Lógica Digital - Circuitos Secuenciales

Primer Cuatrimestre 2025

Sistemas Digitales
DC - UBA

Introducción

Sobre la clase de hoy

Hoy vamos a ver los principios de diseño, práctica y ejemplos de circuitos secuenciales, la estructura de la clase va ser la siguiente:

- **Repaso de circuitos combinatorios**
- **Retroalimentación y cambio de modelo**
- **Circuitos secuenciales asincrónicos**
- **Circuitos secuenciales sincrónicos**
- **Latchs - Flip-flops, registros y memorias**

Sobre la clase de hoy

Hoy vamos a ver los principios de diseño, práctica y ejemplos de circuitos secuenciales, la estructura de la clase va ser la siguiente:

- **Repaso de circuitos combinatorios**
- **Retroalimentación y cambio de modelo**
- **Circuitos secuenciales asincrónicos**
- **Circuitos secuenciales sincrónicos**
- **Latchs - Flip-flops, registros y memorias**

Latchs - Flip-flops

Latchs

Son circuitos que permiten *trabar o asegurar* el valor de su salida

Latchs

Son circuitos que permiten *trabar o asegurar* el valor de su salida

- Permiten el cambio de sus salidas según el **nivel** de las entradas.

Latchs

Son circuitos que permiten *trabar o asegurar* el valor de su salida

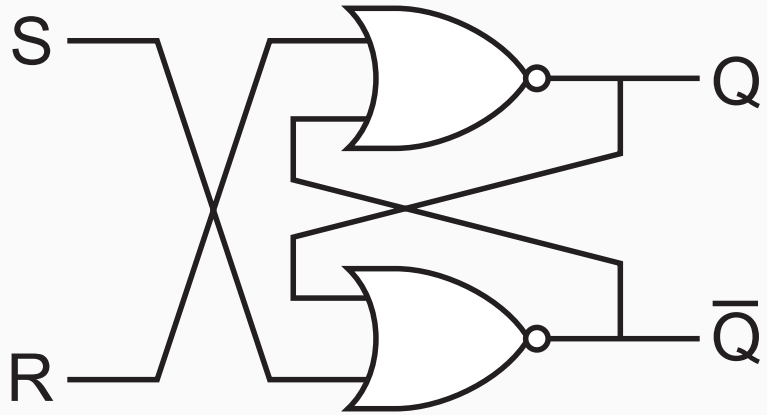
- Permiten el cambio de sus salidas según el **nivel** de las entradas.
- Utilizan **realimentación**

Latchs

Son circuitos que permiten *trabar o asegurar* el valor de su salida

- Permiten el cambio de sus salidas según el **nivel** de las entradas.
- Utilizan **realimentación**

Ejemplo:



Latch RS (Reset-Set)

Analicemos el ejemplo anterior:

Latch RS implementado con NOR:

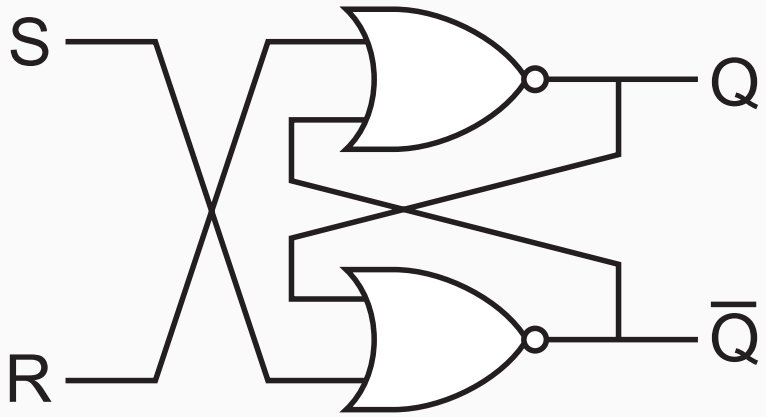


Tabla de verdad:

S	R	Q	$\overline{Q}$
1	0		
0	1		
0	0		
1	1		

Latch RS (Reset-Set)

Analicemos el ejemplo anterior:

Latch RS implementado con NOR:

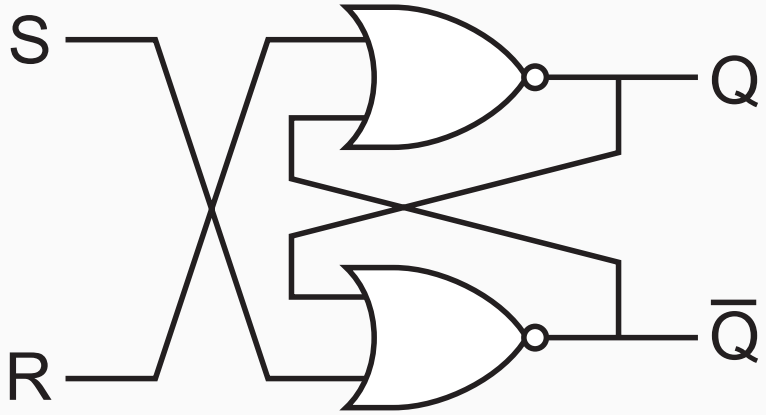


Tabla de verdad:

S	R	Q	$\overline{Q}$
1	0		0
0	1		
0	0		
1	1		

Latch RS (Reset-Set)

Analicemos el ejemplo anterior:

Latch RS implementado con NOR:

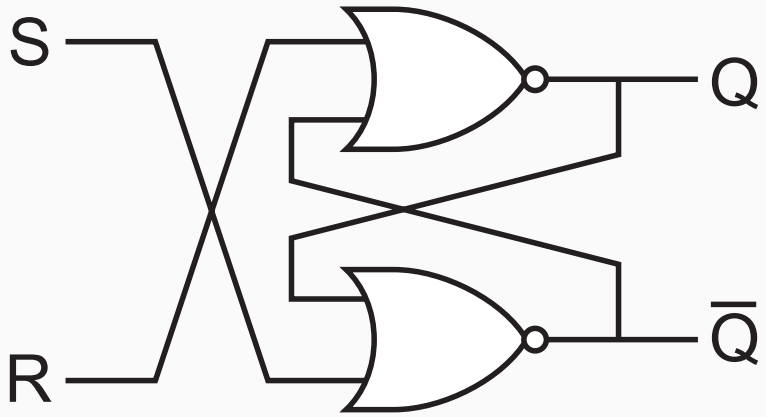


Tabla de verdad:

S	R	Q	$\overline{Q}$
1	0	1	0
0	1		
0	0		
1	1		

Latch RS (Reset-Set)

Analicemos el ejemplo anterior:

Latch RS implementado con NOR:

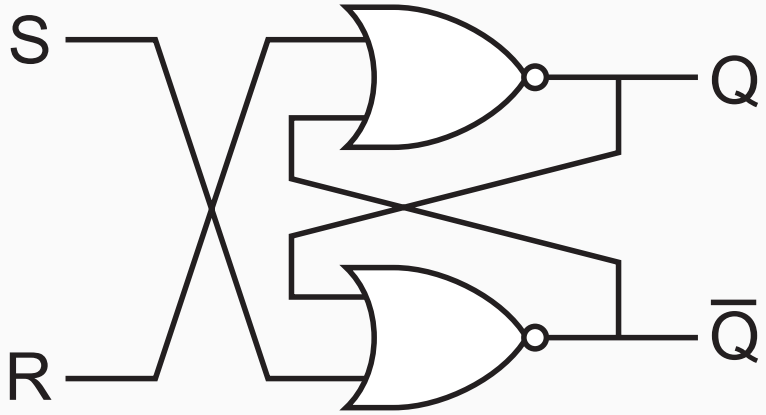


Tabla de verdad:

S	R	Q	$\overline{Q}$
1	0	1	0
0	1	0	1
0	0		
1	1		

Latch RS (Reset-Set)

Analicemos el ejemplo anterior:

Latch RS implementado con NOR:

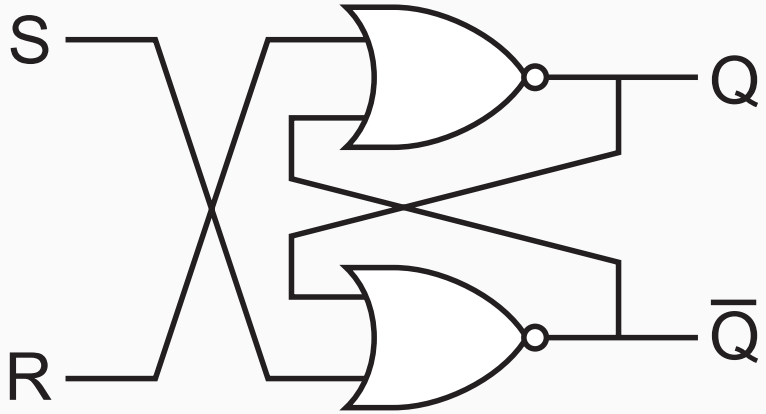


Tabla de verdad:

S	R	Q	$\overline{Q}$
1	0	1	0
0	1	0	1
0	0	$Q*$	$\overline{Q}*$ [1]
1	1		

[1] $Q*$ o $\overline{Q}*$ refiere al *estado* anterior de la salida

Latch RS (Reset-Set)

Analicemos el ejemplo anterior:

Latch RS implementado con NOR:

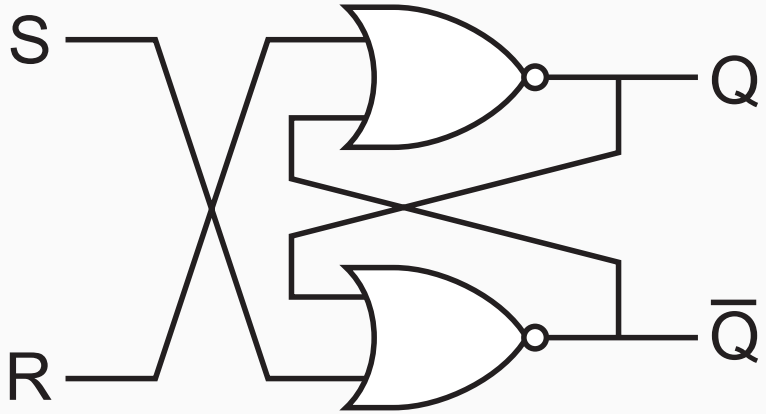


Tabla de verdad:

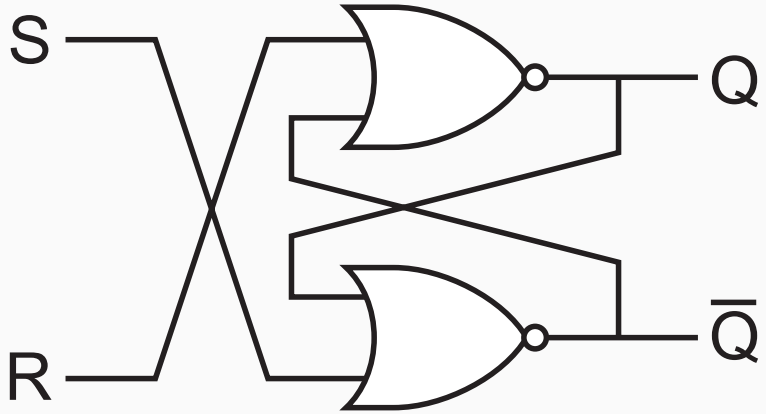
S	R	Q	$\overline{Q}$
1	0	1	0
0	1	0	1
0	0	$Q*$	$\overline{Q}*$ [1]
1	1	0	0

[1] $Q*$ o $\overline{Q}*$ refiere al *estado* anterior de la salida

Latch RS (Reset-Set)

Analicemos el ejemplo anterior:

Latch RS implementado con NOR:



Con $S, R = (1, 1)$:

Tabla de verdad:

S	R	Q	$\overline{Q}$
1	0	1	0
0	1	0	1
0	0	$Q*$	$\overline{Q}*$ [1]
1	1	0	0

[1] $Q*$ o $\overline{Q}*$ refiere al *estado* anterior de la salida

Latch RS (Reset-Set)

Analicemos el ejemplo anterior:

Latch RS implementado con NOR:

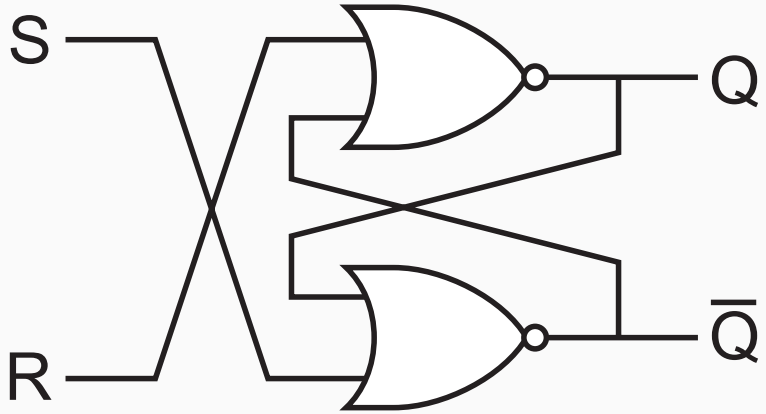


Tabla de verdad:

S	R	Q	$\overline{Q}$
1	0	1	0
0	1	0	1
0	0	$Q*$	$\overline{Q}*$ [1]
1	1	0	0

Con $S, R = (1, 1)$:

- El valor de las salidas es inconsistente con la especificación

[1] $Q*$ o $\overline{Q}*$ refiere al *estado* anterior de la salida

Latch RS (Reset-Set)

Analicemos el ejemplo anterior:

Latch RS implementado con NOR:

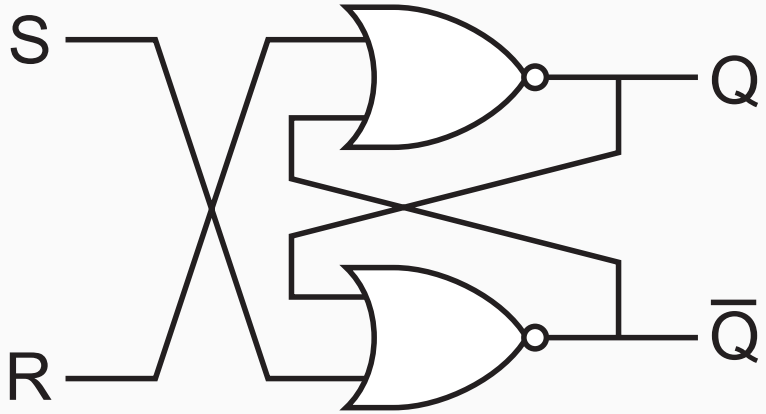


Tabla de verdad:

S	R	Q	$\overline{Q}$
1	0	1	0
0	1	0	1
0	0	$Q*$	$\overline{Q}*$ [1]
1	1	0	0

Con $S, R = (1, 1)$:

- El valor de las salidas es inconsistente con la especificación
- El valor de las salidas depende de la implementación. **Tarea:** implementar con NANDs

[1] $Q*$ o $\overline{Q}*$ refiere al *estado* anterior de la salida

Latch JK

Tratemos de modificar el comportamiento para el caso cuando las entradas son $(1, 1)$:

Latch JK:

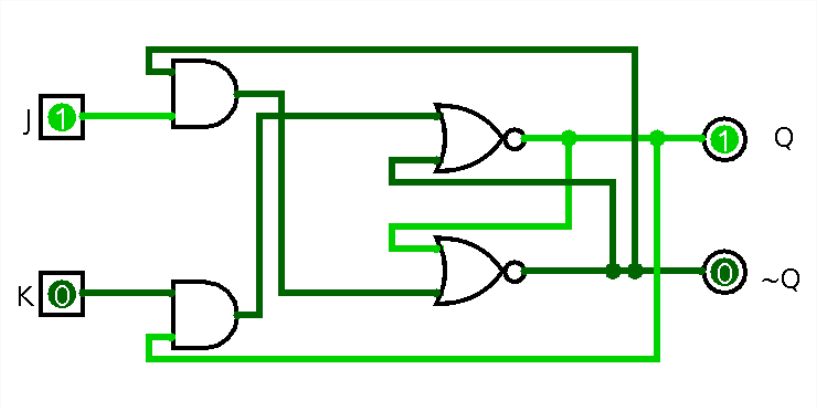


Tabla de verdad:

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

Latch JK

Tratemos de modificar el comportamiento para el caso cuando las entradas son $(1, 1)$:

Latch JK:

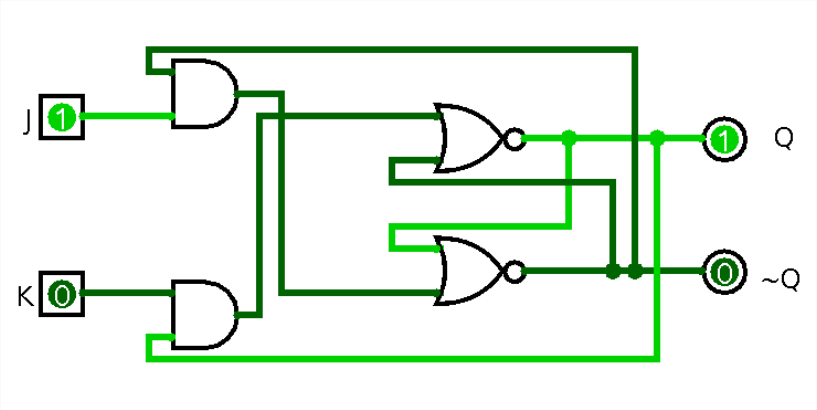


Tabla de verdad:

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

Latch JK

Tratemos de modificar el comportamiento para el caso cuando las entradas son $(1, 1)$:

Latch JK:

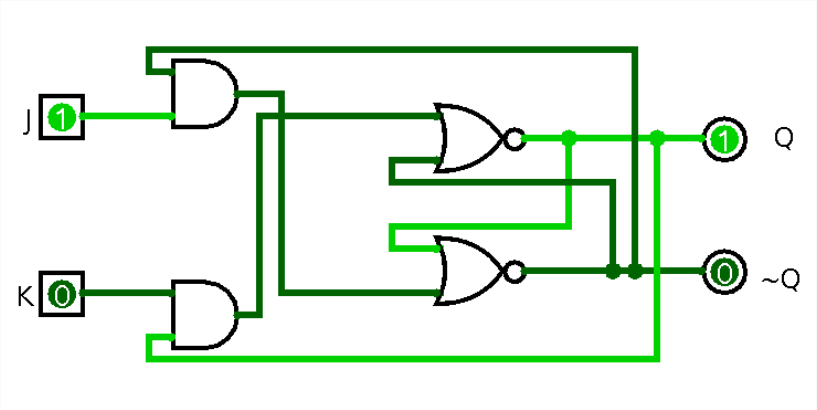


Tabla de verdad:

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

Latch JK

Tratemos de modificar el comportamiento para el caso cuando las entradas son $(1, 1)$:

Latch JK:

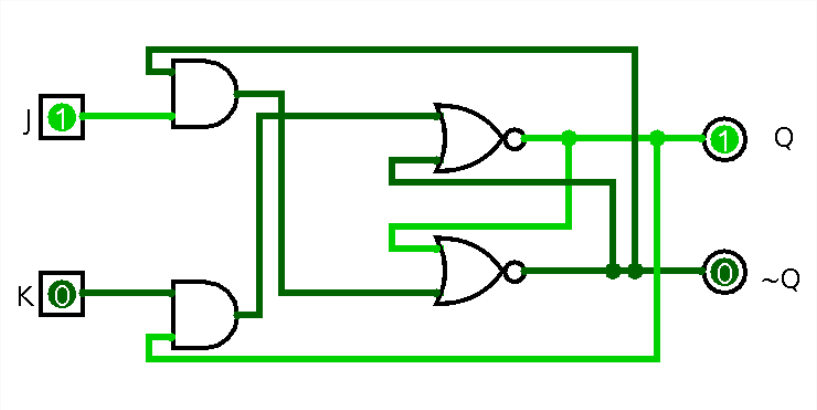


Tabla de verdad:

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

Latch JK

Tratemos de modificar el comportamiento para el caso cuando las entradas son $(1, 1)$:

Latch JK:

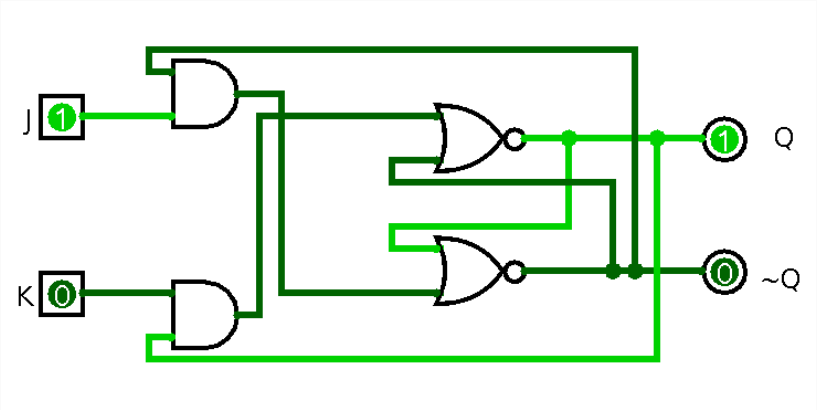


Tabla de verdad:

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

Latch JK

Tratemos de modificar el comportamiento para el caso cuando las entradas son $(1, 1)$:

Latch JK:

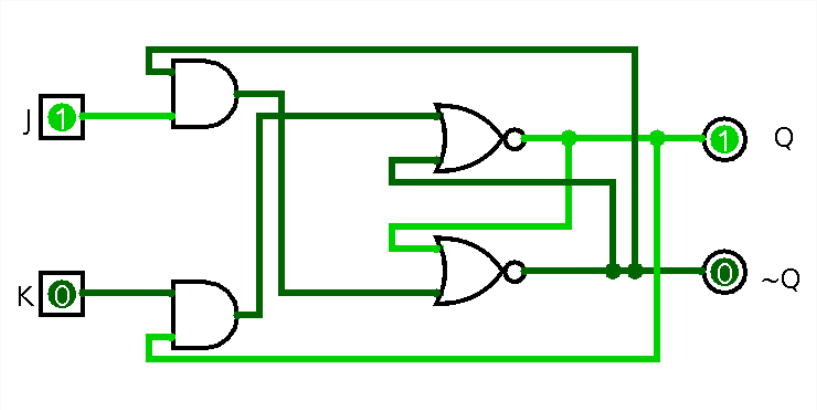


Tabla de verdad:

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

Latch JK

Tratemos de modificar el comportamiento para el caso cuando las entradas son $(1, 1)$:

Latch JK:

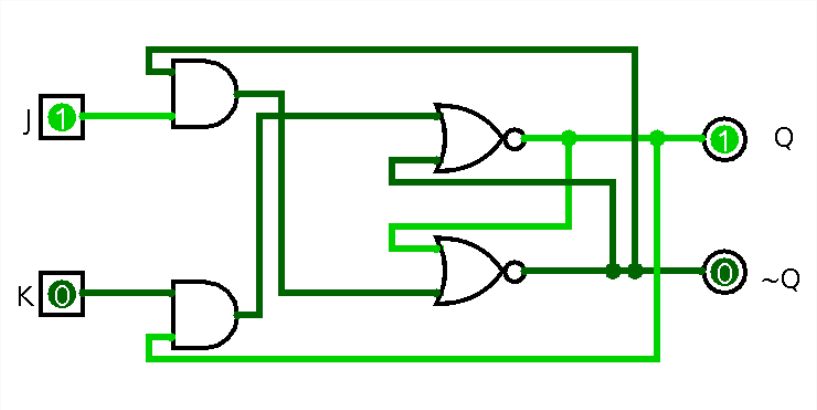


Tabla de verdad:

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

Con $S, R = (1, 1)$:

- El valor de las salidas está ahora definido.

Latch JK

Tratemos de modificar el comportamiento para el caso cuando las entradas son $(1, 1)$:

Latch JK:

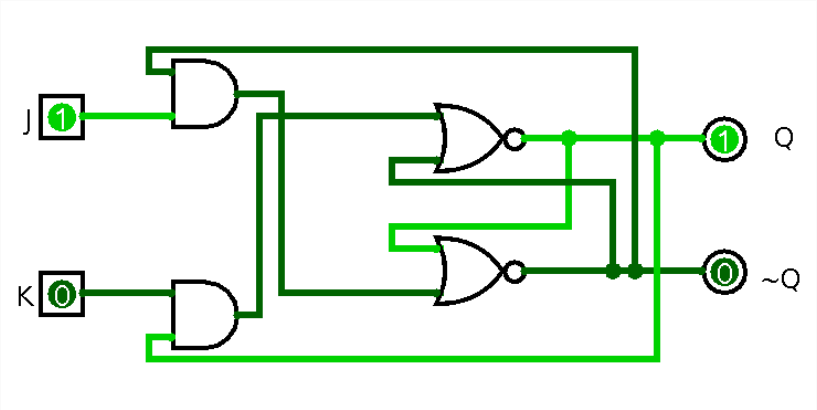


Tabla de verdad:

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

Con $S, R = (1, 1)$:

- El valor de las salidas está ahora definido.
- El circuito oscila (estado inestable).

Latch D

- Nos permite almacenar 1 bit
- Tiene una entrada de **datos** y una de **control**

Latch D:

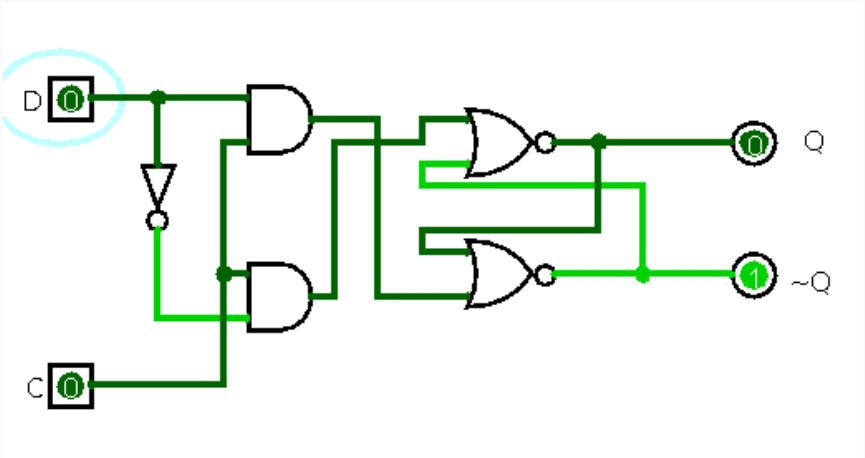


Tabla de verdad:

D	C	Q	$\overline{Q}$
1	0		
0	1		
0	0		
1	1		

Latch D

- Nos permite almacenar 1 bit
- Tiene una entrada de **datos** y una de **control**

Latch D:

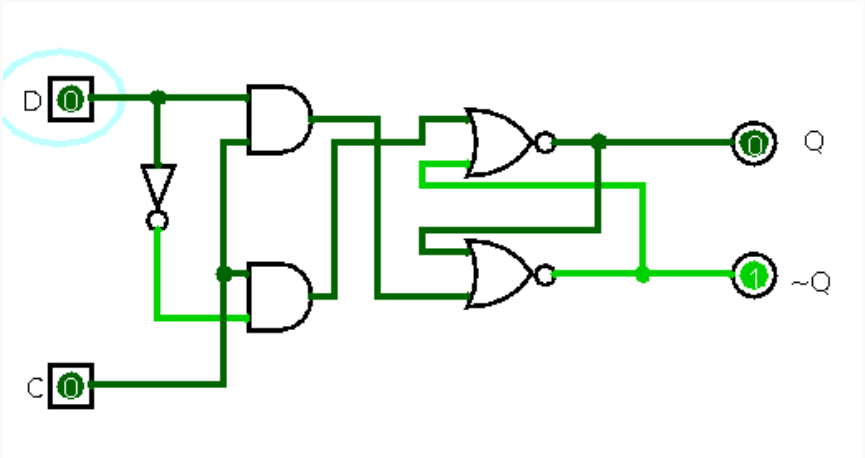


Tabla de verdad:

D	C	Q	$\overline{Q}$
1	0		$\overline{Q}*$
0	1		
0	0		
1	1		

Latch D

- Nos permite almacenar 1 bit
- Tiene una entrada de **datos** y una de **control**

Latch D:

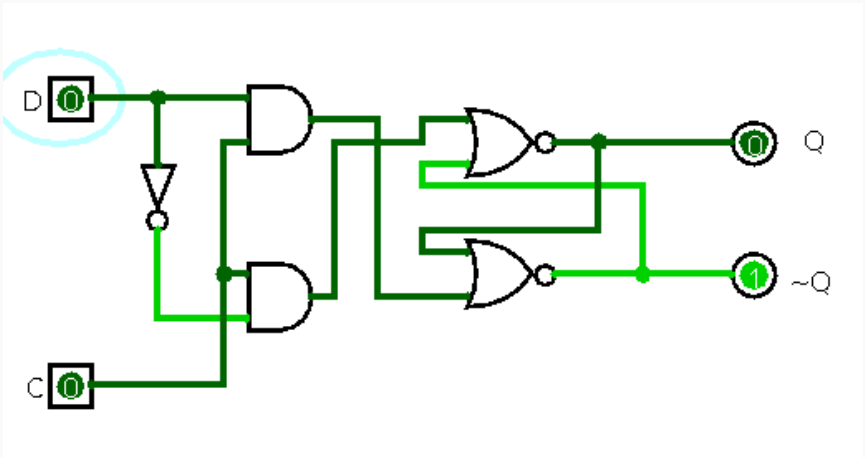


Tabla de verdad:

D	C	Q	$\overline{Q}$
1	0	$Q*$	$\overline{Q}*$
0	1		
0	0		
1	1		

Latch D

- Nos permite almacenar 1 bit
- Tiene una entrada de **datos** y una de **control**

Latch D:

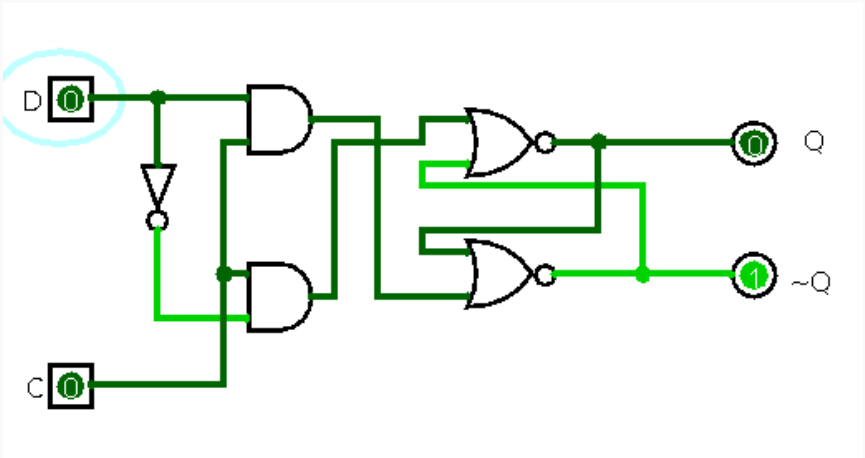


Tabla de verdad:

D	C	Q	$\overline{Q}$
1	0	$Q*$	$\overline{Q}*$
0	1	0	1
0	0		
1	1		

Latch D

- Nos permite almacenar 1 bit
- Tiene una entrada de **datos** y una de **control**

Latch D:

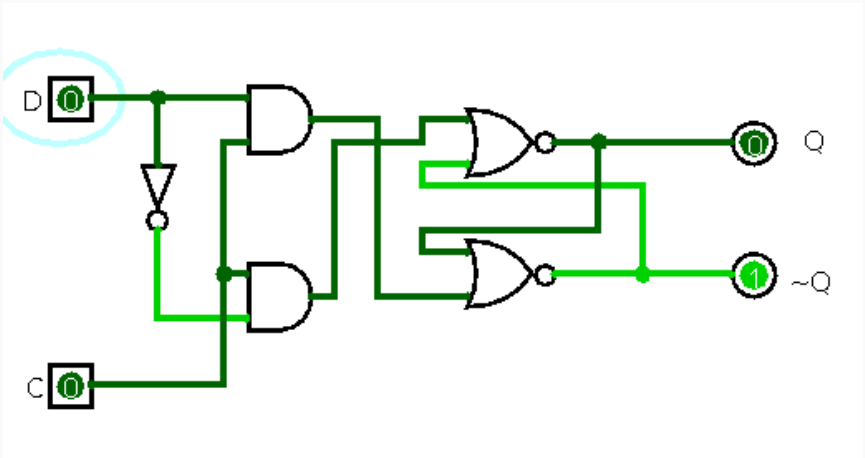


Tabla de verdad:

D	C	Q	$\overline{Q}$
1	0	$Q*$	$\overline{Q}*$
0	1	0	1
0	0	$Q*$	$\overline{Q}*$
1	1		

Latch D

- Nos permite almacenar 1 bit
- Tiene una entrada de **datos** y una de **control**

Latch D:

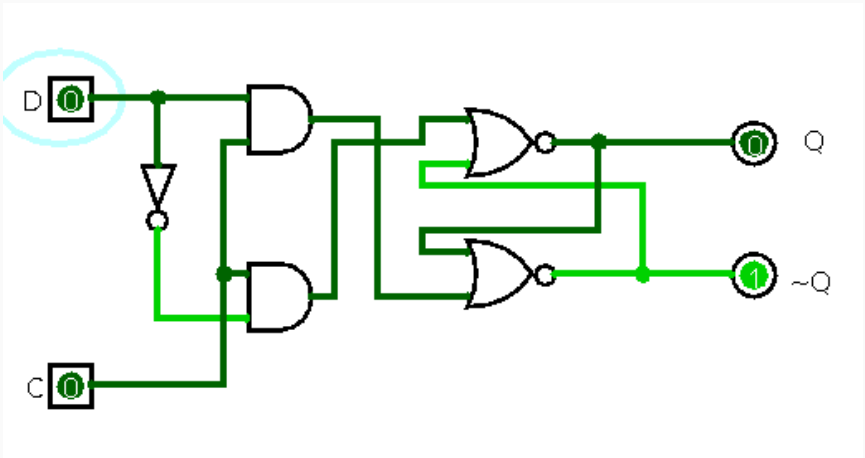


Tabla de verdad:

D	C	Q	$\overline{Q}$
1	0	$Q*$	$\overline{Q}*$
0	1	0	1
0	0	$Q*$	$\overline{Q}*$
1	1	1	0

Latch D

- Nos permite almacenar 1 bit
- Tiene una entrada de **datos** y una de **control**

Latch D:

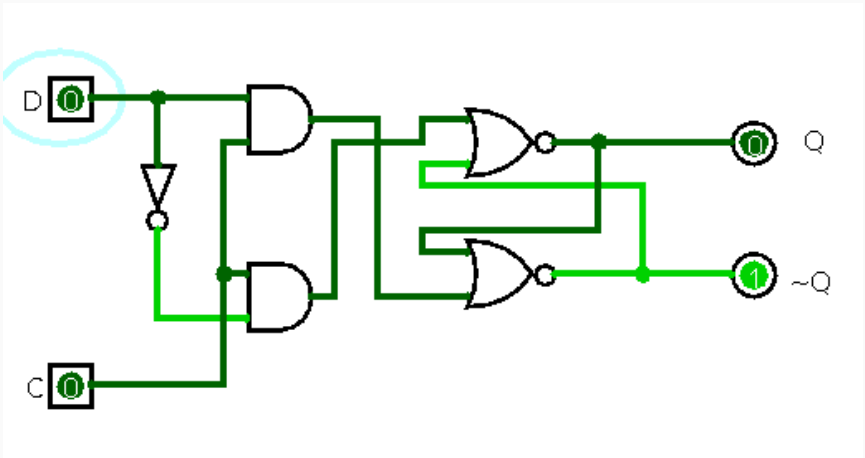


Tabla de verdad:

D	C	Q	$\overline{Q}$
1	0	$Q*$	$\overline{Q}*$
0	1	0	1
0	0	$Q*$	$\overline{Q}*$
1	1	1	0

En este caso el circuito es estable en todos los estados. Sin embargo:

Latch D

- Nos permite almacenar 1 bit
- Tiene una entrada de **datos** y una de **control**

Latch D:

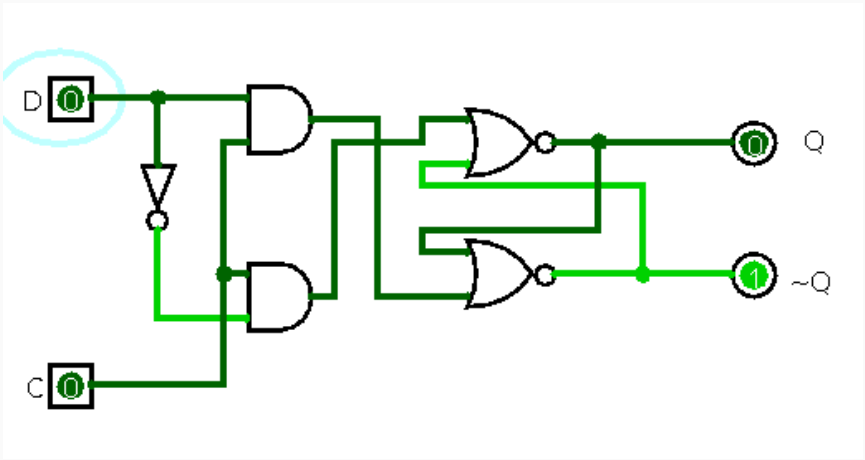


Tabla de verdad:

D	C	Q	$\overline{Q}$
1	0	$Q*$	$\overline{Q}*$
0	1	0	1
0	0	$Q*$	$\overline{Q}*$
1	1	1	0

En este caso el circuito es estable en todos los estados. Sin embargo:

- Los tiempos no se pueden predecir (dependen de D)

Latch D

- Nos permite almacenar 1 bit
- Tiene una entrada de **datos** y una de **control**

Latch D:

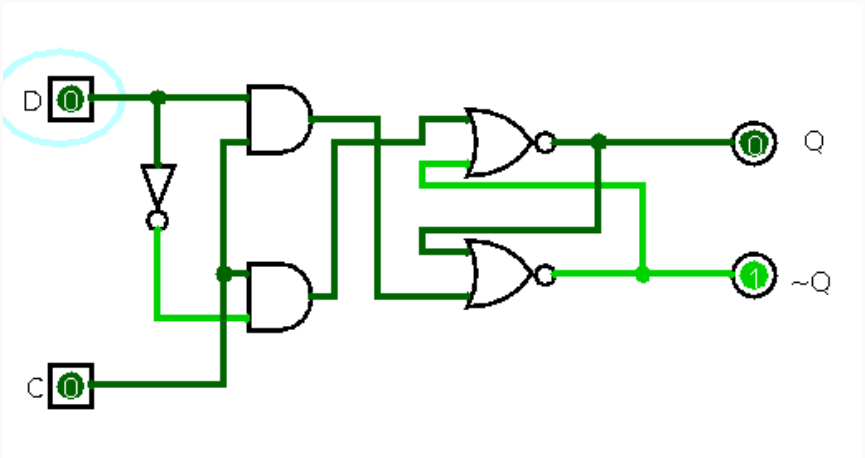


Tabla de verdad:

D	C	Q	$\overline{Q}$
1	0	$Q*$	$\overline{Q}*$
0	1	0	1
0	0	$Q*$	$\overline{Q}*$
1	1	1	0

En este caso el circuito es estable en todos los estados. Sin embargo:

- Los tiempos no se pueden predecir (dependen de D)
- Puede causar carreras si existe un lazo en el circuito externo.

Sincronizando...

Como vimos en la primer parte, nos interesa poder tener un control de los momentos de transición de estados $\Rightarrow$ **CLOCK**

Sincronizando...

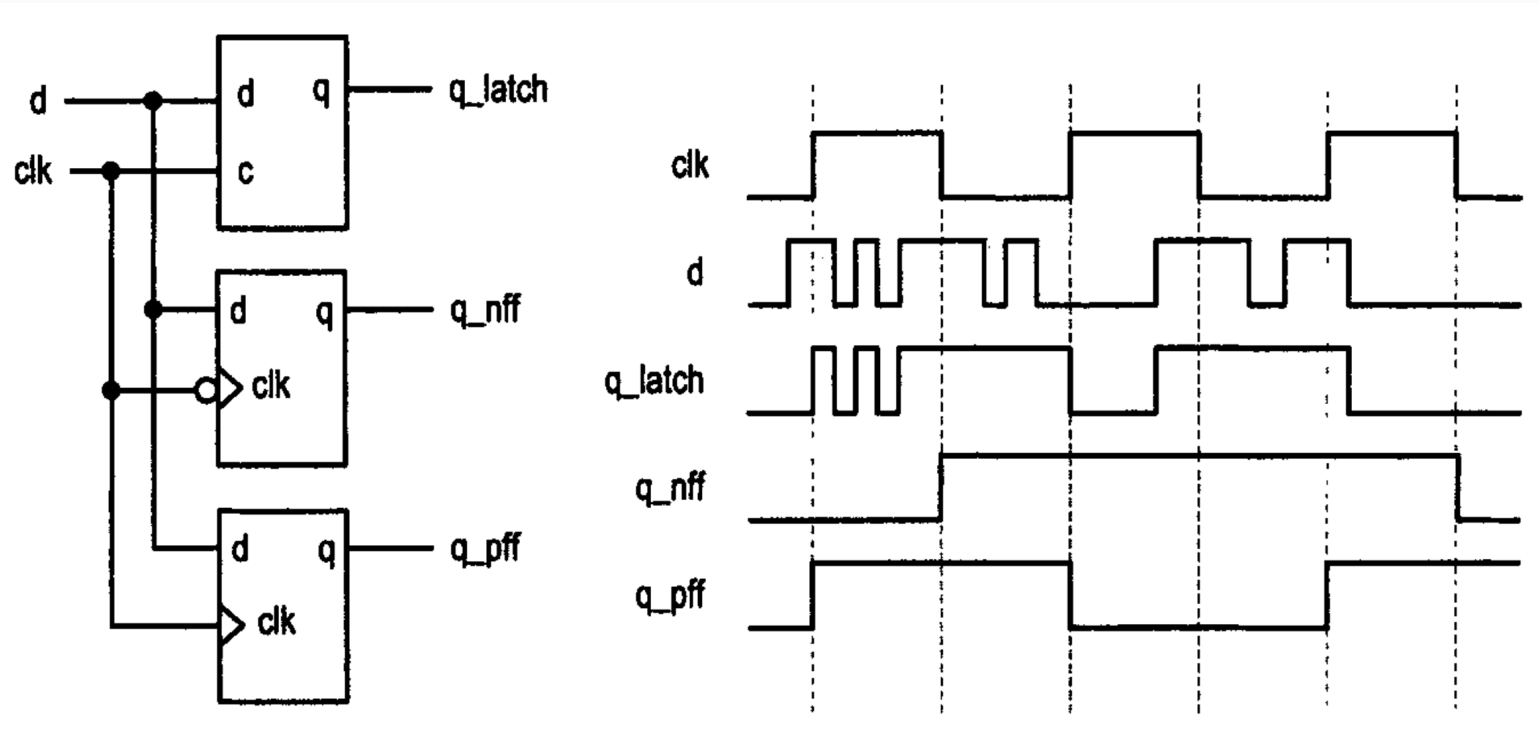
Como vimos en la primer parte, nos interesa poder tener un control de los momentos de transición de estados $\Rightarrow$ **CLOCK**

Vimos también que ser reactivo al nivel de una señal no es conveniente $\Rightarrow$ **Sensibilidad al flanco**

Sincronizando...

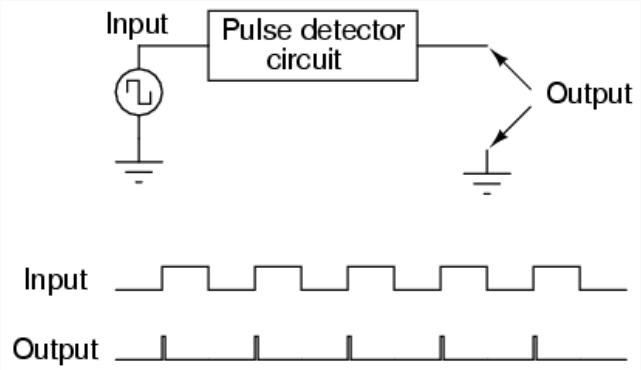
Como vimos en la primer parte, nos interesa poder tener un control de los momentos de transición de estados ⇒ **CLOCK**

Vimos también que ser reactivo al nivel de una señal no es conveniente ⇒ **Sensibilidad al flanco**



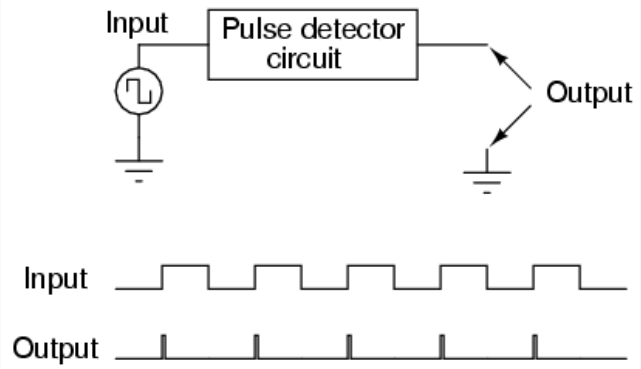
Detector de flanco

Necesitamos un circuito que se comporte de la siguiente manera:

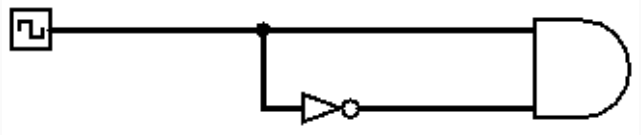


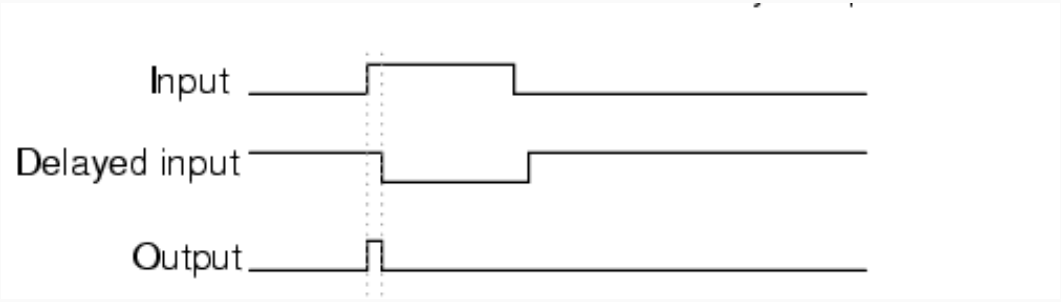
Detector de flanco

Necesitamos un circuito que se comporte de la siguiente manera:



Entonces, aprovechando los tiempos de propagación:





Flip-Flop D (Delay)

Ahora nuestro latch es sólo sensible a los flancos ascendentes de clock, entonces:

Lo podemos representar:

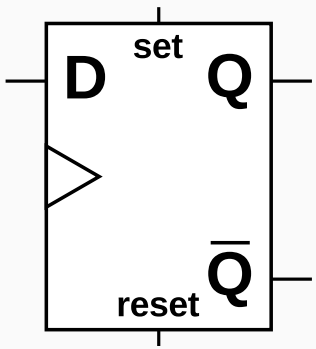


Tabla de verdad:

D	clk	Q_{T+1}	$\overline{Q_{T+1}}$
1	0		
0	$1\uparrow$		
0	0		
1	$1\uparrow$		

Flip-Flop D (Delay)

Ahora nuestro latch es sólo sensible a los flancos ascendentes de clock, entonces:

Lo podemos representar:

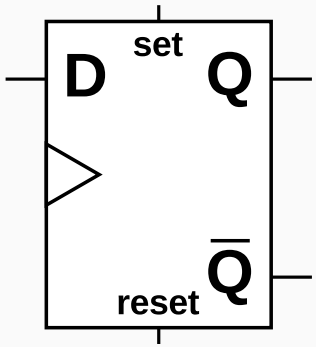


Tabla de verdad:

D	clk	Q_{T+1}	$\overline{Q_{T+1}}$
1	0		$\overline{Q_T}$
0	$1\uparrow$		
0	0		
1	$1\uparrow$		

Flip-Flop D (Delay)

Ahora nuestro latch es sólo sensible a los flancos ascendentes de clock, entonces:

Lo podemos representar:

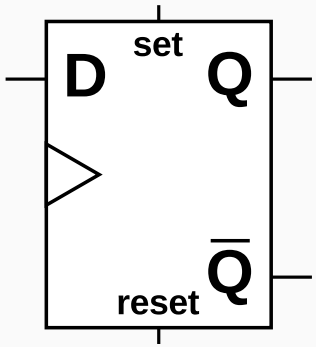


Tabla de verdad:

D	clk	Q_{T+1}	$\overline{Q_{T+1}}$
1	0	Q_T	$\overline{Q_T}$
0	$1\uparrow$		
0	0		
1	$1\uparrow$		

Flip-Flop D (Delay)

Ahora nuestro latch es sólo sensible a los flancos ascendentes de clock, entonces:

Lo podemos representar:

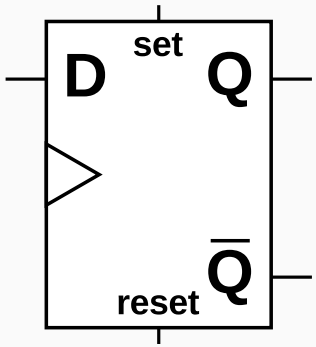


Tabla de verdad:

D	clk	Q_{T+1}	$\overline{Q_{T+1}}$
1	0	Q_T	$\overline{Q_T}$
0	$1\uparrow$	0	1
0	0		
1	$1\uparrow$		

Flip-Flop D (Delay)

Ahora nuestro latch es sólo sensible a los flancos ascendentes de clock, entonces:

Lo podemos representar:

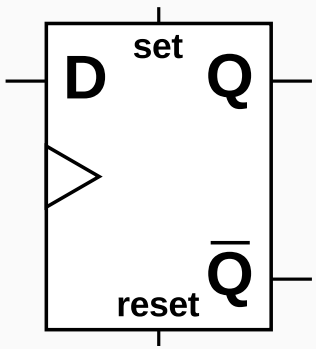


Tabla de verdad:

D	clk	Q_{T+1}	$\overline{Q_{T+1}}$
1	0	Q_T	$\overline{Q_T}$
0	$1\uparrow$	0	1
0	0	Q_T	$\overline{Q_T}$
1	$1\uparrow$		

Flip-Flop D (Delay)

Ahora nuestro latch es sólo sensible a los flancos ascendentes de clock, entonces:

Lo podemos representar:

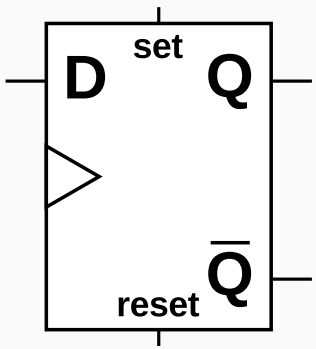


Tabla de verdad:

D	clk	Q_{T+1}	$\overline{Q_{T+1}}$
1	0	Q_T	$\overline{Q_T}$
0	$1\uparrow$	0	1
0	0	Q_T	$\overline{Q_T}$
1	$1\uparrow$	1	0

Flip-Flop D (Delay)

Ahora nuestro latch es sólo sensible a los flancos ascendentes de clock, entonces:

Lo podemos representar:

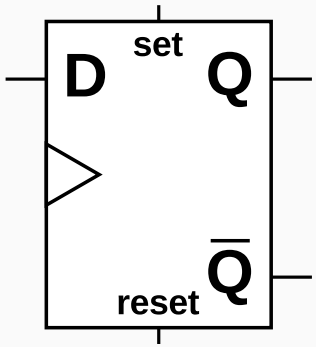


Tabla de verdad:

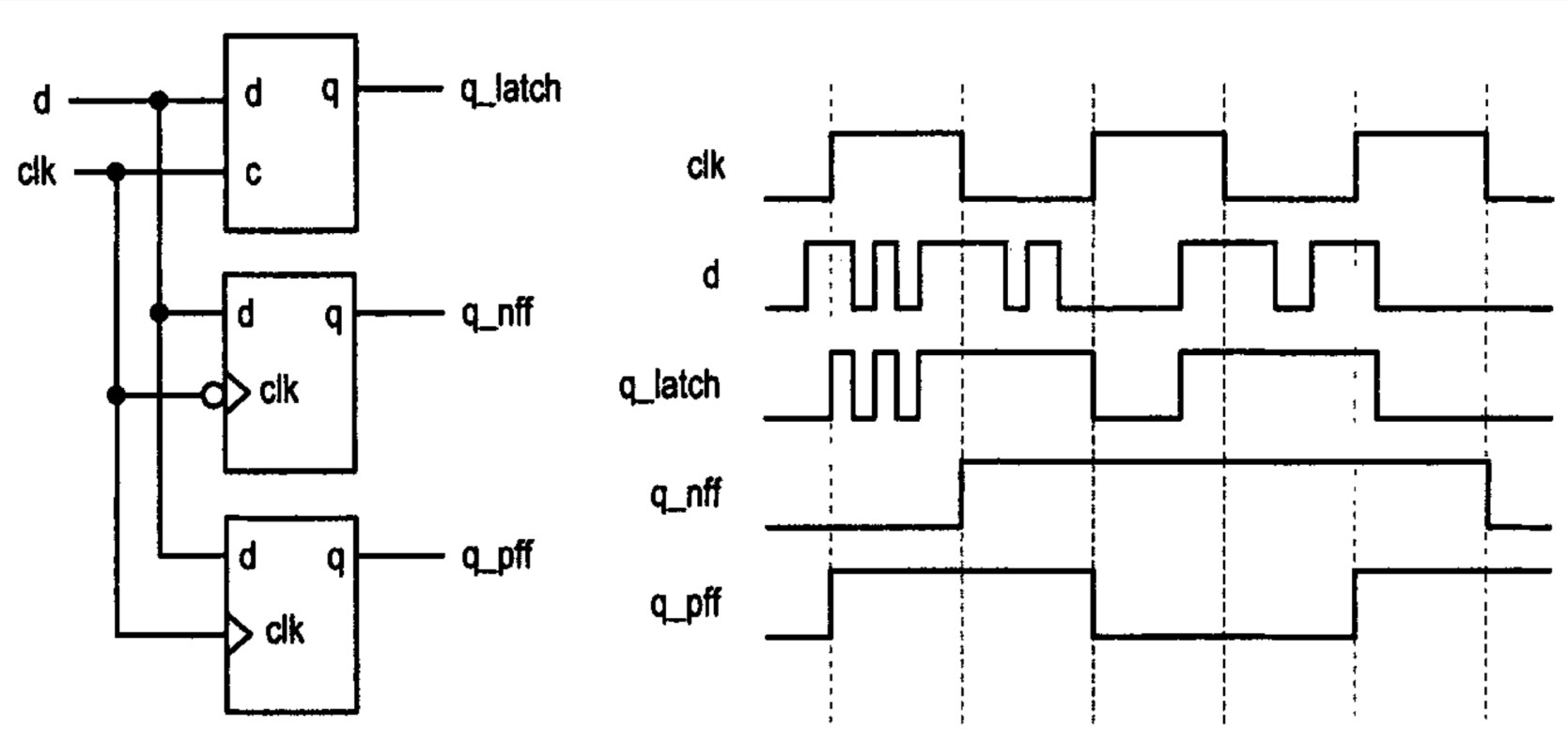
D	clk	Q_{T+1}	$\overline{Q_{T+1}}$
1	0	Q_T	$\overline{Q_T}$
0	1↑	0	1
0	0	Q_T	$\overline{Q_T}$
1	1↑	1	0

Siendo $T = n.T_{clock}$ y $T + 1 = (n + 1)T_{clock}$, donde:

- T_clock es el período del clock (tiempo que dura un ciclo)
- n es una cierta cantidad de pulsos de clock

Flip-Flop D (Delay)

Ahora podemos entender bien las diferencias:



Flip-Flop J-K

Volviendo al latch J-K, ahora con detección de flanco podemos obtener un comportamiento más *adecuado*:

Ahora lo podemos representar como:

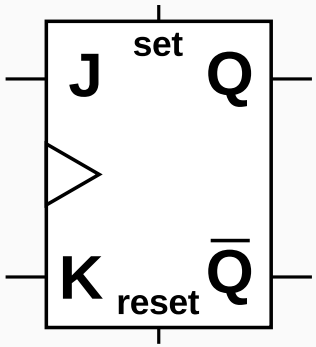


Tabla de verdad:

J	K	clk	Q_{T+1}	$\overline{Q}_{T+1}$
1	0	$1\uparrow$		
0	1	$1\uparrow$		
0	0	$1\uparrow$		
1	1	$1\uparrow$		
x	x	0		

Flip-Flop J-K

Volviendo al latch J-K, ahora con detección de flanco podemos obtener un comportamiento más *adecuado*:

Ahora lo podemos representar como:

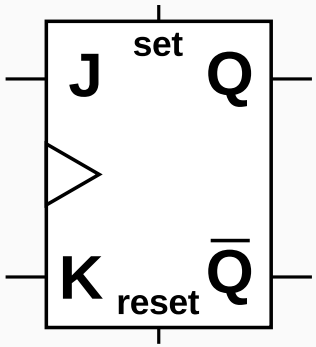


Tabla de verdad:

J	K	clk	Q_{T+1}	$\overline{Q}_{T+1}$
1	0	$1\uparrow$		0
0	1	$1\uparrow$		
0	0	$1\uparrow$		
1	1	$1\uparrow$		
x	x	0		

Flip-Flop J-K

Volviendo al latch J-K, ahora con detección de flanco podemos obtener un comportamiento más *adecuado*:

Ahora lo podemos representar como:

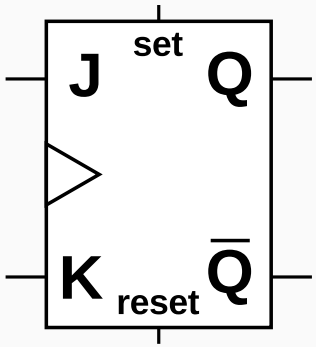


Tabla de verdad:

J	K	clk	Q_{T+1}	$\overline{Q}_{T+1}$
1	0	$1\uparrow$	1	0
0	1	$1\uparrow$		
0	0	$1\uparrow$		
1	1	$1\uparrow$		
x	x	0		

Flip-Flop J-K

Volviendo al latch J-K, ahora con detección de flanco podemos obtener un comportamiento más *adecuado*:

Ahora lo podemos representar como:

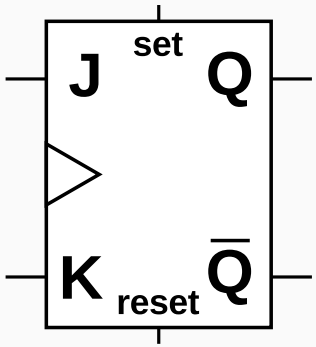


Tabla de verdad:

J	K	clk	Q_{T+1}	$\overline{Q}_{T+1}$
1	0	$1\uparrow$	1	0
0	1	$1\uparrow$	0	1
0	0	$1\uparrow$		
1	1	$1\uparrow$		
x	x	0		

Flip-Flop J-K

Volviendo al latch J-K, ahora con detección de flanco podemos obtener un comportamiento más *adecuado*:

Ahora lo podemos representar como:

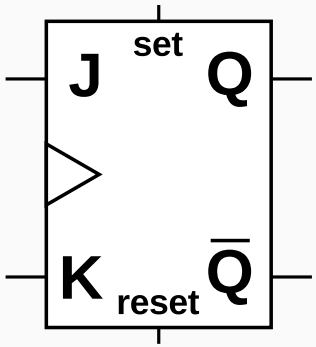


Tabla de verdad:

J	K	clk	Q_{T+1}	$\overline{Q}_{T+1}$
1	0	$1\uparrow$	1	0
0	1	$1\uparrow$	0	1
0	0	$1\uparrow$	Q_T	$\overline{Q}_T$
1	1	$1\uparrow$		
x	x	0		

Flip-Flop J-K

Volviendo al latch J-K, ahora con detección de flanco podemos obtener un comportamiento más *adecuado*:

Ahora lo podemos representar como:

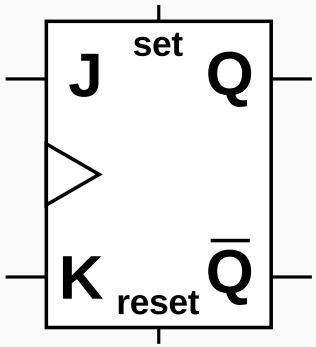


Tabla de verdad:

J	K	clk	Q_{T+1}	$\overline{Q}_{T+1}$
1	0	$1\uparrow$	1	0
0	1	$1\uparrow$	0	1
0	0	$1\uparrow$	Q_T	$\overline{Q}_T$
1	1	$1\uparrow$	$\overline{Q}_T$	Q_T
x	x	0		

Flip-Flop J-K

Volviendo al latch J-K, ahora con detección de flanco podemos obtener un comportamiento más *adecuado*:

Ahora lo podemos representar como:

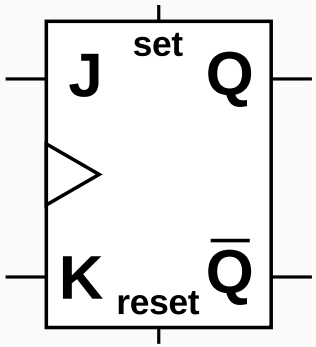


Tabla de verdad:

J	K	clk	Q_{T+1}	$\overline{Q}_{T+1}$
1	0	$1\uparrow$	1	0
0	1	$1\uparrow$	0	1
0	0	$1\uparrow$	Q_T	$\overline{Q}_T$
1	1	$1\uparrow$	$\overline{Q}_T$	Q_T
x	x	0	Q_T	Q_T

Flip-Flop J-K

Volviendo al latch J-K, ahora con detección de flanco podemos obtener un comportamiento más *adecuado*:

Ahora lo podemos representar como:

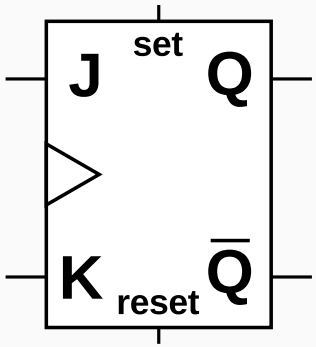


Tabla de verdad:

J	K	clk	Q_{T+1}	$\overline{Q}_{T+1}$
1	0	$1\uparrow$	1	0
0	1	$1\uparrow$	0	1
0	0	$1\uparrow$	Q_T	$\overline{Q}_T$
1	1	$1\uparrow$	$\overline{Q}_T$	Q_T
x	x	0	Q_T	Q_T

Ahora en el caso crítico donde $J, K = (1, 1)$ la salida tiene un estado y un tiempo de cambio bien definido:

Se niega el valor anterior cada 1 colck

Registros y memorias

Registros

Ya vimos como un FF D puede almacenar un bit... ¡pero sólo durante un clock!

Registros

Ya vimos como un FF D puede almacenar un bit... ¡pero sólo durante un clock!

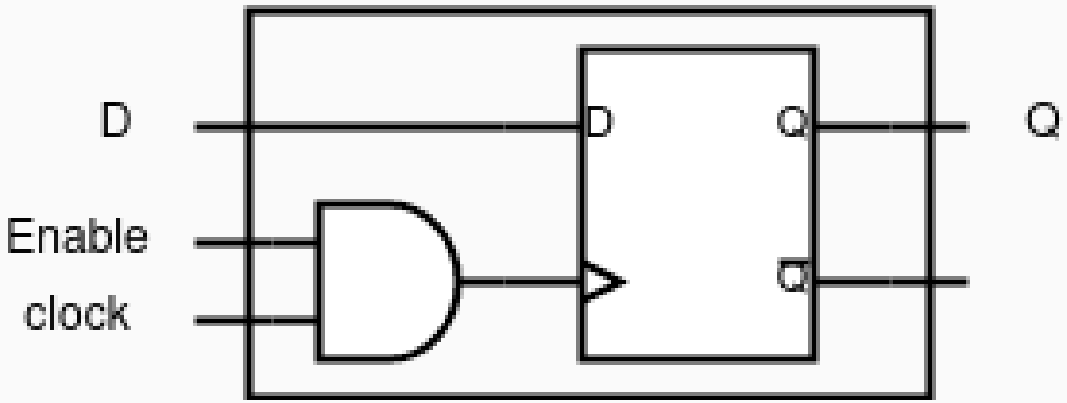
- Debemos poder elegir con una entrada adicional de control por cuanto tiempo queremos almacenar $\Rightarrow$ **enable**.

Registros

Ya vimos como un FF D puede almacenar un bit... ¡pero sólo durante un clock!

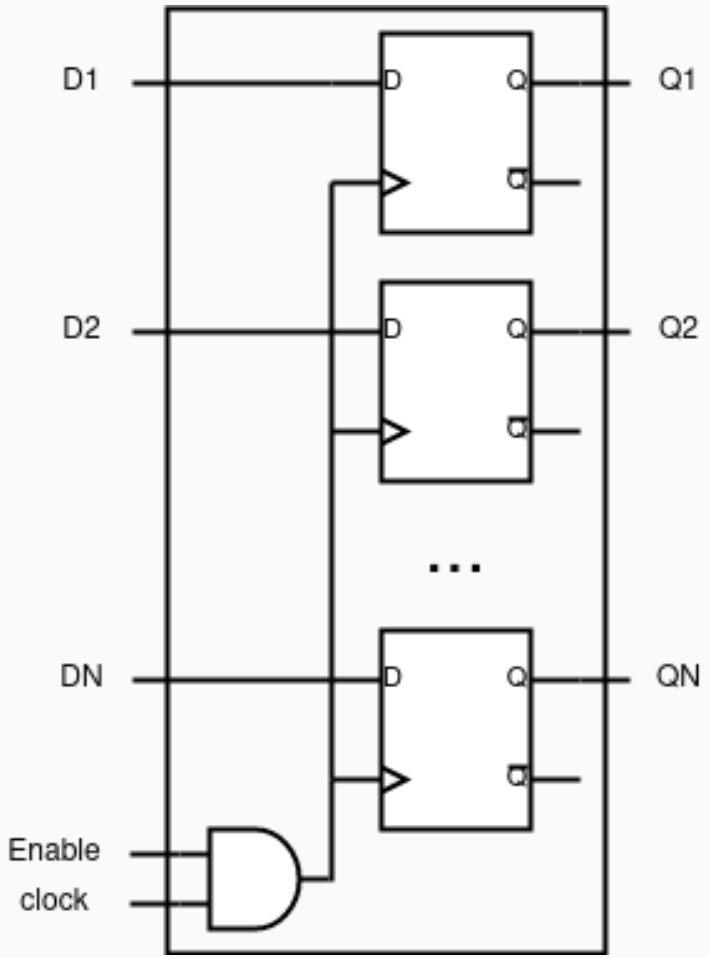
- Debemos poder elegir con una entrada adicional de control por cuanto tiempo queremos almacenar $\Rightarrow$ **enable**.

¡Sencillo!:



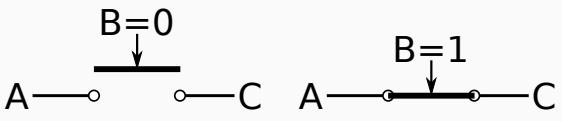
Registro de N-bits

Podemos componer la solución anterior para poder almacenar N bits:



Componentes de Tres Estados

Noción Eléctrica



Símbolo

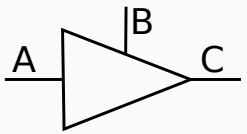
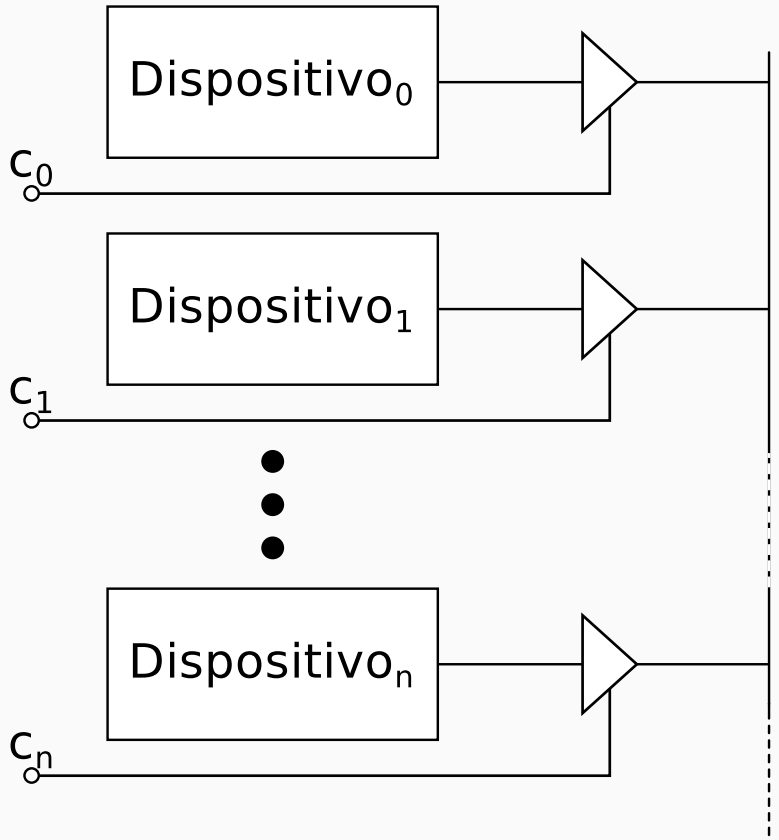


Tabla de Verdad

A	B	C
0	1	0
1	1	1
_	0	Hi-Z

Hi-Z significa "alta impedancia", es decir, que tiene una resistencia alta al pasaje de corriente. Como consecuencia de esto, podemos considerar al pin C como desconectado del circuito.

Componentes de Tres Estados



IMPORTANTE: Sólo deben ser usados a la salida de componentes para permitirles conectarse a un medio compartido (bus).

Ejercicio 0

a) Diseñar un registro de 3 *bits*. El mismo debe contar con 3 entradas $e_0, \ldots, e_2$ para ingresar el dato a almacenar, 3 salidas $s_0, \ldots, s_2$ para ver el dato almacenado y las señales de control CLK, RESET y WRITEENABLE.

Ejercicio 0

a) Diseñar un registro de 3 *bits*. El mismo debe contar con 3 entradas $e_0, \ldots, e_2$ para ingresar el dato a almacenar, 3 salidas $s_0, \ldots, s_2$ para ver el dato almacenado y las señales de control CLK, RESET y WRITEENABLE.

b) Modificar el diseño anterior agregándole componentes de 3 estados para que sólo cuando se active la señal de control ENABLEOUT muestre el dato almacenado.

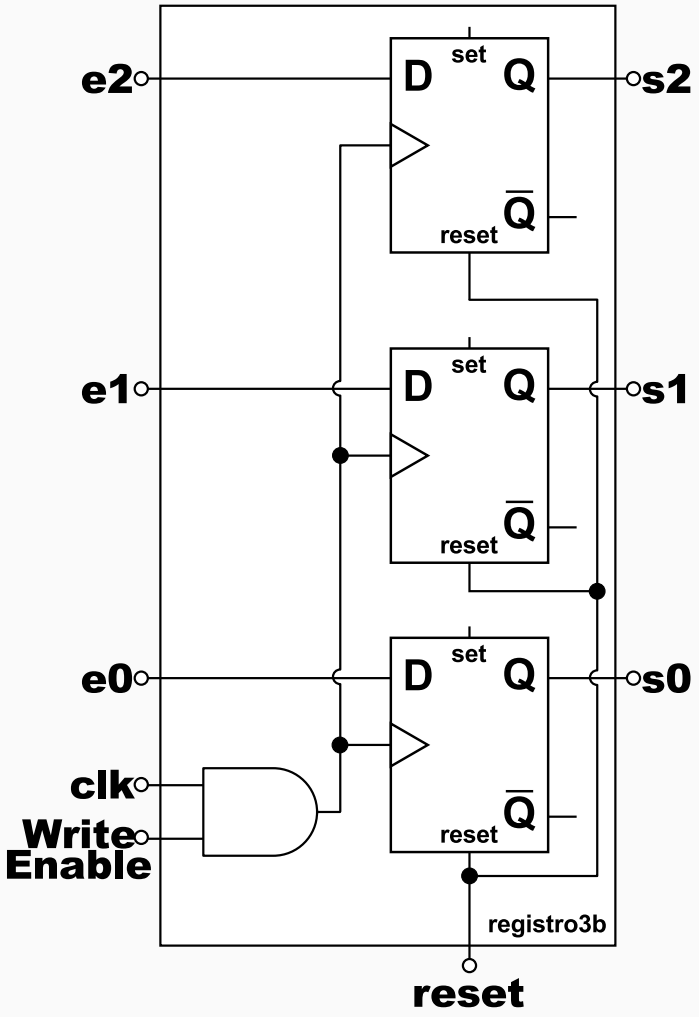
Ejercicio 0

a) Diseñar un registro de 3 *bits*. El mismo debe contar con 3 entradas $e_0, \ldots, e_2$ para ingresar el dato a almacenar, 3 salidas $s_0, \ldots, s_2$ para ver el dato almacenado y las señales de control CLK, RESET y WRITEENABLE.

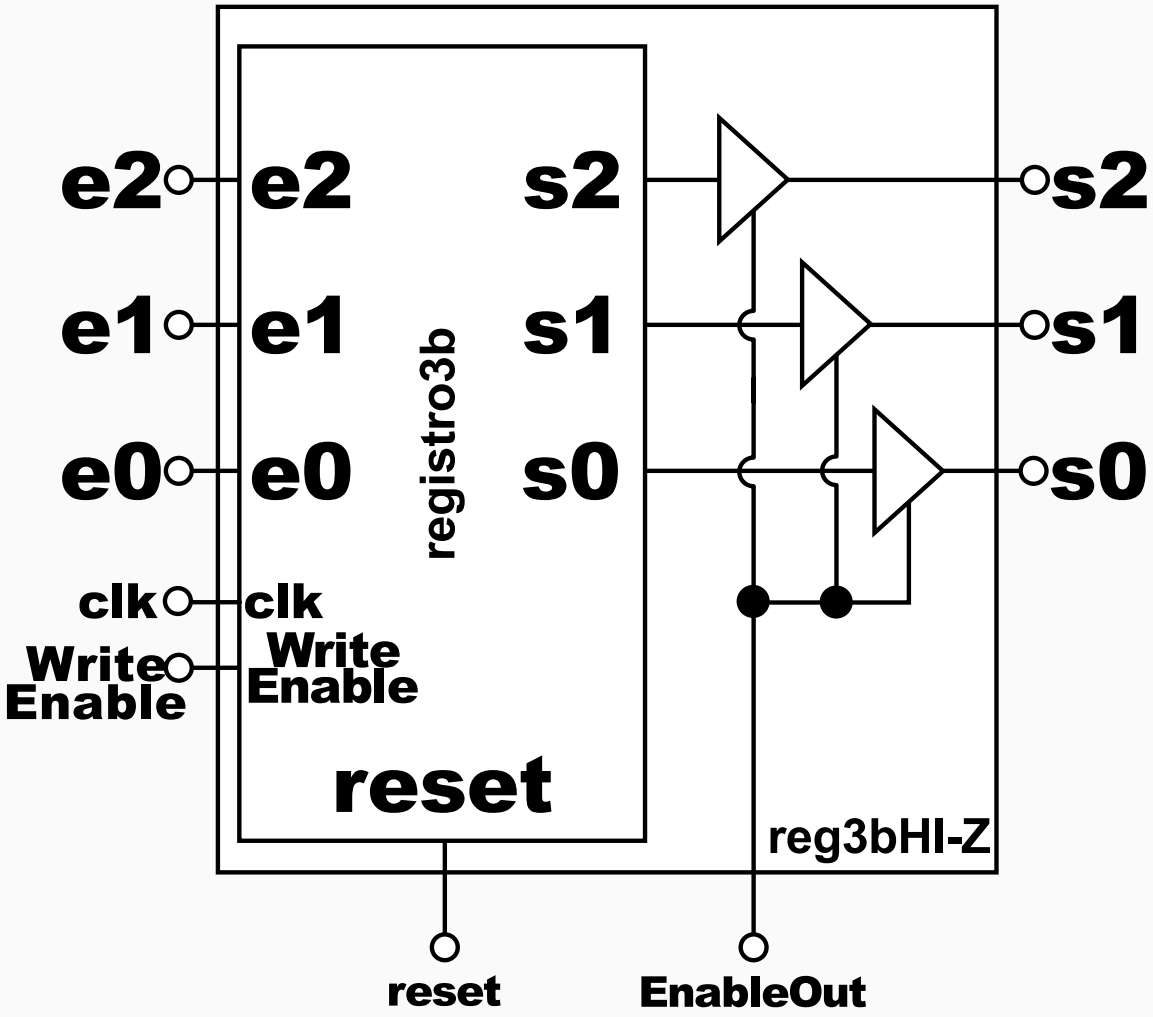
b) Modificar el diseño anterior agregándole componentes de 3 estados para que sólo cuando se active la señal de control ENABLEOUT muestre el dato almacenado.

c) Modificar nuevamente el diseño para que e_i y s_i estén conectadas entre sí al mismo tiempo teniendo en lugar de 3 entradas y 3 salidas, 3 entrada-salidas

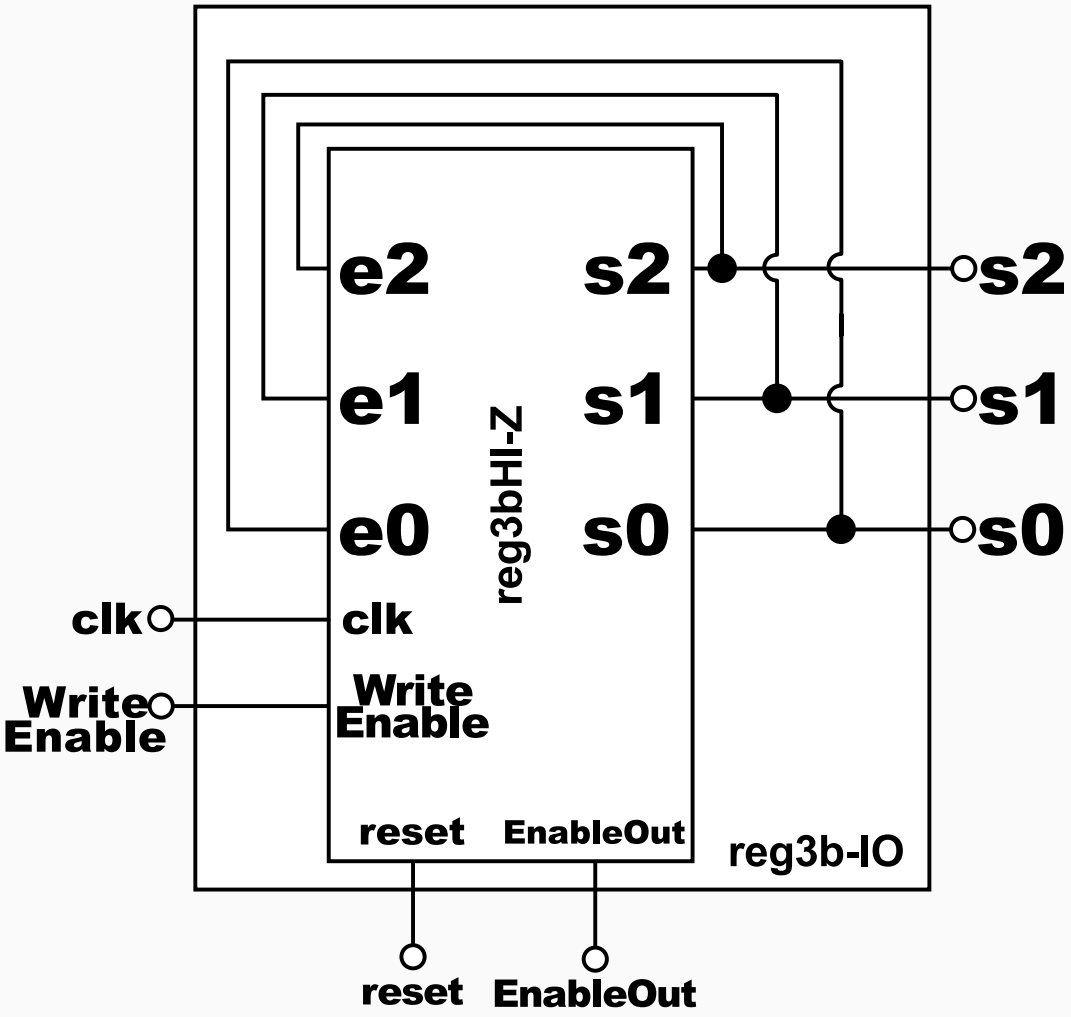
Solución - Ejercicio 0.a



Solución - Ejercicio 0.b



Solución - Ejercicio 0.c



Ejercicio 1

a) Realizar el esquema de interconexión de n registros como el diseñado

Ejercicio 1

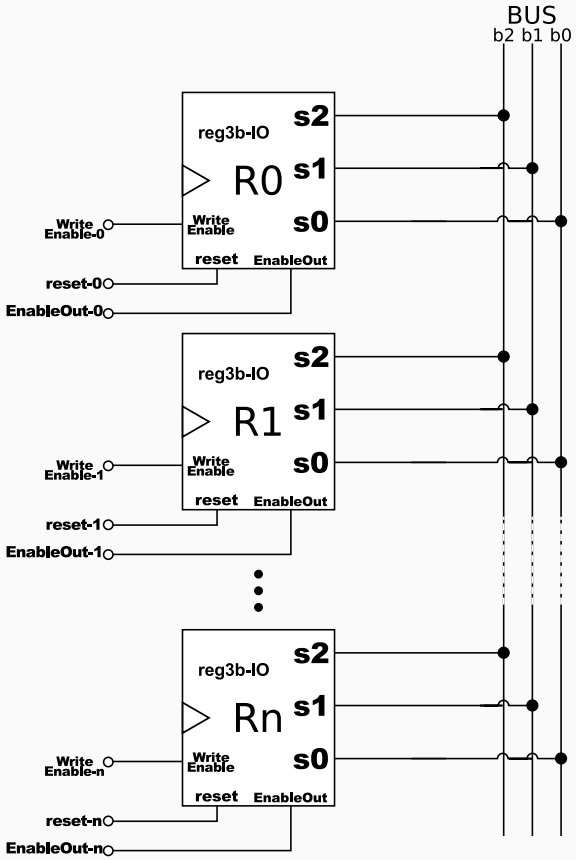
a) Realizar el esquema de interconexión de n registros como el diseñado

b) Dar una secuencia de valores de las señales de control para que se copie el dato del R1 al R0

Ejercicio 1

a) Realizar el esquema de interconexión de n registros como el diseñado

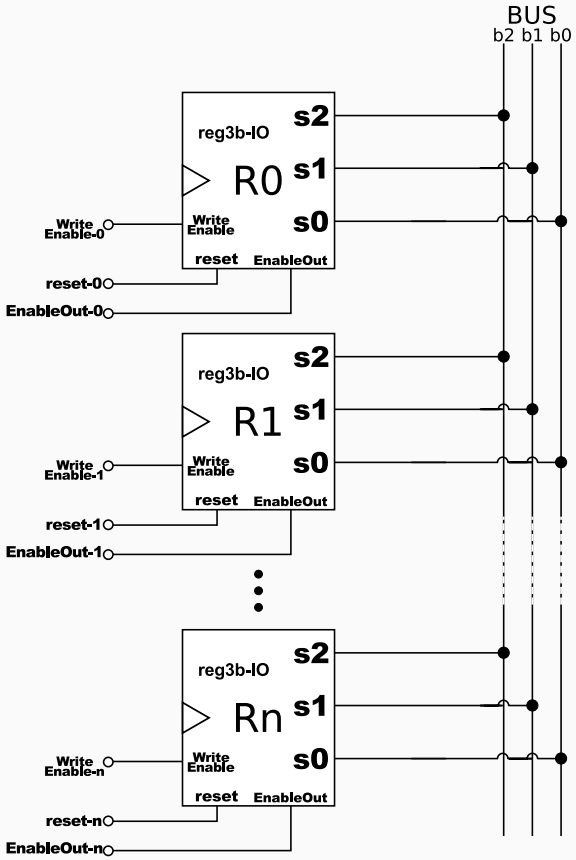
b) Dar una secuencia de valores de las señales de control para que se copie el dato del R1 al R0



Ejercicio 1

a) Realizar el esquema de interconexión de n registros como el diseñado

b) Dar una secuencia de valores de las señales de control para que se copie el dato del R1 al R0



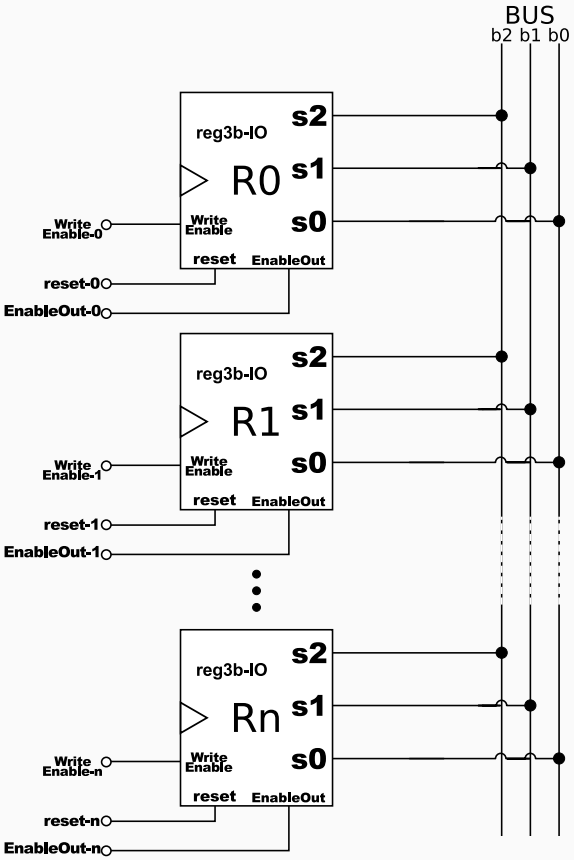
Señales de control:

R0	R1	...	Rn
WriteEnable-0	WriteEnable-1	...	WriteEnable-n
reset-0	reset-1	...	reset-n
EnableOut-0	EnableOut-1	...	EnableOut-n

Ejercicio 1

a) Realizar el esquema de interconexión de n registros como el diseñado

b) Dar una secuencia de valores de las señales de control para que se copie el dato del R1 al R0



Señales de control:

R0	R1	...	Rn
WriteEnable-0	WriteEnable-1	...	WriteEnable-n
reset-0	reset-1	...	reset-n
EnableOut-0	EnableOut-1	...	EnableOut-n

Inician todas las señales en 0. Luego se sigue la siguiente secuencia:

- EnableOut-1 $\leftarrow$ 1
- WriteEnable-0 $\leftarrow$ 1
- ...clk....
- WriteEnable-0 $\leftarrow$ 0
- EnableOut-1 $\leftarrow$ 0

Memorias (intro)

Conceptualmente podemos pensar una memoria como M posiciones de almacenamiento de N bits cada una.

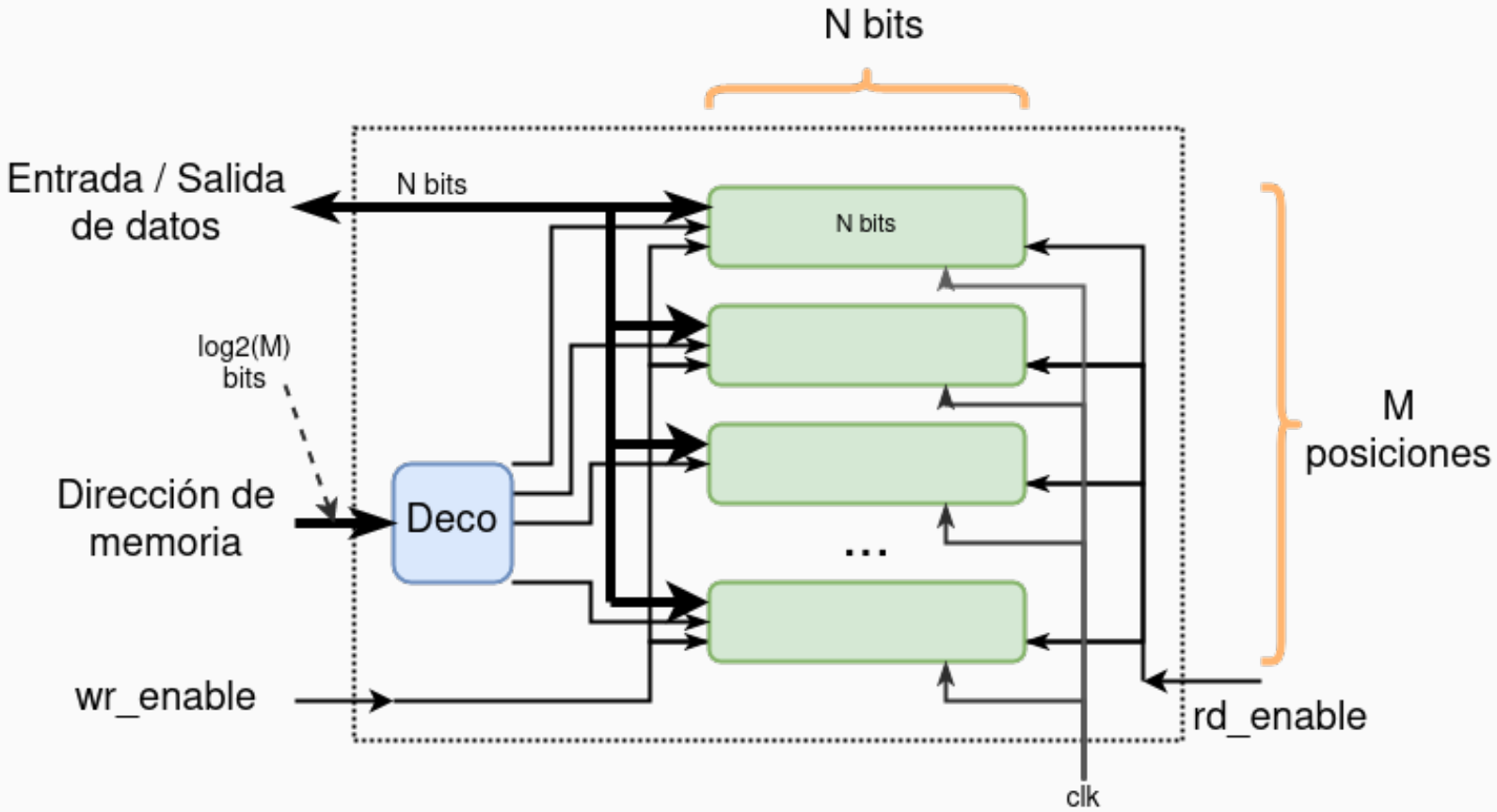
Memorias (intro)

Conceptualmente podemos pensar una memoria como M posiciones de almacenamiento de N bits cada una.

Debemos poder seleccionar a cuál queremos acceder

Memorias (intro)

Conceptualmente podemos pensar una memoria como M posiciones de almacenamiento de N bits cada una.
Debemos poder seleccionar a cuál queremos acceder



Conclusiones

Conclusiones

- Estudiamos la realimentación en circuitos para mantener un dato en el tiempo y vimos implementaciones de *latchs*
- Estudiamos los problemas asociados a los tiempos de propagación de las señales
- Analizamos el uso de un *clock* para limitar la cantidad de estados, controlar las transiciones y evitar carreras
- Vimos algunas implementaciones de flip-flops, registros y memorias

Conclusiones

- Estudiamos la realimentación en circuitos para mantener un dato en el tiempo y vimos implementaciones de *latchs*
- Estudiamos los problemas asociados a los tiempos de propagación de las señales
- Analizamos el uso de un *clock* para limitar la cantidad de estados, controlar las transiciones y evitar carreras
- Vimos algunas implementaciones de flip-flops, registros y memorias

Conclusiones

- Estudiamos la realimentación en circuitos para mantener un dato en el tiempo y vimos implementaciones de *latchs*
- Estudiamos los problemas asociados a los tiempos de propagación de las señales
- Analizamos el uso de un *clock* para limitar la cantidad de estados, controlar las transiciones y evitar carreras
- Vimos algunas implementaciones de flip-flops, registros y memorias

Conclusiones

- Estudiamos la realimentación en circuitos para mantener un dato en el tiempo y vimos implementaciones de *latchs*
- Estudiamos los problemas asociados a los tiempos de propagación de las señales
- Analizamos el uso de un *clock* para limitar la cantidad de estados, controlar las transiciones y evitar carreras
- Vimos algunas implementaciones de flip-flops, registros y memorias

La práctica...

- Con lo visto hoy pueden realizar la **parte A de la práctica 2**.
- Pueden usar el purpleLogisim evolution (Requiere Java 16 o superior. Para ejecutarlo, teclear en una consola `java -jar logisim-evolution-3.8.0-all.jar` desde la carpeta donde se encuentra el archivo descargado.)
- El próximo jueves deberíamos el **primer taller** de la materia, el cual es **obligatorio**. Será en los laboratorios del pabellón **Cero+Infinito** (ver cuales en el cronograma que está en el campus).
- Bibliografía recomendada: *The Essentials of Computer Organization and Architecture - Linda Null - Capítulo 3*