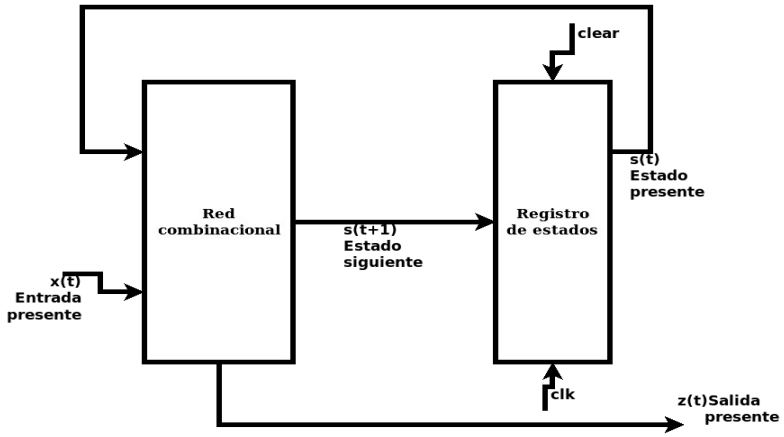
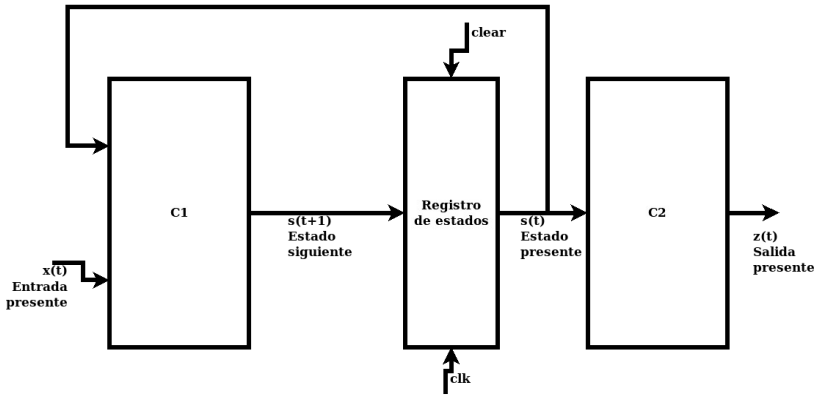
Máquinas de estados finitos JLB©

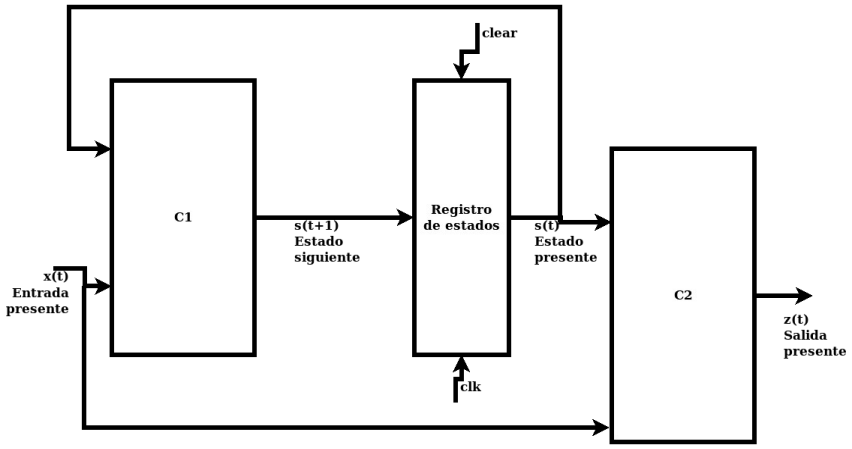
Una red secuencial canónica puede ser descrita por el diagrama de la figura:



Para representar y describir el modelo de la máquina de estados en un lenguaje de hardware, es conveniente añadir dos bloques combinacionales, como muestra la figura para la **máquina de Moore**, en donde la salida presente z(t) depende sólo del estado presente :

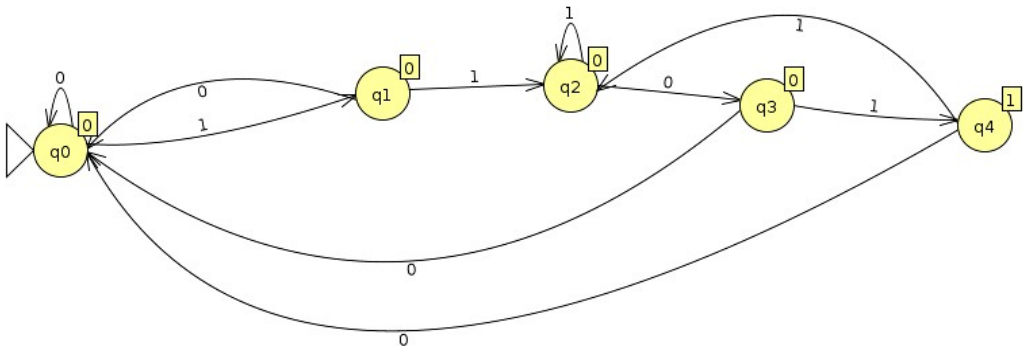


Esta máquina de estados en la cuál la salida presente z(t) depende del estado presente s(t) y de la entrada presente x(t), se llama ***máquina de Mealy***.



Ejemplo 1: Diseñar una máquina de estados finitos que envíe un "1" lógico cuando detecte la secuencia 1101. Elaborar su diagrama de estados y posteriormente su código en VHDL, usando el modelo de Moore y luego el de Mealy.

Diagrama de estados del modelo de ***Moore***:



Código en VHDL:

Librerías:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL; --Incluir esta librería para el prescaler
```

Entidad:

```
entity Seq_detect is
    Port ( clk : in  STD_LOGIC;
           reset : in  STD_LOGIC;
           input : in  STD_LOGIC;
           clk_1Hz   : out std_logic; --Salida temporal para visualizar el reloj
           output : out  STD_LOGIC);
end Seq_detect;
```

Señales intermedias:

```
architecture Behavioral of Seq_detect is
--Señales y tipos para los estados
   type state_type is (s0,s1,s2,s3,s4);
   signal present_state, next_state : state_type;

--Señales temporales para el prescaler
   signal count : std_logic_vector(25 downto 0);
   signal slow_clk : std_logic;
```

Divisor de frecuencia a 1 Hz:

```
begin
--Prescaler o divisor de frecuencia
   count <= count+1 when rising_edge(clk);
   slow_clk <= count(25);
   clk_1Hz <= slow_clk;
```

Bloque de registro de estados:

```
   --Proceso secuencial
   state_register: process (slow_clk,reset)
   begin
      if (reset = '1') then
         present_state <= s0;
      elsif (rising_edge(slow_clk)) then
         present_state <= next_state;
      end if;
   end process state_register;
```

Bloque del proceso combinacional de entrada C1:

```
   --Proceso combinacional 1
   C1: process (present_state,input)
   begin
      case present_state is
         when s0 =>
            if (input = '1') then
               next_state <= s1;
            else
               next_state <= s0;
            end if;
         when s1 =>
            if (input = '1') then
               next_state <= s2;
            else
               next_state <= s0;
            end if;
         when s2 =>
            if (input = '0') then
               next_state <= s3;
            else
               next_state <= s2;
            end if;
         when s3 =>
            if (input = '1') then
               next_state <= s4;
            else
               next_state <= s0;
            end if;
         when s4 =>
            if (input = '0') then
               next_state <= s0;
            else
               next_state <= s2;
            end if;
         when others =>
            null;
      end case;
   end process C1;
```

Proceso combinacional de salida C2:

```
   --Proceso combinacional 2
   C2: process (present_state)
   begin
      if (present_state = s4) then
         output <= '1';
      else
         output <= '0';
      end if;
   end process C2;

end Behavioral;
```

Asignación de pines en la tarjeta **BASYS2**:

```
NET "clk" LOC = "B8";
NET "clk_1Hz" LOC = "M11"; //LD1 indica reloj de 1Hz
NET "reset" LOC = "G12"; //BTN0
NET "input" LOC = "C11"; //BTN1
NET "output" LOC = "M5"; //LD0 Indica salida
```

Diagrama general de la solución:

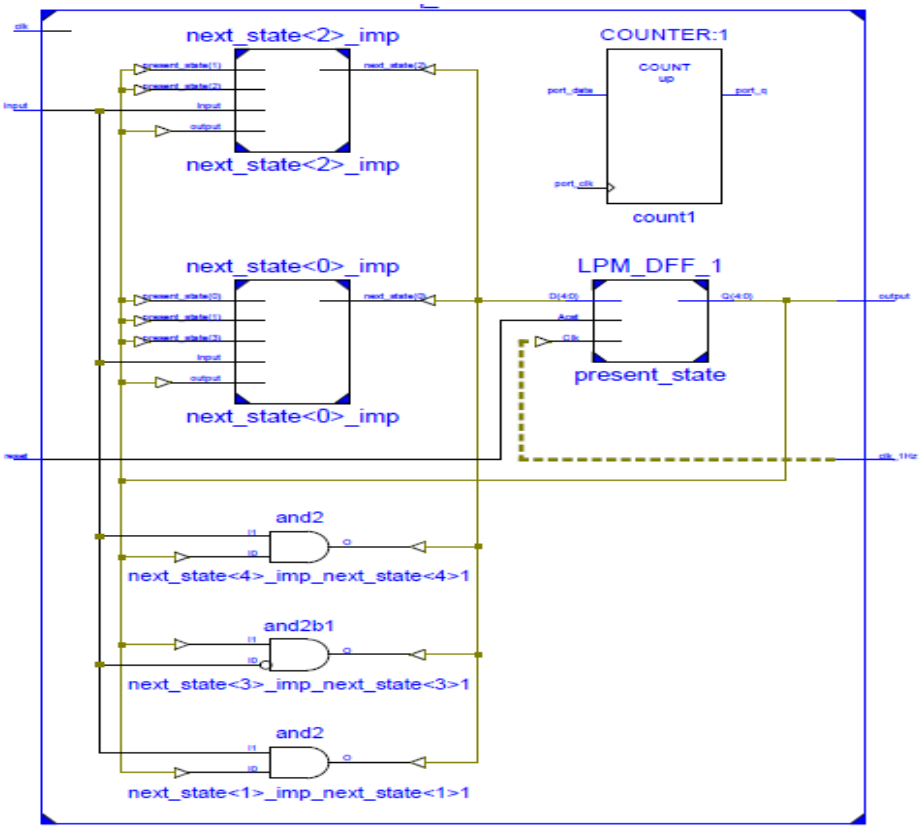
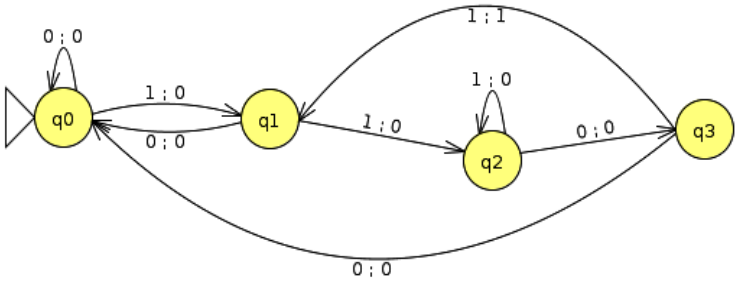


Diagrama de estados del modelo de ***Mealy***:



Es importante entender que el valor de salida desplegado es la salida combinacional que depende de la entrada y del estado presente. Esto significa que la salida nunca tendrá registro (latch). Si queremos que el valor se mantenga, podemos añadir un flip-flop a la salida. Una manera equivalente de hacerlo en VHDL, es sustituir el componente combinacional (C2) por un componente secuencial.

Código en VHDL:

Librerías:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL; --Incluir esta librería para el prescaler
```

Entidad:

```
entity Seq_detect is
    Port ( clk : in  STD_LOGIC;
           reset : in  STD_LOGIC;
           input : in  STD_LOGIC;
           clk_1Hz   : out std_logic; --Salida temporal para visualizar el reloj
           output : out  STD_LOGIC);
end Seq_detect;
```

Señales intermedias:

```
architecture Behavioral of Seq_detect is
--Señales y tipos para los estados
   type state_type is (s0,s1,s2,s3);
   signal present_state, next_state : state_type;

--Señales temporales para el prescaler
   signal count : std_logic_vector(25 downto 0);
   signal slow_clk : std_logic;
```

Divisor de frecuencia a 1 Hz:

```
begin
--Prescaler o divisor de frecuencia
   count <= count+1 when rising_edge(clk);
   slow_clk <= count(25);
   clk_1Hz <= slow_clk;
```

Bloque de registro de estados:

```
   --Proceso secuencial
   state_register: process (slow_clk,reset)
   begin
      if (reset = '1') then
         present_state <= s0;
      elsif (rising_edge(slow_clk)) then
         present_state <= next_state;
      end if;
   end process state_register;
```

Bloque del proceso combinacional de entrada C1:

```
   --Proceso combinacional 1
   C1: process (present_state,input)
   begin
      case present_state is
         when s0 =>
            if (input = '1') then
               next_state <= s1;
            else
               next_state <= s0;
            end if;
         when s1 =>
            if (input = '1') then
               next_state <= s2;
            else
               next_state <= s0;
            end if;
         when s2 =>
            if (input = '0') then
               next_state <= s3;
            else
               next_state <= s2;
            end if;
         when s3 =>
            if (input = '1') then
               next_state <= s1;
            else
               next_state <= s0;
            end if;
         when others =>
            null;
      end case;
   end process C1;
```

Proceso secuencial de salida (C2 sustituido por Seq2 en la máquina de Mealy):

```
   --Proceso secuencial 2 para registrar la salida
   Seq2: process (slow_clk,reset)
   begin
      if (reset = '1') then
         output <= '0';
      elsif (rising_edge(slow_clk)) then
         if ((present_state = s3) and (input = '1')) then
               output <= '1';
         else
               output <= '0';
         end if;
      end if;
   end process Seq2;

end Behavioral;
```

Asignación de pines en la tarjeta **BASYS2**:

```
NET "clk" LOC = "B8";
NET "clk_1Hz" LOC = "M11"; //LD1 indica reloj de 1Hz
NET "reset" LOC = "G12"; //BTN0
NET "input" LOC = "C11"; //BTN1
NET "output" LOC = "M5"; //LD0 Indica salida
```

Diagrama general de la solución: