

Práctica 2

Creación de un banco de registros y su análisis temporal



12 de marzo de 2021

javier mengod y álvaro fraidias

Contenido

[1. Trabajo previo 2](#_Toc66466921)

[2. Diseño de un banco de registros en logisim 4](#_Toc66466922)

[3. Analisis temporal 6](#_Toc66466923)

[3.1 Multiplexor 4:1 6](#_Toc66466924)

[3.2 Multiplexor 8:1 7](#_Toc66466925)

[3.3 Decodificador 2:4 7](#_Toc66466926)

[3.4 Decodificador 3:8 8](#_Toc66466927)

[3.5 Análisis de los caminos combinacionales de la lectura 8](#_Toc66466928)

[3.6 Análisis de los caminos combinacionales de la escritura 8](#_Toc66466929)

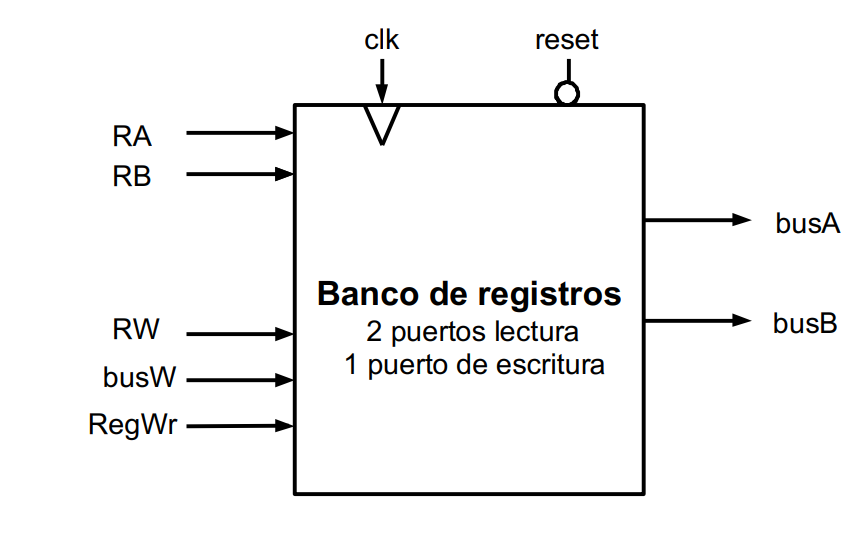
[4. Conclusión 9](#_Toc66466930)

[5. Bibliografía 9](#_Toc66466931)



# 1. Trabajo previo

Para diseñar un banco de registros primero debemos saber exactamente cuantas entradas –salidas y cuantos bits tendrá cada señal:



ENTRADAS:

El banco de registros tiene 7 entradas:

1. **RA**: 5 bits ya que habrá que seleccionar 32 bits (). Las 32 posibilidades se codifican con 5 bits.
2. **RB**: Igual que en el caso RA, también habrá que codificarlo con 5 bits ya que tiene la misma función que RA.
3. **RD**: Esta entrada hace prácticamente lo mismo que las entradas anteriores salvo que esta entrada no lee del registro, sino que selecciona el registro donde se va a escribir el dato. Como también selecciona uno de los 32 registros, también se necesita 5 bits para codificar todas las posibilidades.
4. **RegWR**: Esta entrada solo tendrá un bit ya que solo decidirá si se puede escribir o no en nuestro banco de registros.
5. **DATA**: Esta entrada estará compuesta de 32 bits ya que nuestra arquitectura será de 32 bits. En esta entrada se obtendrá los datos que queremos escribir en el banco de registros.
6. **CLK**: Esta entrada hace referencia al reloj y será de un bit ya que indicará la subida o bajada de flanco.
7. **RESET**: La entrada **Reset** sirve para poner a cero el banco de registros si se activa, por consiguiente, esta entrada será solo de un bit, igual que CLK.

SALIDAS:

El banco de registros tiene dos salidas:

1. **BusA**: Esta salida obtiene un dato del banco de registros por lo que debe ser de 32 bits.
2. **BusB**: Esta salida actúa igual que **BusA** obteniendo un dato del banco de registros por lo que también tiene que ser de 32 bits.

Teniendo claras las distintas entradas y salidas, vamos a diseñar en papel lo que más tarde será nuestro banco de registros:

Primero empezamos diseñando el BR4 con 4 registros, un puerto de escritura y dos de lectura:

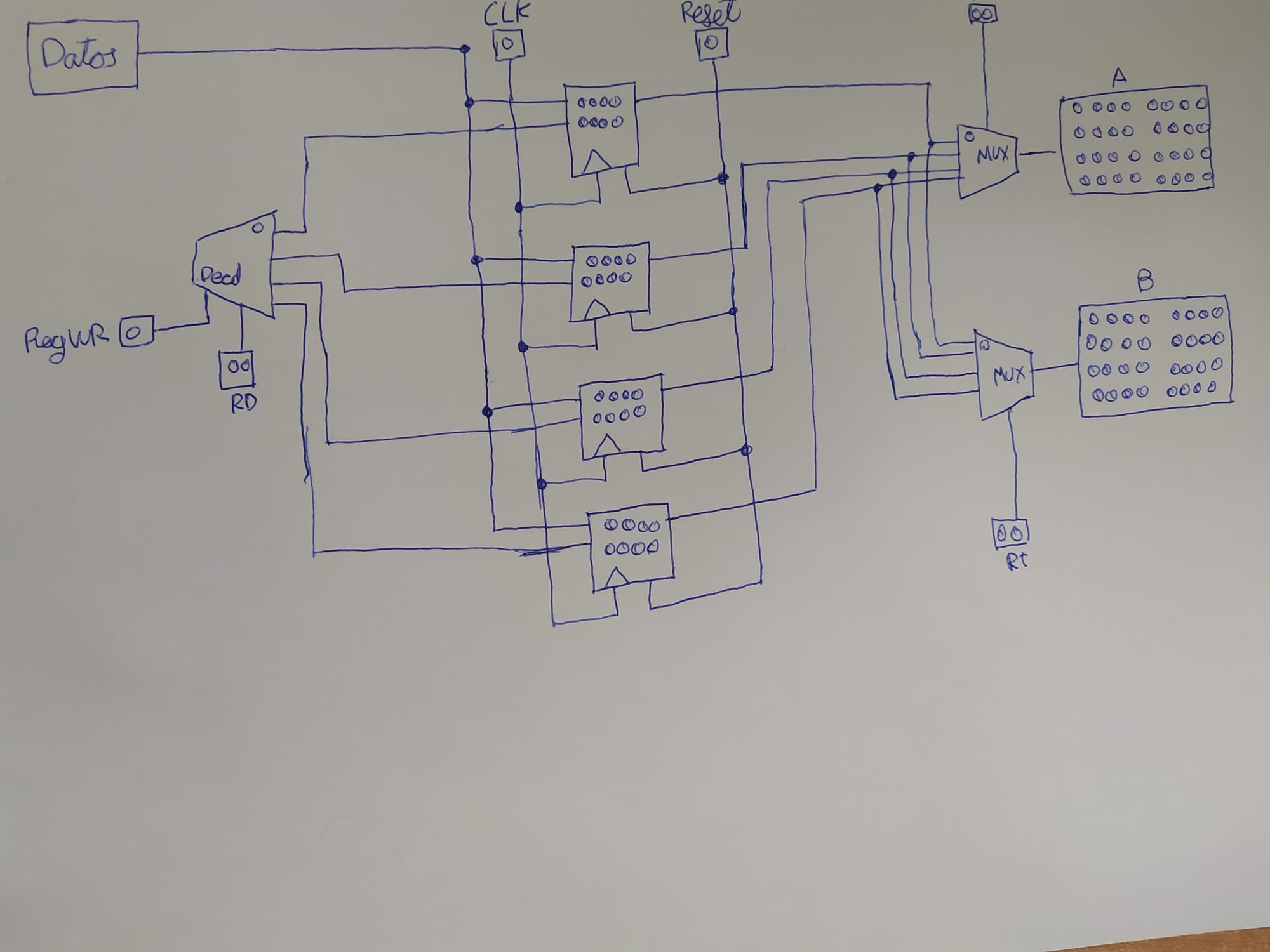


Ilustración : Diseño del BR4 sobre papel

Una vez tenemos una idea de como sería un BR4, debemos diseñar sobre papel también un BR32:

(AQUÍ FALTA TU MIERDA DE FOTO DEL BANCO DE REGISTROS TOCHO COMPADRE)

# 2. Diseño de un banco de registros en logisim

Una vez tenemos claras las codificaciones de las entradas-salidas del banco de registro y hemos escrito en papel como sería su estructura es hora de llevarlo a logisim:

1. Primero debemos hacer un pequeño banco de registros de 4 registros:

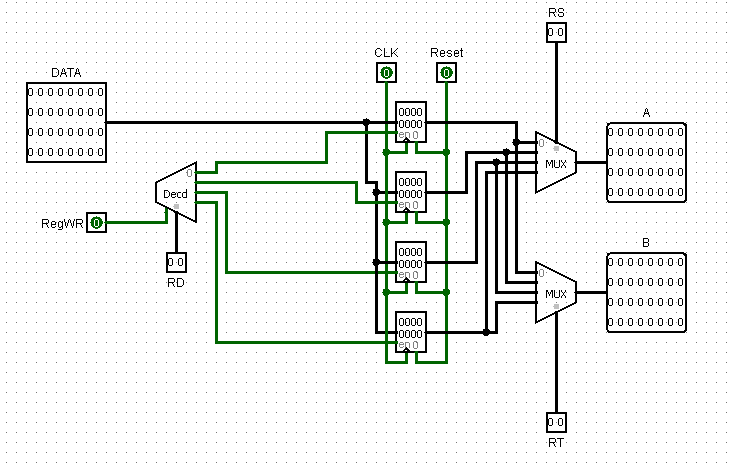


Ilustración : Bloque BR4 formado por 4 registros de 32 bits

Para ello necesitamos:

* **Un decodificador 2:4** para seleccionar que registro es el elegido para ser escrito.

La entrada regWr lo que hará es activar o desactivar este decodificador para que se pueda escribir o no en los registros. Con los otros bits de entrada (2 ya que debemos codificar 4 posibilidades) elegimos en que registro escribir.

* **Cuatro registros** **de 32 bits** para guardar la información.
* **Dos multiplexores 4:1** para seleccionar los datos de los 4 registros ya que tendremos dos puertos de lectura.

La selección del registro que se lee viene dada por los 2 bits más a la izquierda de RA (en el caso del puerto de lectura BusA) o RB (en el caso del puerto de lectura BusB).

* Las diferentes entradas para su correcto funcionamiento (Reset, CLK y DATA)

1. Encapsular este circuito nombrándolo BR4 y construir un BR32 con ellos:

Para ello cogemos 8 BR4 y los conectamos de la siguiente manera:

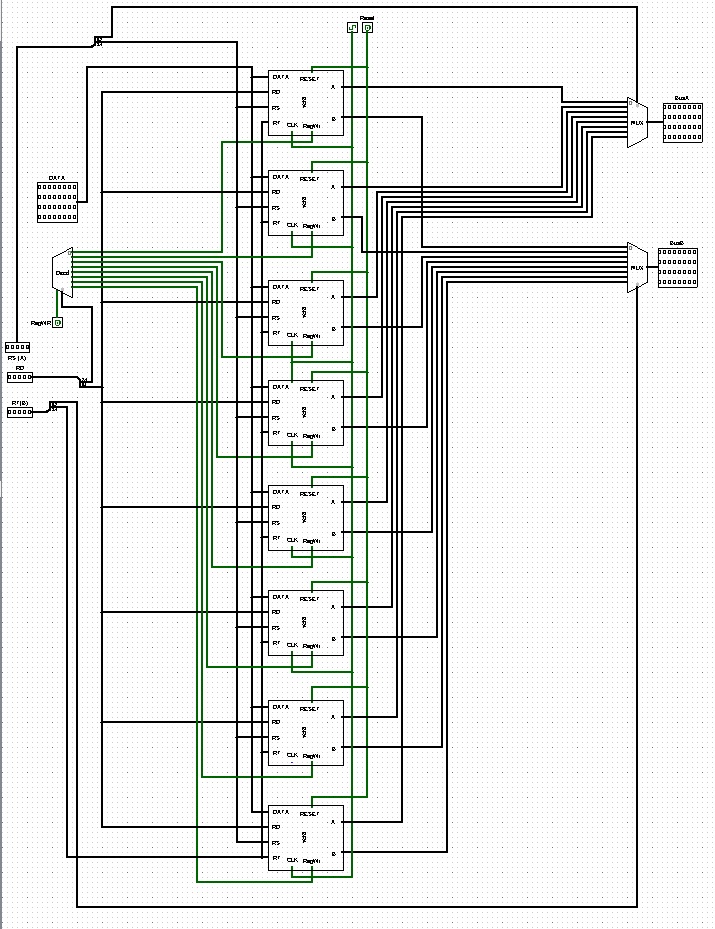


Ilustración : Banco de registros de 32 registros formado por 8 bloques BR4

Para conectar los 8 circuitos BR4 necesitamos varios componentes:

* **Un** **decodificador 3:8** para elegir que bloque de BR4 debemos elegir para la escritura (bloques de 4 registros cada uno).

Para que funcione correctamente la entrada RD de 5 bits se divide en dos entradas:

Una entrada de 3 bits que selecciona el bloque de BR4 correcto y una entrada de 2 bits que elige dentro de ese bloque, cual de los 4 registros es el indicado para escribir.

* **Dos multiplexores 8:1** para elegir que circuito de BR4 debemos escoger para la lectur ya que tenemos dos puertos de lectura.

Para que funcione correctamente la entrada RA o RB de 5 bits cada una se dividen en dos entradas:

Una entrada de 3 bits que selecciona el bloque de BR4 correcto y una entrada de 2 bits que elige dentro de ese bloque, cual de los 4 registros es el indicado para leer.

* **Ocho bloques BR4** para que haya 32 registros (8x4 registros).
* Las diferentes entradas para su correcto funcionamiento (Reset, CLK y DATA)

# 3. Analisis temporal

Primero vamos a hacer un análisis temporal de los circuitos combinacionales:

## 3.1 Multiplexor 4:1

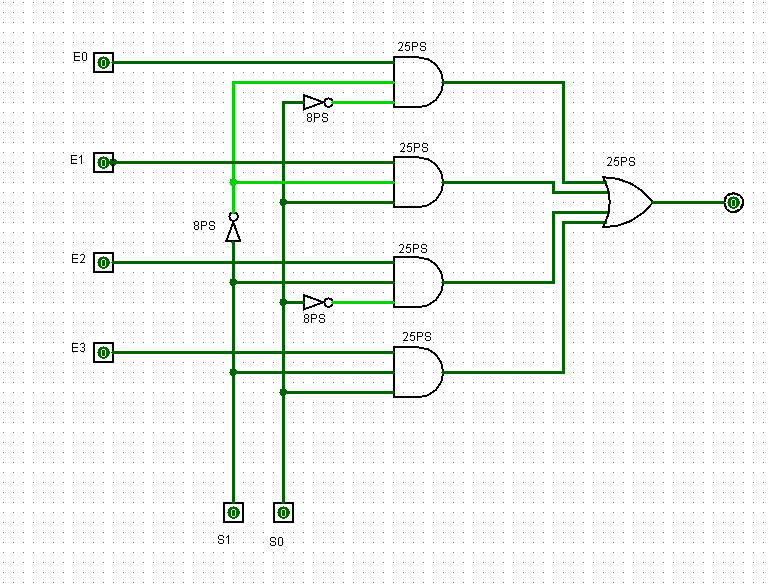


Ilustración : Multiplexor 4:1

Para calcular el retardo de este componente debemos prestar atención a dos caminos:

1. Desde el control a la salida: DelayNOT + DelayAND + DelayOR = 8ps + 25ps + 25ps = 58ps
2. Desde los datos a la salida: DelayAND + DelayOr = 25ps + 25ps = 50ps

Por consiguiente, el camino crítico es 58ps.

## 3.2 Multiplexor 8:1

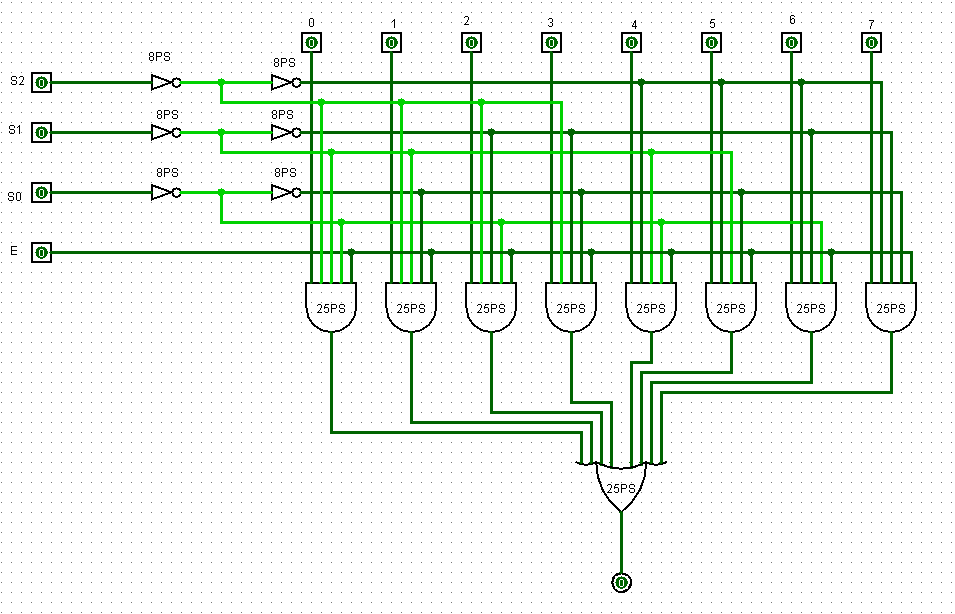


Ilustración : Multiplexor 8:1

Igual que en el caso anterior debemos diferenciar dos caminos:

1. Desde control a la salida: DelayNOT + DelayNOT + DelayAND + DelayOR = 8ps + 8ps + 25ps + 25ps = 66ps.
2. Desde datos a la salida: DelayAND + DelayOR = 25ps + 25ps = 50ps.

El camino crítico es 66ps.

## 3.3 Decodificador 2:4

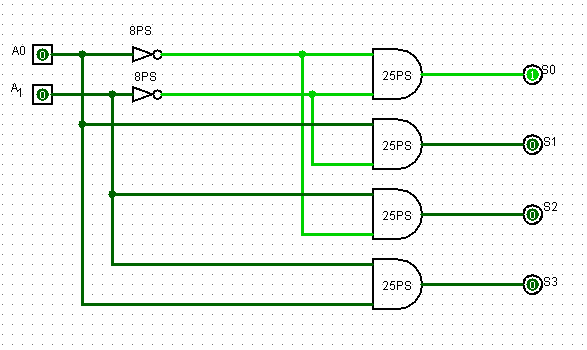
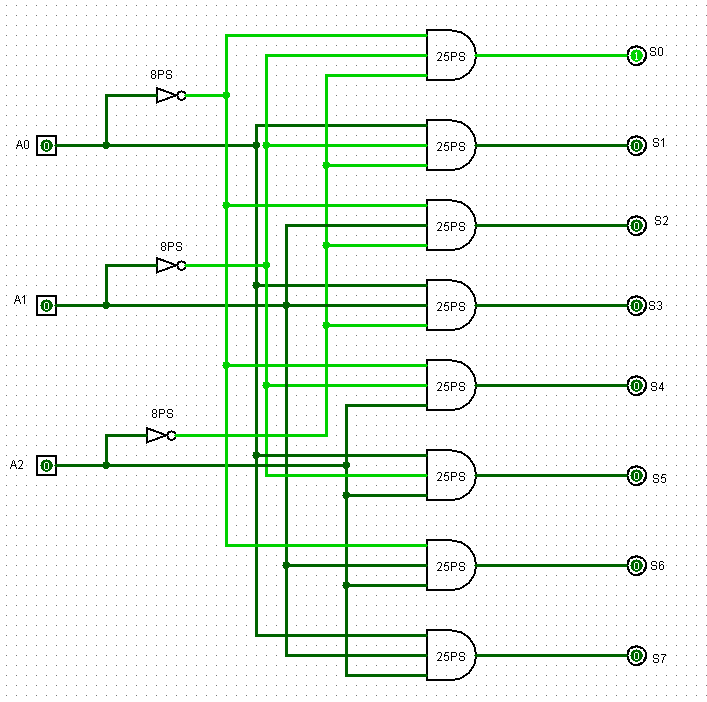


Ilustración : Decodificador 2:4

A diferencia de los multiplexores, los datos solo toman un camino:

Camino crítico: DelayNOT + DelayAND = 8ps + 25ps = 33ps

## 3.4 Decodificador 3:8



Igual que en el caso del decodificador 2:4, solo existe un camino:

Camino crítico: DelayNOT + DelayAND = 8ps + 25ps = 33ps

## 3.5 Análisis de los caminos combinacionales de la lectura

En la lectura del BR32 existen dos caminos que debemos tener en cuenta a la hora de leer un registro:

1. **De RS a BusA:** delay Registro + delay Multiplexor 4:1 desde control a salida + delay Multiplexor 8:1 desde control a salida = 60ps + 58ps + 66ps = 184ps
2. **De Reg a BusA:** delay Registro + delay Multiplexor 4:1 desde datos a salida + delay Multiplexor 8:1 desde datos a salida = 60ps + 50ps + 50ps = 160ps.

Por consiguiente, podemos decir que el delay del BR32 es de 184ps.

## 3.6 Análisis de los caminos combinacionales de la escritura

En el caso de la escritura en el BR32 debemos tener en cuenta 3 caminos:

1. **De RD a Reg:** delay Decodificador 3:8 + delay Decodificador 2:4 + tiempo setup Registro = 33ps + 33ps +35ps = 101ps
2. **De DATA a Reg:** Tiempo setup registro = 35ps
3. **De RegWr a Reg:** delay decodificador 3:8 + delay decodificador 2:4 + tiempo setup registro = 33ps + 33ps + 35ps = 101ps.

Por consiguiente, podemos decir que el tiempo de setup del BR32 es de 101ps.

# 4. Conclusión

Un banco de registros es una pieza clave e indiscutible para el correcto funcionamiento de cualquier procesador. Con esta práctica hemos entendido como funciona un banco de registros al detalle, cosa que más tarde usaremos para implementar un procesador.

# 5. Bibliografía

-Apuntes de la asignatura Arquitectura y organización de computadores 2 del grado de ingeniería informática.

-Estructura y diseño de computadores from David A.Patterson y Jhon L.Hennessy