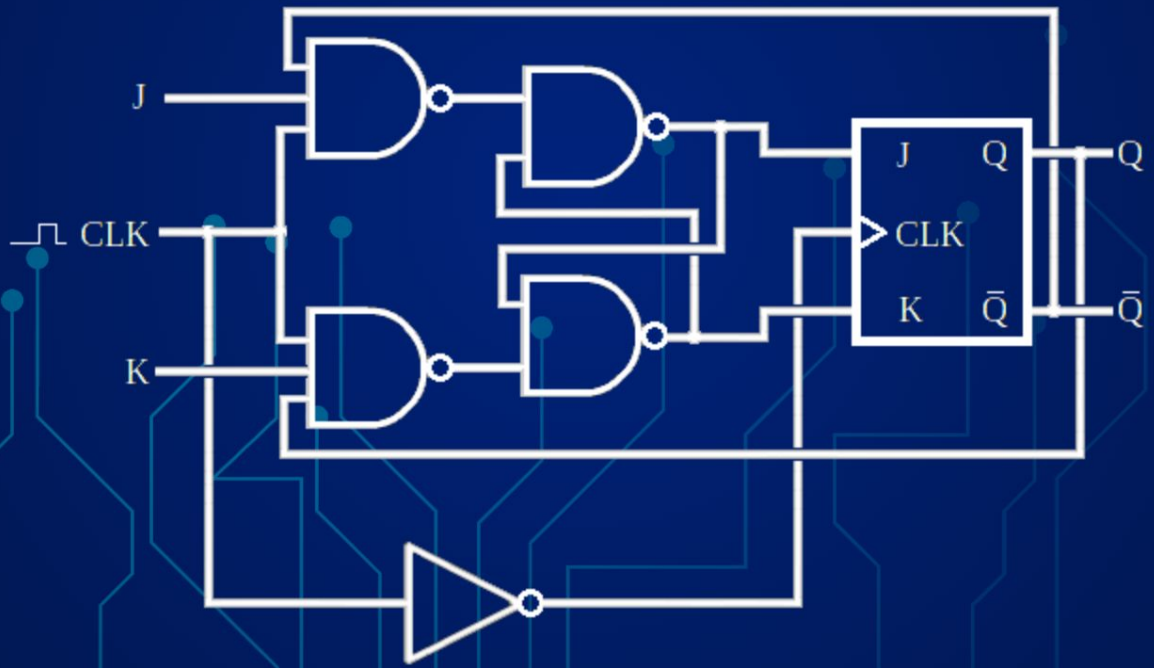
Arquitectura del Computador y Sistemas Operativos

Segunda Clase

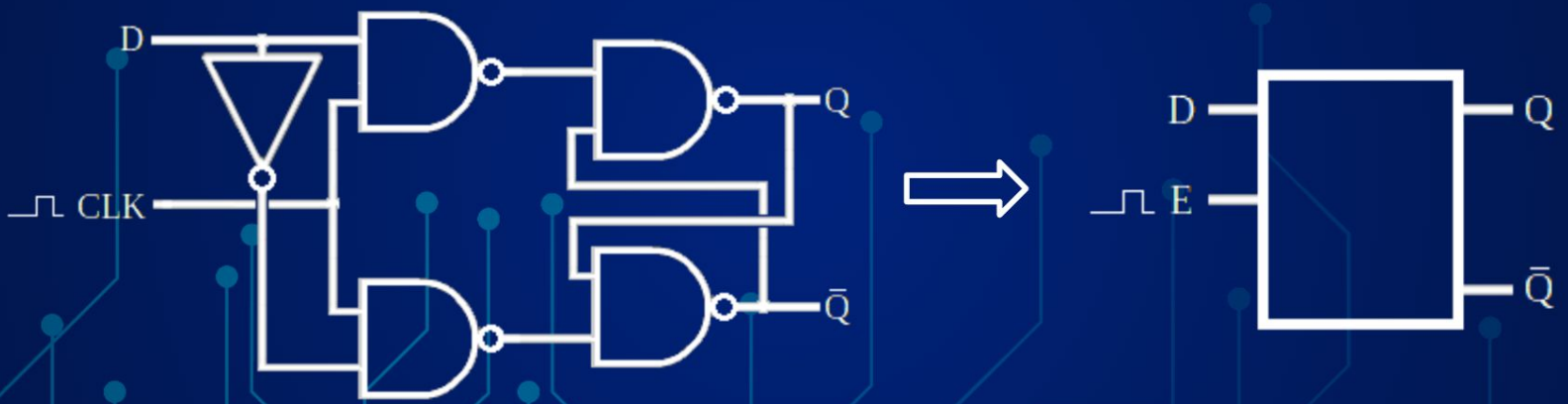
Flip Flops (3/6)

El Flip Flop JK MS (Master-Slave):



Flip Flops (4/6)

El Flip Flop D (Transparent Latch):



El FF de tipo D copia el valor de la entrada a la salida durante todo el tiempo en que la señal CLK está en 1. Al caer CLK a 0, mantiene el último valor independientemente de lo que haga la entrada.

Flip Flops (5/6)

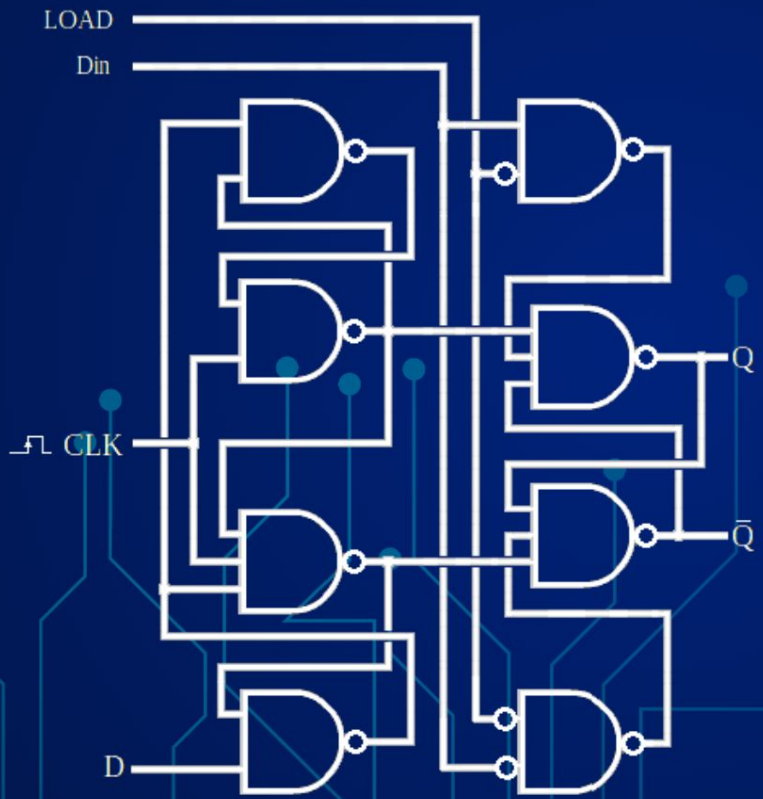
El Flip Flop D disparado por flanco ascendente:

Este FF copia la entrada D a la salida Q en el instante en que CLK pasa de 0 a 1. Un cambio en D en cualquier otro instante no altera la salida..

Ejercicio

- Analice el funcionamiento del FF tipo D disparado por flanco.
- Demuestre que los cambios en la entrada D, esté CLK estable en 0 o en 1 no producen cambios en la salida.
- Realice un diagrama de estados donde se vean todas las posibles combinaciones de entradas y salidas.

Flip Flops (6/6)



Podemos agregarle una última función a este FF.

Con las dos compuertas NAND nuevas se incorpora la posibilidad de inicializar el valor de la salida colocando en valor deseado en Din y subiendo la línea LOAD a 1 por unos instantes.

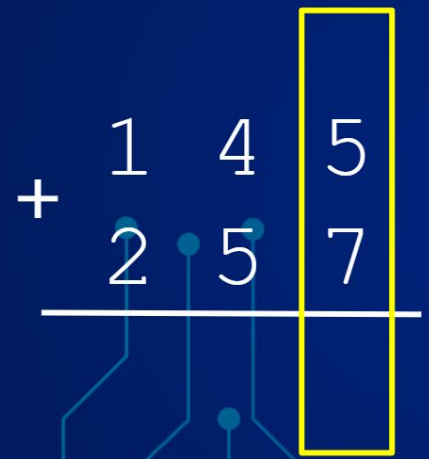
Sumador (1/3)

Pensemos en el algoritmo que usamos para sumar:

$$\begin{array}{r} 1\ 4\ 5 \\ +\ 2\ 5\ 7 \\ \hline \end{array}$$

Sumador (1/3)

Pensemos en el algoritmo que usamos para sumar:



Unidades:

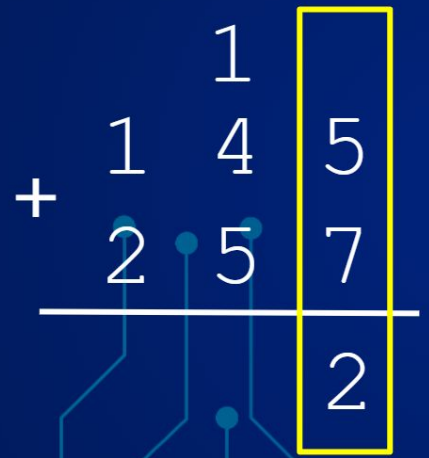
5 + 7 = 12

Resultado: 2

Llevo : 1

Sumador (1/3)

Pensemos en el algoritmo que usamos para sumar:



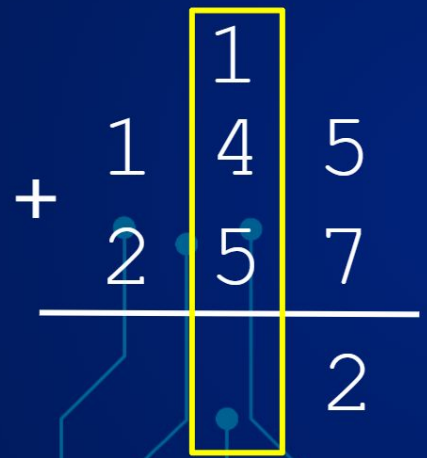
Unidades:

5 + 7 = 12

Resultado: 2
Llevo : 1

Sumador (1/3)

Pensemos en el algoritmo que usamos para sumar:



Unidades:

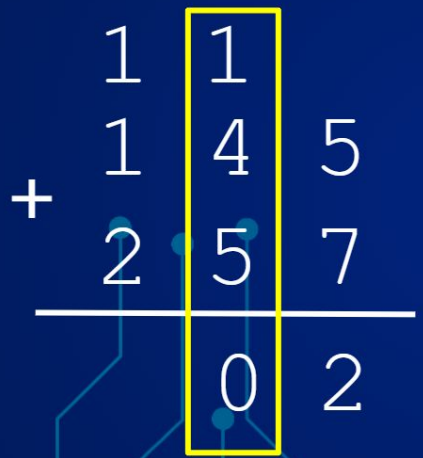
5 + 7 = 12 Resultado: 2
Llevo : 1

Decenas:

1 + 4 + 5 = 10 Resultado: 0
Llevo : 1

Sumador (1/3)

Pensemos en el algoritmo que usamos para sumar:



Unidades:

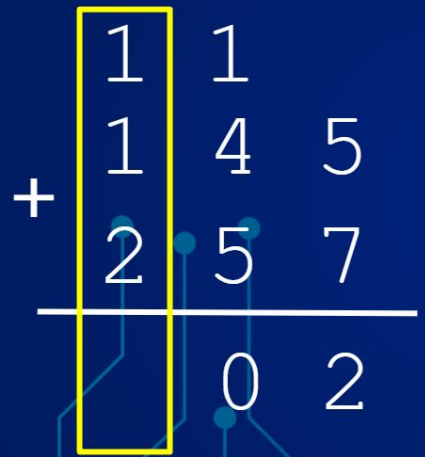
5 + 7 = 12 Resultado: 2
Llevo : 1

Decenas:

1 + 4 + 5 = 10 Resultado: 0
Llevo : 1

Sumador (1/3)

Pensemos en el algoritmo que usamos para sumar:



Unidades:

5 + 7 = 12 Resultado: 2
Llevo : 1

Decenas:

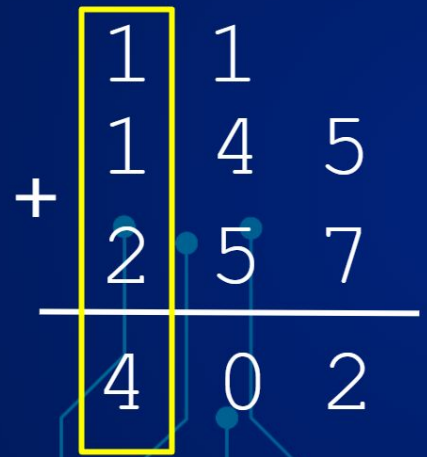
1 + 4 + 5 = 10 Resultado: 0
Llevo : 1

Centenas::

1 + 1 + 2 = 4 Resultado: 4
Llevo : 0

Sumador (1/3)

Pensemos en el algoritmo que usamos para sumar:



Unidades:

5 + 7 = 12 — Resultado: 2, Llevo : 1

Decenas:

1 + 4 + 5 = 10 — Resultado: 0, Llevo : 1

Centenas::

1 + 1 + 2 = 4 — Resultado: 4, Llevo : 0

Sumador (1/3)

Pensemos en el algoritmo que usamos para sumar:

```
  1  1
  1  4  5
+ 2  5  7
-----------
  4  0  2
```

Unidades:

5 + 7 = 12 — Resultado: 2, Llevo : 1

Decenas:

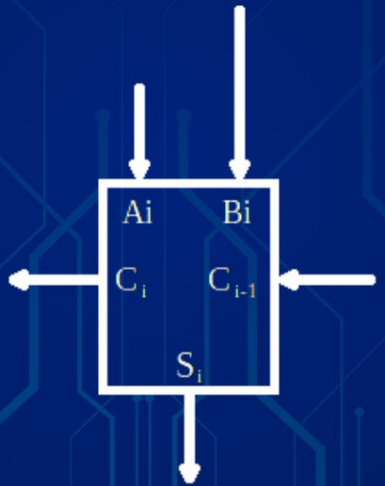
1 + 4 + 5 = 10 — Resultado: 0, Llevo : 1

Centenas::

1 + 1 + 2 = 4 — Resultado: 4, Llevo : 0

Sumador (2/3)

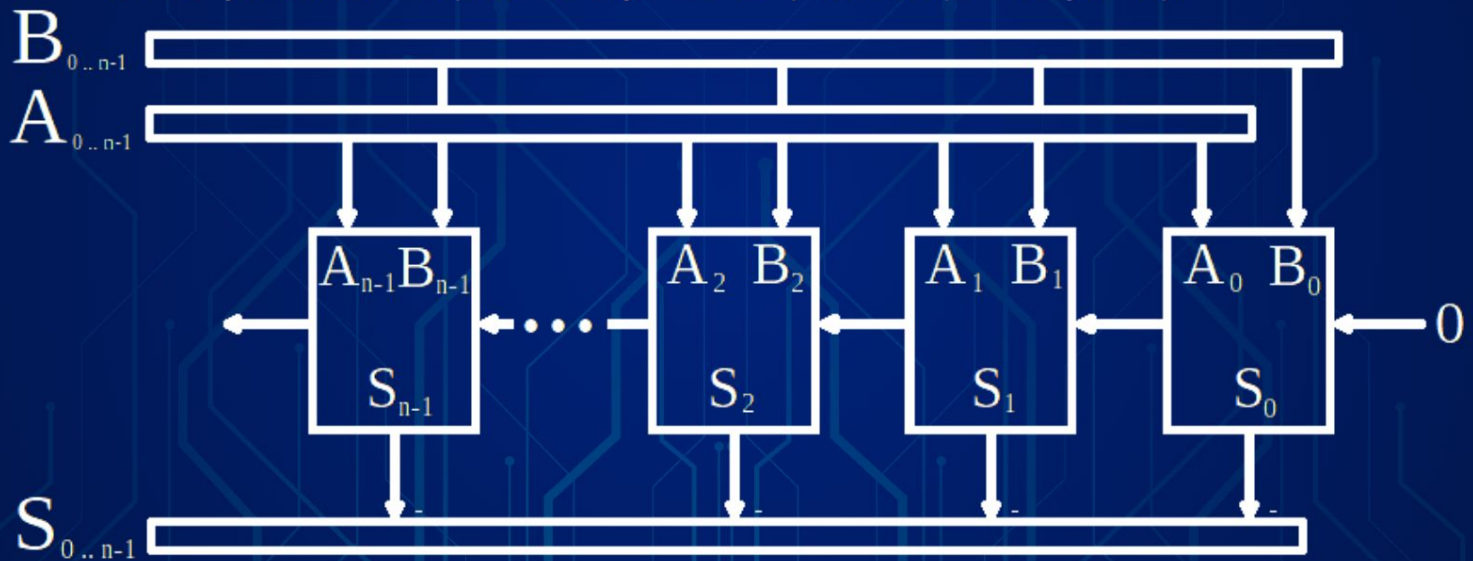
Podemos encapsular ese algoritmo en un bloque, que tiene por entrada los dos dígitos de una columna (unidades, decenas, centenas, etc) y el dígito que nos llevamos de la columna anterior.
Tiene dos salidas, el dígito suma y el indicador de que nos llevamos una unidad a la columna siguiente:



Si ahora extrapolamos el algoritmo a base 2, cada dígito puede ser solamente 0 o 1, por lo que cada una de las flechas de arriba es un cable que conduce (1) o no (0) electricidad. En otras palabras cada flecha es un bit de información.

Sumador (3/3)

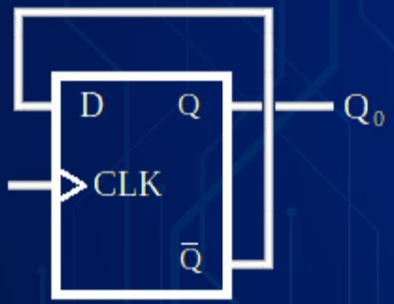
Para sumar números de más de un bit podemos repetir el algoritmo columna a columna tal como hicimos al sumar manualmente, colocando un conjunto de bloques trabajando en paralelo (en conjunto).

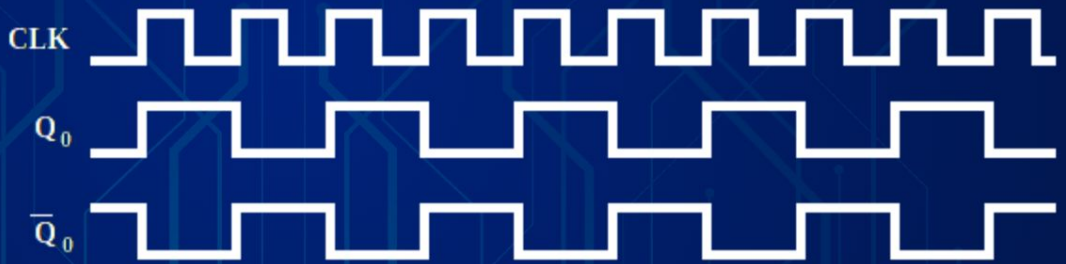


Las líneas con propósitos generales se agruparon por simplicidad. Llamamos a estos conjuntos BUSES.

Contador

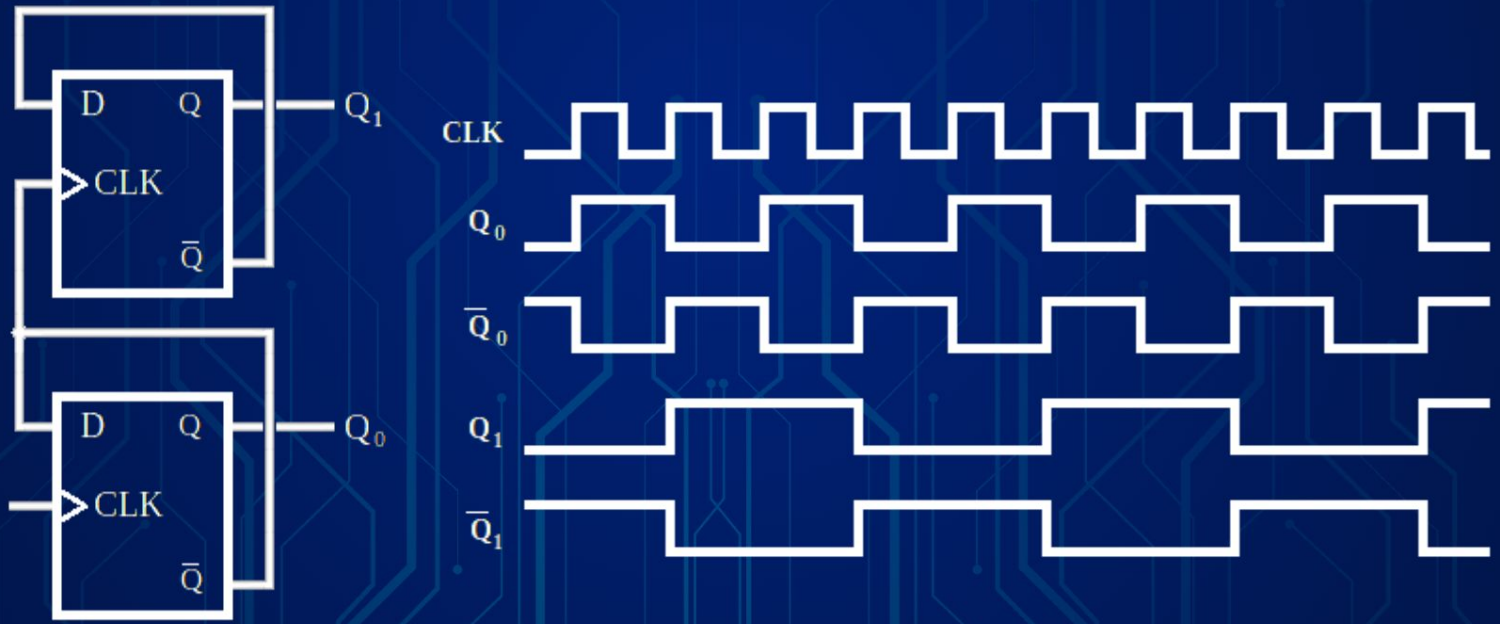
Esta misma técnica de lograr tareas complejas en base a un conjunto de bloques simples puede usarse para otras cosas.
Por ejemplo tomemos el FF tipo D disparado por flanco:





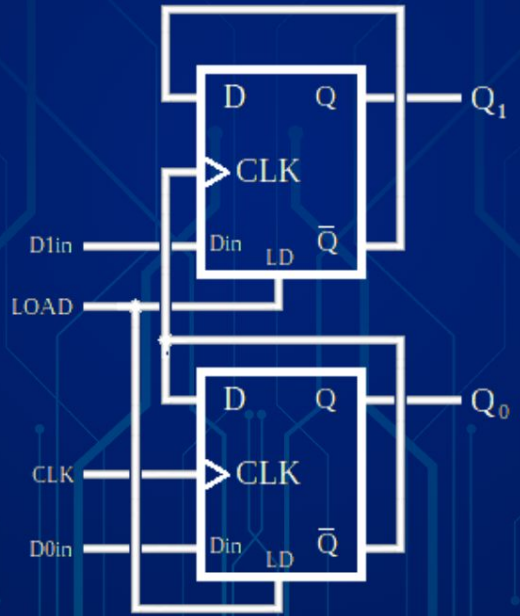
Contador

Esta misma técnica de lograr tareas complejas en base a un conjunto de bloques simples puede usarse para otras cosas.
Por ejemplo tomemos el FF tipo D disparado por flanco:



Contador

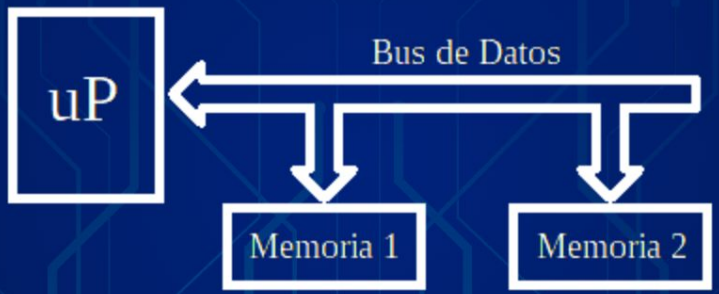
Esta misma técnica de lograr tareas complejas en base a un conjunto de bloques simples puede usarse para otras cosas.
Por ejemplo tomemos el FF tipo D disparado por flanco:



Tri-State (1/2)

Por ahora hemos visto a un BUS como un conjunto de señales afines que salen de determinado bloque y entran a otro. En algunos casos se desea que las líneas del BUS sean compartidas entre varios dispositivos.

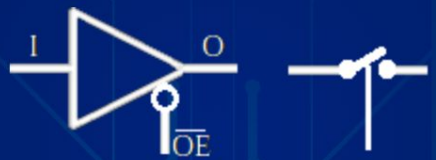
Por ejemplo:



Este procesador tiene conectados dos bancos de memoria. Para poder compartir el bus de datos, se desea que cuando al procesador lee de la Memoria 1, sea ésta la que coloque los datos en el bus, sean 0s o 1s de acuerdo al contenido almacenado. La Memoria 2 no debería imponer ni 0s ni 1s, simplemente debería *desconectarse* para dejar que los datos fluyan de la Memoria 1 al procesador.
De la misma forma, cuando el procesador accede a la Memoria 2 la que debería *desconectarse* es la Memoria 1.

Tri-State (2/2)

La capacidad de un dispositivo conectado a un BUS de *desconectarse* del mismo para no influir en la transferencia de datos entre otros elementos conectados a ese BUS se conoce como Tri-State.

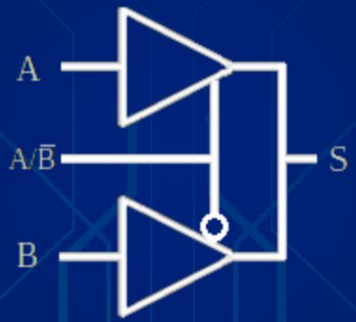


Gráficamente se representa como un triángulo sin inversor a la salida (dado que cuando el dispositivo está con la salida habilitada no invierte).

En un BUS *nunca* debería haber más de un dispositivo colocando datos en el mismo, dado que si hay más de uno, particularmente si alguno de los bits difiere, habrá un corto-circuito que dañaría los dispositivos involucrados.

Multiplexor

Podemos usar los buffers tri-state vistos antes para *elegir* qué dispositivo envía datos a otro a través del BUS.



Este dispositivo se llama *multiplexor*. En este caso se ejemplifica con uno de dos entradas, la señal A/B hace que la salida sea la entrada A o la B. Sin embargo se puede generalizar el concepto y tener multiplexores de N entradas y una salida.

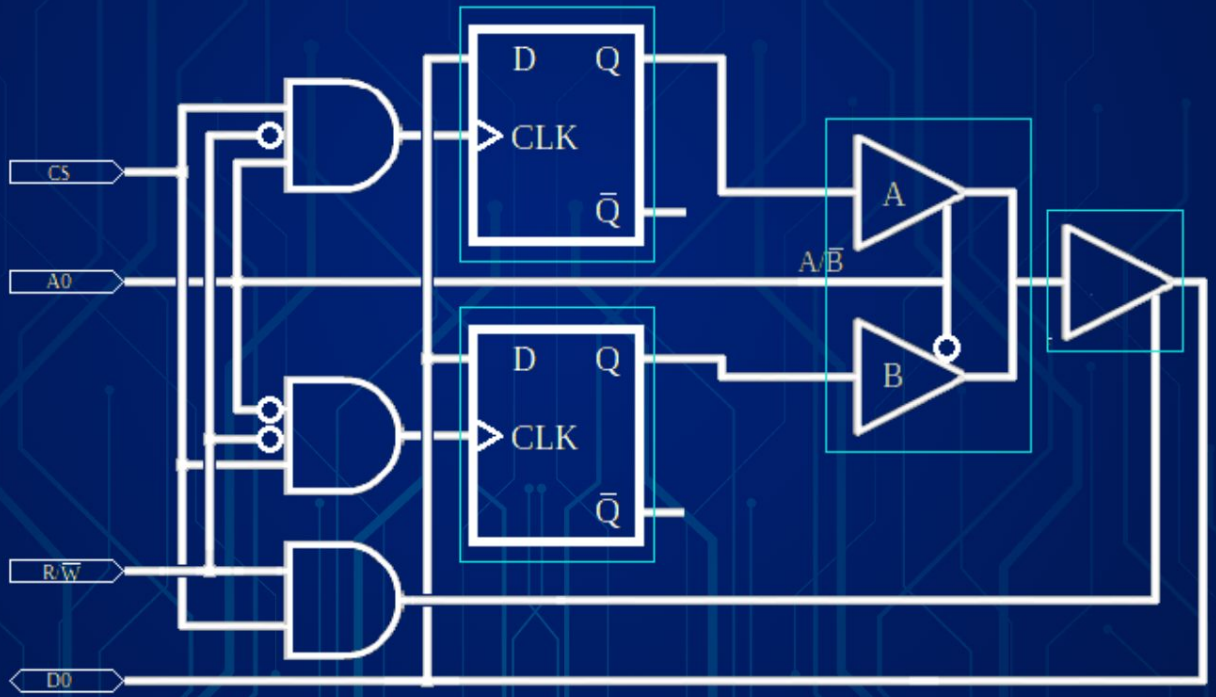
El Chip de Memoria Estática (1/2)

Ya vimos que un FF de tipo D almacena un bit de información. Además vimos que se pueden colocar varios en paralelo para almacenar varios bits en lo que llamamos un *registro*. La idea de un Chip de Memoria es colocar varios registros juntos de forma tal que podamos leer o modificar cualquiera de ellos.
El tamaño (cantidad de bits) del registro se denomina *palabra*. Por lo que decimos que la capacidad de la memoria es de N palabras de M bits cada una.

Estos chips tienen un BUS de DATOS reversible (es salida cuando leemos un registro y entrada para escribirlo), un BUS DE DIRECCIONES que no es más que un número binario que identifica el registro a leer/escribir, y lógica combinacional que permite que la única entrada de clock llegue al registro correcto al escribir o la salida del registro indicado llegue al bus de datos para leer.

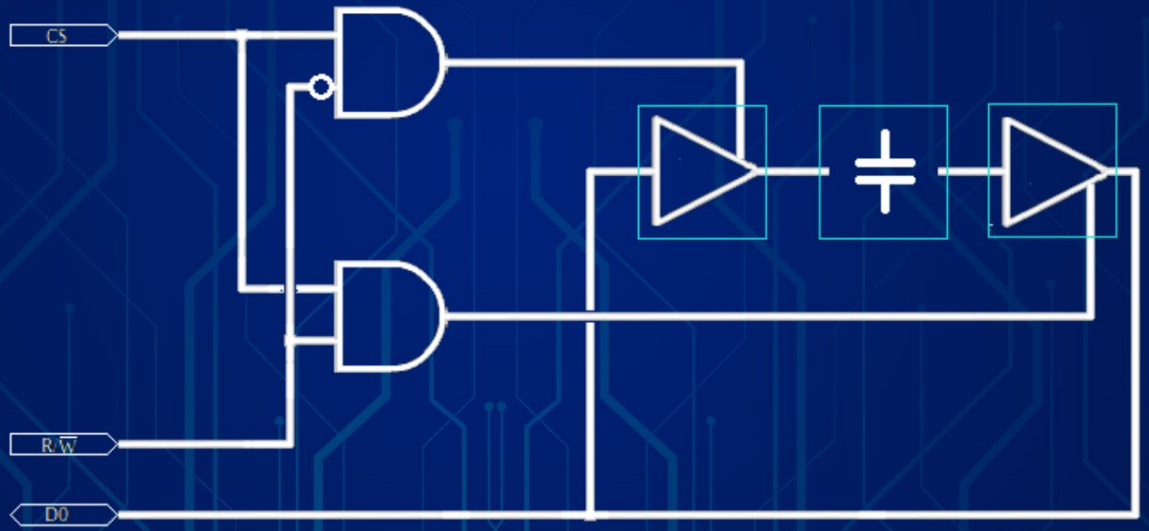
El Chip de Memoria Estática (2/2)

A continuación tenemos un ejemplo de una memoria de dos palabras de un solo bit cada una:



El Chip de Memoria Dinámica

El bit de memoria estática, una vez cargado con un valor, lo mantiene mientras haya energía. Sin embargo, el estar compuesto de muchas compuertas, hace que ocupe mucho lugar. Por este motivo, las RAMs estáticas no tienen mucha capacidad.



La utilización de un dispositivo de almacenamiento de carga la hace mucho más pequeñas, pero deben ser refrescadas peiódicamente.

Fin

¿Preguntas?