# Tema 7. Análisis de Circuitos Secuenciales

- Máquinas de estado finito (FSM).
- Análisis de circuitos secuenciales síncronos.
- Introducción al diseño de circuitos secuenciales síncronos.
- Contadores.
- Registros de desplazamiento.

# Máquinas de estado finito

- Existen problemas que no pueden resolverse con circuitos combinacionales, por ejemplo circuitos contadores: un circuito en el que sus salidas siguen una secuencia fija que cuando acaba vuelve a empezar, o circuitos que reciben sus datos en forma serial ordenados en distintos intervalos de tiempo, o circuitos que se utilizan para controlar como se aplican los datos en el tiempo y las operaciones a realizar con ellos.

- Estos circuitos se realizan con circuitos secuenciales. El modelo que se utiliza para estos problemas lógicos es la máquina de estados, que describe el problema mediante una serie de entradas, salidas y estados. El estado representa la situación interna del circuito en cada instante de tiempo en función de la secuencia de entradas introducida previamente. El número de estados de la máquina de estados debe ser finito para que el circuito sea realizable de ahí el nombre de FSM (Finite State Machine).

- Normalmente las FSMs se implementan con circuitos síncronos, que evolucionan en base a un flanco de una señal de reloj. En los circuitos síncronos se simplifica el modelo matemático para descripción de circuitos de forma que sean prácticamente independientes de problemas temporales.

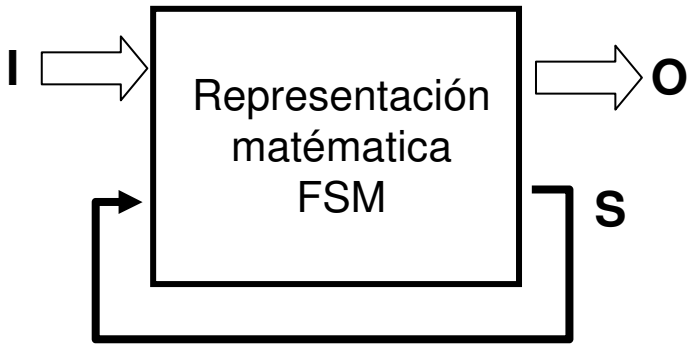
# Máquinas de estado finito

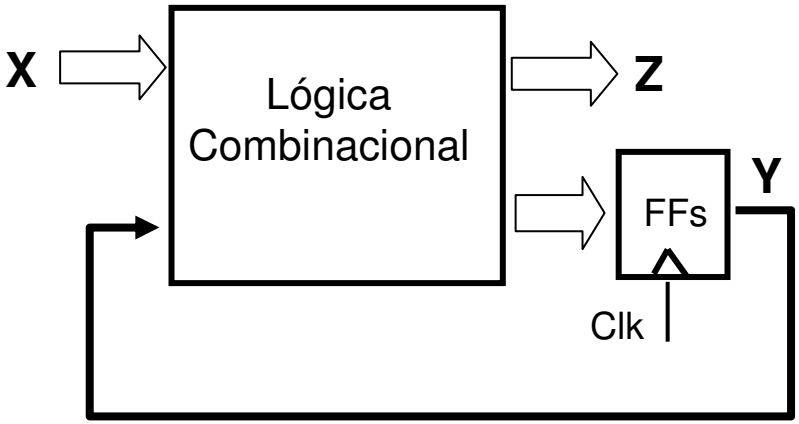
- En las FSMs el problema lógico se describe mediante:

  - Un conjunto de entradas I = {I1, …, Ip}
  - Un conjunto de salidas O = {O1, …, Oq}
  - Un conjunto finito de estados S = {S1, …, Sr}

- En la implementación circuital I, O y S son codificados por:

  - Un conjunto de variables de entrada X = {x1, … , xn}
  - Un conjunto de variables de salida Z = {z1, …, zm}
  - Un conjunto de variables de estado Y = {y1, …, yk}, que se almacenan en flip-flops.



# Máquinas de estado finito

- La evolución de la FSM se describe mediante las funciones de siguiente estado NS = G(I, S), que indican las transiciones entre estados en función del estado presente y de las entradas actuales (equivaldría a una representación S+ = G(I, S) en flip-flops.
  En la implementación circuital corresponden a lógica combinacional que genera el valor que deben tomar las entradas de los flip-flops, para que cada variable de estado *yi* realice una determinada transición al llegar el flanco de reloj, en función del valor actual de las variables de estado y de las entradas. Esta lógica se denomina decodificador del siguiente estado.

Y1 = G1(x1, …, xn, y1, …, yk)
Y2 = G2(x1, …, xn, y1, …, yk)
……
Yk = Gk(x1, …, xn, y1, …, yk)

J1 = GJ1(x1, …, xn, y1, …, yk)
K1 = GK1(x1, …, xn, y1, …, yk)

Para J-Ks

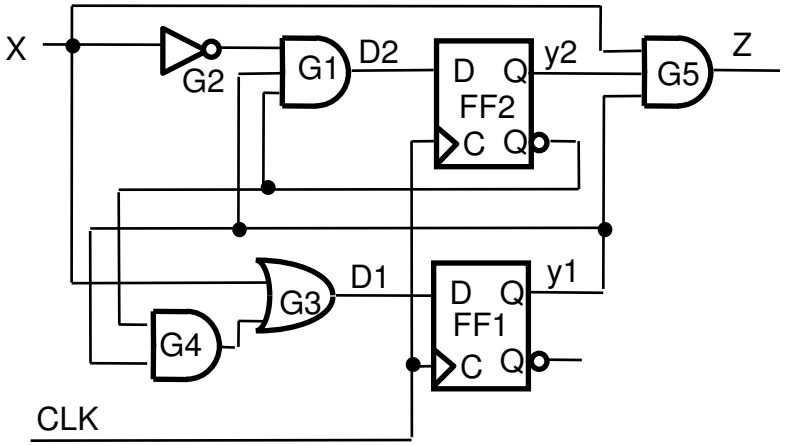
Jk = GJk(x1, …, xn, y1, …, yk)
Kk = GKk(x1, …, xn, y1, …, yk)

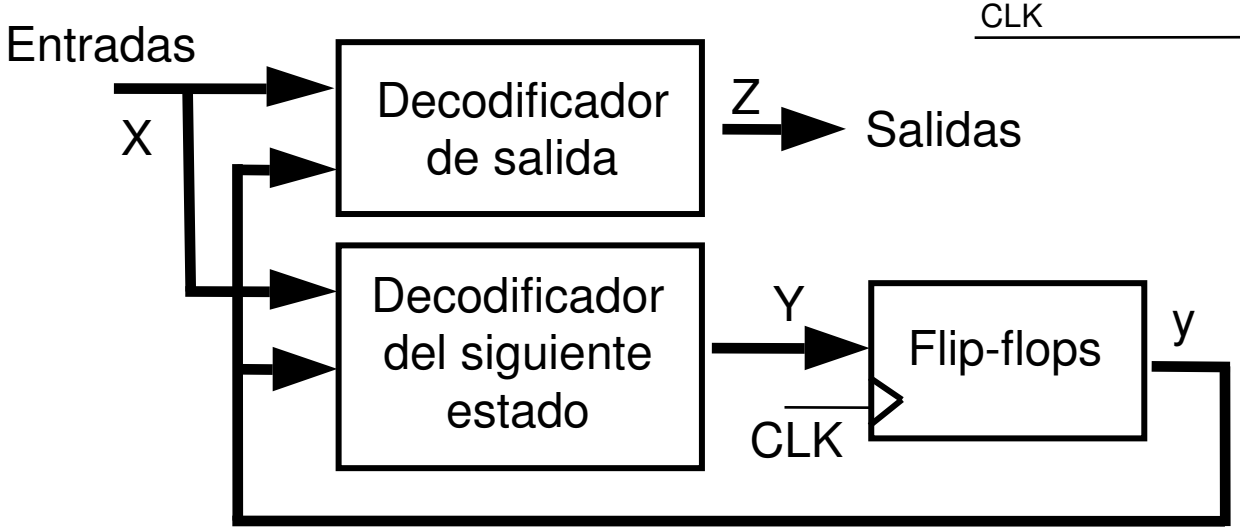
- Las salidas se obtienen como una función del estado actual y de las entradas O = F(I, S).
  En la implementación circuital corresponden a lógica combinacional que genera el valor que deben tomar las salidas Z. Existen dos tipos de implementaciones y, por tanto, dos tipos de FSMs.

# Máquinas de estado finito

- Máquina de Mealy: Z = F(X, Y). Las salidas dependen de las variables de estado y de las entradas.

z1 = F1(x1, …, xn, y1, …, yk)
z2 = F2(x1, …, xn, y1, …, yk)
……
zm = Fm(x1, …, xn, y1, …, yk)

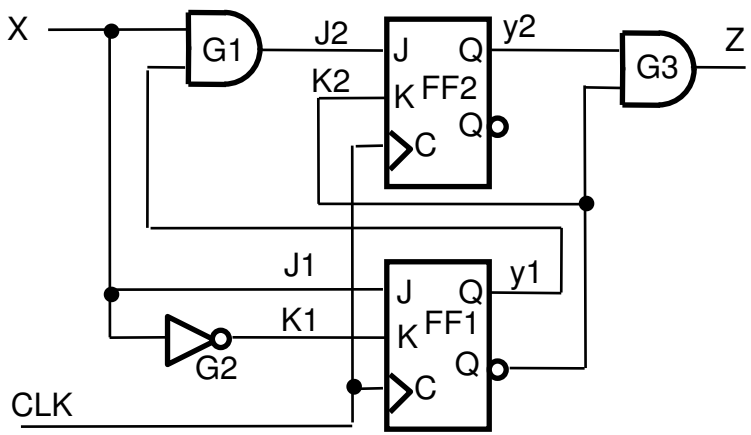


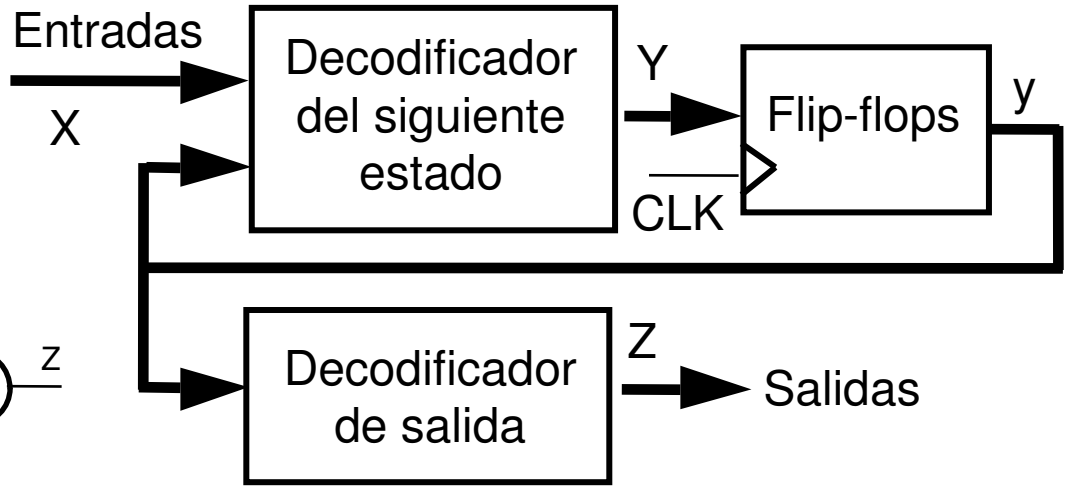



# Máquinas de estado finito

- Máquina de Moore: Z = F(Y). Las salidas dependen únicamente de las variables de estado. En principio está máquina es más “síncrona” que la máquina de Mealy, ya que las salidas permanecen estables todo el ciclo de reloj, por el contrario la definición de la FSM requiere más estados y un circuito más grande.
  En la máquina de Moore las entradas fijan la salida después del flanco de reloj (al cambiar el estado), en la máquina de Mealy antes de dicho flanco.

z1 = F1(y1, …, yk)
z2 = F2(y1, …, yk)
……
zm = Fm(y1, …, yk)

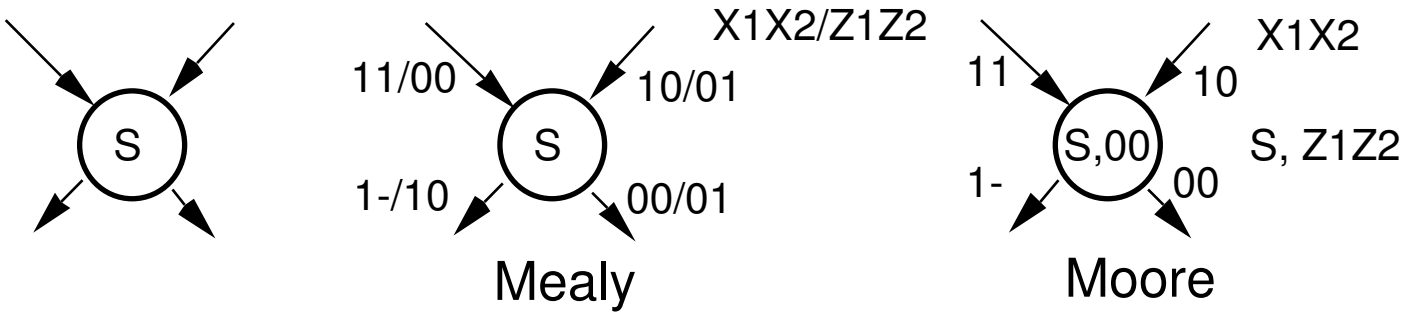


# Máquinas de estado finito

- En la fase de diseño de un problema lógico, las FSMs se representan por varios métodos: diagramas de estado, diagramas ASM, tablas de estado, descripciones HDL.

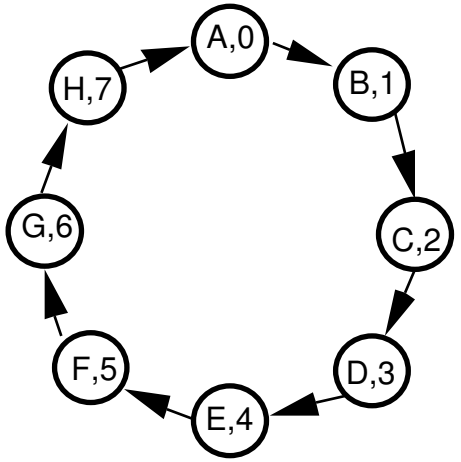
- El diagrama de estado es un método gráfico en el que cada estado se representa por un círculo, y las transiciones entre estados por flechas entre estados controladas por los valores de las entradas. En un circuito secuencial síncrono las transiciones entre estados se realizan cuando aparece un flanco en la señal de reloj.
  Un diagrama de estado debería ser “cerrado”: de cualquier estado se debería poder pasar a cualquier otro a través de una o de varias transiciones.
  Los valores de las salidas se indican en las transiciones si es una máquina de Mealy o dentro de los círculos se trata de una máquina de Moore.



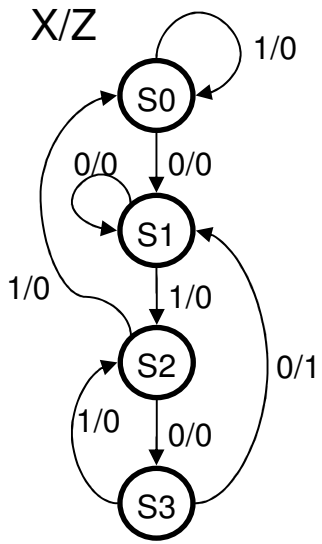
# Máquinas de estado finito

- Un contador se puede describir mediante una FSM de tipo Moore por un diagrama de estado cíclico.

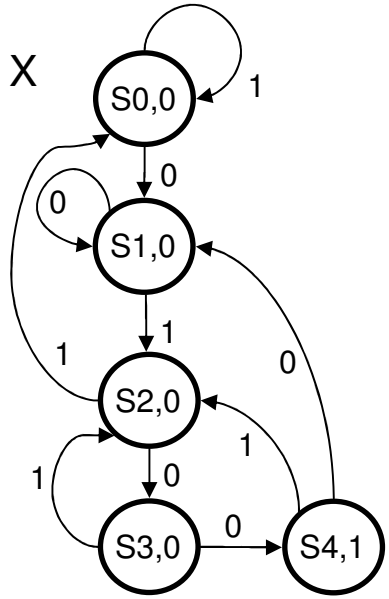
  Contador binario ascendente de 3 bits. Cuenta de 0 a 7, de 7 se pasa a 0 y se vuelve a empezar. No tiene entradas. Cada estado se nombra por una letra y se indica el valor que numérico que toma la salida en cada estado; en realidad la salida son tres bits que codifican en binario el valor mostrado en decimal.




- Detección de la secuencia 0100 en la entrada X. Entran datos en las entrada X continuamente, cuando los últimos 4 datos son 0100 la salida se fija a 1, en el resto de los casos permanece a 0.



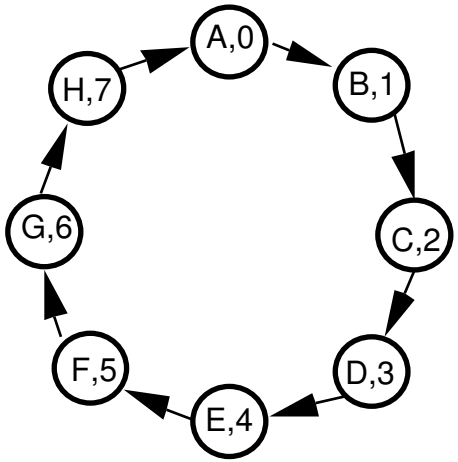

Mealy



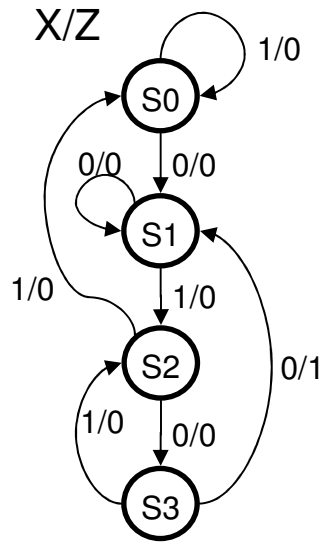

Moore

# Máquinas de estado finito

- La tabla de estados es una representación del diagrama de estados donde se indica en función del estado actual (por filas) y la entrada (cada valor en una columna), el valor del siguiente estado y la salida (por elemento en una máquina de Mealy, por filas en una máquina de Moore.
  Esta representación permite trabajar luego a nivel lógico.

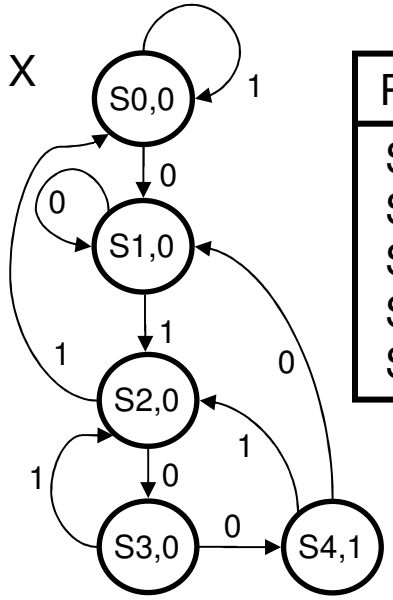



| P.S. | N.S. | Z3Z2Z1 |
|---|---|---|
| A | B | 0 0 0 |
| B | C | 0 0 1 |
| C | D | 0 1 0 |
| D | E | 0 1 1 |
| E | F | 1 0 0 |
| F | G | 1 0 1 |
| G | H | 1 1 0 |
| H | A | 1 1 1 |




N.S., Z

| P.S. | X = 0 | X = 1 |
|---|---|---|
| S0 | S1, 0 | S0, 0 |
| S1 | S1, 0 | S2, 0 |
| S2 | S3, 0 | S0, 0 |
| S3 | S1, 1 | S2, 0 |




N.S.

| P.S. | X = 0 | X = 1 | Z |
|---|---|---|---|
| S0 | S1 | S0 | 0 |
| S1 | S1 | S2 | 0 |
| S2 | S3 | S0 | 0 |
| S3 | S4 | S2 | 0 |
| S4 | S2 | S1 | 1 |

# Máquinas de estado finito

- La representación VHDL de estos circuitos puede hacerse a nivel de comportamiento usando el concepto de estado. El estado se puede definir mediante un nuevo tipo de datos enumerado que tiene como valores los nombres de los estados, y una señal para el valor del estado en el que se encuentra la FSM.

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

entity sec_0100  is
   port (CLK, Reset, X: in std_logic;
         Z: out std_logic);
end sec_0100;

architecture Mealy of sec_0100 is
-- Se define los estados
type estados is (S0, S1, S2, S3);
-- Señal que almacena el estado
signal state: estados := S0;
begin
-- Decodificador de salida
process (X, state)
   begin
   if ( state = S3 and X = '0' ) then
      Z <= '1';
   else Z <= '0';
end if;
end process;
```

```
-- Decodificador del siguiente estado
process (CLK, Reset)
   begin
    if ( Reset = '1') then
       state <= S0;
    elsif ( CLK'EVENT and CLK = '1' ) then
       case state is
          when S0 =>
             if ( X = '0' ) then state <= S1;
             end if;
          when S1 =>
             if ( X = '1' ) then state <= S2;
             end if;
          when S2 =>
             if ( X = '0' ) then state <= S3;
             else state <= S0;
             end if;
          when S3 =>
             if ( X = '0' ) then state <= S1;
             else state <= S2;
             end if;
         end case;
     end if;
  end process;
end Mealy;
```

# Máquinas de estado finito

```
architecture Moore of sec_0100 is
-- Se define los estados
type estados is (S0, S1, S2, S3, S4);
-- Señal que almacena el estado
signal state: estados := S0;
begin
-- Decodificador de salida
process (state)
   begin
   if ( state = S4) then
      Z <= '1';
   else Z <= '0';
end if;
end process;

-- Decodificador del siguiente estado
process (CLK, Reset)
   begin
    if ( Reset = '1') then
       state <= S0;
    elsif ( CLK'EVENT and CLK = '1' ) then
        case state is
        end case;
    end if;
  end process;
end Moore;
```

N.S.

| P.S. | X = 0 | X = 1 | Z |
|---|---|---|---|
| S0 | S1 | S0 | 0 |
| S1 | S1 | S2 | 0 |
| S2 | S3 | S0 | 0 |
| S3 | S4 | S2 | 0 |
| S4 | S2 | S1 | 1 |

```
when S0 =>
     if ( X = '0' ) then state <= S1;
     end if;
when S1 =>
      if ( X = '1' ) then state <= S2;
      end if;
when S2 =>
       if ( X = '0' ) then state <= S3;
       else state <= S0;
       end if;
when S3 =>
        if ( X = '0' ) then state <= S4;
        else state <= S2;
       end if;
when S4 =>
        if ( X = '0' ) then state <= S2;
        else state <= S1;
       end if;
```

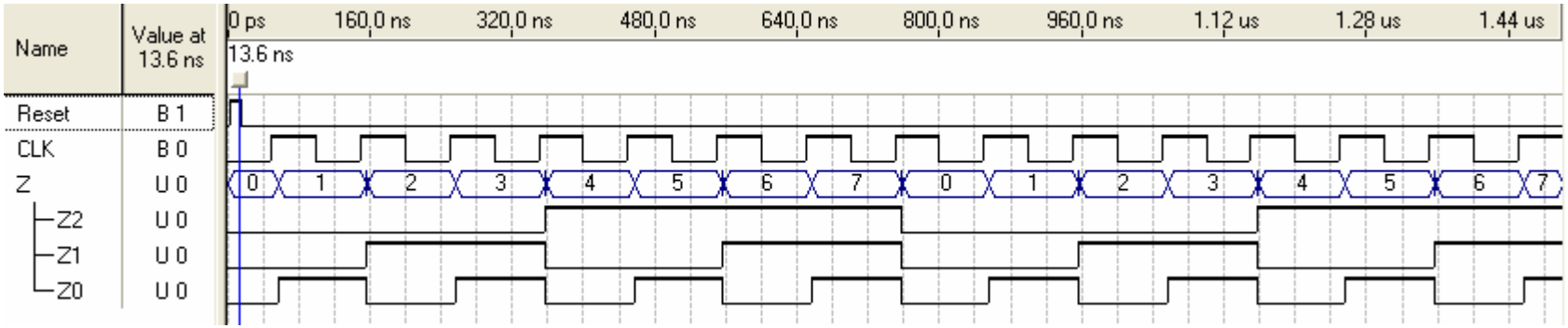
# Máquinas de estado finito

- El contador se puede describir mediante el mismo método, aunque se puede realizar una descripción más sencilla utilizando una propiedad de los contadores por la que los estados son codificados con el valor de las salidas.

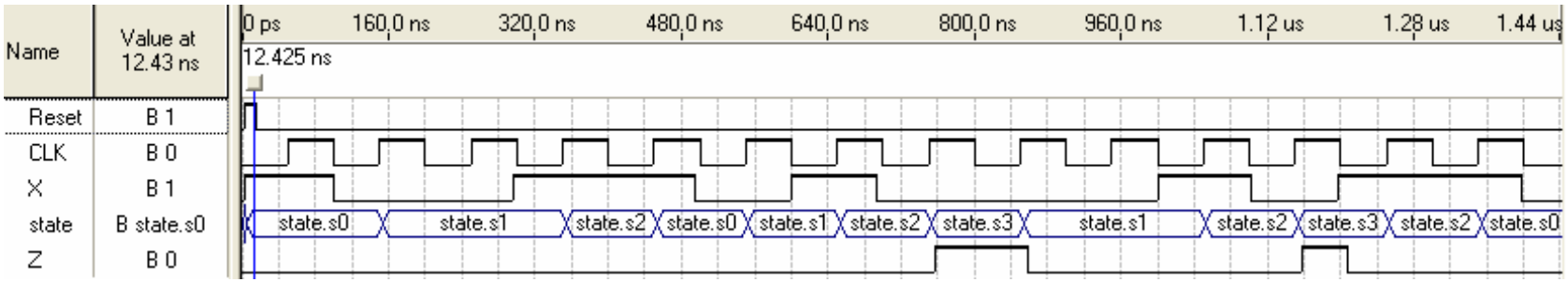
```
library ieee;
use ieee.std_logic_1164.all;

entity contador is
   port (CLK, Reset: in std_logic;
         Z2, Z1, Z0: out std_logic);
end contador;

architecture comp of contador is
-- Entradas de los flip-flops
signal Z_I: std_logic_vector (2 downto 0);
-- Salidas de los flip-flops
signal Z_O: std_logic_vector (2 downto 0);

begin
process (CLK, Reset)
   begin
    if ( Reset = '1') then
       Z_O <= "000";
    elsif ( CLK'EVENT and CLK = '1' ) then
       Z_O <= Z_I;
    end if;
end process;
```

```
-- Decodificador del siguiente estado
process (Z_O)
begin
    case Z_O is
         when "000" => Z_I <= "001";
         when "001" => Z_I <= "010";
         when "010" => Z_I <= "011";
         when "011" => Z_I <= "100";
         when "100" => Z_I <= "101";
         when "101" => Z_I <= "110";
         when "110" => Z_I <= "111";
         when "111" => Z_I <= "000";
         when others => Z_I <= "---";
    end case;
end process;

Z2 <= Z_O(2); Z1 <= Z_O(1); Z0 <= Z_O(0);
end comp;
```

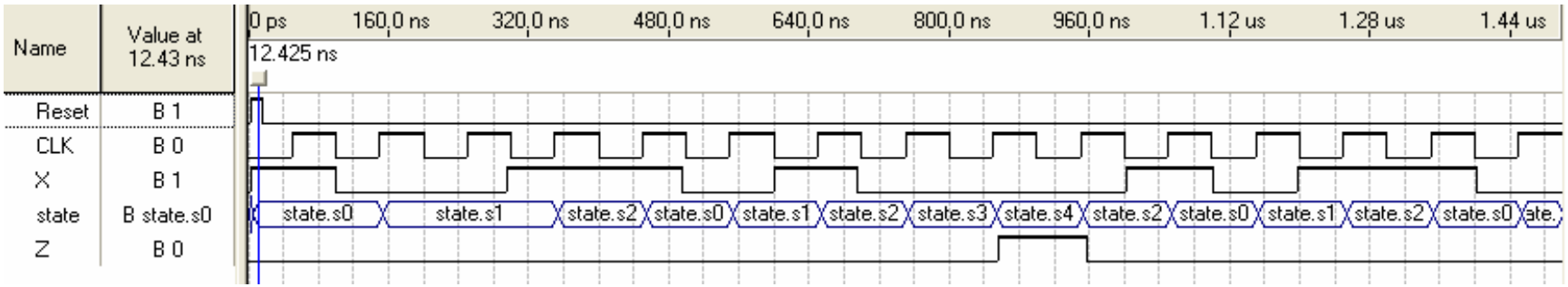
# Máquinas de estado finito




Contador




Mealy




Moore

# Análisis de circuitos secuenciales síncronos

- El análisis de un circuito secuencial síncrono consiste en el paso de una descripción estructural de un circuito mediante flip-flops y puertas lógicas a una descripción funcional de una FSM, principalmente una tabla de estados, y de ahí un diagrama de estados o una descripción VHDL. Hay que realizar estos pasos:

1. Determinar el número de estados. Dado un circuito con N flip-flops se dispone de N variables de estado y1, …, yN, y el número de estados posible de la FSM es $2^N$, que corresponde a cada codificación binaria distinta de las variables de estado. Existen otras codificaciones pero esta es la más habitual ("one-hot", una y sólo una variable de estado a 1 en la codificación => N variables de estado permiten N estados). Denominar los estados como Si, de S0 a S($2^N$ -1), y asociar a cada estado Si la codificación i en binario en los flip-flops.

| | y1 |
|---|---|
| S0 | 0 |
| S1 | 1 |

| | y2 | y1 |
|---|---|---|
| S0 | 0 | 0 |
| S1 | 0 | 1 |
| S2 | 1 | 0 |
| S3 | 1 | 1 |

| | y2 | y1 |
|---|---|---|
| S0 | 0 | 1 |
| S1 | 1 | 0 |

One-Hot

# Análisis de circuitos secuenciales síncronos

2. Dada la descripción estructural del circuito encontrar las funciones lógicas que definen el decodificador de salida y el decodificador del siguiente estado en función de las variables de estado y de las entradas.

3. Realizar una tabla de en la que en las filas se sitúa cada estado descrito mediante su codificación en binario en las variables de estado, y en cada columna cada posible combinación de valores lógicos en las entradas del circuito. Cada casilla de la tabla se debe rellenar con el valor de las entradas de cada flip-flop (en subcolumnas), obtenido a partir de las funciones del decodificador del siguiente estado.

   Determinar si se trata de una máquina de Mealy o de Moore y obtener los valores lógicos de las salidas mediante las funciones del decodificador de salida, y situar esos valores en la tabla como en una tabla de estados según el tipo de máquina que se trate (en cada fila y en cada columna si es tipo Mealy; en cada fila si es tipo Moore).

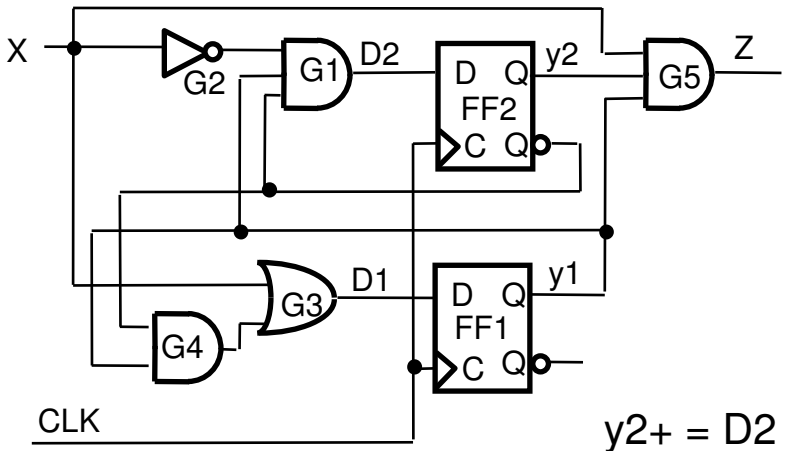
# Análisis de circuitos secuenciales síncronos

4. Convertir la tabla anterior en una nueva tabla con la misma relación de filas-columnas, situando en cada casilla los nuevos valores que se cargan en las variables de estado (al llegar el flanco de reloj), obtenidos para una variable de estado yi en función de los valores de las entradas del flip-flop i, del valor actual de la variable yi y de la tabla de operación del flip-flop i. Realmente se está haciendo yi+ = F(Inp, yi).

   Mantener las salidas como en la tabla anterior.

5. Generar la tabla de estados sustituyendo las combinaciones de valores en las variables de estado que aparecen en la tabla anterior por el nombre correspondiente del estado. Esta ya es una representación en alto nivel. Se puede desarrollar la tabla de estados en un diagrama de estados o en una descripción VHDL.

# Análisis de circuitos secuenciales síncronos




1

| | y2 | y1 |
|---|---|---|
| S0 | 0 | 0 |
| S1 | 0 | 1 |
| S2 | 1 | 0 |
| S3 | 1 | 1 |

2

$D2 = \overline{X}\ y1\ \overline{y2}$

$D1 = X + y1\ \overline{y2}$

$Z = X\ y1\ y2$

$y2+ = D2$

$y1+ = D1$

3

| | X = 0 | X = 1 |
|---|---|---|
| y2 y1 | D2D1, Z | D2D1, Z |
| 0 0 | 0 0, 0 | 0 1, 0 |
| 0 1 | 1 1, 0 | 0 1, 0 |
| 1 0 | 0 0, 0 | 0 1, 0 |
| 1 1 | 0 0, 0 | 0 1, 1 |

4

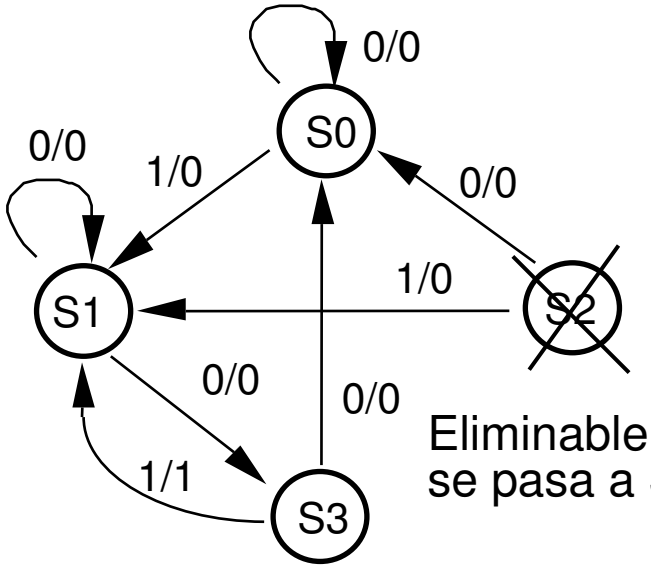
| | X = 0 | X = 1 |
|---|---|---|
| y2 y1 | y2+y1+,Z | y2+y1+,Z |
| 0 0 | 0 0, 0 | 0 1, 0 |
| 0 1 | 1 1, 0 | 0 1, 0 |
| 1 0 | 0 0, 0 | 0 1, 0 |
| 1 1 | 0 0, 0 | 0 1, 1 |

5

| | X = 0 | X = 1 |
|---|---|---|
| PS | NS,Z | NS, Z |
| S0 | S0, 0 | S1, 0 |
| S1 | S3, 0 | S1, 0 |
| S2 | S0, 0 | S1, 0 |
| S3 | S0, 0 | S1, 1 |

# Análisis de circuitos secuenciales síncronos

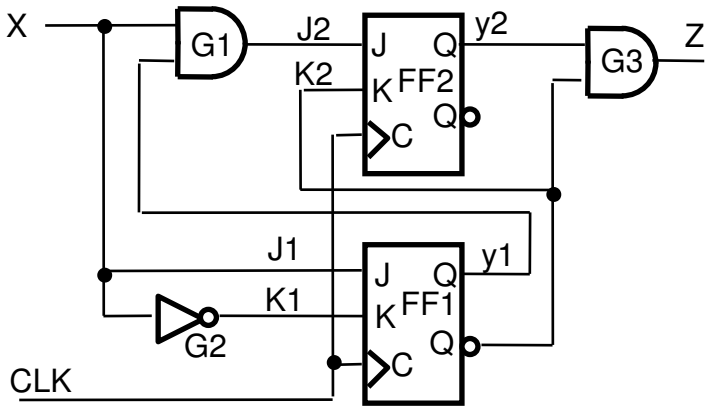
| | X = 0 | X = 1 |
|---|---|---|
| PS | NS,Z | NS, Z |
| S0 | S0, 0 | S1, 0 |
| S1 | S3, 0 | S1, 0 |
| S2 | S0, 0 | S1, 0 |
| S3 | S0, 0 | S1, 1 |




Eliminable. Nunca se pasa a S2

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

entity Sec_D  is
   port (CLK, X: in std_logic;
         Z: out std_logic);
end Sec_D;

architecture comp of Sec_D is
type estados is (S0, S1, S3);
signal state: estados := S0;

begin
Z <= '1' when (state = S3 and X = '1') else '0';
process (CLK)
    if ( CLK'EVENT and CLK = '1' ) then
       case state is
          when S0 =>  if X = '1' then state <= S1;
                      end if;
          when S1 =>  if X = '0' then state <= S3;
                      end if;
          when S3 =>  if X = '0' then state <= S0;
                      else state <= S1;
                      end if;
       end case;
   end if;
end process;

end comp;
```

# Análisis de circuitos secuenciales síncronos




① 

| | y2 | y1 |
|---|---|---|
| S0 | 0 | 0 |
| S1 | 0 | 1 |
| S2 | 1 | 0 |
| S3 | 1 | 1 |

② 

$J2 = X\,y1$

$K2 = \overline{y1}$

$J1 = X$

$K1 = \overline{X}$

$Z = y2\,\overline{y1}$

③ 

| | X = 0 | | X = 1 | | |
|---|---|---|---|---|---|
| y2 y1 | J2K2 | J1K1 | J2K2 | J1K1 | Z |
| 0 0 | 0 1 | 0 1 | 0 1 | 1 0 | 0 |
| 0 1 | 0 0 | 0 1 | 1 0 | 1 0 | 0 |
| 1 0 | 0 1 | 0 1 | 0 1 | 1 0 | 1 |
| 1 1 | 0 0 | 0 1 | 1 0 | 1 0 | 0 |

| J2 | K2 | y2+ |
|---|---|---|
| 0 | 0 | y2 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | $\overline{y2}$ |

| J1 | K1 | y1+ |
|---|---|---|
| 0 | 0 | y1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | $\overline{y1}$ |

# Análisis de circuitos secuenciales síncronos

3

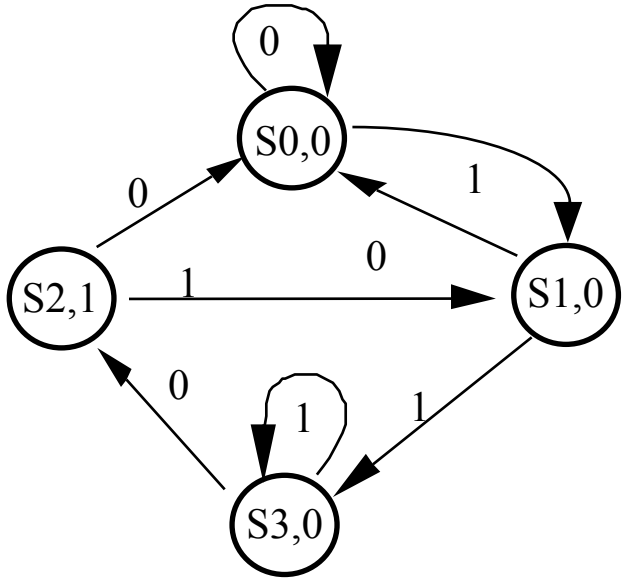
| | X = 0 | | X = 1 | | |
|---|---|---|---|---|---|
| y2 y1 | J2K2 | J1K1 | J2K2 | J1K1 | Z |
| 0 0 | 0 1 | 0 1 | 0 1 | 1 0 | 0 |
| 0 1 | 0 0 | 0 1 | 1 0 | 1 0 | 0 |
| 1 0 | 0 1 | 0 1 | 0 1 | 1 0 | 1 |
| 1 1 | 0 0 | 0 1 | 1 0 | 1 0 | 0 |

4

| | X = 0 | X = 1 | |
|---|---|---|---|
| y2 y1 | y2+y1+ | y2+y1+ | Z |
| 0 0 | 0 0 | 0 1 | 0 |
| 0 1 | 0 0 | 1 1 | 0 |
| 1 0 | 0 0 | 0 1 | 1 |
| 1 1 | 1 0 | 1 1 | 0 |

5

| | X = 0 | X = 1 | |
|---|---|---|---|
| PS | NS | NS | Z |
| S0 | S0 | S1 | 0 |
| S1 | S0 | S3 | 0 |
| S2 | S0 | S1 | 1 |
| S3 | S2 | S3 | 0 |



# Introducción al diseño de circuitos secuenciales síncronos

- El diseño de circuitos secuenciales parte de un problema lógico y el objetivo final es llegar a un circuito digital síncrono. Para ello hay que realizar los siguientes pasos de los que sólo la implementación final se desarrollará en esta introducción.

1. Describir el problema mediante un diagrama de estados o un método similar.

2. Minimizar el diagrama de estados. Las especificaciones del problema o la resolución del mismo pueden contener redundancias o situaciones no especificadas; se pueden aplicar métodos algorítmicos que minimicen el números de estados de una tabla. En esta introducción se supone que la descripción es mínima.

3. "Asignación secundaria óptima“: asignar codificaciones binarias a los estados de la tabla que minimicen o, al menos reduzcan en lo posible la lógica combinacional del circuito (decodificadores del siguiente estado y de salida). En esta introducción se supone una asignación secundaria aleatoria.

# Introducción al diseño de circuitos secuenciales síncronos

4. Representación de la tabla de estados del problema y su conversión a nivel lógico (0s y 1s) por la codificación binaria de cada estado de la tabla.

5. Generación de los decodificadores de salida y de siguiente estado del circuito utilizando un tipo de flip-flops previamente determinado. De la tabla de estados codificados en binario se pueden conocer las transiciones entre yi e yi+ para cada variable de estado y para cada combinación de las entradas. A partir de la tabla de excitación o transición de cada flip-flop se pueden conocer los valores lógicos que deben aplicarse a las entradas de los flip-flops para cada valor de las variables de estado y de las entradas. Por último se pueden calcular las funciones lógicas minimizadas para cada entrada de cada flip-flop (decodificador del siguiente estado), y también para las salidas.

6. Construir el circuito con puertas lógicas y flip-flops.

# Introducción al diseño de circuitos secuenciales síncronos

① 

| | X = 0 | X = 1 | |
|---|---|---|---|
| PS | NS | NS | Z |
| S0 | S0 | S1 | 0 |
| S1 | S0 | S3 | 0 |
| S2 | S0 | S1 | 1 |
| S3 | S2 | S3 | 0 |

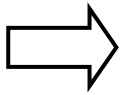
② Se supone que la tabla es mínima

③ Asignación secundaria aleatoria

| | y2 | y1 |
|---|---|---|
| S0 | 0 | 1 |
| S1 | 1 | 0 |
| S2 | 1 | 1 |
| S3 | 0 | 0 |

④

| | X = 0 | X = 1 | |
|---|---|---|---|
| PS | NS | NS | Z |
| S0 | S0 | S1 | 0 |
| S1 | S0 | S3 | 0 |
| S2 | S0 | S1 | 1 |
| S3 | S2 | S3 | 0 |



| | X = 0 | | X = 1 | | |
|---|---|---|---|---|---|
| y2 | y1 | y2+ | y1+ | y2+ | y1+ | Z |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 |

# Introducción al diseño de circuitos secuenciales síncronos

| | X = 0 | X = 1 | |
|---|---|---|---|
| y2 y1 | y2+y1+ | y2+y1+ | Z |
| 0 1 | 0 1 | 1 0 | 0 |
| 1 0 | 0 1 | 0 0 | 0 |
| 1 1 | 0 1 | 1 0 | 1 |
| 0 0 | 1 1 | 0 0 | 0 |

5

| | X = 0 | | X = 1 | | |
|---|---|---|---|---|---|
| y2 y1 | J2K2 | J1K1 | J2K2 | J1K1 | Z |
| 0 1 | 0 Ø | Ø 0 | 1 Ø | Ø 1 | 0 |
| 1 0 | Ø 1 | 1 Ø | Ø 1 | 0 Ø | 0 |
| 1 1 | Ø 1 | Ø 0 | Ø 0 | Ø 1 | 1 |
| 0 0 | 1 Ø | 1 Ø | 0 Ø | 0 Ø | 0 |

| y2 | y2+ | J2 | K2 |
|---|---|---|---|
| 0 | 0 | 0 | Ø |
| 0 | 1 | 1 | Ø |
| 1 | 0 | Ø | 1 |
| 1 | 1 | Ø | 0 |

| y1 | y1+ | J1 | K1 |
|---|---|---|---|
| 0 | 0 | 0 | Ø |
| 0 | 1 | 1 | Ø |
| 1 | 0 | Ø | 1 |
| 1 | 1 | Ø | 0 |

Con J-Ks. Tabla de excitación

| y2 \ y1 | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | (1) |

Z = y2 y1

| X \ y2y1 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 0 | Ø | Ø |
| 1 | 0 | 1 | Ø | Ø |

J2 = X y1 + $\overline{X}$ $\overline{y1}$

| X \ y2y1 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | Ø | Ø | 1 |
| 1 | 0 | Ø | Ø | 0 |

J1 = $\overline{X}$

| X \ y2y1 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | Ø | Ø | 1 | 1 |
| 1 | Ø | Ø | 0 | 1 |

K2 = $\overline{X}$ + $\overline{y1}$

| X \ y2y1 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | Ø | 0 | 0 | Ø |
| 1 | Ø | 1 | 1 | Ø |

K1 = X

# Introducción al diseño de circuitos secuenciales síncronos

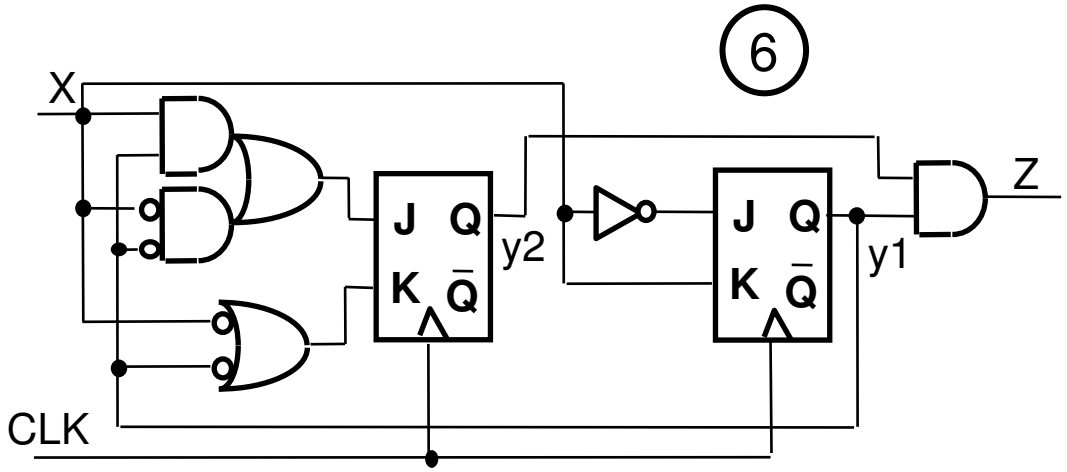
$J2 = X\,y1 + \overline{X}\,\overline{y1}$

$K2 = \overline{X} + \overline{y1}$

$J1 = \overline{X}$

$K1 = X$

$Z = y2\,y1$



# Contadores

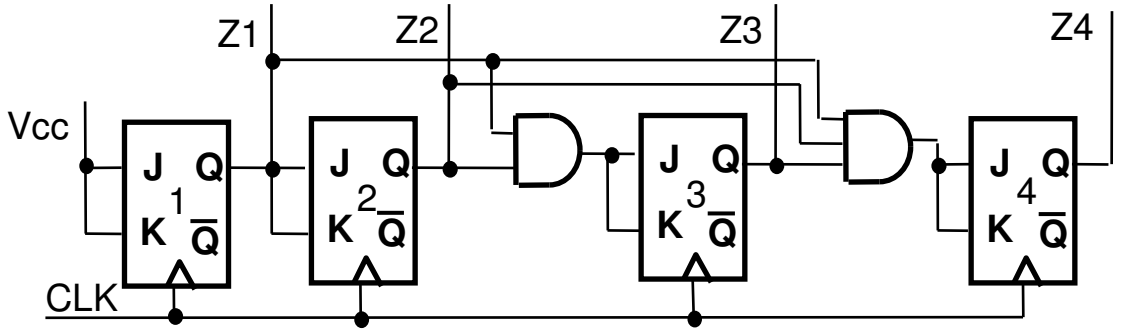
- Los contadores son unos circuitos específicos que realizan una cuenta o secuencia fija de salida, pasando de un elemento a otro de la cuenta en cada ciclo de reloj. Cuando la secuencia llega a su final el circuito vuelve a su comienzo al siguiente ciclo de reloj.

- Una de las características principales del contador es el número de elementos de la cuenta. Así, se dice que un contador es módulo N o divide por N, haciendo referencia a que la frecuencia de la señal de reloj puede ser dividida por N utilizando el contador mediante una señal que se fija a 1 al llegar al final de la cuenta.

- La estructura básica de un contador es un circuito secuencial síncrono (aunque también hay estructuras asíncronas) de tipo máquina de Moore formado por N flip-flops (la secuencia puede tener un máximo de $2^N$ datos) disparados por flanco, donde las salidas de los flip-flops son las salidas del circuito (no hay decodificador de salida). Además del reloj, el contador puede tener otras entradas de control, que permitan poner la salida a 0, precargar datos, mantener la salida estable, realizar varias secuencias distintas (cuenta ascendente ó descendente, binario ó BCD, etc).

# Contadores

- El análisis de un circuito contador se realiza siguiendo el proceso de análisis de un circuito secuencial síncrono. El siguiente circuito realiza una cuenta binaria ascendente de 4 bits (de 0 a 15 y vuelta a 0).




J1 = K1 = 1

J2 = K2 = Z1

J3 = K3 = Z2 Z1

J4 = K4 = Z3 Z2 Z1

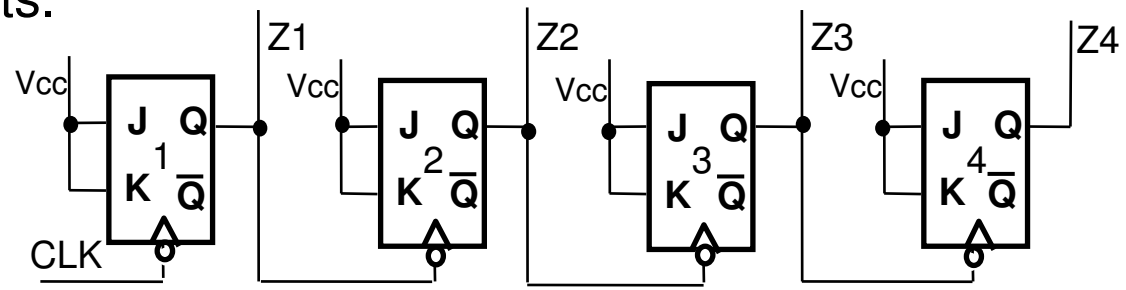
| Z4 | Z3 | Z2 | Z1 | J4 | K4 | J3 | K3 | J2 | K2 | J1 | K1 | Z4+ | Z3+ | Z2+ | Z1+ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

Al ser Ji = Ki el flip-flop J-K opera como un flip-flop T. Así todo:

| Ji | Ki | Zi+ |
|---|---|---|
| 0 | 0 | Zi |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | $\overline{Zi}$ |

# Contadores

- La estructura asíncrona de circuitos contadores utiliza las salidas de algunos flip-flops como entradas de otros flip-flops, de forma que un flip-flop mantiene un dato hasta que la salida del haga el flanco adecuado. El siguiente circuito es un contador binario ascendente de 4 bits.




J1 = K1 = 1

J2 = K2 = 1

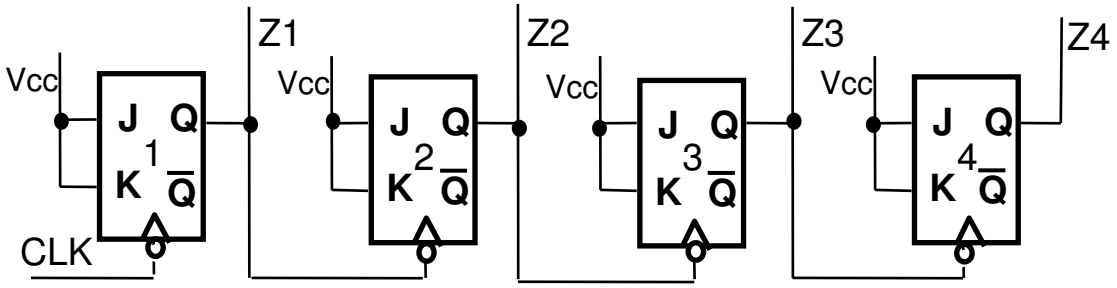
J3 = K3 = 1

J4 = K4 = 1

El flip-flop 2 sólo se activará (y cambiará de valor ya que J = K = 1) cuando haya un flanco negativo en Z1, igual que el flip-flop 3 con Z2 y el 4 con Z3. Si el reloj no se activa el valor de las entradas del flip-flop es indiferente.

El problema de estas estructuras es que el tiempo de propagación del contador es mayor que en los circuitos síncronos, ya que en esta estructura en el peor caso el reloj se propaga desde el primer flip-flop hasta el último (Tp(contador) = N Tp(ff) ), mientras que en los circuitos síncronos todos los flips-flops cambian en paralelo.

# Contadores




J1 = K1 = 1

J2 = K2 = 1

J3 = K3 = 1

J4 = K4 = 1

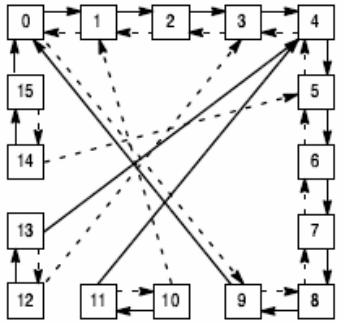
| Z4 | Z3 | Z2 | Z1 | J4 | K4 | CP4 | J3 | K3 | CP3 | J2 | K2 | CP2 | J1 | K1 | Z4+ | Z3+ | Z2+ | Z1+ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | X | X | No | X | X | No | X | X | No | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | X | X | No | X | X | No | 1 | 1 | Si | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | X | X | No | X | X | No | X | X | No | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | X | X | No | 1 | 1 | Si | 1 | 1 | Si | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | X | X | No | X | X | No | X | X | No | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | X | X | No | X | X | No | 1 | 1 | Si | 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | X | X | No | X | X | No | X | X | No | 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | Si | 1 | 1 | Si | 1 | 1 | Si | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | X | X | No | X | X | No | X | X | No | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | X | X | No | X | X | No | 1 | 1 | Si | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | X | X | No | X | X | No | X | X | No | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | X | X | No | 1 | 1 | Si | 1 | 1 | Si | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | X | X | No | X | X | No | X | X | No | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | X | X | No | X | X | No | 1 | 1 | Si | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | X | X | No | X | X | No | X | X | No | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | Si | 1 | 1 | Si | 1 | 1 | Si | 1 | 1 | 0 | 0 | 0 | 0 |

CP2, CP3 y CP4 son los relojes de los flip-flops 2, 3, 4, el valor *No* indica que no hay flanco activo de reloj, el valor *S*i indica que sí lo hay.

# Contadores

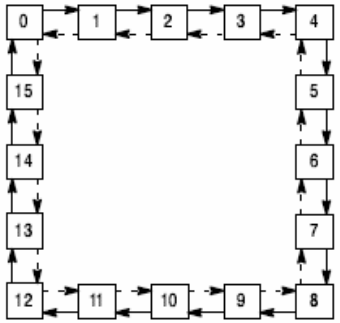
- Existen muchos contadores en los catálogos de los dispositivos digitales comerciales. Los contadores 74LS168/74LS169 son un buen ejemplo de contadores síncronos de varias secuencias con señales de control. El contador 74'168 realiza cuentas en código NBCD (de 0 a 9 y vuelta a 0) ascendente/descendente, mientras que el 74'169 realiza la cuenta en binario de 4 bits.

**SN54/74LS168**
**UP/DOWN DECADE COUNTER**




Count Up
Count Down

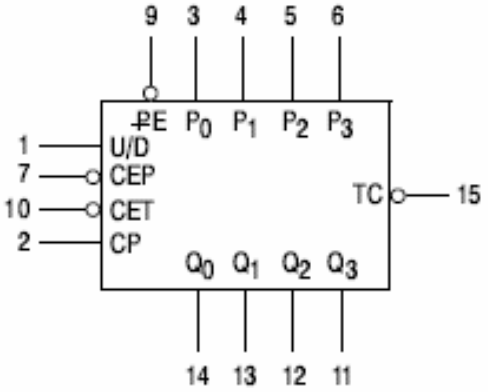
**SN54/74LS169**




**SN54/74LS168**
UP: $TC = \overline{Q_0} \cdot \overline{Q_3} \cdot (U/\overline{D})$
DOWN: $TC = \overline{Q_0} \cdot \overline{Q_1} \cdot \overline{Q_2} \cdot \overline{Q_3} \cdot \overline{(U/D)}$

**SN54/74LS169**
UP: $TC = \overline{Q_0} \cdot \overline{Q_1} \cdot \overline{Q_2} \cdot \overline{Q_3} \cdot (U/\overline{D})$
DOWN: $TC = \overline{Q_0} \cdot \overline{Q_1} \cdot \overline{Q_2} \cdot \overline{Q_3} \cdot \overline{(U/D)}$

**LOGIC SYMBOL**



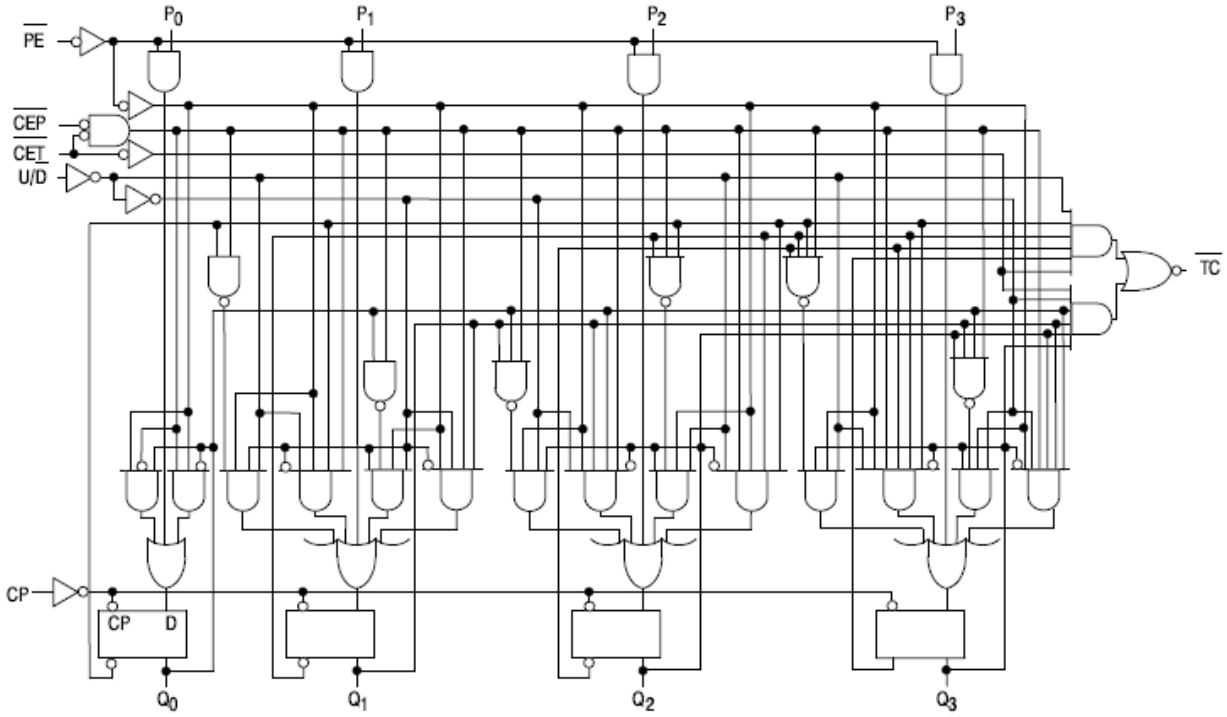

| | |
|---|---|
| $\overline{CEP}$ | Count Enable Parallel (Active LOW) Input |
| CET | Count Enable Trickle (Active LOW) Input |
| $\overline{CP}$ | Clock Pulse (Active positive going edge) Input |
| $\overline{PE}$ | Parallel Enable (Active LOW) Input |
| U/D | Up-Down Count Control Input |
| $P_0$–$P_3$ | Parallel Data Inputs |
| $\overline{Q_0}$–$\overline{Q_3}$ | Flip-Flop Outputs |
| $\overline{TC}$ | Terminal Count (Active LOW) Output |

**MODE SELECT TABLE**

| $\overline{PE}$ | $\overline{CEP}$ | $\overline{CET}$ | U/$\overline{D}$ | Action on Rising Clock Edge |
|---|---|---|---|---|
| L | X | X | X | Load (Pn Qn) |
| H | L | L | H | Count Up (increment) |
| H | L | L | L | Count Down (decrement) |
| H | H | X | X | No Change (Hold) |
| H | X | H | X | No Change (Hold) |

H = HIGH Voltage Level
L = LOW Voltage Level
X = Immaterial

# Contadores




**AC CHARACTERISTICS** ($T_A$ = 25°C, $V_{CC}$ = 5.0 V)

| Symbol | Parameter | Limits | | | Unit |
|---|---|---|---|---|---|
| | | Min | Typ | Max | |
| $f_{MAX}$ | Maximum Clock Frequency | 25 | 32 | | MHz |
| $t_{PLH}$<br>$t_{PHL}$ | Propagation Delay,<br>Clock to TC | | 23<br>23 | 35<br>35 | ns |
| $t_{PLH}$<br>$t_{PHL}$ | Propagation Delay,<br>Clock to any Q | | 13<br>15 | 20<br>23 | ns |
| $t_{PLH}$<br>$t_{PHL}$ | Propagation Delay,<br>CET to TC | | 15<br>15 | 20<br>20 | ns |
| $t_{PLH}$<br>$t_{PHL}$ | Propagation Delay,<br>U/D to TC | | 17<br>19 | 25<br>29 | ns |

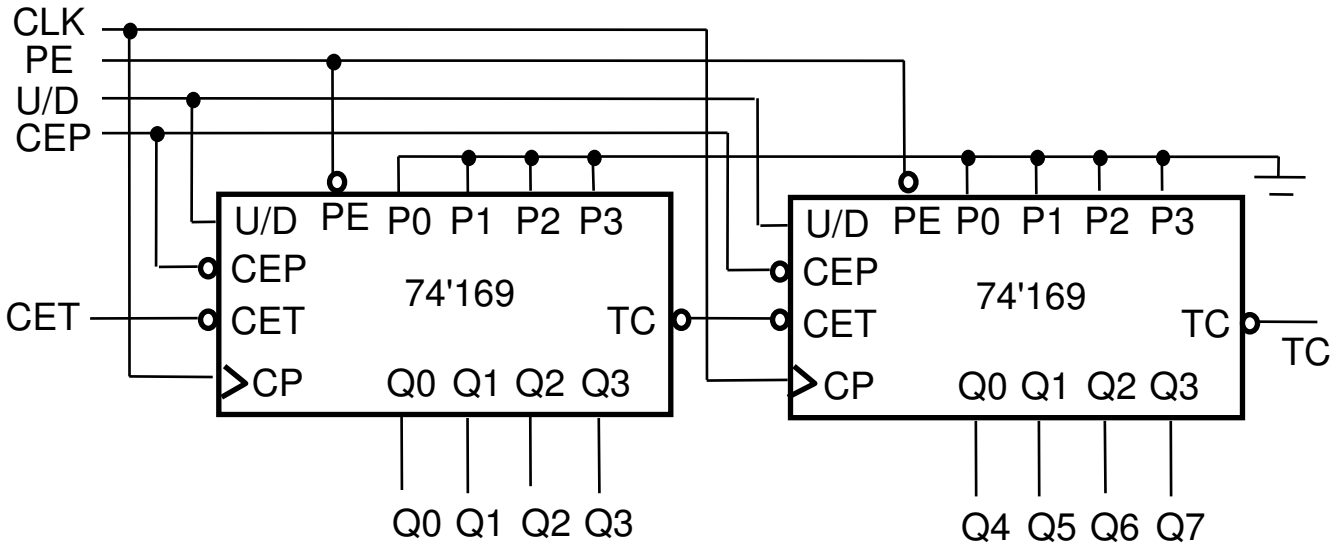
**AC SETUP REQUIREMENTS** ($T_A$ = 25°C)

| Symbol | Parameter | Limits | | | Unit |
|---|---|---|---|---|---|
| | | Min | Typ | Max | |
| $t_W$ | Clock Pulse Width | 25 | | | ns |
| $t_s$ | Setup Time,<br>Data or Enable | 20 | | | ns |
| $t_s$ | Setup Time<br>PE | 25 | | | ns |
| $t_s$ | Setup Time<br>U/D | 30 | | | ns |
| $t_h$ | Hold Time<br>Any Input | 0 | | | ns |

# Contadores

- Las entradas de control son síncronas. Al activarse PE se realiza una carga en paralelo de las entradas P (Q <= P); CEP (en paralelo) y CET (en serie) activas permiten realizar la cuenta, si alguna esta desactivada la cuenta se detiene y se mantiene; U/D determina cuenta ascendente o descendente.
  La salida de control TC determina si la cuenta ha llegado a su final: 0 en cuenta descendente y el valor máximo en cuenta ascendente (15 en el 169, 9 en el 168), y además CET está activo.

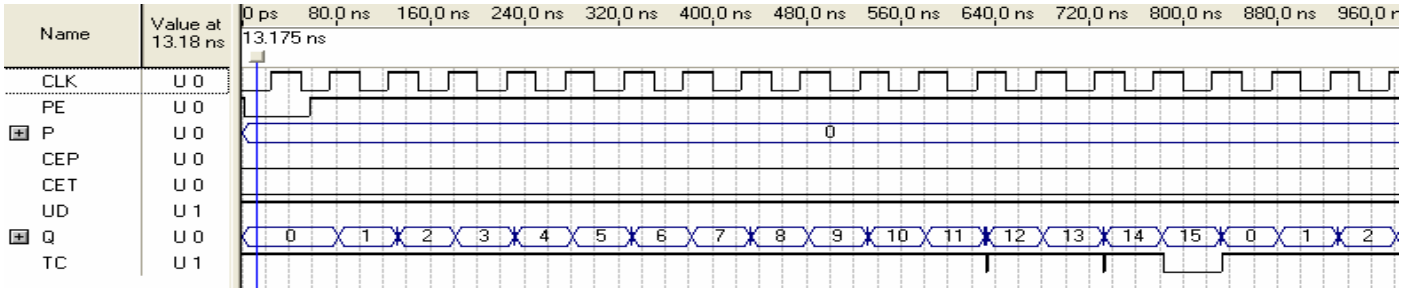
  La señales TC y CET permiten conectar varios contadores en serie para realizar cuentas de más bits: TC del menos significativo (LSB) se conecta a CET del más significativo (MSB). Así, el contador MSB mantiene su valor y sólo cambia cuando el contador LSB llega al final de cuenta. Otra opción es conectar Q3 del LSB al reloj del MSB.



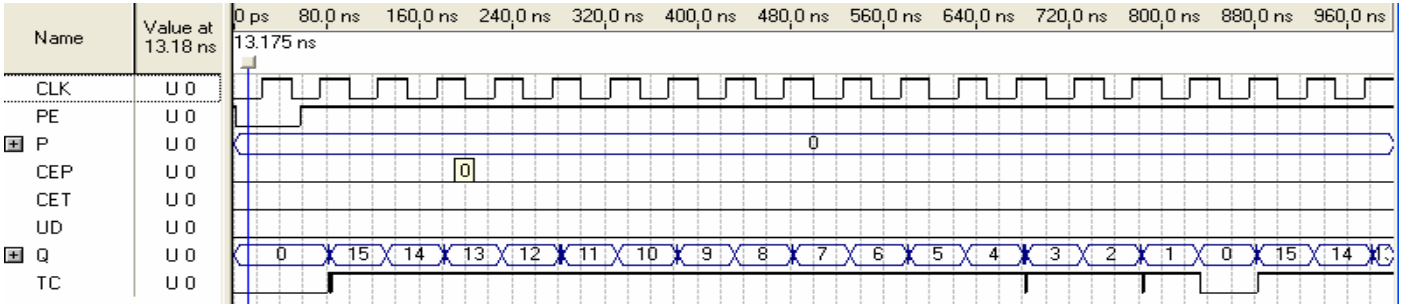
# Contadores

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

entity cont_169 is
   port (CLK, PE,CET,CEP,UD: in std_logic;
         P: in std_logic_vector(3 downto 0);
         TC: out std_logic;
         Q: out std_logic_vector(3 downto 0));
end cont_169;

architecture comp of cont_169 is

-- Salidas internas de los flip-flops
signal Z: std_logic_vector (3 downto 0);

begin
-- Salida de cuenta
process (CLK)
   begin
   if ( CLK'EVENT and CLK = '1' ) then
      if (PE = '0') then
         Z <= P;
      elsif ( CET = '0' and CEP = '0' ) then
         if ( UD = '1' ) then
            Z <= Z + 1;
         else Z <= Z -1;
         end if;
      end if;
   end if;
end process;

-- Salida TC
process (Z, UD, CET)
begin
   if ( CET = '1') then
      TC <= '1'
   elsif ( UD = '1'  and Z = 15 ) then
      TC <= '0';
   elsif ( UD = '0'  and Z = 0 ) then
      TC <= '0';
   else TC <= '1';
   end if;
end process;

Q <= Z;
end comp;
```

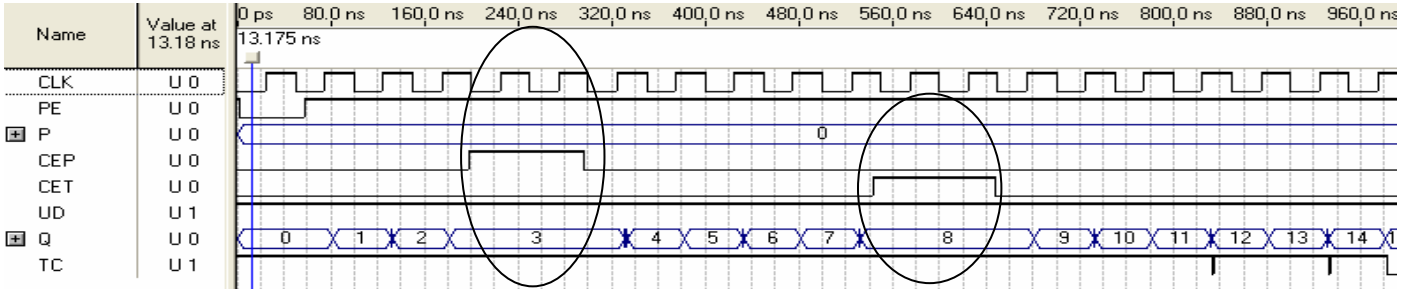
# Contadores



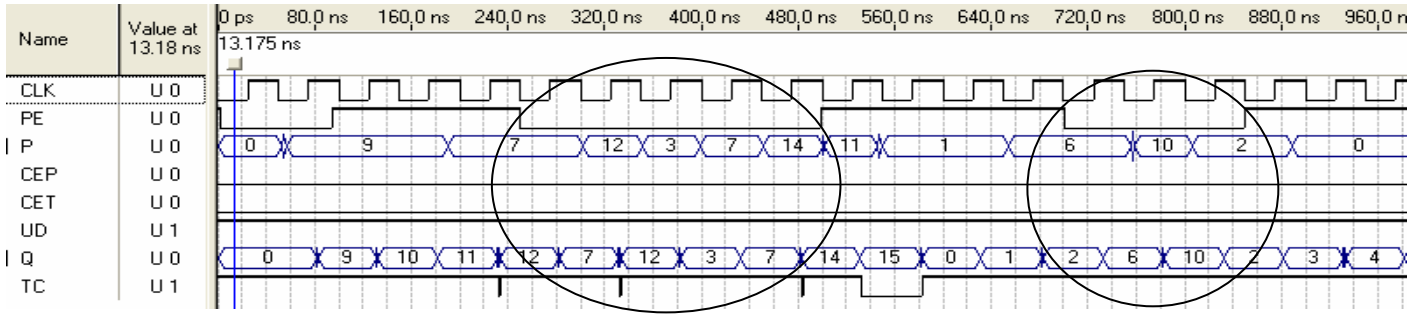

Cuenta hacia arriba




Cuenta hacia abajo




Cuentas detenidas




Precarga de datos

# Registros de desplazamiento

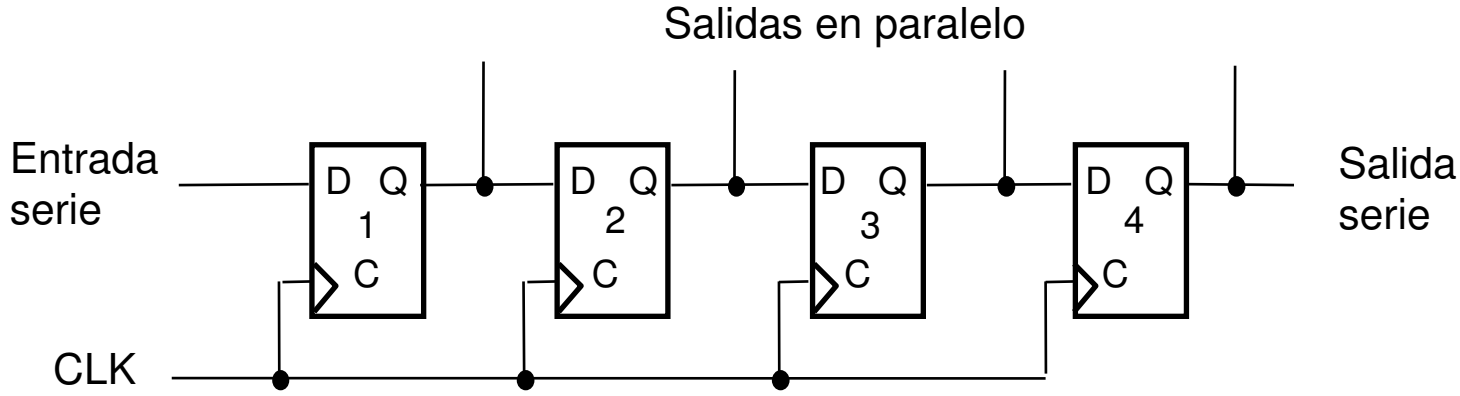
- Un registro es un elemento lógico que carga y almacena datos. En función del número N de bits almacenados se define un registro de N bits. Una forma intuitiva de construir estos elementos es con flip-flops de tipo D disparado por flanco. Los registros son un elemento básico de un sistema digital.

- Los registros de desplazamiento son registros en los que los elementos internos están conectados de forma que se puedan desplazar datos de uno a otro. Las conexiones se realizan entre la salida de un flip-flop con la entrada del siguiente flip-flop, entre elementos consecutivos del registro. Se pueden construir registros que permitan desplazamientos a la derecha, a la izquierda o en ambos sentidos (seleccionando un sentido) dentro del registro.

- Con las operaciones de desplazamiento se pueden realizar operaciones aritméticas: multiplicar por $2^N$ es desplazar una posición a la izquierda (añadiendo 0 al LSB), dividir de forma entera por $2^N$ es desplazar N posiciones a la derecha.
  También se puede desarrollar contadores divide por N o por 2N en registros de N bits.

# Registros de desplazamiento

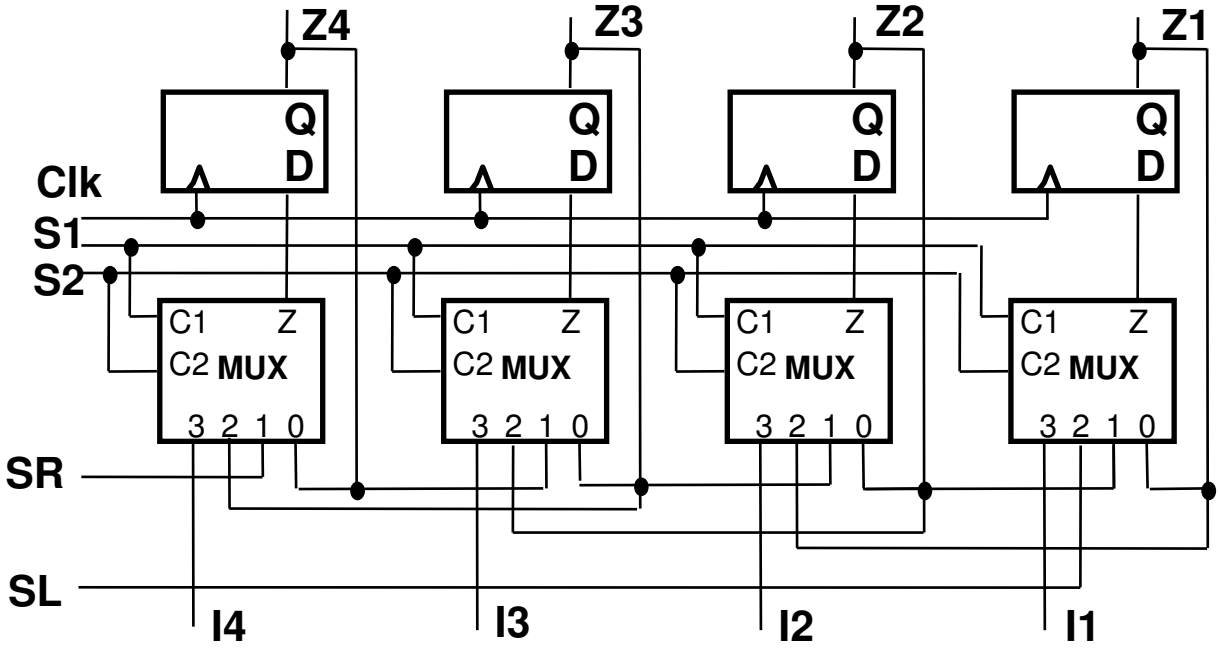
- Al establecer las operaciones de desplazamiento los datos en las entradas y salidas se pueden utilizar en paralelo o en serie. En un registro convencional de N bits se introducen los datos en N entradas y se leen en N salidas en paralelo (registro PIPO: Parallel-In; Parallel-Out). En un registro serial los datos se leen en una única entrada serie y se obtienen en una única salida serie (registro SISO: Serial-In; Serial-Out).
  Además se pueden establecer registros mixtos como el SIPO (Serial-In; Parallel-Out) o el PISO (Parallel-In; Serial-Out).



# Registros de desplazamiento

- Un modelo de un registro de desplazamiento con operaciones de carga en paralelo y desplazamientos a la derecha y a la izquierda se puede construir con flip-flips D y multiplexores. El circuito digital tiene dos señales de control S1, S2 para 4 operaciones; dos entradas serial SR y SL (desplazamientos a derecha e izquierda), una entrada paralelo I de 4 bits y una salida paralelo Z de 4 bits.

| S0 | S1 | Z4+ | Z3+ | Z2+ | Z1+ |
|---|---|---|---|---|---|
| 0 | 0 | Z4 | Z3 | Z2 | Z1 |
| 0 | 1 | SR | Z4 | Z3 | Z2 |
| 1 | 0 | Z3 | Z2 | Z1 | SL |
| 1 | 1 | I4 | I3 | I2 | I1 |

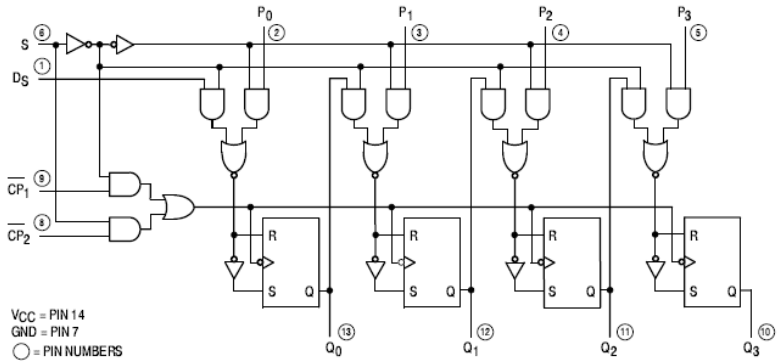


# Registros de desplazamiento

- Existen varios registros de desplazamientos en los catálogos de los dispositivos digitales comerciales. Entre otros los registros 74’95 puede operar como SIPO ó PIPO, el 74’164 como SIPO y el 74’166 como SISO ó como PISO. El registro 74’194 es un registro universal.

## SN54/74LS95B

LOGIC DIAGRAM



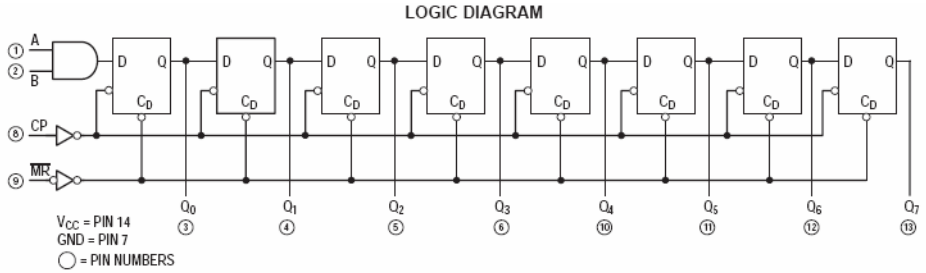

MODE SELECT — TRUTH TABLE

| OPERATING MODE | INPUTS | | | | | OUTPUTS | | | |
|---|---|---|---|---|---|---|---|---|---|
| | S | $CP_1$ | $CP_2$ | $D_S$ | $P_n$ | $Q_0$ | $Q_1$ | $Q_2$ | $Q_3$ |
| Shift | L | ↓ | X | l | X | L | $q_0$ | $q_1$ | $q_2$ |
| | L | ↓ | X | h | X | H | $q_0$ | $q_1$ | $q_2$ |
| Parallel Load | H | X | ↓ | X | $P_n$ | $P_0$ | $P_1$ | $P_2$ | $P_3$ |
| Mode Change | ↓ | L | L | X | X | No Change | | | |
| | ↑ | L | L | X | X | No Change | | | |
| | ↓ | H | L | X | X | No Change | | | |
| | ↑ | H | L | X | X | Undetermined | | | |
| | ↓ | L | H | X | X | Undetermined | | | |
| | ↑ | L | H | X | X | No Change | | | |
| | ↓ | H | H | X | X | Undetermined | | | |
| | ↑ | H | H | X | X | No Change | | | |

L = LOW Voltage Level
H = HIGH Voltage Level
X = Don’t Care
l = LOW Voltage Level one set-up time prior to the HIGH to LOW clock transition.
h = HIGH Voltage Level one set-up time prior to the HIGH to LOW clock transition.
$P_n$ = Lower case letters indicate the state of the referenced input (or output) one set-up time prior to the HIGH to LOW clock transition.

## SN74LS164

LOGIC DIAGRAM




MODE SELECT — TRUTH TABLE

| OPERATING MODE | INPUTS | | | OUTPUTS | |
|---|---|---|---|---|---|
| | $\overline{MR}$ | A | B | $Q_0$ | $Q_1$–$Q_7$ |
| Reset (Clear) | L | X | X | L | L – L |
| Shift | H | l | l | L | $q_0$ – $q_6$ |
| | H | l | h | L | $q_0$ – $q_6$ |
| | H | h | l | L | $q_0$ – $q_6$ |
| | H | h | h | H | $q_0$ – $q_6$ |

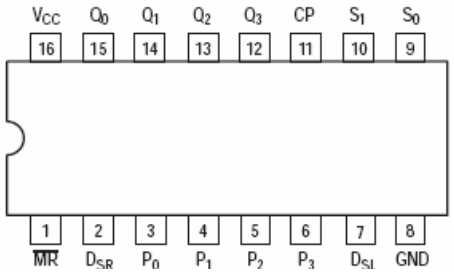
L (l) = LOW Voltage Levels
H (h) = HIGH Voltage Levels
X = Don’t Care
$q_n$ = Lower case letters indicate the state of the referenced input or output one set-up time prior to the LOW to HIGH clock transition.

# Registros de desplazamiento

## SN74LS194A

## 4-Bit Bidirectional Universal Shift Register

CONNECTION DIAGRAM DIP (TOP VIEW)



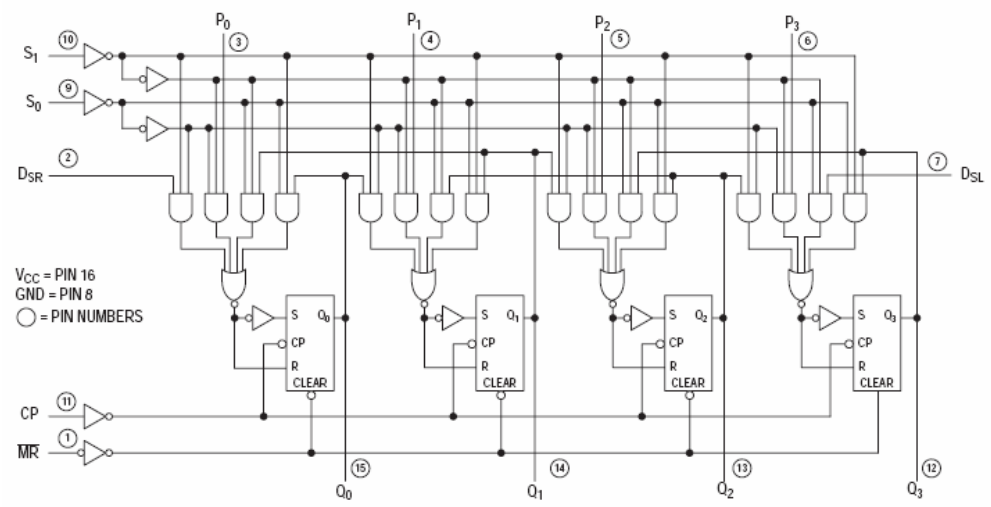

**PIN NAMES**

| | |
|---|---|
| $S_0$, $S_1$ | Mode Control Inputs |
| $P_0 - P_3$ | Parallel Data Inputs |
| $D_{SR}$ | Serial (Shift Right) Data Input |
| $D_{SL}$ | Serial (Shift Left) Data Input |
| CP | Clock (Active HIGH Going Edge) Input |
| $\overline{MR}$ | Master Reset (Active LOW) Input |
| $Q_0 - Q_3$ | Parallel Outputs |

MODE SELECT — TRUTH TABLE

| OPERATING MODE | INPUTS | | | | | | OUTPUTS | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $\overline{MR}$ | $S_1$ | $S_0$ | $D_{SR}$ | $D_{SL}$ | $P_n$ | $Q_0$ | $Q_1$ | $Q_2$ | $Q_3$ |
| Reset | L | X | X | X | X | X | L | L | L | L |
| Hold | H | l | l | X | X | X | $q_0$ | $q_1$ | $q_2$ | $q_3$ |
| Shift Left | H<br>H | h<br>h | l<br>l | X<br>X | l<br>h | X<br>X | $q_1$<br>$q_1$ | $q_2$<br>$q_2$ | $q_3$<br>$q_3$ | L<br>H |
| Shift Right | H<br>H | l<br>l | h<br>h | l<br>h | X<br>X | X<br>X | L<br>H | $q_0$<br>$q_0$ | $q_1$<br>$q_1$ | $q_2$<br>$q_2$ |
| Parallel Load | H | h | h | X | X | $P_n$ | $P_0$ | $P_1$ | $P_2$ | $P_3$ |

L = LOW Voltage Level
H = HIGH Voltage Level
X = Don't Care
l = LOW voltage level one set-up time prior to the LOW to HIGH clock transition
h = HIGH voltage level one set-up time prior to the LOW to HIGH clock transition
$p_n$ ($q_n$) = Lower case letters indicate the state of the referenced input (or output) one set-up time prior to the LOW to HIGH clock transition.




AC CHARACTERISTICS ($T_A$ = 25°C)

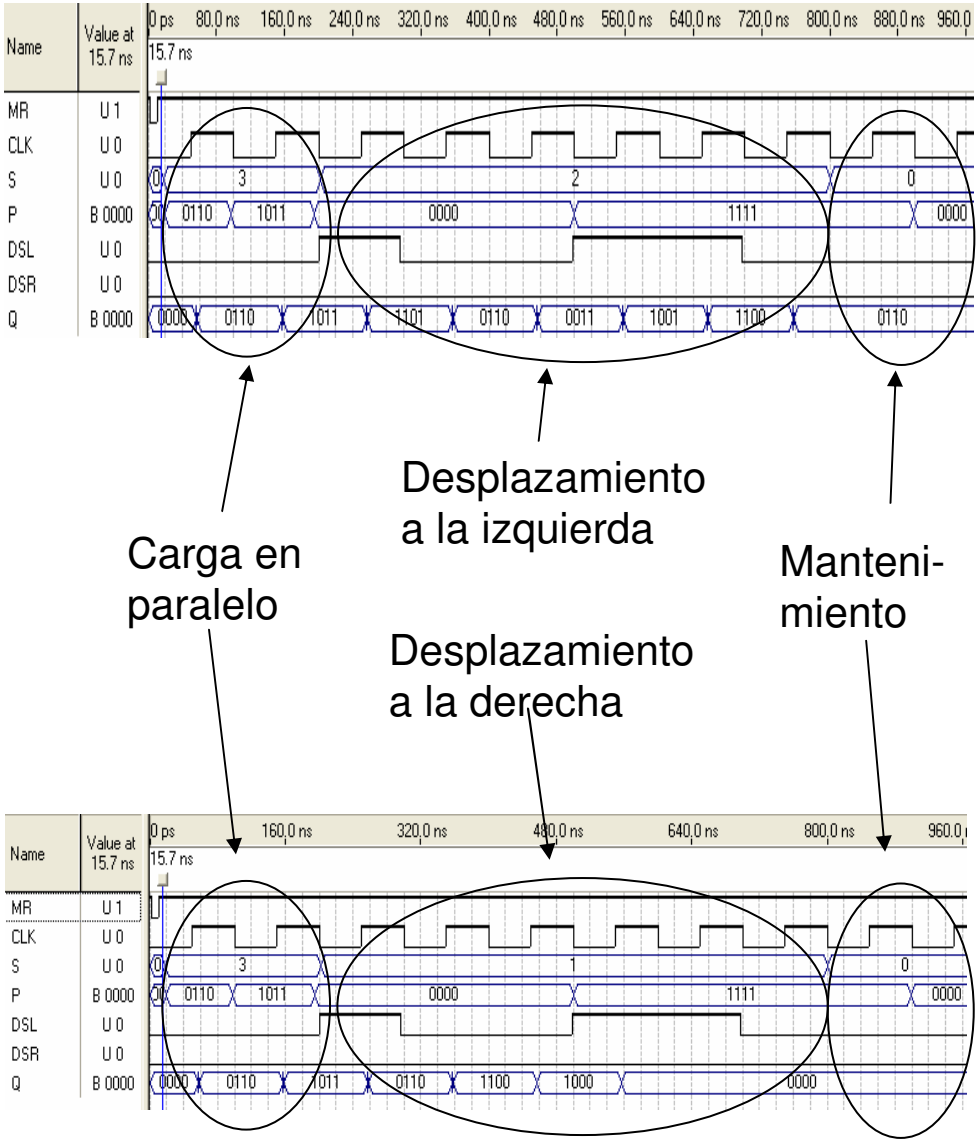
| Symbol | Parameter | Limits Min | Limits Typ | Limits Max | Unit |
|---|---|---|---|---|---|
| $f_{MAX}$ | Maximum Clock Frequency | 25 | 36 | | MHz |
| $t_{PLH}$<br>$t_{PHL}$ | Propagation Delay, Clock to Output | | 14<br>17 | 22<br>26 | ns |
| $t_{PHL}$ | Propagation Delay, $\overline{MR}$ to Output | | 19 | 30 | ns |

AC SETUP REQUIREMENTS ($T_A$ = 25°C)

| Symbol | Parameter | Limits Min | Limits Typ | Limits Max | Unit |
|---|---|---|---|---|---|
| $t_W$ | Clock or $\overline{MR}$ Pulse Width | 20 | | | ns |
| $t_s$ | Mode Control Setup Time | 30 | | | ns |
| $t_s$ | Data Setup Time | 20 | | | ns |
| $t_h$ | Hold time, Any Input | 0 | | | ns |
| $t_{rec}$ | Recovery Time | 25 | | | ns |

# Registros de desplazamiento

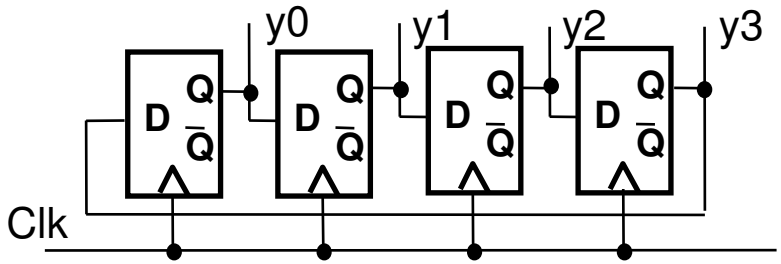
```
library ieee;
use ieee.std_logic_1164.all;

entity SRL_194 is
   port (CLK, MR, DSR, DSL: in std_logic;
         S: in std_logic_vector(0 to 1);
         P: in std_logic_vector(0 to 3);
         Q: out std_logic_vector(0 to 3));
end SRL_194;

architecture comp of SRL_194 is

-- Salidas internas de los flip-flops
signal Z: std_logic_vector (0 to 3);

begin
process (CLK, MR)
   begin
   if ( MR = '0' ) then   -- Reset asincrono
      Z <= "0000";
   elsif ( CLK'EVENT and CLK = '1' ) then
       case S is
          when "01" => Z <= DSR & Z(0 to 2);
          when "10" => Z <= Z(1 to 3) & DSL;
          when "11" => Z <= P;
          when others => NULL;
       end case;
   end if;
end process;

Q <= Z;

end comp;
```



# Registros de desplazamiento

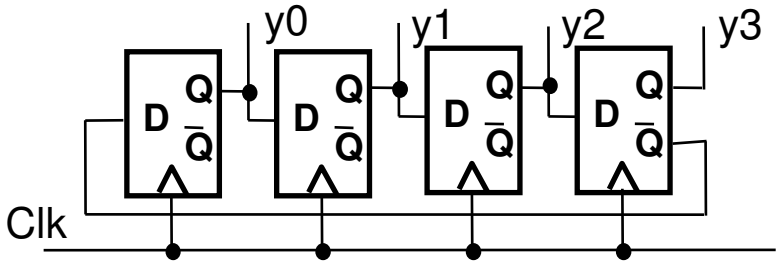
- Los registros de desplazamiento de N bits en modo SISO se pueden configurar como contadores mediante realimentaciones de la salida serial a a la entrada serial.

  El contador en anillo realiza una secuencia de N datos mediante el desplazamiento y rotación de un 1 (hay que precargar de alguna manera la situación inicial en el registro) por el registro de desplazamiento.




| y0 | y1 | y2 | y3 |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 |

  El contador Johnson realiza una secuencia de 2N datos mediante la realimentación desde la salida complementada (hay que hacer un reset previo del registro).




| y0 | y1 | y2 | y3 |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 |