Tema 4. Elementos Lógicos Secuenciales

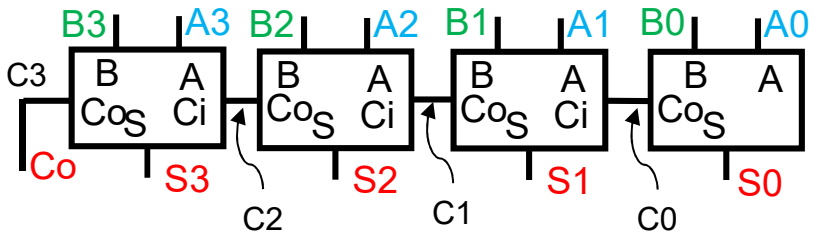
- Introducción a los circuitos secuenciales síncronos y asíncronos.
- Latch S-R.
- Flip-flops de tipo “clocked-latch”. Flip-flops D, T, J-K.
- Conversiones entre flip-flops.
- Estructuras de reloj síncrono: “master-slave”, “disparados por flanco”.

Circuitos Secuenciales

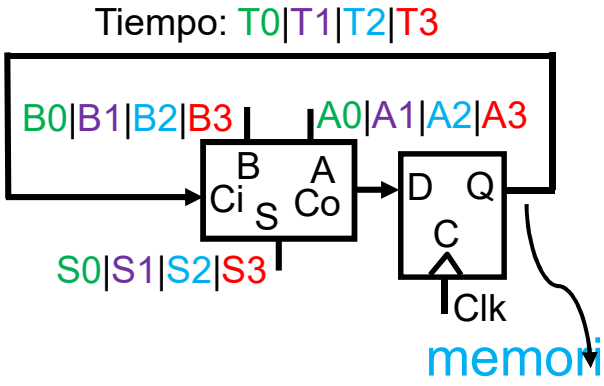
- Hasta ahora se han estudiado circuitos combinacionales, en los que las salidas son función instantánea del valor de las entradas. Con estos circuitos no es posible plantear ciertas aplicaciones, por ejemplo un contador: un circuito en el que sus salidas siguen una secuencia fija que cuando acaba vuelve a empezar. La definición lógica de este circuito ni siquiera tiene entradas ya que la salida depende de sí misma.

- Los circuitos secuenciales resuelven problemas en los que se tienen en cuenta valores en las entradas a lo largo del tiempo. Por ejemplo, un sumador se puede realizar como circuito combinacional sumando varios bits a la vez, o secuencialmente sumando solo un bit en el tiempo y almacenando en "memoria" el bit de acarreo para la siguiente operación en el tiempo.

Combinacional

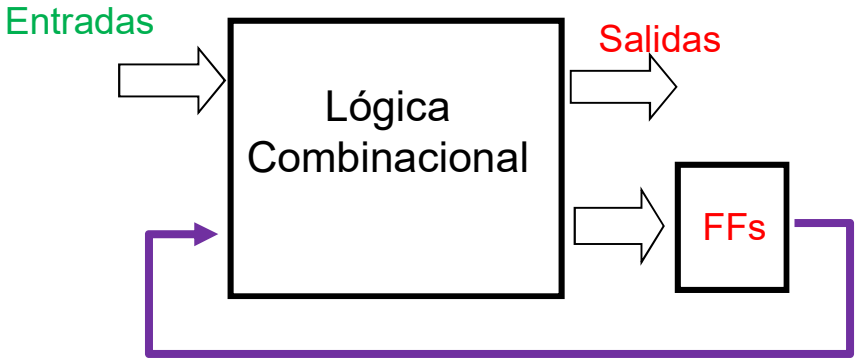


Secuencial



Circuitos Secuenciales

- Los circuitos combinacionales no sirven para resolver este tipo de problemas. Se necesitan circuitos con "memoria" capaces de recordar o almacenar datos lógicos. Esta memoria se consigue mediante realimentaciones en los circuitos digitales y, normalmente, mediante el uso de los flip-flops que son unos elementos lógicos específicos de almacenamiento.



- Los circuitos secuenciales se modelan mediante máquinas de estados finito (FSM, Finite State Machine). Los estados representan las posibles situaciones internas de la máquina, que evoluciona de una a otra según el estado actual y las entradas.

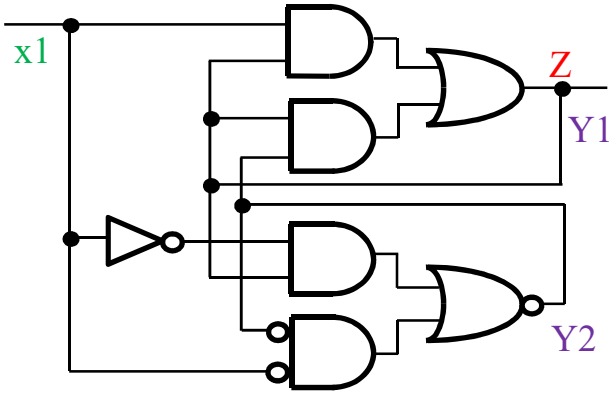
  La FSM describe un problema lógico mediante:
  - Las entradas I = {I1, …, Ip}
  - Las salidas O = {O1, …, Oq}
  - Los estados S = {S1, …, Sr}

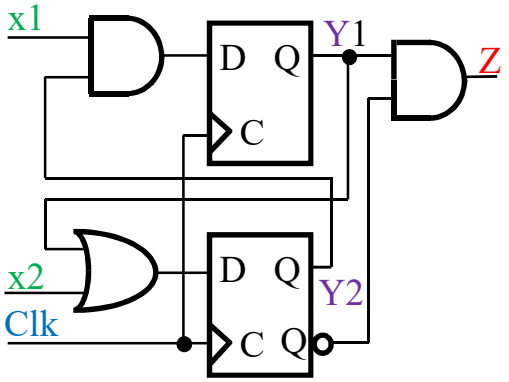
Circuitos Secuenciales

- Los "estados" del circuito, se codifican en binario en variables de estado Y realimentadas, y que evoluciona según el estado actual y las entradas. Estos circuitos se llaman secuenciales, ya que su funcionamiento depende de la secuencia de entrada. Existen dos tipos de circuitos secuenciales:

Asíncronos: evolucionan a cada cambio de entrada, las realimentaciones son directas y el modelo matemático es complicado y restrictivo (solo cambia una entrada cada vez).

Síncronos: el circuito solo evoluciona al cambiar una señal específica de reloj. Las realimentaciones se realizan a través de flip-flops controlados por el reloj. El modelo matemático es sencillo y con pocas restricciones temporales.





Circuitos secuenciales asíncronos

- Los circuitos secuenciales asíncronos evolucionan según los cambios de cualquiera de las entradas.
  Esto conlleva problemas cuando se supone la evolución de dos o más entradas simultáneamente: por razones físicas es imposible suponer que cambian en el mismo tiempo exactamente, y aunque lo hiciesen los retrasos internos del circuito pueden producir desviaciones en los tiempos cuando se producen los cambios.

  Así, un cambio en las entradas (AB) 00 -> 11, puede ser:
  - Exacto: 00 -> 11
  - Cambia primero B: 00->01->11
  - Cambia primero A: 00->10->11

  Como la secuencia de entradas es distinta en cada caso la evolución del circuito sería distinta (e impredecible en el modelo matemático) según aspectos físicos del circuito.

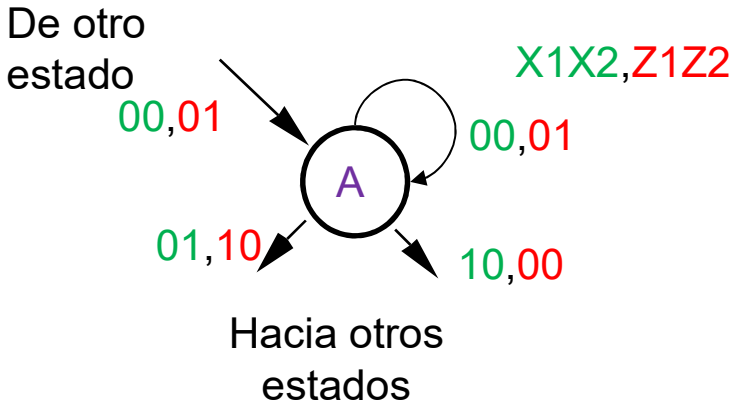
- El modelo matemático de diseño de estos circuitos supone un "modo fundamental" en el que solo cambia una entrada cada vez, y que la siguiente entrada no cambia hasta que el circuito se ha estabilizado.

Circuitos secuenciales asíncronos

- El desarrollo de estos circuitos parte de la descripción de su funcionamiento mediante el "diagrama de flujo". En él cada situación distinta del problema se representa por un estado, que es estable mientras que no cambie ninguna entrada. Cuando se está en un estado y cambia una entrada se salta en el diagrama a otro estado.

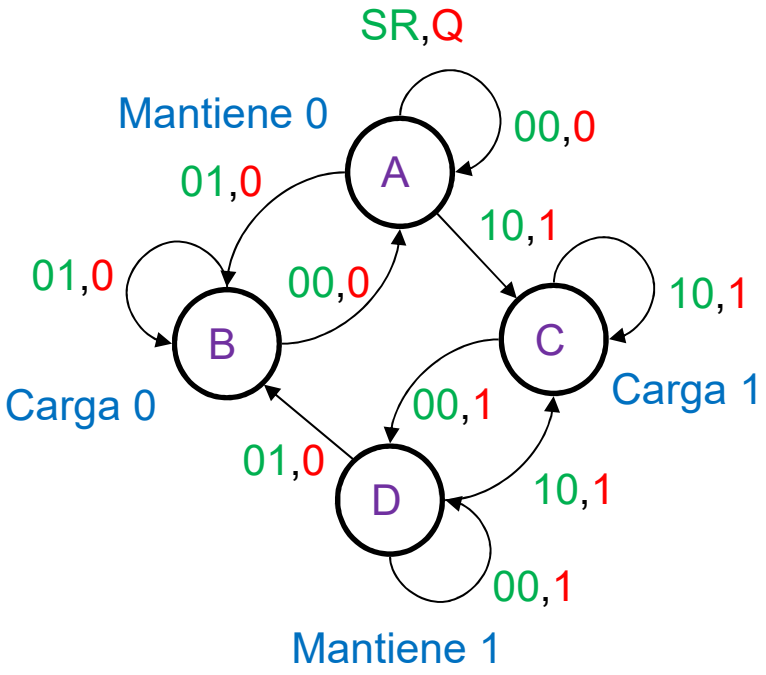
  Los estados se representan por círculos y las transiciones entre estados por flechas. En las transiciones se indica el valor que toman las entradas y las salidas en formato Entradas (0s o 1s), Salidas (0s, 1s, -s).

  Todo estado tiene una transición sobre si mismo representando la situación estable.



Circuitos secuenciales asíncronos

- Ejemplo: Circuito S-R. Este es un elemento secuencial básico, cuyo diseño se puede hacer usando el modo fundamental. Tiene dos entradas S (Set) y R (Reset).
  Cuando se activa R se carga 0 en la salida Q (SR = 01).
  Cuando se activa S se carga 1 en la salida Q (SR = 10).
  Con S y R no activas se mantiene el valor anterior de Q (SR = 00).
  No se pueden activar a la vez S y R (SR = 11).



estado \ SR	00	01	11	10
A	(A), 0	B, 0	-, -	C, 1
B	A, 0	(B), 0	-, -	-, -
C	D, 1	-, -	-, -	(C), 1
D	(D), 1	B, 0	-, -	C, 1

El diagrama de flujo se puede pasar a una tabla de estados (o de flujo) donde se muestran las transiciones entre estados.
Los estados con círculo son los estados estables.

Circuitos secuenciales asíncronos

- A partir de la tabla de estados se hace la síntesis lógica del circuito. Hay que realizar varios pasos que, debido a su complejidad solo se enunciarán:

  - Reducir el número de estados mediante "compatibilidad". Dos o más estados son compatibles si desde ellos cualquier secuencia de entradas de cualquier longitud no genera incompatibilidad, es decir no genera valores (que no sean don't cares) distintos en las salidas desde un estado y desde el otro.
  Una vez estudiados los grupos de estados compatibles se selecciona el menor número de grupos que genere una "cobertura cerrada", es decir que cubran todos los estados al menos una vez, y que permitan realizar todas las transiciones de la tabla original.

SR

estado	00	01	11	10
A	A, 0	B, 0	-, -	C, 1
B	A, 0	B, 0	-, -	-, -
C	D, 1	-, -	-, -	C, 1
D	D, 1	B, 0	-, -	C, 1

SR

estado	00	01	11	10
AB	A, 0	B, 0	-, -	C, 1
CD	D, 1	B, 0	-, -	C, 1

Estados compatibles:
(AB) = {A, B} y (CD) = {C, D}.
Producen cobertura cerrada

Circuitos secuenciales asíncronos

- Codificar o asignar los estados en variables de estado *yi* (con N variables de estado se pueden codificar hasta 2^N estados).

La codificación debe ser "libre de carreras críticas": en una transición entre dos estados estables solo puede variar una variable de estado. Cambiar varias variables de estado produciría la misma situación que el cambio de varias entradas a la vez.

Hacer asignaciones "libres de carreras críticas" no es trivial. Se pueden hacer asignaciones iniciales con carreras que se pueden arreglar modificando los estados transitorios para generar "ciclos" (se pasa por varios estados transitorios cambiando una variable cada vez) o "carreras no críticas" (siempre se llega al estado estable final). También es posible utilizar más variables de estado y asignaciones estándar libres de carreras donde los estados se representan por más de una asignación.

SR

estado	00	01	11	10
AB	A, 0	B, 0	-, -	C, 1
CD	D, 1	B, 0	-, -	C, 1

AB => y = 0
CD => y = 1

y

SR

	00	01	11	10
0	0, 0	0, 0	-, -	1, 1
1	1, 1	0, 0	-, -	1, 1

Asignación libre de carreras críticas
(solo hay una variable de estado)

Circuitos secuenciales asíncronos

- Encontrar las funciones lógicas que definen el circuito.

Obtener *Y = F(Inp,y)* y *Z = F(Inp, y)* de la tabla. *Yi* es el valor que se fijará en las variables de estado al cambiar una entrada, en función de los valores de las entradas y de los valores actuales de todas las variables de estado *y*. Físicamente *Yi* e *yi* son el mismo nudo.

Las funciones lógicas de las variables de estado deben estar libres de peligros, ya que si se introduce un peligro y está continuamente realimentándose puede causar un mal funcionamiento. Solo es necesario evitar los peligros entre casillas del mapa de Karnaugh que tienen transiciones entre ellas.

y \ SR	00	01	11	10
0	0	0	-	1
1	1	0	-	1

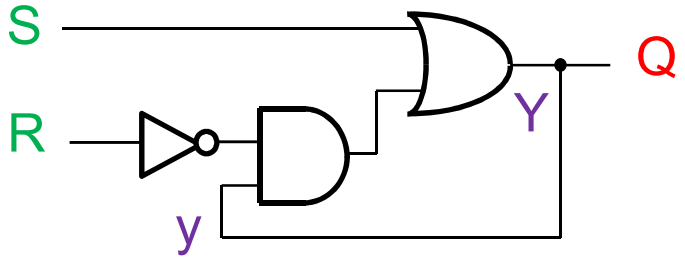
$Y = S + \overline{R}\, y$

y \ SR	00	01	11	10
0	0	0	-	1
1	1	0	-	1

$Q = S + \overline{R}\, y = Y$

Función lógica de Y libre de peligros

- Implementar el circuito con puertas lógicas.

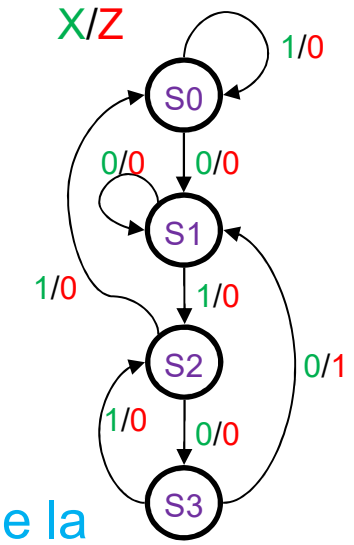


Circuitos secuenciales síncronos

- Los circuitos secuenciales síncronos evolucionan según el cambio en una única entrada, por lo que siempre se está por defecto en modo fundamental. La entrada que cambia es una señal periódica de reloj y la evolución se realiza cuando esa entrada tiene una transición o flanco de bajo a alto (flanco positivo) o de alto a bajo (flanco negativo). El resto de señales pueden cambiar durante el resto del ciclo de reloj, pero el circuito debe estar estable cuando se produzca el nuevo flanco.

  La descripción de las operación de estos circuitos se realiza mediante FSMs (máquinas de estado finito) que admiten varias tipos de máquinas (de Mealy o de Moore) y varios tipos de descripciones (diagramas de estado, diagramas ASM, etc).

- Los diagramas de estado describen la operación mediante transiciones entre estados como el diagrama de flujo. La evolución se realiza al llegar el flanco de reloj en función del valor actual de las entradas. Las salidas se generan en función del estado y las entradas actuales. El reloj no aparece en el diagrama estado.

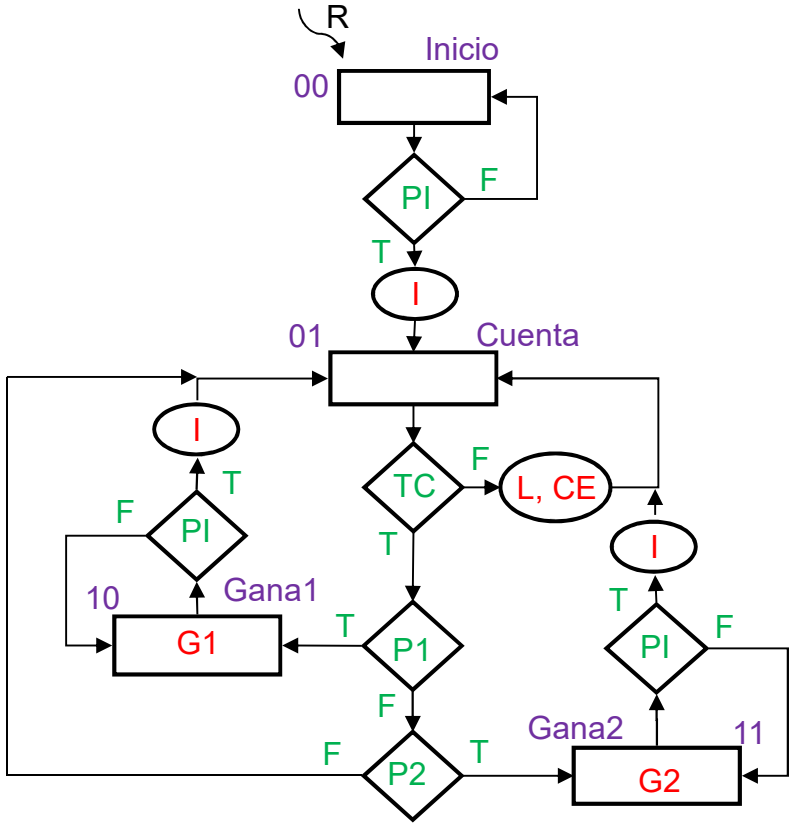


Detector de la secuencia 0100

Circuitos secuenciales síncronos

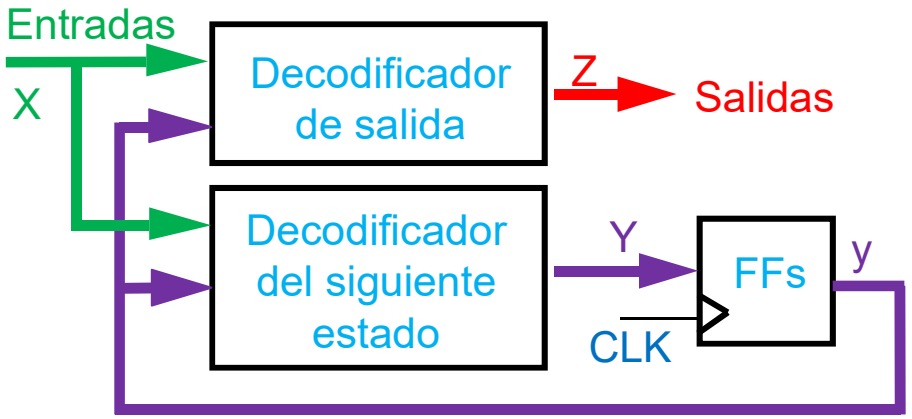
- El juego se inicia cuando el presentador pulsa un interruptor PI. Al hacerlo se arranca un reloj (I) y se enciende una luz (L).
- Finalizado el tiempo (TC) la luz se apaga y el reloj se para (CE). Los concursantes deben pulsar su interruptor (P1 o P2). El que pulse primero gana y se enciende su bombilla (G1 o G2).
- La situación anterior se mantiene hasta que el presentador pulsa otra vez el interruptor PI y se repite otra vez el proceso.

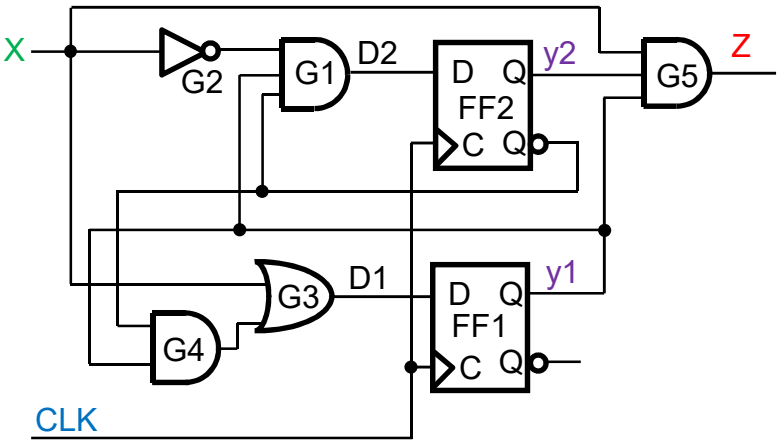
Diagrama ASM síncrono del control de un concurso



Circuitos secuenciales síncronos

- El esquema de estos circuitos es una serie de elementos de memoria (flip-flops, FFs) controlados por el reloj, que generan las variables de estado y unos circuitos combinacionales que generan las salidas (decodificadores de salida) y el nuevo valor de las variables de estado (decodificador del siguiente estado).





- El circuito operará bien si el periodo del reloj T es mayor que el tiempo que tarda en estabilizarse el circuito:

  $T > t_{pff} + t_{pdec} + t_{setup}$

  - t_{pff} (tiempo propagación FFs)
  - t_{pdec} (máximo tiempo de propagación del decodificador del siguiente estado)
  - t_{setup} (tiempo de asentamiento en las entradas de los FFs)

Circuitos secuenciales síncronos

- El diseño de circuitos secuenciales síncrono será desarrollado en Electrónica Digital II. De forma muy abreviada los pasos para realizar un diseño de este tipo serían:

  - Desarrollar la descripción del problema mediante una FSM.

  - Minimizar los estados de la FSM buscando grupos de estados equivalentes (diagramas completamente especificados) o grupos de estado compatibles (diagramas incompletamente especificados).

  - Codificar o asignar valores a los estados en las variables de estado para minimizar o, al menos reducir la lógica combinacional del circuito.

  - Generar las funciones lógicas de los decodificadores de salida y de siguiente estado del circuito utilizando un tipo de flip-flops previamente determinado.

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

- Los circuitos secuenciales síncronos requieren de flip-flops específicos, en especial disparados por flanco, que se describen en este tema.

Latch S-R

- El término "latch" (cerrojo) referencia un elemento lógico secuencial biestable en el que todas las entradas del circuito actúan por nivel: el circuito opera según los valores 0 o 1, en las entradas.
  El término "flip-flop" es un término más general que referencia a los elementos lógicos capaces de tener dos estados estables (biestables), y que incluye a elementos lógicos con entradas que actúan por flanco o transiciones 0->1 (flanco positivo), o 1->0 (flanco negativo).

- El "latch S-R" es un biestable que contiene las operaciones básicas para realizar un circuito secuencial. Al ser un circuito secuencial las operaciones indican el valor que tomará la salida al evolucionar Q+ (para indicar que es el nuevo valor) como una función de las entradas y del valor actual de la salida del elemento Q. Las operaciones son:

  Reset o puesta a 0: carga un 0 en la salida Q, Q+ = 0.
  Set o puesta a 1: carga un 1 en la salida Q, Q+ = 1.
  Mantenimiento del valor en la salida, Q+ = Q.

  Estas tres operaciones necesitan al menos dos entradas para realizarse: la entrada S activa realiza la operación de Set, la entrada R activa realiza la operación de Reset, ninguna de las dos entradas activas mantienen el dato. El caso con las dos entradas S y R activas no se considera.

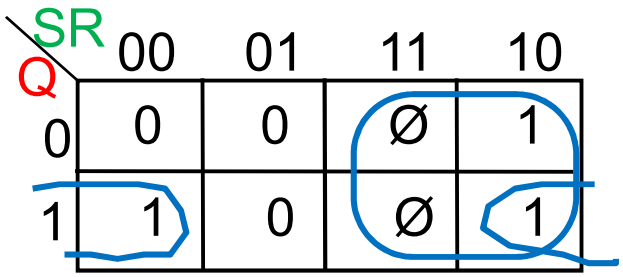
Latch S-R

S	R	Q+
0	0	Q
0	1	0
1	0	1
1	1	Ø

Tabla de operación

S	R	Q	Q+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	Ø
1	1	1	Ø

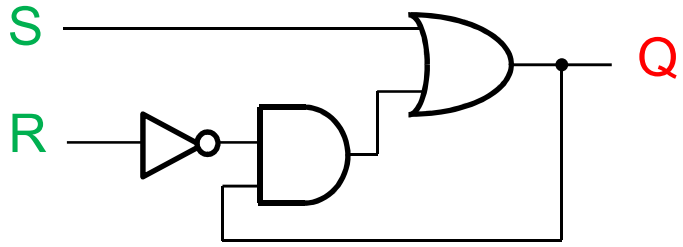
Tabla característica



$Q+ = S + \overline{R}\, Q$

Función o ecuación característica

$Q+ = F(S, R, Q)$



Este método de diseño no es totalmente correcto ya que no se han usado técnicas de diseño de circuitos asíncronos.
Sin embargo en este caso el resultado es el mismo.

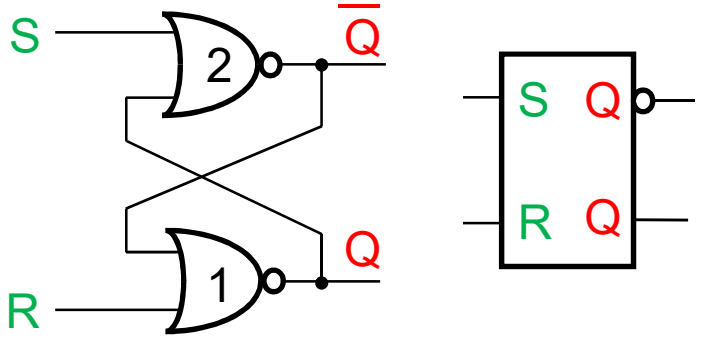
Latch S-R

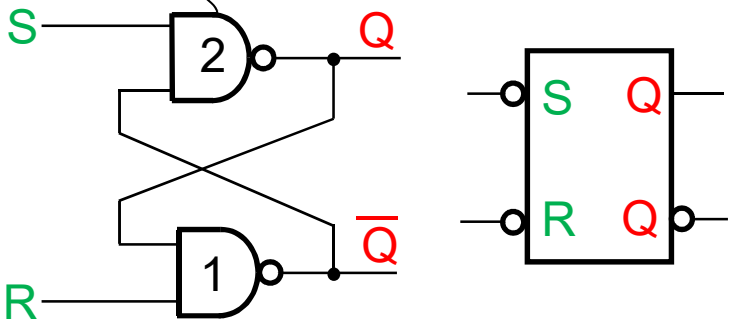
- El "latch S-R" estándar está construido con puertas mutuamente acopladas NOR o NAND. Tiene la ventaja de que genera Q y $\overline{Q}$, y la desventaja de que las entradas S y R no pueden estar activas a la vez, ya que pueden generar problemas de funcionamiento.

S	R	Q+	$\overline{Q}$+
0	0	Q	$\overline{Q}$
0	1	0	1
1	0	1	0
1	1	No valido	

** En algún libro de texto a este circuito se le asocia esta tabla con las entradas en polaridad positiva

S	R	Q+	$\overline{Q}$+
1	1	Q	$\overline{Q}$
1	0	0	1
0	1	1	0
0	0	No valido	

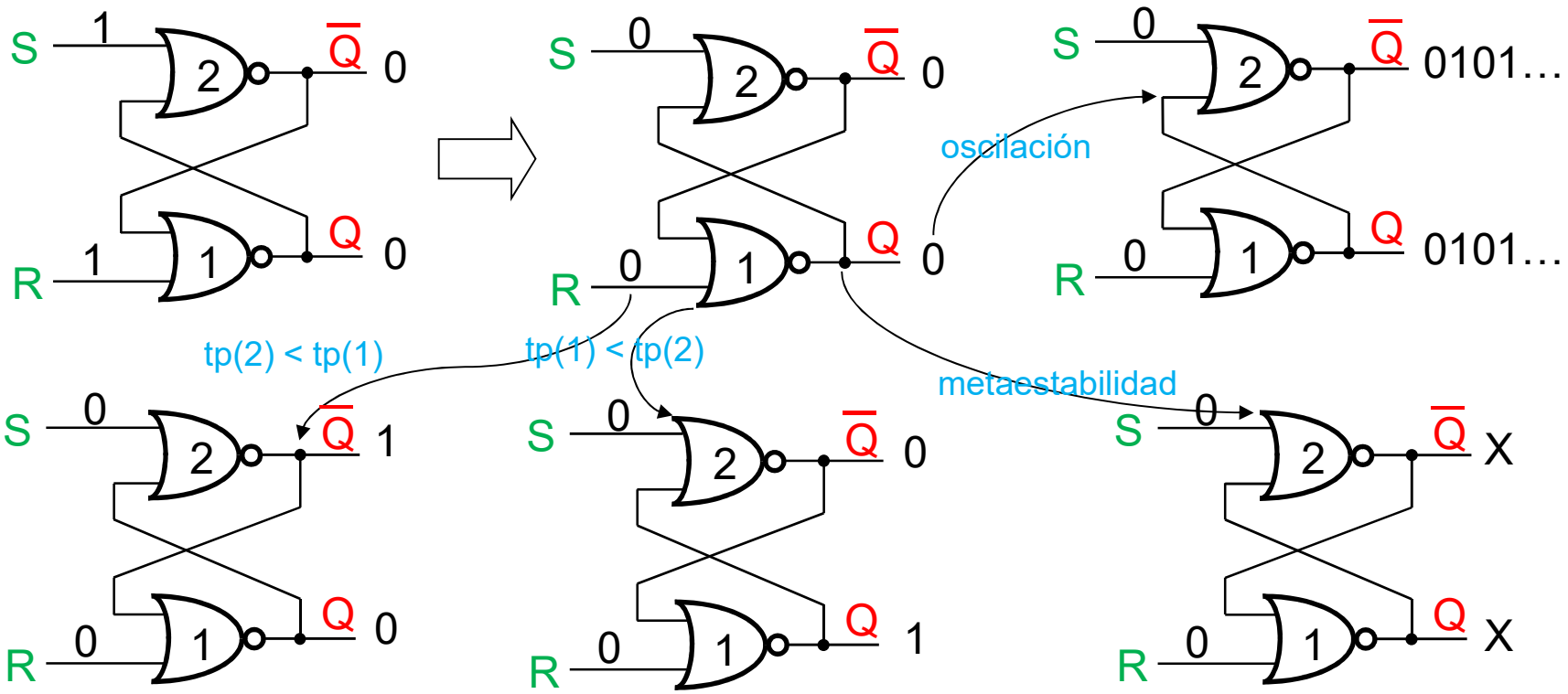




$$Q+ = \overline{R}\,(S + Q)$$

Latch S-R

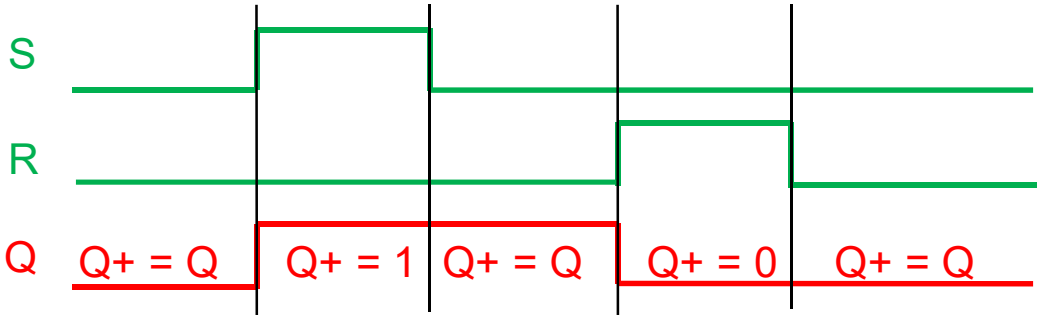
- No se permiten activar las dos entradas a la vez porque puede dar problemas si las dos entradas se desactivan simultáneamente (o en tiempo muy cercano).

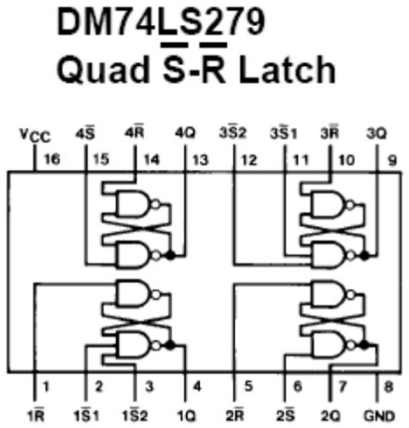


- Al no ser predecible la situación final no se puede permitir la situación inicial. El mismo efecto puede producirse si a las entradas S y R se les aplica un pulso (0->1->0) muy estrecho. Los flip-flops deben cumplir restricciones temporales para operar bien.

Latch S-R

- Cronograma de tiempos: evolución del circuito en el tiempo.





- Modelo VHDL del latch S-R.

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

entity SRlatch is
port (S, R: in std_logic;
        Q, NO_Q: out std_logic);
end SRlatch;
```

```
architecture behav of SRlatch is
begin
process (S, R)
begin

assert (S = '0' or R = '0')   -- Detecta error en S y R a 1
report "Error: S y R están a 1"
severity error;

if (S = '1') then             -- Operacion de Set
   Q <= '1'; NO_Q <= '0';
elsif (R = '1') then          -- Operacion de Reset
   Q <= '0'; NO_Q <= '1';
end if;

end process;
end behav;
```

Si no se cumple la condición de *assert*, se muestra el mensaje de *report* y se realiza una acción según el grado de error de *severity*.

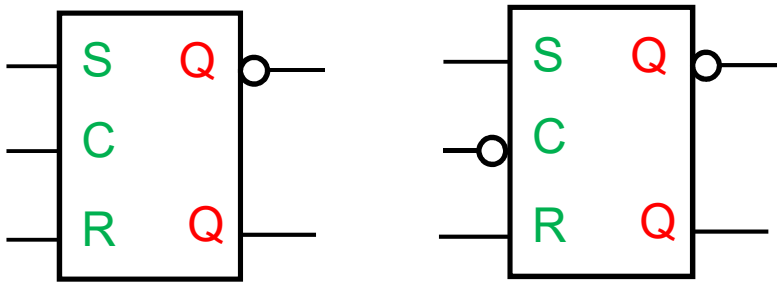
Si no se indica asignación se mantiene el valor anterior, luego con S = R = '0', Q se mantiene

“Clocked-Latch”

- El circuito S-R es un circuito asíncrono, por lo que su uso genera problemas al diseñar circuitos secuenciales. Lo usual es realizar los circuitos secuenciales como circuitos síncronos, con una señal periódica de reloj (clock), en los que el estado del circuito evoluciona solo una vez por ciclo de reloj. Para trabajar en estos circuitos hay que utilizar circuitos controlados por reloj, cuya versión inicial es el “clocked-latch”, aunque esta estructura de reloj no sea todavía válida para realizar circuitos síncronos.

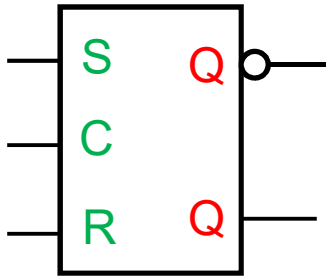
- Un “clocked-latch” tiene una señal de reloj C que cuando está inactiva hace que el flip-flop mantenga el dato, mientras que si está activa el flip-flop opera según su tabla de operación. Por ejemplo, para el circuito S-R. El reloj puede estar en polaridad positiva o negativa.

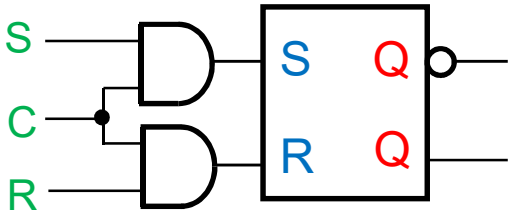
C	S	R	Q+	$\overline{Q}$+
0	X	X	Q	$\overline{Q}$
1	0	0	Q	$\overline{Q}$
1	0	1	0	1
1	1	0	1	0
1	1	1	No valido	

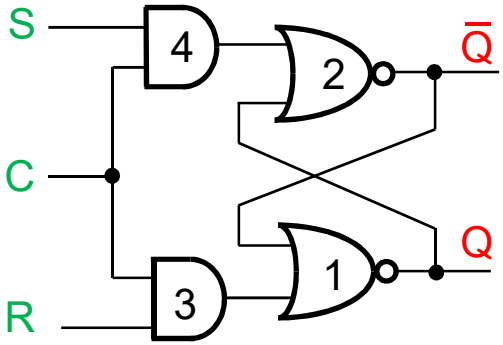


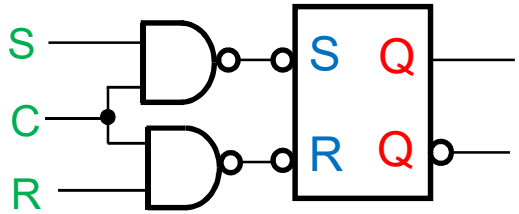
“Clocked-Latch”

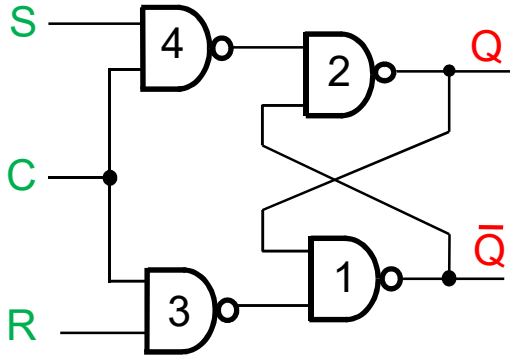
- La construcción de un “clocked-latch” puede hacerse intuitivamente desde el “latch” original, de forma que si el reloj está inactivo se genere una situación que mantenga el dato, y si está activo se comporte como el “latch” original.
  Para el “latch” S-R supone añadir puertas de tipo AND (AND en el NOR S-R, NAND en el NAND S-R), ya que si C es 0 las entradas del “latch” interno son 0 y si C es 1, las entradas externas pasan directamente a las entradas del “latch” interno.











Conversión entre flip-flops

- Un método más formal consiste en generar un flip-flop de un tipo FF2 en base a otro flip-flop interno FF1. En este caso el FF1 es un S-R y el FF2 es un S-R "clocked-latch". Los pasos para generar un flip-flop FF2 en base a un FF1 son los siguientes:

1. Plantear la tabla característica del FF2 a crear.
2. Obtener la tabla de transición o tabla de excitación del FF1. Esta tabla indica que valor deben tener las entradas de un flip-flop para que su salida haga cada una de las cuatro transiciones posibles: 0 -> 0, 0 -> 1, 1 -> 0 y 1 -> 1.
3. En cada fila de la tabla característica del FF2 hay que encontrar la relación Q -> Q+, e indicar los valores en las entradas del FF1 interno según su tabla de transición. Queda una tabla de verdad en la que las entradas del FF1 son función de los valores de las entradas de FF2 y de la salida Q (tanto de FF2 como de FF1).
4. Encontrar las ecuaciones que definen la lógica combinacional para generar las entradas del FF1 en función de las entradas de FF2 y de la salida del flip-flop.

“Clocked-Latch”

- Desarrollo de un S-R “clocked-latch” en base a un “latch” S-R.

1

C	Sc	Rc	Q	Q+
0	X	X	0	0
0	X	X	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	Ø
1	1	1	1	Ø

2

Q	Q+	S	R	S	R
0	0	0	0	0	Ø
		0	1		
0	1	1	0	1	0
1	0	0	1	0	1
1	1	0	0	Ø	0
		1	0		

3

C	Sc	Rc	Q	Q+	S	R
0	X	X	0	0	0	Ø
0	X	X	1	1	Ø	0
1	0	0	0	0	0	Ø
1	0	0	1	1	Ø	0
1	0	1	0	0	0	Ø
1	0	1	1	0	0	1
1	1	0	0	1	1	0
1	1	0	1	1	Ø	0
1	1	1	0	Ø	Ø	Ø
1	1	1	1	Ø	Ø	Ø

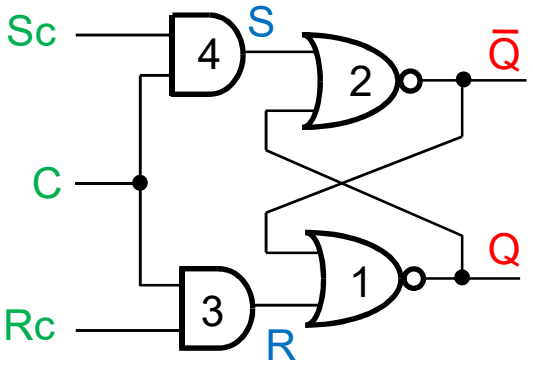
4

CQ \ ScRc	00	01	11	10
00	0	0	0	0
01	Ø	Ø	Ø	Ø
11	Ø	0	Ø	Ø
10	0	0	Ø	1

$S = C\ Sc$

CQ \ ScRc	00	01	11	10
00	Ø	Ø	Ø	Ø
01	0	0	0	0
11	0	1	Ø	0
10	Ø	Ø	Ø	0

$R = C\ Rc$



Flip-flop D

- Flip-Flop D (Delay) realiza la función característica Q+ = D. Es el flip-flop más utilizado en circuitos síncronos ya que pasa el valor de la entrada a la salida. El mantenimiento del dato Q+ = Q, se realiza mediante la señal de reloj: cuando está inactiva se mantiene el dato.
  El desarrollo de un flip-flop de tipo "clocked-latch" se puede hacer en base a un "latch" S-R interno añadiendo un reloj mediante puertas AND.

D	Q+
0	0
1	1

D	Q	Q+	S	R
0	0	0	0	Ø
0	1	0	0	1
1	0	1	1	0
1	1	1	Ø	0

Tabla de excitación

Q	Q+	D
0	0	0
0	1	1
1	0	0
1	1	1

D \ Q	0	1
0	0	0
1	1	Ø

S = D

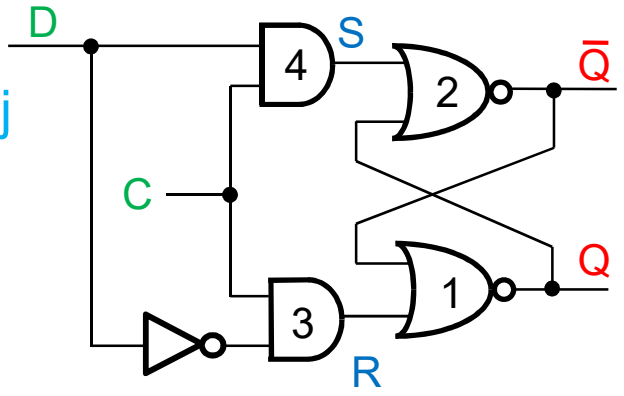
D \ Q	0	1
0	Ø	1
1	0	0

R = $\overline{D}$

Al añadir el reloj

S = C D

R = C $\overline{D}$



Flip-flop T

- Flip-Flop T (“Toogle”). El desarrollo de un flip-flop de tipo “clocked-latch” se puede hacer en base a un latch S-R interno añadiendo un reloj mediante puertas AND.

T	Q+
0	Q
1	$\overline{Q}$

T	Q	Q+	S	R
0	0	0	0	Ø
0	1	1	Ø	0
1	0	1	1	0
1	1	0	0	1

Tabla de excitación

Q	Q+	T
0	0	0
0	1	1
1	0	1
1	1	0

T \ Q	0	1
0	0	Ø
1	1	0

$S = T\,\overline{Q}$

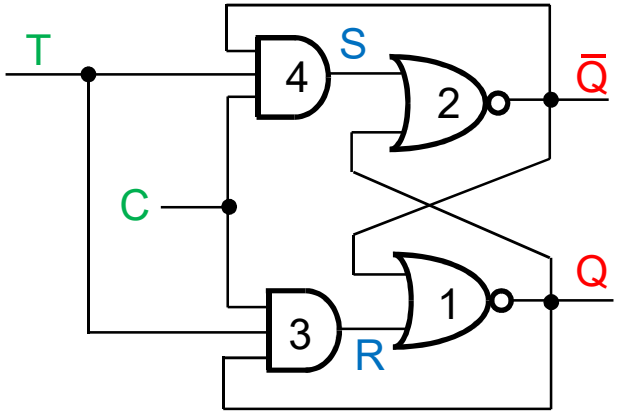
T \ Q	0	1
0	Ø	0
1	0	1

$R = T\,Q$

Al añadir el reloj

$S = C\,T\,\overline{Q}$

$R = C\,T\,Q$



Flip-flop J-K

J	K	Q+
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

J	K	Q	Q+	S	R
0	0	0	0	0	Ø
0	0	1	1	Ø	0
0	1	0	0	0	Ø
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	Ø	0
1	1	0	1	1	0
1	1	1	0	0	1

Tabla de excitación

Q	Q+	J	K	J	K
0	0	0	0	0	Ø
		0	1		
0	1	1	0	1	Ø
		1	1		
1	0	0	1	Ø	1
		1	1		
1	1	0	0	Ø	0
		1	0		

Q \ JK	00	01	11	10
0	0	0	1	1
1	Ø	0	0	Ø

$S = J\,\overline{Q}$

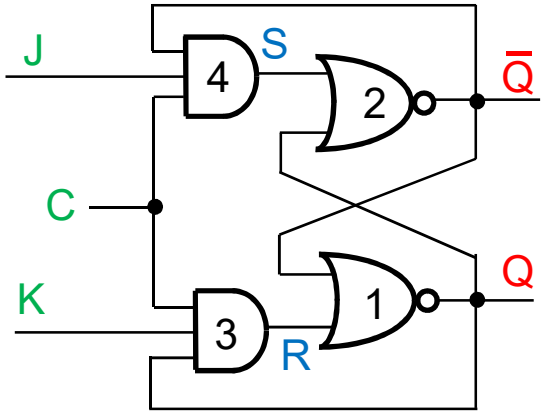
Q \ JK	00	01	11	10
0	Ø	Ø	0	0
1	0	1	1	0

$R = K\,Q$

Al añadir el reloj

$S = C\,J\,\overline{Q}$

$R = C\,K\,Q$



Conversión entre flip-flops

- Desarrollo de un flip-flop J-K en base a un flip-flop D.

1

J	K	Q	Q+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

2

Q	Q+	D
0	0	0
0	1	1
1	0	0
1	1	1

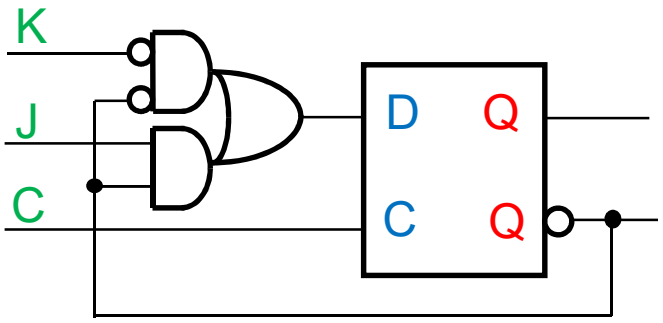
3

J	K	Q	Q+	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	0	0

4

Q \ JK	00	01	11	10
0	0	0	1	1
1	1	0	0	1

$$D = J\,\overline{Q} + \overline{K}\,Q$$



Conversión entre flip-flops

- Desarrollo de un flip-flop D en base a un flip-flop J-K.

1

D	Q	Q+
0	0	0
0	1	0
1	0	1
1	1	1

2

Q	Q+	J	K	J	K
0	0	0	0	0	Ø
		0	1		
0	1	1	0	1	Ø
		1	1		
1	0	0	1	Ø	1
		1	1		
1	1	0	0	Ø	0
		1	0		

3

D	Q	Q+	J	K
0	0	0	0	Ø
0	1	0	Ø	1
1	0	1	1	Ø
1	1	1	Ø	0

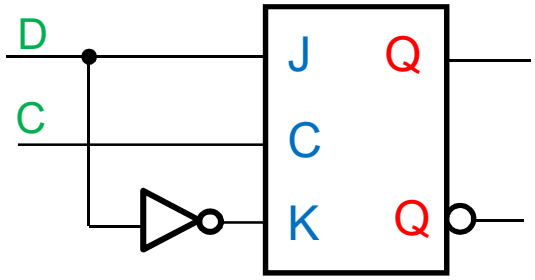
4

D \ Q	0	1
0	0	Ø
1	1	Ø

$J = D$

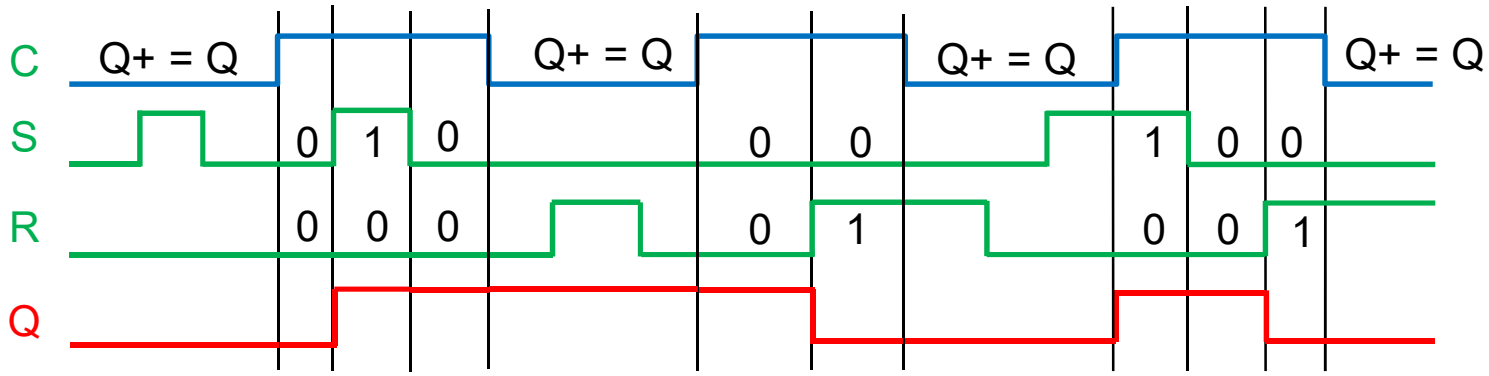
D \ Q	0	1
0	Ø	1
1	Ø	0

$K = \overline{D}$



“Clocked-Latch”

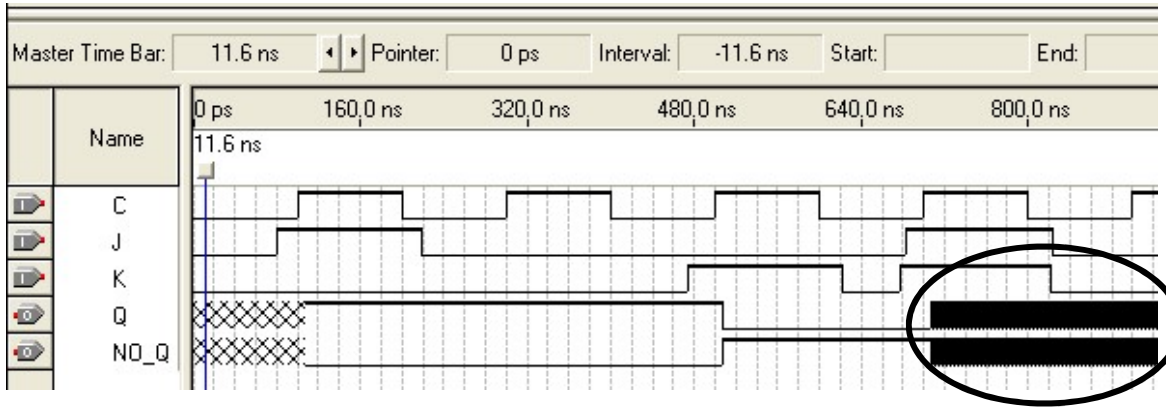
- Cronograma de tiempos: evolución del circuito en el tiempo.



- Problemas temporales:

  - Cambios simultáneos y anchura mínima de las señales de reloj y de entrada: mismos problemas que en el latch S-R.

  - Mientras el reloj está activo los datos pueden realimentarse varias veces (por ejemplo en un J-K con entradas a 1). Este circuito todavía no es válido para el diseño de circuitos síncronos.



J = K = 1 => Q+ <= $\overline{Q}$, el circuito oscila

“Clocked-Latch”

- Modelo VHDL de flip-flops D y J-K

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

entity Dclocked is
port (D, C: in std_logic;
        Q, NO_Q: out std_logic);
end Dclocked;

architecture comp of Dclocked is
begin
process (C, D)
   begin
   if (C = ‘1’) then  -- Carga el valor
       Q <= D;
       NO_Q <= not D;
   end if;
end process;
end comp;
```

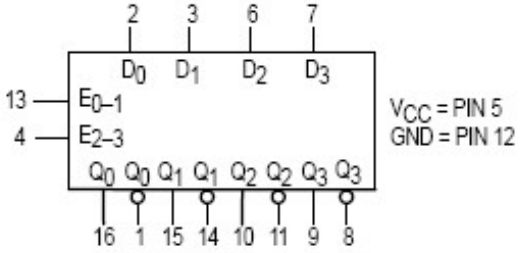
Como Q es de salida, se usa la señal *inter* para realimentar el valor interno, luego se asigna esa señal a Q. La sentencia Q <= not Q es ilegal.

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

entity JKclocked is
port (J, K, C: in std_logic;
         Q, NO_Q: out std_logic);
end JKclocked;

architecture comp of JKclocked is
-- Se utiliza una señal para Q interno
signal Q_I: std_logic;

begin
process (C, J, K, Q_I)
variable entradas: std_logic_vector(2 downto 1);
begin
    if (C = ‘1’) then        -- Carga el valor
       entradas := J & K;
       case entradas is
            when “01” => Q_I <= ‘0’;
            when “10” => Q_I <= ‘1’;
            when “11” => Q_I <= not Q_I;
            when others => null;
       end case;
  end if;
end process;
Q <= Q_I;     -- Se obtiene la salida
NO_Q <= not Q_I;
end comp;
```

“Clocked-Latch”

- Circuito 74LS75: D “clocked-latch”. Está construido básicamente con un multiplexor de dos entradas: si Enable (o C, entrada S del Mux) es 1, Q se carga con Data (D, en I1 del Mux); si Enable es 0, Q mantiene el dato (en I0 del Mux).

SN54/74LS75

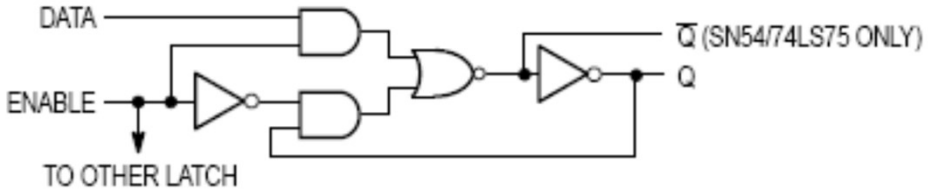


TRUTH TABLE
(Each latch)

t_n	t_{n+1}
D	Q
H	H
L	L

NOTES:
t_n = bit time before enable negative-going transition
t_{n+1} = bit time after enable negative-going transition

LOGIC DIAGRAM



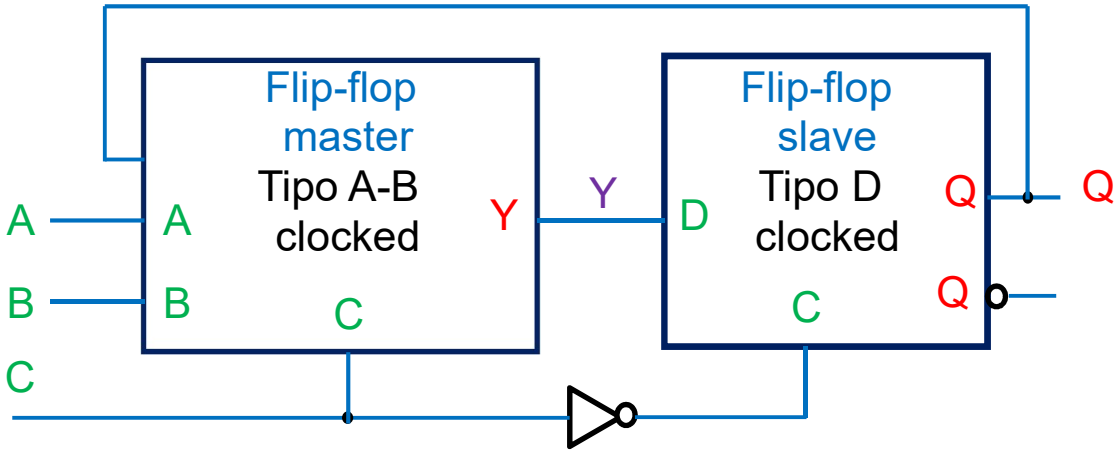
AC CHARACTERISTICS (T_A = 25°C, V_{CC} = 5.0 V)

Symbol	Parameter	Limits Min	Limits Typ	Limits Max	Unit	Test Conditions
t_{PLH} t_{PHL}	Propagation Delay, Data to Q		15 9.0	27 17	ns	V_{CC} = 5.0 V C_L = 15 pF
t_{PLH} t_{PHL}	Propagation Delay, Data to $\overline{Q}$		12 7.0	20 15	ns	
t_{PLH} t_{PHL}	Propagation Delay, Enable to Q		15 14	27 25	ns	
t_{PLH} t_{PHL}	Propagation Delay, Enable to $\overline{Q}$		16 7.0	30 15	ns	

“Master-Slave”

- Para obtener circuitos síncronos se necesita que el circuito evolucione una vez por ciclo de reloj, donde lo mejor es una vez por flanco (de subida o de bajada) del reloj.
- La estructura “master-slave” se puede utilizar como una estructura síncrona, que evoluciona una vez por ciclo de reloj, aunque en algún tipo de flip-flop (J-K, por ejemplo) pueden presentar problemas de operación. Está basada en una estructura formada por dos “clocked-latch” controlados por fases complementadas de reloj (uno activo a alto y otro activo bajo).

  El primer latch o *maestro* (“master”) realiza la función lógica, mientras que el segundo latch o *esclavo* (“slave”) es de tipo D y pasa el dato a la salida.

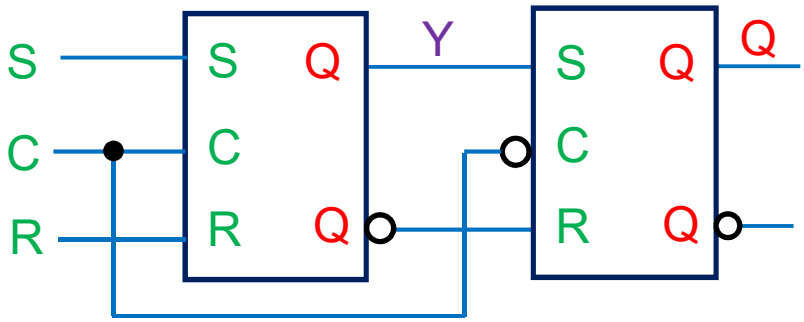


Y y Q deberían tener el mismo valor. Al cargar Y (con C a 1), puede que Y sea distinto de Q, lo que produce algún problema de operación.

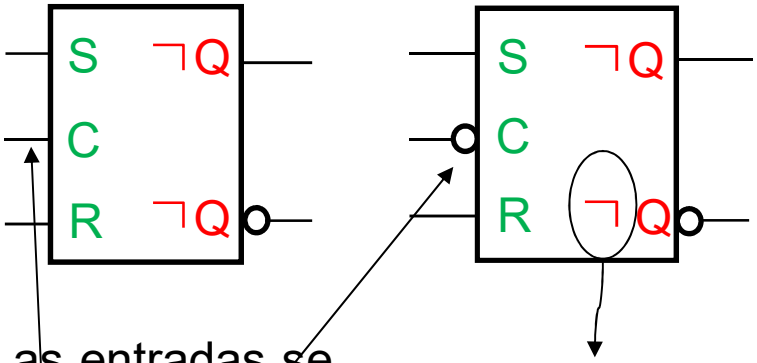
C	Y+	Q+
0	Y	Y
1	F(A,B,Y,Q)	Q

“Master-Slave”

C	S	R	Y+	Q+
0	X	X	Y	Y
1	0	0	Y(Q)	Q
1	0	1	0	Q
1	1	0	1	Q
1	1	1	No valido	

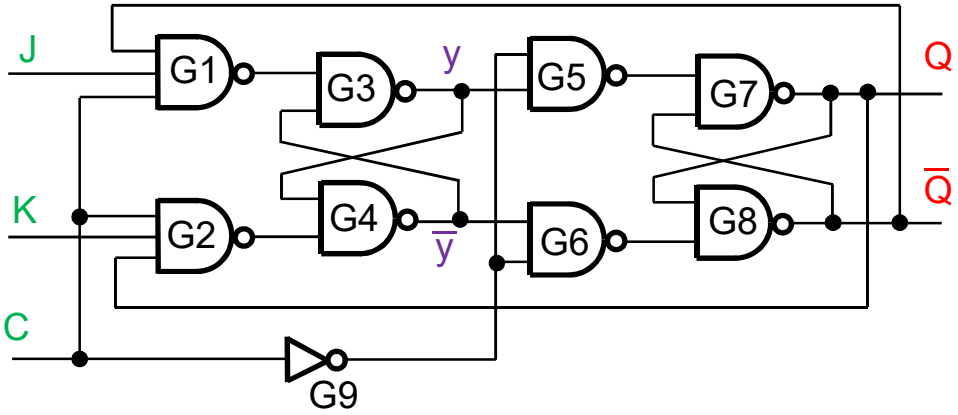


S-R Master-Slave



Las entradas se leen cuando el reloj está activo a H o a L

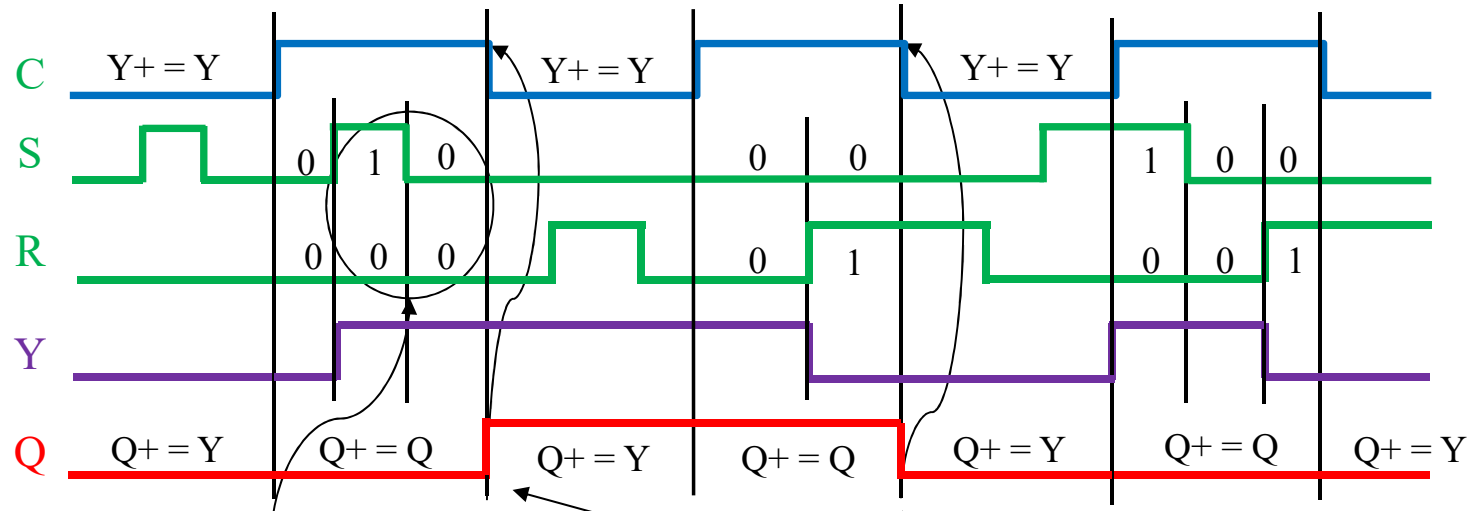
Indica “salida pospuesta”: la salida cambia cuando el reloj pasa de activo a inactivo



J-K Master-Slave

“Master-Slave”

- Cronograma de tiempos: evolución del circuito en el tiempo.
  Inicialmente CLK a 0: el master mantiene el valor de Y, Y+ = Y; el slave opera como un tipo D, Q+ = Y.
  CLK pasa a 1: el master está activado, Y+ = F(S,R,Y,Q); el slave mantiene el dato en Q+ = Q.
  CLK pasa a 0: el master mantiene el valor de Y, Y+ = Y; el slave opera como un tipo D, Q+ = Y.

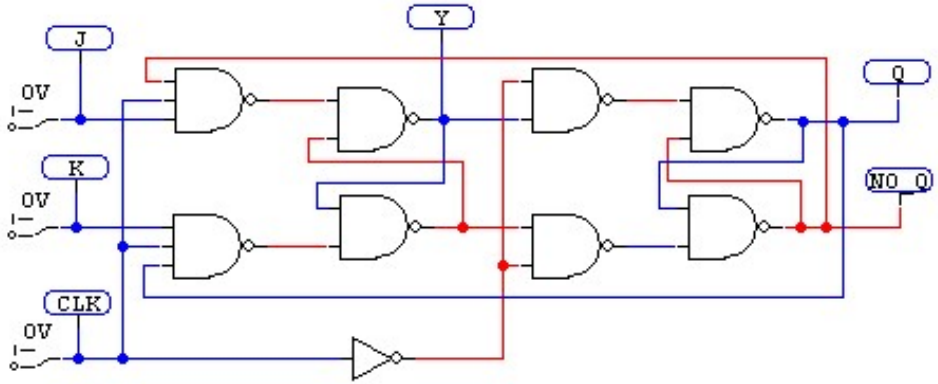


Problema: Y y Q son distintos al cambiar Y. Las entradas S y R a 0, por definición deberían mantener Q pero por construcción mantienen Y. Las entradas no se leen por flanco de reloj sino por nivel

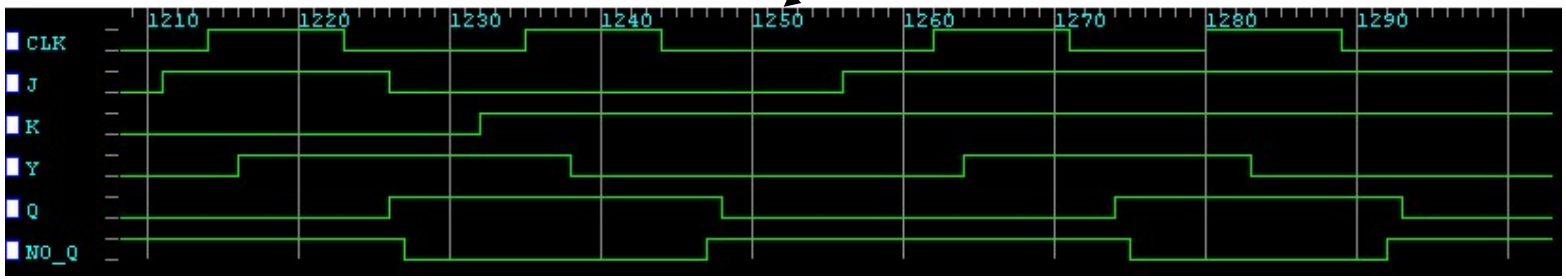
La salida solo cambia una vez por ciclo de reloj en uno de sus flancos (bajada en este caso)

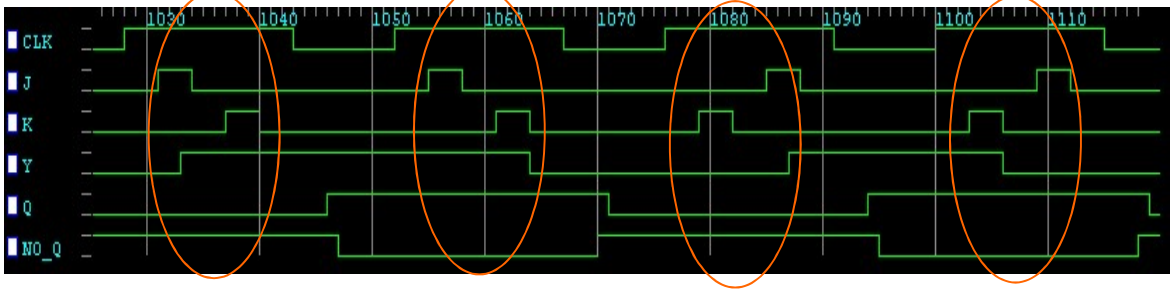
“Master-Slave”

- Cronograma de tiempos: J-K master slave.



Operación correcta: las entradas se fijan antes de que se active el master y permanecen estables en todo el intervalo





Captura de 1s en el J-K: Si una entrada del master está a 1 con el reloj activo se comporta como si estuviese a 1 en todo el intervalo: estos casos son equivalentes a la situación J = K = 1.

“Master-Slave”

- Modelo VHDL de flip-flops D y J-K master-slave

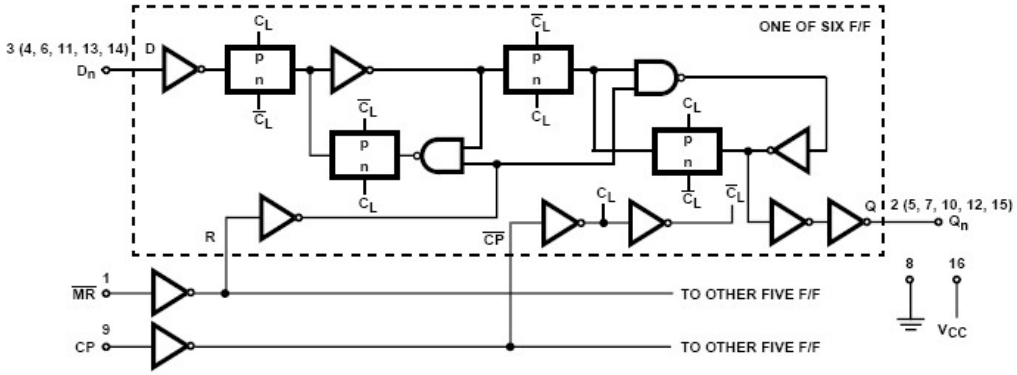
```
library ieee;
use ieee.std_logic_1164.all;

entity D_MS is
port (D, C: in std_logic;
        Q, NO_Q: out std_logic);
end D_MS;

architecture comp of D_MS is
signal Y: std_logic;

begin
process (C, D)
   begin
   if (C = ‘1’) then  -- Carga el master
      Y <= D;
   else                    -- Carga el slave
      Q <= Y;
      NO_Q <= not Y;
   end if;
end process;
end comp;
```

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

entity JK_MS is
port (J, K, C: in std_logic;
        Q, NO_Q: out std_logic);
end JK_MS;

architecture comp of JK_MS is
signal Y, Q_I: std_logic;

begin
process (C, J, K, Y, Q_I)
variable entradas: std_logic_vector(2 downto 1);
begin
    if (C = ‘1’) then        -- Carga el master
       entradas := J & K;
       case entradas is
            when “01” => Y <= ‘0’;
            when “10” => Y <= ‘1’;
            when “11” => Y <= not Q_I;
            when others => null;    -- o Y <= Q_I;
       end case;
    else
       Q_I <= Y;              -- Carga el slave
   end if;
end process;
Q <= Q_I;     -- Se obtiene la salida
NO_Q <= not Q_I;
end comp;
```

“Master-Slave”

- Circuito 74HC174: flip-flop D master-slave. Incluye una entrada de inicialización (MR, Reset para puesta a 0) asíncrona.

CD54HC174, CD74HC174,
CD54HCT174, CD74HCT174



Hex D-Type Flip-Flop with Reset

INPUTS			OUTPUT
RESET ($\overline{MR}$)	CLOCK CP	DATA D_n	Q_n
L	X	X	L
H	↑	H	H
H	↑	L	L
H	L	X	Q_0

H = High Voltage Level, L = Low Voltage Level, X = Irrelevant, ↑ = Transition from Low to High Level, Q_0 = Level Before the Indicated Steady-State Input Conditions Were Established

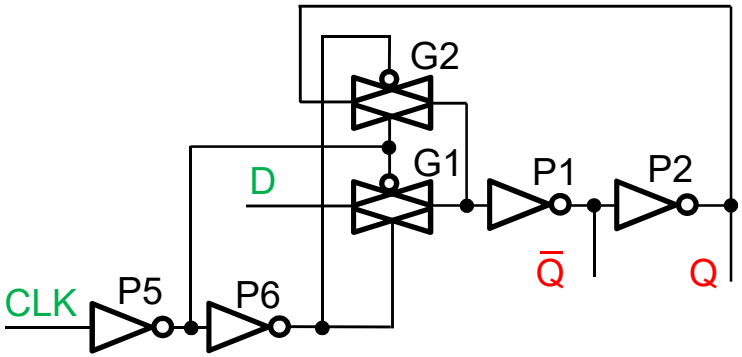
PARAMETER	SYMBOL	TEST CONDITIONS	V_{CC} (V)	25°C		-40°C TO 85°C		-55°C TO 125°C		UNITS
				MIN	MAX	MIN	MAX	MIN	MAX	
Clock Pulse Width	t_w	-	4.5	20	-	25	-	30	-	ns
$\overline{MR}$ Pulse Width	t_w	-	6	25	-	31	-	38	-	ns
Setup Time, Data to Clock	t_{SU}	-	4.5	16	-	20	-	24	-	ns
Hold Time, Data to Clock	t_H	-	6	5	-	5	-	5	-	ns
Removal Time, $\overline{MR}$ to Clock	t_{REM}	-	4.5	12	-	15	-	18	-	ns
Clock Frequency	f_{MAX}	-	6	25	-	20	-	17	-	MHz

PARAMETER	SYMBOL	TEST CONDITIONS	V_{CC} (V)	25°C		-40°C TO 85°C	-55°C TO 125°C	UNITS
				TYP	MAX	MAX	MAX	
Propagation Delay, Clock to Q	t_{PLH}, t_{PHL}	C_L = 50pF	4.5	-	40	50	60	ns
		C_L = 15pF	5	17	-	-	-	ns
Propagation Delay, $\overline{MR}$ to Q	t_{PLH}, t_{PHL}	C_L = 50pF	4.5	-	44	55	66	ns
		C_L = 15pF	5	18	-	-	-	ns

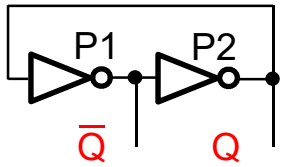
Parámetros temporales

“Master-Slave”

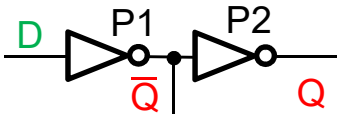
D Clocked-latch



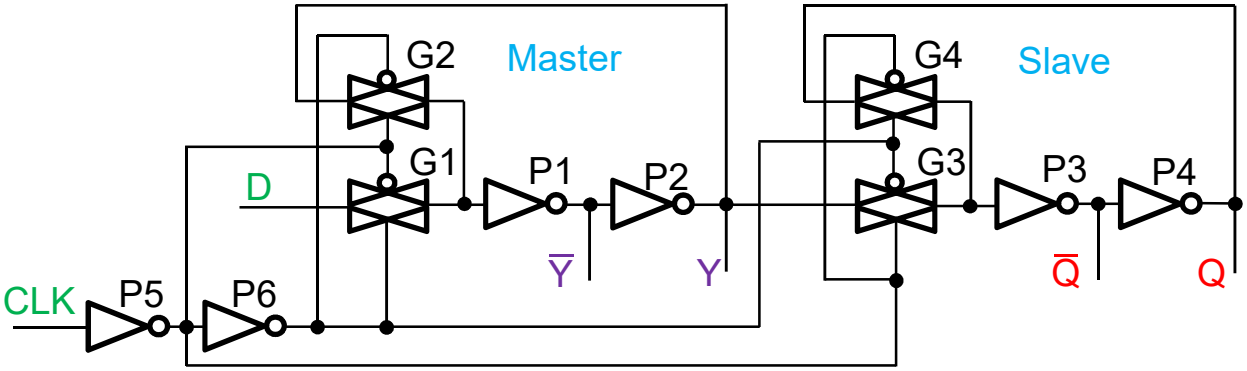
Si CLK = 0 => G1 OFF, G2 ON
Q+ = Q



Si CLK = 1 => G1 ON, G2 OFF
Q+ = D



D Master-Slave



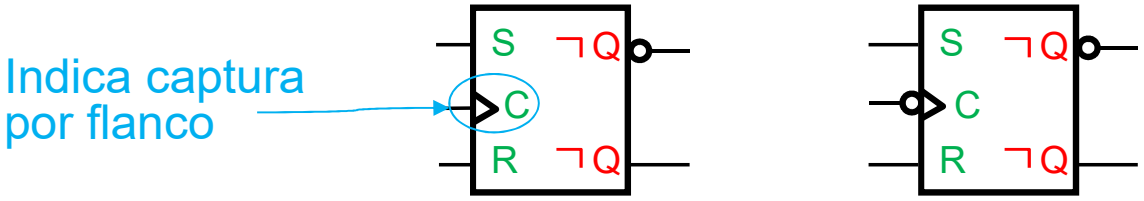
Si CLK = 0 => G1 OFF, G2 ON; G3 ON, G4 OFF => Y+ = Y; Q+ = Y
Si CLK = 1 => G1 ON, G2 OFF; G3 OFF, G4 ON => Y+ = D; Q+ = Q

Flip-flops disparados por flanco

- Los problemas temporales en los flip-flop master-slave se deben a que en la fase activa del reloj el master lee las entradas por nivel, por ello cambios en las entradas pueden producir variaciones indeseadas en el flip-flop.
- Para solucionar este problema se puede:

  - Imponer restricciones temporales a los cambios de las entradas.

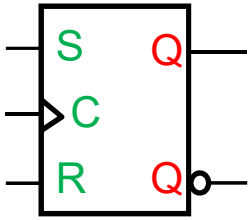
  - Diseñar flip-flops para que solo lean las entradas en el flanco que activa el master: flip-flops master-slave data-lockout.



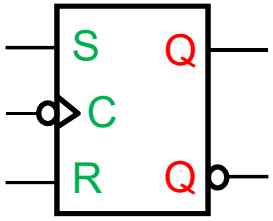
- La mejor solución consiste en el diseño de flip-flops disparados por flanco: la lectura de los datos de entrada y los cambios en la salida se realizan en función de la aparición de un flanco en la señal de reloj. El flanco puede ser positivo (Clk de L-> H) o negativo (Clk de H->L).
  El flip-flop D master-slave se comporta como un flip-flop disparado por flanco (el que activa el slave). Utilizando las técnicas de conversión entre flip-flops se pueden diseñar otros flip-flops master-slave que se comporten como disparados por flanco (por ejemplo el CD4027 J-K master-slave).

Flip-flops disparados por flanco

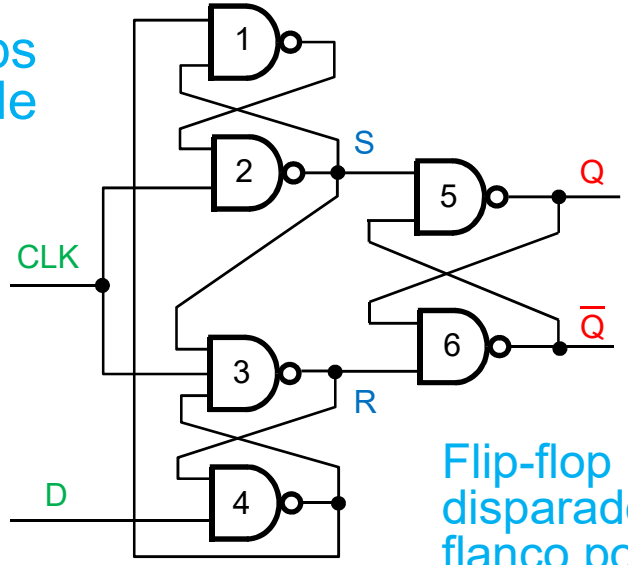
- Para el diseño de flip-flops disparados se utilizan técnicas de diseño de circuitos asíncronos.



Disparo por flanco positivo

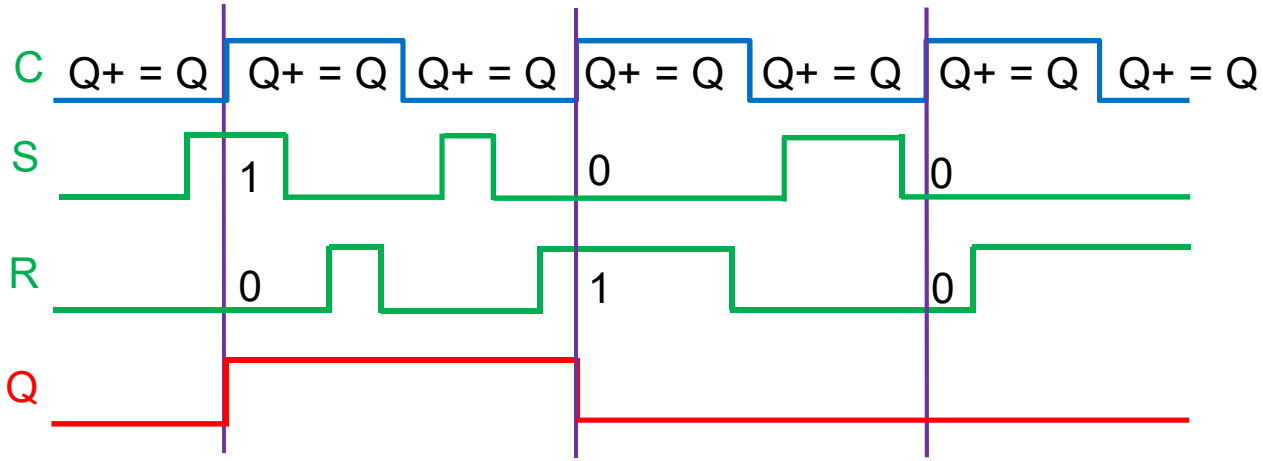


Disparo por flanco negativo



Flip-flop D disparado por flanco positivo

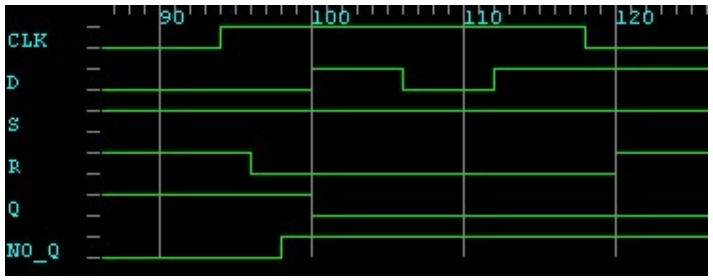
- Los flip-flops disparados por flanco operan en torno al flanco positivo o negativo de disparo, el resto del tiempo el flip-flop mantiene su estado.

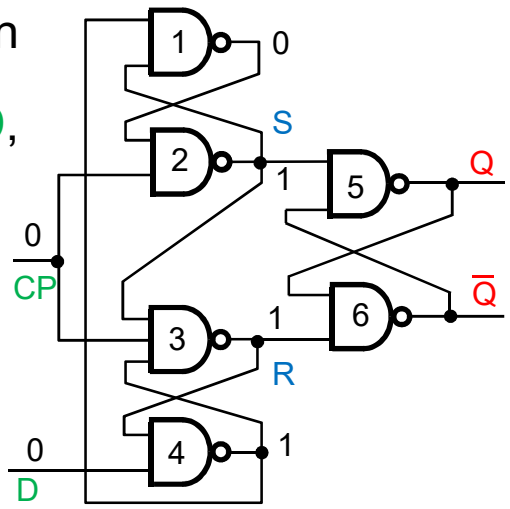


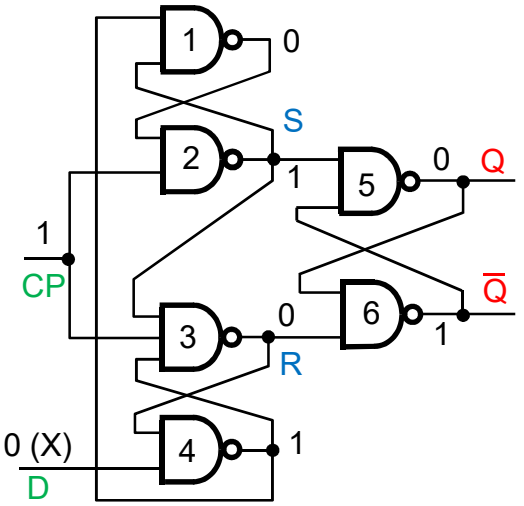
Flip-flops disparados por flanco

- Operación del flip-flop D disparado por flanco positivo.
  Situación inicial reloj a 0 => S = R = 1 => Q = Q+

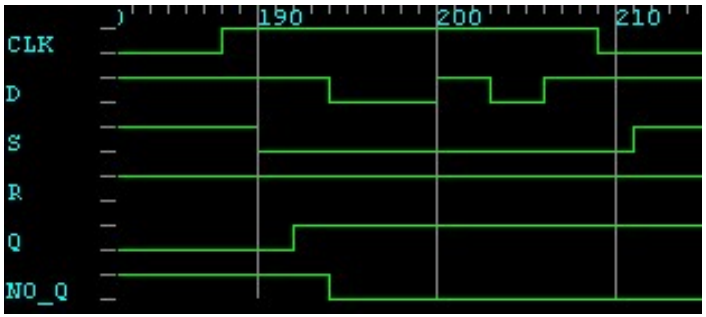
Entrada D a 0. Flanco positivo en el reloj => S = 1, R = 0 => Q = 0. R a 0 bloquea los cambios en D, el circuito queda estable.

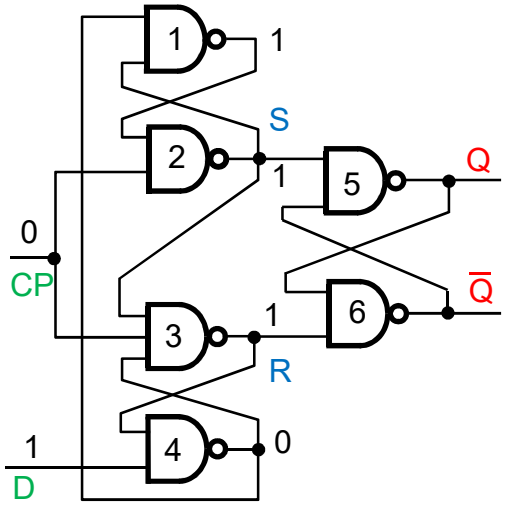


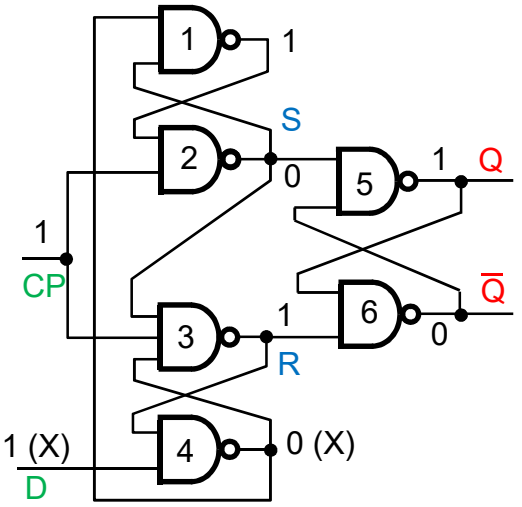




Entrada D a 1. Flanco positivo en el reloj => S = 0, R = 1 => Q = 1. S a 0 bloquea los cambios en puertas 3 y 1, el circuito queda estable.







Flip-flops disparados por flanco

- Modelo VHDL de flip-flops D y J-K disparados por flanco

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

entity D_FF is
port (D, C: in std_logic;
        Q, NO_Q: out std_logic);
end D_FF;

architecture comp of D_FF is

begin
process (C)      -- No hace falta D
  begin
  -- Detecta flanco positivo en CLK
  if (C'EVENT and C = '1') then
     Q <= D;
     NO_Q <= not D;
  end if;
end process;
end comp;
```

(C'EVENT and C = '0') detecta un flanco negativo en C

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

entity JK_FF is
port (J, K, C: in std_logic;
        Q, NO_Q: out std_logic);
end JK_FF;

architecture comp of JK_FF is
signal Q_I: std_logic;

begin
process (C)        -- No hacen falta J, K
variable entradas: std_logic_vector(2 downto 1);
begin
   if (C'EVENT and C = '1') then
      entradas := J & K;
      case entradas is
          when "01" => Q_I <= '0';
          when "10" => Q_I <= '1';
          when "11" => Q_I <= not Q_I;
          when others => null;
      end case;
   end if;
end process;
Q <= Q_I;     -- Se obtiene la salida
NO_Q <= not Q_I;
end comp;
```

Flip-flops disparados por flanco

- Los flip-flops y los circuitos secuenciales en general tienen señales de control asíncronas (independiente del reloj) o síncronas (dependiente del reloj) que permiten realizar inicializaciones de las señales u otras operaciones como mantenimientos de dato (CE o CS).
- Es necesario inicializar un circuito secuencial o un flip-flop ya que al alimentarlo no se conoce en que estado ha quedado. Antes de operar hay que llevar las salidas de los flip-flops a un estado conocido. Los flip-flops suelen tener entradas de Reset (o Clear) para fijar a 0 la salida y de Set (o Preset) para fijar a 1 la salida.

```
architecture comp of D_FF_ResA is

begin
process (C, Reset)
   begin
   if ( Reset = '1' ) then
      Q <= '0'; NO_Q <= '1';
   elsif (C'EVENT and C = '1') then
      Q <= D;
      NO_Q <= not D;
   end if;
end process;
end comp;
```

Flip-flop D disparado por flanco positivo con Reset asíncrono

```
architecture comp of D_FF_ResS is

begin
process (C)
   begin
   if (C'EVENT and C = '1') then
      if ( Reset = '1' ) then
         Q <= '0'; NO_Q <= '1';
      else
         Q <= D;
         NO_Q <= not D;
      end if;
   end if;
end process;
end comp;
```

Flip-flop D disparado por flanco positivo con Reset síncrono

Flip-flops disparados por flanco

- Los flip-flops deben cumplir unos requerimientos temporales para que operen correctamente. Por ejemplo, si en un flip-flop D disparado por flanco se genera el flanco en el reloj cuando cambia la entrada D, ¿qué valor se carga en el flip-flop?.
- Los tiempos definidos en las hojas de características son los siguientes:

  - Tiempo de establecimiento (Tsetup): tiempo mínimo que deben estar las entradas de datos estables antes de que aparezca el flanco de reloj que realiza la captura de datos.

  - Tiempo de mantenimiento (Thold): tiempo mínimo que deben mantenerse las entradas de datos estables después de que haya aparecido el flanco de reloj que realiza la captura de datos.

  - Tw(H): tiempo mínimo que debe mantenerse la señal de reloj, u otra señal de control a valor H.

  - Tw(L): tiempo mínimo que debe mantenerse la señal de reloj, u otra señal de control a valor L.

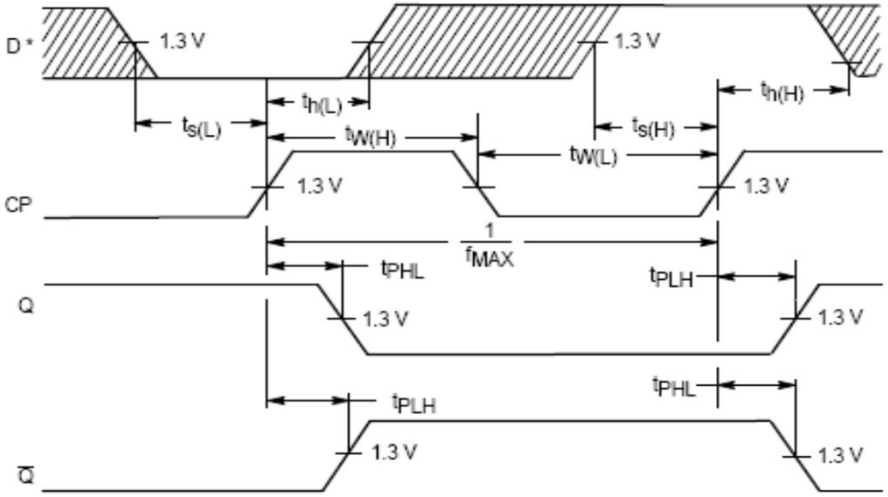
Flip-flops disparados por flanco

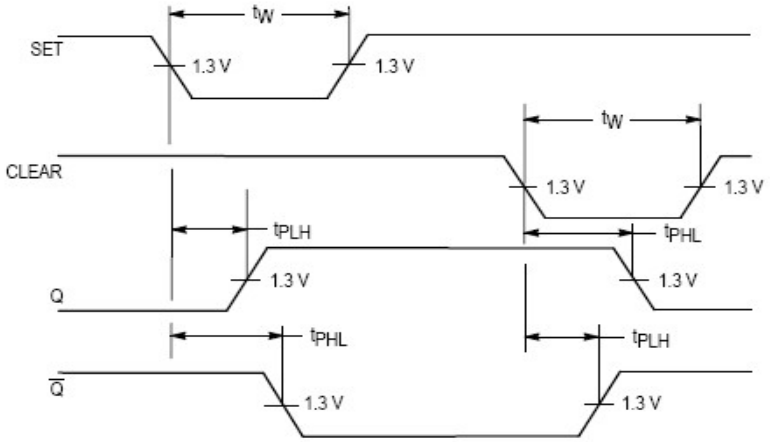
- Además las hojas de características muestra parámetros temporales de los circuitos:

  - Fmax: máxima frecuencia de la señal de reloj. Para frecuencias más altas no se asegura el correcto funcionamiento del flip-flop.

  - Tiempos de propagación (TpLH o TpHL):

  En dispositivos del tipo "latch" se indican los Tp desde las entradas de datos o de reloj hasta la salida.
  En dispositivos de tipo síncrono los tiempos se miden desde el flanco en el reloj hasta la salida del flip-flop. Los cambios en las entradas de datos D, J-K, etc, no producen variaciones en la salida.

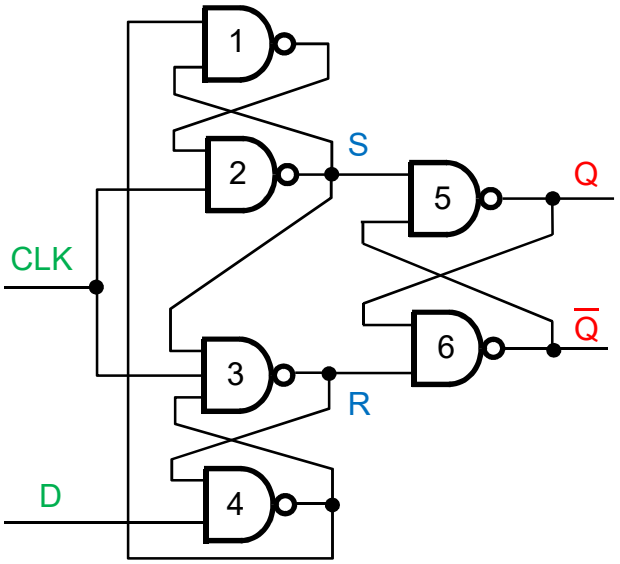




Flip-flops disparados por flanco

- Evaluación del Tp. De CLK a Q, $\overline{Q}$.
  - Si R pasa a 0 cambian las puertas 3, 6 ($\overline{Q}$) y 5 (Q).
  Tp1 = tp(3) + tp(6) + tp(5).
  - Si S pasa a 0 cambian las puertas 2, 5 (Q), y 6 ($\overline{Q}$).
  Tp2 = tp(2) + tp(5) + tp(6).

  Tp = máximo {Tp1, Tp2}.



- Evaluación del Tsetup. Cambia D.
  Antes del flanco CLK = 0, S = R = 1. El cambio en D se propaga por las puertas 4 y 1. Tsetup = tp(4) + tp(1).

- Evaluación del Thold. Flanco positivo en CLK.
  - Si R pasa a 0 (tp(3)). La entrada R de la puerta 4 se fija a 0, luego el circuito queda indiferente de la entrada D. Thold1 = tp(3).
  - Si S pasa a 0 (tp(2)). La entrada S de las puertas 1 y 3 se fija a 0, luego el circuito queda indiferente del valor de la salida de la puerta 4. Como los cambios en D se tienen que propagar hasta la puerta 4 Thold2 = tp(2) - tp(4).

  Thold = máximo {Thold1, Thold2}.

Flip-flops disparados por flanco

- Circuito 74LS74: D disparado por flanco positivo. Incluye dos entradas de inicialización CD (Clear o Reset para puesta a 0) y SD (Set o Preset para puesta a 1) asíncronas.

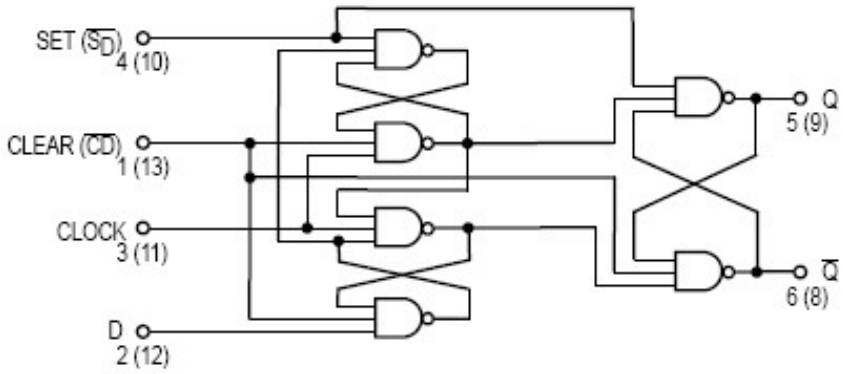
SN54/74LS74A — **DUAL D-TYPE POSITIVE EDGE-TRIGGERED FLIP-FLOP**

MODE SELECT — TRUTH TABLE

OPERATING MODE	INPUTS			OUTPUTS	
	$\overline{S_D}$	$\overline{S_D}$	D	Q	$\overline{Q}$
Set	L	H	X	H	L
Reset (Clear)	H	L	X	L	H
*Undetermined	L	L	X	H	H
Load "1" (Set)	H	H	h	H	L
Load "0" (Reset)	H	H	l	L	H

* Both outputs will be HIGH while both S_D and C_D are LOW, but the output states are unpredictable if S_D and C_D go HIGH simultaneously. If the levels at the set and clear are near V_{IL} maximum then we cannot guarantee to meet the minimum level for V_{OH}.

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

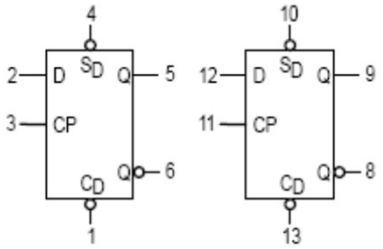


AC CHARACTERISTICS (T_A = 25°C, V_{CC} = 5.0 V)

Symbol	Parameter	Limits Min	Typ	Max	Unit	Test Conditions	
f_{MAX}	Maximum Clock Frequency	25	33		MHz	Figure 1	V_{CC} = 5.0 V, C_L = 15 pF
t_{PLH} t_{PHL}	Clock, Clear, Set to Output		13	25	ns	Figure 1	
			25	40	ns		

AC SETUP REQUIREMENTS (T_A = 25°C)

Symbol	Parameter	Limits Min	Typ	Max	Unit	Test Conditions	
$t_{W(H)}$	Clock	25			ns	Figure 1	V_{CC} = 5.0 V
$t_{W(L)}$	Clear, Set	25			ns	Figure 2	
t_s	Data Setup Time — HIGH	20			ns	Figure 1	
	LOW	20			ns		
t_h	Hold Time	5.0			ns	Figure 1	



V_{CC} = PIN 14
GND = PIN 7

Flip-flops disparados por flanco

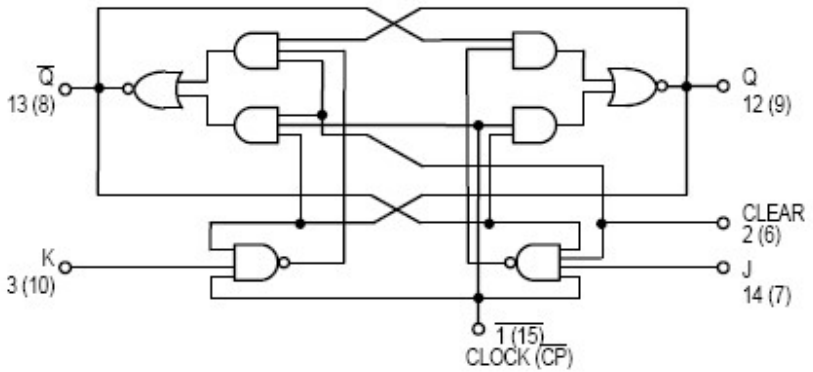
- Circuito 74LS73: J-K disparado por flanco negativo. Existen versiones de este dispositivo (7473) en estructura master-slave. Incluye una entrada de inicialización (MR, Reset para puesta a 0) asíncrona.

SN54/74LS73A DUAL JK NEGATIVE EDGE-TRIGGERED FLIP-FLOP

MODE SELECT — TRUTH TABLE

OPERATING MODE	INPUTS			OUTPUTS	
	$\overline{C}_D$	J	K	Q	$\overline{Q}$
Reset (Clear)	L	X	X	L	H
Toggle	H	h	h	$\overline{q}$	q
Load "0" (Reset)	H	l	h	L	H
Load "1" (Set)	H	h	l	H	L
Hold	H	l	l	q	$\overline{q}$

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

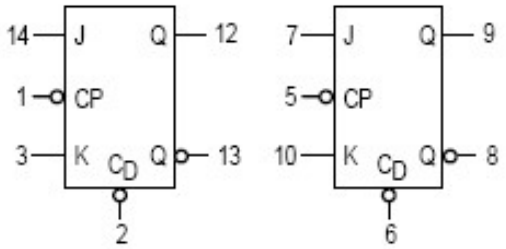


AC CHARACTERISTICS (T_A = 25°C, V_{CC} = 5.0 V)

Symbol	Parameter	Limits Min	Limits Typ	Limits Max	Unit	Test Conditions	
f_{MAX}	Maximum Clock Frequency	30	45		MHz	Figure 1	V_{CC} = 5.0 V, C_L = 15 pF
t_{PLH} / t_{PHL}	Propagation Delay, Clock to Output		15	20	ns	Figure 1	
			15	20	ns		

AC SETUP REQUIREMENTS (T_A = 25°C)

Symbol	Parameter	Limits Min	Limits Typ	Limits Max	Unit	Test Conditions	
t_W	Clock Pulse Width High	20			ns	Figure 1	V_{CC} = 5.0 V
t_W	Clear Pulse Width	25			ns	Figure 2	
t_s	Setup Time	20			ns	Figure 1	
t_h	Hold Time	0			ns		



Circuitos temporizadores

- Los flip-flops son circuitos biestables que tienen dos estados estables, durante el funcionamiento el circuito pasa de un estado a otro. Mediante realimentaciones también se pueden conseguir otros circuitos:

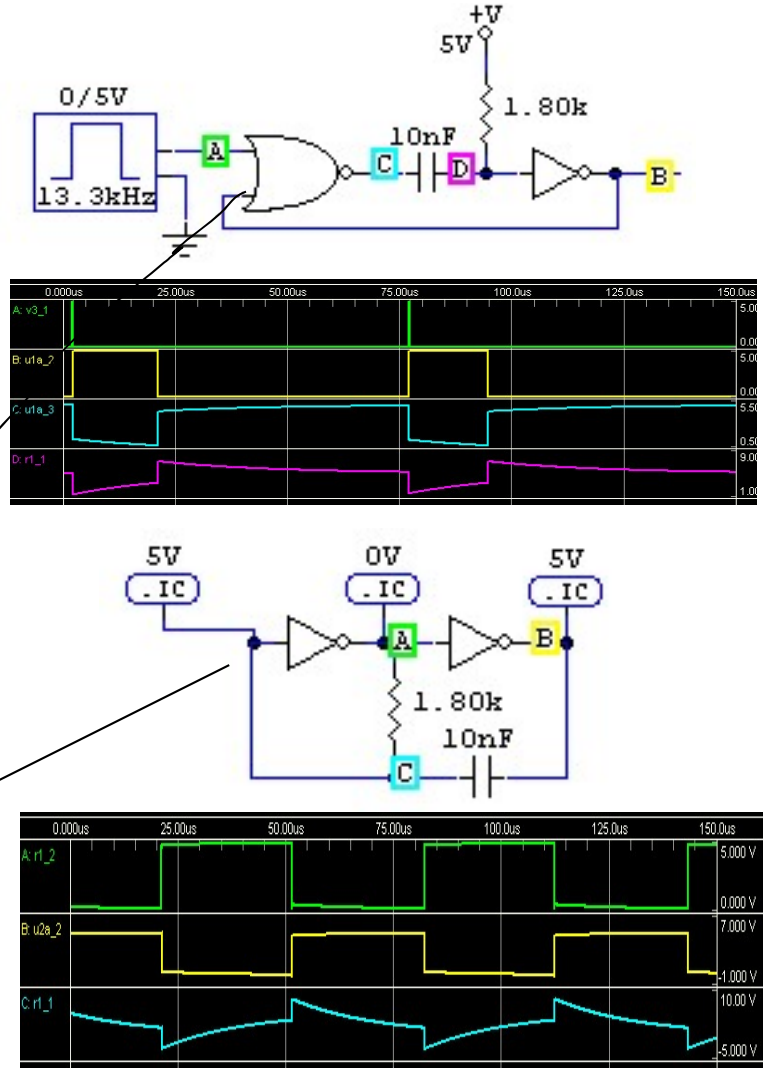
  - Circuitos monoestables: tienen un estado estable y un estado inestable. El circuito permanece en el estado estable hasta que es excitado, pasando al estado inestable del que al cabo de un tiempo T vuelve al estado estable.

$$T = RC \ln\left[\frac{Vdd}{Vdd - VTh}\right]$$

  - Circuitos astables: tiene dos estados inestables, el circuito pasa de un estado a otro cada cierto tiempo.

$$T = 2RC \ln\left[\frac{Vdd + VTh}{VTl}\right]$$

VTh (VTl) tensión umbral cuando sube (baja) la tensión de entrada en la VTC.
VTh = 3.3V VTl = 1.65V



Introducción a las Memorias

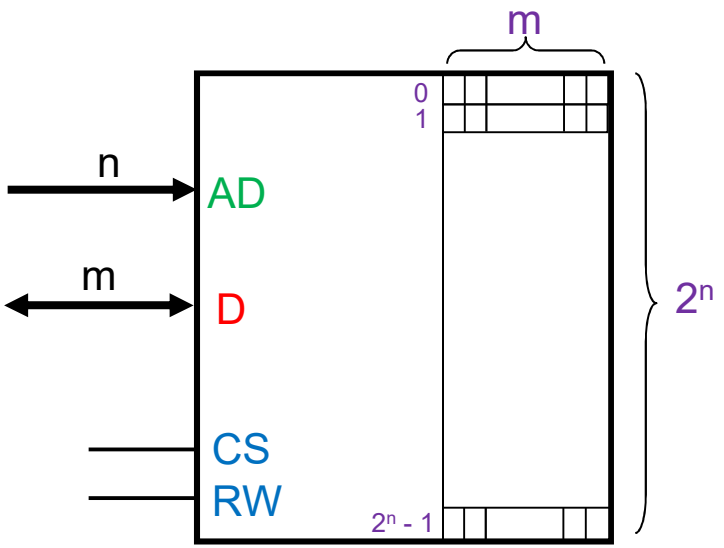
- Existen aplicaciones electrónicas que requieren almacenar gran cantidad de bits (por ejemplo la memoria de un ordenador Gbytes 10^9). En estas aplicaciones el uso de flip-flops como elementos de memoria es inviable ya que ocuparían demasiado espacio por lo que se usan memorias que utilizan celdas más pequeñas que comparten de alguna manera la lógica de control y de entrada-salida. Por contra son circuitos más lentos que los flip-flops.

La estructura básica de una memoria tiene:

- Un bus de direcciones AD de n bits. Se generan 2^n direcciones a través de un decodificador.

- Un bus de datos D de m bits de salida en memorias de solo lectura o de entrada/salida en memorias de lectura y escritura.

- 2^n * m celdas de memoria.



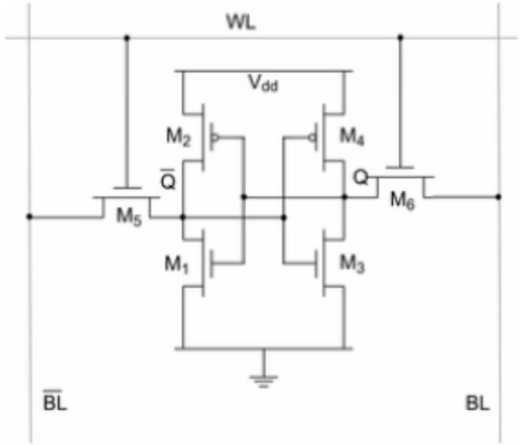
- Una lógica de control formada por una habilitador asíncrono, o un reloj de disparo, señales que indican la operación que se realiza (lectura, escritura, etc).

Introducción a las Memorias

- Existen muchos tipos de memoria, según las operaciones que realicen y según la lógica de control. En una primera clasificación por operaciones se tienen:

  - Memorias ROM (Read-Only Memories). Solo permite la operación de lectura (el bus de datos es de salida), la información se precarga en la memoria durante su fabricación (MROM) o en una fase de programación (PROM, EPROM, EEPROM). Este circuito es combinacional, ya que no tienen que almacenarse datos, están cargados por construcción.

  - Memorias RAM (Random-Access Memories). Permiten operaciones de lectura y de escritura (el bus de datos es de salida para leer y de entrada para escribir). Según el tipo de celdas que se usen se tiene memoria estática (SRAM) o dinámica (DRAM). Las celdas DRAM son más pequeñas pero pierden la información, por lo que la memoria necesita una operación de "refresco" cada cierto tiempo (y un circuito que controle dicha operación).



Celda SRAM

Celdas DRAM