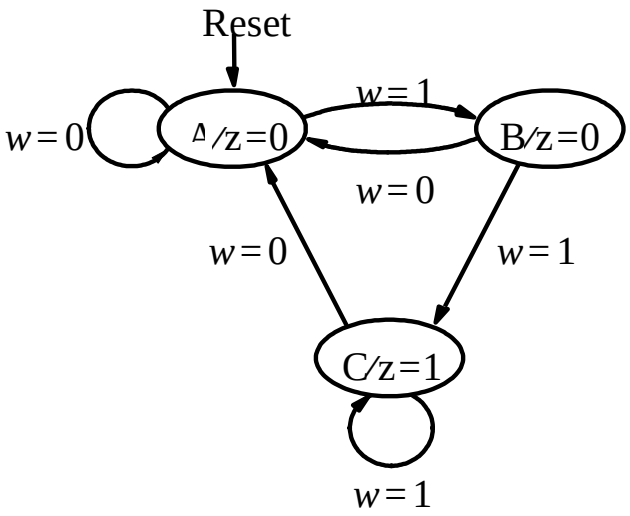

```
type state is (S0, S1, S2);
signal Moore_state: state;
U_Moore: process(clock, reset)
Begin
      if(reset = '1') then
            Moore_state <= S0;
      elsif (clock = '1' and clock'event) then
            case Moore_state is
                  when S0 =>
                    if input = '1' then Moore_state <= S1; end if;
                  when S1 =>
                    if input = '0' then Moore_state <= S2; end if;
                  when S2 =>
                    if input = '0' then Moore_state <= S0;
                    else Moore_state <= S1; end if;
            end case;
      end if;
End process;
Output <= '1' when Moore_state = S2 else '0';
```

FSM Mealy reconocimiento de secuencia 10

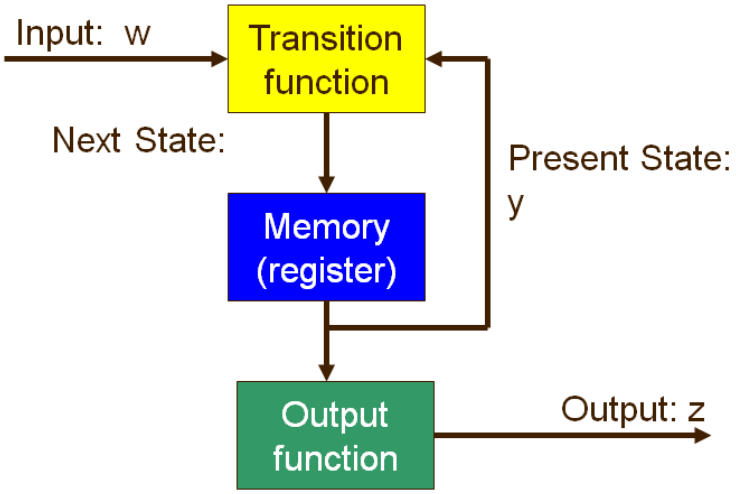



```
type state is (S0, S1);
signal Mealy_state: state;
U_Mealy: process(clock, reset)
Begin
      if(reset = '1') then
            Mealy_state <= S0;
      elsif (clock = '1' and clock'event) then
            case Mealy_state is
                  when S0 =>
                    if input = '1' then Mealy_state <= S1; end if;
                  when S1 =>
                    if input = '0' then Mealy_state <= S0; end if;
            end case;
      end if;
End process;
Output <= '1' when (Mealy_state = S1 and input = '0') else '0';
```

Ejemplo 2 FSM Moore:



Present state	Next state		Output z
	$w = 0$	$w = 1$	
A	A	B	0
B	A	C	0
C	A	C	1



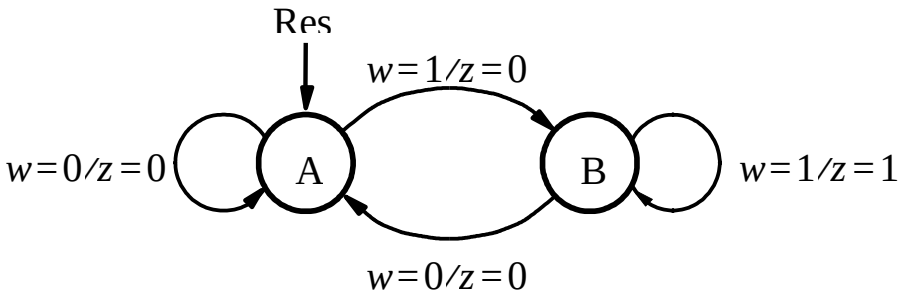
Código VHDL

```
USE ieee.std_logic_1164.all;
ENTITY simple IS
        PORT ( Clock, Resetn, w : IN  STD_LOGIC ;
                        z: OUT STD_LOGIC ) ;
END simple;
ARCHITECTURE Behavior OF simple IS
        TYPE State_type IS (A, B, C) ;
        SIGNAL y: State_type;
BEGIN
        PROCESS (Resetn, Clock)
        BEGIN
                IF Resetn = '0' THEN
                        y <= A;
                ELSIF (Clock'EVENT AND Clock = '1') THEN

                        CASE y IS
                                WHEN A =>
                                        IF w = '0' THEN
                                                y <= A;
                                        ELSE
                                                y <= B;
                                        END IF;
                                WHEN B =>
                                        IF w = '0' THEN
                                                y <= A;
                                        ELSE
                                                y <= C;
                                        END IF;
                                WHEN C =>
                                        IF w = '0' THEN
                                                y <= A;
                                        ELSE
                                                y <= C;
                                        END IF;
                        END CASE;
                END IF;
        END PROCESS;
        z <= '1' WHEN y = C ELSE '0';
END Behavior;
```

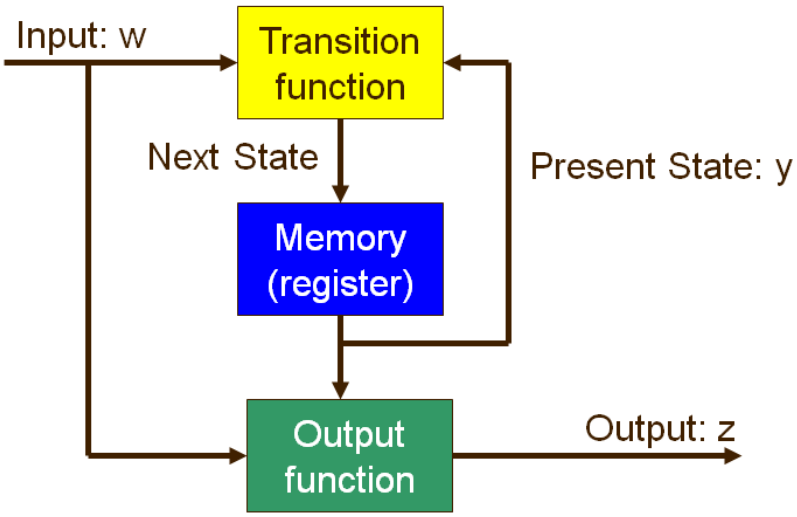
Código alternativo

```
ARCHITECTURE Behavior OF simple IS
        TYPE State_type IS (A, B, C);
        SIGNAL y_present, y_next: State_type;
BEGIN
        PROCESS (w, y_present)
        BEGIN
                CASE y_present IS
                        WHEN A =>
                                IF w = '0' THEN
                                        y_next <= A;
                                ELSE
                                        y_next <= B;
                                END IF;
                        WHEN B =>
                                IF w = '0' THEN
                                        y_next <= A;
                                ELSE
                                        y_next <= C;
                                END IF;

                        WHEN C =>
                                IF w = '0' THEN
                                        y_next <= A;
                                ELSE
                                        y_next <= C;
                                END IF;
                END CASE;
        END PROCESS;

        PROCESS (Clock, Resetn)
          BEGIN
                IF Resetn = '0' THEN
                        y_present <= A;
                ELSIF (Clock'EVENT AND Clock = '1') THEN
                        y_present <= y_next;
                END IF;
          END PROCESS;

        z <= '1' WHEN y_present = C ELSE '0';

END Behavior;
```

Ejemplo 2 FSM Mealy



Present state	Next state		Output z	
	w = 0	w = 1	w = 0	w = 1
A	A	B	0	0
B	A	B	0	1



Código VHDL

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY mealy IS
        PORT (Clock, Resetn, w: IN STD_LOGIC;
                        z: OUT STD_LOGIC) ;
END mealy;
ARCHITECTURE Behavior OF mealy IS
        TYPE State_type IS (A, B);
        SIGNAL y: State_type;
BEGIN
        PROCESS (Resetn, Clock)
        BEGIN
                IF Resetn = '0' THEN
                        y <= A;
                ELSIF (Clock'EVENT AND Clock = '1') THEN
                        CASE y IS
                                WHEN A =>
                                        IF w = '0' THEN y <= A;
                                        ELSE y <= B;
                                        END IF;

                                WHEN B =>
                                        IF w = '0' THEN y <= A;
                                        ELSE y <= B;
                                        END IF;
                        END CASE;
                 END IF;
         END PROCESS;
         with y select
                z <= w when B,
                z <= '0' when others;
END Behavior;
```