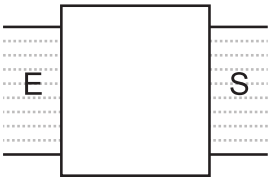
Circuitos Secuenciales

David Alejandro González Márquez

Clase disponible en: `https://github.com/fokerman/computingSystemsCourse`

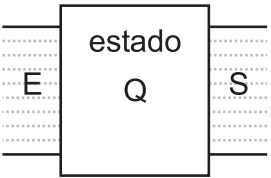
Introducción

Circuitos Combinacionales



La salida está determinada únicamente por la entrada del circuito

Circuitos Secuenciales

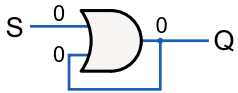


La salida está determinada por la entrada y el **estado interno** del circuito

El estado estará determinado por una memoria, pero **¿cómo almacenamos bits?**

¿Cómo almacenar un bit?

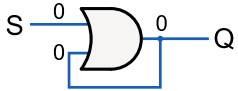
Proponemos el siguiente circuito:



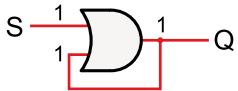
Supongamos inicialmente S=0 y Q=0.
El circuito se encuentra en reposo.

¿Cómo almacenar un bit?

Proponemos el siguiente circuito:



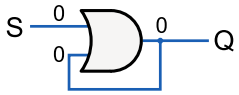
Supongamos inicialmente S=0 y Q=0.
El circuito se encuentra en reposo.



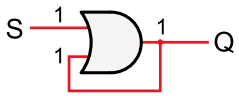
Si cambiamos el valor de S a 1,
luego que el circuito se **estabiliza**,
la salida valdrá Q=1.

¿Cómo almacenar un bit?

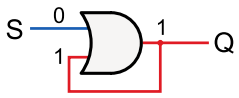
Proponemos el siguiente circuito:



Supongamos inicialmente S=0 y Q=0.
El circuito se encuentra en reposo.



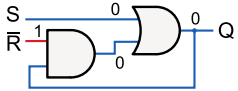
Si cambiamos el valor de S a 1,
luego que el circuito se **estabiliza**,
la salida valdrá Q=1.



Si ahora cambiamos S a 0,
el valor de la salida Q continuará en 1.

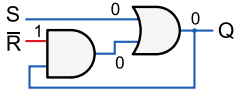
Construimos un circuito que **recuerda** un estado, pero **¿cómo hacemos para reiniciarlo?**

¿Cómo construir un *biestable*?

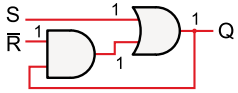


Supongamos inicialmente S=0, $\overline{R}$=1 y Q=0.
El circuito se encuentra en reposo.

¿Cómo construir un *biestable*?

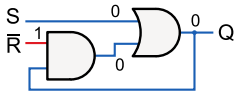


Supongamos inicialmente S=0, $\overline{R}$=1 y Q=0.
El circuito se encuentra en reposo.

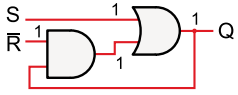


Si S=1 y $\overline{R}$=1 entonces la salida Q cambia a 1

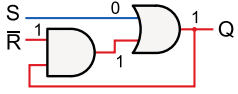
¿Cómo construir un *biestable*?



Supongamos inicialmente S=0, $\overline{R}$=1 y Q=0.
El circuito se encuentra en reposo.

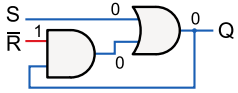


Si S=1 y $\overline{R}$=1 entonces la salida Q cambia a 1

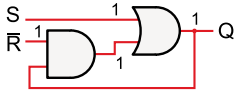


Si luego cambiamos S a 0, manteniendo $\overline{R}$=1,
la salida Q continúa en 1

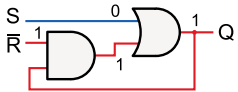
¿Cómo construir un *biestable*?



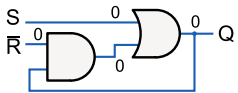
Supongamos inicialmente S=0, $\overline{R}$=1 y Q=0.
El circuito se encuentra en reposo.



Si S=1 y $\overline{R}$=1 entonces la salida Q cambia a 1



Si luego cambiamos S a 0, manteniendo $\overline{R}$=1,
la salida Q continúa en 1



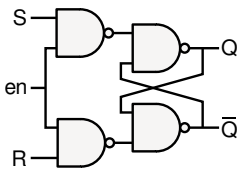
Ahora si cambiamos $\overline{R}$ por 0, manteniendo S=0,
entonces la salida Q cambia a 0

Construimos un circuito que puede **guardar** un estado y modificarlo. **Un biestable.**

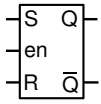
Flip-Flops (*biestable*)

Los `flip-flop` son circuitos *biestables* que permiten **almacenar 1 bit** de memoria.
Existen de diferentes tipos con distintas características.

Tipo R-S



en	S	R	Q_n	Q_{n+1}
1	0	0	Q_n	Q_n
1	0	1	Q_n	0
1	1	0	Q_n	1
1	1	1	Q_n	NO
0	×	×	Q_n	Q_n

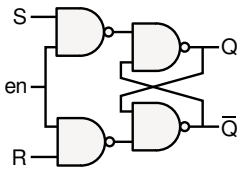


Señal S para setear a uno Q,
señal R para setear a cero Q y
señal en para impedir cambios.

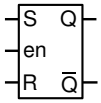
Flip-Flops (*biestable*)

Los `flip-flop` son circuitos *biestables* que permiten **almacenar 1 bit** de memoria.
Existen de diferentes tipos con distintas características.

Tipo R-S

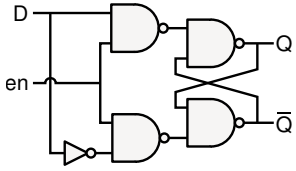


en	S	R	Q_n	Q_{n+1}
1	0	0	Q_n	Q_n
1	0	1	Q_n	0
1	1	0	Q_n	1
1	1	1	Q_n	NO
0	×	×	Q_n	Q_n

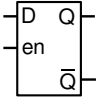


Señal S para setear a uno Q,
señal R para setear a cero Q y
señal en para impedir cambios.

Tipo D



en	D	Q_n	Q_{n+1}
1	0	Q_n	0
1	1	Q_n	1
0	×	Q_n	Q_n



Si la señal en está en 1,
el estado Q toma el valor
de la señal D.

Tabla Característica

Así como existen tablas de verdad para describir circuitos combinacionales.
Para los circuitos secuenciales tenemos **tablas características**.

Tabla Característica

Así como existen tablas de verdad para describir circuitos combinacionales.
Para los circuitos secuenciales tenemos **tablas características**.

Estas incluyen además de las entradas y salidas,
el **estado interno** del circuito y su **próximo estado interno**.

. . . Entradas . . .	Q_n . . .	Q_{n+1} . . .	. . . Salidas . . .
. . .	. . .	. . .	. . .

- Q_n: Representa un conjunto de bits de estados internos.
- Q_{n+1}: Representa el próximo estado de un conjunto de bits de estados internos.

Tabla Característica

Así como existen tablas de verdad para describir circuitos combinacionales.
Para los circuitos secuenciales tenemos **tablas características**.

Estas incluyen además de las entradas y salidas,
el **estado interno** del circuito y su **próximo estado interno**.

... Entradas ...	Q_n ...	Q_{n+1} ...	... Salidas ...
...	...	...	...

- Q_n: Representa un conjunto de bits de estados internos.
- Q_{n+1}: Representa el próximo estado de un conjunto de bits de estados internos.

Ejemplo:
Con 2 bits como estado interno y 1 bit de entrada, el circuito tendrá $2^2 \cdot 2^1 = 8$ combinaciones posibles de salidas. Es decir, 8 filas en su tabla característica.

Ejemplo: Tabla Característica

Supongamos un circuito con 2 bits de salida que almacena 4 estados posibles y los genera secuencialmente (Contador).

	$Q1_n$	$Q0_n$	$Q1_{n+1}$	$Q0_{n+1}$				
	0	0	0	1				
	0	1	1	0				
	1	0	1	1				
	1	1	0	0				

Ejemplo: Tabla Característica

Supongamos un circuito con 2 bits de salida que almacena 4 estados posibles y los genera secuencialmente (Contador).

Agregamos una entrada, que no afecta el estado interno del circuito.

E0	$Q1_n$	$Q0_n$	$Q1_{n+1}$	$Q0_{n+1}$				
0	0	0	0	1				
0	0	1	1	0				
0	1	0	1	1				
0	1	1	0	0				
1	0	0	0	1				
1	0	1	1	0				
1	1	0	1	1				
1	1	1	0	0				

Ejemplo: Tabla Característica

Supongamos un circuito con 2 bits de salida que almacena 4 estados posibles y los genera secuencialmente (Contador).

Agregamos una entrada, que no afecta el estado interno del circuito.

Y generamos 4 salidas, que representan el valor del contador extendido a 4 bits. Si E0 es 1, entonces las salidas son el inverso aditivo del número.

E0	$Q1_n$	$Q0_n$	$Q1_{n+1}$	$Q0_{n+1}$	S3	S2	S1	S0
0	0	0	0	1	0	0	0	1
0	0	1	1	0	0	0	1	0
0	1	0	1	1	0	0	1	1
0	1	1	0	0	0	0	0	0
1	0	0	0	1	1	1	1	1
1	0	1	1	0	1	1	1	0
1	1	0	1	1	1	1	0	1
1	1	1	0	0	0	0	0	0

Ejemplo: Tabla Característica

Supongamos un circuito con 2 bits de salida que almacena 4 estados posibles y los genera secuencialmente (Contador).

Agregamos una entrada, que no afecta el estado interno del circuito.

Y generamos 4 salidas, que representan el valor del contador extendido a 4 bits. Si E0 es 1, entonces las salidas son el inverso aditivo del número.

E0	$Q1_n$	$Q0_n$	$Q1_{n+1}$	$Q0_{n+1}$	S3	S2	S1	S0	
0	0	0	0	1	0	0	0	1	1
0	0	1	1	0	0	0	1	0	2
0	1	0	1	1	0	0	1	1	3
0	1	1	0	0	0	0	0	0	0
1	0	0	0	1	1	1	1	1	-1
1	0	1	1	0	1	1	1	0	-2
1	1	0	1	1	1	1	0	1	-3
1	1	1	0	0	0	0	0	0	0

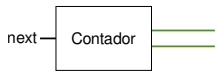
Ejemplo: Tabla Característica

Supongamos un circuito con 2 bits de salida que almacena 4 estados posibles y los genera secuencialmente (Contador).

Agregamos una entrada, que no afecta el estado interno del circuito.

Y generamos 4 salidas, que representan el valor del contador extendido a 4 bits. Si E0 es 1, entonces las salidas son el inverso aditivo del número.

E0	$Q1_n$	$Q0_n$	$Q1_{n+1}$	$Q0_{n+1}$	S3	S2	S1	S0	
0	0	0	0	1	0	0	0	1	1
0	0	1	1	0	0	0	1	0	2
0	1	0	1	1	0	0	1	1	3
0	1	1	0	0	0	0	0	0	0
1	0	0	0	1	1	1	1	1	-1
1	0	1	1	0	1	1	1	0	-2
1	1	0	1	1	1	1	0	1	-3
1	1	1	0	0	0	0	0	0	0



Ejemplo: Tabla Característica

Supongamos un circuito con 2 bits de salida que almacena 4 estados posibles y los genera secuencialmente (Contador).

Agregamos una entrada, que no afecta el estado interno del circuito.

Y generamos 4 salidas, que representan el valor del contador extendido a 4 bits. Si E0 es 1, entonces las salidas son el inverso aditivo del número.

E0	$Q1_n$	$Q0_n$	$Q1_{n+1}$	$Q0_{n+1}$	S3	S2	S1	S0	
0	0	0	0	1	0	0	0	1	1
0	0	1	1	0	0	0	1	0	2
0	1	0	1	1	0	0	1	1	3
0	1	1	0	0	0	0	0	0	0
1	0	0	0	1	1	1	1	1	-1
1	0	1	1	0	1	1	1	0	-2
1	1	0	1	1	1	1	0	1	-3
1	1	1	0	0	0	0	0	0	0

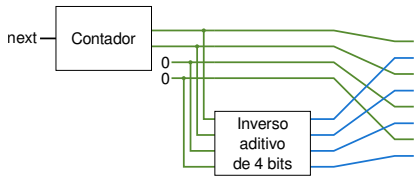
Ejemplo: Tabla Característica

Supongamos un circuito con 2 bits de salida que almacena 4 estados posibles y los genera secuencialmente (Contador).

Agregamos una entrada, que no afecta el estado interno del circuito.

Y generamos 4 salidas, que representan el valor del contador extendido a 4 bits. Si E0 es 1, entonces las salidas son el inverso aditivo del número.

E0	$Q1_n$	$Q0_n$	$Q1_{n+1}$	$Q0_{n+1}$	S3	S2	S1	S0	
0	0	0	0	1	0	0	0	1	1
0	0	1	1	0	0	0	1	0	2
0	1	0	1	1	0	0	1	1	3
0	1	1	0	0	0	0	0	0	0
1	0	0	0	1	1	1	1	1	-1
1	0	1	1	0	1	1	1	0	-2
1	1	0	1	1	1	1	0	1	-3
1	1	1	0	0	0	0	0	0	0



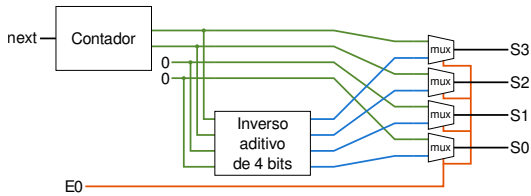
Ejemplo: Tabla Característica

Supongamos un circuito con 2 bits de salida que almacena 4 estados posibles y los genera secuencialmente (Contador).

Agregamos una entrada, que no afecta el estado interno del circuito.

Y generamos 4 salidas, que representan el valor del contador extendido a 4 bits. Si E0 es 1, entonces las salidas son el inverso aditivo del número.

E0	$Q1_n$	$Q0_n$	$Q1_{n+1}$	$Q0_{n+1}$	S3	S2	S1	S0	
0	0	0	0	1	0	0	0	1	1
0	0	1	1	0	0	0	1	0	2
0	1	0	1	1	0	0	1	1	3
0	1	1	0	0	0	0	0	0	0
1	0	0	0	1	1	1	1	1	-1
1	0	1	1	0	1	1	1	0	-2
1	1	0	1	1	1	1	0	1	-3
1	1	1	0	0	0	0	0	0	0



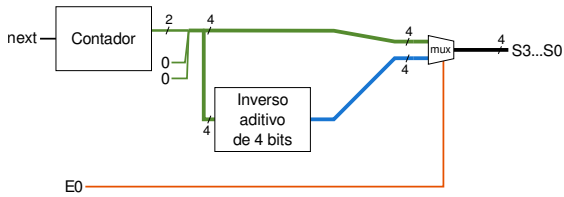
Ejemplo: Tabla Característica

Supongamos un circuito con 2 bits de salida que almacena 4 estados posibles y los genera secuencialmente (Contador).

Agregamos una entrada, que no afecta el estado interno del circuito.

Y generamos 4 salidas, que representan el valor del contador extendido a 4 bits. Si E0 es 1, entonces las salidas son el inverso aditivo del número.

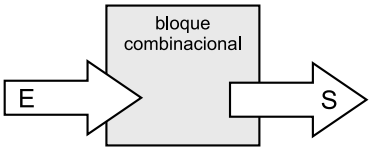
E0	$Q1_n$	$Q0_n$	$Q1_{n+1}$	$Q0_{n+1}$	S3	S2	S1	S0	
0	0	0	0	1	0	0	0	1	1
0	0	1	1	0	0	0	1	0	2
0	1	0	1	1	0	0	1	1	3
0	1	1	0	0	0	0	0	0	0
1	0	0	0	1	1	1	1	1	-1
1	0	1	1	0	1	1	1	0	-2
1	1	0	1	1	1	1	0	1	-3
1	1	1	0	0	0	0	0	0	0



Circuitos Secuenciales

Un circuito secuencial, se puede separar en dos partes:

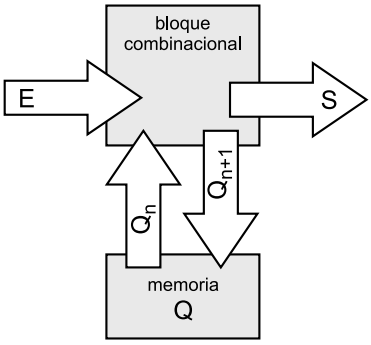
Circuitos Secuenciales



Un circuito secuencial, se puede separar en dos partes:

1. un *bloque combinacional*

Circuitos Secuenciales

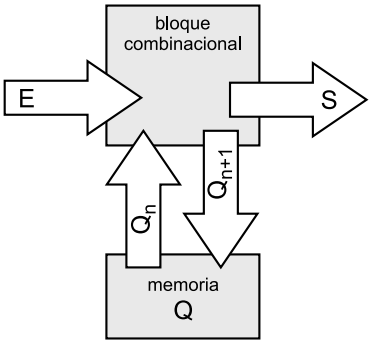


Un circuito secuencial, se puede separar en dos partes:

1. un *bloque combinacional*
2. un *bloque con memoria*

La memoria almacena bits que determinan el estado del circuito.

Circuitos Secuenciales



Un circuito secuencial, se puede separar en dos partes:

1. un *bloque combinacional*
2. un *bloque con memoria*

La memoria almacena bits que determinan el estado del circuito.

Las entradas del bloque combinacional son las entradas (E) y las salidas de la memoria (Q_n).

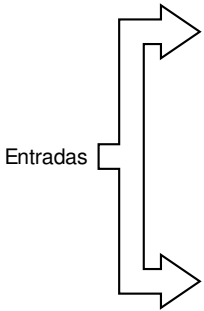
El bloque combinacional genera la salida del circuito (S) y el nuevo estado de la memoria (Q_{n+1}).

Circuitos Secuenciales

El esquema anterior lo podemos implementar usando `flip-flops` de la siguiente forma:

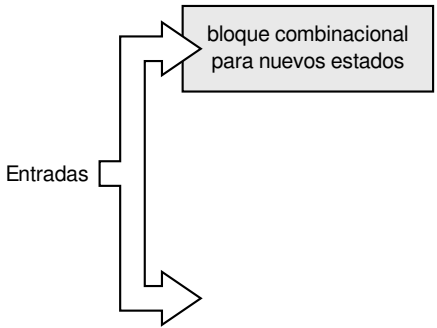
Circuitos Secuenciales

El esquema anterior lo podemos implementar usando `flip-flops` de la siguiente forma:



Circuitos Secuenciales

El esquema anterior lo podemos implementar usando `flip-flops` de la siguiente forma:

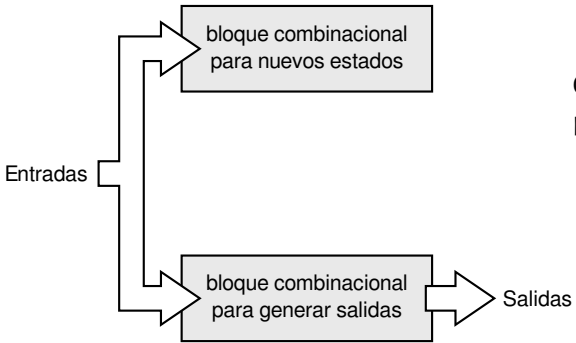


Contamos con dos bloques combinacionales para generar:

1. los nuevos estados estados de los flip-flops

Circuitos Secuenciales

El esquema anterior lo podemos implementar usando `flip-flops` de la siguiente forma:

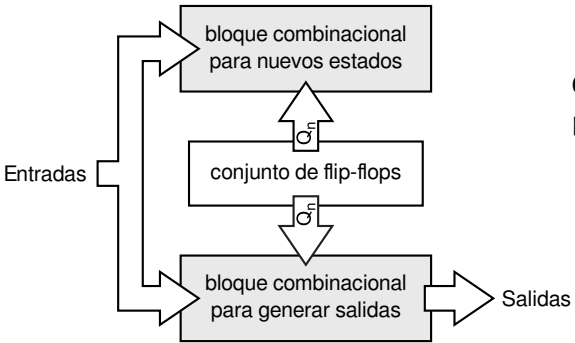


Contamos con dos bloques combinacionales para generar:

1. los nuevos estados estados de los flip-flops
2. y otro para generar las salidas del circuito

Circuitos Secuenciales

El esquema anterior lo podemos implementar usando `flip-flops` de la siguiente forma:



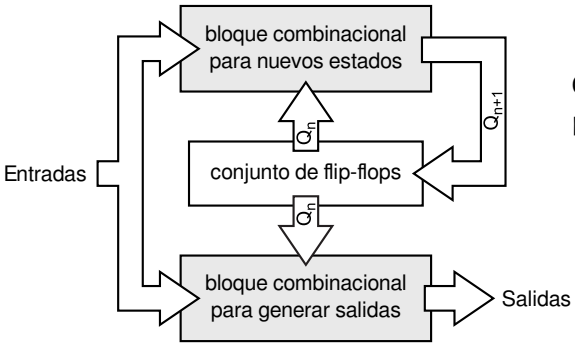
Contamos con dos bloques combinacionales para generar:

1. los nuevos estados estados de los flip-flops
2. y otro para generar las salidas del circuito

Q_n representa el estado actual de los flip-flops,

Circuitos Secuenciales

El esquema anterior lo podemos implementar usando `flip-flops` de la siguiente forma:



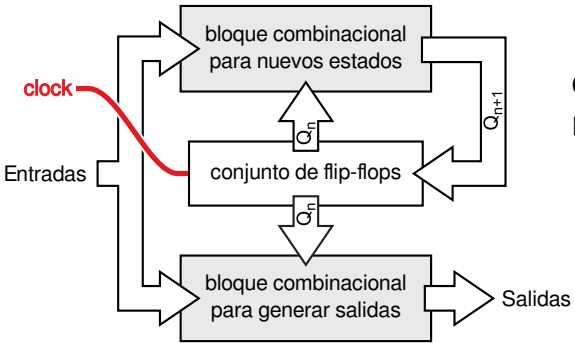
Contamos con dos bloques combinacionales para generar:

1. los nuevos estados estados de los flip-flops
2. y otro para generar las salidas del circuito

Q_n representa el estado actual de los flip-flops, mientras que Q_{n+1} representa el estado siguiente.

Circuitos Secuenciales

El esquema anterior lo podemos implementar usando `flip-flops` de la siguiente forma:



Contamos con dos bloques combinacionales para generar:

1. los nuevos estados estados de los flip-flops
2. y otro para generar las salidas del circuito

Q_n representa el estado actual de los flip-flops, mientras que Q_{n+1} representa el estado siguiente.

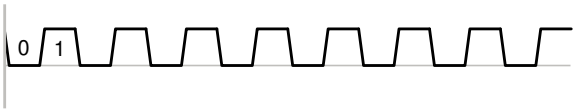
Pero, **¿cuándo se modifica el estado interno de un circuito?**

Reloj (Clock)

El reloj o señal de reloj, se utiliza para **temporizar** los cambios que se suceden en un circuito. Permite coordinar el momento exacto en que se **modifica** el estado de los biestables.

Reloj (Clock)

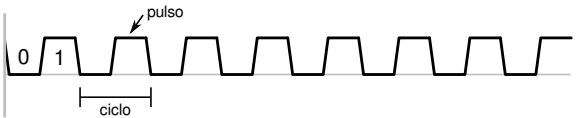
El reloj o señal de reloj, se utiliza para **temporizar** los cambios que se suceden en un circuito. Permite coordinar el momento exacto en que se **modifica** el estado de los biestables.



La señal varía entre 0 y 1 en intervalos regulares de tiempo.

Reloj (Clock)

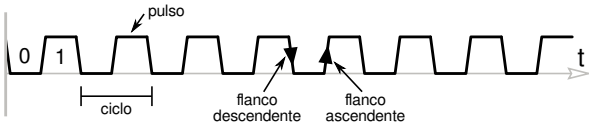
El reloj o señal de reloj, se utiliza para **temporizar** los cambios que se suceden en un circuito.
Permite coordinar el momento exacto en que se **modifica** el estado de los biestables.



La señal varía entre 0 y 1 en intervalos regulares de tiempo.
Su ciclo es simétrico, ya sea comenzando desde 0 o desde 1.

Reloj (Clock)

El reloj o señal de reloj, se utiliza para **temporizar** los cambios que se suceden en un circuito.
Permite coordinar el momento exacto en que se **modifica** el estado de los biestables.



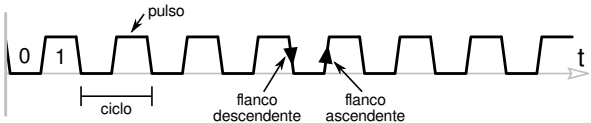
La señal varía entre 0 y 1 en intervalos regulares de tiempo.
Su ciclo es simétrico, ya sea comenzando desde 0 o desde 1.

Se denomina **flanco**, al momento en que la señal cambia entre estados.

Flanco descendente: Cambia entre 1 a 0.
Flanco ascendente: Cambia entre 0 a 1.

Reloj (Clock)

El reloj o señal de reloj, se utiliza para **temporizar** los cambios que se suceden en un circuito.
Permite coordinar el momento exacto en que se **modifica** el estado de los biestables.



La señal varía entre 0 y 1 en intervalos regulares de tiempo.
Su ciclo es simétrico, ya sea comenzando desde 0 o desde 1.

Se denomina **flanco**, al momento en que la señal cambia entre estados.

Flanco descendente: Cambia entre 1 a 0.
Flanco ascendente: Cambia entre 0 a 1.

Ahora, **¿cómo hacemos que un flip-flop cambie su estado por flanco?**

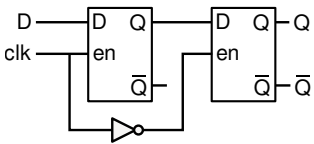
`Flip-Flops` (*biestable* Master-Slave)

Utilizamos dos `flip-flops`, uno como `master` y otro `slave`.
El `master` toma un valor, y el `slave` lo copia en el momento del **cambio de flanco**.

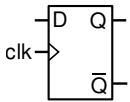
Flip-Flops (*biestable* Master-Slave)

Utilizamos dos `flip-flops`, uno como `master` y otro `slave`.
El `master` toma un valor, y el `slave` lo copia en el momento del **cambio de flanco**.

Tipo D Master-Slave



en	D	Q_n	Q_{n+1} ↓
1	0	Q_n	0
1	1	Q_n	1
0	×	Q_n	Q_n

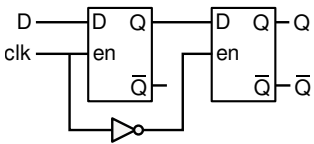


Si la señal `clk` cambia de 1 a 0 (flanco descendente), Q toma el valor del último valor guardado durante el puso 1 de la señal `clk`

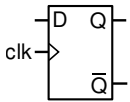
Flip-Flops (*biestable* Master-Slave)

Utilizamos dos `flip-flops`, uno como `master` y otro `slave`.
El `master` toma un valor, y el `slave` lo copia en el momento del **cambio de flanco**.

Tipo D Master-Slave



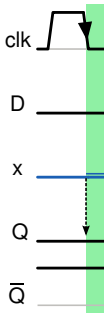
en	D	Q_n	Q_{n+1} ↓
1	0	Q_n	0
1	1	Q_n	1
0	×	Q_n	Q_n



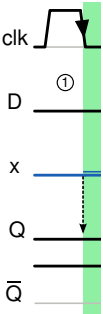
Si la señal `clk` cambia de 1 a 0 (flanco descendente), Q toma el valor del último valor guardado durante el puso 1 de la señal `clk`

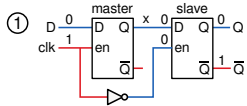
En el contexto de la materia solo vamos a utilizar `flip-flops` de tipo **D Master-Slave**, ya sea por flaco positivo como negativo.

Funcionamiento Flip-Flop D Master-Slave



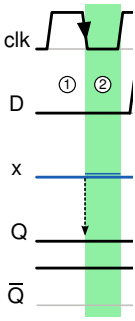
Funcionamiento Flip-Flop D Master-Slave

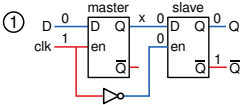


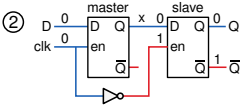


1. El circuito inicia en cero. El clock cambia a 1.

Funcionamiento Flip-Flop D Master-Slave

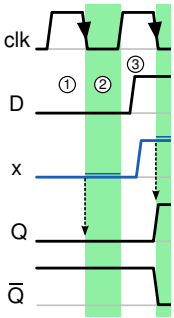


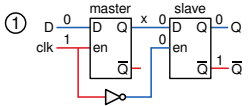


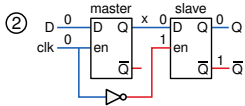


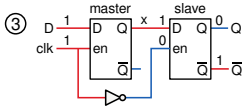
1. El circuito inicia en cero. El clock cambia a 1.
2. Luego del flanco descendente el estado de Q permanece en 0.

Funcionamiento Flip-Flop D Master-Slave



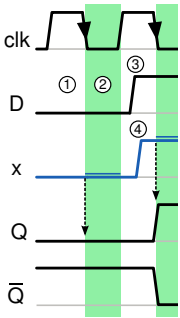


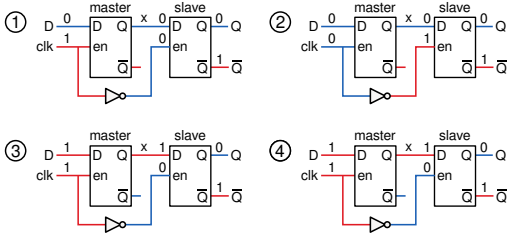




1. El circuito inicia en cero. El clock cambia a 1.
2. Luego del flanco descendente el estado de Q permanece en 0.
3. La entrada D cambia a 1.

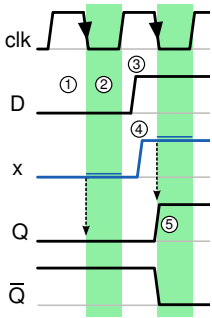
Funcionamiento Flip-Flop D Master-Slave

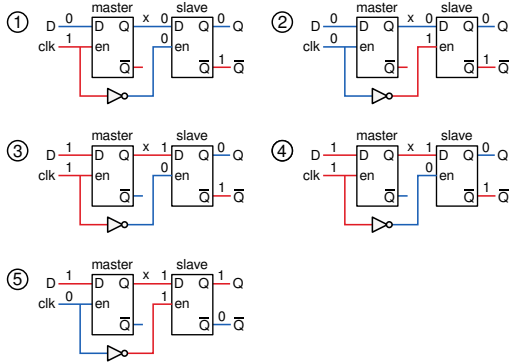




1. El circuito inicia en cero. El clock cambia a 1.
2. Luego del flanco descendente el estado de Q permanece en 0.
3. La entrada D cambia a 1.
4. Como el clock está en 1, el flip-flop master toma el 1.

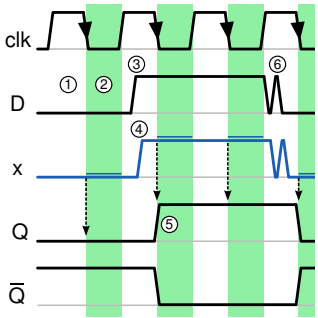
Funcionamiento Flip-Flop D Master-Slave

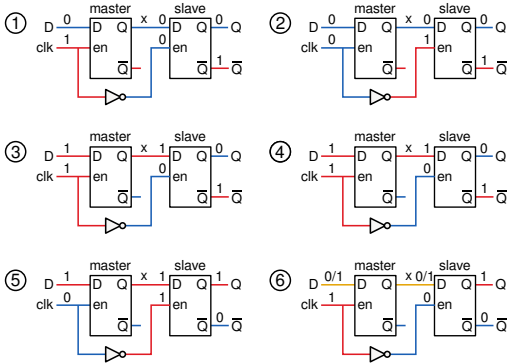




1. El circuito inicia en cero. El clock cambia a 1.
2. Luego del flanco descendente el estado de Q permanece en 0.
3. La entrada D cambia a 1.
4. Como el clock está en 1, el flip-flop master toma el 1.
5. Cuando se produce el flanco descendente, cambia el clock y se copia el estado entre master y slave.

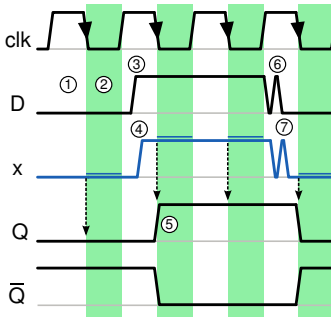
Funcionamiento Flip-Flop D Master-Slave

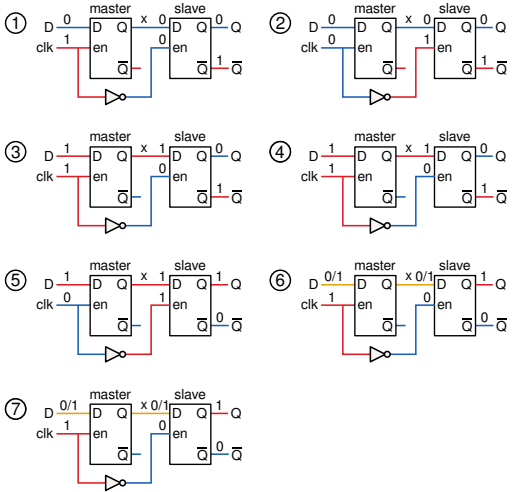




1. El circuito inicia en cero. El clock cambia a 1.
2. Luego del flanco descendente el estado de Q permanece en 0.
3. La entrada D cambia a 1.
4. Como el clock está en 1, el flip-flop master toma el 1.
5. Cuando se produce el flanco descendente, cambia el clock y se copia el estado entre master y slave.
6. Los cambios de D durante la parte alta del ciclo no se reflejan en la salida Q.

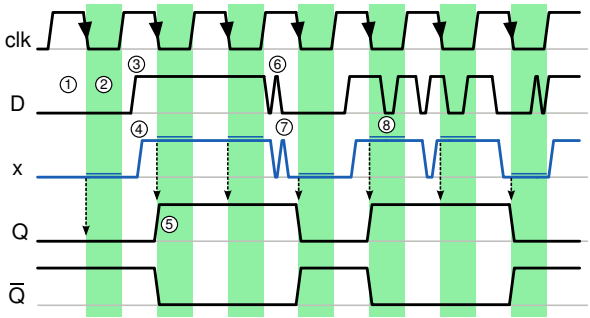
Funcionamiento Flip-Flop D Master-Slave

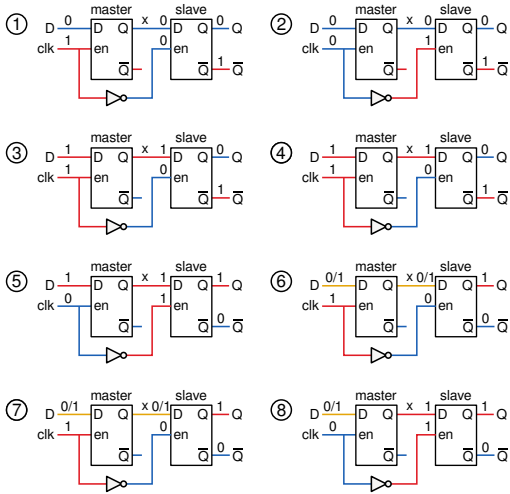




1. El circuito inicia en cero. El clock cambia a 1.
2. Luego del flanco descendente el estado de Q permanece en 0.
3. La entrada D cambia a 1.
4. Como el clock está en 1, el flip-flop master toma el 1.
5. Cuando se produce el flanco descendente, cambia el clock y se copia el estado entre master y slave.
6. Los cambios de D durante la parte alta del ciclo no se reflejan en la salida Q.
7. Pero sí son tomados por el flip-flop master.

Funcionamiento Flip-Flop D Master-Slave





1. El circuito inicia en cero. El clock cambia a 1.
2. Luego del flanco descendente el estado de Q permanece en 0.
3. La entrada D cambia a 1.
4. Como el clock está en 1, el flip-flop master toma el 1.
5. Cuando se produce el flanco descendente, cambia el clock y se copia el estado entre master y slave.
6. Los cambios de D durante la parte alta del ciclo no se reflejan en la salida Q.
7. Pero sí son tomados por el flip-flop master.
8. Los cambios de D durante la parte baja del ciclo no son tomados por el flip-flop master.

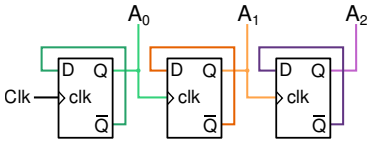
Contadores

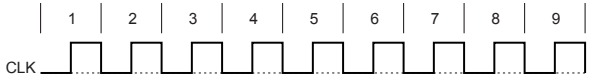
Los circuitos contadores generan una **secuencia** de valores en su salida por cada ciclo de clock.

Contadores

Los circuitos contadores generan una **secuencia** de valores en su salida por cada ciclo de clock.

Ejemplo: Contador módulo potencia de 2 de 3 bits. Cuenta de `000`b a 111b.

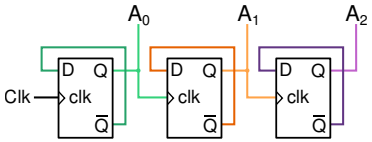


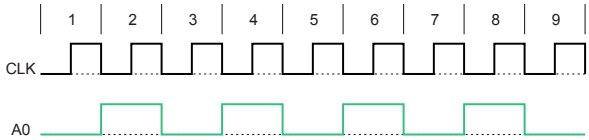


Contadores

Los circuitos contadores generan una **secuencia** de valores en su salida por cada ciclo de clock.

Ejemplo: Contador módulo potencia de 2 de 3 bits. Cuenta de `000`b a 111b.

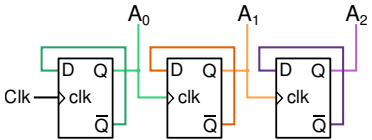


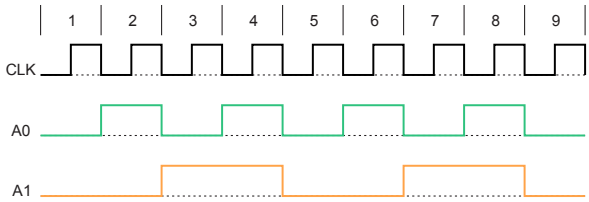


Contadores

Los circuitos contadores generan una **secuencia** de valores en su salida por cada ciclo de clock.

Ejemplo: Contador módulo potencia de 2 de 3 bits. Cuenta de `000`b a 111b.

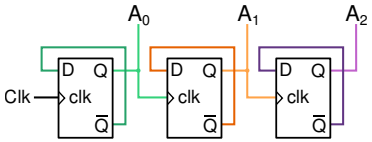


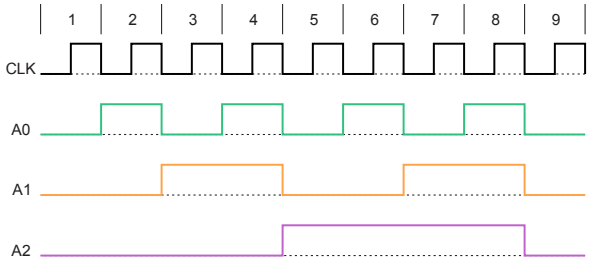


Contadores

Los circuitos contadores generan una **secuencia** de valores en su salida por cada ciclo de clock.

Ejemplo: Contador módulo potencia de 2 de 3 bits. Cuenta de `000`b a 111b.

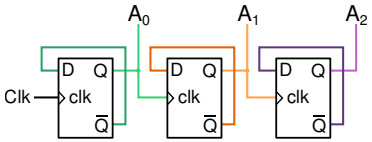


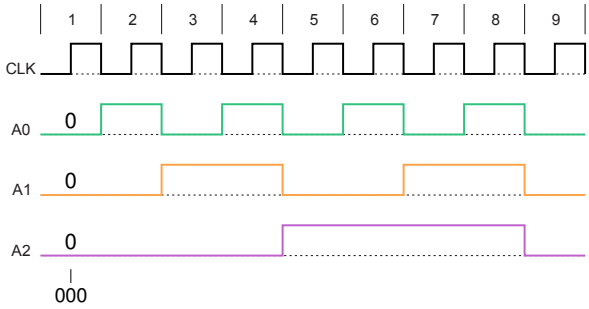


Contadores

Los circuitos contadores generan una **secuencia** de valores en su salida por cada ciclo de clock.

Ejemplo: Contador módulo potencia de 2 de 3 bits. Cuenta de `000`b a 111b.

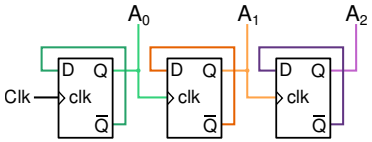


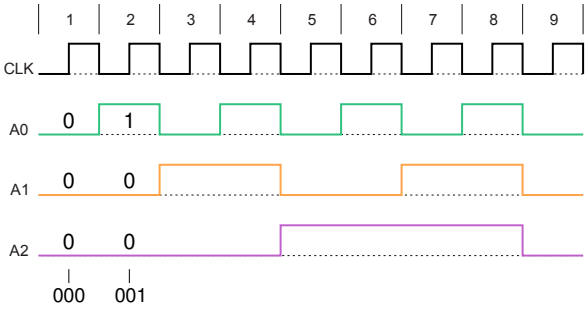


Contadores

Los circuitos contadores generan una **secuencia** de valores en su salida por cada ciclo de clock.

Ejemplo: Contador módulo potencia de 2 de 3 bits. Cuenta de `000b` a `111b`.

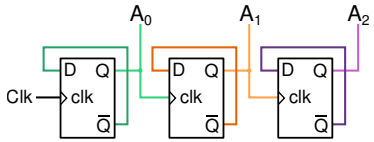




Contadores

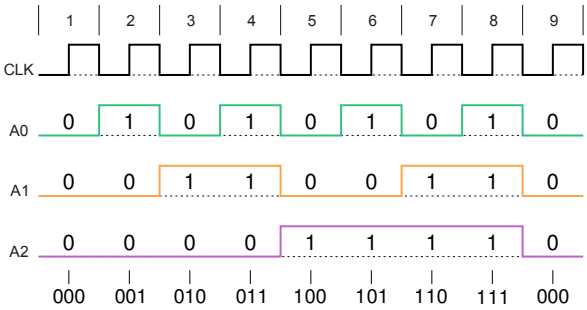
Los circuitos contadores generan una **secuencia** de valores en su salida por cada ciclo de clock.

Ejemplo: Contador módulo potencia de 2 de 3 bits. Cuenta de `000`b a 111b.



Cada uno de los flip-flops cambia de estado por el contrario al que tiene almacenado.

El cambio se produce cuando el flip-flop anterior, que alimenta su clock, cambia de estado.

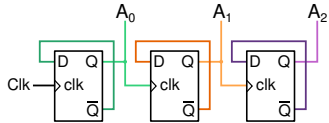


Contadores

Contador módulo potencia de 2

Secuencia ascendente tal que su módulo siempre es una potencia de 2 por cada bit que se agrega al contador.

Ejemplos:



Contadores

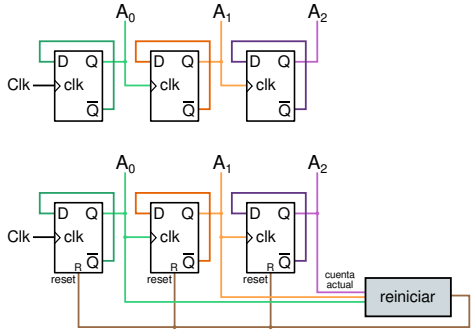
Contador módulo potencia de 2

Secuencia ascendente tal que su módulo siempre es una potencia de 2 por cada bit que se agrega al contador.

Contador módulo arbitario

Cuando la secuencia supera valor máximo es reiniciada a cero gracias a un circuito combinatorio provisto para tal fin.

Ejemplos:



Contadores

Contador módulo potencia de 2

Secuencia ascendente tal que su módulo siempre es una potencia de 2 por cada bit que se agrega al contador.

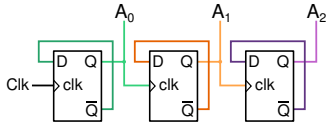
Contador módulo arbitario

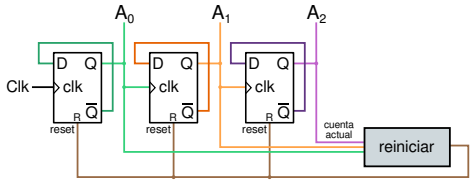
Cuando la secuencia supera valor máximo es reiniciada a cero gracias a un circuito combinatorio provisto para tal fin.

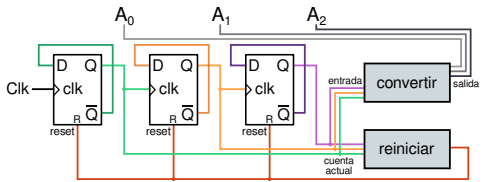
Contador módulo y cuenta arbitraria

Se provee un circuito combinatorio que transforma cada valor de la secuencia en uno arbitrario.

Ejemplos:







Registros

Un registro es un circuito secuencial que contiene un conjunto de *n* flip-flops asociados, que permiten **almacenar temporariamente** una palabra o grupo de *n* bits.

Registros

Un registro es un circuito secuencial que contiene un conjunto de *n* flip-flops asociados, que permiten **almacenar temporariamente** una palabra o grupo de *n* bits.

Los tipos de registro dependen de la forma en que los datos son **leídos** o **almacenados**:

1. Registro paralelo-paralelo (entrada paralelo, salida paralelo)
2. Registro serie-paralelo (entrada serie, salida paralelo)
3. Registro paralelo-serie (entrada paralelo, salida serie)
4. Registro serie-serie (entrada serie, salida serie)

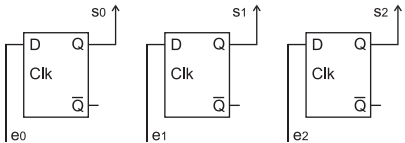
Serie: Los bits entran o salen del registro secuencialmente, uno a continuación del otro.

Paralelo: Los bits entran o salen del registro simultáneamente, todos al mismo tiempo.

Registros

Ejemplos para registros de 3 bits.

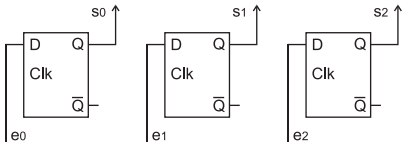
Registro paralelo-paralelo



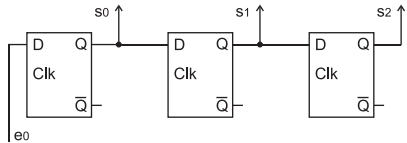
Registros

Ejemplos para registros de 3 bits.

Registro paralelo-paralelo



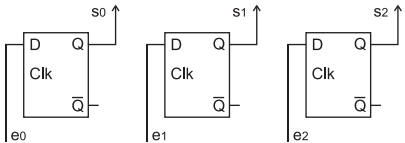
Registro serie-paralelo



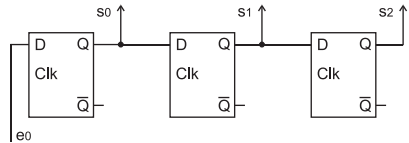
Registros

Ejemplos para registros de 3 bits.

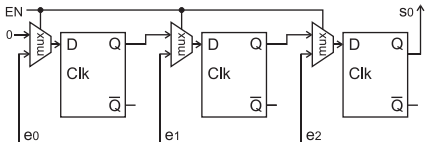
Registro paralelo-paralelo



Registro serie-paralelo



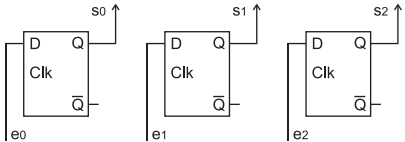
Registro paralelo-serie



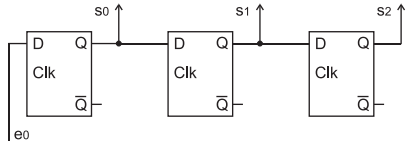
Registros

Ejemplos para registros de 3 bits.

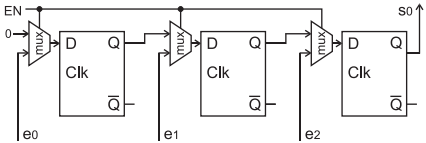
Registro paralelo-paralelo



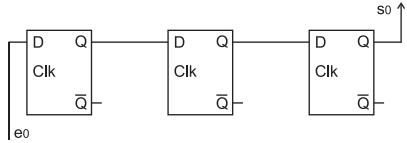
Registro serie-paralelo



Registro paralelo-serie



Registro serie-serie



Registros bidireccionales

En términos prácticos, vamos a querer **conectar registros entre sí**, tanto para leer como para escribir.

Para esto vamos a necesitar registros que operen **bidireccionalmente**.
Es decir que, utilizando las mismas entradas, podamos escribir y leer el registro.

Usando **componentes de 3 estados**, vamos a construir registros bidireccionales.
Esta idea se puede extender a cualquier circuito, no necesariamente registros.

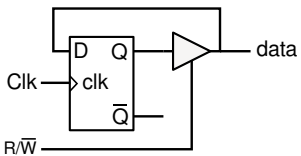
Registros bidireccionales

En términos prácticos, vamos a querer **conectar registros entre sí**, tanto para leer como para escribir.

Para esto vamos a necesitar registros que operen **bidireccionalmente**.
Es decir que, utilizando las mismas entradas, podamos escribir y leer el registro.

Usando **componentes de 3 estados**, vamos a construir registros bidireccionales.
Esta idea se puede extender a cualquier circuito, no necesariamente registros.

Ejemplo: registro de 1 bit.



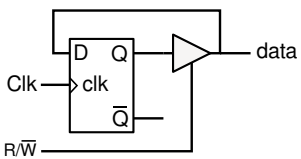
Registros bidireccionales

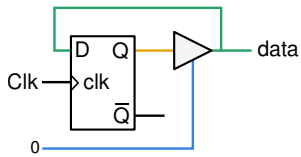
En términos prácticos, vamos a querer **conectar registros entre sí**, tanto para leer como para escribir.

Para esto vamos a necesitar registros que operen **bidireccionalmente**.
Es decir que, utilizando las mismas entradas, podamos escribir y leer el registro.

Usando **componentes de 3 estados**, vamos a construir registros bidireccionales.
Esta idea se puede extender a cualquier circuito, no necesariamente registros.

Ejemplo: registro de 1 bit.





Si la señal R/$\overline{W}$ es 0, el valor en la **entrada** data puede llegar a la entrada D del biestable.

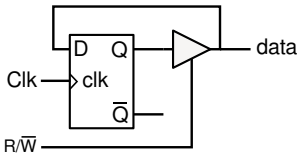
Registros bidireccionales

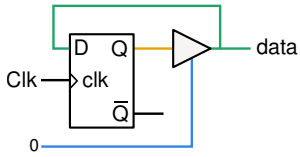
En términos prácticos, vamos a querer **conectar registros entre sí**, tanto para leer como para escribir.

Para esto vamos a necesitar registros que operen **bidireccionalmente**.
Es decir que, utilizando las mismas entradas, podamos escribir y leer el registro.

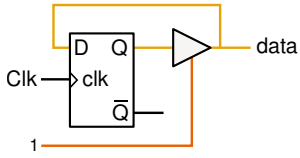
Usando **componentes de 3 estados**, vamos a construir registros bidireccionales.
Esta idea se puede extender a cualquier circuito, no necesariamente registros.

Ejemplo: registro de 1 bit.





Si la señal R/$\overline{W}$ es 0, el valor en la **entrada** data puede llegar a la entrada D del biestable.



Si la señal R/$\overline{W}$ es 1, el estado Q del biestable se expone en la **salida** data.

Bibliografía

- Tanenbaum, “Organización de Computadoras. Un Enfoque Estructurado”, 4ta Edición, 2000.
  - **Capítulo 3 - El nivel de lógica digital** - Páginas 141-146

- Null, “Essentials of Computer Organization and Architecture”, 5th Edition, 2018.
  - **Chapter 3 - Boolean Algebra and Digital Logic:**
    - 3.7 - Sequential Circuits
      3.7.1 - Basic Concepts
      3.7.2 - Clocks
      3.7.3 - Flip-Flops

Ejercicios

Con lo visto, ya pueden resolver todos los ejercicios de la Guía de Lógica Digital.

¡Gracias!

Recuerden leer los comentarios adjuntos
en cada clase por aclaraciones.