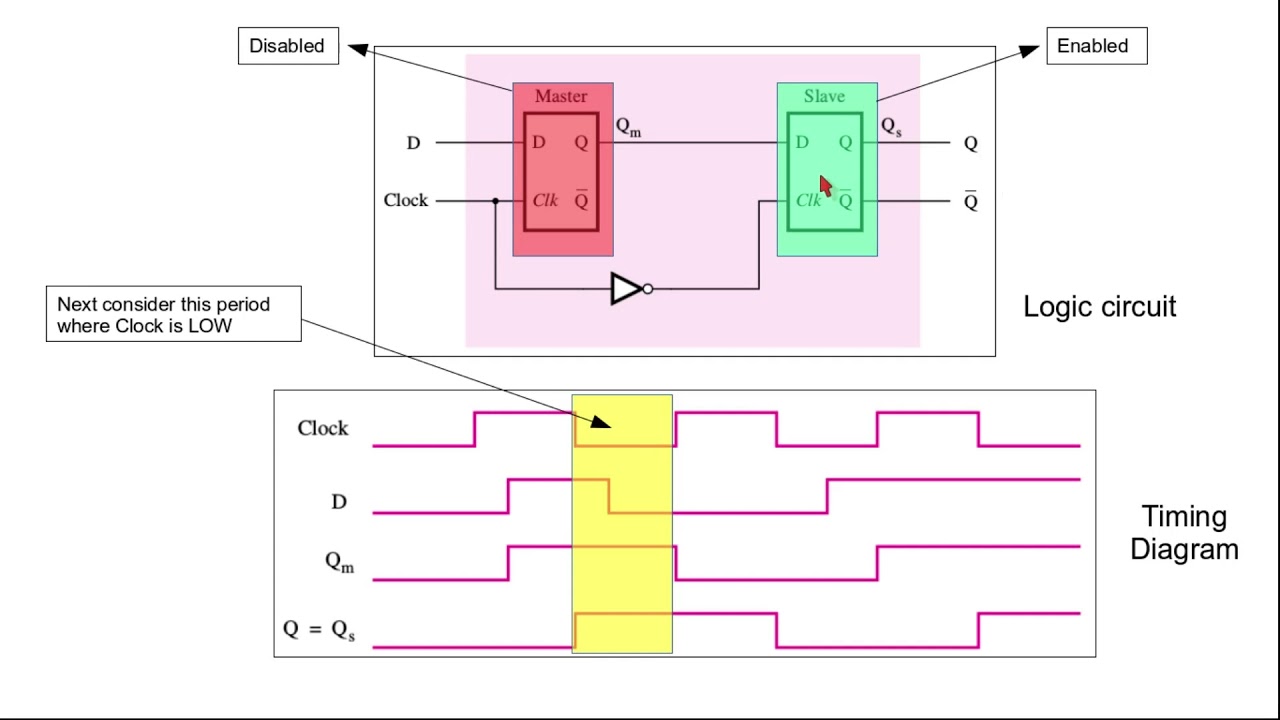
**VLSI**

**MEMORIAS PRÁCTICAS 1 Y 4**

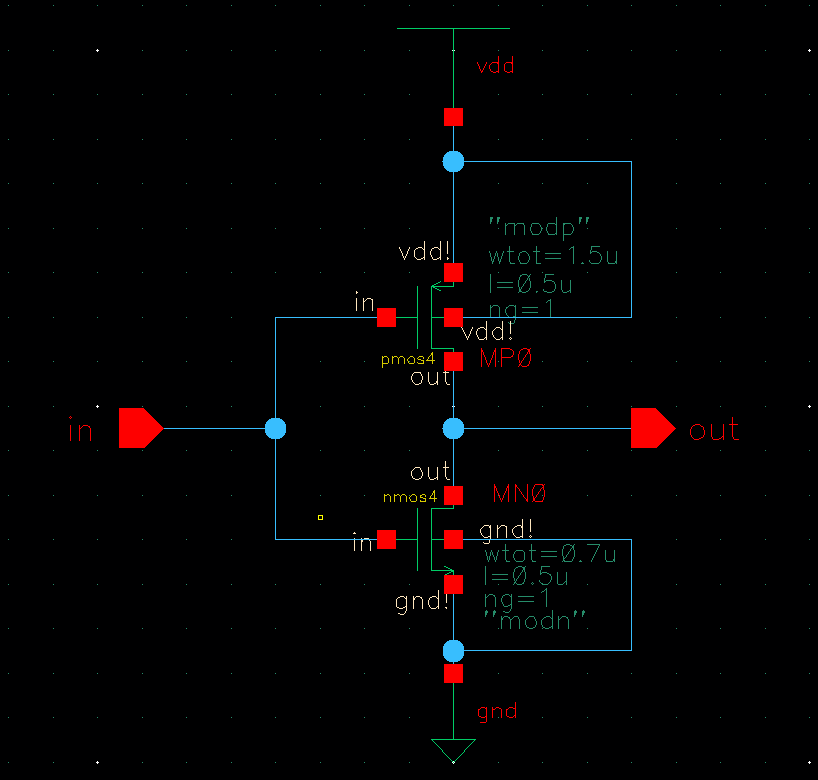


En la siguiente memoria se describe la realización de las prácticas 1 y 4 de la asignatura donde hemos tratado de diseñar un Flip-Flop de tipo D activo en flanco de subida de arquitectura maestro-esclavo. Para ello hemos diseñado la vista esquemática de los distintos elementos además de sus Layouts correspondientes y hemos simulado todos los módulos y comparado entre ellos. Al módulo maestro lo llamamos MasterD y al esclavo SlaveD.

***PRÁCTICA 1***

*TRICKLE*

En primer lugar, empezamos haciendo el diseño del *Trickle* (figura 1); el cual tiene como función invertir la señal que entra. Este módulo es usado más adelante para el diseño del MasterD y Slave. Está compuesto por dos transistores, uno *pmos* y uno *nmos*.

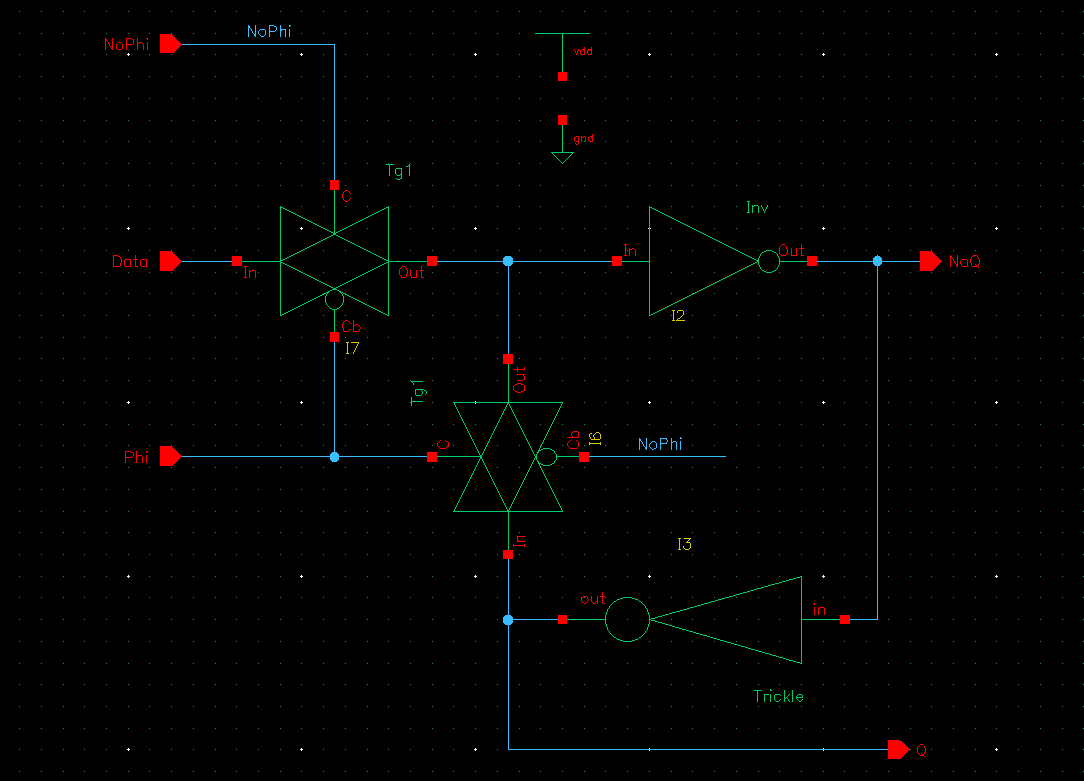


***Figura 1: Esquemático del Trickle***

Una vez terminado hicimos su representación simbólica para poder usarlo en futuras instancias. Cabe destacar la anchura del canal de sus transistores los cuales en los layouts de mas adelante no se realizan con tal anchura causando “Paramter Mismatches” en el Assura LVS.

*SLAVE D*

Posteriormente hicimos el *SlaveD*, el slave D es un latch activo a nivel bajo lo que quiere decir que es transparente cuando el reloj vale cero. Lo primero que diseñamos fue el esquemático (figura 2) y posteriormente verificamos su funcionamiento mediante un test bench Su simulación se hace más adelante. Por otro lado, implementamos el símbolo del *SlaveD*.



***Figura 2: Esquemático del SlaveD***

En la imagen de abajo observamos el diseño del test bench para la verificación.

***A computer screen shot of a computer

Description automatically generated***

***Figura 3: Testbench del SlaveD***

En esta figura podeos observar un correcto comportamiento del Slave D el cual está siendo transparente cuando CLK vale cero y manteniendo el valor previo en caso contrario. En este caso eso da lugar a la replica exacta de la señal “Data”.

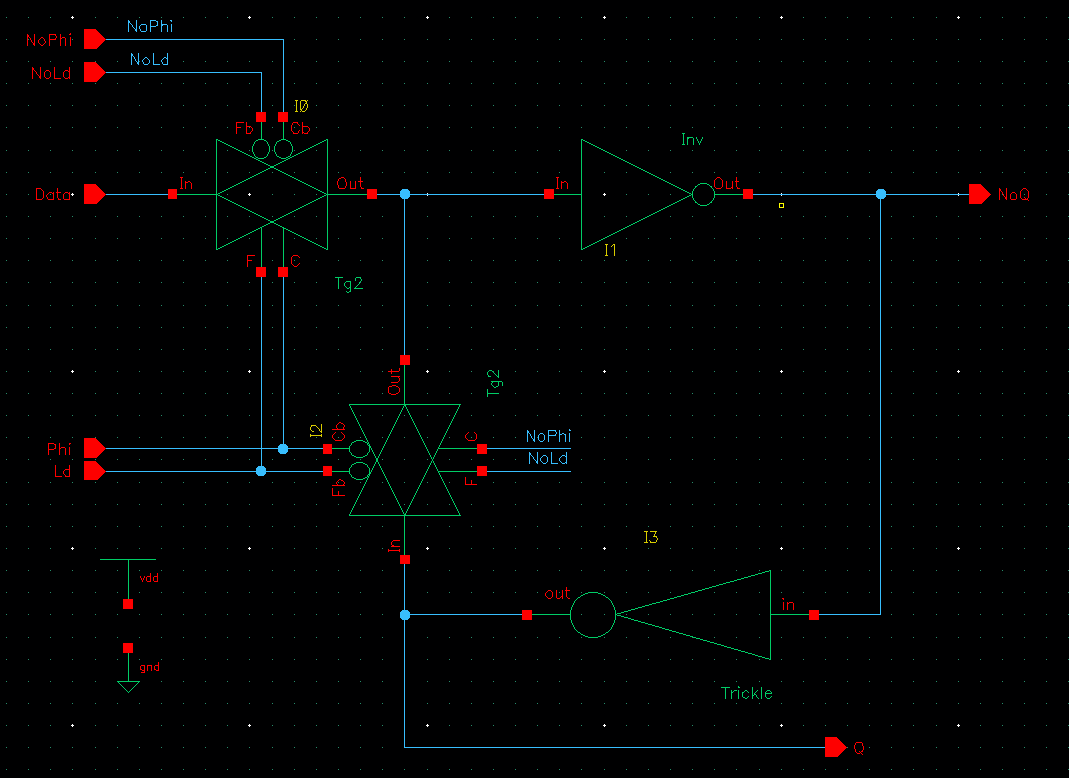
***A screen shot of a computer

Description automatically generated***

***Figura 4: Waveform del SlaveD***

*MASTER D*

Seguidamente realizamos el diseño del *MasterD* (figura 5), con su respectivo esquemático. El MasterD no es más que un latch con “Enable” activo a nivel alto lo que quiere decir que es transparente cuando el reloj vale y el “Enable” valen uno.



***Figura 5: Esquemático del MasterD***

También hemos hecho el respectivo símbolo del *MasterD* yhemos procedido a verificar el diseño con un test bench adecuado.

***Figura 6:Testbench del MasterD***

***A computer screen shot of a computer

Description automatically generated***

A screenshot of a computer

Description automatically generated

***Figura 7: Waveform del MasterD***

Podemos observar los resultados donde se observa claramente como la señal “Data” pasa a la salida cuando el “Enable” y el reloj están a uno. Las señales negadas son simplemente para el correcto funcionamiento del latch y se ven como son la inversa de sus correspondientes sin negar.

*FFDMS*

Posteriormente continuamos uniendo el MasterD con el SlaveD y usando inversores para generar las señales necesarias. Esto da lugar al Flip-Flop final el cual ahora verificaremos para ver si se ha diseñado correctamente.

A screenshot of a computer

Description automatically generated

***Figura 8: Esquemático del FFDMS***

En la figura de debajo observamos el test bench diseñado para la comprobación.

A computer screen shot of a diagram

Description automatically generated

***Figura 9: Test Bench del FFDMS***

A screenshot of a computer screen

Description automatically generated

***Figura 10: Waveform del FFDMS***

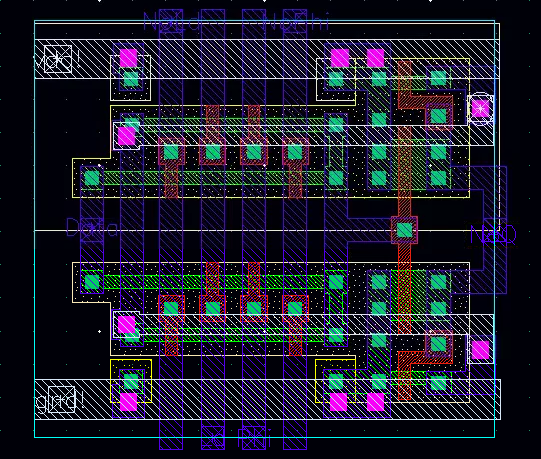
Los resultados del test bench indican un correcto funcionamiento de un Flip-Flop tipo D activo con flanco de subida. Se ve claramente como la entrada “Data” es muestreada y pasa a la salida cada vez que tenemos un flanco de subida de reloj siempre y cuando esté el “Enable” también activo.

***PRÁCTICA 4***

Esta práctica consiste en el desarrollo de los *layout* correspondientes a la practica 1, siguiendo las reglas de diseño, con una ampliación usando el *Phase Splitter* el cualhará la función de sustituir a los inversores usados en el esquemático final del FFDMS*.* Todo el conjunto se unificará creando un flip-flop y tras realizar las conexiones pertinentes, se deberá comprobar su correcto funcionamiento.

*MASTER D*

En primer lugar, creamos el diseño del layout del *MasterD* (figura 11).



***Figura 11: Layout MasterD***

Una vez diseñado el layout se procedo con un chequeo de las reglas de diseño a través de la herramienta de “Assura DRC”. Podemos obsevar un correcto diseño en la imagen inferior.

A screenshot of a computer error

Description automatically generated

***Figura 12: Asura DRC del MasterD***

Una vez comprobado esto se procede a comparar nuestro diseño layout con el esquemático de ese mismo módulo el cual se generó en la práctica 1. Esto se hace mediante la herramienta “Assura LVS”. En el resultado en la siguiente imagen (figura 13) observamos un “Parameter mismatch” lo cual es completamente normal ya que los transistores del trickle se diseñaron con una anchura de canal distinta a la del layout.

A screenshot of a computer

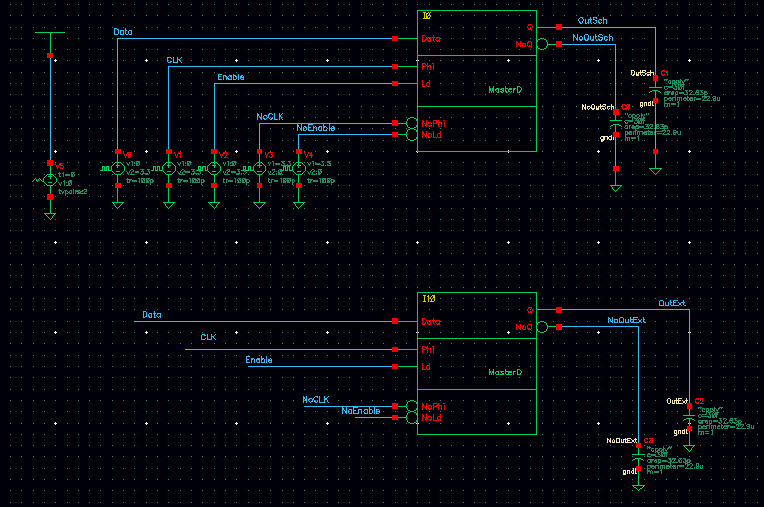
Description automatically generated

***Figura 13: Asura LVS del MasterD***

Una vez comprobado todo esto se procede con generar la vista extraída de nuestro layout para que el programa pueda interpretarlo y posteriormente lo simulemos para ver su funcionamiento. Para ello empleamos la herramienta “Assura-Quantus” siendo importante escribir el nodo de referencia, “gnd!”.

Una vez generada la vista extraída denominada “av\_extracted” comenzamos con el desarrollo de los *test bench.*

Para ello instanciamos el diseño del esquemático con el diseño del layout para poder compararlo en un único *test bench.* (figura 14). Esto comparación se realiza mediante un archivo del tipo “config”.



***Figura 14: Instancia de ambos módulos del MasterD***

Respectivamente simulando obtuvimos el siguiente resultado (figura 15):

A screenshot of a computer

Description automatically generated

***Figura 15: Test bench comparativo del MasterD***

De esta forma podemos interpretar con las salidas de *OutSch* que corresponde al esquemático y *OutExt* correspondiente al layout, que debido a que coinciden las señales, se ha implementado la versión layout de forma adecuada.

*SLAVE D*

Posteriormente empezamos a hacer el diseño del *layout* del *SlaveD* (figura 16), que es el segundo *latch* introducido en el diseño del *flip-flop* activo a nivel bajo y con la peculiaridad de que carece de los pines *LD* Y *NoLD* usados en el MasterD para la señal de “Enable”*.* El diseño fue sencillo simplemente bastó con copiar y pegar el MasterD y realizar un “chop” de las entradas que no se iban a usar y su posterior reajuste mediante un “Stretch”.

A computer screen shot of a computer generated diagram

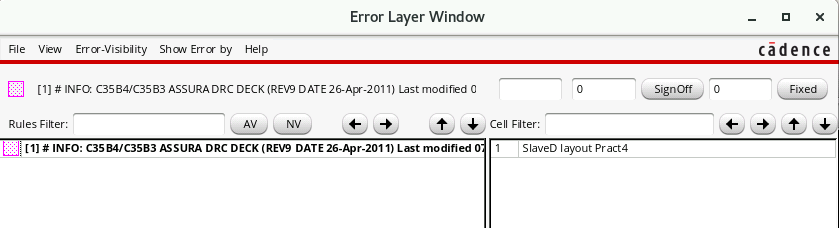
Description automatically generated with medium confidence

***Figura 16: Layout del SlaveD***

Como se puede observar, el *layout* es bastante parecido al del *MasterD* (figura 7). Únicamente se diferencia en que el *SlaveD* carece de dos entradas.

Posteriormente, al igual que con el Master, se realiza el chequeo de reglas de diseño y la comparación con el esquemático antes de generar el *av\_exctracted* para hacer una verificación básica de que se están generando de manera adecuada todos los componentes internos del *layout*. (figura 13)

En la figura 17 observamos que se cumplen todas las reglas de diseño y en la 18 el resultado de la comparación Layout-Esquemático.



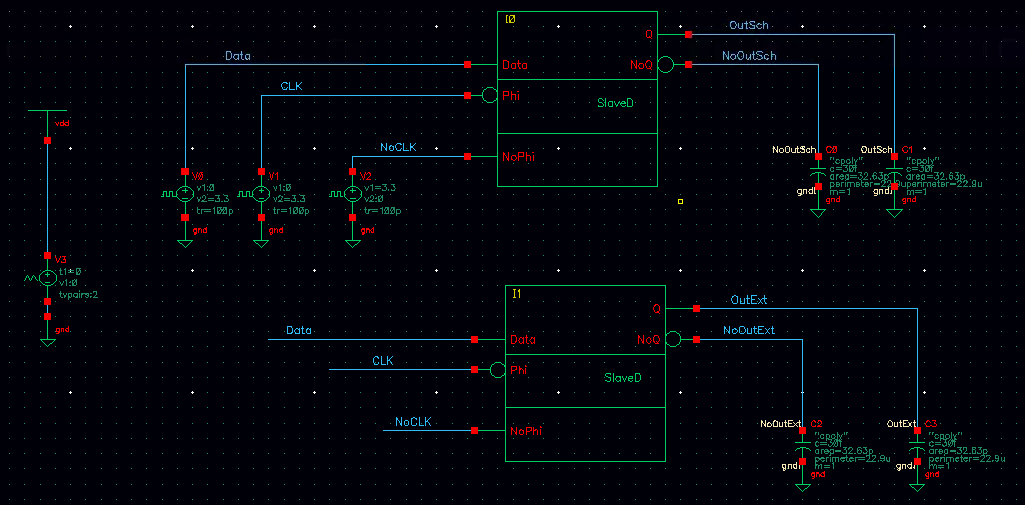
***Figura 17: Asura DRC del SlaveD***

A screenshot of a computer

Description automatically generated

***Figura 18: Asura LVS del SlaveD***

Seguidamente diseñamos el test bench y al igual que con el Slave usando el archivo “config” para poder comparar layout con esquemático en el diseño siguiente:



***Figura 19: Instancia de ambos módulos del SlaveD para el test bench***

Una vez conectado todo, iniciamos el *testbench* y este fue el resultado obtenido: (figura 20 y 21)

A screen shot of a computer

Description automatically generated

A screen shot of a computer

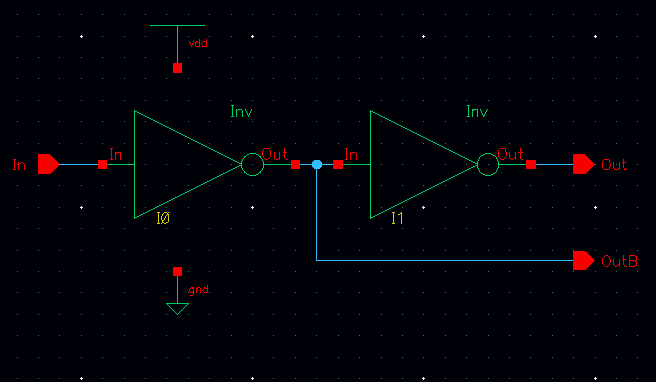
Description automatically generated

***Figuras 20 y 21: Waveform resultado test bench comparativo Slave D***

Al igual que en las comparativas de la simulación realizada previamente, en esta también coinciden las salidas de ambos módulos. Se utilizan estímulos adecuados para un latch a nivel bajo donde observamos como en la primera imagen los esímulos ocurren cuando el reloj vale uno haciendo que éstos no se reflejen en la salida. Lo contrario ocurre en la imagen de debajo donde esto si ocurre ya que los cambios de “Data” ocurren con “CLK” a cero. Hay un pequeño detalle a destacar, y es que la salida de la señal del extracted está mas suavizada, es decir tiene un retardo mayor lo que puede deberse a la modelización de las capacidades parásitas.

*PHASE SPLITTER*

El *Phase Splitter* tiene como función dar la señal negada y sin negar. En nuestro diseño es usado para las conexiones de las distintas entradas ya que en muchos casos necesitamos por ejemplo el reloj negado y sin negar. Aparte es útil para corregir los flancos de subida y estabilizarlos. Este es su respectivo esquemático: (figura 22)



***Figura 22: Esquemático del Phase Splitter***

Tiene un diseño simple, una entrada y dos salidas; una invertida y la otra invertida dos veces. También fue diseñado su símbolo.

A screenshot of a computer screen

Description automatically generatedPosteriormente hicimos el diseño del *layout*: (figura 19)

Se observa que se trata de un layout bastante sencillo. Como de costumbre posteriormente se prosiguió con el chequeo de reglas de diseño (figura 24) y la comparación Layout-Esquemático. Seguidamente se extrajo su vista “av\_extracted” para la simulación del módulo.

***Figura 23: Layout del Phase Splitter***

***A screenshot of a computer

Description automatically generated***

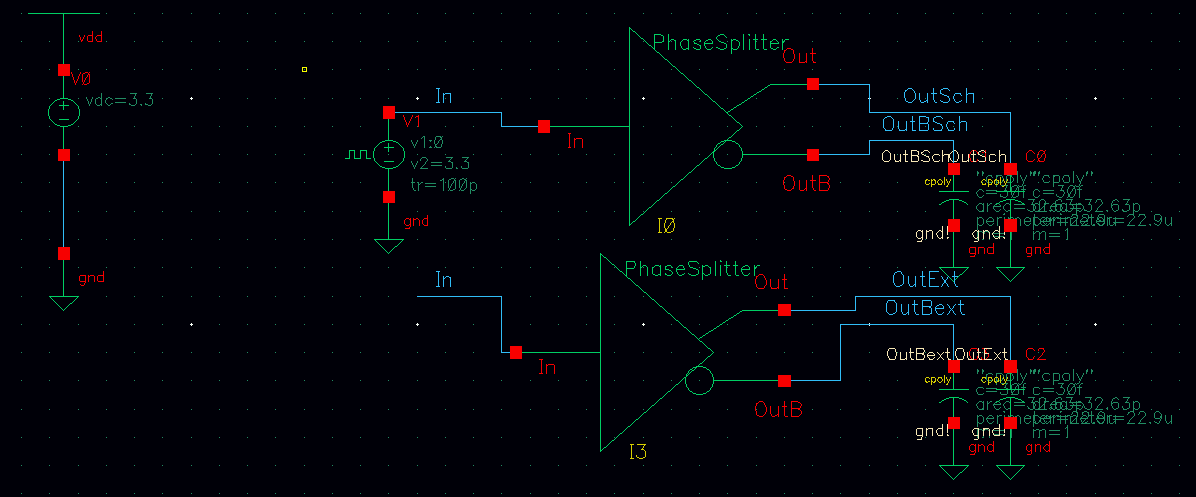
***Figura 24: Assura DRC del Phase Splitter***

***A screenshot of a computer error

Description automatically generated***

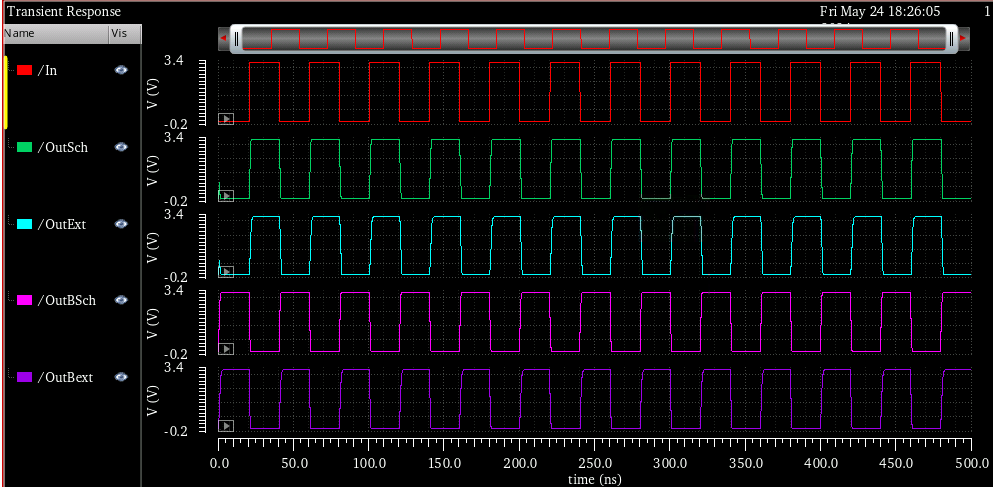
***Figura 25: Assura LVS del Phase Splitter***

En la siguiente imagen se observa el diseño del test bench adecuado mediante el archivo “config” para la comparación de los resultados entre el extraído y el esquemático. (figura 21)

******

***Figura 26: Test bench del Phase Splitter***

Con esto pudimos hacer la simulación del *Phase Splitter* y comprobar su funcionamiento. (figura 27)

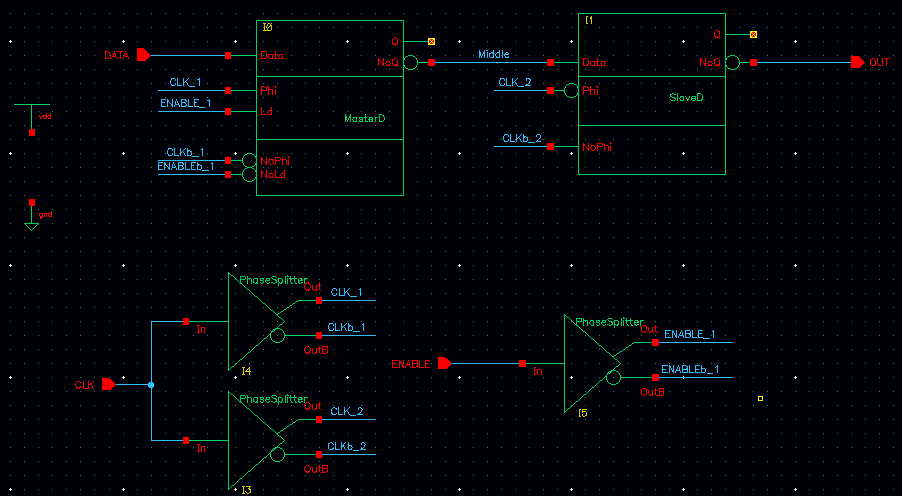


***Figura 27: Waveform comparativo del Phase Splitter***

Como siempre, se comprueban que coinciden las señales de las distintas instancias además de tener sentido el resultado ya que tenemos la señal negada y sin negar en las salidas correspondientes.

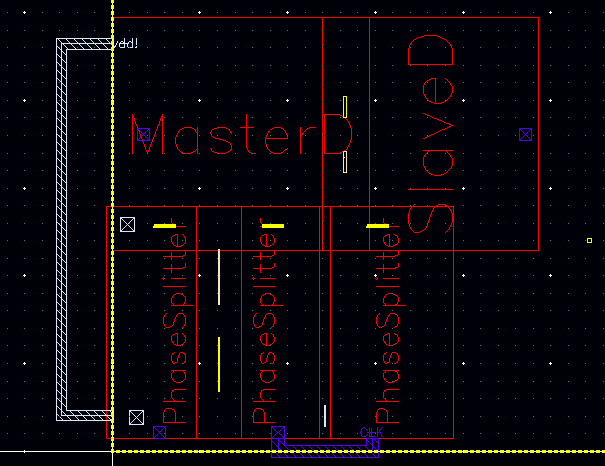
*FFDMSc*

Este es el diseño del *Flip-Flop D* el cual se encuentra en lo más alto de la jerarquía. Está compuesto por todos los anteriores módulos y este es su esquemático. (figura 28)



***Figura 28: Esquemático del FFDMS***

Su estructura es una implementación de los módulos anteriores donde se aprecia la jerarquía. (figura 29) Además se realiza una conexión “ByName” de los pines CLK y GND.



***Figura 29: Estructura del layout FFDMSc***

El layout definitivo del FFDMSc: (figura 30)

A computer drawing of a circuit board

Description automatically generated

***Figura 30: layout del FFDMSc***

Este layout al igual que los previos se le ejecutó el Assura-DRC y Assura LVS (Figuras 31 y 32). En el LVS observamos 3 “Parameter Mismatches” lo cual tiene sentido debido a lo comentado anteriormente en referencia al transistor dimensionado de otra forma en las 3 instancias distintas.

***A screenshot of a computer

Description automatically generated***

***Figura 31: ASSURA DRC del FFDMSc***

A screenshot of a computer error

Description automatically generated

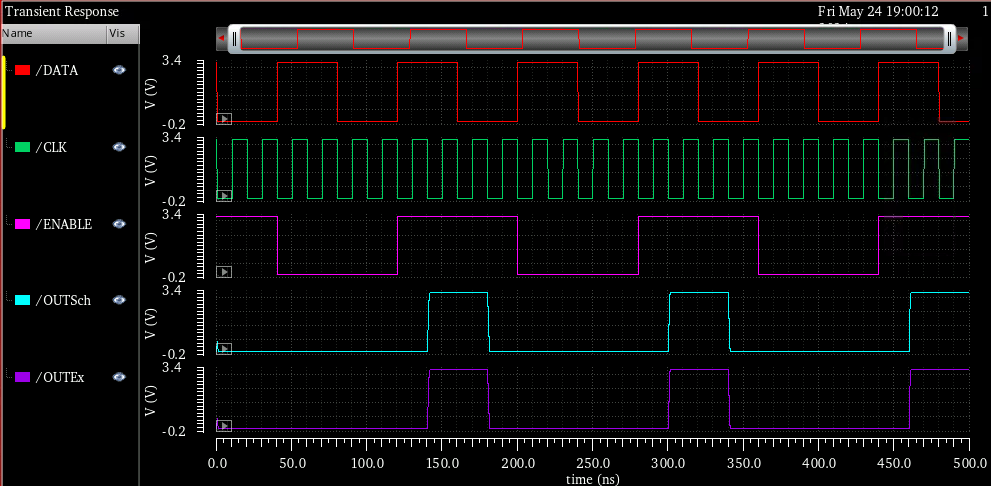
***Figura 32: ASSURA LVS del FFDMS***

Como es costumbre implementamos las instancias en el *testbench* y la configuración para la comparativa mediante el archivo “config”*.* (figura 33)



***Figura 33: Test bench comparativo de ambos módulos.***

Por ende, realizamos su respectiva simulación. (figura 34)



***Figura 34: Testbench del FFDMSc***

Observamos unos resultados prometedores que reafirman un correcto diseño de los módulos previos donde en primer lugar observamos como coinciden a la perfección esquemático y extraído además de ver claramente como el flip flop tipo D cumple con su función y en cada flanco de subida, siempre y cuando el “Enable” esté activado, la entrada “Data” pasa a la salida “Out”.

*Conclusiones*

Podemos concluir que nuestro diseño se ha realizado de manera satisfactoria debido a la coincidencia de los distintos test bench comparativos además de haber podido observar como cada módulo realiza la función lógica correspondiente. Podríamos estar aun mas seguros del correcto diseño del Flip Flop poniendo los módulos al límite y buscando estímulos que pudieran haber puesto en duda el correcto funcionamiento. Es por ello que creemos que nos ha faltado una verificación aun mas exhaustiva.