

Universitat Politècnica de València

Escola Tècnica d'Enginyeria de Telecomunicació

Fundaments de VLSI

Memoria Trabajo Final

Diseño de Flip-Flop JK

Realizado por:

Jaime Lloret Cuñat

Adam Cecetka Ortiz



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



Índice

1. Introducción.....	4
2. Planteamiento del Flip Flop JK.....	5-7
2.1 Multiplexor 4-1.....	5
2.2 Flip Flop D.....	5-7
3. Estructura de ficheros.....	8
4. Puerta NOR.....	9-10
4.1 Diseño eléctrico.....	9
4.2 Símbolo.....	9
4.3 Layout.....	9
4.4 Simulación.....	10
5. Phase Splitter.....	10-12
5.1 Diseño.....	10
5.2 Símbolo.....	11
5.3 Layout.....	11-12
6. Multiplexor	12-15
6.1 Diseño.....	12
6.2 Símbolo.....	13
6.3 Layout.....	14
6.4 Simulación.....	15
7. SlaveD.....	16-19
7.1 Diseño.....	16
7.2 Símbolo.....	16
7.3 Layout.....	17
7.4 Simulación.....	18-19
8. MasterD	20-23

8.1 Diseño.....	20
8.2 Símbolo.....	21
8.3 Layout.....	21-22
8.4 Simulación.....	23
9. Flip Flop D.....	24-27
9.1 Diseño.....	24
9.2 Símbolo.....	24
9.3 Layout.....	25-26
9.4 Simulación.....	27
10. Flip Flop JK.....	28-32
10.1 Diseño.....	28
10.2 Símbolo.....	28
10.3 Layout.....	29-30
10.4 Simulación.....	31-32
11. Especificaciones técnicas.....	32-35
11.1 Frecuencia máxima.....	32-34
11.2 Potencia media.....	34-35
11.3 Retardo Layout-Esquemático.....	35
12. Análisis Crítico de los resultados.....	36
12.1 Conclusiones.....	36
12.2 Otras formas de hacerlo.....	36
13. Bibliografía.....	36

1. Introducción

El Flip-flop JK es un dispositivo de almacenamiento biestable, esencial en la electrónica digital. Este tipo de Flip-flop puede cambiar su estado, mantenerlo o invertirlo, dependiendo de las entradas de control J y K. Es ampliamente utilizado en aplicaciones que requieren almacenamiento temporal de datos, como en sistemas de memoria, contadores y registros de desplazamiento. Su diseño permite que sea utilizado en configuraciones más complejas, facilitando la construcción de circuitos lógicos secuenciales avanzados. A continuación, observamos su funcionamiento:

Truth Table			
J	K	CLK	Q
0	0	↑	Q_0 (no change)
1	0	↑	1
0	1	↑	0
1	1	↑	\overline{Q}_0 (toggles)

Figura 1: Tabla verdad Flip-flop JK

Para nuestro diseño emplearemos un multiplexor y un Flip-flop tipo D activo en flanco de subida. A la hora de plantear el diseño nos damos cuenta de que el multiplexor va a ser un 4 a 1 ya que dependiendo de las entradas J y K vamos a tener a la entrada del Flip-flop o un cero, o un uno, o la salida anterior o la salida anterior negada. Es decir, claramente J y K van a ser las entradas de selección y las entradas del mux serán vdd, gnd, la salida del Flip-flop D y la salida del Flip-flop D negada.

Para el diseño de nuestro Flip-flop tipo D vamos a emplear la arquitectura Master-Slave ya que así nos puede ser de inspiración la práctica 4. Cabe recalcar que nuestro Flip-flop tipo D debe tener una entrada de reset asíncrona y activa a nivel bajo por lo que habrá que pensar como implementar esto.

2. Planteamiento Inicial

2.1 Multiplexor

Para el diseño del multiplexor 4 a 1 partiremos de la base de un diseño muy típico mediante puertas de transmisión.

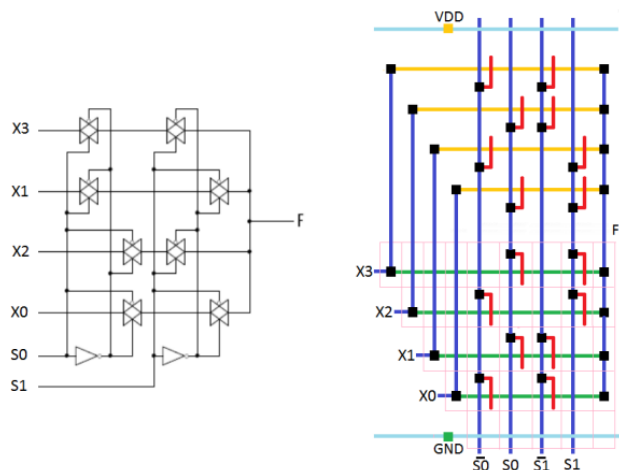


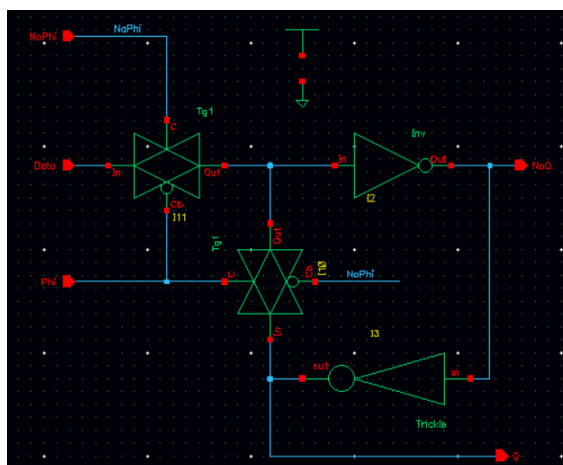
Figura 2: Stick-diagram y esquemático MUX 4:1

Observamos como necesitaremos algún tipo de inversor para obtener las entradas de selección deseadas. Es por ello por lo que nos decidimos a reutilizar el Phase-Splitter de la práctica 4 por la ubicación adyacente entre entradas negadas y sin negar.

Sin embargo, si queremos poner dos Phase-Splitter en espejo (que es para lo que los diseñamos), la señal S1 se debe intercambiar por la S1 negada lo cual hay que tener en cuenta ya que eso puede dar futuros mismatches en la comparación esquemático-layout.

2.2 Flip-flop D

Como ya hemos comentado en la introducción el diseño se hará mediante la arquitectura Master-Slave. Es importante darse cuenta de que necesitamos un reset asíncrono activo a nivel bajo por lo que decidimos modificar el Master-Slave de la práctica cuatro para



obtener el funcionamiento buscado. Además, inspirándonos en el diseño de esa práctica habrá que realizar las modificaciones pertinentes para lograr que el Flip-flop sea activo en flanco de subida. Para lograr un reset nos empezaremos centrando en el Slave. A continuación, observamos el diseño eléctrico empleado en la práctica 4. (figura 3)

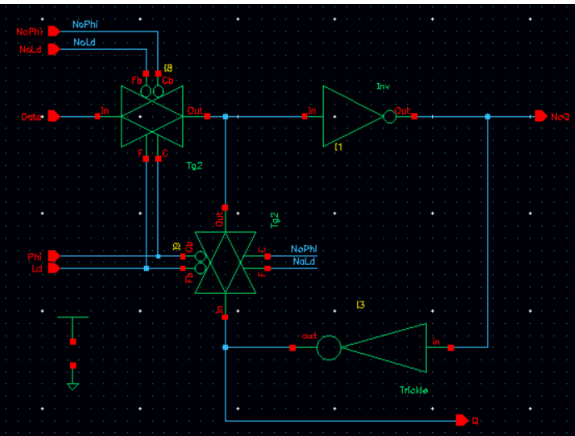
Figura 3: Diseño eléctrico slave práctica 4

El Slave tiene que sacar un cero al instante una vez activo el RSTa por lo que nos dimos cuenta de que para ello el inversor de la parte superior convendría sustituirlo por una puerta distinta. Procedemos realizando una pequeña tabla de la verdad para ver por dónde van los tiros. Recordemos que NoQ es la salida del Flip-flop. Llamamos Data' a la salida de la puerta de transmisión con entrada Data.

Rsta	Data'	NoQ
0	0	0
0	1	0
1	0	1
1	1	0

Observando la tabla nos damos cuenta de que no es más que un inversor seguido de una puerta NOR por lo que tendremos que implementar esta puerta NOR en el layout del Slave mientras que el inversor lo pondremos fuera ya que simplemente convierte el RST de activo nivel alto a bajo.

Con respecto al Master también será necesario realizar algún cambio.



Para el Master, en el caso de activar el RSTa queremos que el cero de la salida se mantenga. Observando su diseño eléctrico de la práctica nos damos cuenta de que a diferencia del Slave ahora habrá que sustituir el inversor de abajo en lugar del de arriba. Los valores que queremos resultan ser los mismos que en el Slave por lo que pasará lo mismo, necesitaremos también un inversor seguido de la puerta NOR.

Figura 4: Diseño eléctrico Master práctica 4

Además, para el caso del Master es necesario retirar las entradas de Load y NoLoad ya que en nuestro dieño no queremos enable por lo que estas puertas de transmisión también tendrán que ser modificadas. Por supuesto para Master y Slave habrá que pensar como situar Phi y NoPhi para obtener un Flip-flop activo en flanco de subida.

La dificultad del trabajo está en como implementar esta puerta NOR en el layout por lo que para ello será necesario realizar un “stick diagram” del Master y otro para el Slave. La parte del Master y Slave de las puertas de transmisión la mantendremos igual que en la práctica cuatro a excepción del Master que en nuestro caso no tenemos ya entrada de Enable por lo que bastará con hacer un “Chop” para retirar esa parte. Por lo tanto, simplemente realizamos el stick diagram de la mitad derecha del layout, que es la que más dificultad va a entrañar. Destacar que por hacerlo más sencillo la salida de la puerta de transmisión con la entrada Data ha sido denominada Data'. A continuación, los observamos: (figura 5 y 6)

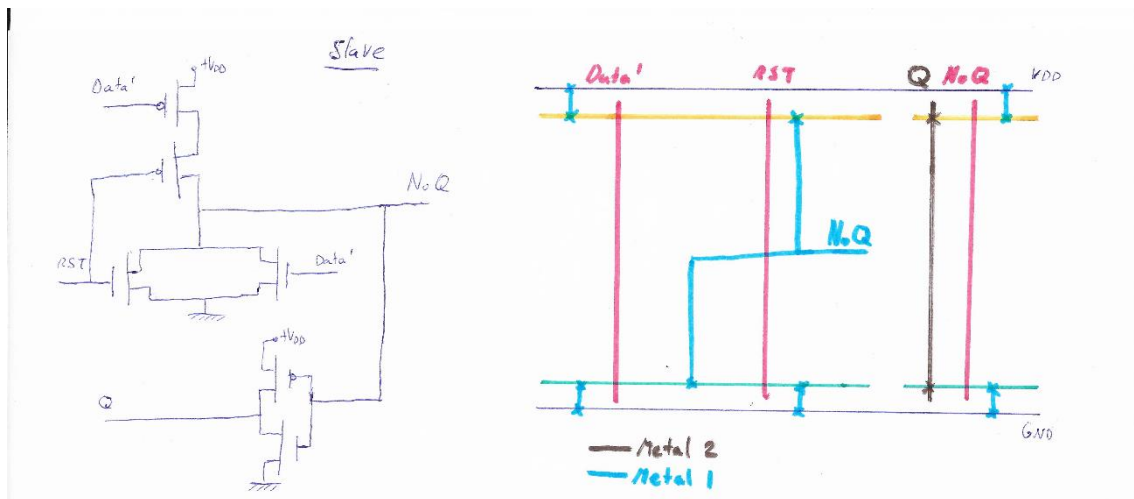


Figura 5: Stick Diagram mitad derecha Slave

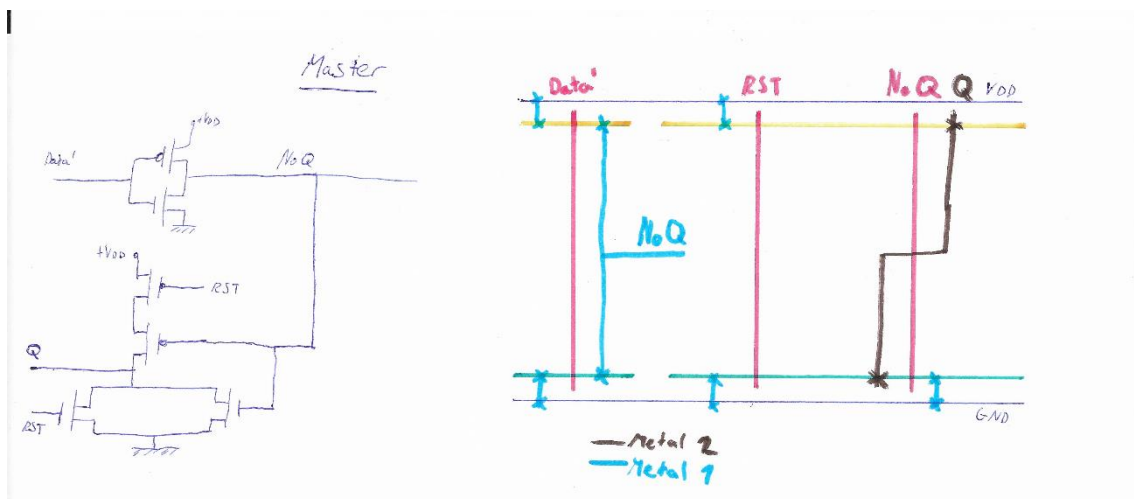


Figura 6: Stick Diagram mitad derecha Master

Con esto ya diseñado podemos ponernos manos a la obra. Recordemos que nuestro diseño será jerárquico por lo que comenzaremos diseñando el Master y luego el Slave para más tarde poder unirlos con los Phase-splitters y los inversores y así tener el Flip-Flop tipo D diseñado.

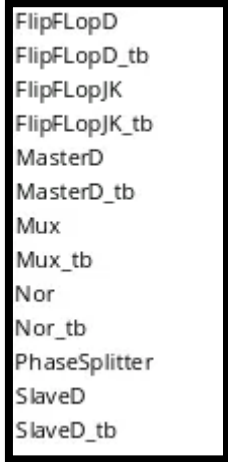
Ya por último solo faltaría unirlo al multiplexor para obtener nuestro Flip-flop JK.

Iremos diseñando los módulos de manera que haremos primero su esquemático, después su layout y finalmente una simulación comparativa entre ambos módulos.

3. Estructura de ficheros

Estos son todos los módulos que podemos encontrar en nuestra librería “Proyecto”

- Nor
 - Schematic
 - Symbol
- Nor_tb
 - Schematic
 - State: “nortb”
- Mux
 - Layout
 - Schematic
 - Av-extracted
 - Symbol
- Mux_tb
 - Config
 - Schematic
 - State: “muxtb”
- FlipFlopD
 - Layout
 - Schematic
 - Av-extracted
 - Symbol
- FlipFlopD_tb
 - Config
 - Schematic
 - State: “flipflopdtb”
- MasterD
 - Layout
 - Schematic
 - Av-extracted
 - Symbol
- MasterD_tb
 - Config
 - Schematic
 - State: “masterdtb”
- SlaveD
 - Layout
 - Schematic
 - Av-extracted
 - Symbol
- SlaveD_tb
 - Config
 - Schematic
 - State: “slavetb”
- PhaseSplitter
 - Layout
 - Schematic
 - Av-extracted
 - Symbol
- FlipFlopJK
 - Layout
 - Schematic
 - Av-extracted
 - Symbol
- FlipFlopJK_tb
 - Config
 - Schematic
 - State: “flipflopjktb”



FlipFlopD
FlipFlopD_tb
FlipFlopJK
FlipFlopJK_tb
MasterD
MasterD_tb
Mux
Mux_tb
Nor
Nor_tb
PhaseSplitter
SlaveD
SlaveD_tb

Figura 7: Imagen librería

4. Puerta NOR

Para la puerta NOR comenzaremos diseñando su diseño eléctrico. No hará falta hacer su layout por separado ya que la NOR irá integrada directamente en el Master y el Slave mediante el diseño previo del “Stick Diagram”.

Observamos su diseño eléctrico y su símbolo:

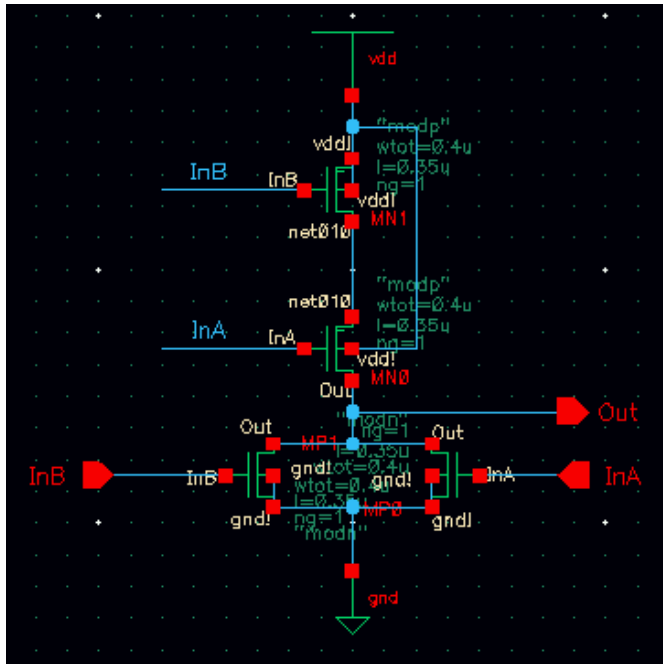


Figura 8: Esquemático de la puerta NOR

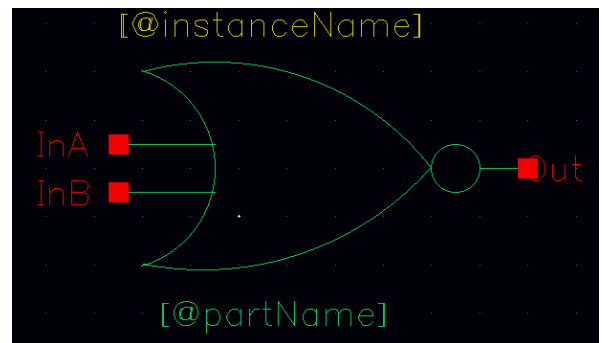


Figura 9: Símbolo de la puerta NOR

Posteriormente continuamos simulando nuestro esquemático:

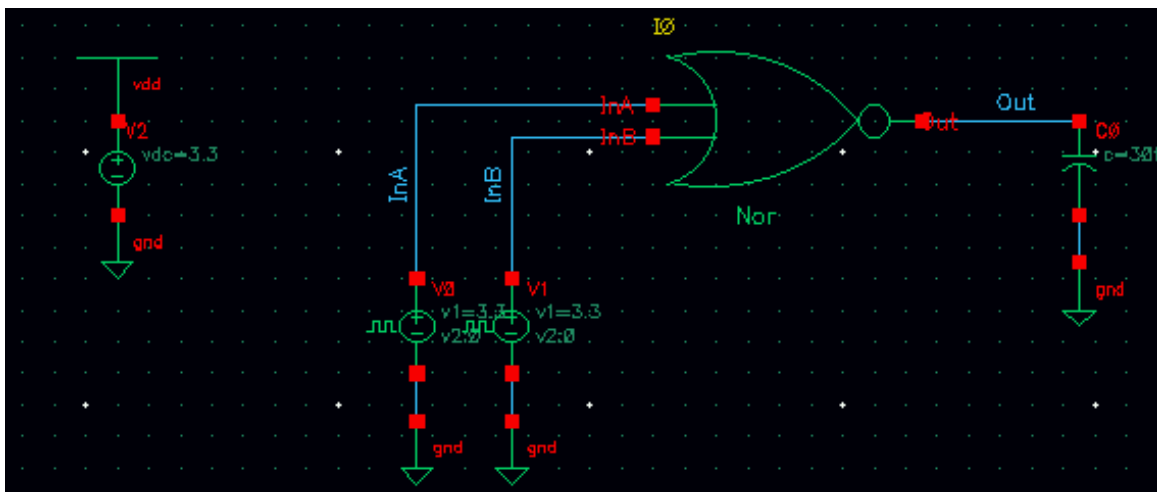


Figura 10: Esquemático del módulo 'NOR_tb'

Así pues, esto fue lo que obtuvimos al simular: (Figura 11)

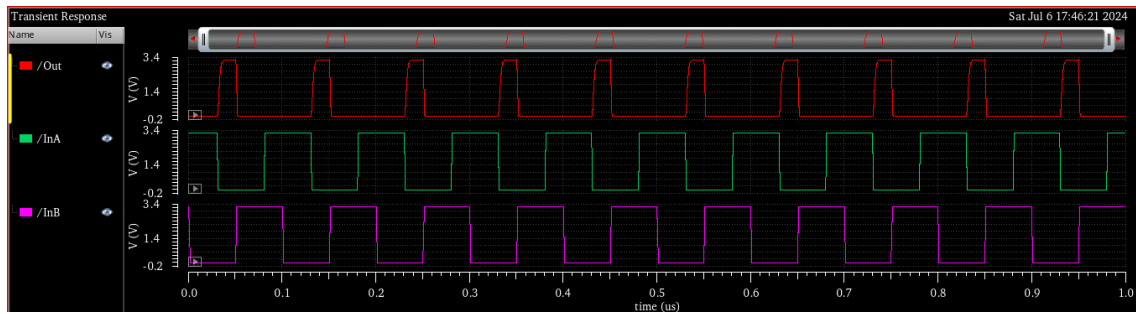


Figura 11: Waveform de la puerta NOR

Puerta: NOR		
Entradas		Salida
A	B	S
0	0	1
0	1	0
1	0	0
1	1	0

Como podemos comprobar en la simulación, la salida Out solo se activaba cuando ambas entradas valen cero. Como se puede comprobar en su respectiva tabla de verdad (figura 12), los resultados de la simulación son correctos.

Figura 12: Tabla de verdad de la puerta NOR.

5. Phase Splitter

Para continuar con el diseño y tener las señales adecuadas en el mux y flip-flop es necesario el phase-splitter. Podemos recordar cómo es su diseño. (figura 13)

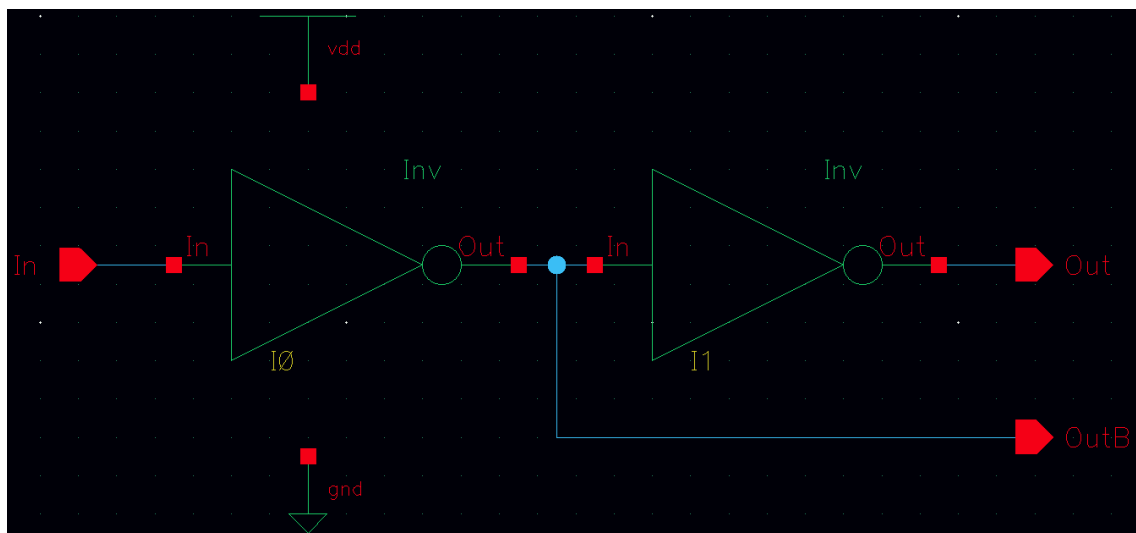


Figura 13: Esquemático del PhaseSplitter

El esquemático es muy simple, consiste en una entrada, dos inversores y dos salidas negadas.

La figura 14 es el símbolo del Phase Splitter.

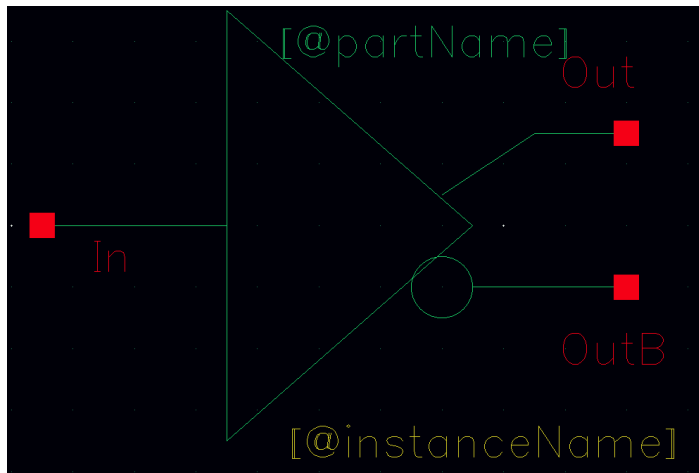
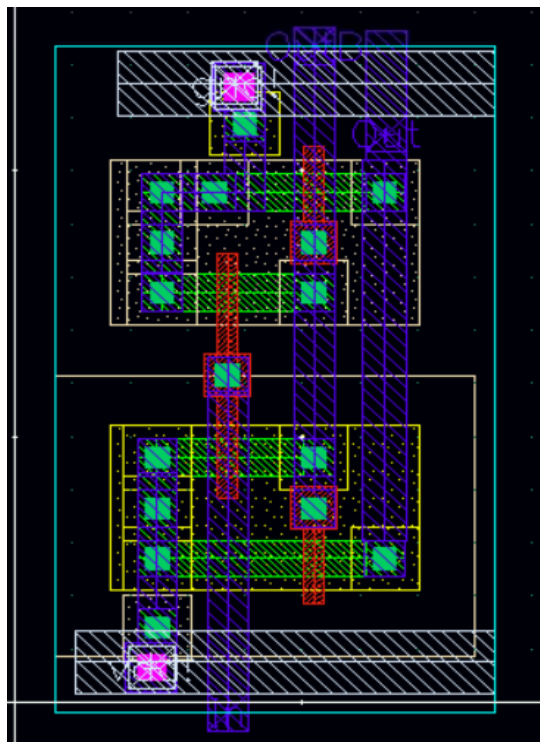


Figura 14: Símbolo del Phase Splitter

Además, observamos su layout a continuación: (figura 15)



Se observa que se trata de un layout bastante sencillo. Como de costumbre posteriormente se prosiguió con el chequeo de reglas de diseño (figura 16) y la comparación Layout-Esquemático (figura 17). Seguidamente se extrajo su vista “av_extracted” para la simulación del módulo.

Figura 15: Layout del Phase Splitter



Figura 16: Asura DRC del Phase Splitter

```

-----
DIO avD43_1

1 devices filtered

The LVS run "PhaseSplitter" has completed successfully.

Compare problems were detected in 1 cells.
  1 cells had parameters mismatches.
  0 cells matched

No Extraction Problems were detected.

You currently have an open run (project).
Press "OK" to close this run and enter the LVS Debug Environment.
Press "Cancel" to leave this run open and close this Dialog box.

LVS Run "PhaseSplitter"
is located in /home/jllocua/E1/ASSURA_LVS/PhaseSplitter

```

Figura 17: Asura LVS del Phase Splitter

Observamos 1 parameter mismatch lo cual es normal ya que no todas las dimensiones de los transistores cuadran a la perfección. Este tipo de mismatch lo observaremos a lo largo de todo nuestro diseño en distintos módulos, pero no hay de que preocuparse.

Cabe mencionar, que no nos ha sido necesario realizar una simulación dedicada al Phase Splitter ya que lo reciclamos de la práctica 4 que ya nos funcionaba correctamente.

6. Multiplexor

Una vez hecha la puerta NOR, diseñamos el multiplexor 4:1, (figura 18) con 8 puertas de transmisión y 2 PhaseSplitters necesarios para obtener las entradas de selección negadas.

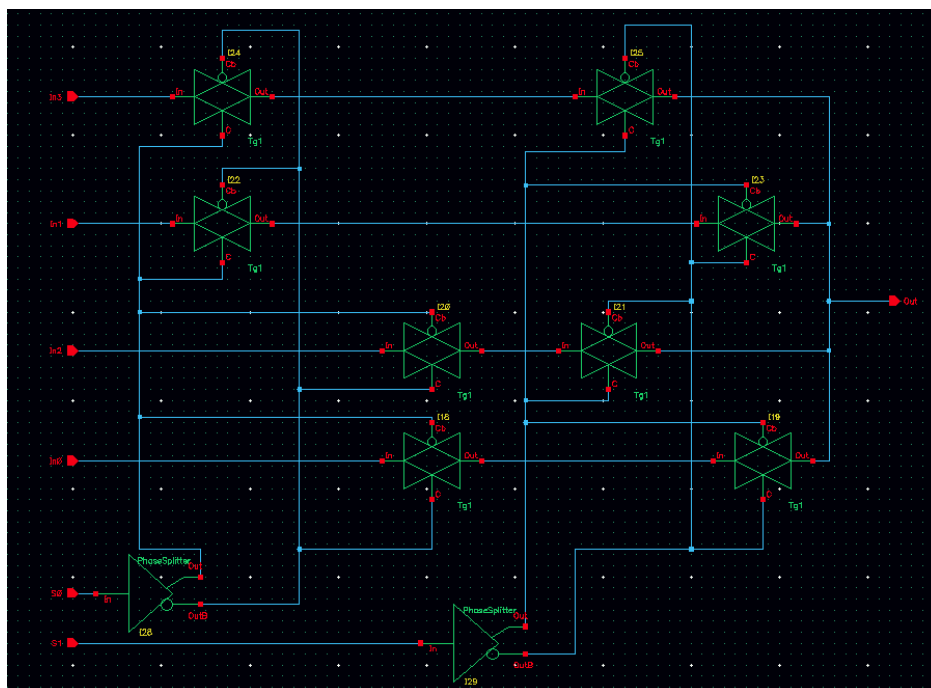


Figura 18: Esquemático del MUX 4:1

Hicimos también el símbolo: (figura 19)

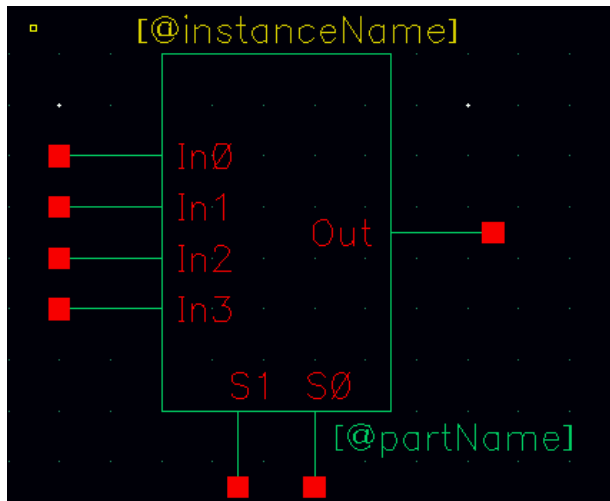


Figura 19: Símbolo del MUX 4:1

Antes de hacer el layout implementamos un nuevo módulo llamado ‘Mux_tb’ para realizar una simulación del diseño eléctrico y para más adelante compararlo con la simulación del layout. (figura 20)

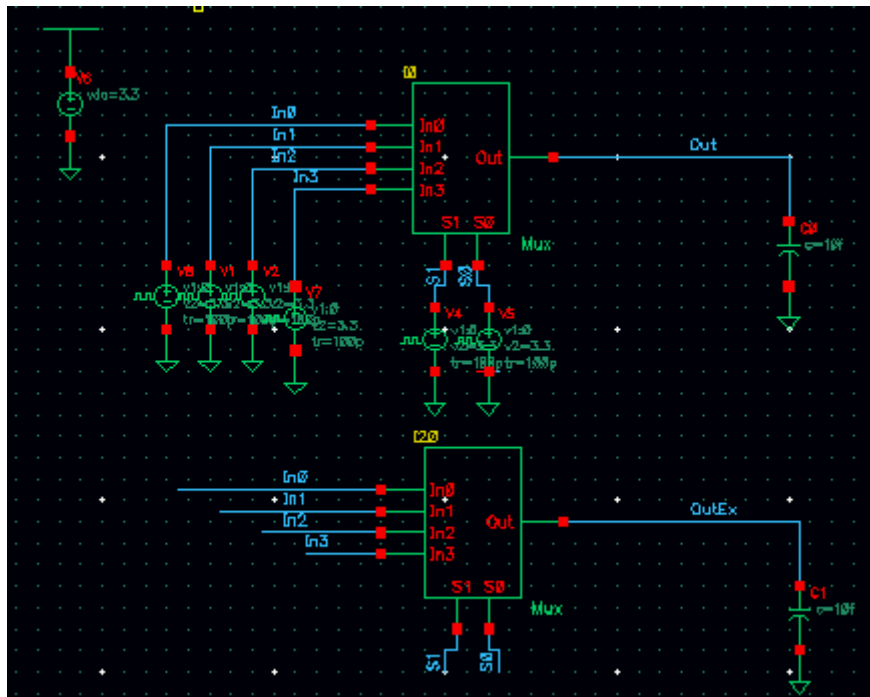


Figura 20: Esquemático de ‘Mux_tb’

Simultáneamente estuvimos haciendo el layout del multiplexor basándonos en el *stick diagram* mostrado en el planteamiento inicial. Se pueden apreciar los dos PhaseSplitter ubicados en la parte inferior del layout para conseguir las entradas negadas requeridas. (figura 21). Las anchuras empleadas han sido las mínimas igual que para el Master y el Slave y además hemos tratado de hacer el diseño lo más compacto posible cumpliendo con las dimensiones exactas que nos permitiesen pegar los transistores al máximo el uno del otro sin violar las reglas de diseño.

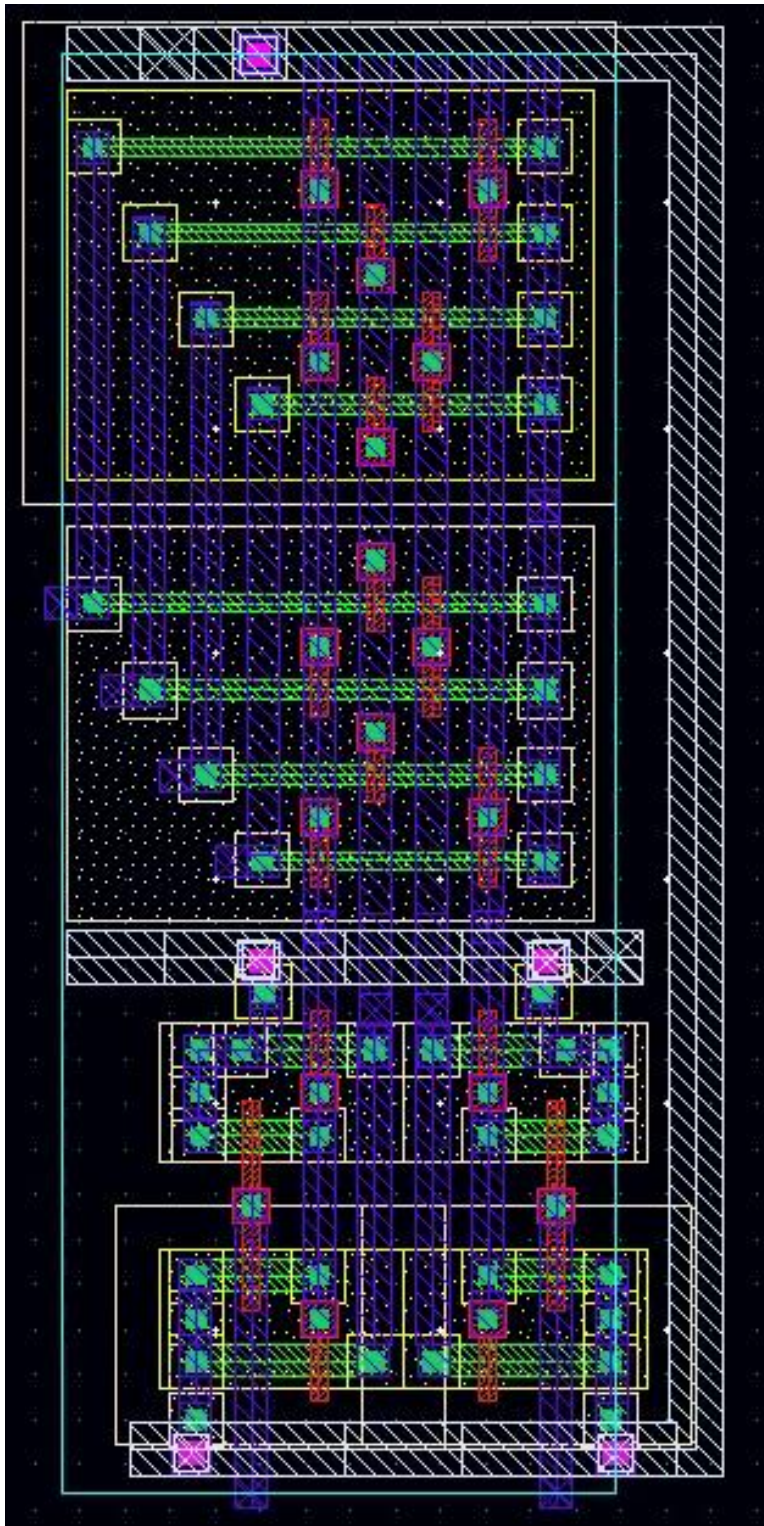


Figura 21: Layout del MUX 4:1

Una vez acabado el layout comprobamos las reglas de diseño e hicimos la comparación Layout-esquemático. (figura 22 y 23). Mas tarde generamos el av_extracted y configuramos los estímulos en el esquemático de la simulación (figura 20) para poder iniciarla con el objetivo de comparar con el esquemático y comprobar que todo funcionara correctamente. Obtuvimos los siguientes resultados: (figura 24).

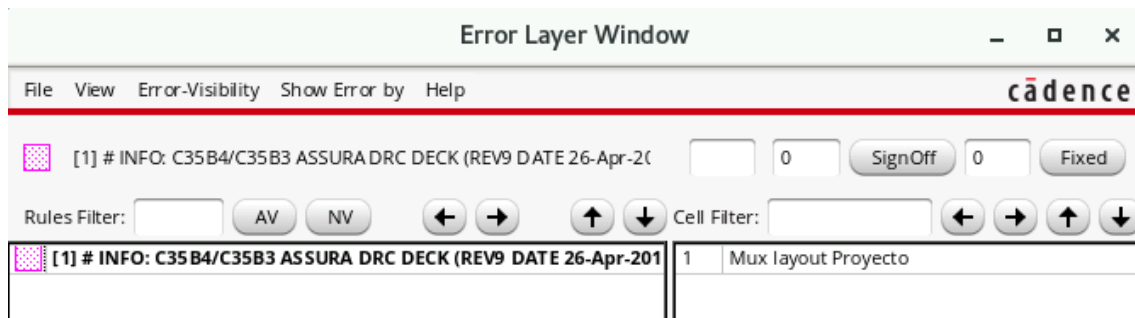


Figura 22: Asura DRC del MUX 4:1



Figura 23: Asura LVS del MUX 4:1

Observamos un DRC satisfactorio. Por el contrario, en el LVS observamos algunos fallos un poco extraños. Sin embargo, esos mismatches enseguida tienen sentido si recordamos que en el planteamiento inicial hemos decidido que las dos últimas columnas del layout íbamos a intercambiarlas para poder cuadrar bien la colocación en espejo de los Phase-splitters.

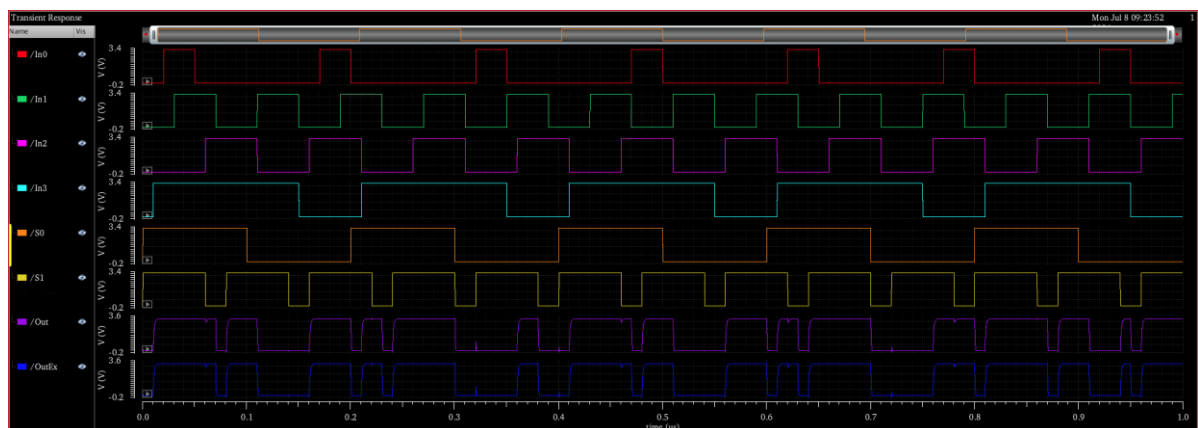


Figura 24: Resultados de la simulación del MUX 4:1

Podemos observar cómo coinciden las señales del Out correspondiente al esquemático y OutEx que proviene del layout y además se obtiene la lógica esperada.

7. SlaveD

Procedemos llevando el Stick Diagram del Slave a la realidad. A continuación, observamos su esquemático, símbolo, layout y la simulación comparativa entre ambos realizada con el archivo “config”.

En primero lugar empezamos haciendo el esquemático: (figura 25)

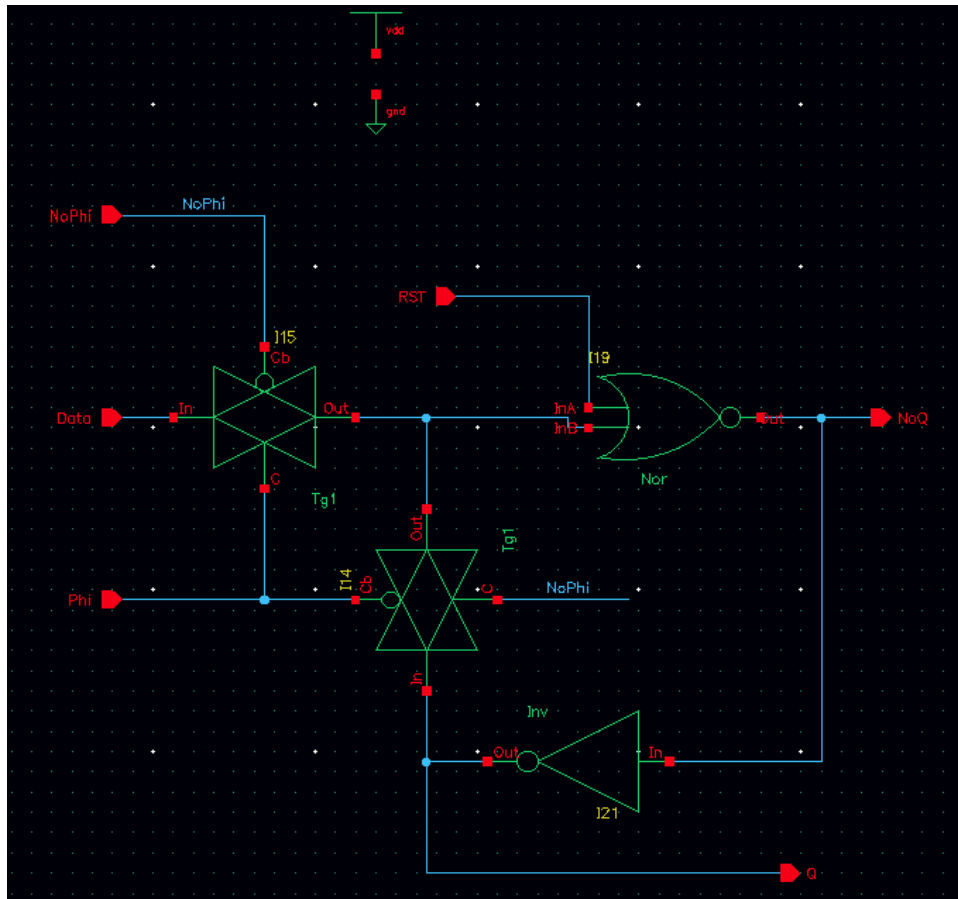


Figura 25: Esquemático del SlaveD

Luego diseñamos el siguiente símbolo: (figura 26)

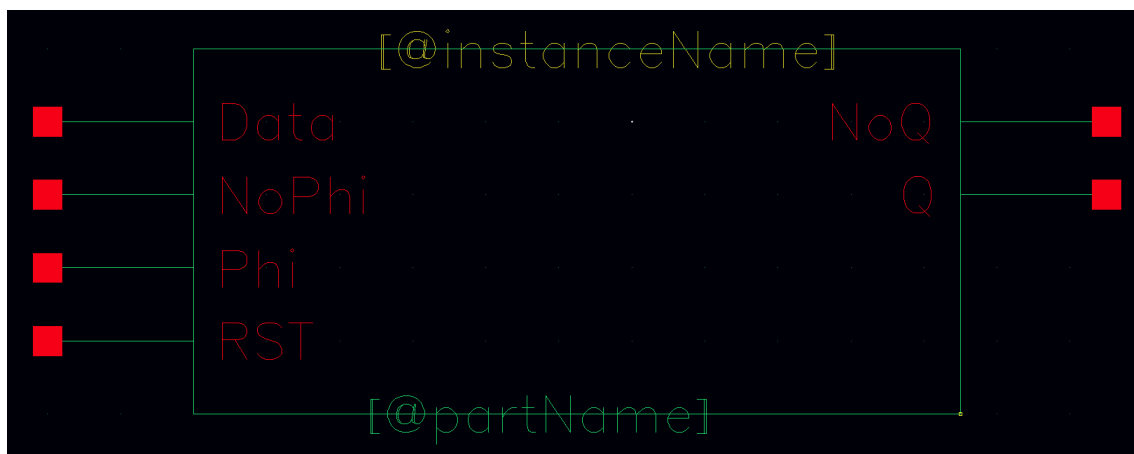


Figura 26: Símbolo del SlaveD

Posteriormente diseñamos el Layout siguiendo los *stick-diagrams* y cumpliendo las reglas de diseño (figura 27)

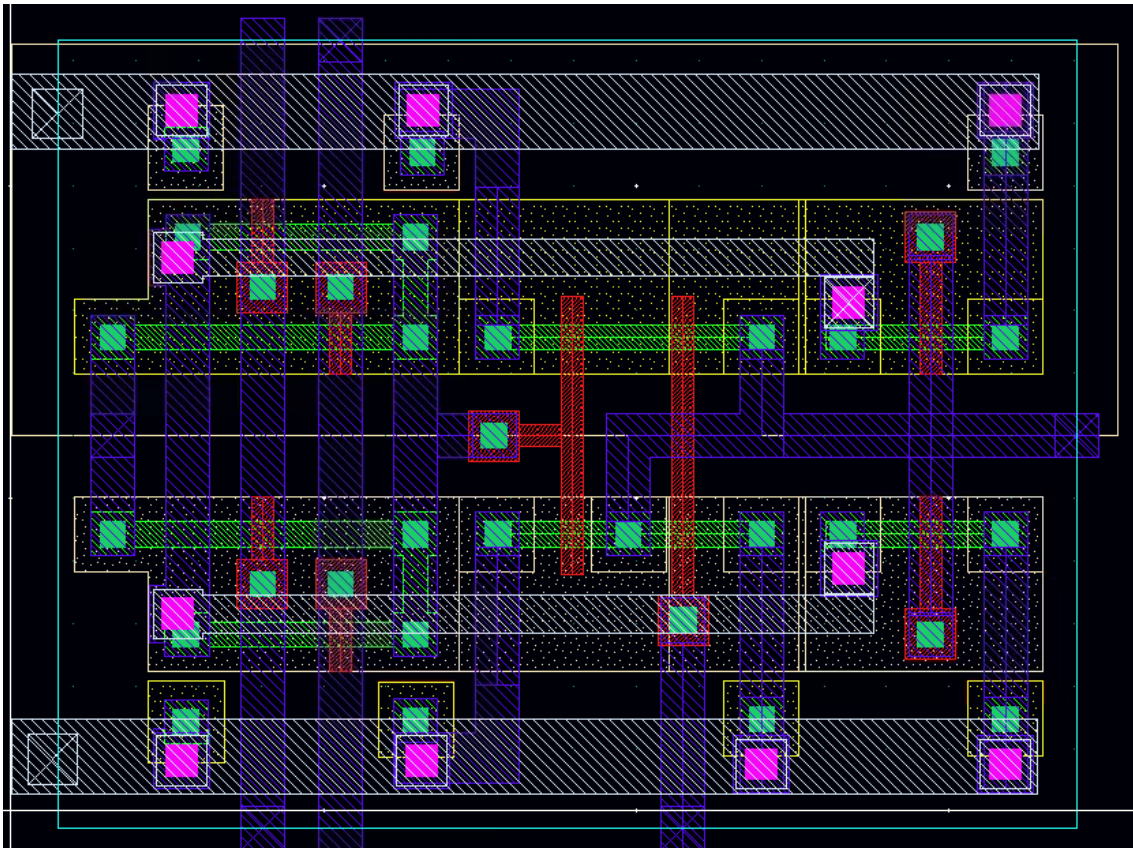


Figura 27: Layout del SlaveD

Aquí podemos observar la implementación de la puerta lógica NOR en la parte central del diseño seguida de un inversor en el lado derecho. Una vez más hemos empleado anchuras mínimas para los transistores y nos hemos asegurado de que este latch sea transparente en nivel alto, para luego hacer el Master transparente a nivel bajo y así obtener la activación por flanco de subida.

Posteriormente hicimos las comprobaciones de DRC y LVS (figura 28 y 29).

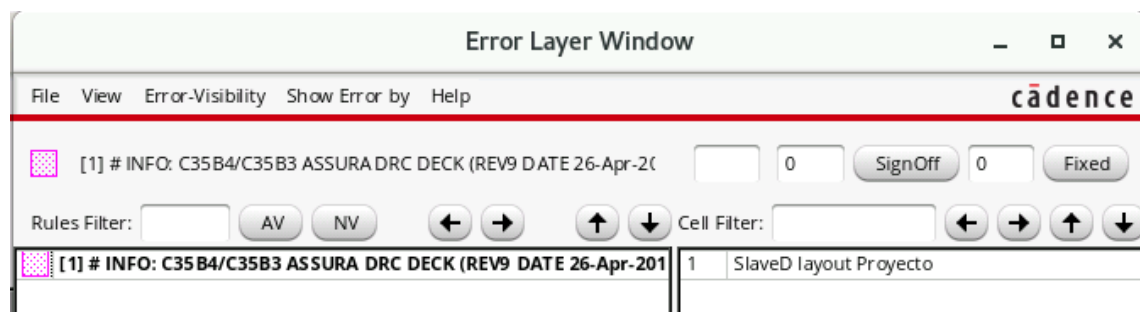


Figura 28: Asura DRC del SlaveD

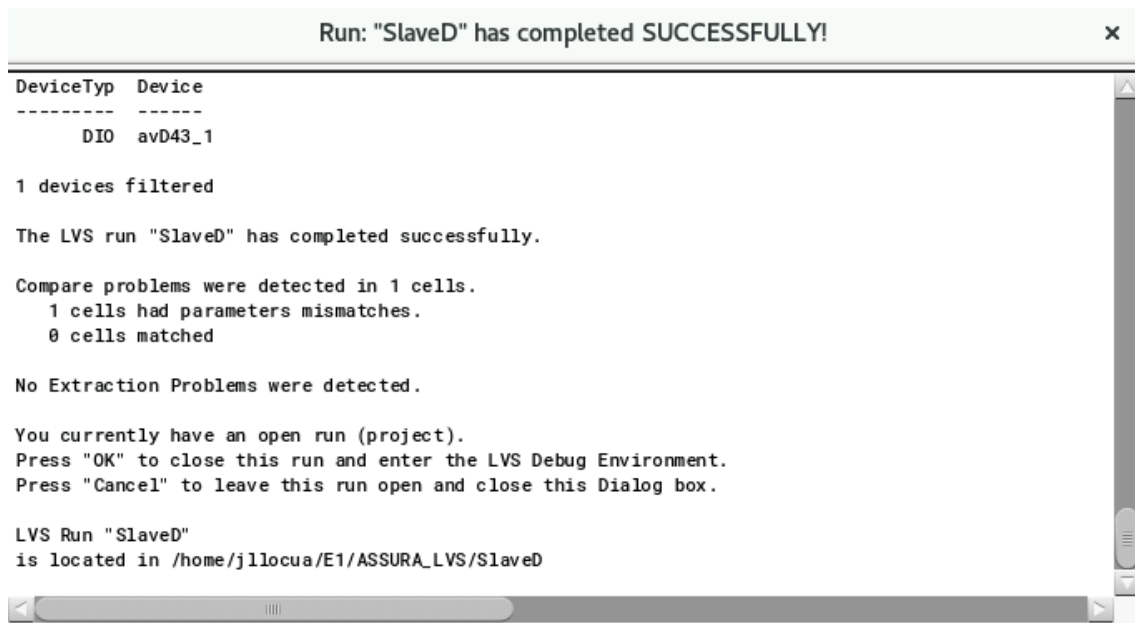


Figura 29: Asura LVS del SlaveD

Se puede apreciar claramente como tanto las reglas de diseño como el LVS se encuentran en orden.

De nuevo, planteamos la simulación en un nuevo módulo llamado 'SlaveD_tb' (figura 30):

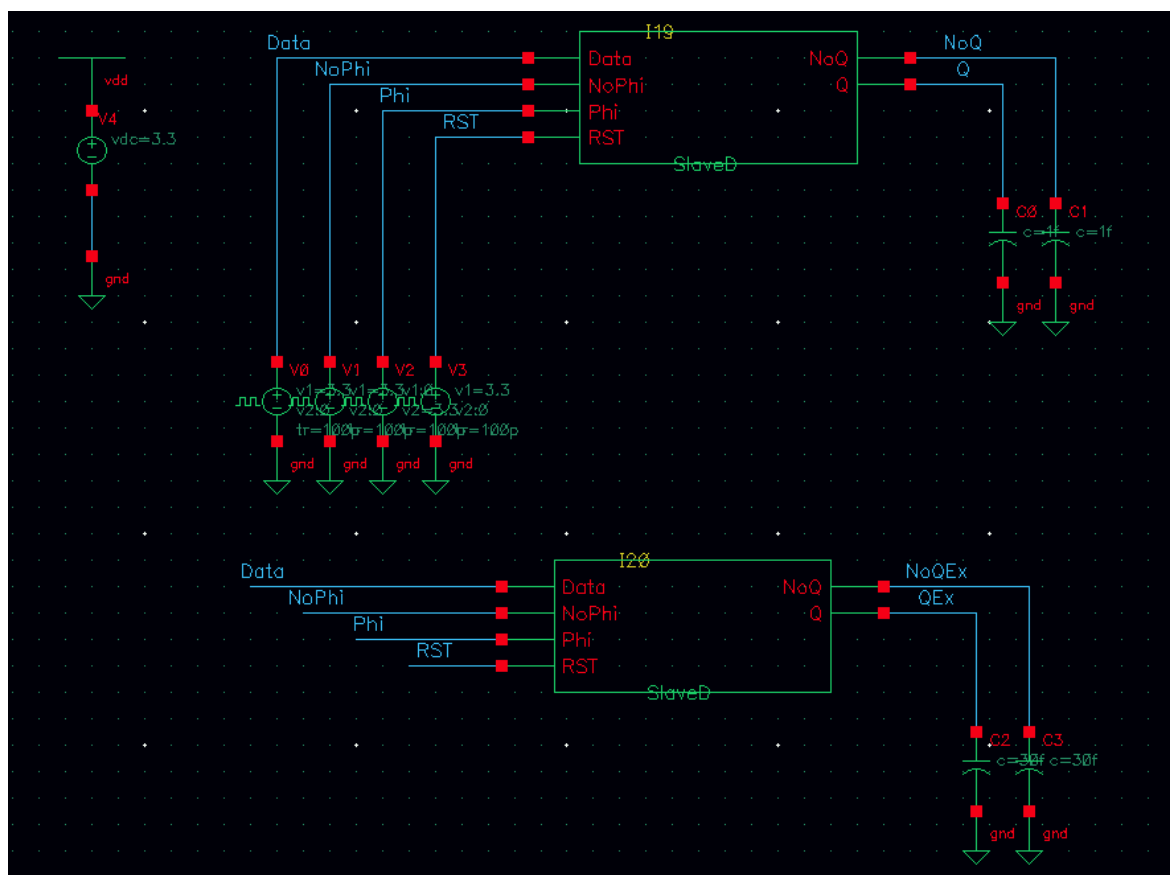


Figura 30: Esquemático de 'SlaveD_tb'

Gracias a este pudimos realizar una simulación comparativa del Layout y del esquemático configurado con el archivo “config”, obteniendo los siguientes resultados: (figura 31)

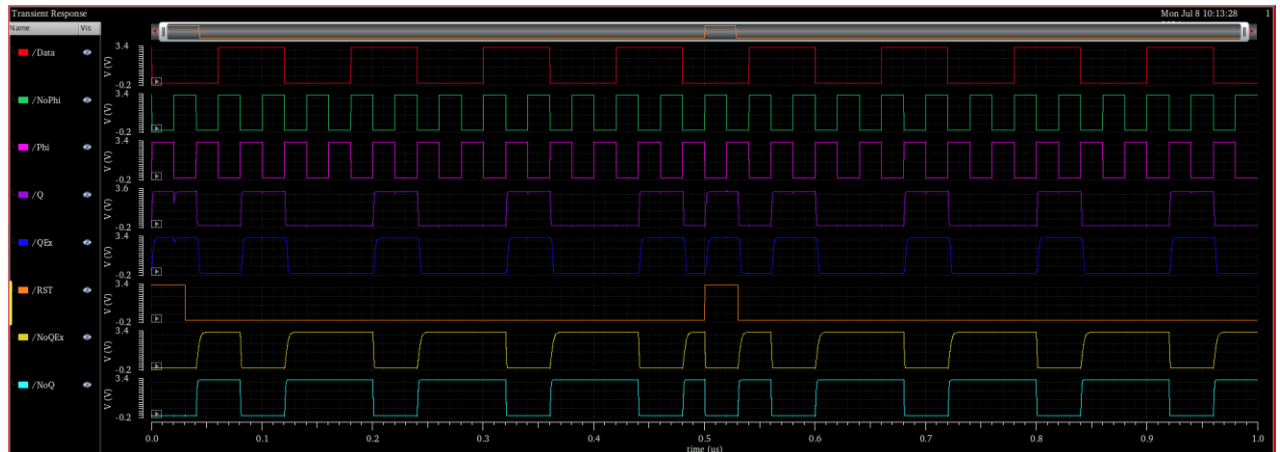


Figura 31: Waveform de la simulación de *SlaveD_tb*

Aquí se ve claramente como coinciden las señales del esquemático y del layout en la salida ‘Q’ y su inversa ‘NoQ’, que para las señales del layout tienen el acrónimo de ‘QEx’ y ‘NoQEx’.

Se observa como el RST (el cual en este caso es activo a nivel alto) resetea de forma adecuada la salida NoQ, que es la futura salida del Flip-flop JK, además de que observamos claramente como la salida Q del latch es transparente con “Phi” a nivel alto.

8. MasterD

Realizamos lo mismo con el Master. A continuación, observamos su esquemático, símbolo, layout y la simulación comparativa entre ambos realizada con el archivo “config”.

Como siempre, empezamos haciendo el esquemático. (figura 32)

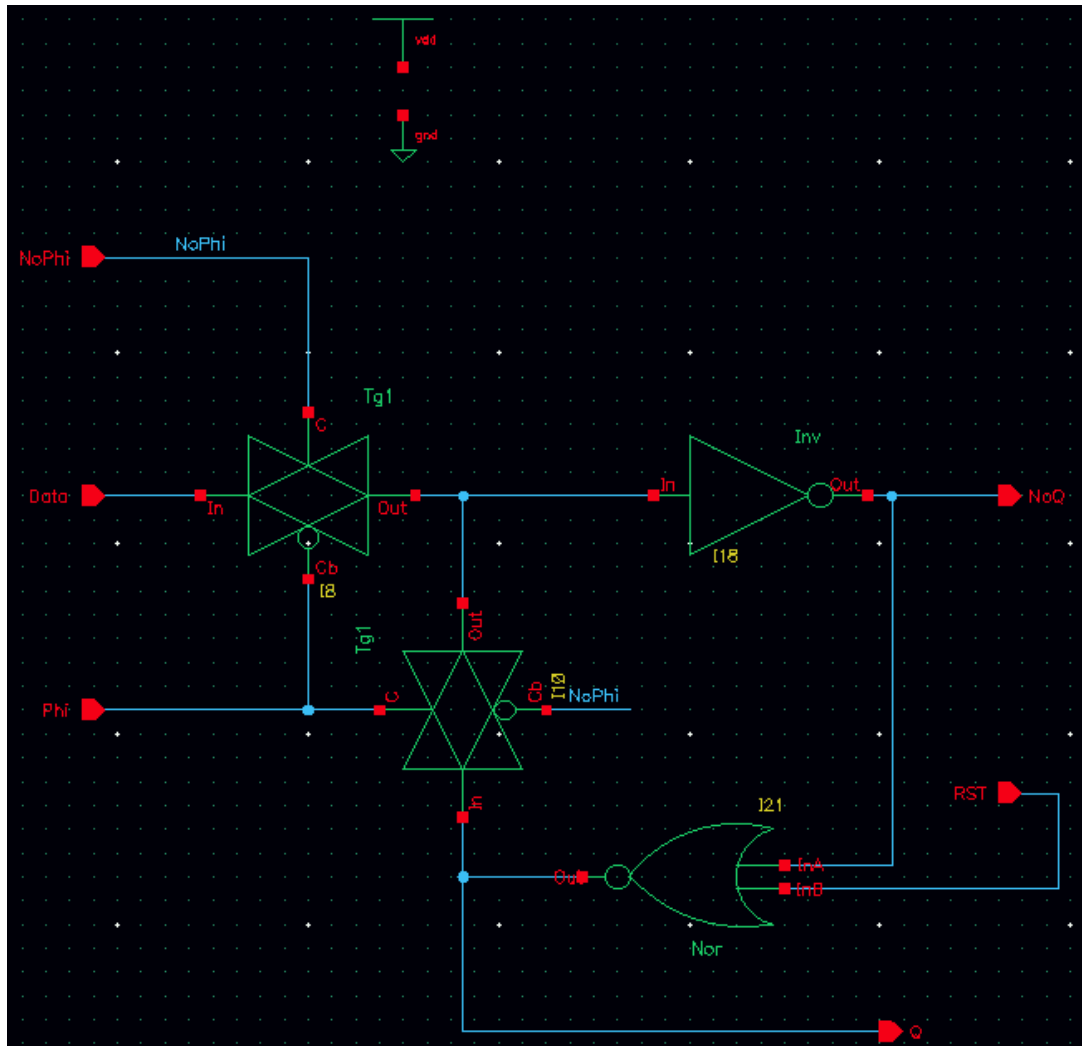


Figura 32: Esquemático del MasterD

Como se puede observar, es muy parecido al del SlaveD; lo único que cambia es la distribución de las puertas NOR y NOT ya que como queremos que en el caso de activar el RST se mantenga ese valor, es necesario realizarlo así en el Master como ya hemos comentado en el planteamiento inicial. También destacar que las entradas Phi y NoPhi están intercambiadas entre Master y Slave ya que este latch es transparente a nivel bajo.

Una vez hecho el esquemático nos pusimos a diseñar su respectivo símbolo (figura 33).

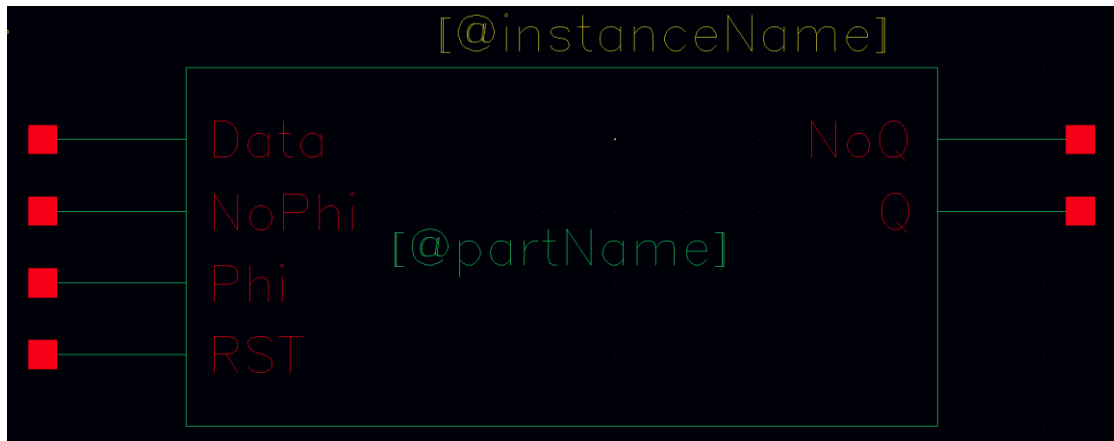


Figura 33: Símbolo del MasterD

Simultáneamente, diseñamos el layout de la figura 34 basándonos en los *stick-diagrams* planteados previamente.

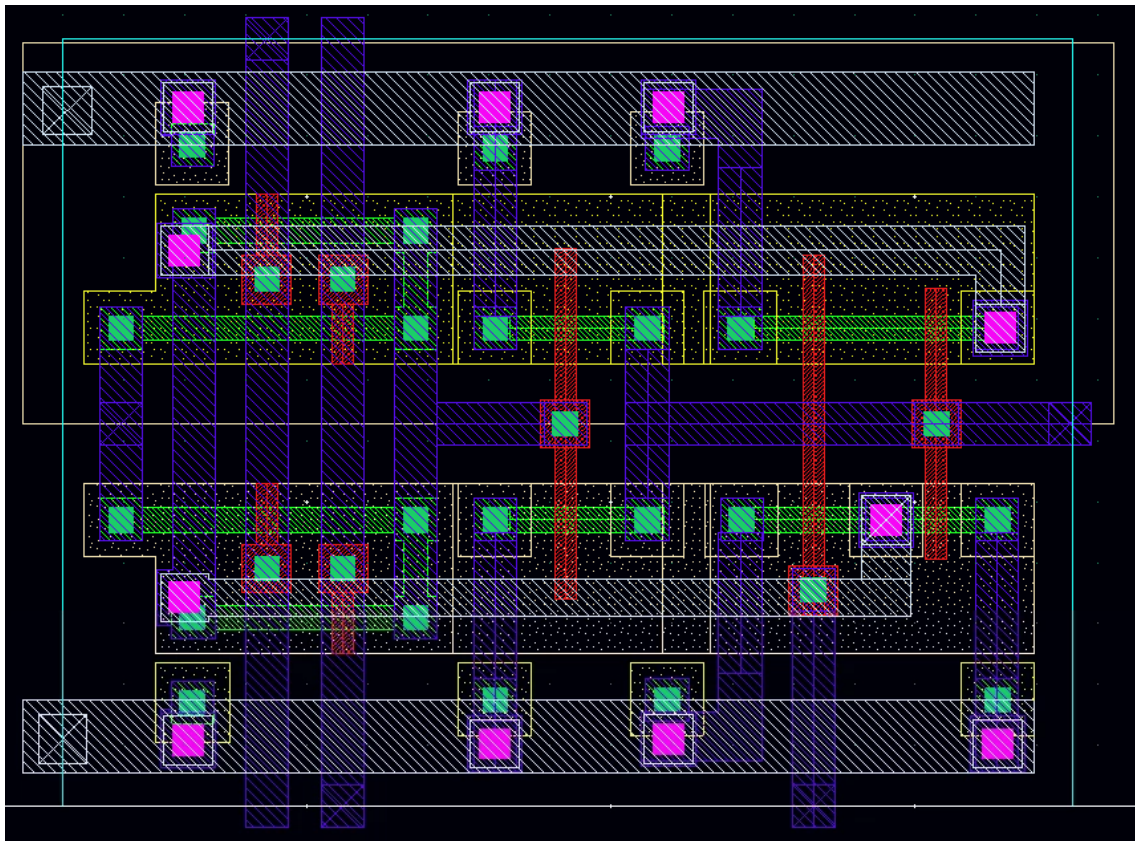


Figura 34: Layout del MasterD

El layout es bastante parecido al del SlaveD. Observamos la puerta NOR a la derecha del todo y el inversor en la zona central. Una vez acabado hicimos las comprobaciones pertinentes, DRC, LVS y simulación comparativa. (figura 35 y 36)

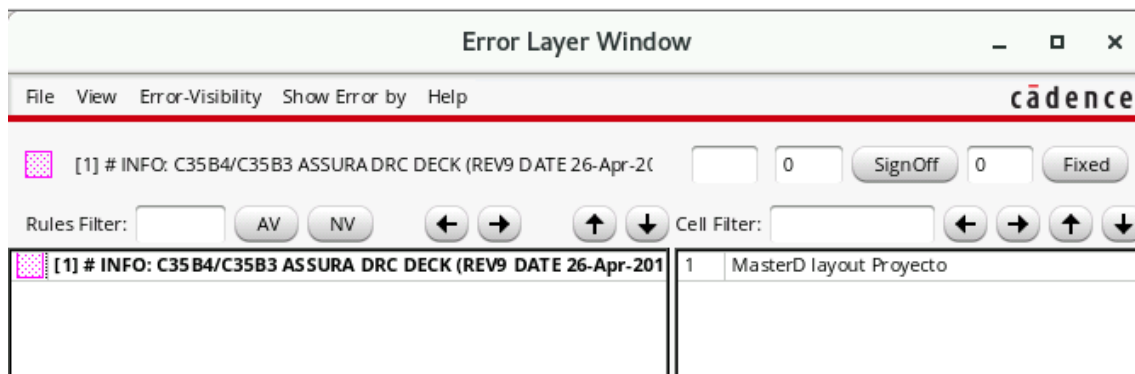


Figura 35: Asura DRC del MasterD

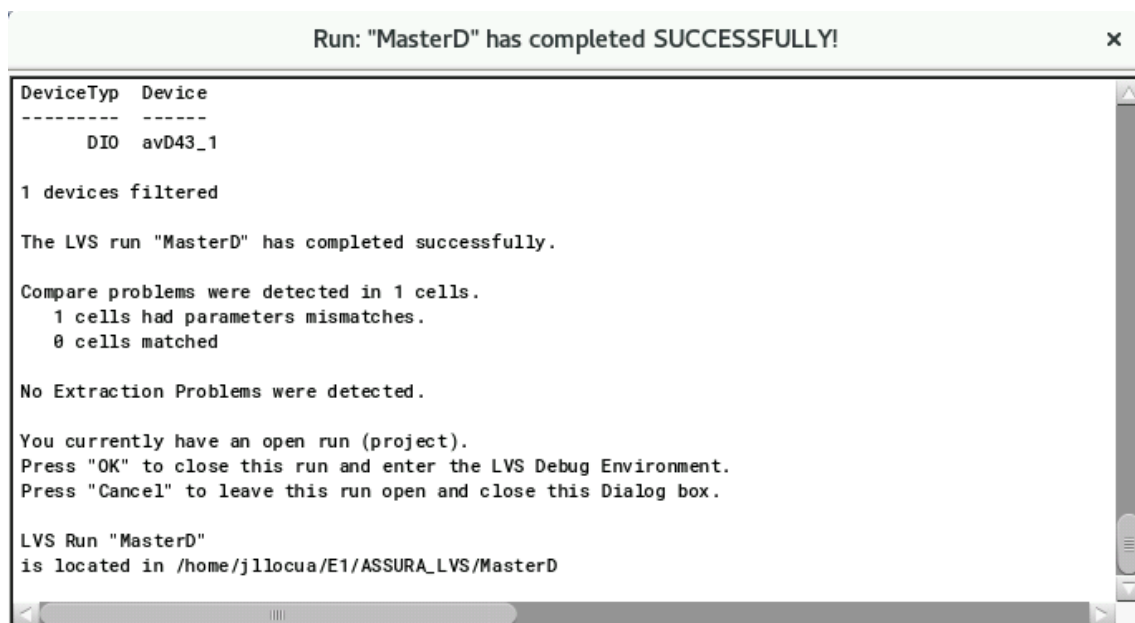


Figura 36: Asura Run LVS del MasterD

Podemos ver claramente como ambas comprobaciones son satisfactorias.

Como siempre, creamos un nuevo módulo llamado 'MasterD_tb' que instanciaba el layout y el esquemático y los comparaba en una misma simulación gracias al archivo "config". (figura 37)

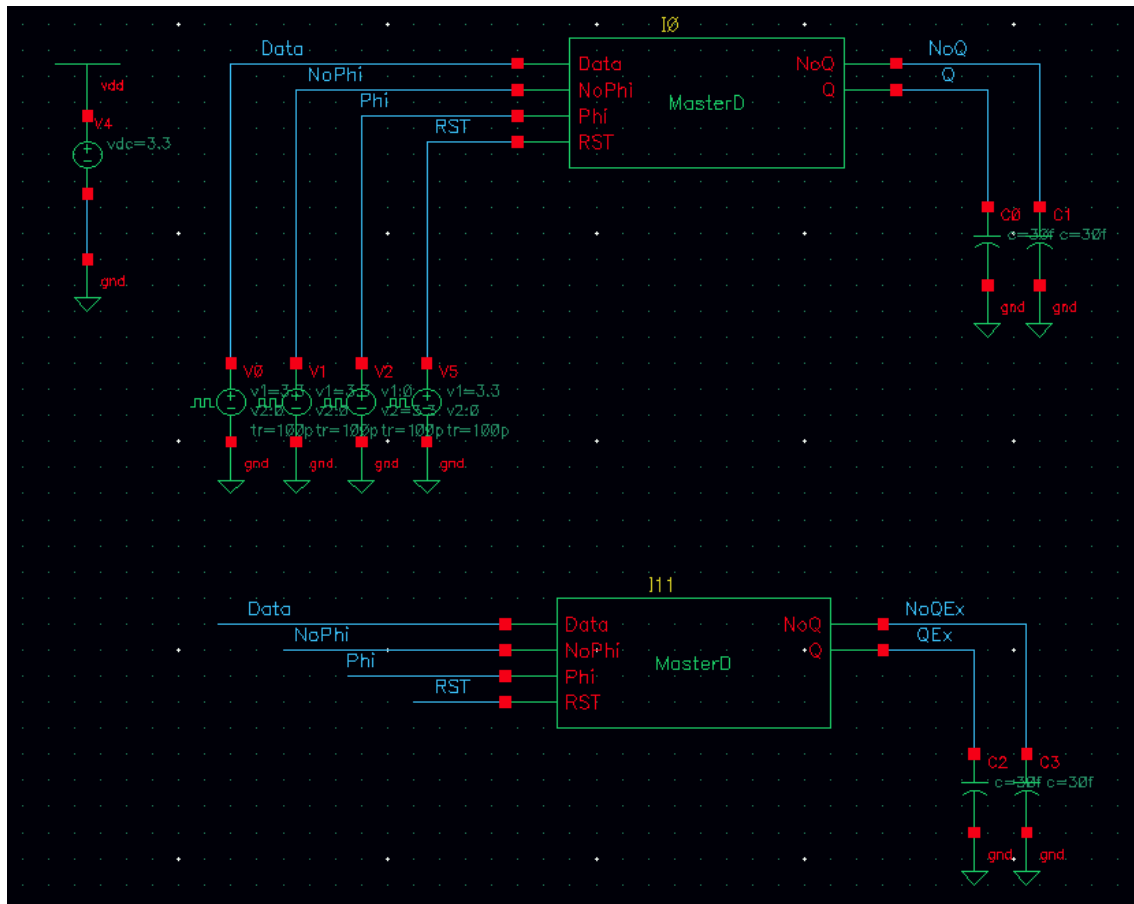


Figura 37: Esquemático de 'MasterD_tb'

Escogimos unos estímulos y estos son los resultados obtenidos: (figura 38)

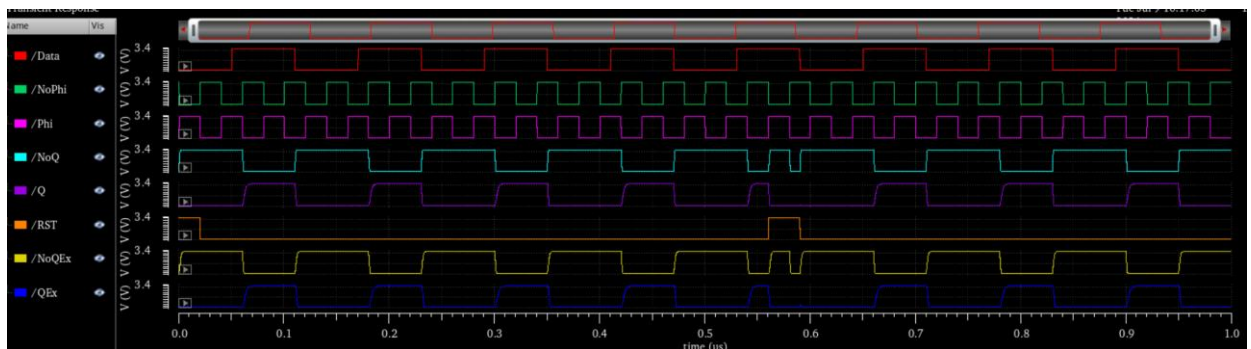


Figura 38: Resultados de la simulación del MasterD

Como podemos ver las señales del esquemático y el layout coinciden, y esta vez al contrario que en SlaveD son las señales de Q las que se ponen a cero cuando se activa el RSTa. No hay que hacer mucho caso a que NoQ y Q haya un momento que no sean complementarias tras el RST ya por como esta diseñado en el Master esto es normal. Lo importante es que funcione bien al luego unirlo con el Slave.

Pese a esto podemos ver un correcto funcionamiento del latch ya que cuando NoPhi se encuentra a nivel alto el latch es transparente hacia la salida Q.

9. Flip-flop D

Una vez diseñados Master, Slave y el Phase-Splitter podemos montar nuestro Flip-flop tipo D con reset asíncrono y activo en flanco de subida. Este es su esquemático: (figura 39). Destacar el uso de los inversores para lograr un RST activo a nivel bajo.

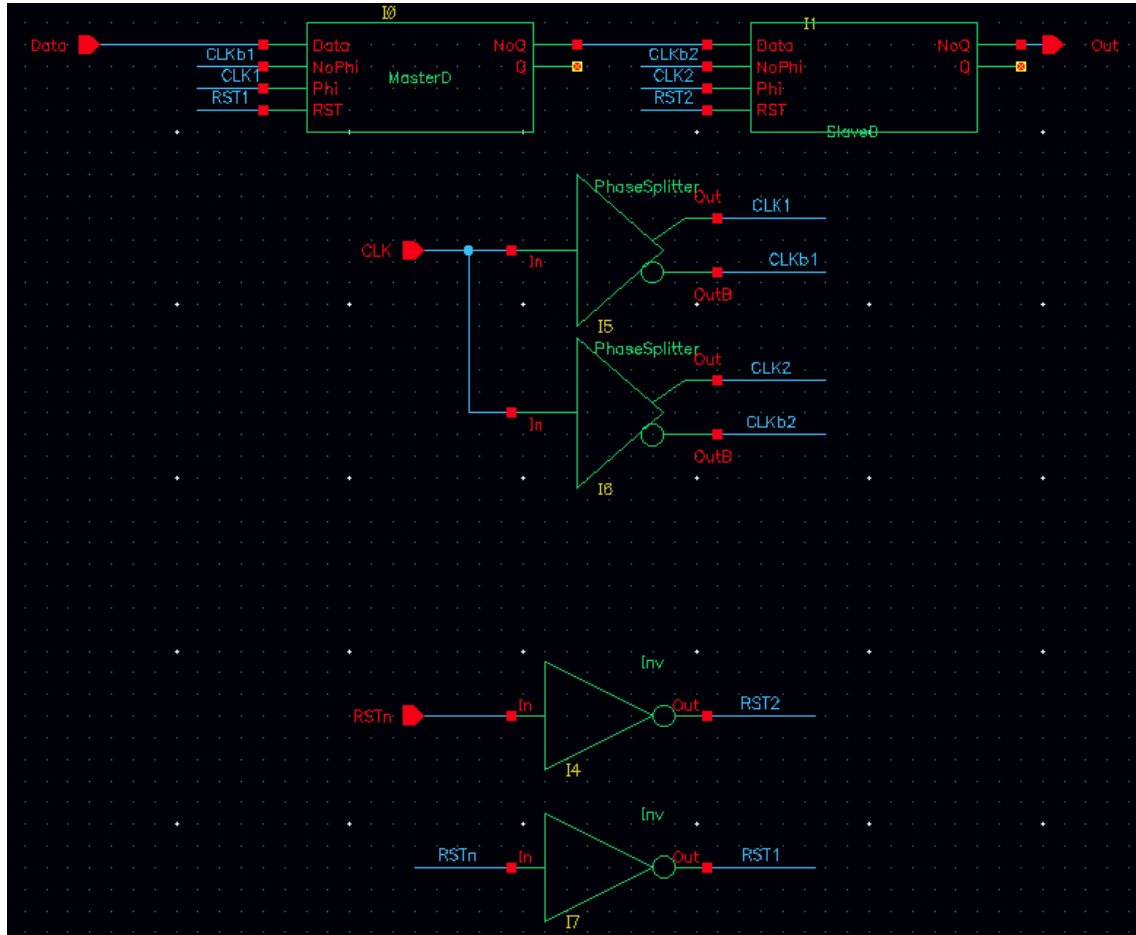


Figura 39: Esquemático del Flip Flop D

Seguidamente hicimos su símbolo: (figura 40)



Figura 40: Símbolo del Flip Flop D

Posteriormente hicimos el layout al que además de los módulos mencionados previamente tuvimos que añadir los 2 inversores (uno para cada latch) en el exterior: (figura 41)

Los inversores fueron tomados de la librería “PIEZAS_DIG” por lo que no se detalla su diseño en esta memoria.

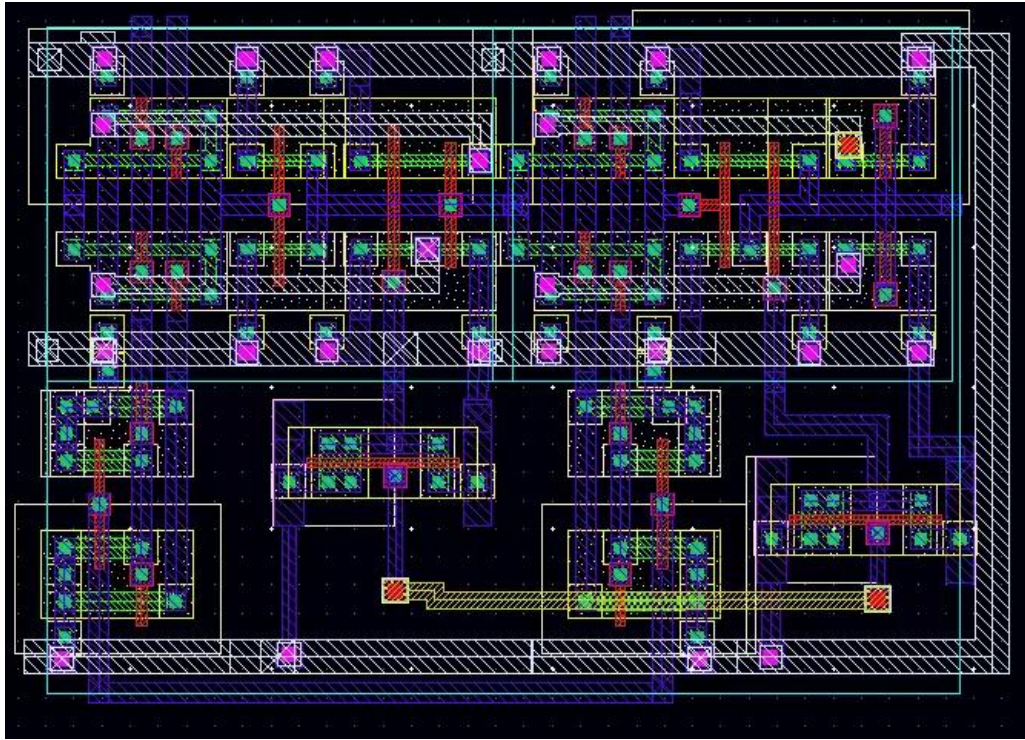


Figura 41: Layout del Flip Flop D

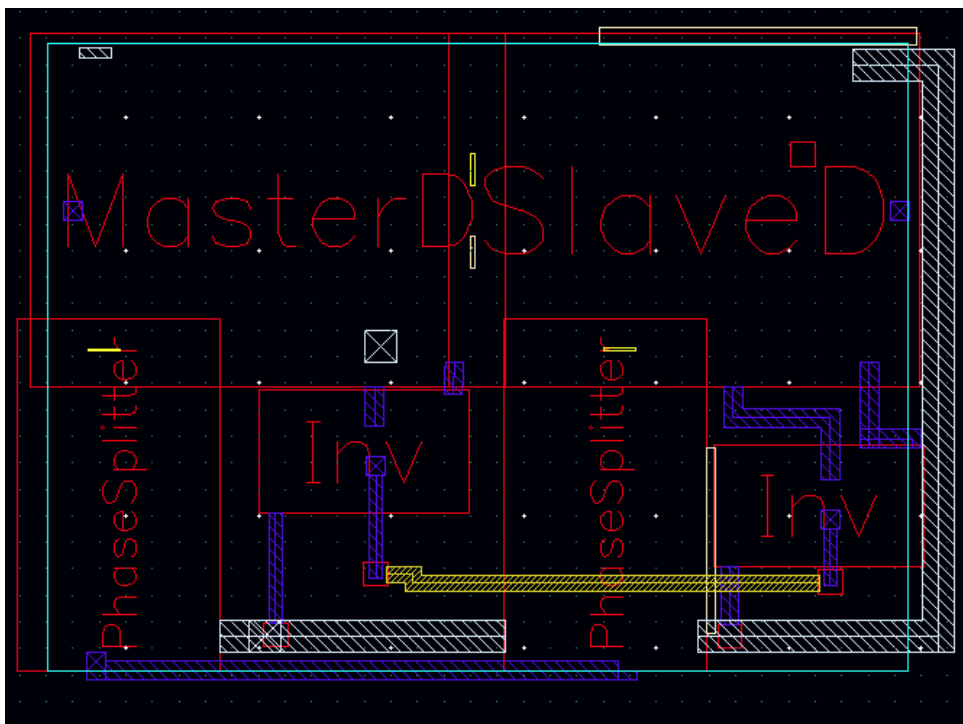


Figura 42: Estructura jerárquica del layout del Flip Flop D

Podemos observar la correcta jerarquía visualizando las instancias del MasterD, SlaveD, Phase Splitter y los inversores dentro del layout. (figura 42)

Una vez finalizado llevamos a cabo las comprobaciones pertinentes y se generó su versión extraída. (figura 43 y 44)

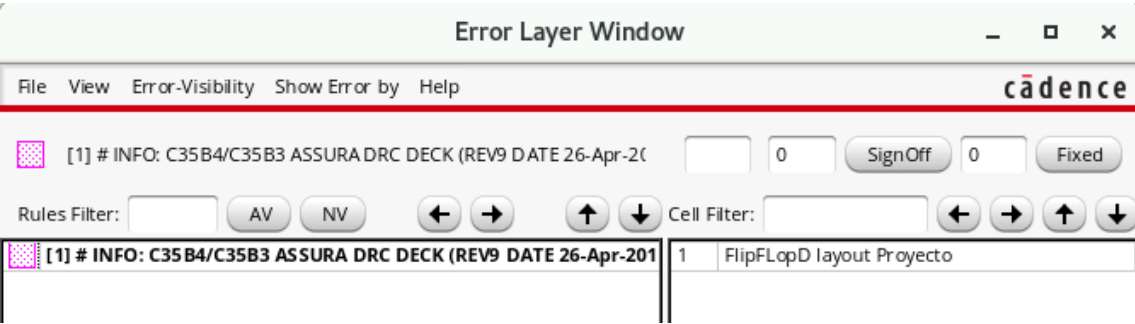


Figura 43: Asura DRC del Flip Flop D

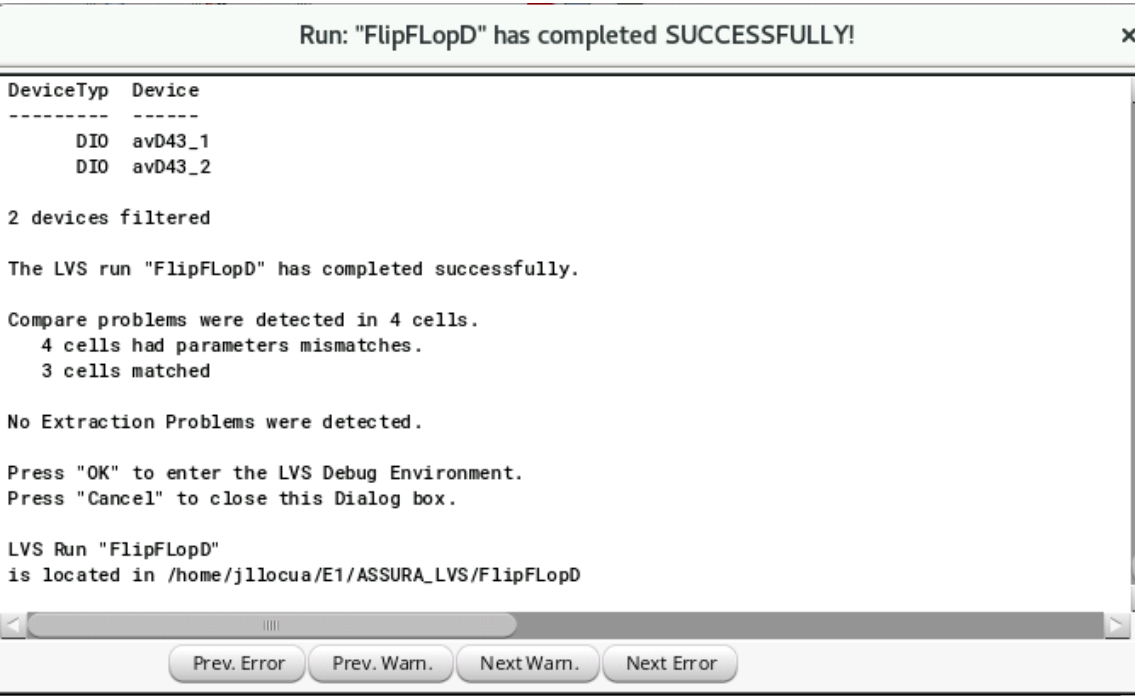


Figura 44: Asura LVS del Flip Flop D

Para asegurarnos de que la simulación entre el extraído y el esquemático coincidieran hicimos otro módulo llamado ‘FlipFlopD_tb’, que instanciaba ambos módulos y los comparaba en una simulación gracias al archivo “config”. (figura 45)

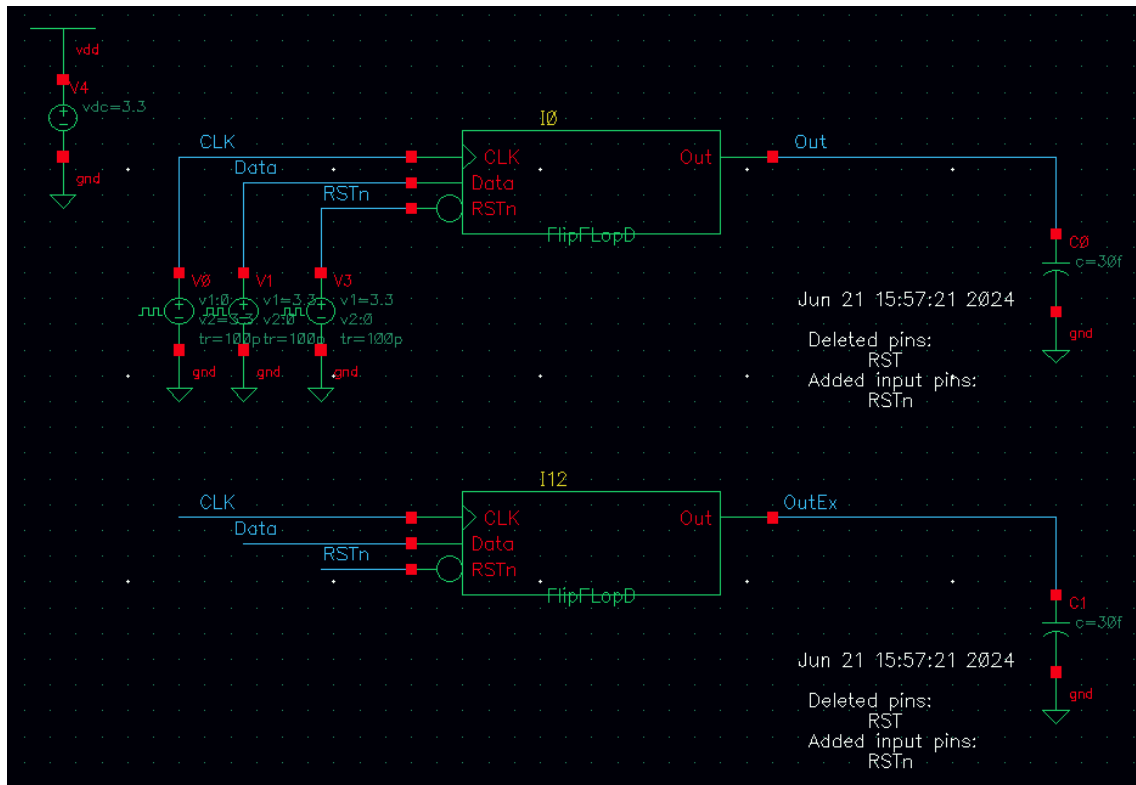


Figura 45: Esquemático de 'FlipFlopD_tb'

Simulando con los estímulos pertinentes obtenemos lo siguiente: (figura 46)

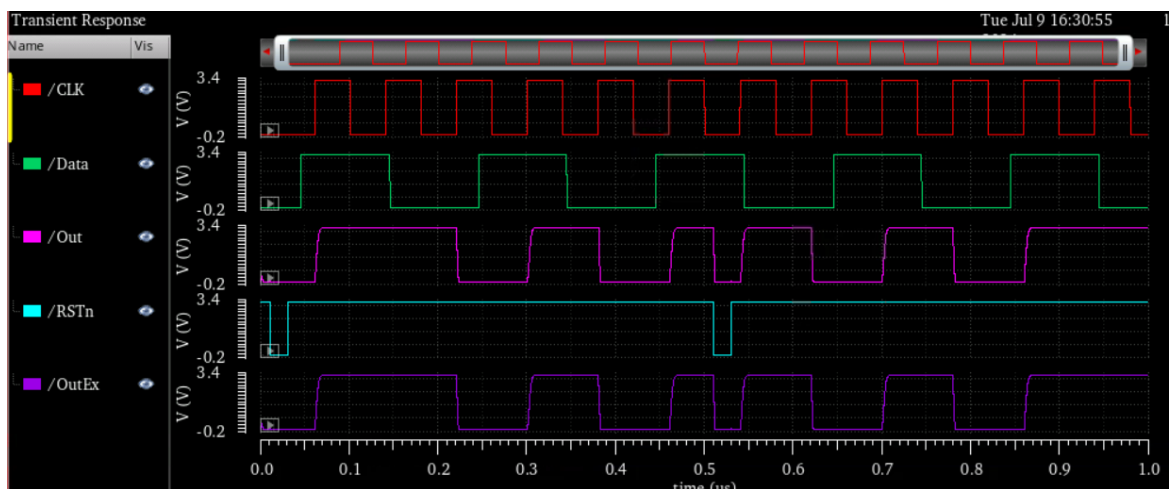


Figura 46: Waveform del Flip Flop D

Donde podemos ver que efectivamente la entrada se muestrea a la salida cada flanco de subida del CLK y cuando activamos el RSTa la salida se pone a cero.

Con esto concluimos que nuestro Flip-flop tipo D, y por lo tanto el Master, Slave y los demás componentes instanciados, han sido diseñados correctamente.

10. Flip-flop JK

Con el mux y el Flip-flop tipo D ya diseñados nuestra jerarquía esta completa y solo queda instanciar ambos módulos y realizar las conexiones necesarias. Para el layout hemos empleado una capa adicional de metalización (Metal3) debido a la complejidad de las conexiones para así evitar cortocircuitos indeseados.

Este es su esquemático: (figura 47)

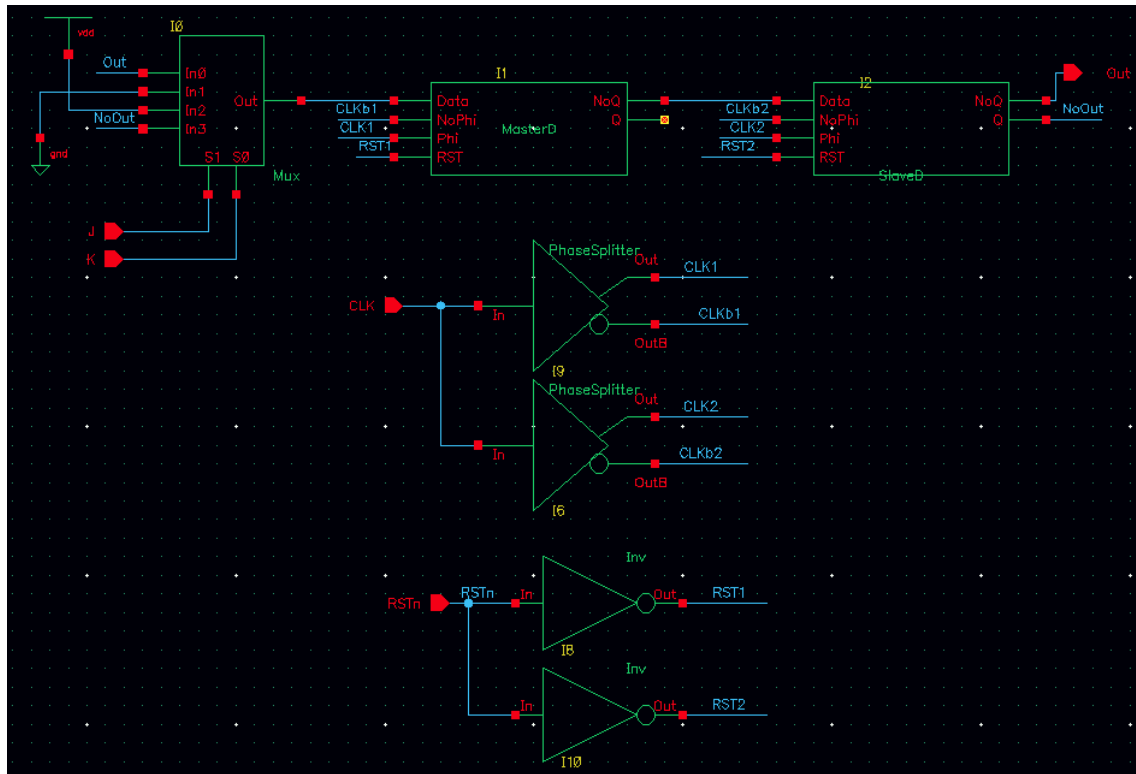


Figura 47: Esquemático del Flip Flop JK

Aquí se pueden apreciar las instancias de todos los módulos realizados previamente. (figura 47).

Luego diseñamos su símbolo: (figura 48)

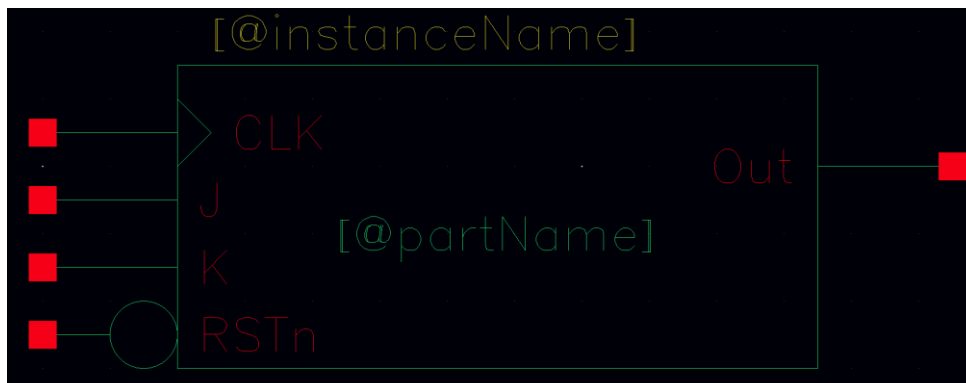


Figura 48: Símbolo del Flip Flop JK

Posteriormente instanciamos los layouts del MUX y el Flip-flop D para finalizar el diseño y conectamos de forma adecuada todas las alimentaciones, salidas y entradas. (figura 49)

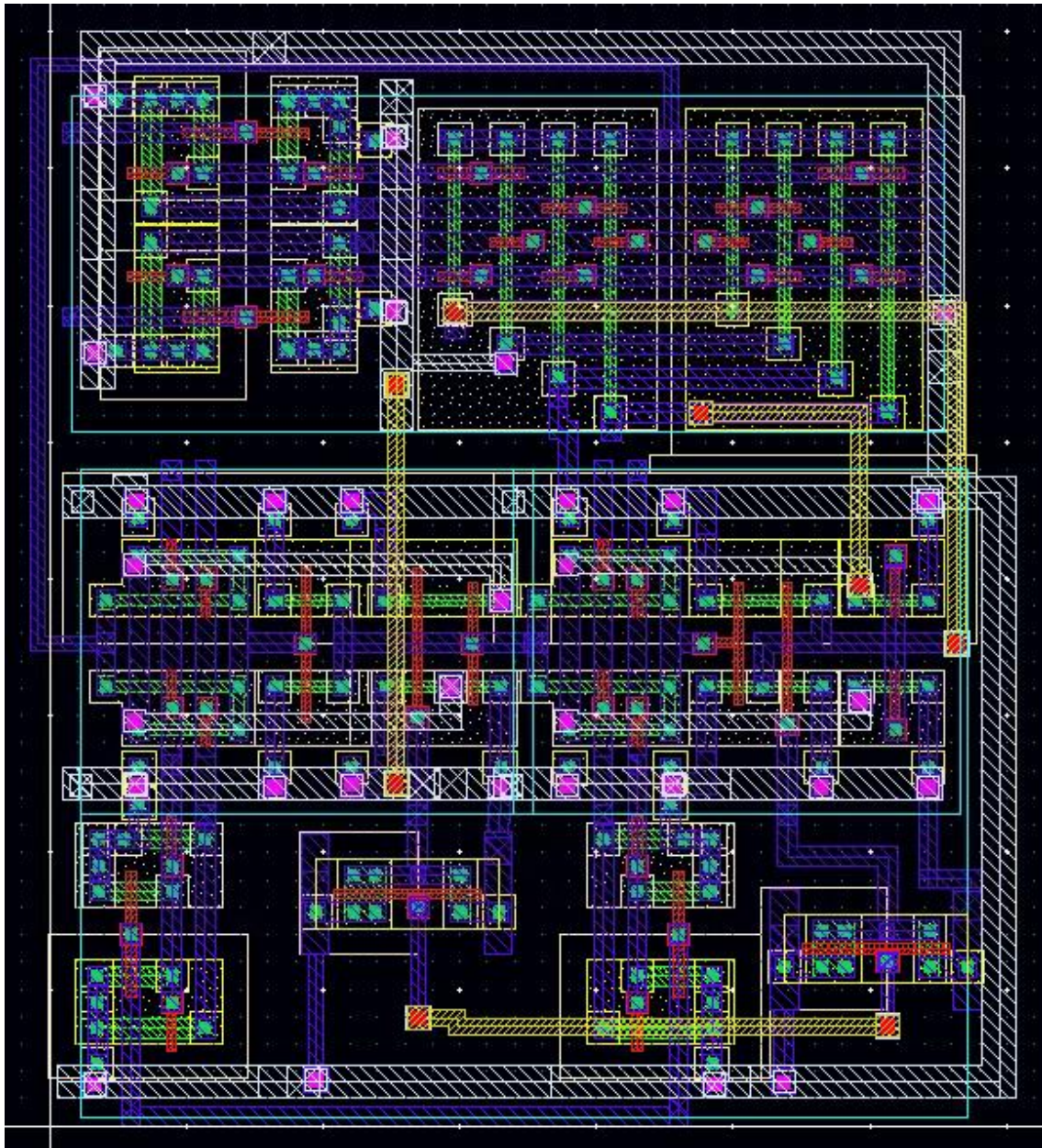


Figura 49: Layout del Flip Flop JK

Como podemos ver en la figura 50 el multiplexor se encuentra en la parte superior y el Flip-Flop tipo D se encuentra en la parte inferior.

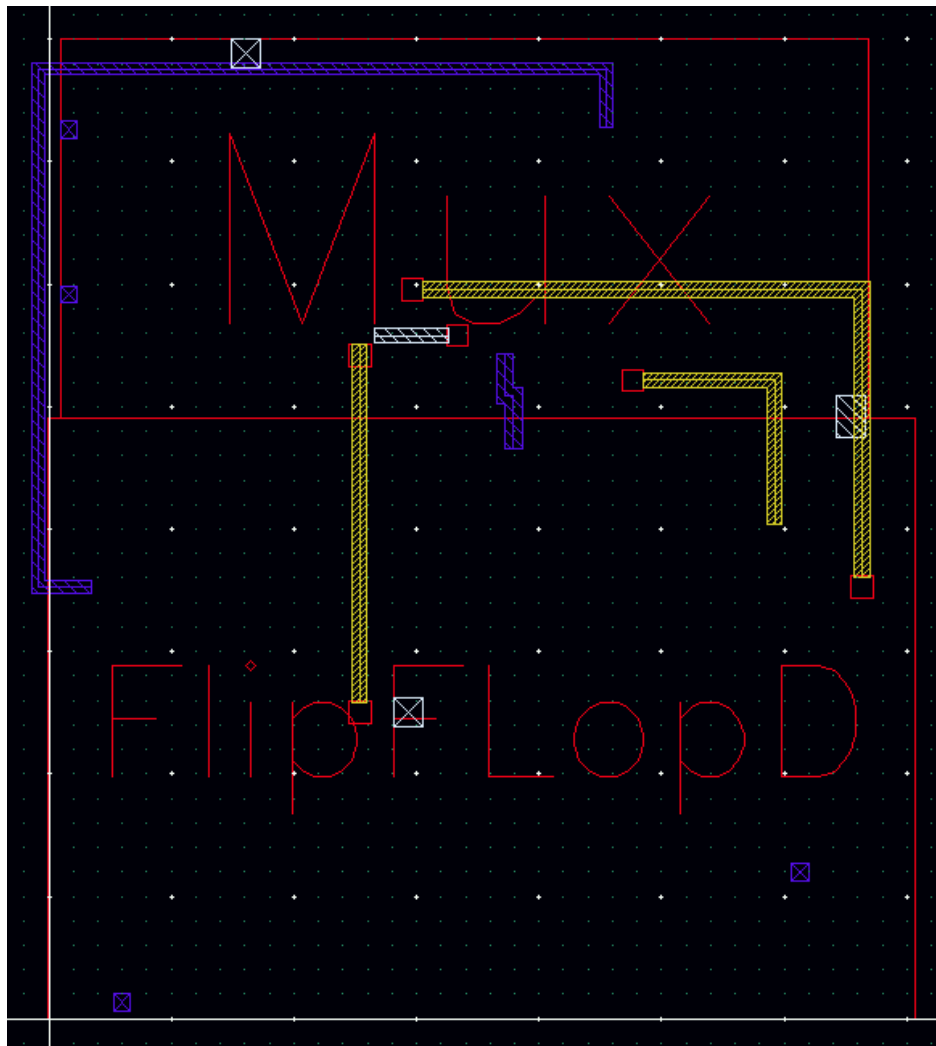


Figura 50: Jerarquía del Flip Flop JK

Simultáneamente hicimos las comprobaciones necesarias con la herramienta Asura para verificar las reglas de diseño y el LVS. (figura 51 y 52)

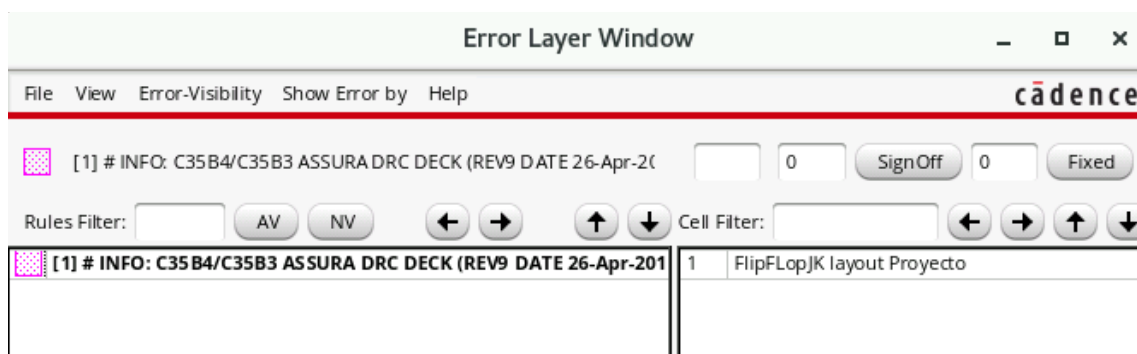


Figura 51: Asura DRC del Flip Flop JK

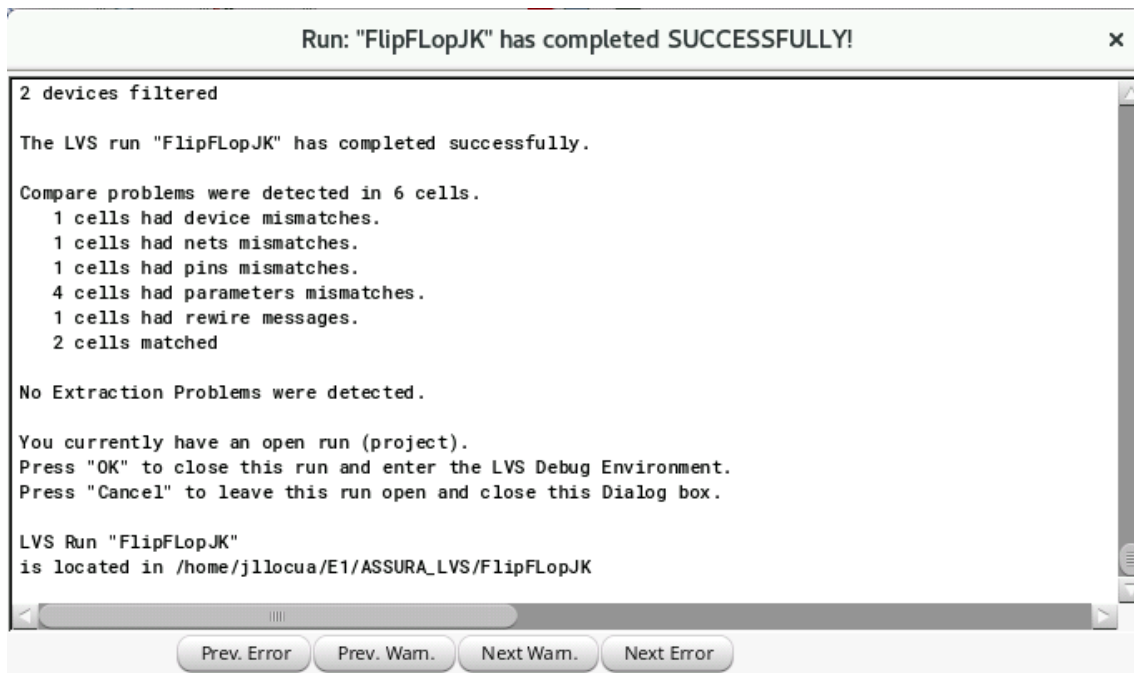


Figura 52: Asura LVS del Flip Flop JK

Se puede apreciar que las reglas de diseño funcionan de maravilla. Sin embargo, el LVS produce desencajes debido a que se está utilizando el multiplexor diseñando al principio y se arrastra el *missmatch* previamente explicado. A pesar de ello, como ya hemos explicado anteriormente, este *missmatches* no va a afectar al funcionamiento ni a la fabricación física del proyecto, de hecho, se ve claramente el mensaje de ‘*No Extraction Problems were detected*’.

Por último y para asegurarnos de que el Flip Flop JK se comportará de manera adecuada creamos un nuevo módulo llamado ‘FlipFlopJK_tb’ para realizar la simulación.

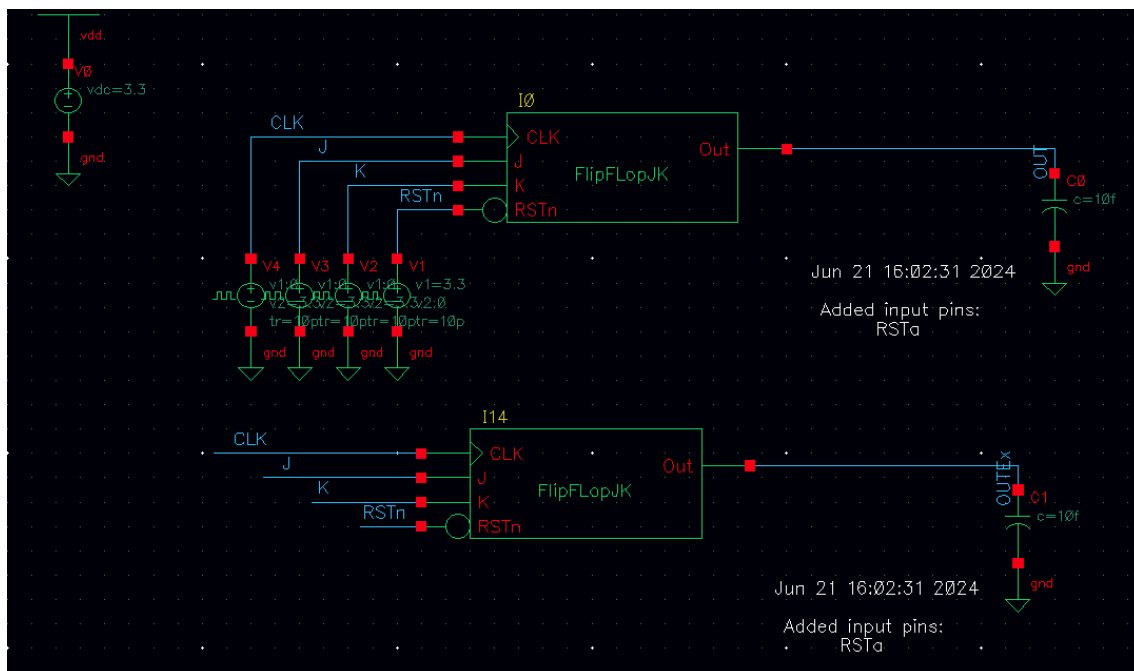


Figura 53: Esquemático del FlipFlopJK_tb

Como hemos hecho siempre, aquí se compara el diseño del layout con el del esquemático (figura 53). Y los resultados obtenidos son los siguientes: (figura 54)

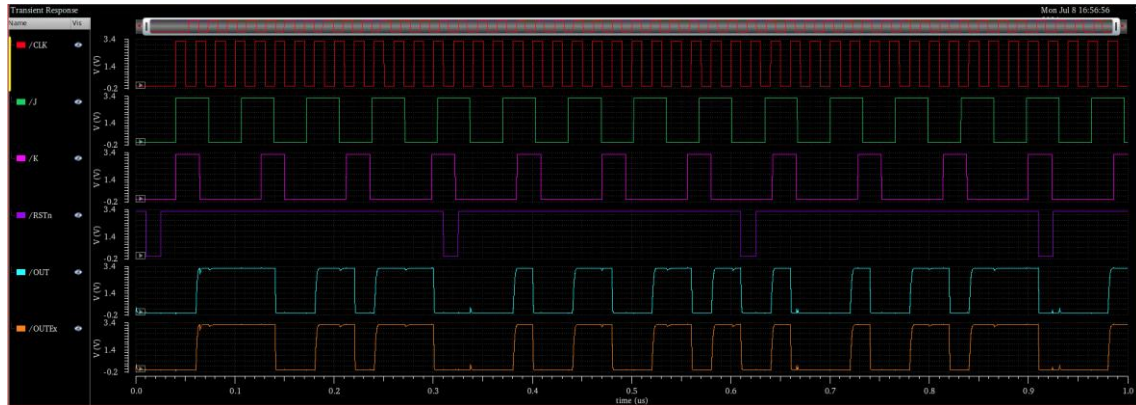


Figura 54: Resultados de la simulación del Flip Flop JK

Aquí se puede apreciar como las salidas Out y OutEx son idénticas y la lógica es la adecuada. El Flip-flop JK conmuta bien, carga bien el 1 y el 0 y mantiene la salida cuando es necesario, al igual que cuando toca negarla.

11. Especificaciones técnicas

11.1 Frecuencia máxima

Para buscar el valor de la frecuencia máxima lo ideal era realizar un análisis paramétrico con la frecuencia como variable. Sin embargo, nuestro programa iba demasiado lento para realizar esto por lo que lo hicimos de manera manual, aumentando la frecuencia de operación en pasos de 100 MHz comenzando por 100 MHz. En cuanto la señal se empieza a distorsionar y veamos que la lógica falla ahí determinaremos que está el límite.



Figura 55: Simulación a 100 MHz

Observamos como a 100 MHz el Flip-flop conmuta de forma apropiada. A continuación, simularemos con 200 MHz. (Figura 55)

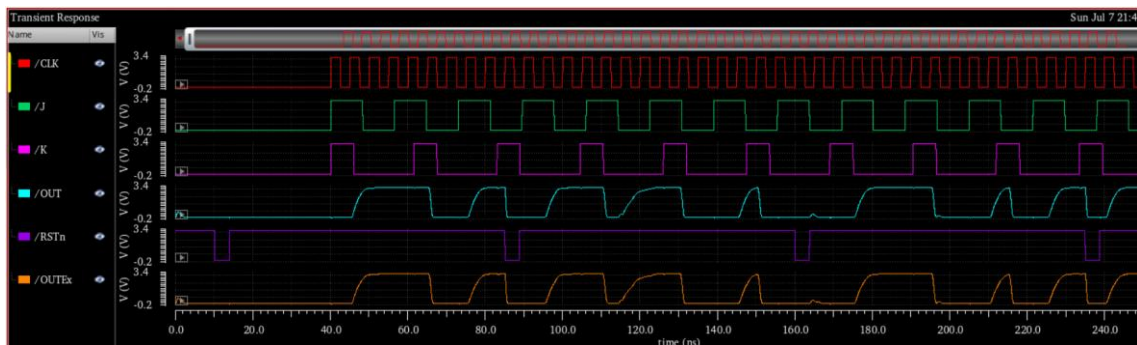


Figura 56: Simulación a 200 MHz

A 200 MHz más de lo mismo, seguimos teniendo el comportamiento esperado. (Figura 56)

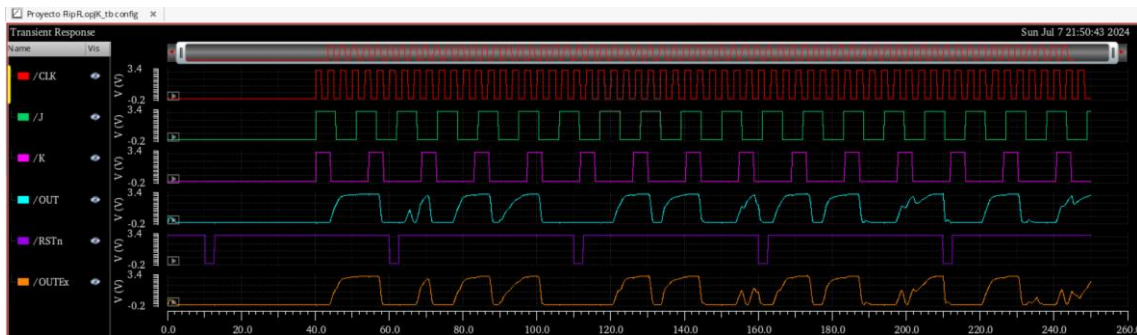


Figura 57: Simulación a 300 MHz

En 300 MHz sí que ya observamos picos no deseados además de zonas donde la lógica falla por lo que nuestra frecuencia máxima de operación debe estar entre 200 y 300 MHz. (Figura 57) Para ello disminuimos un poco el valor para intentar ser más precisos en nuestra búsqueda y simulamos con 250 MHz.

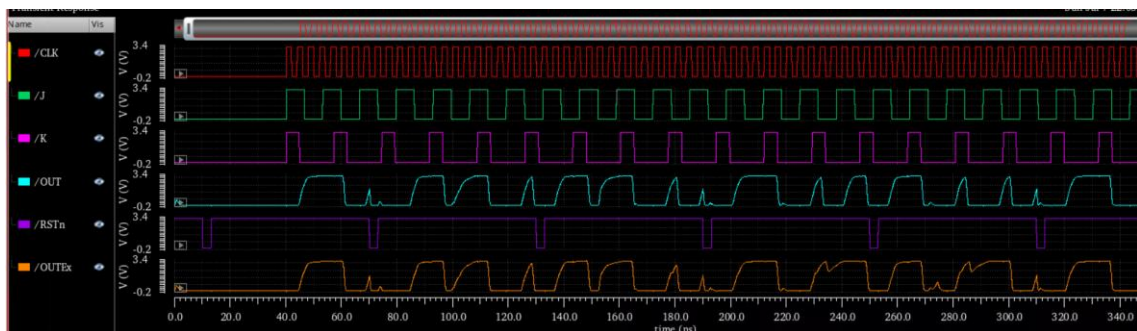


Figura 58: Simulación a 250 MHz

A 250 MHz podemos ver que casi toda la lógica la hace bien, pero en una zona del final el esquemático empieza a ya no coincidir con el el layout aparte de haber algunos picos con valores de tensión que no llegan al umbral para la tecnología

Para confirmar este valor como nuestro límite vamos a realizar una simulación mas a 240 Mhz para confirmar que nuestro diseño funciona correctamente.

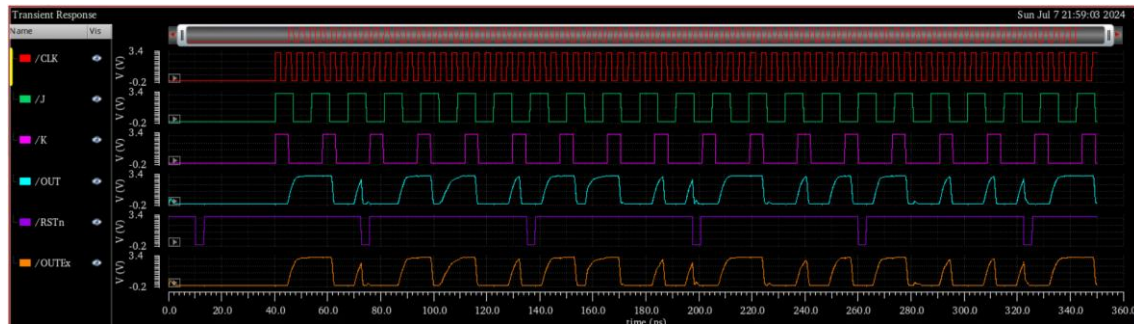


Figura 59: Simulación a 240 MHz

Observamos que conmuta decentemente, aunque los retardos ya empiezan a ser notorios y algunos picos llegan justo a 2,6V. (Figura 59) Concluimos que nuestro valor máximo es **240 MHz**.

11.2 Potencia media

Para realizar la medida de la potencia media de nuestro diseño será necesario hacer uso de la fórmula $P=I \cdot V$ donde V es Vdd que tiene valor 3,3V e I es la corriente que media que demanda el circuito.

Para sacar el valor de la corriente media será necesario colocar “I probes” para después multiplicar el valor medio por 3,3 y así obtener la potencia media.



Figura 60: Forma de onda corriente generador

Aquí observamos la forma de onda de la corriente que debido a que las fluctuaciones son pequeñas no se observa con claridad su forma.

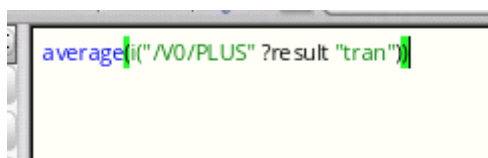


Figura 61: Cálculo corriente media

A continuación, enviamos la forma de onda a la calculadora para calcular su valor medio mediante la función “average”. Obtenemos el siguiente valor:

Expression	Value
1 average(i["/V0/P...]	-20.29E-6

Figura 62: Valor corriente media

Observamos un valor de corriente media de 20.3uA. Multiplicando por 3,3 encontramos el valor de potencia media obteniendo así un valor de **67uW**.

11.3 Retardo Layout-Esquemático

Otro valor importante para considerar es el retardo entre el layout diseñado y el esquemático. Para ello iremos a la simulación y buscaremos un punto de referencia. A partir de ahí cogeremos 2 markers verticales y uno horizontal con los cuales podremos calcular la diferencia temporal: (Figura 61)

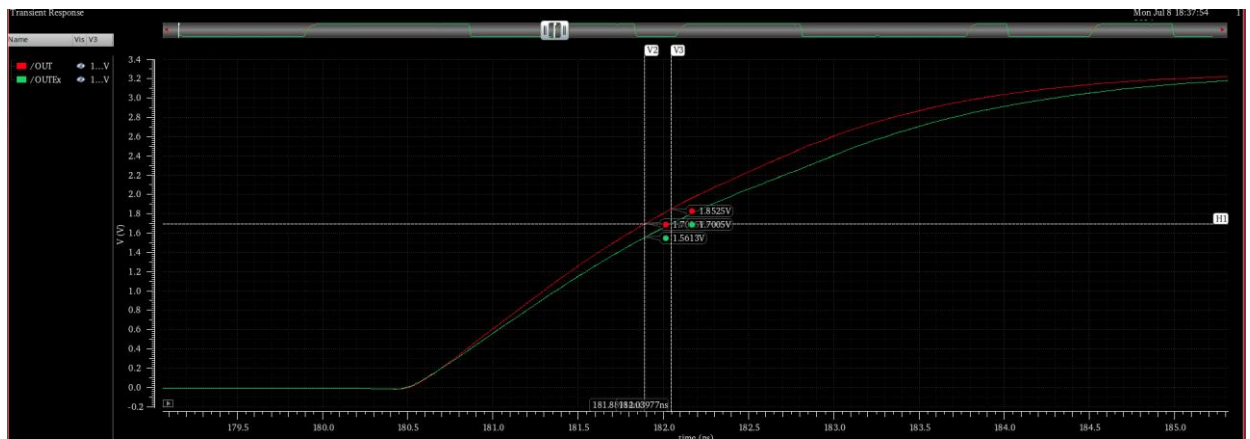


Figura 63: Cálculo del retardo Layout-esquemático

El retardo calculado es de: $182.04 - 181.88 \text{ ns} = 160 \text{ ps}$ de retardo.

12. Análisis Crítico de los resultados

12.1 Conclusiones

En conclusión, el diseño del Flip-flop JK desarrollado ha demostrado conmutar de forma adecuada entre sus estados, respondiendo correctamente a las señales de entrada y mostrando estabilidad en sus salidas. Sin embargo, se han identificado áreas de mejora que podrían optimizar su rendimiento y fiabilidad. Por ejemplo, se podría reducir el tiempo de retardo (propagation delay) mediante el uso de componentes de mayor velocidad, lo cual permitiría una conmutación más rápida y eficiente. Esta mejora no solo perfeccionaría la funcionalidad del Flip-flop JK, sino que también aumentarían su aplicabilidad en sistemas electrónicos más complejos y exigentes.

12.2 Otras formas de hacerlo

Nuestro diseño se puede realizar de distintas maneras. Una posibilidad sería implementar el Flip-flop JK con lógica booleana que es como un grupo de compañeros de nuestra clase lo ha realizado. Consistiría en diseñar el Flip flop mediante 2 latches máster slave del mismo modo que hemos hecho nosotros, pero la construcción de estos latches sería en base a puertas NAND y/o NOR. Para el mux mas de lo mismo. De esta forma el diseño layout no sería más que una instanciación de una multitud de puertas conectadas de la forma adecuada por lo que también seria una forma de conseguir un diseño compacto además de ser escalable.

13. Bibliografía

- CMOS VLSI Design A Circuits and Systems Perspective
- Recursos de Poliformat, transparencias de la asignatura