MEMORIAS PRÁCTICAS 1 Y 4

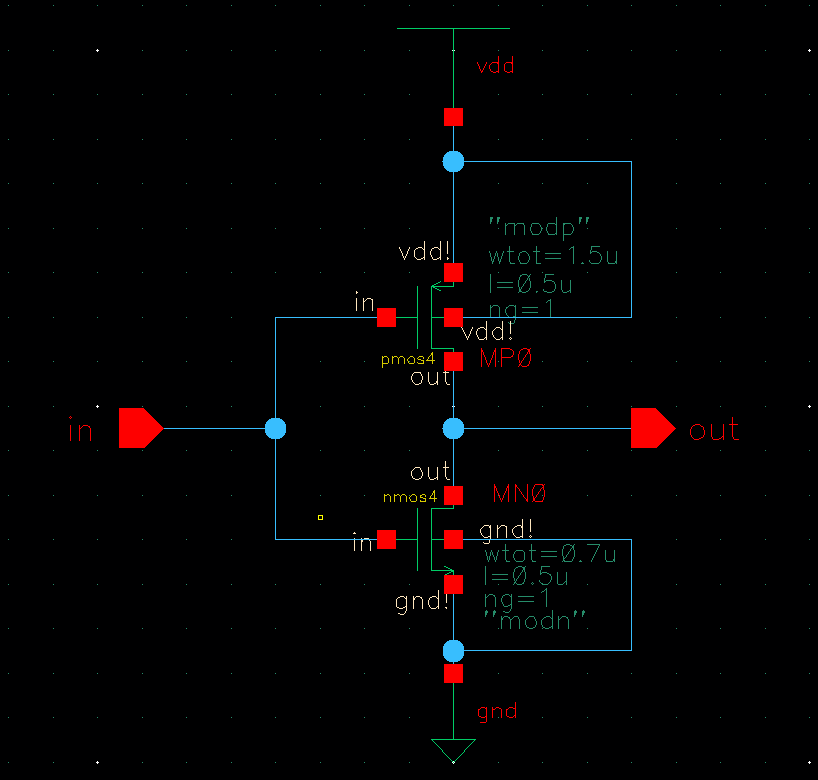
Jaime Lloret Cuñat

Adam Cecetka Ortiz

PRÁCTICA 1

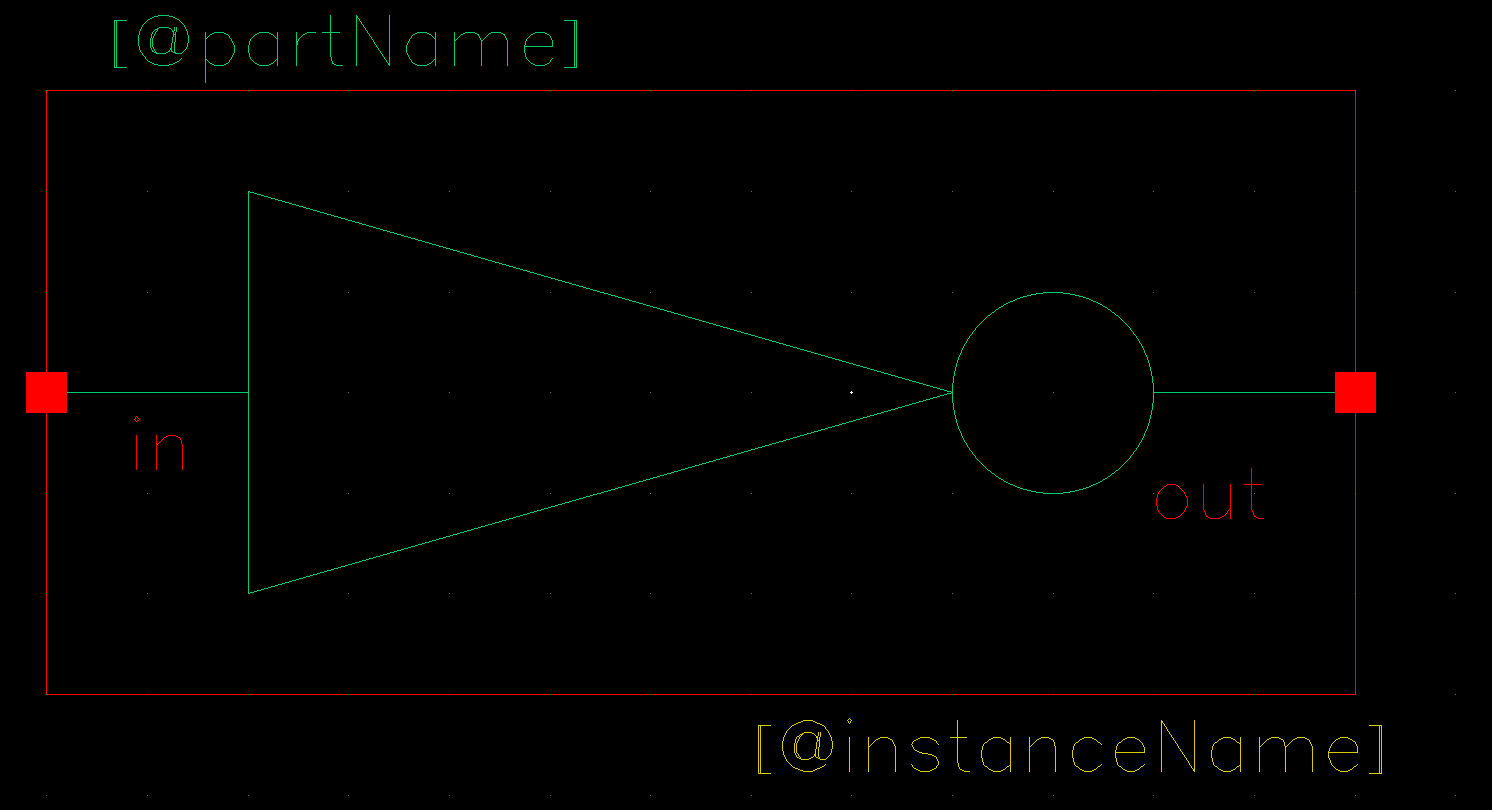
TRICKLE

Primeramente, empezamos haciendo el diseño del *Trickle* (figura 1); el cual tiene como función invertir la señal que entra. Está compuesto por dos transistores, uno *pmos* y uno *nmos*.



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

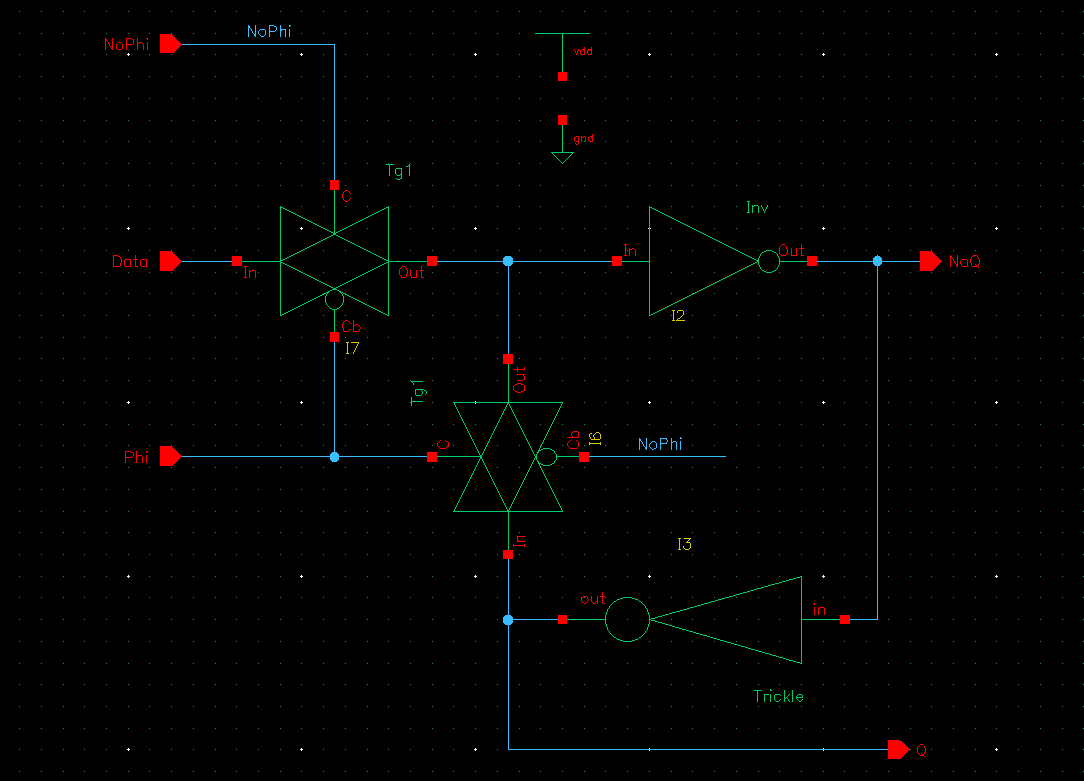
Una vez terminado hicimos su representación simbólica (figura 2) para poder usarlo en futuras instancias.



***Figura 2: Símbolo del Trickle***

SLAVE D

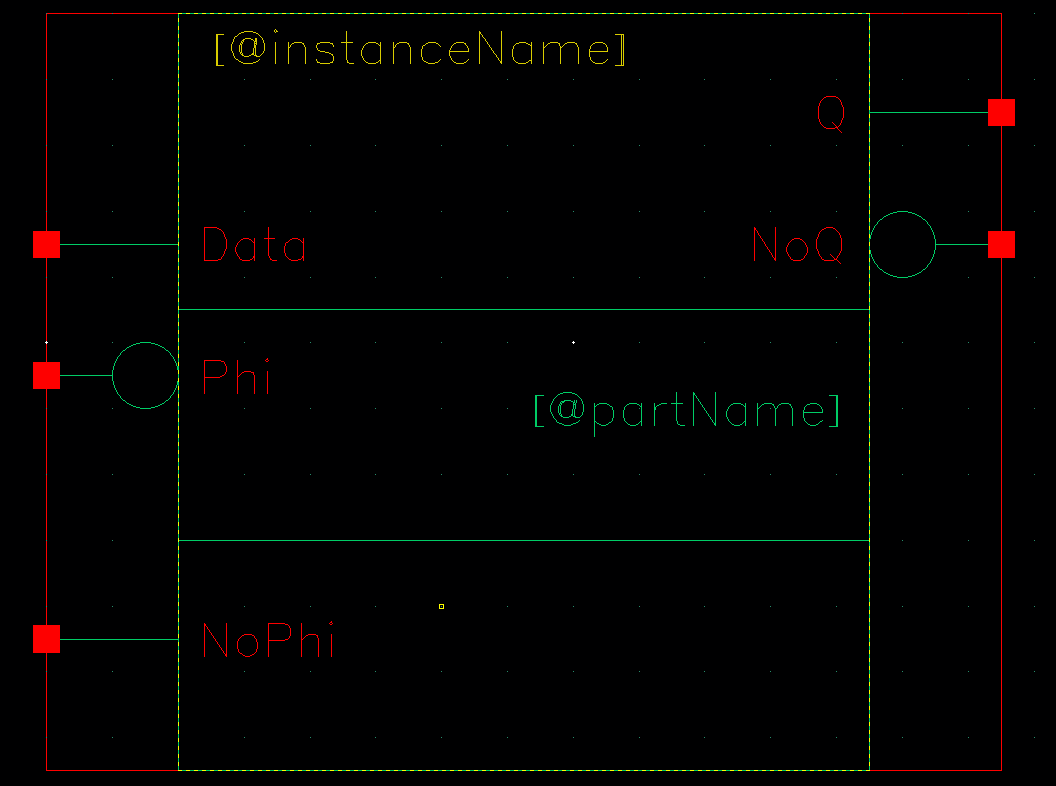
Posteriormente hicimos el *SlaveD* en otro archivo distinto, lo primero que diseñamos fue el esquemático (figura 3).



***Figura 3: Esquemático del SlaveD***

Este está compuesto por dos multiplexores un inversor y la instancia del *Trickle* implementado anteriormente (figura 2).

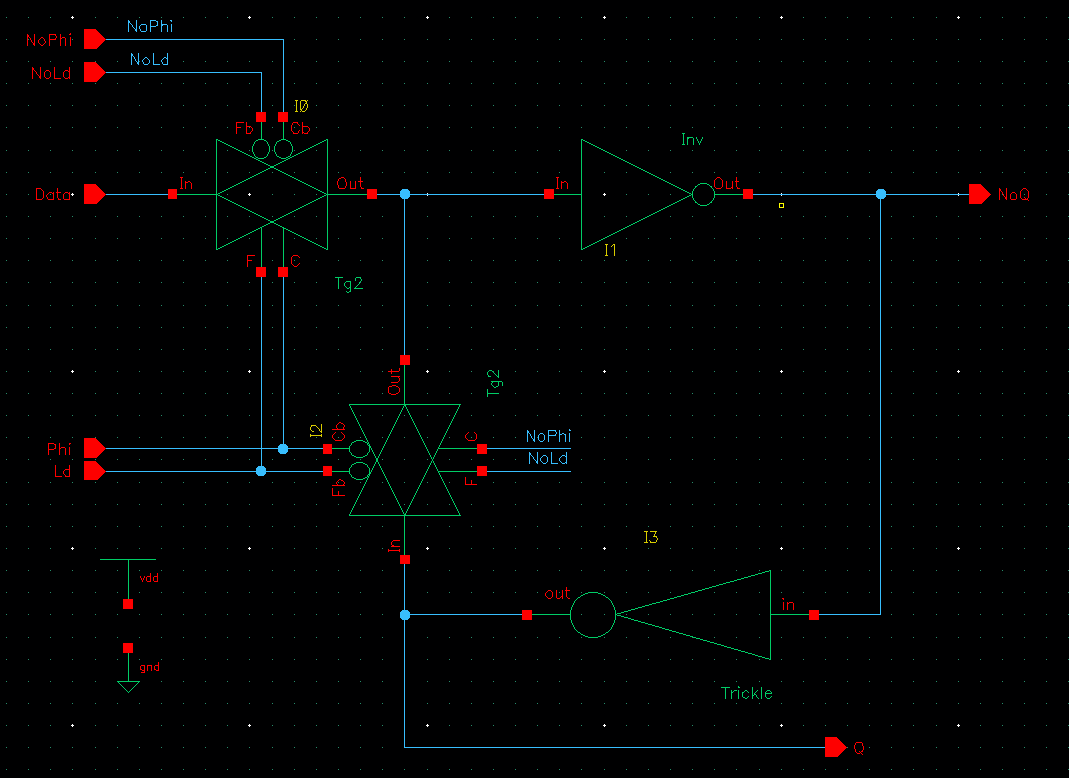
Por otro lado, implementamos el símbolo del *SlaveD* (figura 4).



***Figura 4: Símbolo del SlaveD***

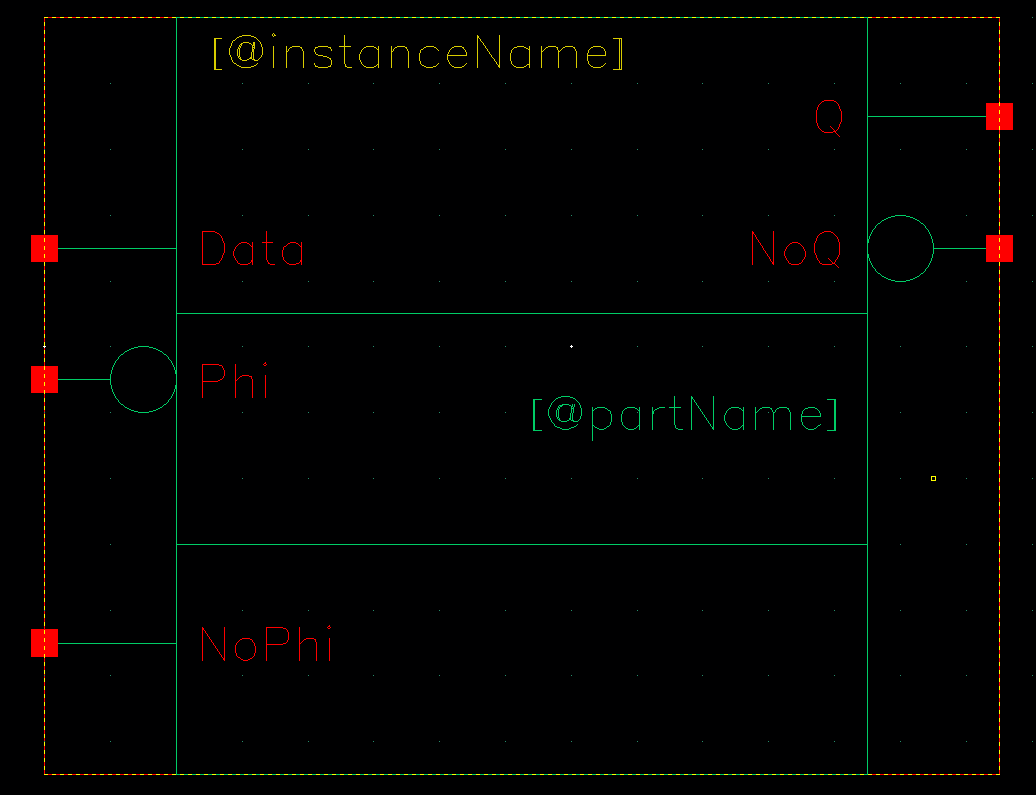
MASTER D

Seguidamente realizamos el diseño del *MasterD* (figura 5), con su respectivo esquemático. El cual está compuesto por dos multiplexores un *Trickle* y un inversor.



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

También hemos hecho el respectivo símbolo del *MasterD* (figura 6).



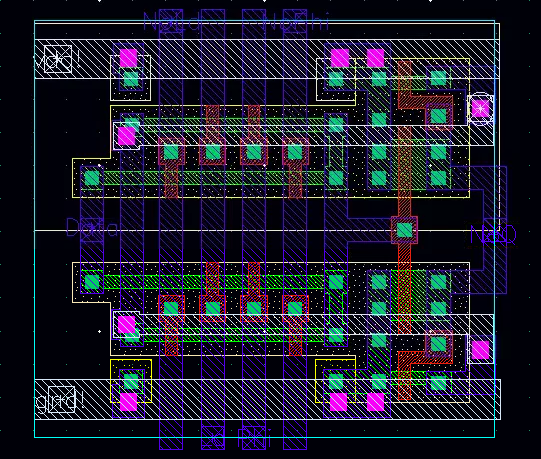
***Figura 6: Símbolo de MasterD***

PRÁCTICA 4

MASTER D

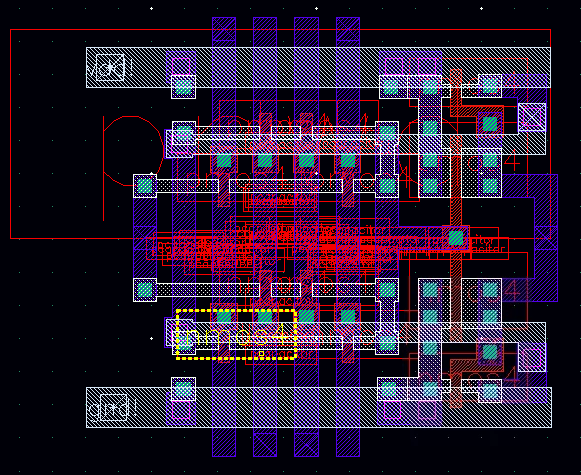
Esta práctica consiste en el desarrollo de los *layout* correspondientes a la practica 1, siguiendo las reglas de diseño, con una ampliación del *Phase Splitter*. Todo el conjunto se unificará creando un flip-flop y tras realizar las conexiones pertinentes, se deberá comprobar su correcto funcionamiento.

En primer lugar, creamos el diseño del layout del *MasterD* (figura 7).



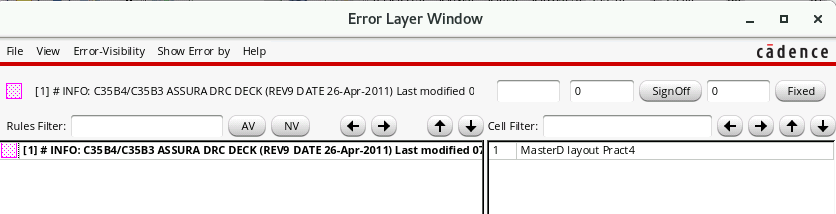
***Figura 7: Layout MasterD***

Simultáneamente se iba generando el *av\_exctracted* (figura 8), que era bastante útil para comprobar el desarrollo del layout con sus respectivos componentes.



***Figura 8: Exctracted del MasterD***

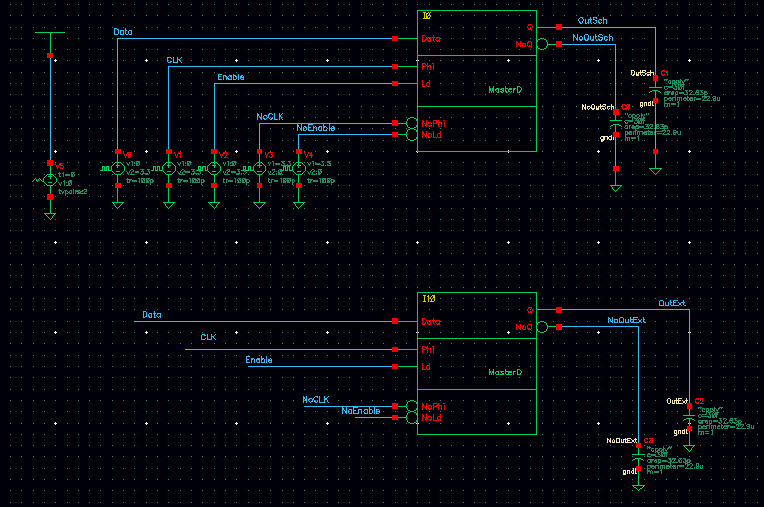
Así pues, iniciamos un *checkeo* de los errores del *layout* mediante la herramienta de *Asura Run DRC*. (figura 9)



***Figura 9: Asura Run DRC del MasterD***

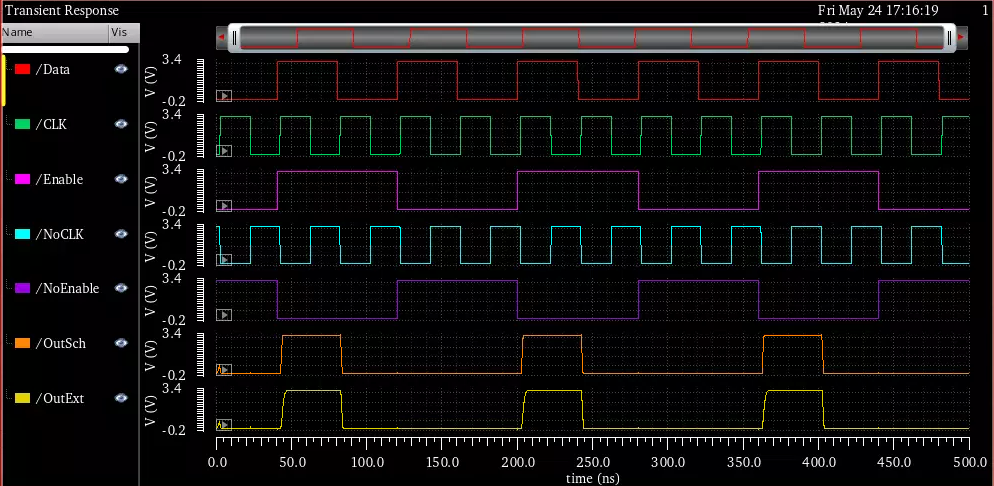
Una vez comprobado que no teníamos errores y cumplíamos todas las reglas de diseño comenzamos con el desarrollo de los *test bench.*

Para ello instanciamos el diseño del esquemático con el diseño del layout para poder compararlo en un único *test bench.* (figura 10)



***Figura 10: Instancia de ambos módulos del MasterD***

Respectivamente simulando obtuvimos el siguiente resultado (figura 11):

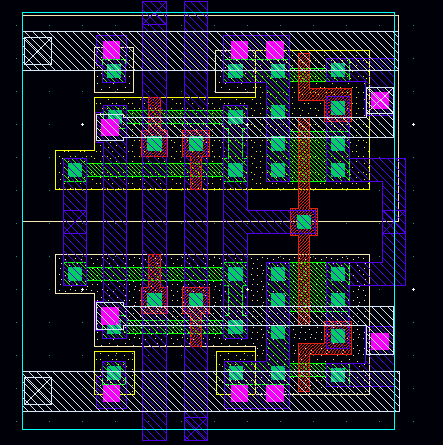


***Figura 11: Test bench comparativo del MasterD***

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 han implementado ambos módulos de forma adecuada.

SLAVE D

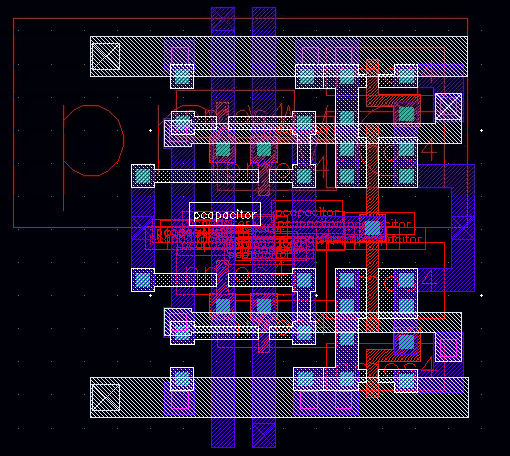
Posteriormente empezamos a hacer el diseño del *layout* del *SlaveD* (figura 12), que es el segundo *latch* introducido en el diseño del *flip-flop* activo a nivel bajo y con la peculiaridad de que carece de los pines *LD* Y *NoLD.*



***Figura 12: Layout del SlaveD***

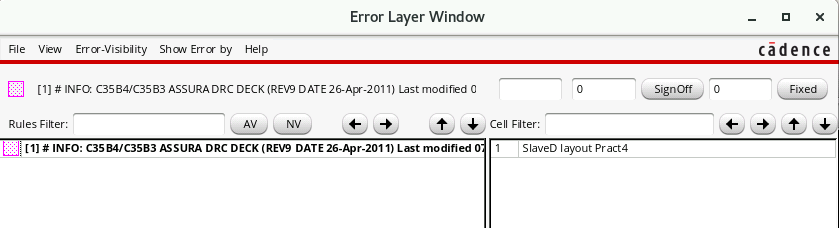
Como se puede observar, el *layout* es bastante parecido al del *MasterD* (figura 7). Únicamente se diferencia en que el *SlaveD* carece de dos entradas.

Posteriormente se realiza el *av\_exctracted* para hacer una verificación básica de que se están generando de manera adecuada todos los componentes internos del *layout*. (figura 13)



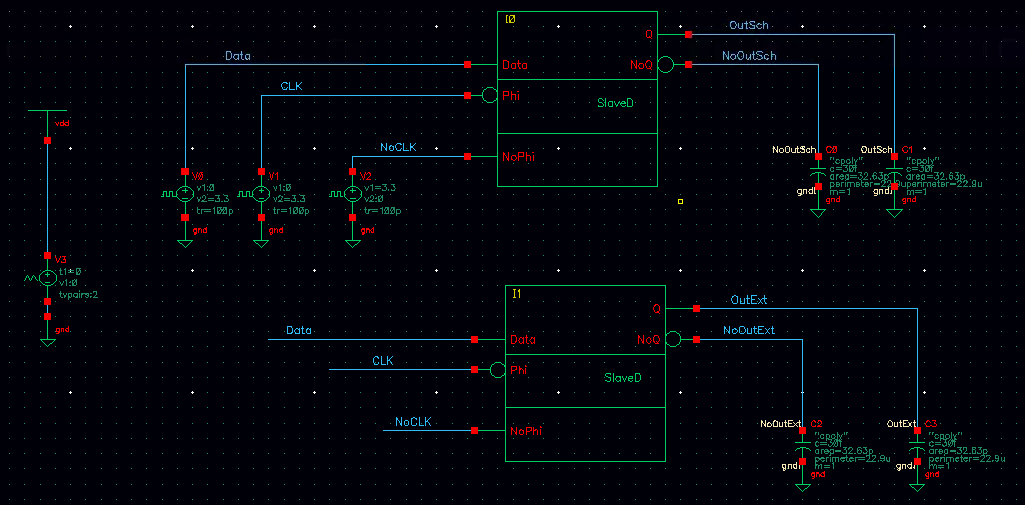
***Figura 13: Exctracted del SlaveD***

Al igual que en el *MasterD*, realizamos un testeo de que todas las reglas de diseño se estuviesen cumpliendo adecuadamente. (figura 14)



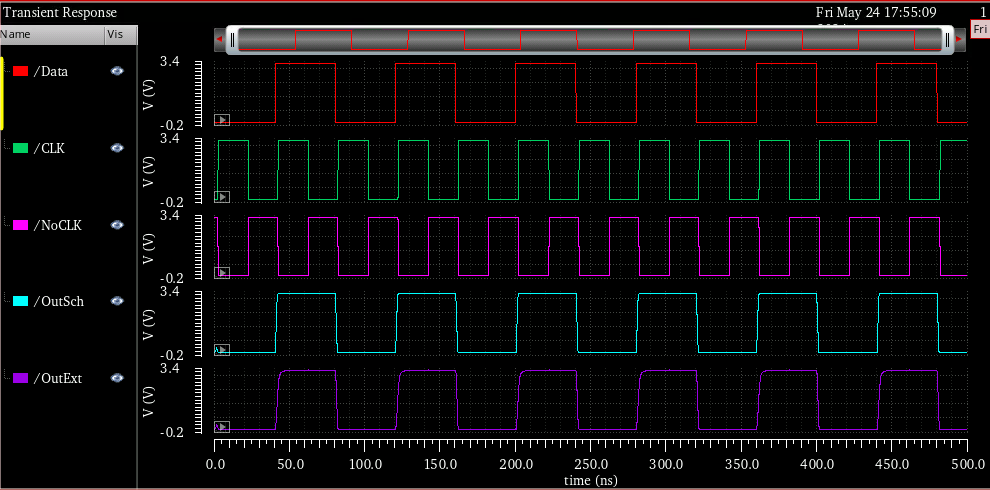
***Figura 14: Asura Run DRC del SlaveD***

Después de verificar que todo estuviera en orden, como ya hicimos anteriormente, instanciamos los módulos del esquemático y del *layout* en un mismo diseño, para poder hacer un *test bench* comparativo para que las señales de ambos coincidan. (figura 15)



***Figura 15: Instancia de ambos módulos del SlaveD***

Una vez conectado todo, iniciamos el *testbench* y este fue el resultado obtenido: (figura 16)

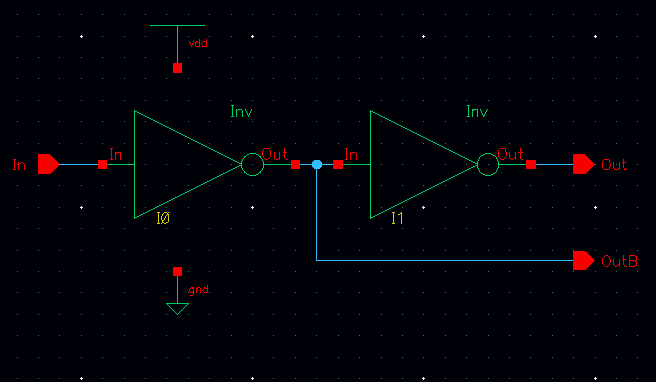


***Figura 16: Test bench comparativo del SlaveD***

Al igual que en las comparativas de la simulación realizada previamente, en esta también coinciden las salidas de ambos módulos. Hay un pequeño detalle a destacar, y es que el pico de la salida del *layout* está más ablandado.

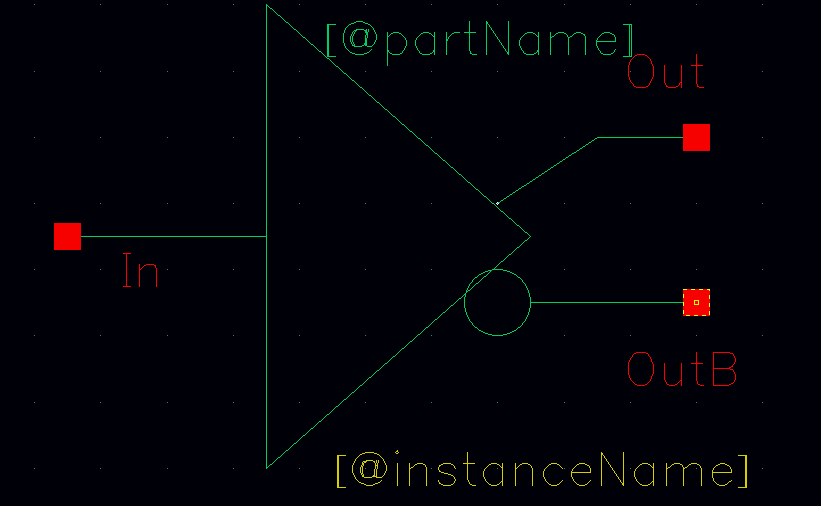
PHASE SPLITTER

El *Phase Splitter* tiene como función conectar los *latches* del *MasterD* y el *SlaveD*. Aparte es útil para corregir los flancos de subida y estabilizarlos. Este es su respectivo esquemático: (figura 17)



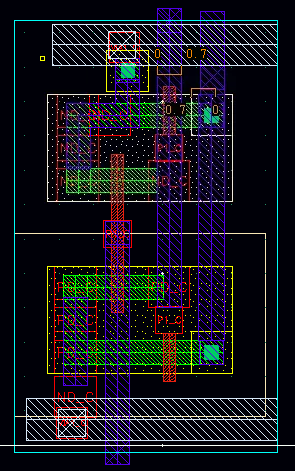
***Figura 17: Esquemático del Phase Splitter***

Tiene un diseño simple, una entrada y dos salidas; una invertida y la otra invertida dos veces. El símbolo es el siguiente. (figura 18)



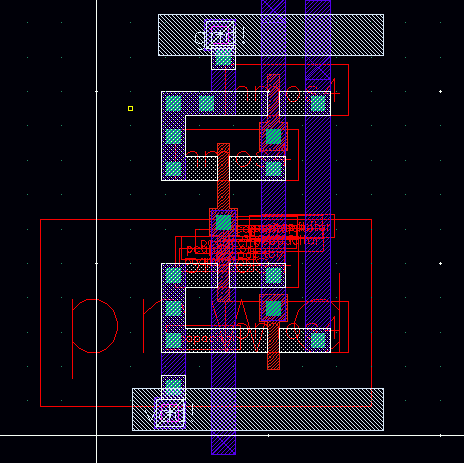
***Figura 18: Símbolo del Phase Splitter***

Posteriormente hicimos el diseño del *layout*: (figura 19)



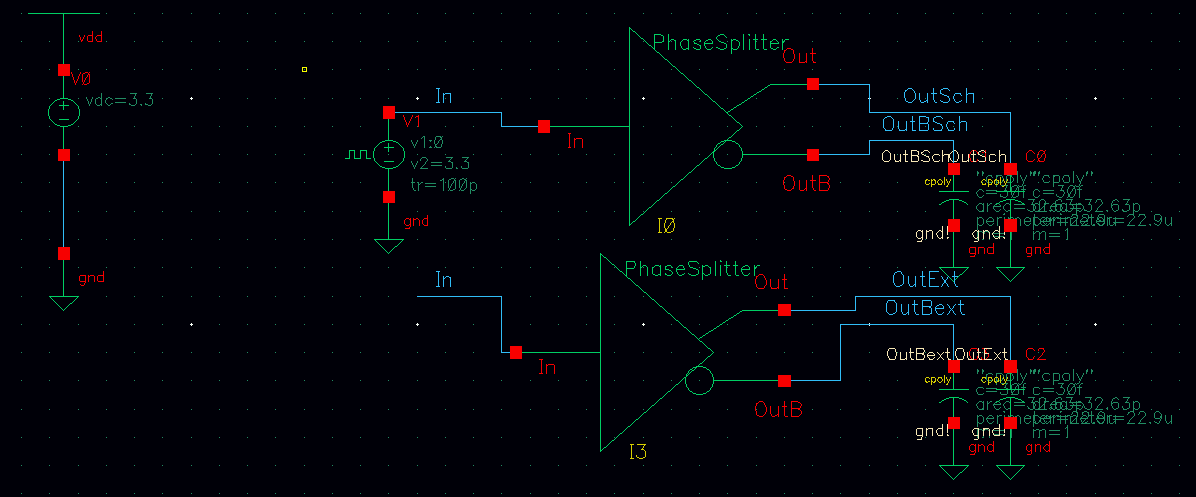
***Figura 19: Layout del Phase Splitter***

De la misma manera se genera también la extracción. (figura 20)



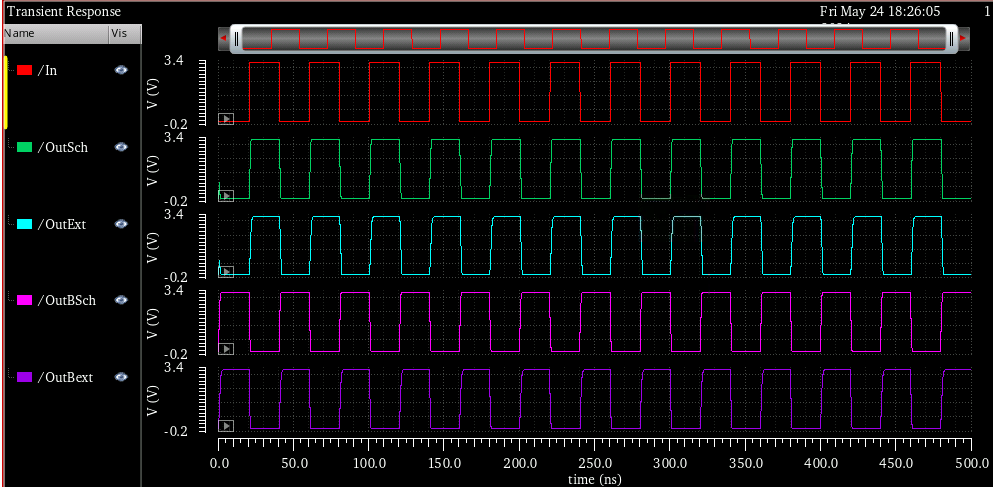
***Figura 20: Exctracted del Phase Splitter***

Seguidamente hicimos la instancia de los módulos para poder compararlos en un *testbench*. (figura 21)

******

***Figura 21: Instancia de ambos módulos del Phase Splitter***

Con esto pudimos hacer la simulación del *Phase Splitter*. (figura 22)

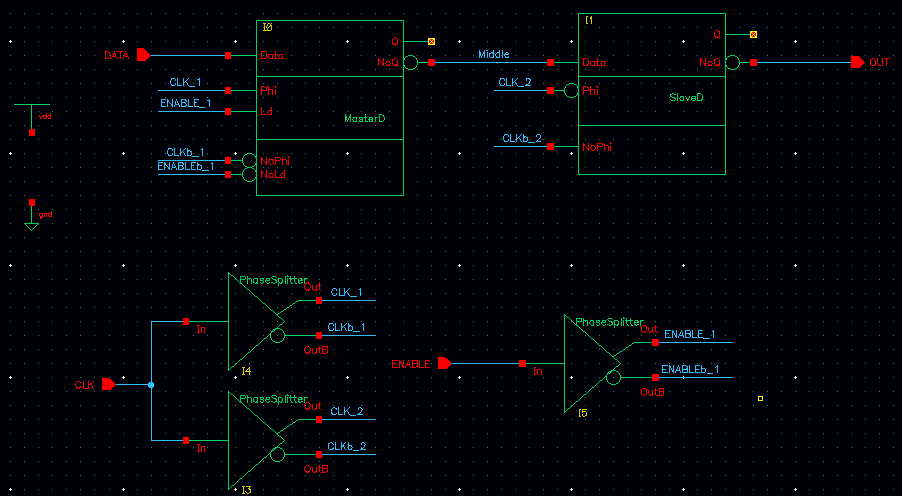


***Figura 22: Test bench comparativo del Phase Splitter***

Como siempre, se comprueban que coinciden las señales de las distintas instancias.

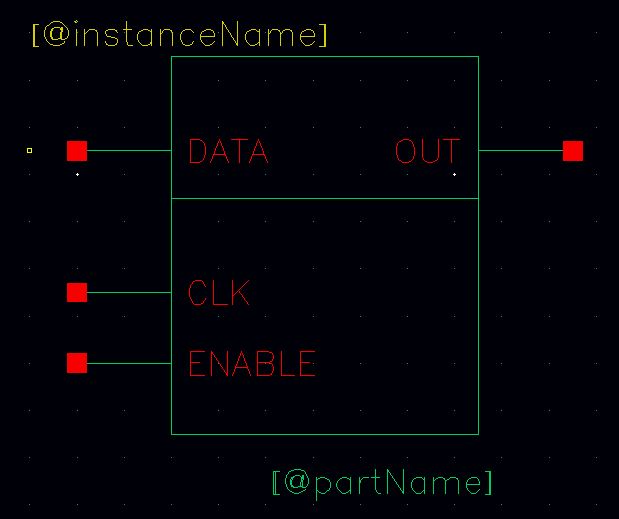
FFDMS

Este es el diseño del *flip-flop*, el cual se encuentra en lo más alto de la jerarquía. Está compuesto por todos los anteriores módulos y este es su esquemático. (figura 23)



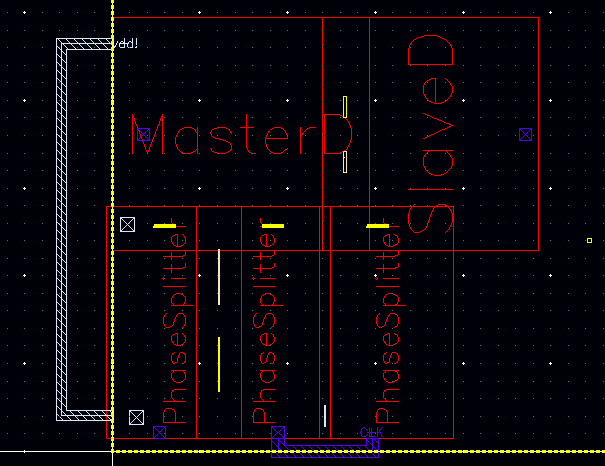
***Figura 23: Esquemático del FFDMS***

Su símbolo es como el de un *flip-flop* tradicional. (figura 24)



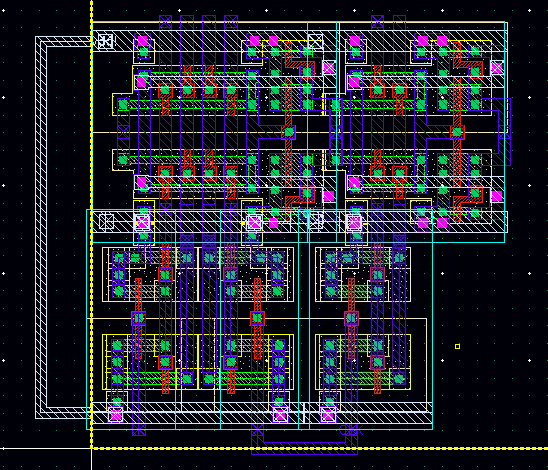
***Figura 24: Símbolo del FFDMS***

Su estructura es una implementación de los módulos anteriores. (figura 25)



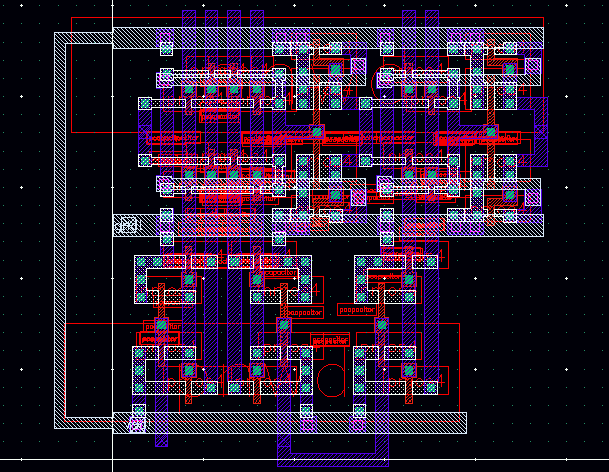
***Figura 25: Estructura del layout***

El layout definitivo del FFDMS: (figura 26)



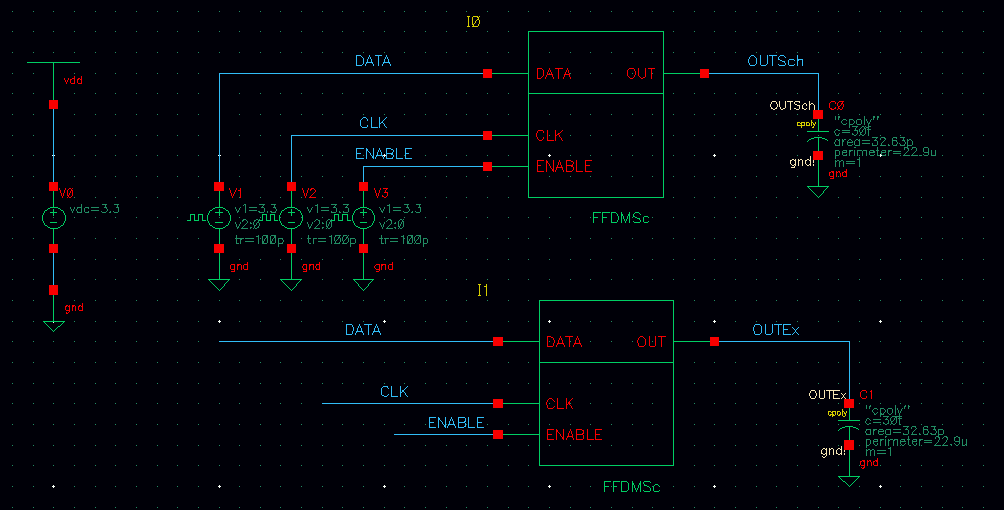
***Figura 26: layout del FFDMS***

Así pues, también podemos fijarnos en el *av\_exctracted.* (figura 27)



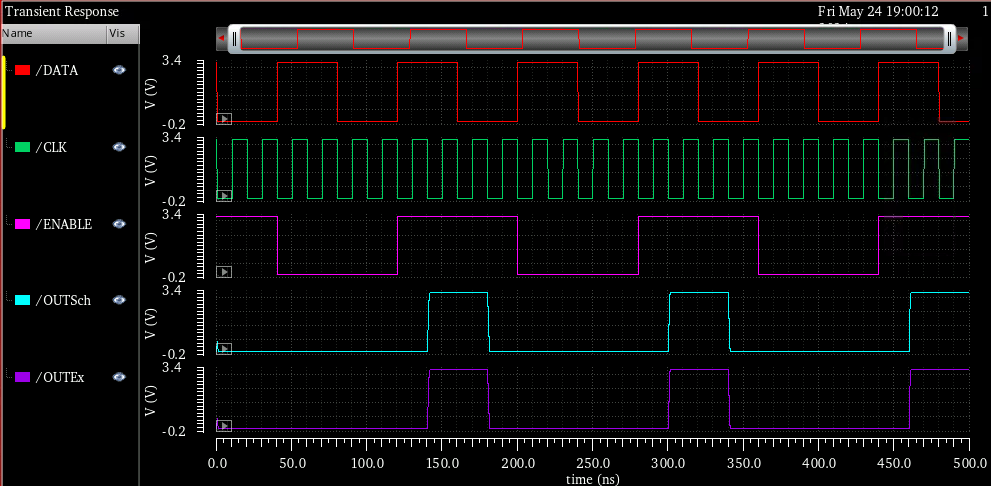
***Figura 27: av\_exctracted del FFDMS***

Como es costumbre implementamos las instancias en el *testbench.* (figura 28)



***Figura 28: Instancia de ambos módulos.***

Por ende, realizamos su respectiva simulación. (figura 29)



***Figura 29: Testbench del FFDMS***