UNIVERSIDAD POLITÉCNICA DE VALENCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

Grado de Ingeniería de Tecnologías y Servicios de Telecomunicación

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

“Prácticas de VLSI”

INFORMES DE LAS PRÁCTICAS 1 Y 4

Autores:
Jara García Barrena
Luis Valls Sansaloni

INTRODUCCIÓN

En la presente memoria se desarrollan las explicaciones del método utilizado para llevar a cabo las Prácticas 1 y 4 de la asignatura de VLSI. El trabajo ha sido realizado por Jara García Barrena y Luis Valls Sansaloni.

En el archivo comprimido adjunto, se añaden todos los ficheros creados a lo largo de las prácticas. La librería donde se ha ido guardando se llama "Pract1". La forma de organizar los ficheros ha sido simple. Para las células que contienen esquemático, símbolo, layout y su av_extracted, el nombre de la carpeta donde se guardan tienen como denominación el nombre de la célula correspondiente. En el caso de los Test Bench, la carpeta contiene el esquemático de dicho diseño, renombrada con el nombre de su célula correspondiente más el sufijo "_TB". Por último, los esquemáticos de Test Bench que comparan el esquemático de una célula con su av_extracted, están guardados en carpetas con el nombre de la célula que simulan, seguido del sufijo "_TB2". También se añaden a estas últimas la carpeta "config" (configuración que nos permite situar el esquemático y el ac_extracted en un mismo tiempo y espacio).

PRÁCTICA 1: DISEÑO ELÉCTRICO Y SIMULACIÓN DE UN FLIP-FLOP MASTER-SLAVE TIPO D

Jara García Barrena
jagarba3@teleco.upv.es
Luis Valls Sansaloni
luivalsa@teleco.upv.es

22 de mayo de 2021

1. INTRODUCCIÓN

En esta primera práctica hemos realizado el diseño de un **"Flip-Flop *Full custom* Master-Slave tipo D"** pseudostático (precisa de los dos relojes ϕ y $/\phi$ para el refresco de datos), en tecnología AMS TECH_C35B4, CMOS de pozo N, un polisilicio y dos metales, de 0.35 μm de mínima longitud dibujada de puerta, mediante Cadence. Para ello, hemos utilizado para realizar el diseño el proceso *Bottom-Up*, es decir estrategia de abajo a arriba, con niveles de complejidad crecientes.

Los niveles de Jerarquía de abajo a arriba han sido:

1. Nivel 0 (Primitivas de diseño): En este nivel hemos utilizado componentes de la librería de diseño PRIMLIB (en particular los transistores nmos4 y pmos4).
2. Nivel 1 y 2 (Jerarquía de Símbolos): Donde se asocia a cada símbolo un esquema que es su representación (los símbolos también se almacenan en librerías). Hemos empleado símbolos como Tg1, Tg2 e Inv creados en la librería PIEZAS_DIG de la asignatura y además, hemos creado nuevos como son el Trickle, MasterD y SlaveD en una nueva librería llamada Pract1 (hemos verificado su funcionamiento mediante Test_Benches.
3. Nivel 3 (Jerarquía de Símbolos): Finalmente hemos creado el FFDMS uniendo los símbolos creados anteriormente y hemos comprobado su correcto funcionamiento (mediante un Test_Bench).

2. INVERSOR TRICKLE

Para el diseño del inversor Trickle hemos utilizado los componentes nmos4 y pmos4 de la librería PRIMLIB asi como los componentes vdd y gnd de la librería ANALOGLIB. Hemos supuesto que Trickle tendrá transistores MOS de menor ganancia así que hemos reducido el ratio W/L tomando un L = 0.5 μm en vez de su valor mínimo, 0.35 μm. Podemos observar su esquema en la Figura 1 y el simbolo que le hemos asociado en la Figura 2.

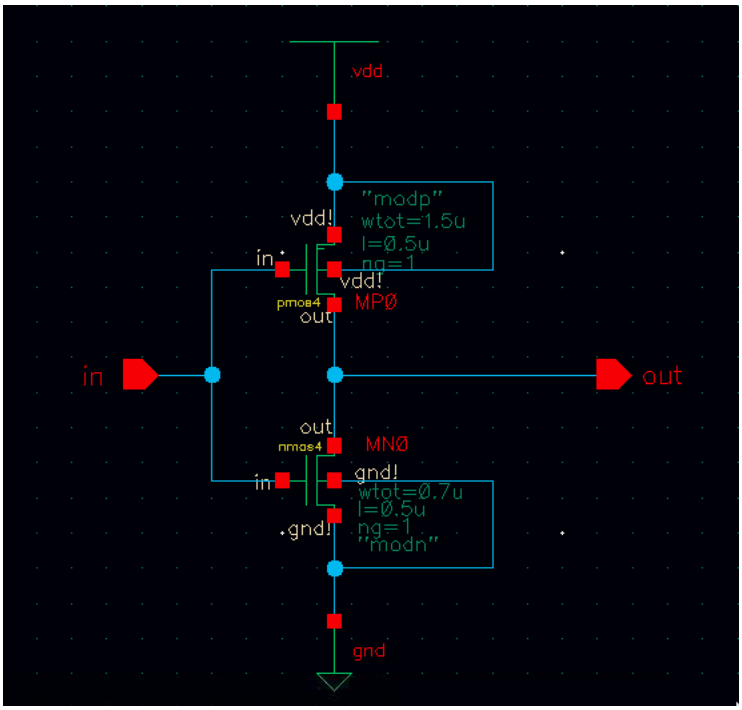

Figura 1: Esquema del Inversor Trickle

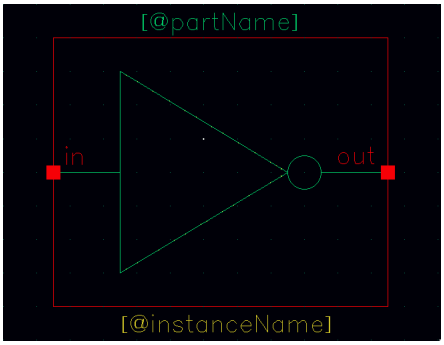

Figura 2: Símbolo Trickle

Ahora bien, tras la realización del diseño (y del símbolo) queríamos comprobar su correcto funcionamiento, así que realizamos una la simulación transitoria de un testbench. En la Figura 3 podemos observar el esquema propuesto para tal simulación. Como podemos ver, además de los componentes utilizados anteriormente (gnd, vdd...), también utilizamos los componentes vpulse, vdc y cap de la librería ANALOGLIB así como el componente a simular Trickle creado por nosotros en la librería Pract1. Seguidamente, arrancamos el entorno de simulación y configuramos el tipo de análisis a realizar ,que en nuestro caso es una simulación de tipo transitoria (tran). Otra parámetro que hemos habilitado es el skipdc para partir en t=0 de las condiciones iniciales y no buscar un innecesario punto de polarización. En la Figura 4 podemos visualizar el correcto funcionamiento del inversor ya que, la salida (out) es contraria a la entrada (in).

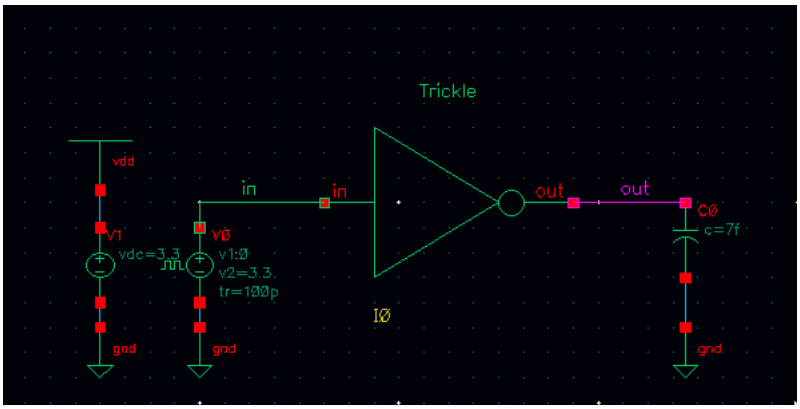

Figura 3: TB_Trickle

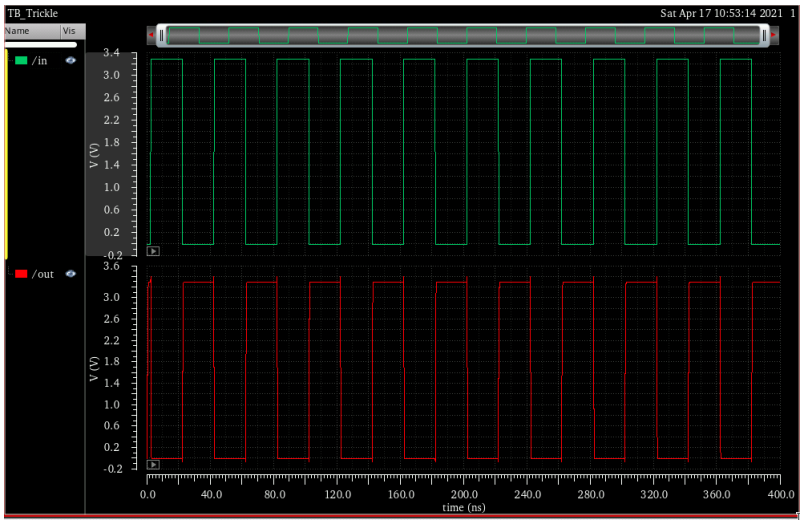

Figura 4: Simulación Trickle

3. LATCHES

Como bien sabemos, el Flip-Flop D Master-Slave está compuesto por 2 latches, uno denominado Master y otro llamado Slave, cuyas funciones observaremos más adelante.

3.1. MASTER D

Empezamos explicando el diseño del esquema del Latch, transparente a nivel alto, con Enable (Ld) Master D. Para su diseño hemos utilizado 2 componentes que no habíamos usado anteriormente, el Tg2 y el Inv, ambos disponibles en la librería PIEZAS_DIG de la asignatura y los hemos conectado con nuestro Trickle (previamente simulado). Este esquema lo podemos ver el la Figura 5. Podemos observar en el esquema que hemos añadido los objetos de potencia vdd y gnd en el diseño, esto lo hemos hecho por coherencia. También hemos creado la vista symbol del MasterD, como podemos observar en la Figura 6.

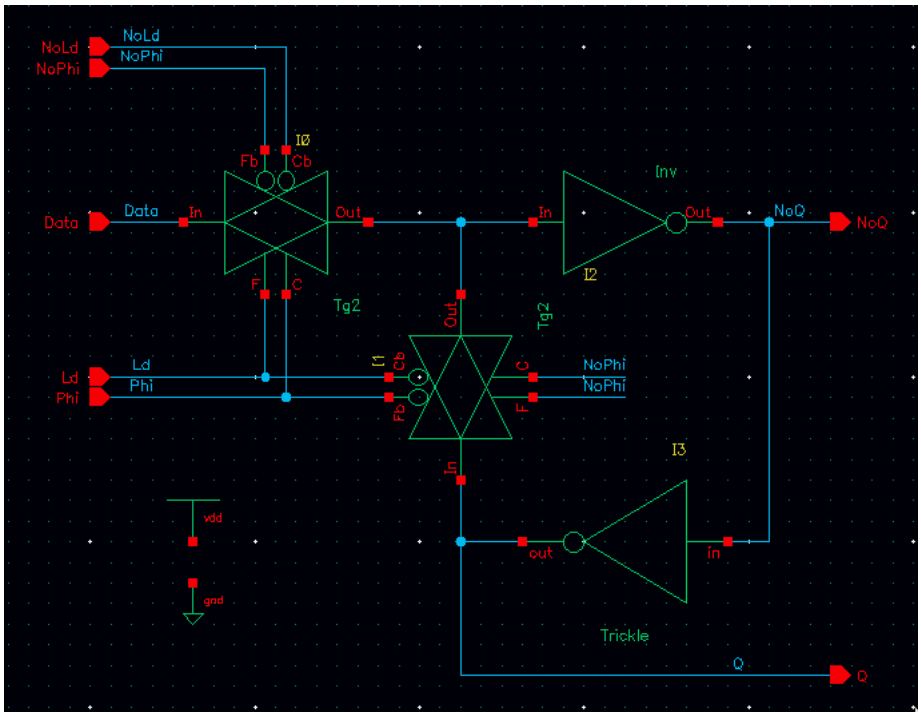

Figura 5: Esquema del Inversor MasterD

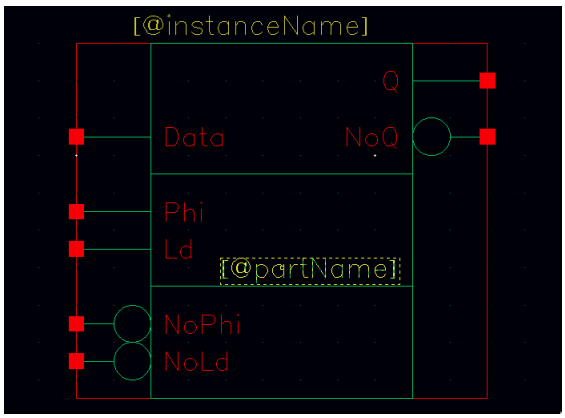

Figura 6: Símbolo MasterD

Tras su diseño, hemos realizado su simulación estimulando las 3 entradas del dispositivo (Datos, CLK y Enable). En la Figura 7 podemos visualizar el esquema propuesto para dicha finalidad, y en la Figura 8 su respuesta transitoria. Mediante el estudio de dicha gráfica, observamos que el Master D funciona correctamente ya que, solo funciona cuando el Enable esta activado y solo actualiza los datos en el flanco de subida del reloj (CLK) estando su salida invertida.

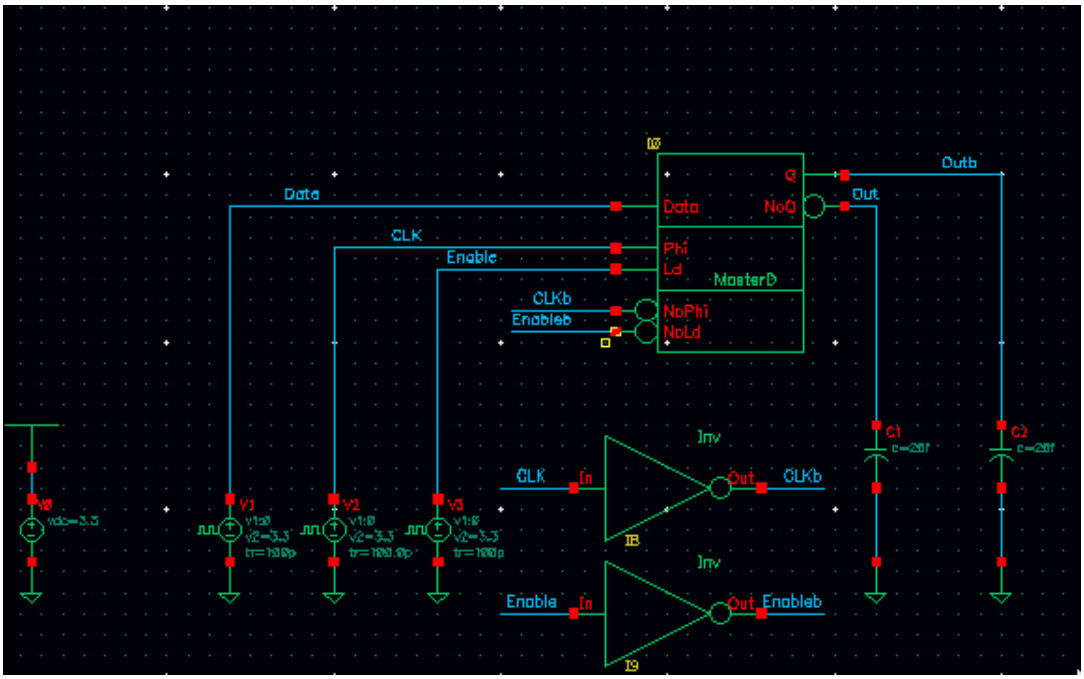

Figura 7: $TB\_MasterD$

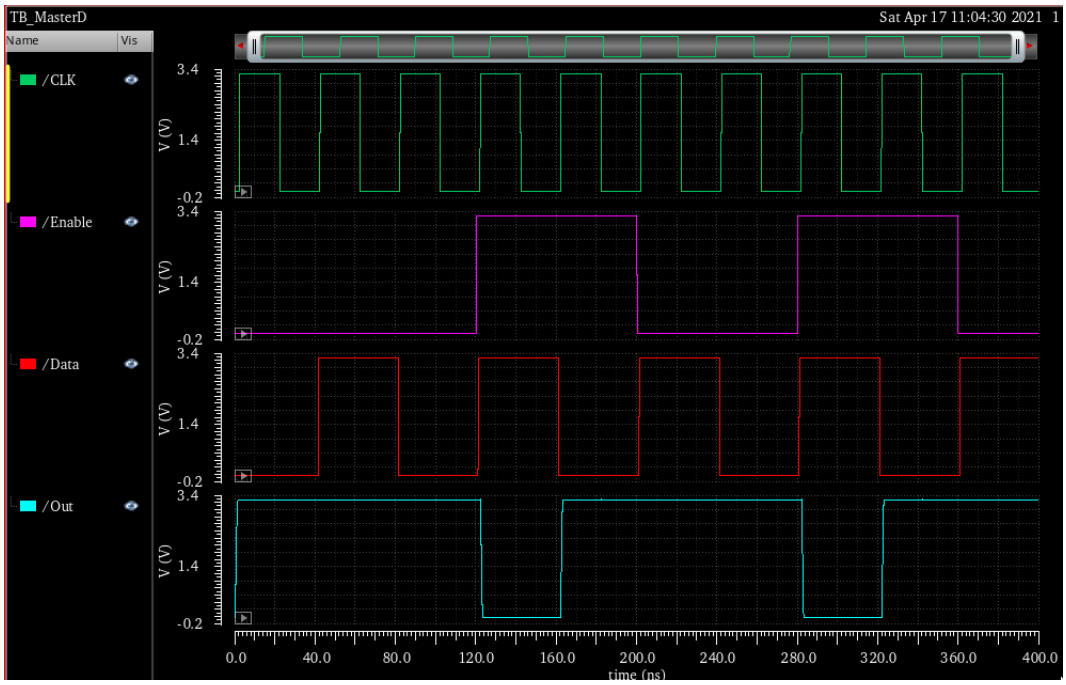

Figura 8: Simulación MasterD

3.2. SLAVE D

El esquema del Latch, transparente a nivel bajo, SlaveD lo podemos visualizar en la Figura 9, y su esquema en la Figura 10. Podemos observar que tiene un parecido con el Latch Master, pero esta vez no tiene Enable.

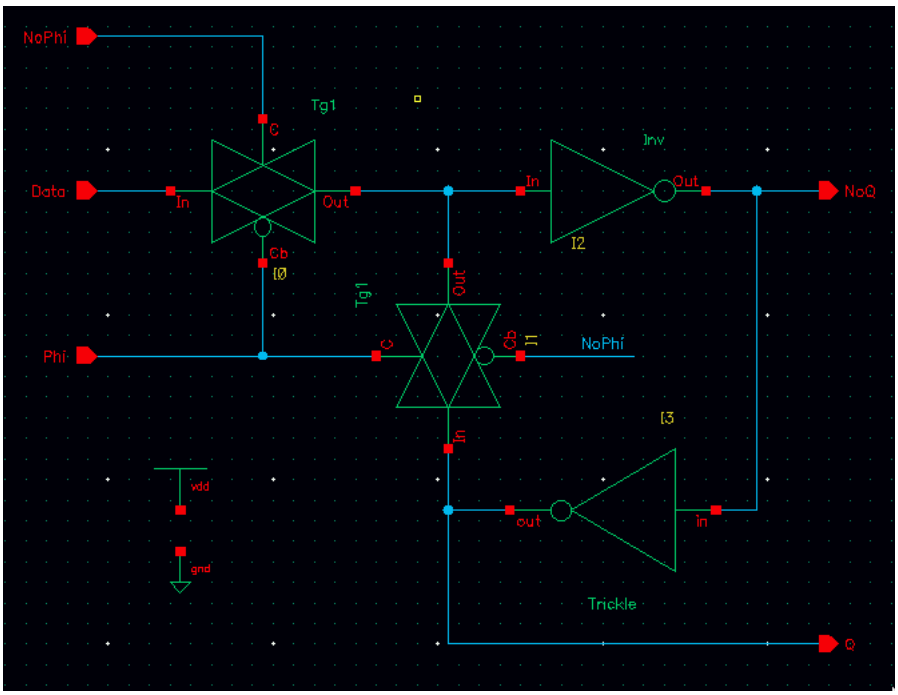

Figura 9: Esquema del Inversor SlaveD

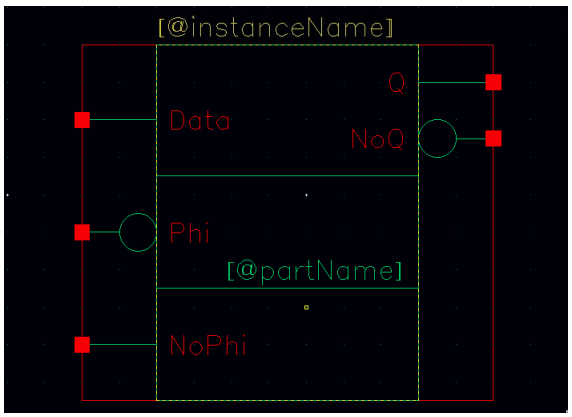

Figura 10: Símbolo SlaveD

Como siempre, vamos a realizar la simulación correspondiente de la célula desarrollada. Para ello preparamos primero el esquema (Figura 11) y, seguidamente su simulación (Figura 12). Desde ella, concluimos que su funcionamiento es el correcto, ya que funciona con el flanco positivo del reloj y

su salida está invertida. Cabe resaltar que cuando la instanciemos en el módulo FFDMS, su entrada estará invertida, por lo que funcionará en el flanco negativo en vez del positivo.

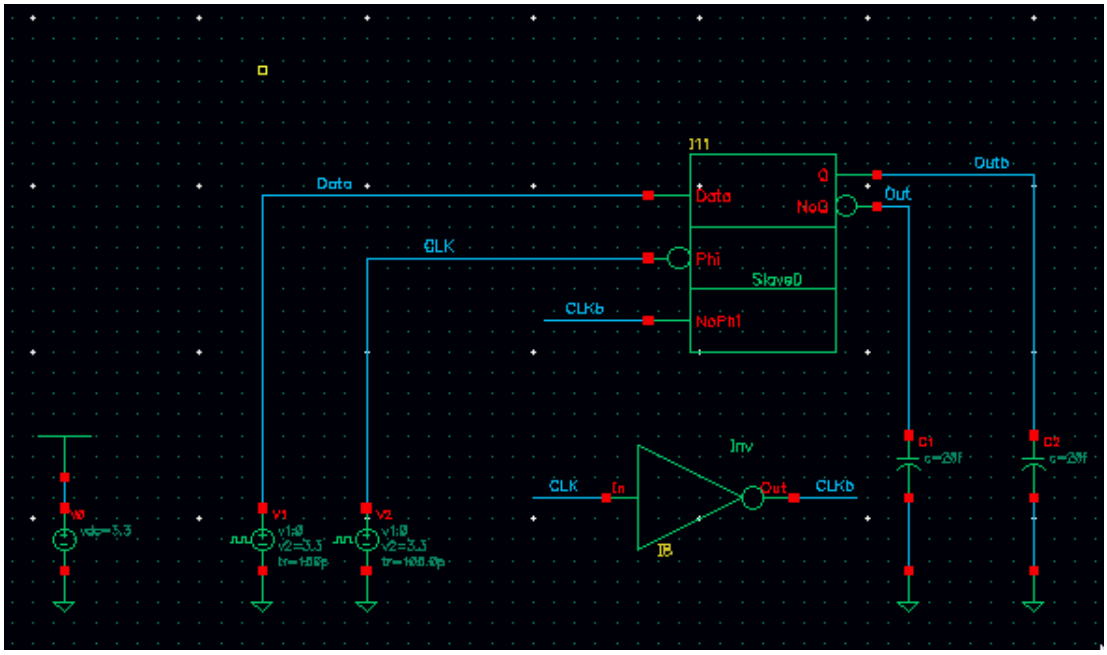

Figura 11: *TB_SlaveD*

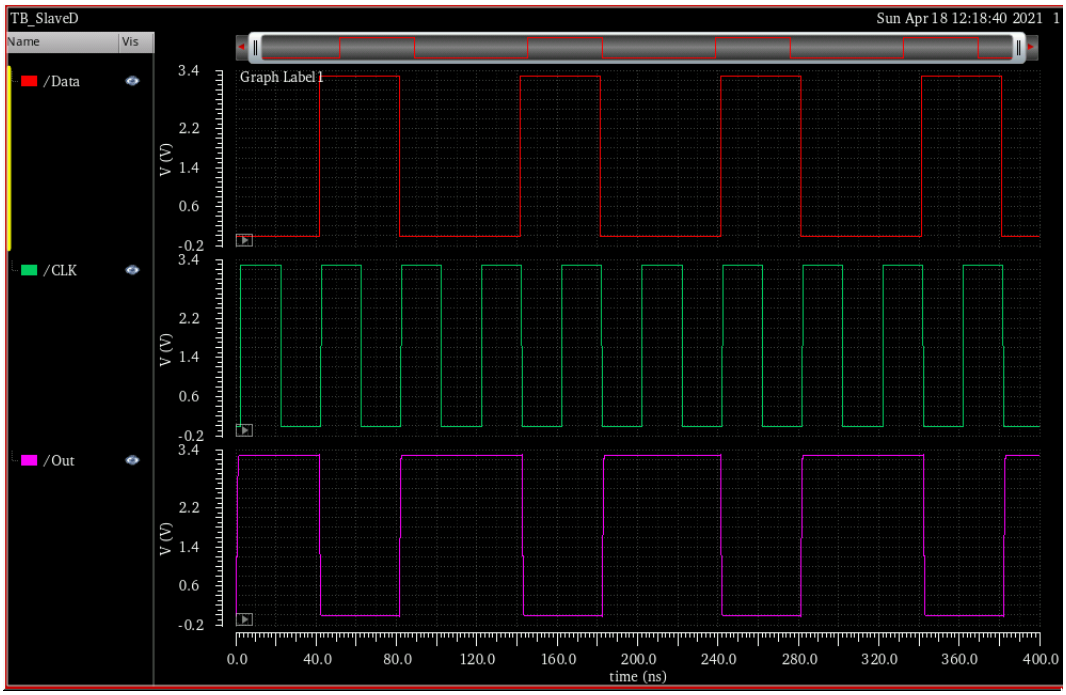

Figura 12: Simulación SlaveD

4. FFDMS

Ahora nos queda el último nivel, que corresponde a unir las células creadas anteriormente para así diseñar el Flip-Flop D Master-Slave. En la Figura 13 vemos el esquema del diseño completo y en la Figura 14 su símbolo.

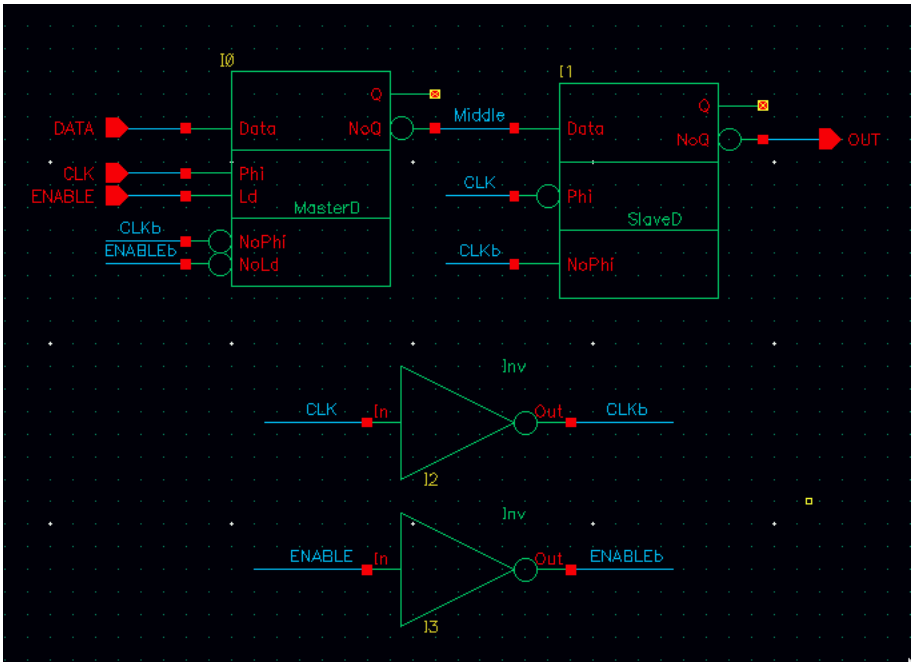

Figura 13: Esquema del Inversor FFDMS

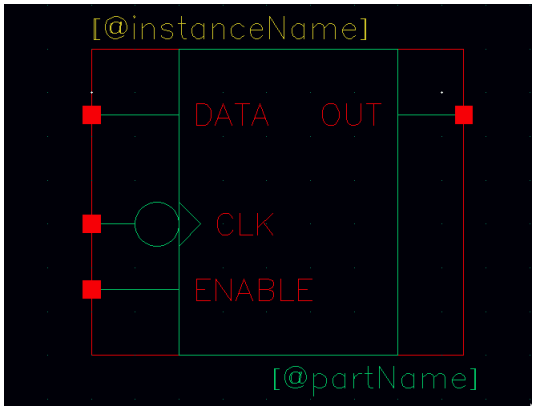

Figura 14: Símbolo FFDMS

Finalmente, solo nos queda la simulación del TestBench, cuyo esquema lo podemos observar en la Figura 15. Tras su simulación obtenemos la salida correspondiente a la Figura 17, cuyo comportamiento es un Flip-Flop sensible a flanco de bajada tras su inicialización síncrona desde Data validada por Load ya que antes, la salida (Out) al no estar inicializado podría haberse cargado a Vdd erróneamente. Otra

simulación que podríamos realizar es la simulación paramétrica variando el parámetro frecuencia de 25 MHz a 400 MHz en pasos de 75 MHz. En nuestro caso, observamos que el Flip-Flop funciona en todas las frecuencias analizadas.

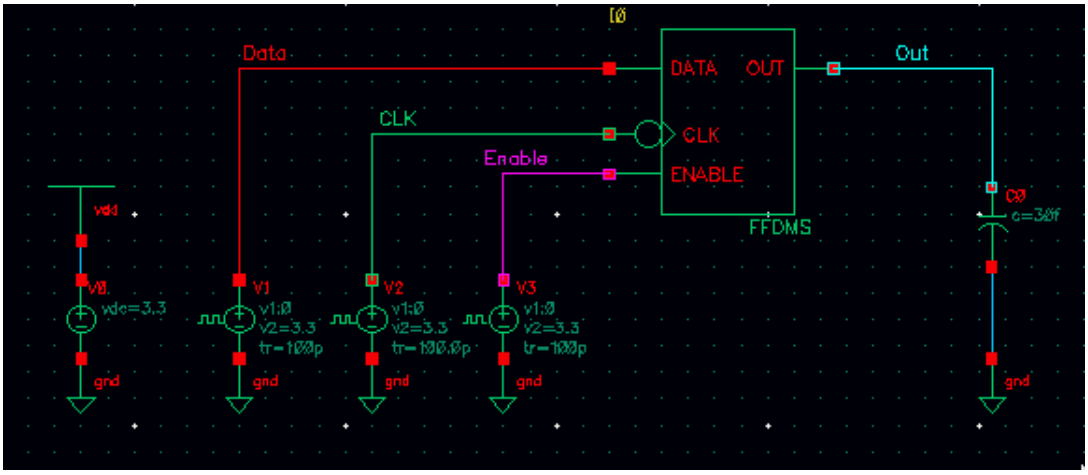

Figura 15: TB_FFDMS

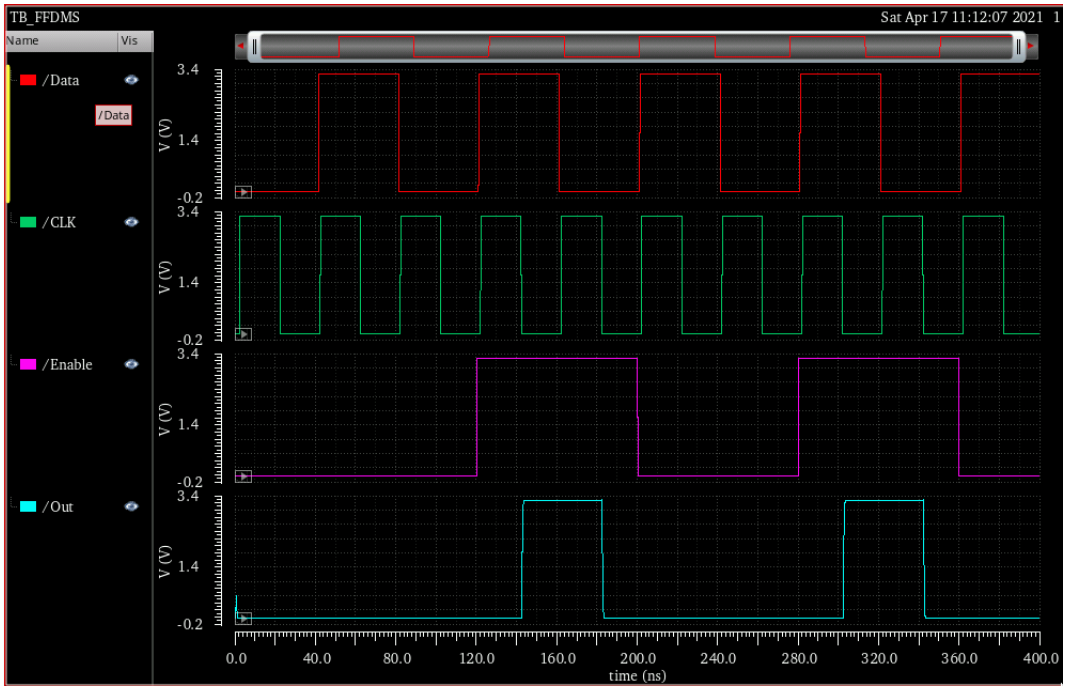

Figura 16: Simulación FFDMS

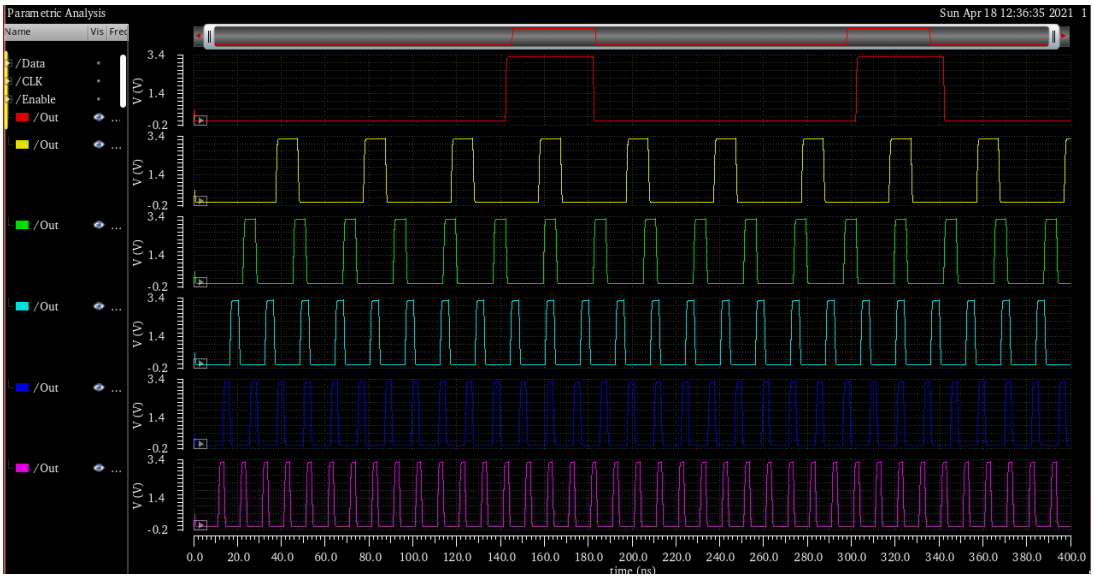

Figura 17: Simulación FFDMS paramétrica

5. ADJUNTOS

En esta sección se pueden observar los distintos valores que se le han dado a cada entrada en la simulación del TB_FFDMS.

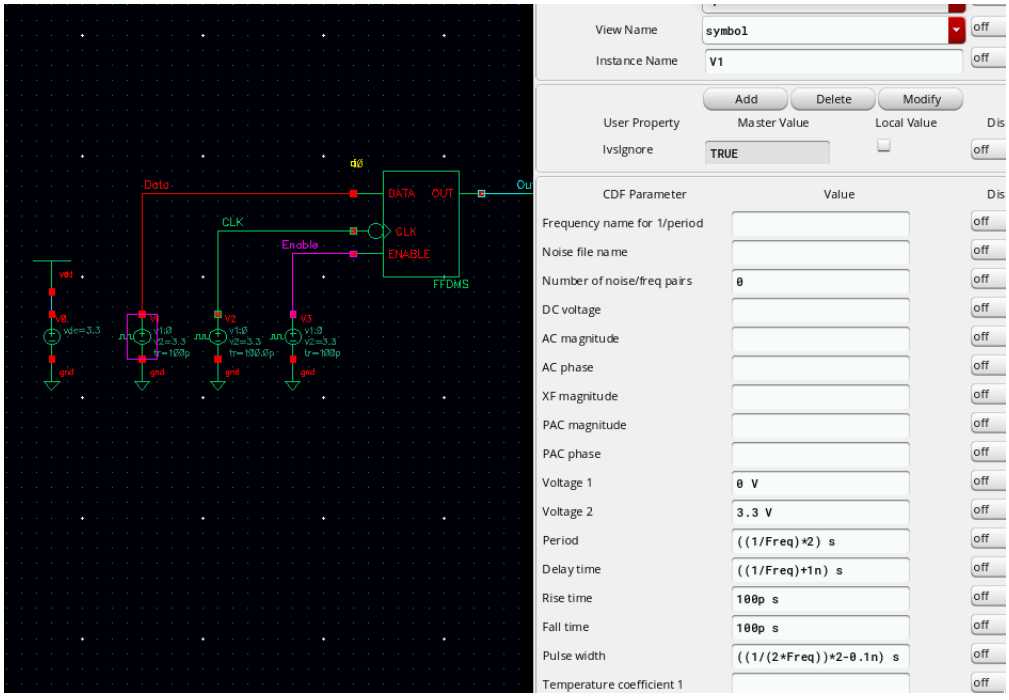

Figura 18: V1 (Data)

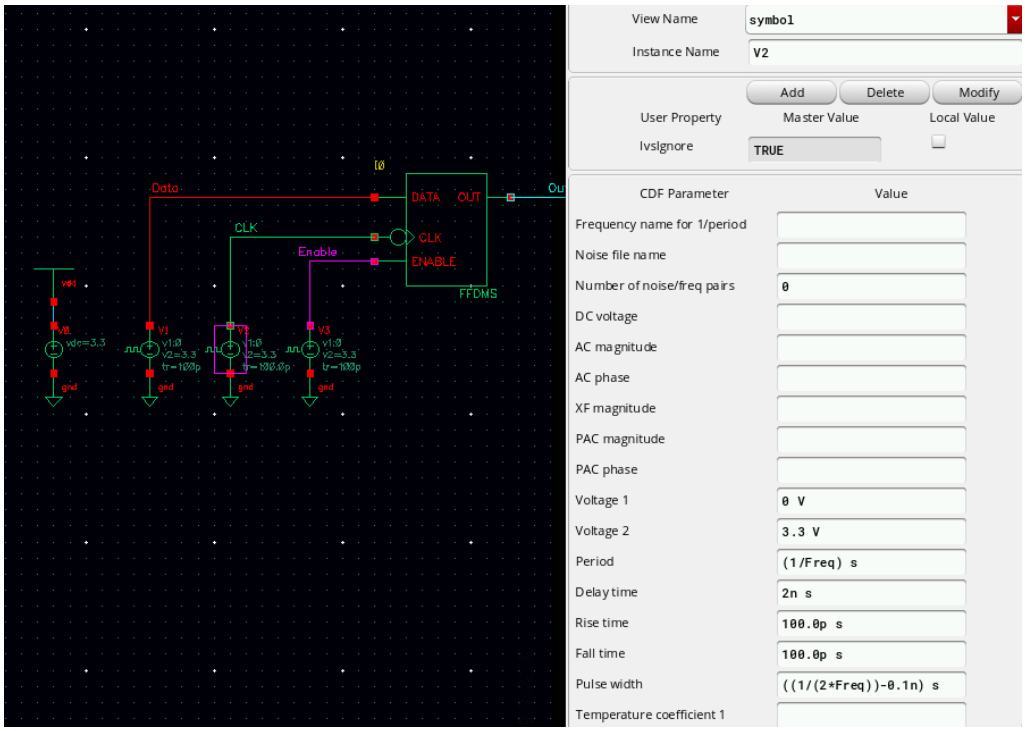

Figura 19: V2 (CLK)

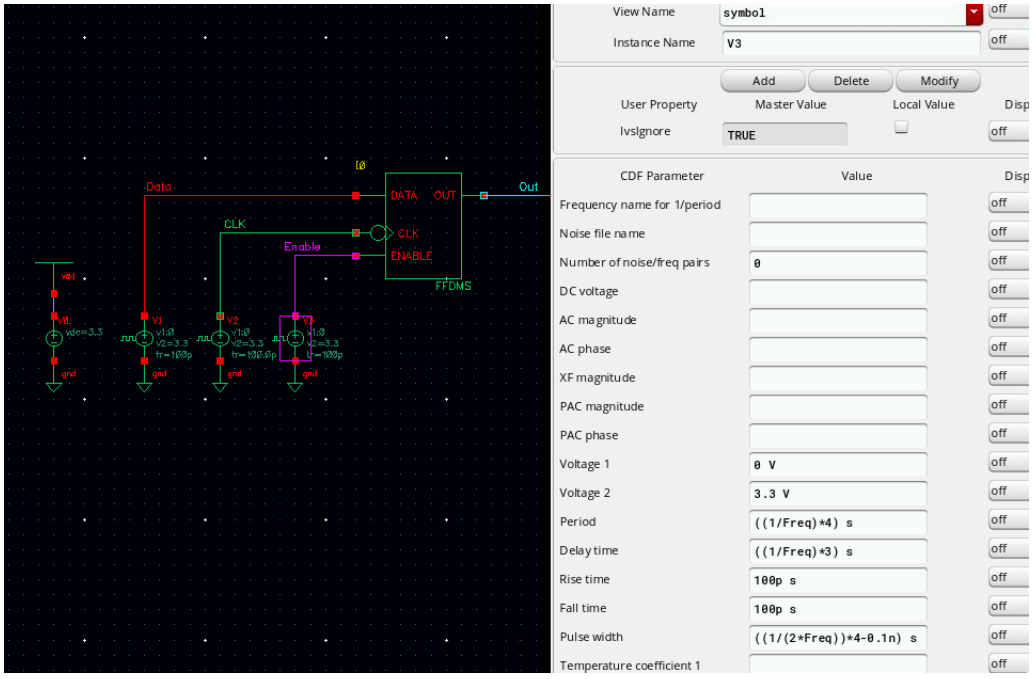

Figura 20: V3 (Enable)

PRÁCTICA 4: DISEÑO FÍSICO Y VERIFICACIÓN DE UN FLIP-FLOP MASTER-SLAVE TIPO D

Jara García Barrena
jagarba3@teleco.upv.es
Luis Valls Sansaloni
luivalsa@teleco.upv.es

22 de mayo de 2021

1. INTRODUCCIÓN

En esta práctica hemos realizado el diseño de un **Diseño Físico y Verificación de un Flip-Flop Master-Slave Tipo D**. Para ello, tratamos de realizar el Diseño Físico de un Flip-Flop Master-Slave tipo D pseudoestático (precisa los relojes ϕ y su negado $/\phi$ para el refresco de datos), en tecnología AMS C35B4, CMOS escalable de pozo n, de 2 polisilicios, 4 metales y 0.35 μ de mínima longitud dibujada de puerta.

En la práctica 1 de la asignatura, se llevaron a cabo el Diseño Eléctrico de las Células y sus Test Benches. Las tareas a realizar son dos:

1. El layout de las células MasterD y SlaveD cumpliendo las reglas de diseño.
2. La verficación funcional de las dos células tras la extracción (av_extrated) y la simulación Spectre. También se podrá llevar a cabo el LVS cuando se considere oportuno.

2. LAYOUT DE LA CÉLULA MASTER

En primer lugar, hemos estudiado las Reglas de Diseño AMS C35B4, CMOS de 0.35μ (documento AMS), con el fin de poder llevar a cabo lo mejor posible el diseño del layout de esta primera célula, el MasterD. Esto nos ha ayudado a comprender mejor las distintas capas, estructuras, elementos, requerimientos, etc. que nos hemos ido encontrando a lo largo de dicho diseño.

Después de este primer contacto que nos sirvió para situarnos en el tema de los diseños físicos, llevamos a cabo el layout en Cadence.
En primer lugar, trabajamos sobre el esquemático MasterD (realizado en la Práctica 1), llamado "MasterD". A partir del esquemático, abrimos el editor de Layout con *Launch/Layout XL*.
Una vez abierto, necesitamos extraer la información del esquemático. Para ello, vamos al menú *Connectivity/Generate/All from Source*. Como nuestra intención es hacer el layout desde cero, debemos desmarcar en la nueva pantalla que nos aparece la opción "Instances" (en el caso de que se dejase habilitada, cada uno de los elementos que tiene su célula layout asociada, se incluirían en el propio diseño). A su vez, en dicho menú, debemos marcar la opción "I/O pins", opción que crea los pines de entrada/salida que definen los terminales, dan información sobre cómo rutar las señales y cómo extraer los nombres de los nudos.

Las capas de fabricación están definidas como propósito pin ("capas de fabricación con nombre"). En la parte "Pin Label As" podemos dar nombre a dichas capas, seleccionando:

- Layer name: PIN
- Layer Purpose: Metal 1 (dicha configuración nos servirá para todas las capas pues no se permite la edición de etiquetas por separado)

Además, las alimentaciones ("gnd" y "vdd") se deberán definir con *MET2 pin*.
Para poder visualizar las etiquetas, seguiremos el menú *Options/Display/Pin Names* y lo configuraremos de manera oportuna.

Una vez llevada a cabo la configuración inicial y localizados los pines de entrada y salida, podemos realizar el diseño del layout. Utilizando la paleta de dibujo (donde están disponibles todas las capas de trabajo, filtrados...) y siguiendo las Reglas de Diseño ya estudiadas (podemos hacer uso de la opción de regla de medida para respetar las distancias entre elementos), llegamos a un layout tal como el que se muestra en la Fig. 1.

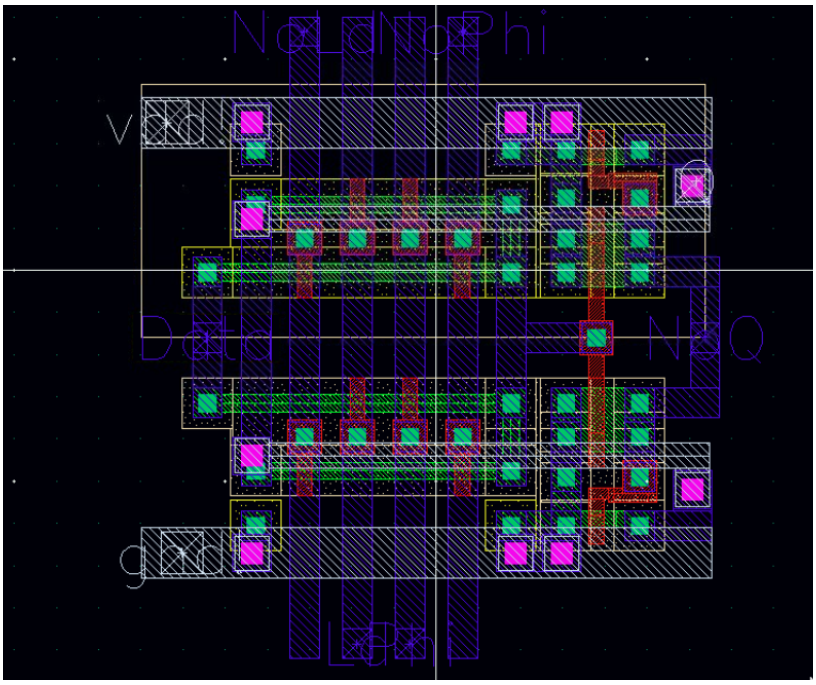

Figura 1: Diseño final del layout MasterD

Para llevar a cabo la verificación del cumplimiento de las Reglas de Diseño que tiene que cumplir nuestro layout, utilizamos el menú *Assura/Run DRC* (ver Fig. 2). Algunos posibles fallos fueron encontrados en ciertas medidas que no cumplían las especificaciones o algún empalme de capas que no llegaba a estar bien unido.
Cabe destacar que los pines anteriormente formados, deben conectarse al material correspondiente (MET1 o MET2, en este caso) junto con sus etiquetas (las cuales deben estar sobre el contacto y el pin. Es un error del que nos dimos cuenta al verificar las Reglas de Diseño).

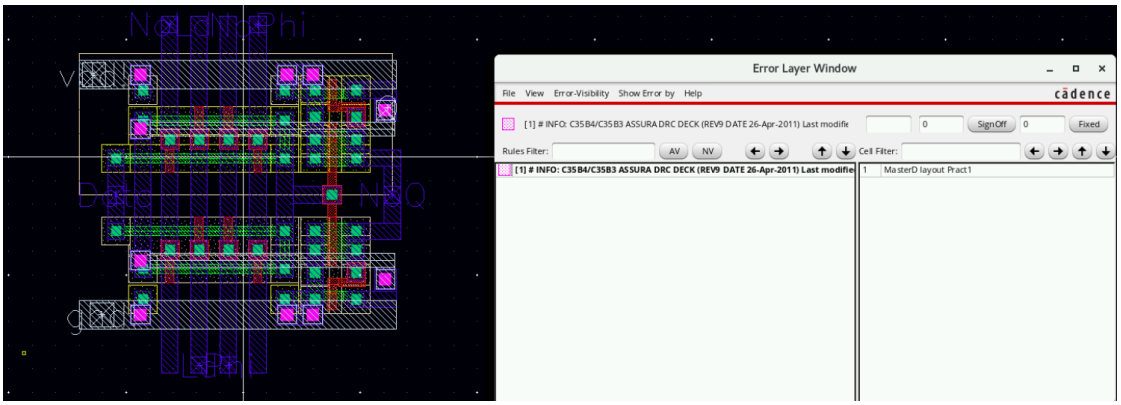

Figura 2: DRC correcto

Podemos comprobar que el layout es correcto y esta libre de errores gracias al único mensaje que obtenemos en la ventana del DRC.

Por último, si llevamos a cabo el LVS, también podemos observar resultados satisfactorios, pues el tipo y número de células que se muestran en la pantalla coincide con el de nuestro diseño (ver Fig. 3).

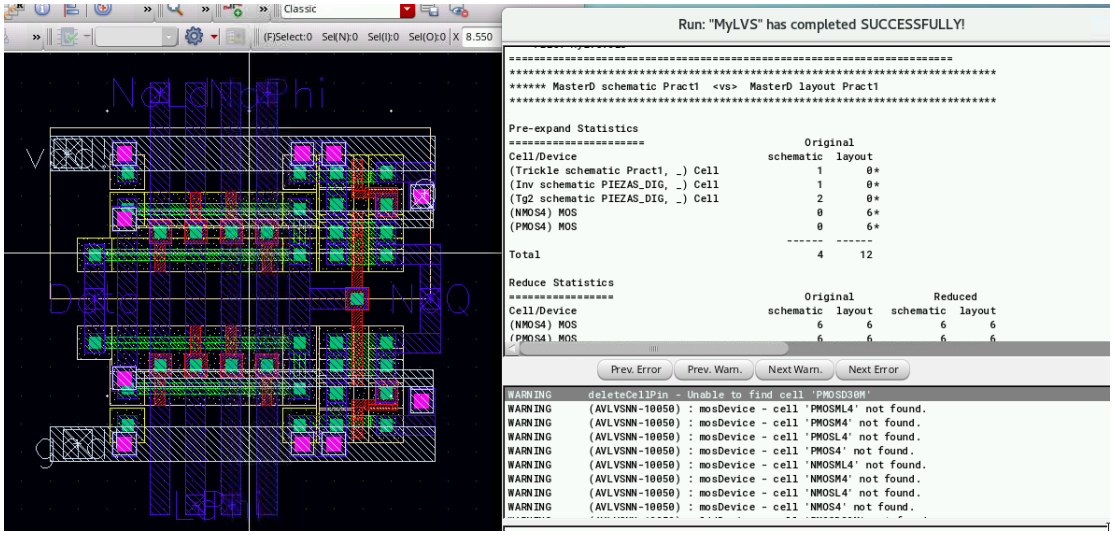

Figura 3: LVS correcto

3. LAYOUT DE LA CÉLULA SLAVE

Una vez diseñado el MasterD, el layout del SlaveD se vuelve más sencillo. Nos damos cuenta que el SlaveD es muy parecido al MasterD. Copiando el diseño, podemos usar la opción *Edit/Basic/Chop* para cortar las secciones que nos sobran. Eliminando y juntando las partes resultantes, obtenemos el siguiente layout:

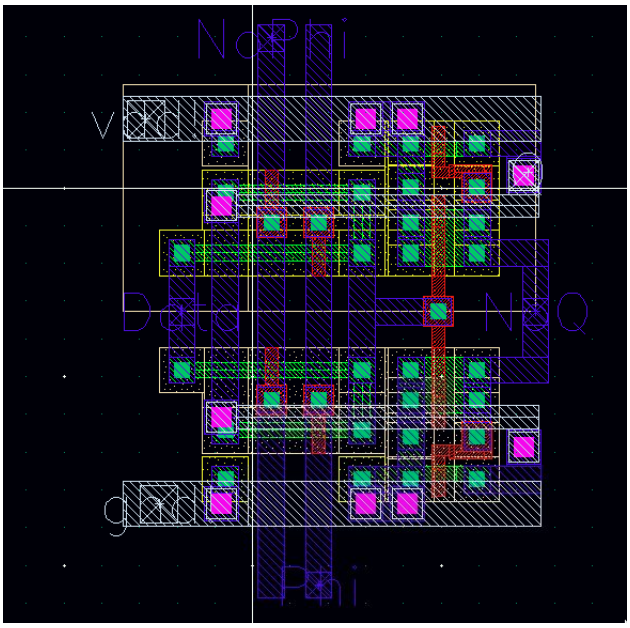

Figura 4: Diseño final del layout SlaveD

Por supuesto, también debemos llevar a cabo la comprobación de que se han seguido correctamente las Reglas de Diseño (*Assura/Run DRC*), ver Fig. 5.

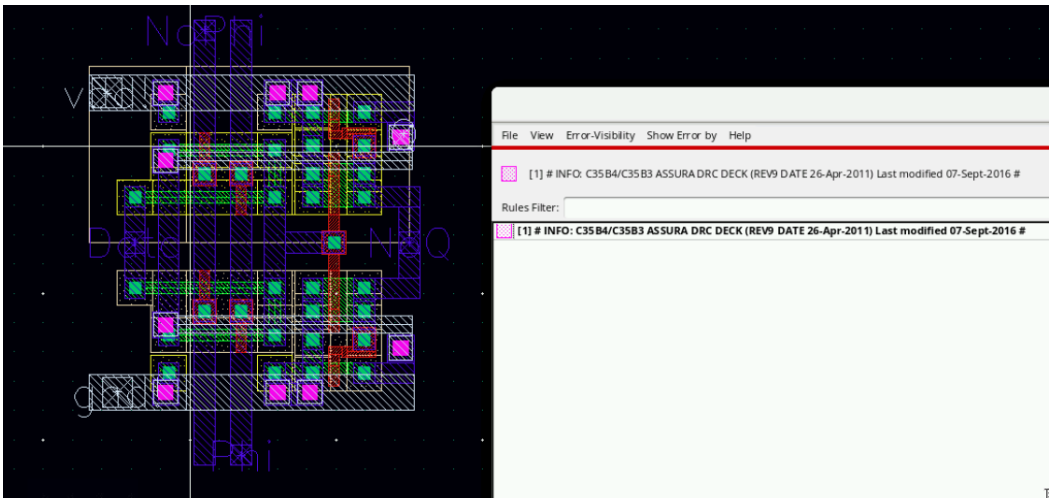

Figura 5: DRC correcto

Podemos comprobar que el layout es correcto y esta libre de errores gracias al único mensaje que obtenemos en la ventana del DRC.

Por último, si llevamos a cabo el LVS, también podemos observar resultados satisfactorios, pues el tipo y número de células que se muestran en la pantalla coincide con el de nuestro diseño (ver Fig. 6).

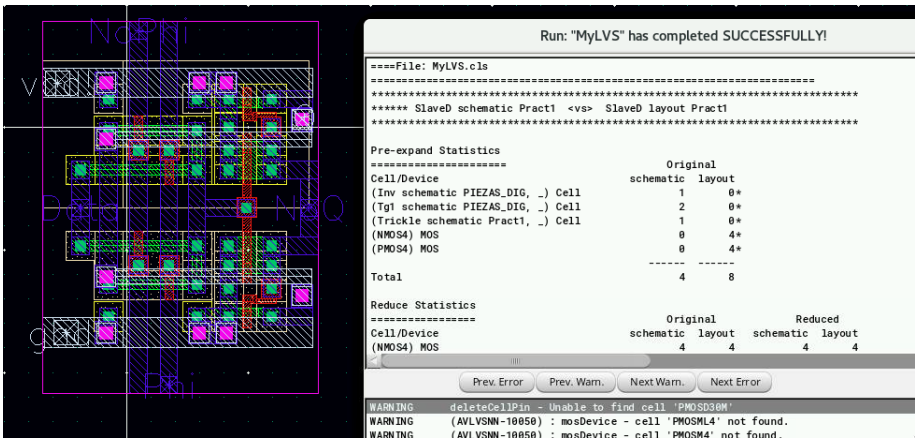

Figura 6: LVS correcto

4. SIMULACIÓN DEL MASTER

A continuación, es importante llevar a cabo un proceso de verificación de las dos células que hemos creado. En este caso, para el MasterD, realizaremos un Test Bench modificado (ver Fig.7) para poder incluir la extracción (para ello, editar su vista "config").

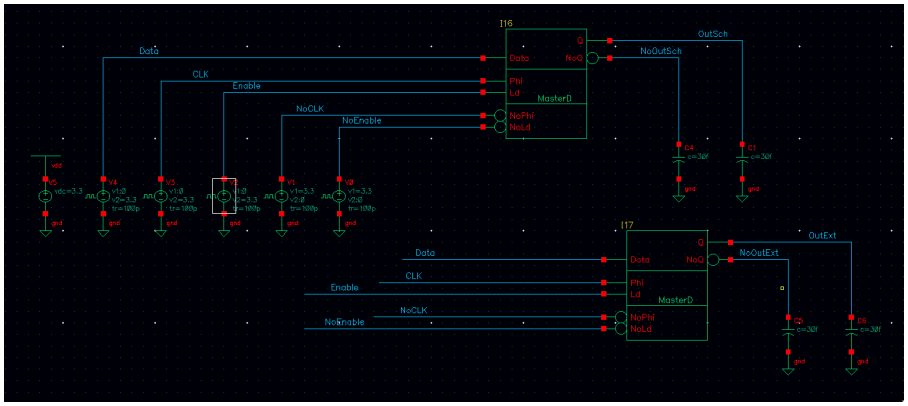

Figura 7: Diseño del Test Bench para MasterD

Llevando a cabo la simulación con el menú ADE, podemos observar la gráfica expuesta en la Fig. 8

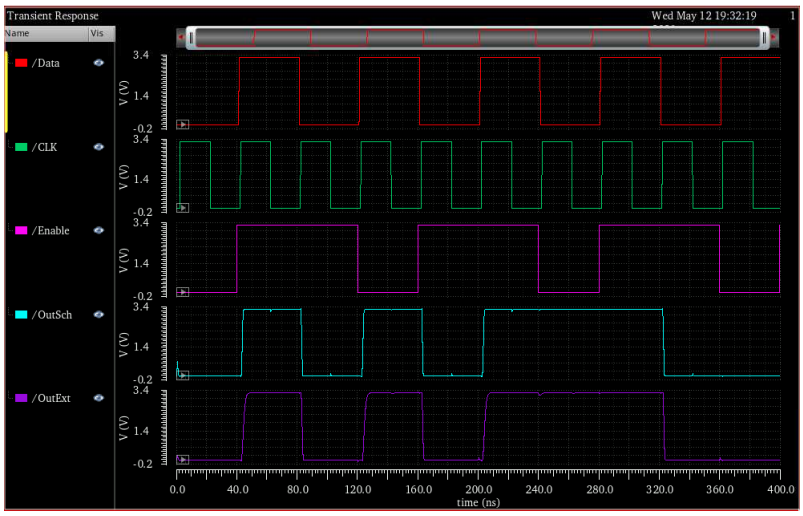

Figura 8: Gráfica obtenida con la simulación del MasterD

Podemos observar que la gráfica del esquemático (azul) y la del av_extracted (morada) son parecidas pero no iguales. Su diferencia principal reside en su forma de onda; mientras que la salida del esquemática es totalmente recta, la salida del ac_extracted presenta una forma más redondeada.

5. SIMULACIÓN DEL SLAVE

Utilizando el mismo procedimiento seguido en la simulación del MasterD, creamos un Test Bench modificado (ver Fig.9) para poder incluir la extracción del mismo.

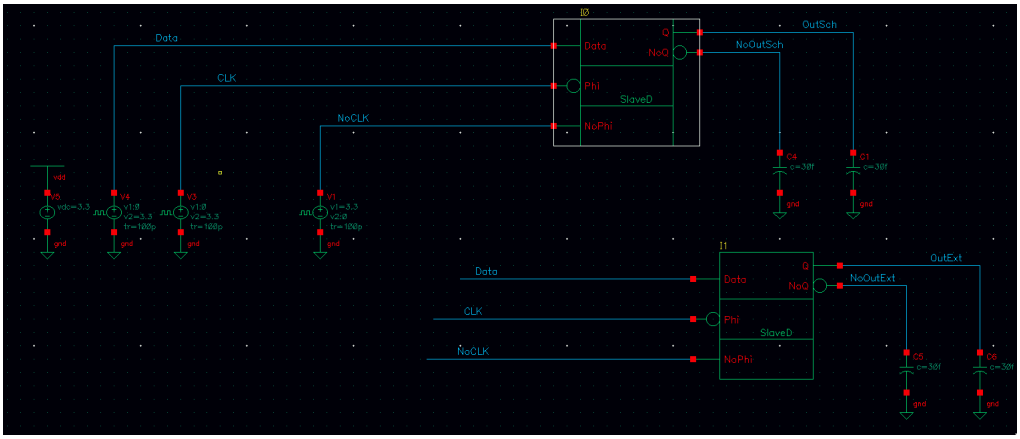

Figura 9: Diseño del Test Bench para SlaveD

Simulando el esquemático anteriormente expuesto, obtenemos las siguientes gráficas:

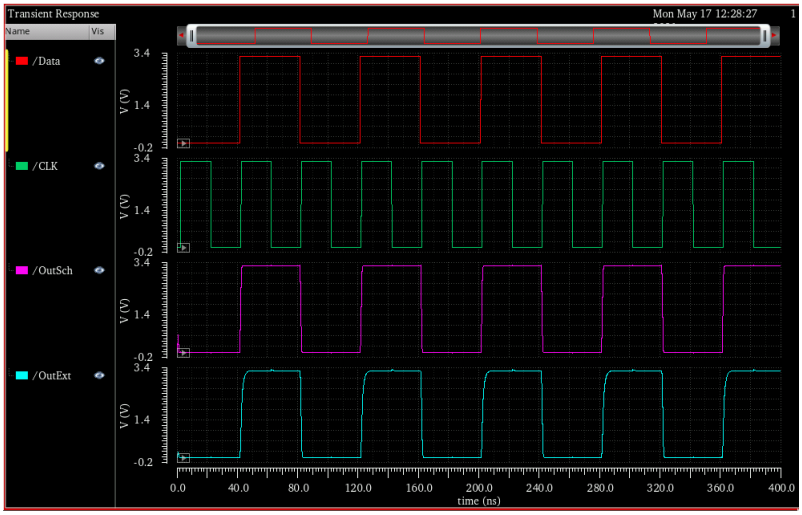

Figura 10: Gráfica obtenida con la simulación del SlaveD

Siguiendo la misma filosofía que en el anterior diseño, vemos que las formas de onda de las dos salidas se parecen pero no llegan a ser iguales. Justamente, la diferencia principal recae en su forma; la onda de la salida del esquemático (esta vez de color rosa) se mantiene más recta en sus cambios de nivel, mientras que la salida del av_extracted presenta una forma más redondeada.

6. CREACIÓN DEL FFDMSC

Una vez verificadas las células MasterD y SlaveD, procedemos a crear el diseño del FFDMS. Dicho diseño consiste en la unión del MasterD y el SlaveD. Juntando ambos símbolos, podemos formar su esquemático para una simulación que verifique el funcionamiento del mismo (ver Fig. 11).

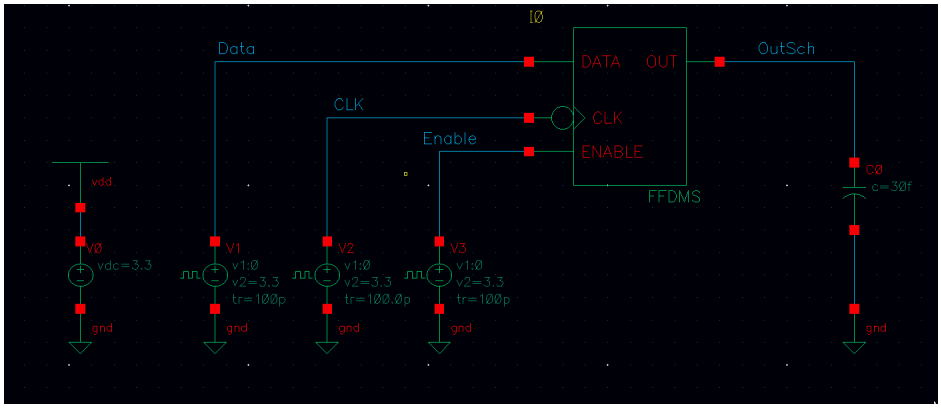

Figura 11: Diseño del Test Bench para FFDMS

La gráfica que obtenemos de esta simulación es la siguiente:

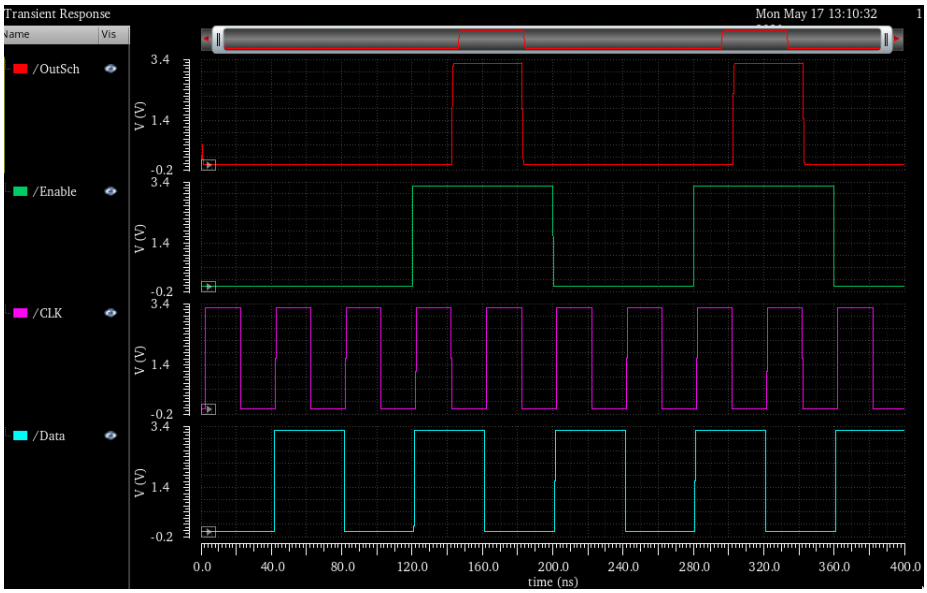

Figura 12: Gráfica obtenida con la simulación del FFDMS

Pensando en un registro paralelo de FFDMS de n-bits independiente, podemos mejorar nuestro diseño. Para ello, introduciremos una nueva célula, el PhaseSplitter.

6.1. PhaseSplitter

El PhaseSplitter es una célula que utilizaremos con frecuencia. Debemos crear el esquemático, símbolo y layout de esta célula para poder usarlo en el nuevo FFDMS.

6.1.1. Esquemático y simulación

El esquemático del PhaseSplitter puede observarse en la Fig. 17.

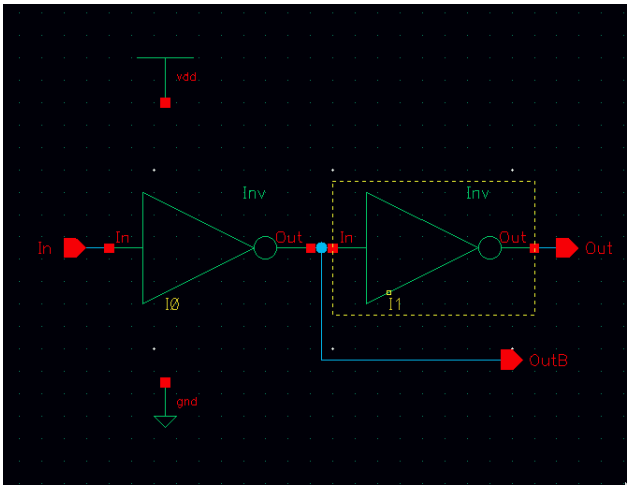

Figura 13: Esquemático del PhaseSplitter

6.1.2. Layout

Siguiendo las Reglas de Diseño, procedemos a dibujar el layout de la anterior célula (ver Fig. 14).

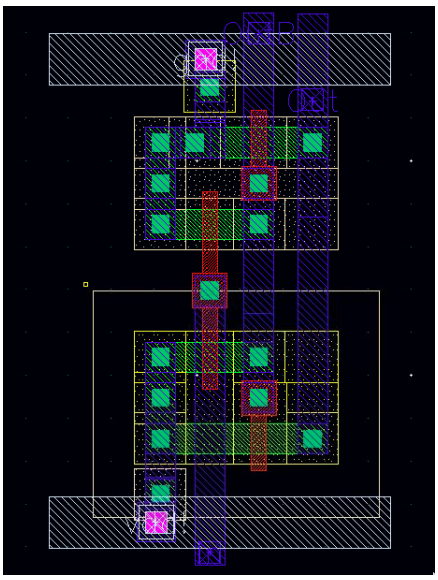

Figura 14: Layout del PhaseSplitter

Para dar por finalizado el layout del PhaseSplitter, llevamos a cabo el DRC y el LVS de la célula, los cuales podemos ver que son correctos (ver Fig. 15 y 16)

Figura 15: DRC correcto

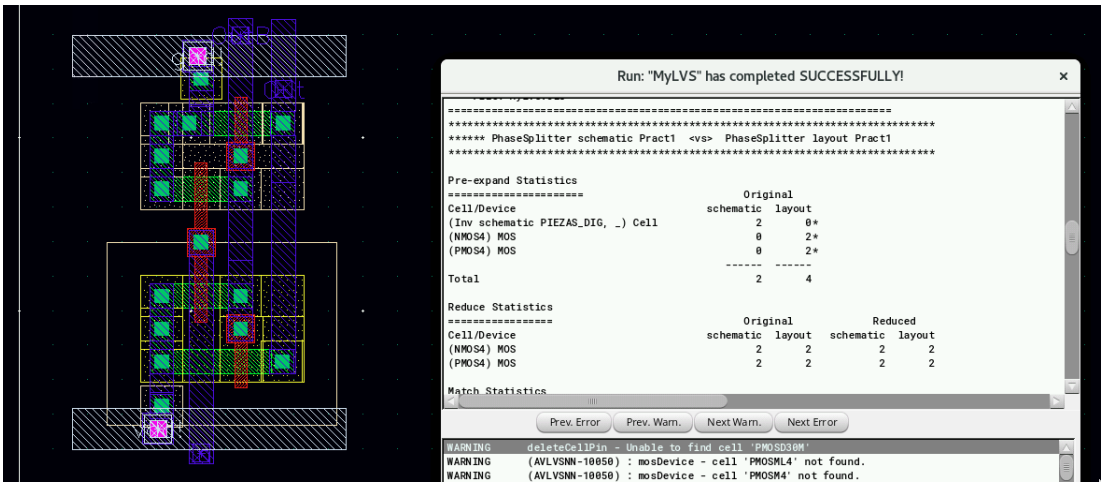

Figura 16: LVS correcto

6.1.3. Simulación comparada

Tras crear el símbolo (se puede observar en el Test Bench) y el layout, diseñamos el esquemático del Test Bench para poder llevar a cabo la simulación (ver Fig. 19).

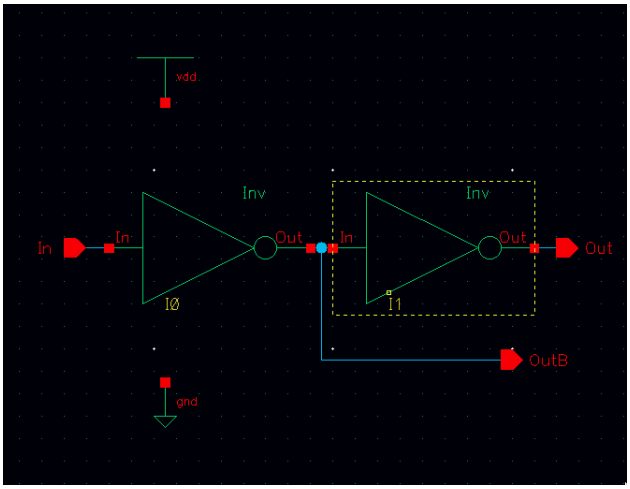

Figura 17: Diseño del esquemático de PhaseSplitter

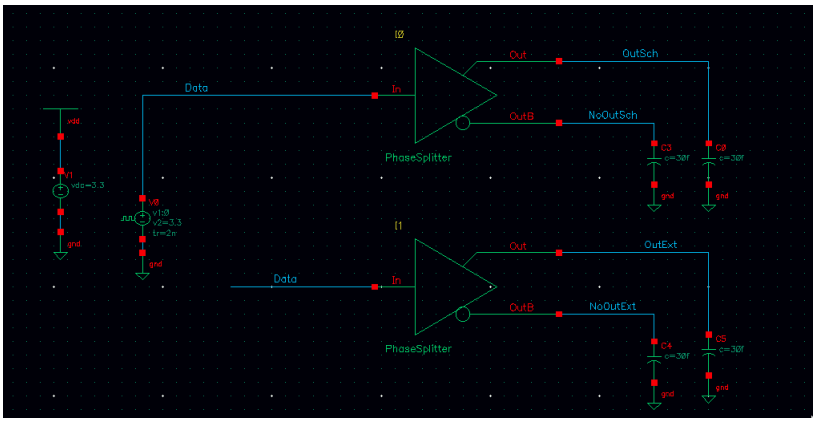

Figura 18: Diseño del Test Bench para el PhaseSplitter

La gráfica resultante de la simulación del Test Bench anteriormente expuesto es la siguiente:

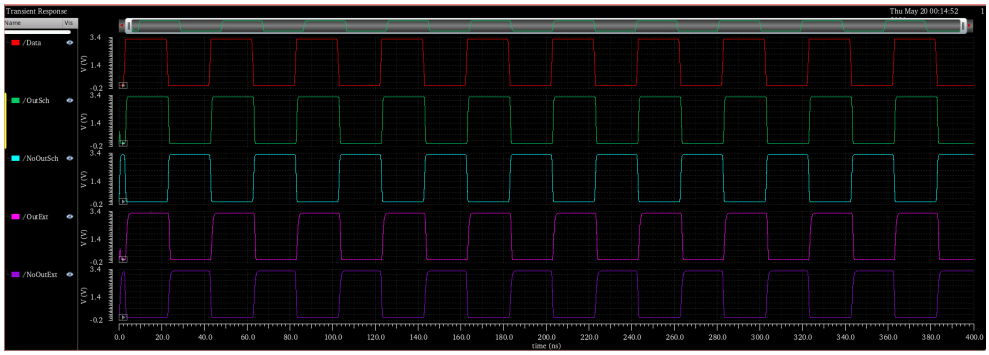

Figura 19: Diseño del Test Bench para el PhaseSplitter

Se puede verificar que el resultado es el correcto, pues la salida "out" tiene la misma forma de onda que la entrada. Por otro lado, la señal "NoOut"tiene la misma forma de la señal pero negada.
Además, si comparamos el esquemático con el av_extracted, observamos la misma diferencia que en las anteriores simulaciones comparadas; forma similar pero con diferencias (una recta y otra redondeada).

6.2. FFDMSC

Una vez creado el PhaseSplitter, podemos llevar a cabo el diseño completo del FFDMSc.

6.2.1. Esquemático

El esquemático del diseño mejorado del FFDMS, es igual que el anterior, pero añadiendo 3 PhaseSplitter, utilizando el símbolo que hemos creado anteriormente (ver Fig. 20).

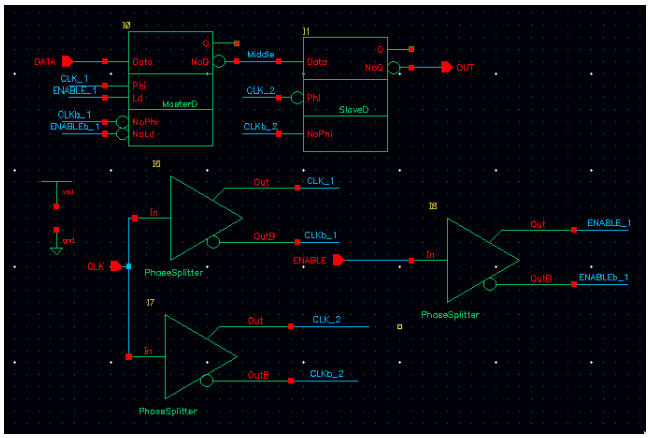

Figura 20: Diseño del esquemático de la célula FFDMSc

6.2.2. Layout

El layout sigue el mismo patrón que los anteriores; utilización de las distintas capas para dibujar el diseño físico, seguimiento y comparación de las Reglas de Diseño. Podemos observar el esquema resultante en la siguiente Figura:

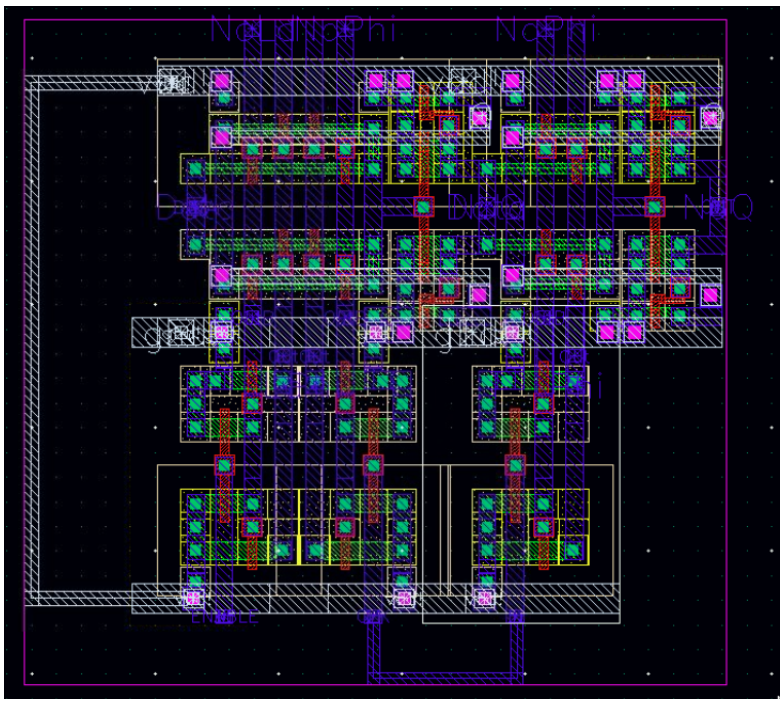

Figura 21: Layout del FFDMSc

Para finalizar, podemos realizar las simulaciones DRC y LVS de la célula con la que estamos trabajando. En las siguientes figuras (ver Fig. 22 y 23) puede verse que los resultados son correctos.

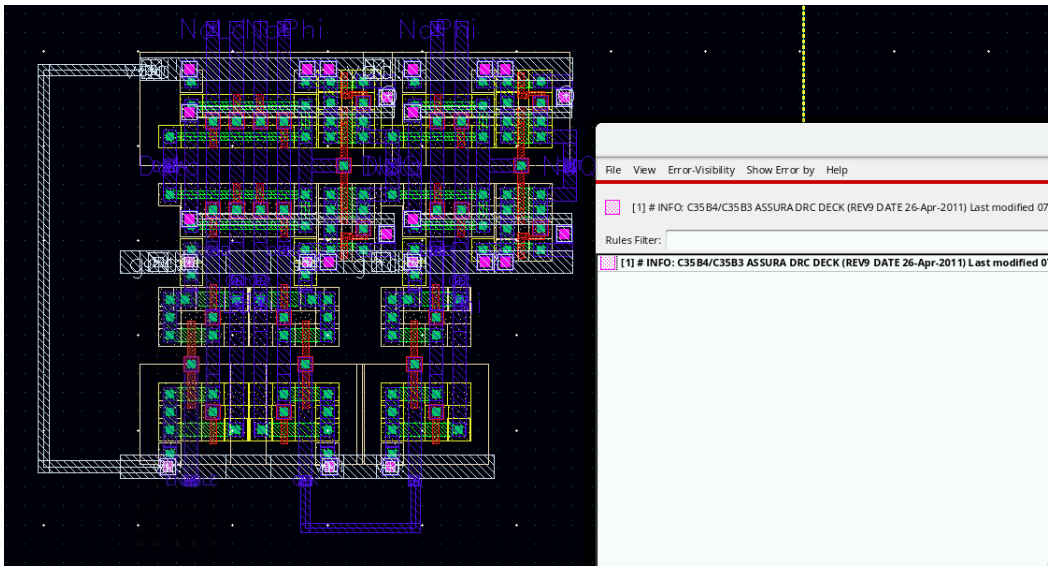

Figura 22: DRC correcto

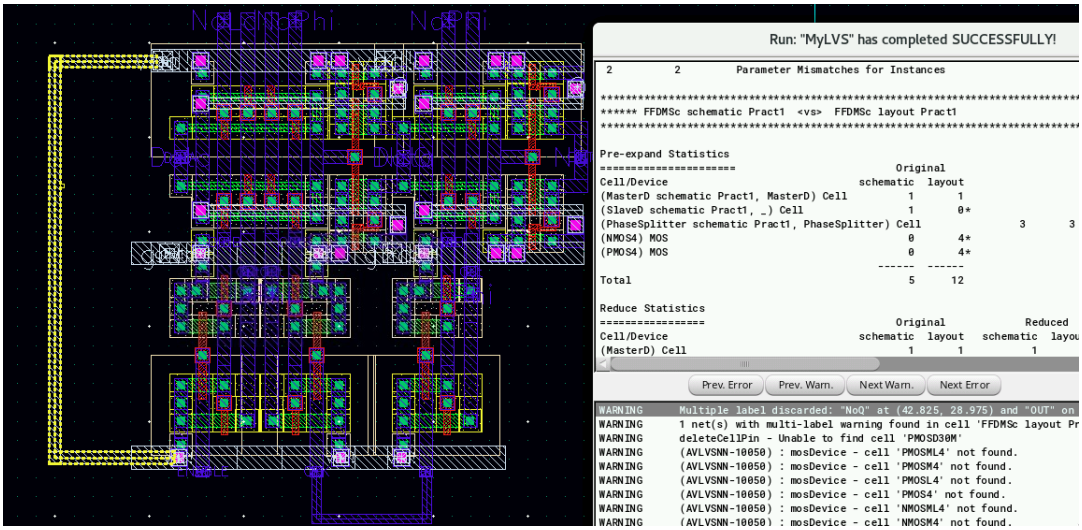

Figura 23: LVS correcto

7. COMPARACIÓN DEL ESQUEMÁTICO FFDMSC Y AV_EXTRACTED

Finalmente, simulamos el esquema FFDMSc total, de forma comprada. Para ello, editaremos la opción "Descend" (ver Fig. 24 y 25).

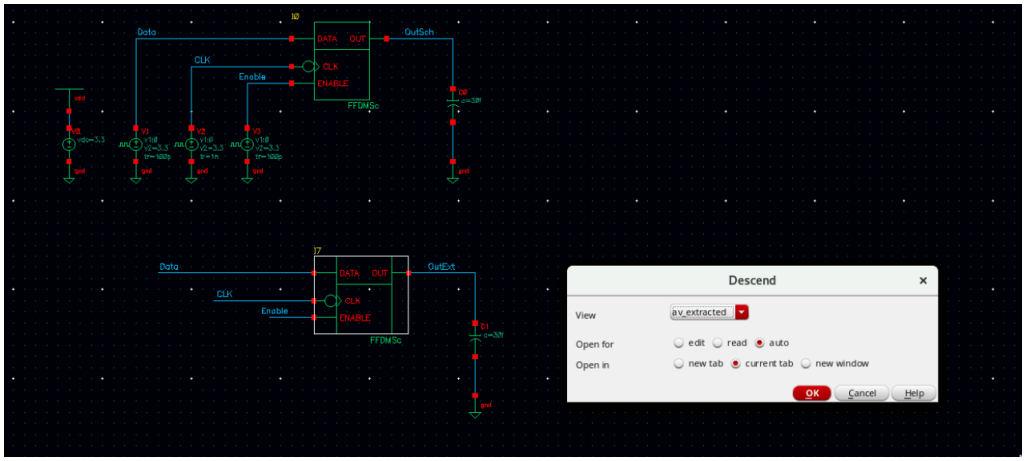

Figura 24: Configuración del TestBench de FFDMSc

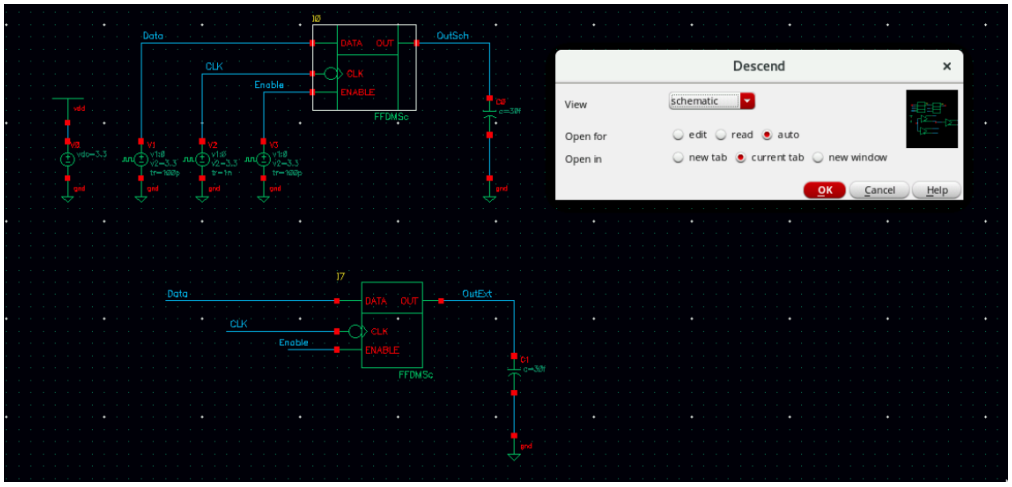

Figura 25: Configuración del TestBench de FFDMSc

Una vez configurados, procedemos a la simulación del Test Bench propuesto (ver Fig. 26). Podemos observar que detrás de la forma de onda en morado (la correspondiente a /OutExt) se encuentra la forma de onda del Schematic. Con lo que concluimos que no son iguales.

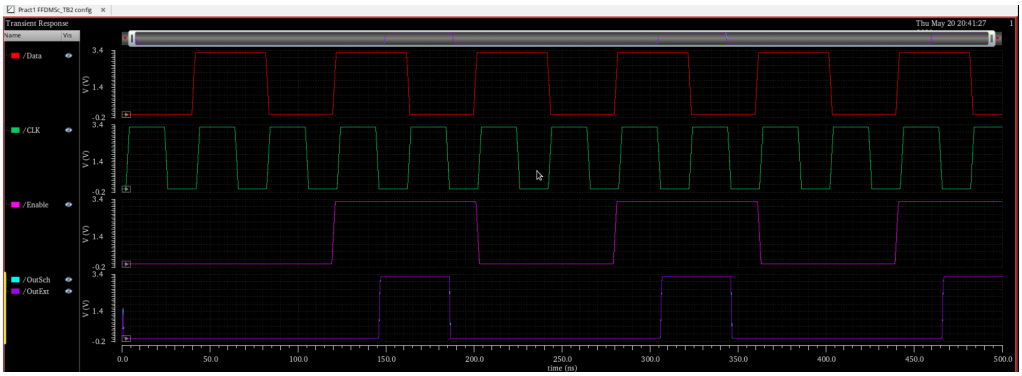

Figura 26: Simulación del FFDMSc

8. CONCLUSIÓN

La realización de las dos prácticas expuestas en esta memoria nos ha servido para reforzar los conocimientos vistos en clase de teoría y tener una primera toma de contacto con el Software Cadence.

En la Práctica 1 se ha podido completar todos los objetivos correctamente. Hemos creado los esquemáticos del MasterD, SlaveD y el FFMSD, así como el del inversor triple. A su vez, se han llevado a cabo la simulación de cada célula con su propio Test Bench. Por último, se ha realizado un análisis paramétrico sobre el FFDMS.

En la Práctica 4 se ha creado el layout del MasterD y el del SlaveD. Tras ello, hemos llevad a cabo la célula PhaseSplitter con su correspondiente layout. Seguidamente, hemos generado el esquemático del nuevo FFMSDc usando 3 PhaseSplitters. Cabe destacar que hemos comparado tanto la simulación del esquemático con la simulación del av_extracted. Por falta de tiempo no hemos podido plantear el diseño de un registro paralelo de 4 bits. Para una futura mejora, nos centraríamos en generar dicho registros así como crear diferentes layouts del PhaseSplitter y del FFDMS para poder comprobar qué comportamiento es el más adecuado.