



Universidad Nacional del Sur

Informe Final:
Desarrollo de un Módulo I2C Esclavo.

Hipperdinger Federico
Julio 2024

Índice

1. Introducción	3
2. Funcionamiento	4
3. Desarrollo	7
3.1. Top Module	7
3.1.1. Puertos	7
3.1.2. Descripción	7
3.2. Start-stop detector Module	10
3.2.1. Puertos	10
3.2.2. Descripción	11
3.3. I2C_submodule Module	13
3.3.1. Puertos	13
3.3.2. Descripción	13
3.3.3. Funcionamiento	13
3.3.4. Comentarios adicionales	14
3.4. Counter Module	17
3.4.1. Puertos	17
3.4.2. Descripción	18
3.5. Address Detection Module	21
3.5.1. Puertos	21
3.5.2. Descripción	21
3.6. Write Module	24
3.6.1. Puertos	24
3.6.2. Descripción	24
3.7. R_W Module	28
3.7.1. Puertos	28
3.7.2. Descripción	28
4. Simulaciones	30
4.0.1. Modo lectura	30
4.0.2. Modo escritura	33
4.0.3. Velocidad	36
5. Conclusiones	37
6. Appendices	38
A. Listado de modulos completo.	38

1. Introducción

En este informe se presentan los resultados del proyecto final de la materia Análisis y Diseño de Circuitos Digitales (ADCD), cuyo objetivo fue diseñar e implementar un módulo esclavo I2C. Este módulo gestiona la comunicación en el bus I2C y utiliza interrupciones para notificar al usuario sobre eventos clave, como el inicio y fin de la transmisión, así como la transferencia de datos.

El diseño y las simulaciones se llevaron a cabo en LTSpice, mientras que el layout fue desarrollado en Electric VLSI Design.

En una conexión directa al bus I2C, el procesador debe interrumpir su ejecución de manera reiterada para leer el bus, identificar si se están comunicando con él e implementar todo el intercambio necesario para leer o escribir datos en la línea (Polling). Esto genera una pérdida de tiempo considerable, ya que la velocidad del procesador suele ser mucho mayor que la velocidad de comunicación del protocolo. La solución propuesta sustituye este mecanismo de polling por un esquema de interrupciones: con solo dos interrupciones, el procesador podrá identificar los momentos de carga y descarga, realizar la lectura o escritura de un mayor volumen de datos en un tiempo reducido, y luego liberar recursos para continuar con otras tareas mientras el módulo se encarga de la transmisión o recepción de datos a través del bus I2C.

En la Figura 1 se presenta un diagrama del módulo. Este módulo está basado en el diseño de referencia "Generic Soft I²C Slave/Peripheral" de Lattice Semiconductors. El modelo desglosa el protocolo I2C, utilizando flags para indicar la modalidad y los momentos de inicio y parada. Además, permite al usuario escribir o leer datos en buses de 8 bits. De esta manera, los flags interrumpen al usuario para que pueda tomar o escribir los datos necesarios rápidamente y luego continuar con su ejecución, sin tener que preocuparse por la comunicación con el módulo maestro.

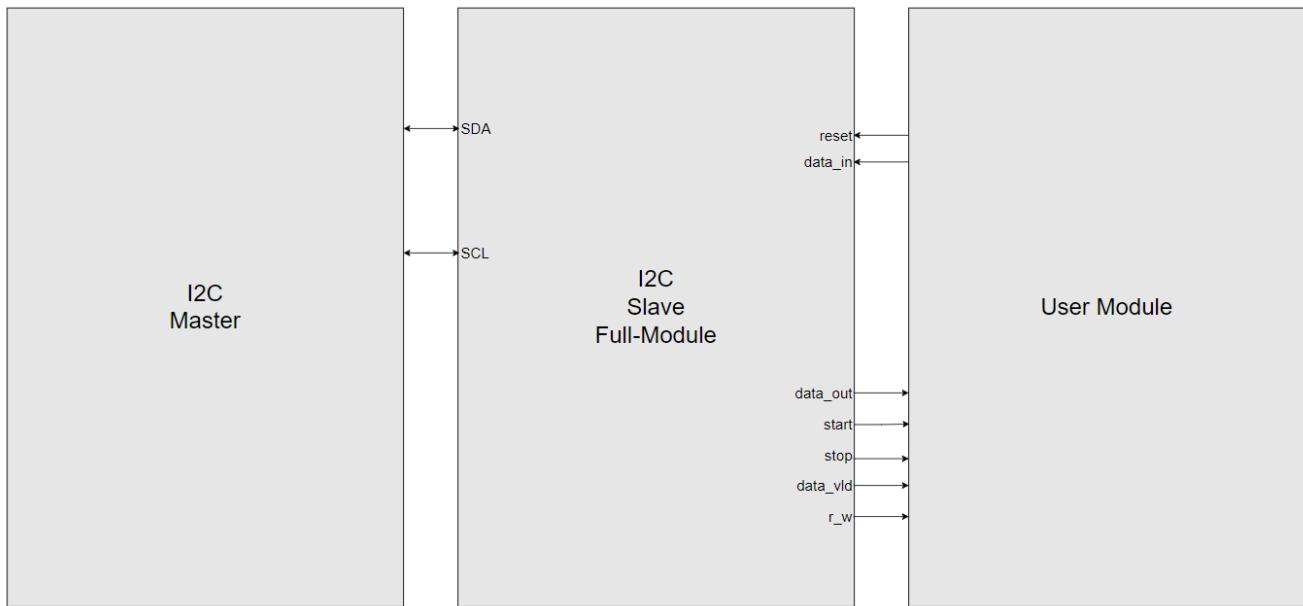


Figura 1: Diagrama del Módulo.

2. Funcionamiento

En el protocolo I2C, los datos se transfieren en mensajes (Figura 2). Cada mensaje contiene la dirección binaria del esclavo, que puede ser de 7 o 10 bits, y uno o más bloques de datos de 8 bits, que contienen los datos que se transmiten. El mensaje también incluye condiciones de inicio y parada, bits de lectura/escritura y bits ACK/NACK entre cada trama de datos.

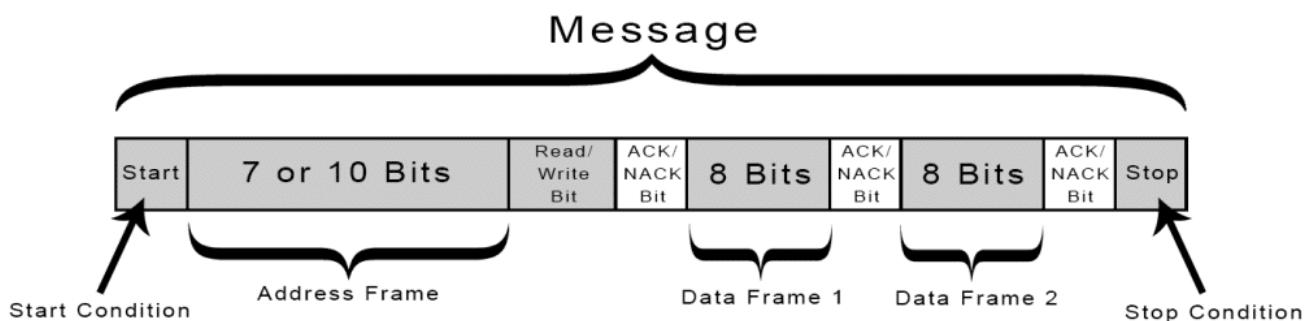


Figura 2: Formato de mensaje I2C.

El bit de Start se da con un cambio de bajo a alto de la señal SDA cuando SCL está en alto. Luego,

SCL se pone en bajo, dando lugar a que la señal SDA cambie al valor del primer bit del Address. De esta manera, cada vez que SCL está en bajo, SDA cambia al siguiente valor y, cuando se encuentra en alto, lo mantiene. Una vez que se transmiten los bits de dirección y el bit de lectura/escritura (RW), el módulo esclavo con esta dirección pone en 0 la línea SDA para indicar ACK. Luego de esto, dependiendo si se solicitó lectura o escritura, el módulo esclavo o maestro escribirá, respectivamente, escribirá sobre la línea SDA y esperará el bit de ACK del otro cada 8 bits. Esto ocurrirá indefinidamente hasta que el módulo maestro envíe el bit de Stop cambiando la línea SDA de alto a bajo cuando SCL esté en alto.

El módulo esclavo se diseñó para detectar los momentos de inicio y parada, verificar si la dirección recibida coincide con la suya y realizar los handshakes necesarios con el módulo maestro mediante los bits de ACK/NACK. Todo lo cual se lleva a cabo mientras lee el bus e intercambia información con el módulo del usuario.

En la Figura 3 se observa una descripción detallada de sus Entradas y salidas. Al llegar el bit de start, el módulo entrega un pulso por la salida "Start" y comienza a leer la línea SDA hasta recibir 8 bits, donde compara los primeros 7 para verificar la dirección. Si corresponde a la dirección introducida en la Entrada "Address", define el modo de funcionamiento a partir del octavo bit del primer mensaje, indicado en la salida "r_w", y comienza a actuar acorde al mismo.

En modo escritura ($r_w = 0$), el módulo debe leer información (en serie) de línea SDA y transmitirla (en paralelo) al usuario. Para ello, cada vez que recibe los 8 bits correspondientes a una palabra, se envía un ACK, se transmiten los 8 bits en la salida "data_out" y se pone en alto la salida "data_vld" durante un ciclo.

En modo lectura ($r_w = 1$), el módulo debe leer información (en paralelo) del usuario y escribirla (en serie) sobre la línea SDA. Para ello, cada vez que requiere escribir los 8 bits de una palabra, se pone en alto la salida "data_vld" durante un ciclo para que el usuario escriba los datos a transmitir en el bus "data_n". Luego, se transmiten estos 8 bits en la línea SDA y se espera a un ACK desde el maestro para volver a solicitar datos al usuario.

Signal Name	I/O Type	Width	Interface	Description
scl	BIDI	1	I2C	Serial Clock Line of the I2C core
sda	BIDI	1	I2C	Serial Data Line of the I2C core
reset	IN	1	Peripheral	Active-HIGH reset that signals the I2C Slave module to initialize.
data_in	IN	8	Peripheral	8-bit input data port used to receive data from the user module that is transmitted to the I2C Master during I2C Read.
Address	IN	7	Peripheral	User defined 7 bits address.
data_out	OUT	8	Peripheral	8-bit output data port used to transmit data to the user module that is transmitted by the I2C Master during I2C Write. This remains 0 all throughout and is only updated when data_vld is high.
start	OUT	1	Peripheral	Active-HIGH strobe Indicates that an I2C START or REPEAT START condition is generated by the I2C master.
stop	OUT	1	Peripheral	Active-HIGH strobe Indicates either of the following: <ul style="list-style-type: none">• An I2C stop condition is generated by the master.• The I2C slave address sent by the I2C Master does not match the I2C Slave module's address.
data_vld	OUT	1	Peripheral	Active-HIGH strobe. Indicates either of the following: <ul style="list-style-type: none">• If r_w = 0, the I2C Master requested a write transaction. The User Module can fetch the data from the data_out port.• If r_w = 1, the I2C Master requested a read transaction. The User Module should feed the data to the data_in port. The data should be held at the same value until the next data_vld strobe.
r_w	OUT	1	Peripheral	0 – Indicates an I2C Write Operation (user module to receive). 1 – Indicates an I2C Read Operation (user module to transmit).

Figura 3: Descripción de señales.

3. Desarrollo

Pensando al sistema como una máquina de estados, pueden definirse 4 estados posibles: reposo, detección de address (y modo de trabajo), lectura y escritura. Con esto en cuenta, el dispositivo final fue diseñado como la interconexión de submódulos más simples. A continuación, se detallará cada uno de estos submódulos y su integración en el sistema final.

3.1. Top Module

3.1.1. Puertos

- SCL: Bidireccional.
- SDA: Bidireccional.
- rst: Entrada.
- a_j : Entrada (bus 8 bits).
- b_j_i : Entrada (bus 8 bits).
- R_W: Salida.
- data_vld: Salida.
- start: Salida.
- stop: Salida.
- b_j_o : Salida (bus 8 bits).

3.1.2. Descripción

En las Figuras 4, 5 y 6, se presenta el esquemático, símbolo y layout del módulo completo. Aquí se realiza una primera partición del sistema como la conexión entre el módulo SS_detector y el módulo I2C_submodule. El primero es el encargado de la detección de los bits de start y stop y generar las interrupciones correspondientes, mientras que el segundo, a partir de dichas interrupciones, implementa las funcionalidades de lectura, escritura y detección de address, junto a la comunicación requerida con el módulo usuario.

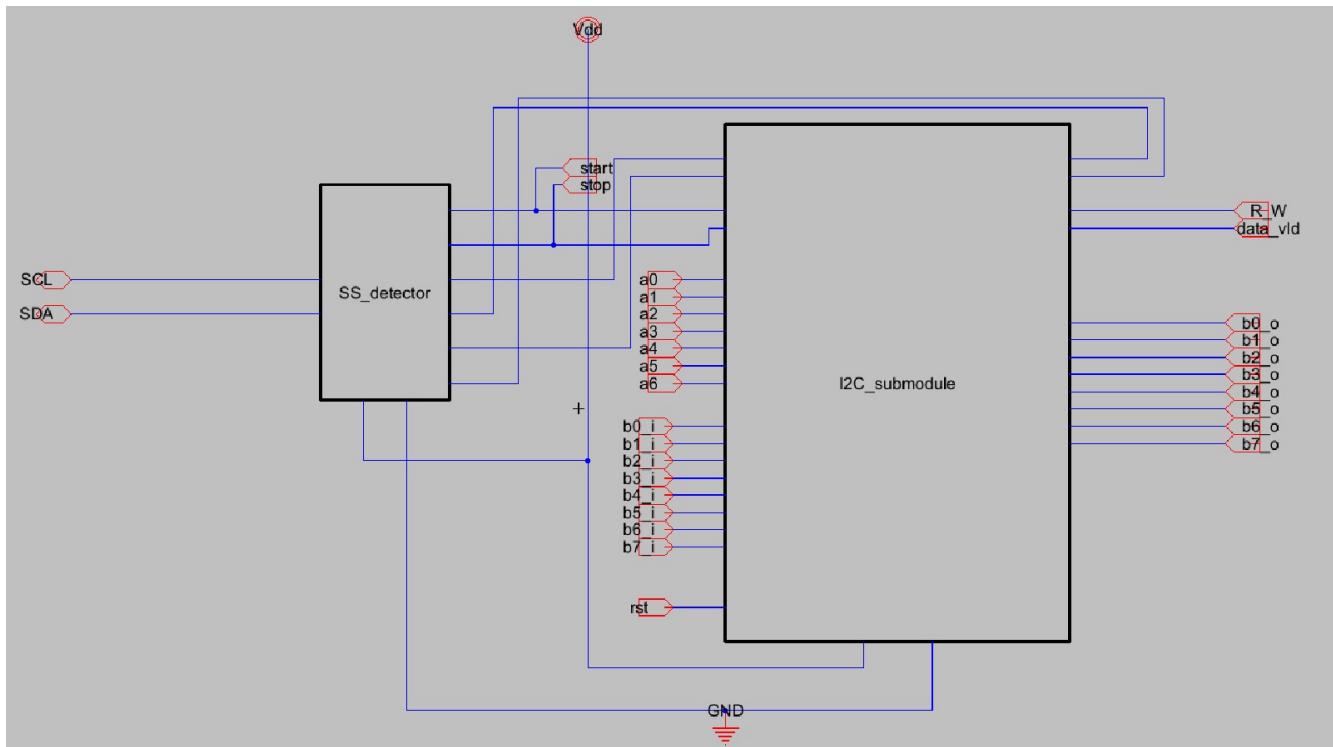


Figura 4: Top Module - Esquemático.

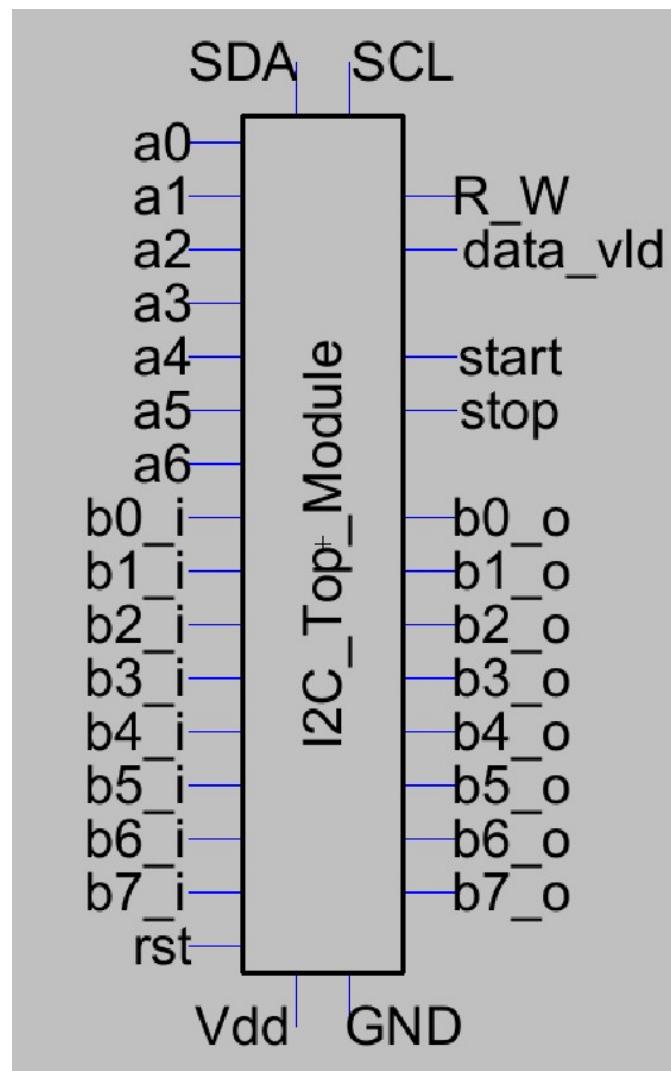


Figura 5: Top Module - Símbolo.

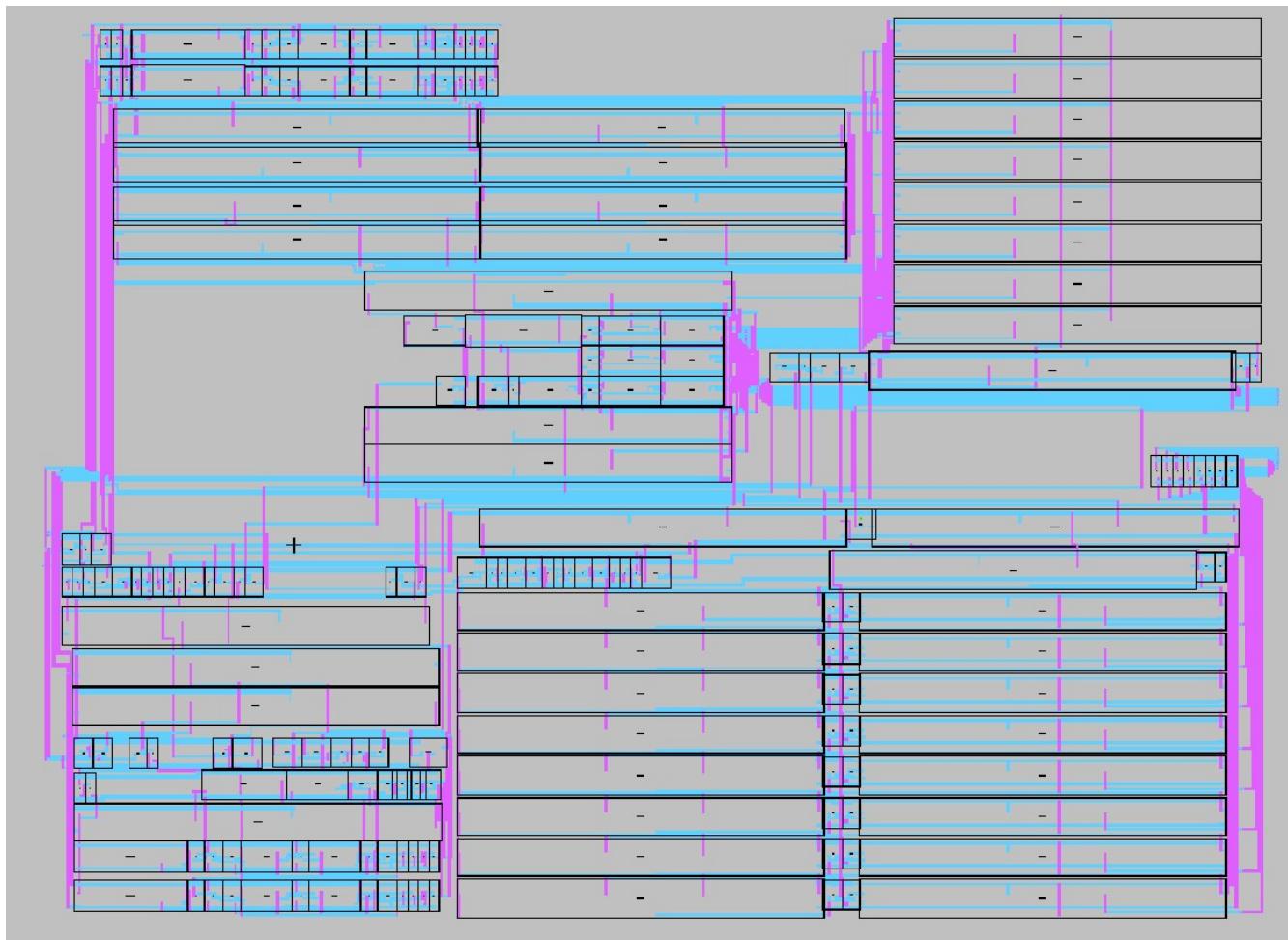


Figura 6: Top Module - Layout.

3.2. Start-stop detector Module

3.2.1. Puertos

- SCL: Bidireccional.
- SDA: Bidireccional.
- I2C_START: Salida.
- I2C_STOP: Salida.
- SDA_READ: Salida.

- SDA_WRITE: Salida.
- SCL_READ: Salida.
- SCL_WRITE: Salida.

3.2.2. Descripción

El circuito de detección de Start/Stop "SS_detector" se presenta en las figuras 7, 8 y 9. Este, cuando la línea SCL se encuentra en alto, se encarga de detectar flancos positivos y negativos, correspondientes a los bits de start y stop, respectivamente. Cuando esto ocurre, envía pulsos por las salidas "I2C_START" e "I2C_STOP". Simultáneamente, toma las líneas bidireccionales SDA y SCL y las separa en Entrada ("SDA_WRITE" y "SCL_WRITE") y salida ("SDA_READ" y "SCL_READ").

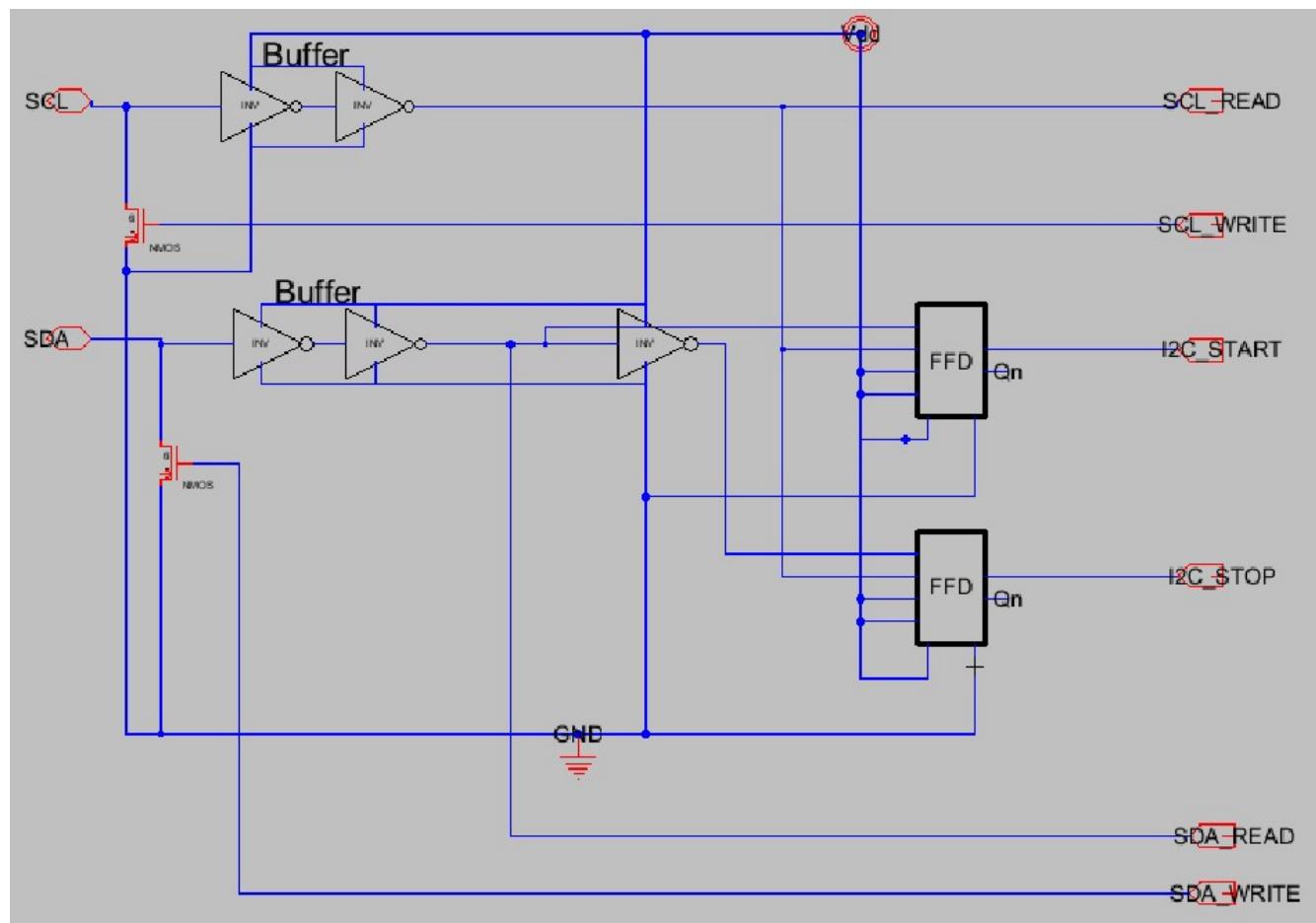


Figura 7: SS Detector - Esquemático.

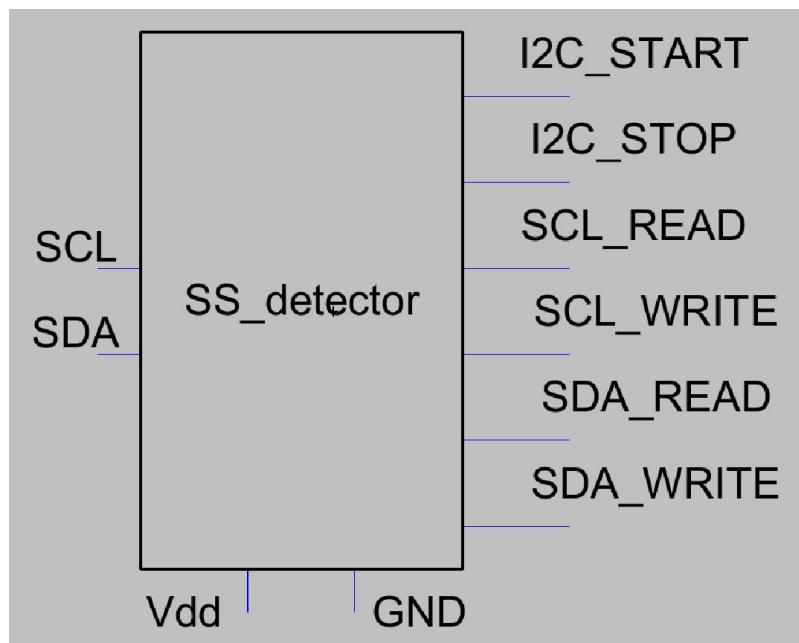


Figura 8: SS Detector - Símbolo.

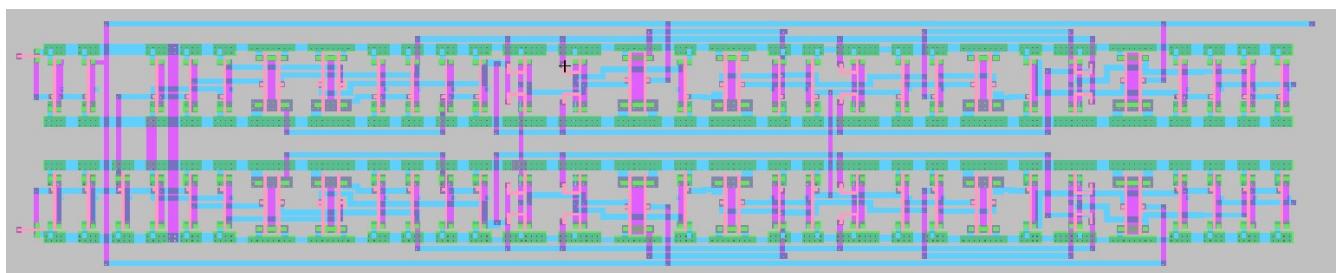


Figura 9: SS Detector - Layout.

3.3. I2C_submodule Module

3.3.1. Puertos

- rst: Entrada.
- start: Entrada.
- stop: Entrada.
- SCL_R: Entrada.
- SDA_R: Entrada.
- b_j _i: Entrada (bus 8 bits).
- a_j : Entrada (bus 8 bits).
- SCL_W: Salida.
- SDA_W: Salida.
- R_W: Salida.
- data_vld: Salida.
- b_j _o: Salida (bus 8 bits).

3.3.2. Descripción

El módulo I2C_submodule (Figuras 10, 11 y 12), controlado por las interrupciones de inicio (start) y parada (stop), se encarga de leer la línea SDA para detectar si la dirección recibida es la correcta. Si es así, implementa la comunicación tanto con el módulo maestro, a través del bus I2C, como con el módulo usuario.

I2C_submodule está compuesto por los módulos "Counter_module", "Add_detect_module", "R_W_module", "CLK_NS", "SR_SIPO_8b", "buff_8bit", "inv_8b" y algunas compuertas lógicas adicionales.

3.3.3. Funcionamiento

1. Al llegar un pulso en la Entrada "start", se resetean los módulos internos y "Count_module" comienza a contar en simultáneo que "SR_SIPO_8b" comienza a recibir y paralelizar los bits de address y R_W desde la línea SDA.
2. Al contar hasta 8, el flag "first_count_f" de "Count_module" se pone en alto y "add_detect_module" determina si el address recibido coincide con el ingresado por el bus " a_j ".

- a) Si la dirección no coincide, el dispositivo se mantendrá inactivo y esperará al próximo bit de start.
- b) Si coincide, "add_detect_module" enviará un bit de ACK, indicara el modo de funcionamiento por su salida R_W (También salida de "I2C_submodule") y pondrá en alto el flag "add_match".
3. Habiendo coincido, el funcionamiento posterior depende del modo "R_W" indicado.
- a) Si "R_W"=0 (Escritura), el módulo esclavo debe leer sobre la línea SDA. "Count_module" vuelve a contar hasta 8. Cuando esto ocurre, "count_f" se pone en alto por un ciclo junto a "data_vld", se escribe un "ACK" en la línea SDA y se entregan los 8 bits de datos por el bus "b_j_o". Luego, el contador se reinicia y se repite este proceso hasta que llegue el bit de stop.
- b) Si "R_W"=1 (Lectura), el módulo esclavo debe escribir sobre la línea SDA. Al comenzar, "data_vld" se pone en alto, indicando al usuario que debe ingresar los datos a enviar por el bus "b_j_i". "Write_module" comienza a tomar estos datos en el ciclo siguiente y los escribe uno por uno a través de "SDA_WRITE". Una vez que "Count_module" vuelve a contar hasta 8, "count_f" se pone en alto por un ciclo. Al ciclo siguiente, se verifica si el módulo maestro ha escrito un ACK sobre la línea SDA. Si no lo hizo, la comunicación se detiene. Si lo hizo, "data_vld" se vuelve a poner en alto, repitiendo el proceso para enviar una nueva palabra de 8 bits. Este proceso se repite indefinidamente hasta que llegue el bit de stop.

3.3.4. Comentarios adicionales

- La Entrada "rst" puede utilizarse para resetear manualmente en módulo.
- El módulo "CLK_NS" se utiliza para tener dos clocks desfasados 180°, que no se solapan entre sí.

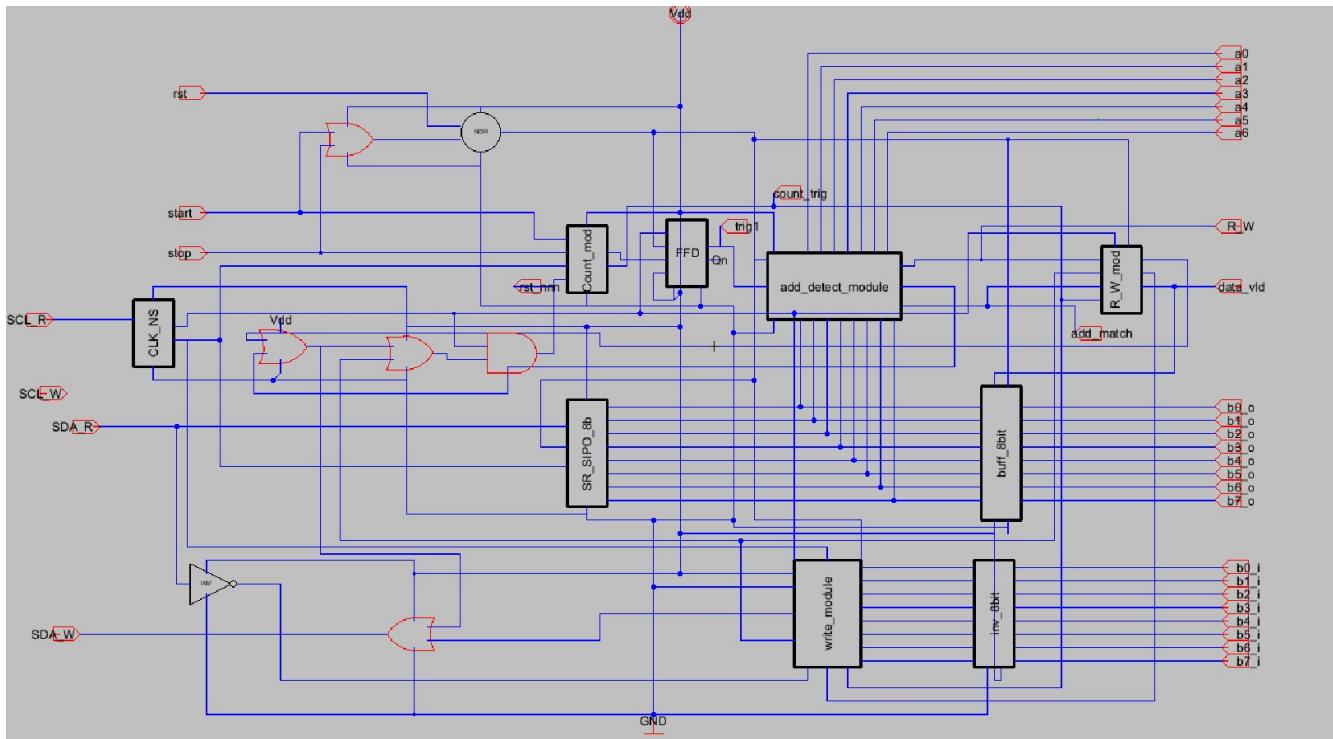


Figura 10: I2C Submodule - Esquemático.

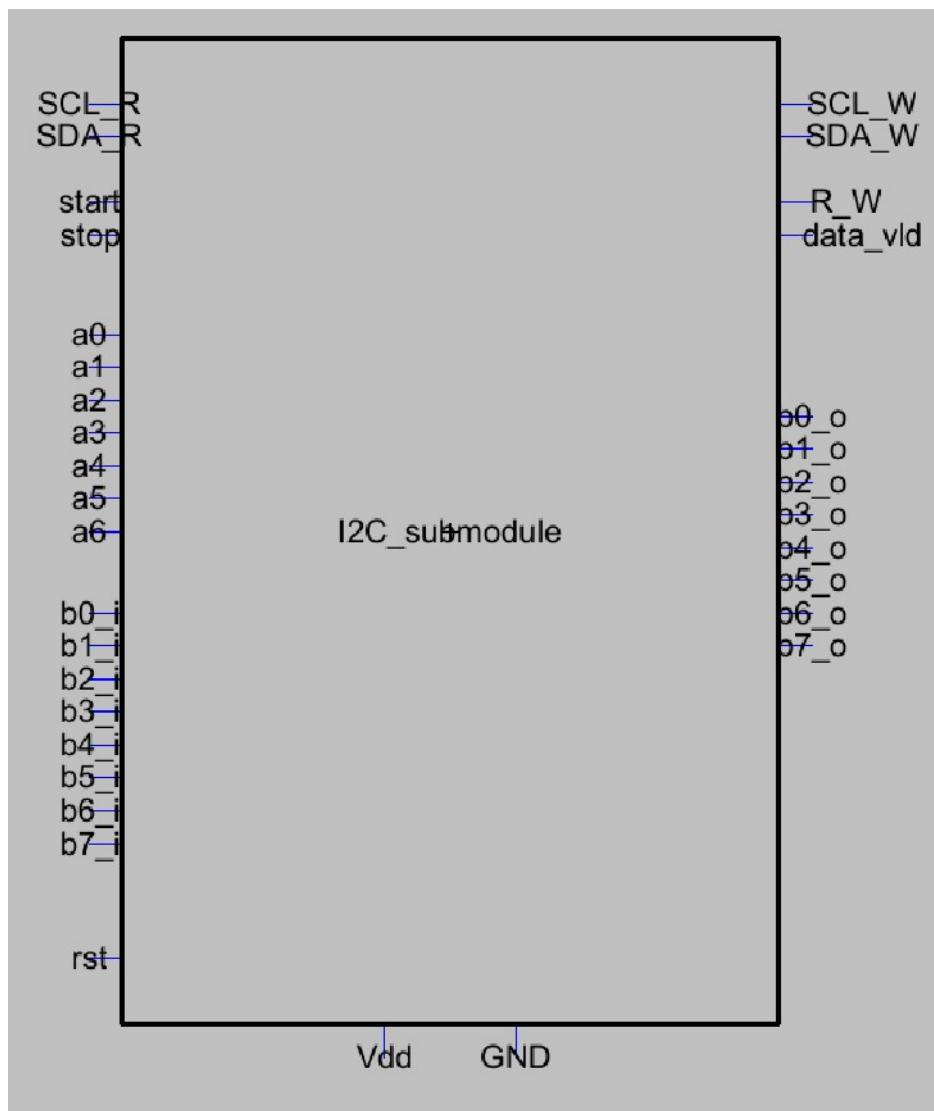


Figura 11: I2C Submodule - Símbolo.

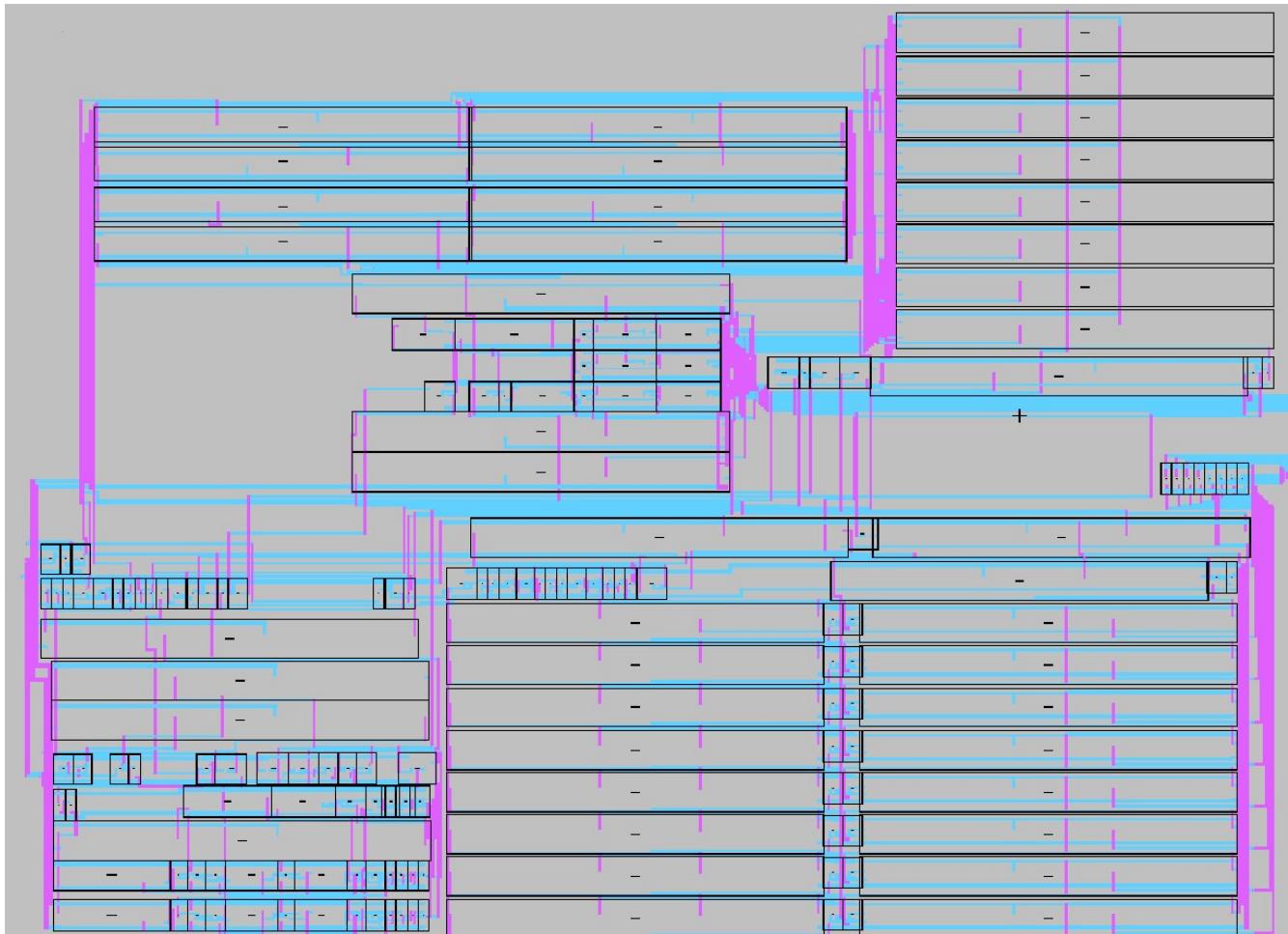


Figura 12: I2C Submodule - Layout.

3.4. Counter Module

3.4.1. Puertos

- Start: Entrada.
- Stop: Entrada.
- Rst: Entrada.
- CLK: Entrada.
- count_f: Salida.

- 1st_count_f: Salida.

3.4.2. Descripción

“Count_mod” es un módulo que no tiene una funcionalidad independiente del resto del sistema. Está compuesto por un contador de 3 bits y lógica adicional pensada para controlar el contador y generar las señales requeridas por el resto del módulo.

Al llegar el bit de start, el contador es habilitado y reseteado simultáneamente. Al llegar a 7 (3'b111) por primera vez, la salida “1st_count_f” se pone en alto y se mantiene así hasta que el módulo reciba una nueva señal de start o stop. Por overflow, el contador se resetea automáticamente al llegar a 7 y continua contando indefinidamente, emitiendo un pulso de un ciclo por “count_f” cada vez que se da esta condición.

La cantidad de bits que el sistema completo requiere contar no es siempre la misma. Aun así, “Count_mod” cumple con las necesidades del sistema, ya que, con lógica externa, el dispositivo es rese teado en momentos clave para que las salidas de este módulo indiquen los tiempos donde el resto de los módulos deben tomar acciones.

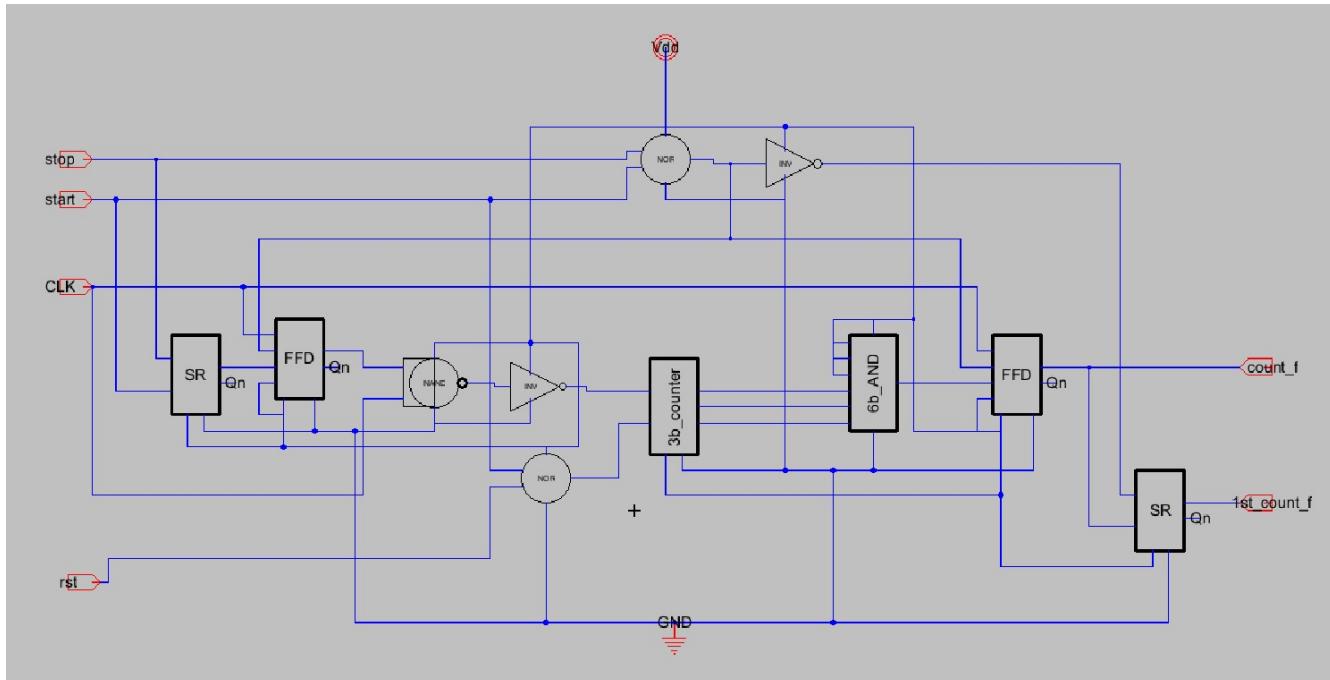


Figura 13: Counter module - Esquemático.

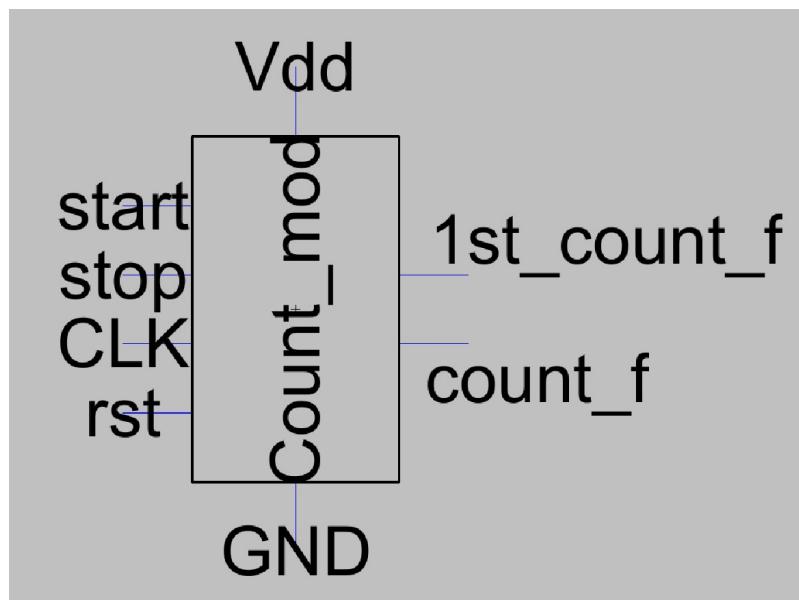


Figura 14: Counter module - Símbolo.

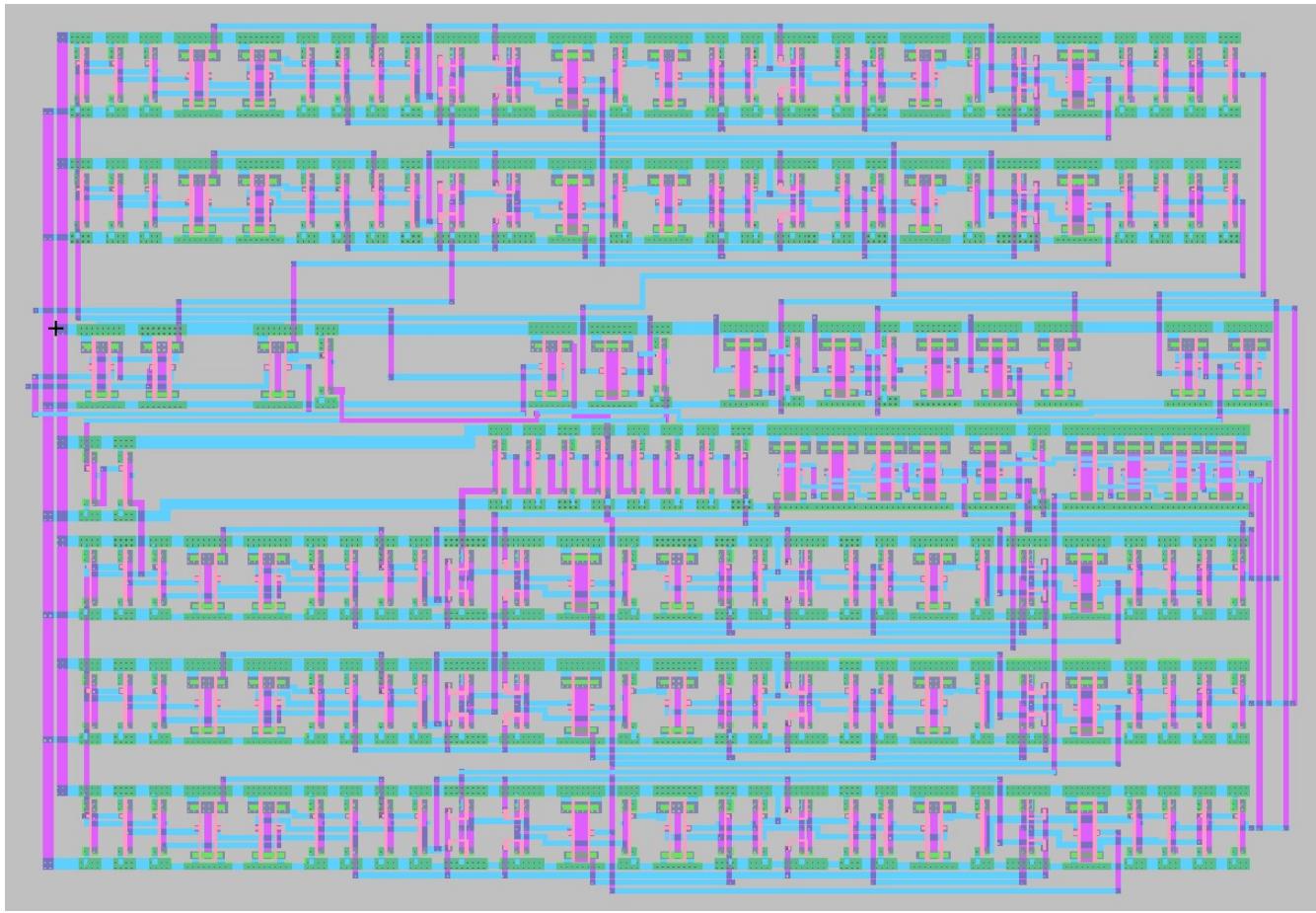


Figura 15: Counter module - Layout.

3.5. Address Detection Module

3.5.1. Puertos

- CLK: Entrada.
- rst_n: Entrada.
- trigg: Entrada.
- b_j (8b): Entrada.
- a_j (7b): Entrada.
- R_W: Salida.
- Add_match: Salida.
- ACK: Salida.

3.5.2. Descripción

El módulo "add_detect_module" es el encargado de recibir el primer conjunto de 8 bits recibido por el módulo I2C maestro, determinar si el address recibido corresponde al propio, indicar el modo de funcionamiento y enviar el bit de ACK, si corresponde.

"add_detect_module" está compuesto por una compuerta AND, una compuerta XOR, 3 Flip FLoPs D y el módulo "14b_EQUAL" que compara los 7 bits del bus " a_j " con los bits de dirección recibidos, correspondientes a los primeros 7 bits del bus " b_j ".

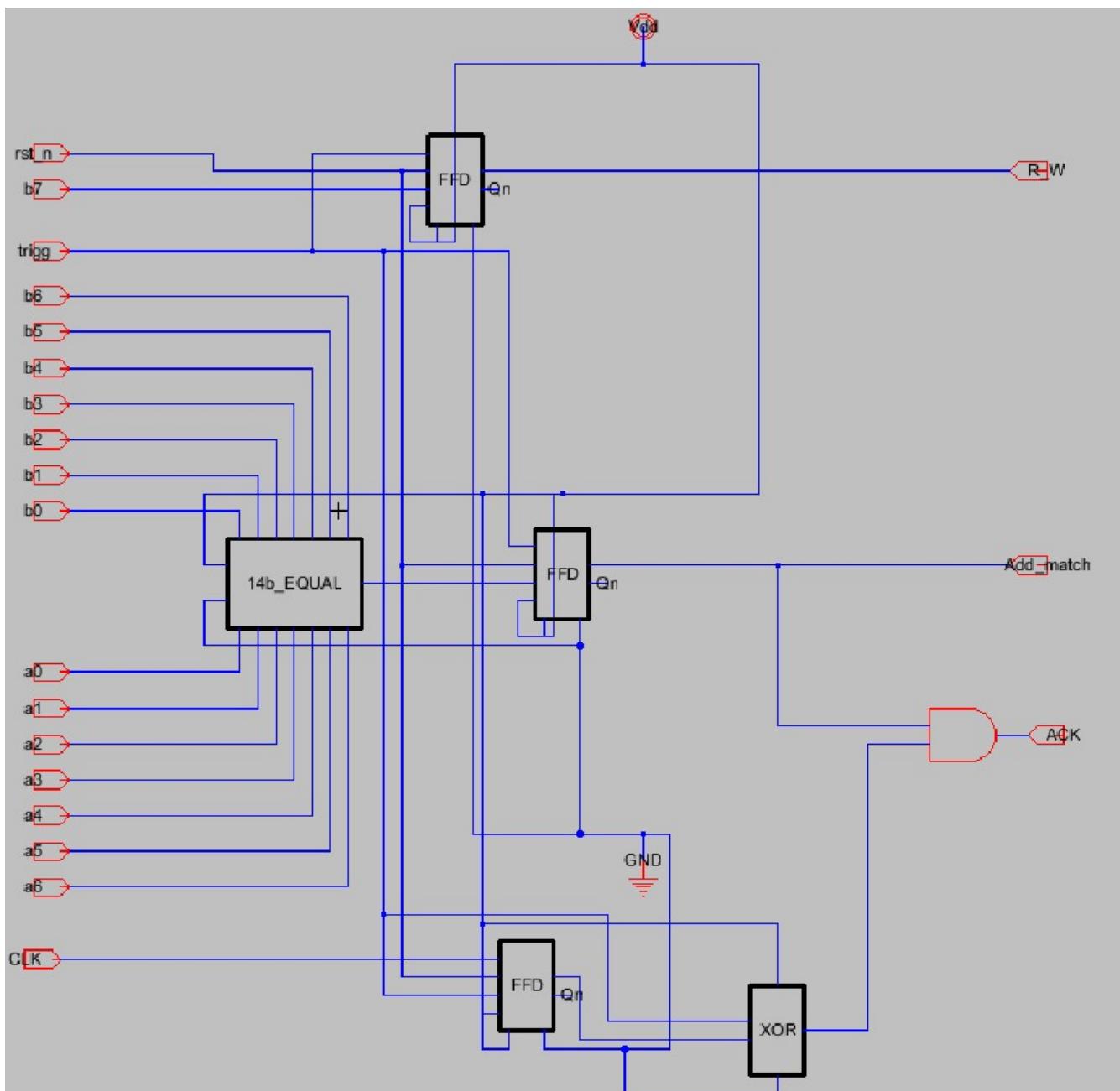


Figura 16: Address Detection Module - Esquemático.

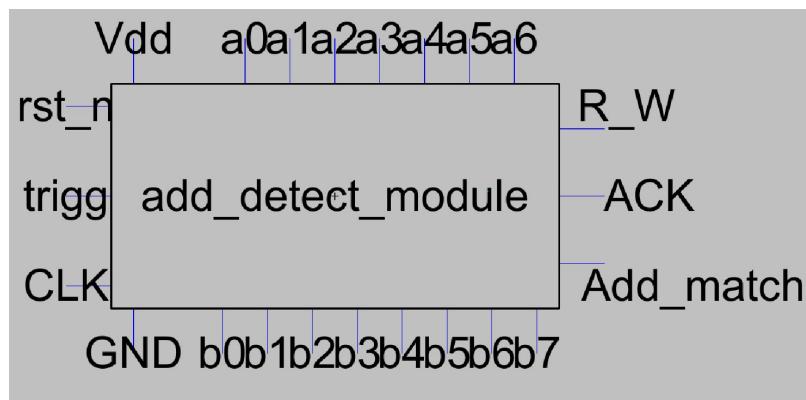


Figura 17: Address Detection Module - Símbolo.

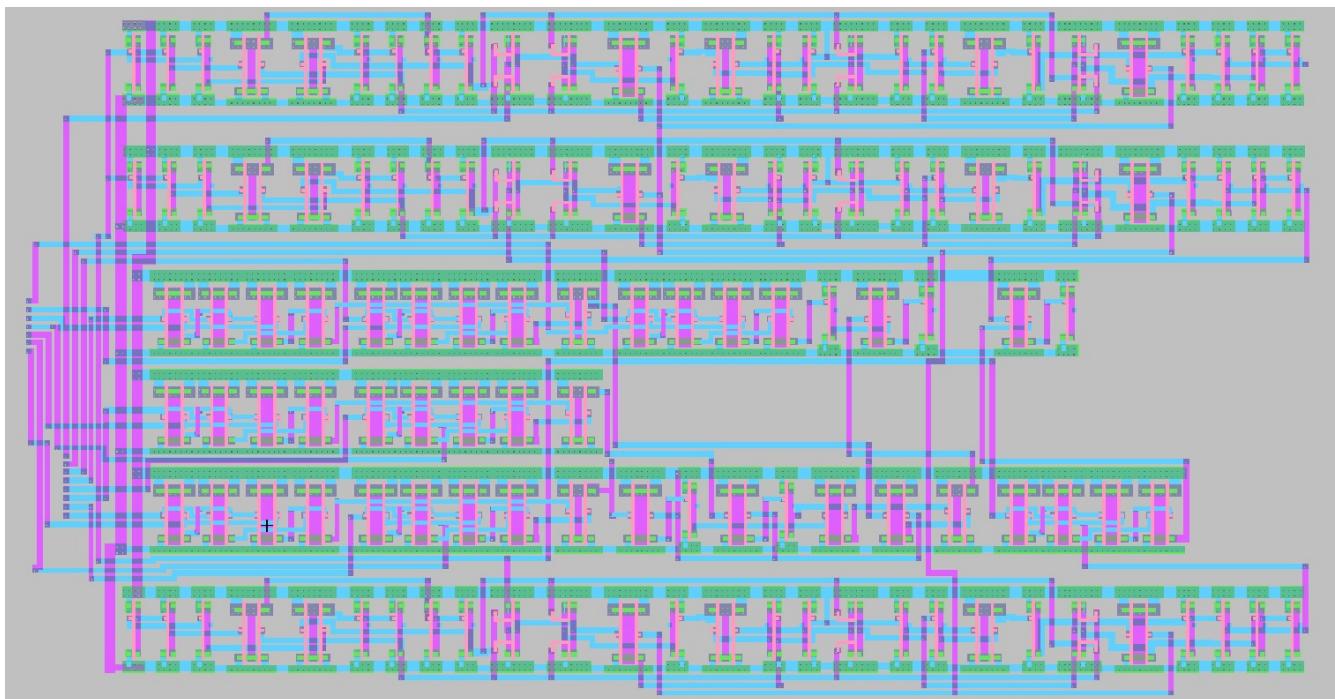


Figura 18: Address Detection Module - Layout.

3.6. Write Module

3.6.1. Puertos

- Trigger: Entrada.
- b_j (8b): Entrada.
- Rst_n: Entrada.
- R_W_Mode: Entrada.
- CLK: Entrada.
- CLK_n: Entrada.
- ACK: Entrada.
- Vout: Salida.
- data_vld: Salida.

3.6.2. Descripción

El módulo "Write_module" implementa la escritura de datos sobre la línea SDA. Cuando "trigger" está en alto y el reloj pasa de bajo a alto, si "R_W" se encuentra en 1 (modo lectura), el módulo verifica si se ha recibido un ACK. Si se ha recibido, se emite un pulso por "data_vld" al mismo tiempo que se pone en 1 la Entrada "LOAD" del shift register de 8 bits "SR_PISO". En el próximo ciclo de reloj, se cargan los datos que el módulo usuario debía ingresar por el bus " b_j " durante el pulso de data_vld y se comienzan a enviar uno por uno en la salida "Vout" por cada ciclo de reloj.

"Write_module" está compuesto por 3 compuertas AND, 3 Flip FLops D y el módulo "SR_PISO", un shift register Parallel-Input Serial-Ouput de 8 bits.

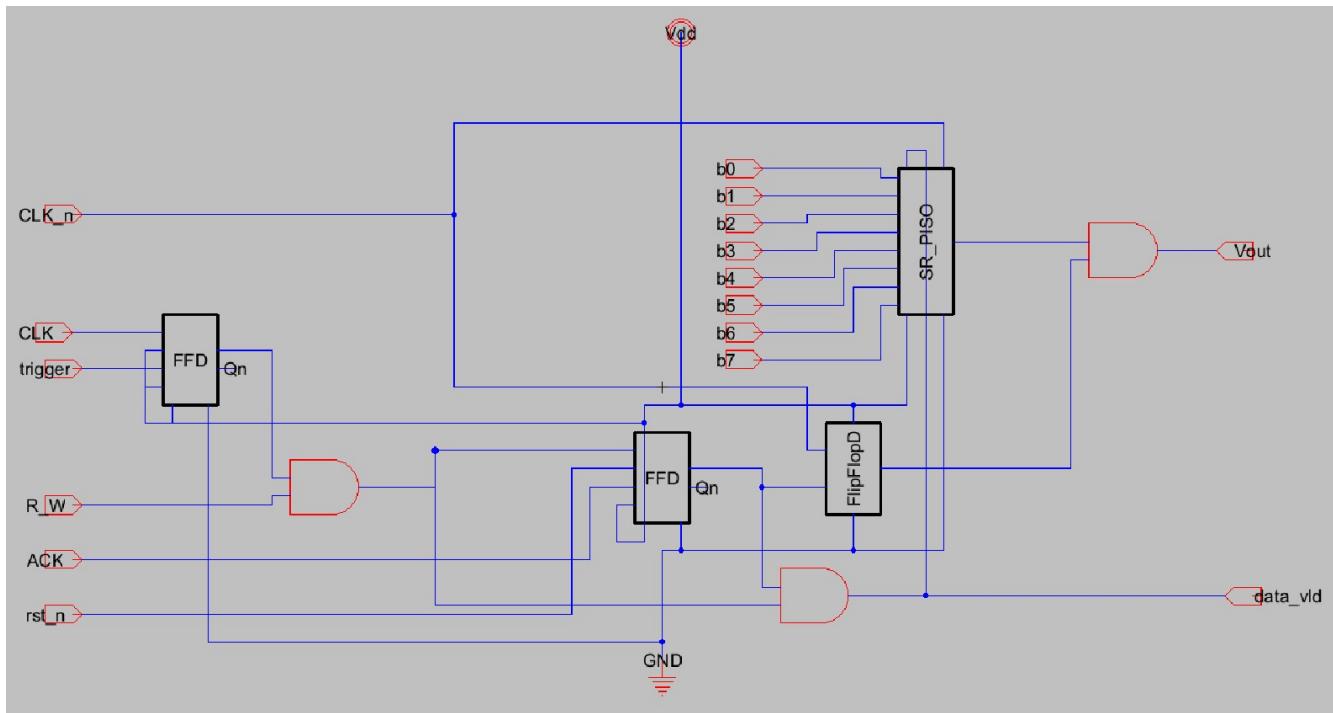


Figura 19: Write Module - Esquemático.

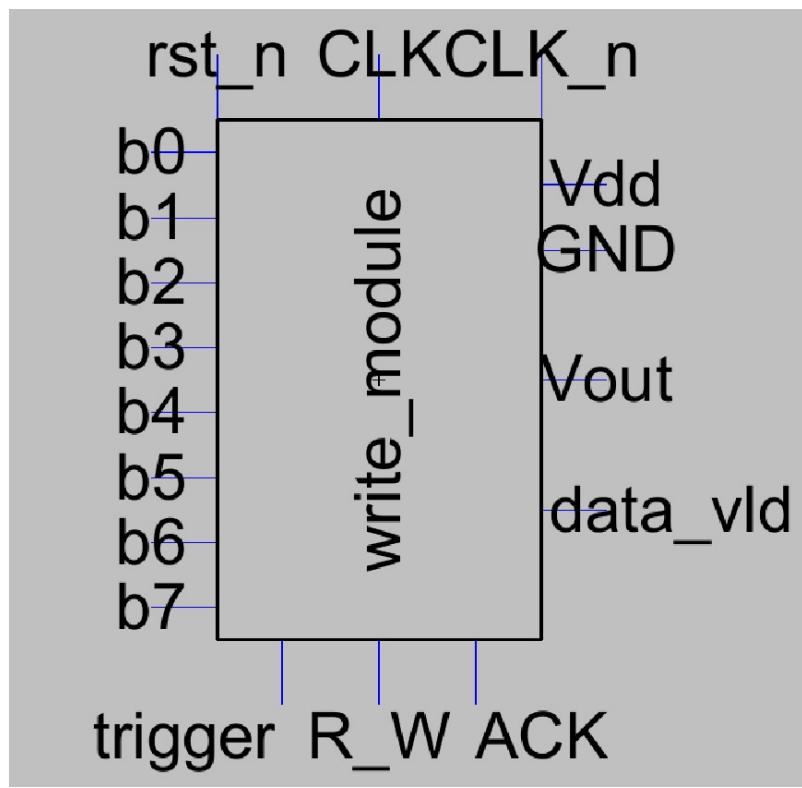


Figura 20: Write Module - Símbolo.

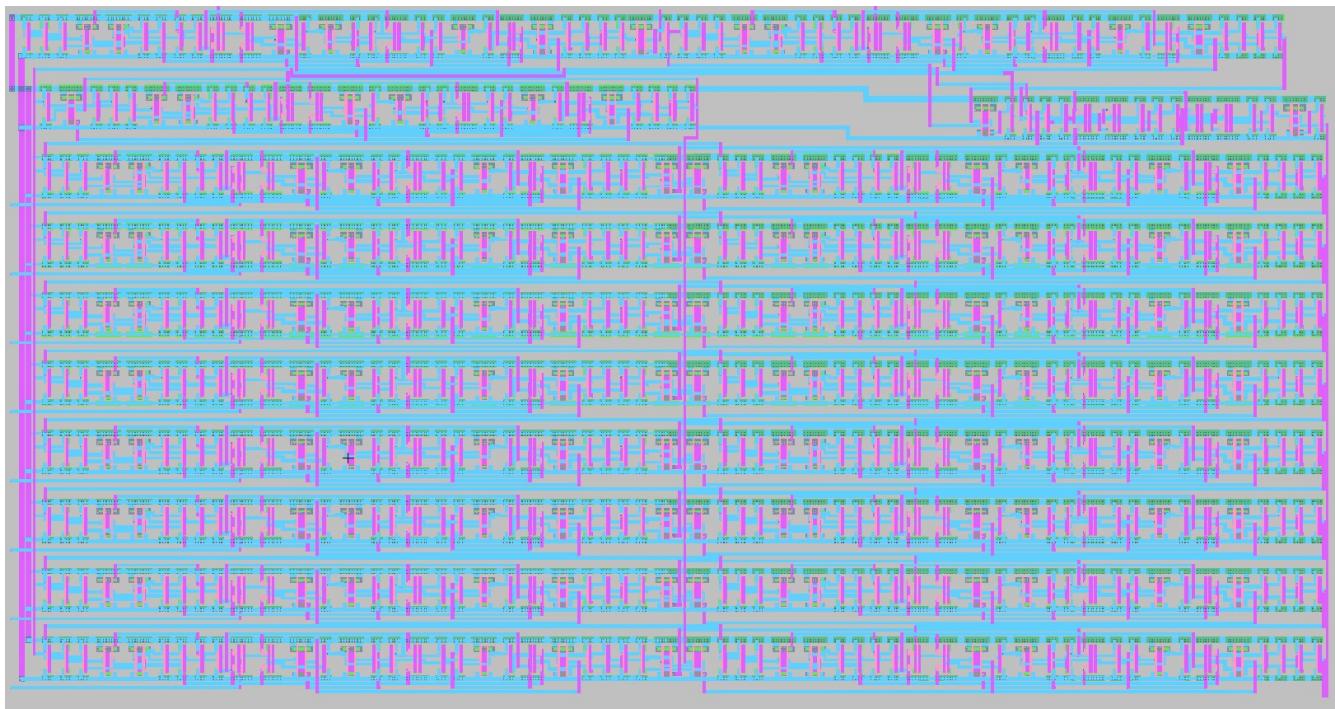


Figura 21: Write Module - Layout.

3.7. R_W Module

3.7.1. Puertos

- CLK: Entrada.
- rst_n: Entrada.
- trigger: Entrada.
- R_W: Entrada.
- Add_match: Entrada.
- W_data_vld: Entrada.
- ACK_r: Salida.
- data_vld: Salida.
- W_en: Salida.

3.7.2. Descripción

El módulo "R_W_mod" es un conjunto de lógica que permite controlar los modos de escritura y lectura, al mismo tiempo que envía los pulsos en la salida "data_vld" que le indican al usuario que debe cargar o leer datos en los buses del sistema.

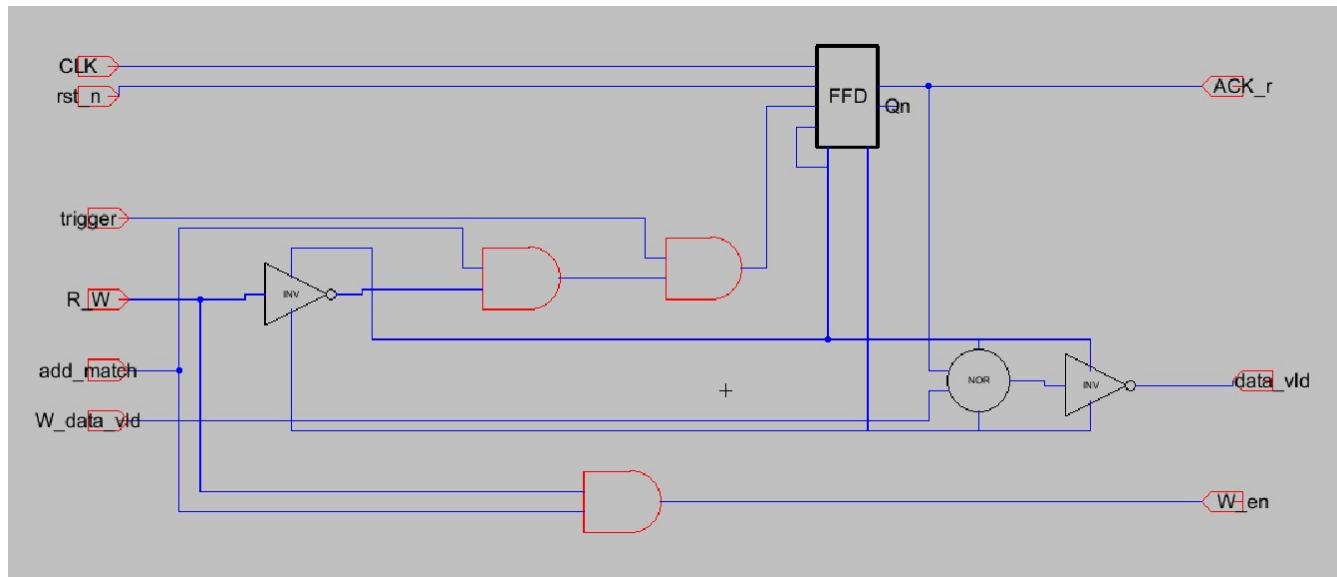


Figura 22: Read Write Module - Esquemático.

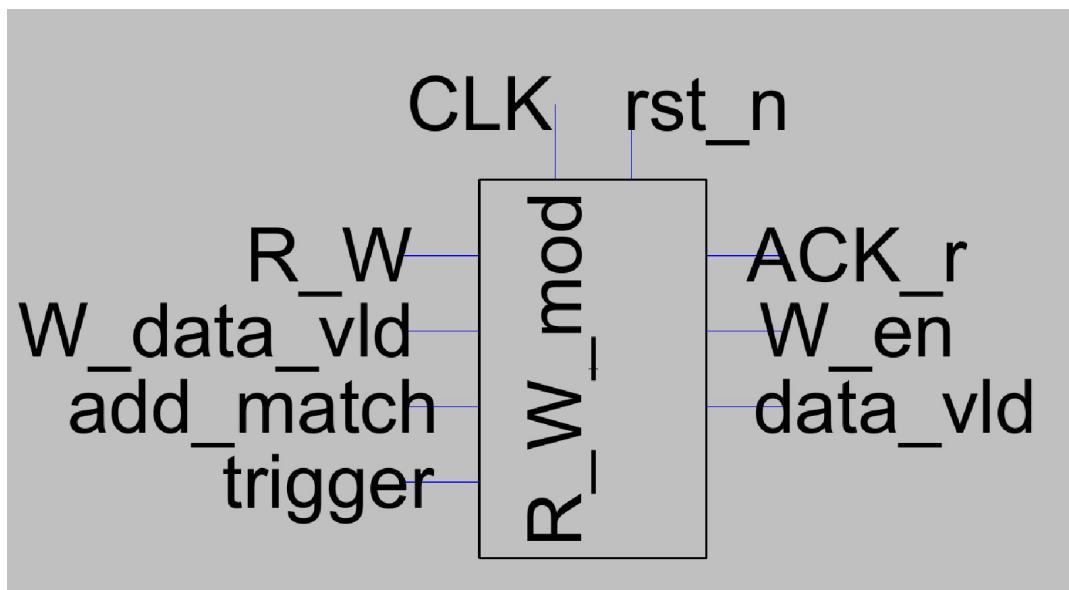


Figura 23: Read Write Module - Símbolo.

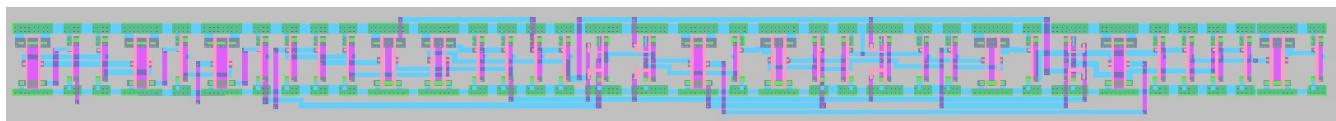


Figura 24: Read Write Module - Layout.

4. Simulaciones

Las simulaciones sobre el sistema fueron realizadas con LTSpice. Para ello, se generó una instancia de “I2C_top_module” y se conectaron múltiples transistores NMOS a la línea SDA (Figura 25) para que, utilizando distintas entradas, se puedan enviar los bits de start y stop, el address y los datos.

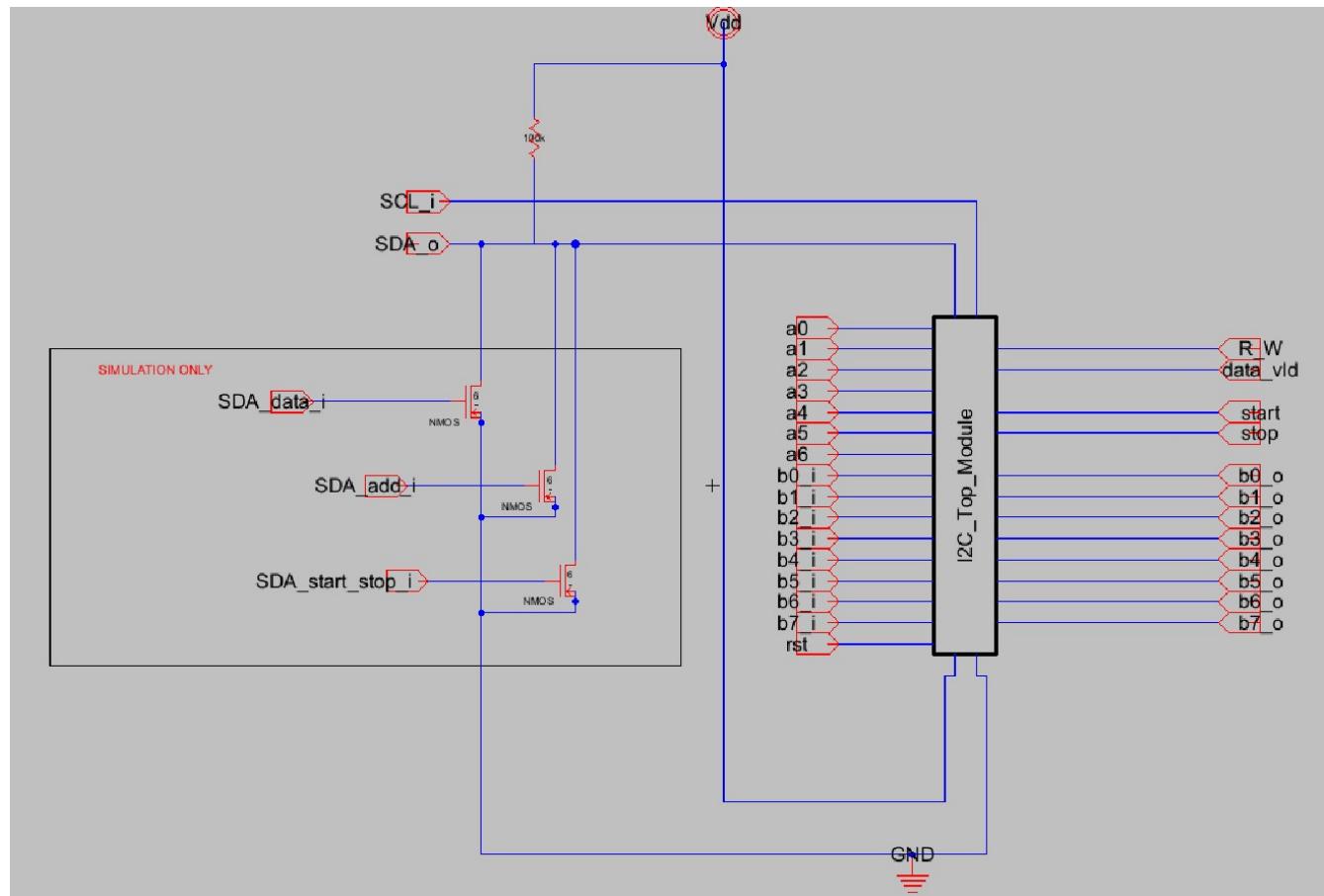


Figura 25: Simulación de Top Module - Esquemático.

4.0.1. Modo lectura

En esta primera simulación, se desea verificar que el módulo identifique su dirección y el modo lectura correctamente, para que luego comience a escribir en la línea SDA los datos recibidos por el bus b_i . Para ello:

1. La señal $SDA_start_stop_i$ se encarga de escribir los bits de start y stop.

2. La señal SDA_add_i envía los bits de address introducidos en el bus a_j junto al bit $R_W = 1$ para indicar el modo lectura.
3. La señal SDA_data_i envía los bits de ACK en los momentos donde el módulo requiere recibirlos para continuar con la transmisión de datos.

A continuación se presenta el código que implementa dicha configuración. Se utilizó una señal de clock de $T = 20\mu S$, lo cual corresponde a una velocidad de 100 kbit/s (Standar Mode).

Listing 1: Top Module Read Mode Test (User Write Mode)

```

1 .INCLUDE Modelos_Transistores.txt
2 .PARAM SUPPLY=5
3 .PARAM maxStep=10uS
4 .PARAM T=20uS
5 .PARAM Ton=10uS
6 .PARAM Tr=1nS
7
8 VDD VDD 0 DC 'SUPPLY'
9 Vrst      rst      0 PULSE 0      'SUPPLY'  {0*'Ton'}   'Tr'  'Tr'  'Ton'
{200*'Ton'}
10
11 Vscl      SCL_i    0 PULSE 0      'SUPPLY'  {0.5*'Ton'}  'Tr'  'Tr'  'Ton'
'T'
12
13 Vstart     SDA_start_stop_i  0 PULSE 'SUPPLY'  0      {9*'Ton'}   'Tr'  'Tr'  {72*'Ton'}
{100*'Ton'}
14 Vsda_add   SDA_add_i     0 PULSE 0      'SUPPLY'  {10*'Ton'}  'Tr'  'Tr'  {2*T}
{4*T}
15 Vsda_ack   SDA_data_i    0 PULSE 0      'SUPPLY'  {44*'Ton'}  'Tr'  'Tr'  'T'
{9*T}
16
17 Va0 a0 0 DC 0
18 Va1 a1 0 DC 0
19 Va2 a2 0 DC 'SUPPLY'
20 Va3 a3 0 DC 'SUPPLY'
21 Va4 a4 0 DC 0
22 Va5 a5 0 DC 0
23 Va6 a6 0 DC 'SUPPLY'
24
25 Vb0_i b0_i 0 PULSE 0 'SUPPLY'  0us      'Tr'  'Tr'  {50*'Ton'}  {80*'Ton'}
26 Vb1_i b1_i 0 PULSE 0 'SUPPLY'  0us      'Tr'  'Tr'  {50*'Ton'}  {80*'Ton'}
27 Vb2_i b2_i 0 PULSE 0 'SUPPLY'  {50*'Ton'}  'Tr'  'Tr'  {50*'Ton'}  {80*'Ton'}
28 Vb3_i b3_i 0 PULSE 0 'SUPPLY'  0us      'Tr'  'Tr'  {50*'Ton'}  {80*'Ton'}
29 Vb4_i b4_i 0 PULSE 0 'SUPPLY'  {50*'Ton'}  'Tr'  'Tr'  {50*'Ton'}  {80*'Ton'}
30 Vb5_i b5_i 0 PULSE 0 'SUPPLY'  0us      'Tr'  'Tr'  {50*'Ton'}  {80*'Ton'}
31 Vb6_i b6_i 0 PULSE 0 'SUPPLY'  {50*'Ton'}  'Tr'  'Tr'  {50*'Ton'}  {80*'Ton'}
32 Vb7_i b7_i 0 PULSE 0 'SUPPLY'  0us      'Tr'  'Tr'  {50*'Ton'}  {80*'Ton'}
33
34 .TRAN 'maxStep'  {100*'Ton'}

```

En la figura 26, se presentan las señales más relevantes de esta simulación. En el panel superior, se encuentran las señales SDA y SCL, en el medio las señales de salida R_W , rst , $data_vld$, $start$ y $stop$, y en inferior cada una de las señales que escriben en la línea SDA (SDA_add_i , SDA_data_i y $SDA_start_stop_i$).

Cuando la señal SDA pasa de bajo a alto, el módulo envía un pulso por la salida $start$ y se prepara internamente para recibir los primeros 8 bits del mensaje. Al hacerlo, detecta que el módulo maestro ha escrito su dirección junto a un bit $R_W = 1$. Por ello, el módulo envía un ACK, la salida R_W se pone en alto y se envía un pulso por $data_vld$ para que el módulo usuario ingrese los datos a enviar por el bus b_i . En este momento, los datos ingresados por este bus son $b_i = 8'b11010101$, los cuales puede observarse como fueron escritos sobre la línea. Luego, SDA_data_i vuelve a escribir un bit de ACK, se envía de nuevo un pulso por $data_vld$ y se escribe la nueva Entrada $b_i = 8'b11010101$ (igual a la anterior). Finalmente, para el tercer mensaje el bus cambia a $b_i = 8'b00101010$. Luego de este mensaje, no se escribe un ACK y se envía un bit de stop, para lo cual el módulo reacciona bajando la salida R_W y enviando un pulso por la salida $stop$.

Observe que, los pulsos enviados por la salida $data_vld$ están retrasados medio ciclo respecto al fin de cada palabra de 8 bits. Esto se debe a un funcionamiento interno del módulo esclavo y se considera que el medio ciclo restante es tiempo suficiente para que el módulo usuario cargue los datos a enviar.



Figura 26: Simulación modo lectura - LTSpice Plot.

4.0.2. Modo escritura

En la simulación del modo escritura, se desea verificar que el módulo escriba correctamente en el bus b_o , los datos recibidos desde la línea SDA. Para ello:

1. La señal $SDA_start_stop_i$ se encarga de escribir los bits de start y stop.
2. La señal SDA_add_i envía los bits de address introducidos en el bus a_j junto al bit $R_W = 0$ para indicar el modo escritura.
3. La señal SDA_data_i envía las palabras de 8 bits luego de los ciclos donde se debería recibir un bit de ACK por parte del módulo esclavo.

A continuación se presenta el código que implementa dicha configuración. Se utilizó una señal de clock de $T = 20\mu S$, lo cual corresponde a una velocidad de 100 kbit/s (Standar Mode).

Listing 2: Top Module Write Mode Test (User Read Mode)

```

1 .INCLUDE Modelos_Transistores.txt
2 .PARAM SUPPLY=5
3 .PARAM maxStep=10uS
4 .PARAM T=20uS
5 .PARAM Ton=10uS
6 .PARAM Tr=1nS
7
8 VDD VDD 0 DC 'SUPPLY'
9 Vrst rst 0 PULSE 0 'SUPPLY' {0*'Ton'} 'Tr' 'Tr' 'Ton' {200*'Ton'}
10
11 Vscl SCL_i 0 PULSE 0 'SUPPLY' {0.5*'Ton'} 'Tr' 'Tr' 'Ton' 'T'
12
13 Vstart SDA_start_stop_i 0 PULSE 'SUPPLY' 0 {9*'Ton'} 'Tr' 'Tr' {70*'Ton'} {80*'Ton'}
14 Vsda_add SDA_add_i 0 PULSE 0 'SUPPLY' {14*'Ton'} 'Tr' 'Tr' {2*T} {4*T} 4
15 Vsda_data SDA_data_i 0 PULSE 0 'SUPPLY' {40*'Ton'} 'Tr' 'Tr' {2*T} {4*T} 3
16
17 Va0 a0 0 DC 'SUPPLY'
18 Va1 a1 0 DC 'SUPPLY'
19 Va2 a2 0 DC 0
20 Va3 a3 0 DC 0
21 Va4 a4 0 DC 'SUPPLY'
22 Va5 a5 0 DC 'SUPPLY'
23 Va6 a6 0 DC 0
24
25 Vb0_i b0_i 0 DC 'SUPPLY'
26 Vb1_i b1_i 0 DC 'SUPPLY'
27 Vb2_i b2_i 0 DC 0
28 Vb3_i b3_i 0 DC 'SUPPLY'
29 Vb4_i b4_i 0 DC 0
30 Vb5_i b5_i 0 DC 'SUPPLY'
31 Vb6_i b6_i 0 DC 0

```

```

32 Vb7_i b7_i 0 DC 'SUPPLY'
33
34 .TRAN 'maxStep' {100*'Ton'}

```

En las figuras 27 y 28, se presentan las señales más relevantes de esta simulación. En el panel superior, se encuentran las señales SDA y SCL, en del medio las señales de salida R_W , rst , $data_vld$, $start$ y $stop$, y en inferior cada una de las señales que escriben en la línea SDA (SDA_add_i , SDA_data_i y $SDA_start_stop_i$).

Cuando la señal SDA pasa de bajo a alto, el módulo envía un pulso por la salida $start$ y se prepara internamente para recibir los primeros 8 bits del mensaje. Al hacerlo, detecta que el módulo maestro ha escrito su dirección junto a un bit $R_W = 0$. Por ello, el módulo envía un ACK y la salida R_W se pone en bajo. En este momento, se comienza a escribir la primer palabra de 8 bits combinando las señales SDA_add_i y SDA_data_i , obteniendo $8'b10011000$. Al terminar de escribirse, el módulo esclavo envía un bit de ACK por la línea SDA y transmite un pulso para indicar al módulo usuario que ya puede tomar los datos del bus b_o . Luego, se repite el proceso para la palabra $8'b10011000$ y finalmente, cuando se está transmitiendo la palabra $8'11111111$, se envía un bit de stop que finaliza la comunicación. En este momento, el módulo envía un pulso por la salida $stop$ y queda a la espera del próximo bit de start.

En la figura 28 puede observarse como el bus b_o entrega las palabras correctas y se pone en cero cuando llega el bit de stop.



Figura 27: Simulación modo escritura - LTSpice Plot.

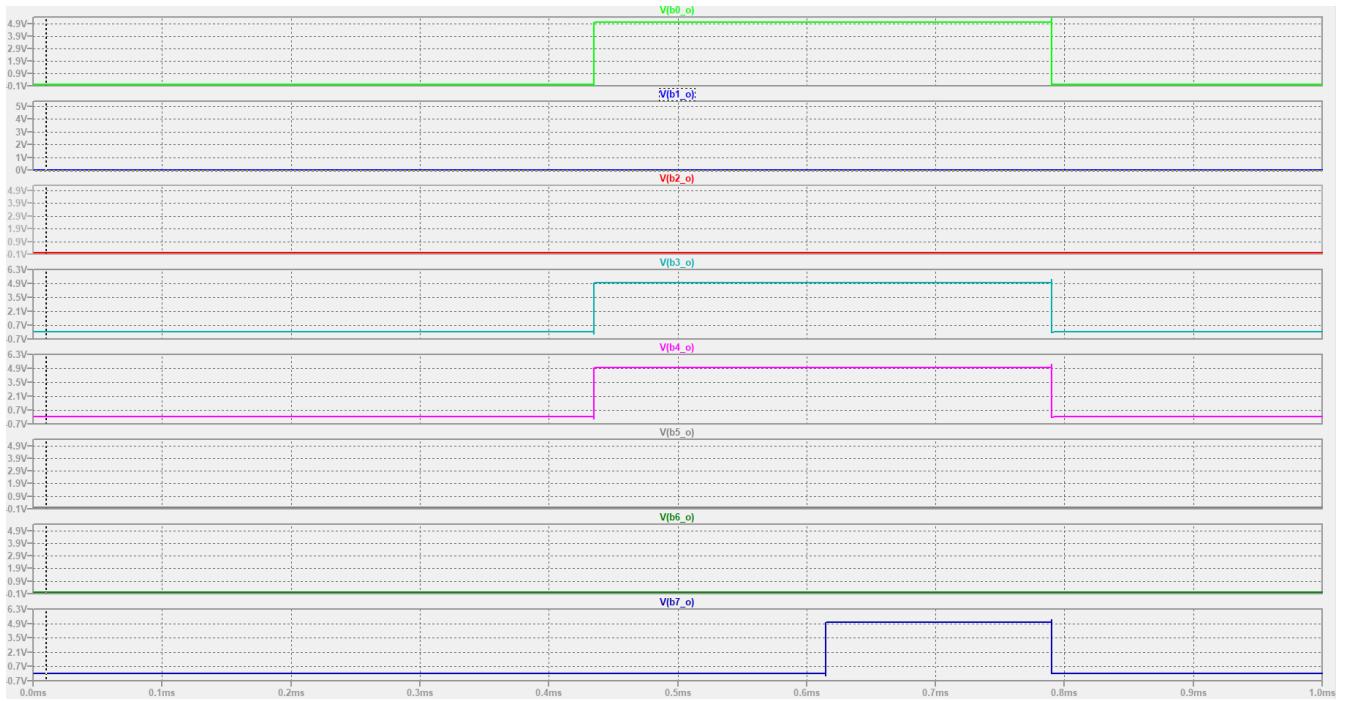


Figura 28: Simulación modo escritura - LTSpice Plot 2.

4.0.3. Velocidad

Finalmente, se decidió verificar si el módulo es capaz de trabajar con todas las velocidades soportadas por el protocolo. Para ello se realizaron las mismas simulaciones desarrolladas para todos los siguientes casos:

1. Standard Mode: Hasta 100 kbit/s.
2. Fast Mode: Hasta 400 kbit/s.
3. Fast Mode Plus: Hasta 1 Mbit/s.
4. High-Speed Mode: Hasta 3.4 Mbit/s.
5. Ultra-Fast Mode: Hasta 5 Mbit/s.

Los cuales fueron realizadas sin problema alguno. En la figura 29, se presenta la simulación para el modo lectura a 5 Mbit/s.

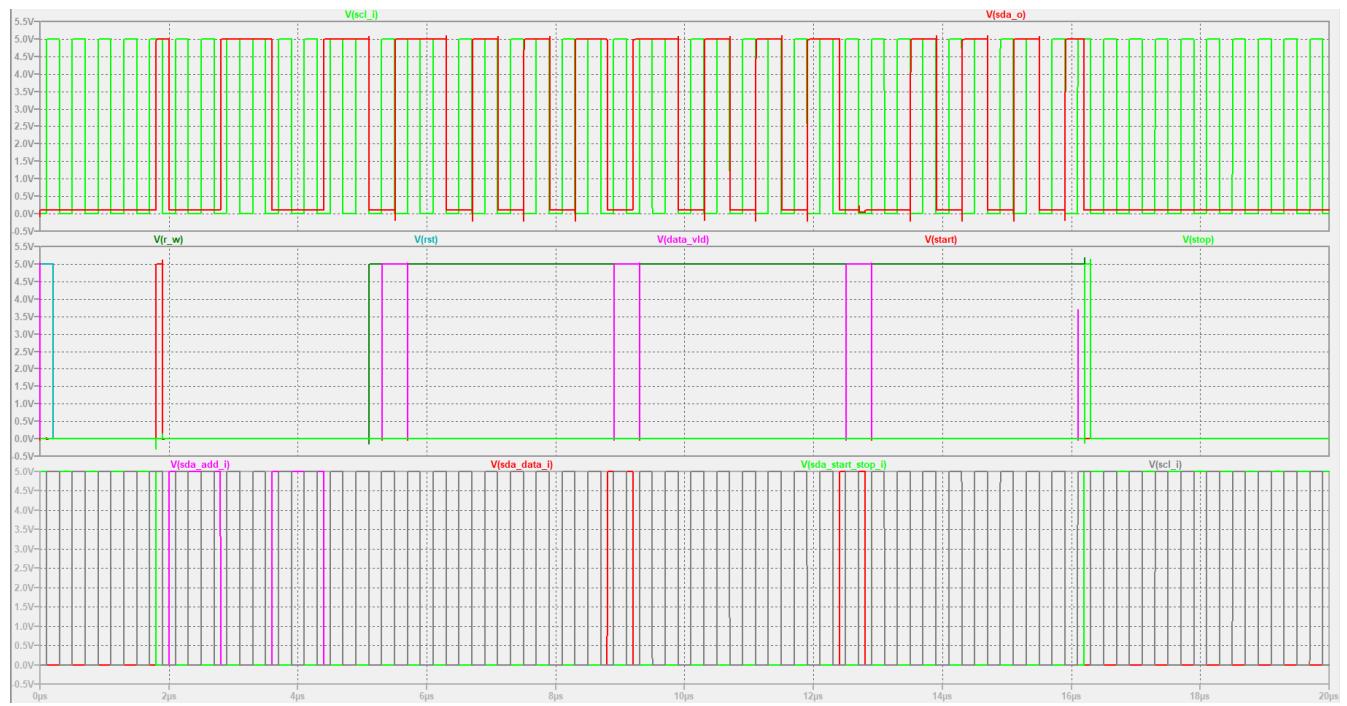


Figura 29: Prueba de velocidad - LTSpice Plot 2.

5. Conclusiones

El desarrollo y la simulación de este módulo I2C han demostrado no solo la viabilidad del diseño propuesto sino también su robustez y adaptabilidad a todas las velocidades definidas por el protocolo, incluyendo el modo Ultra Fast. Estos resultados proporcionan una base sólida para su implementación en una variedad de aplicaciones de VLSI, posicionándolo como una solución confiable para sistemas de comunicación I2C.

6. Appendices

A. Listado de modulos completo.

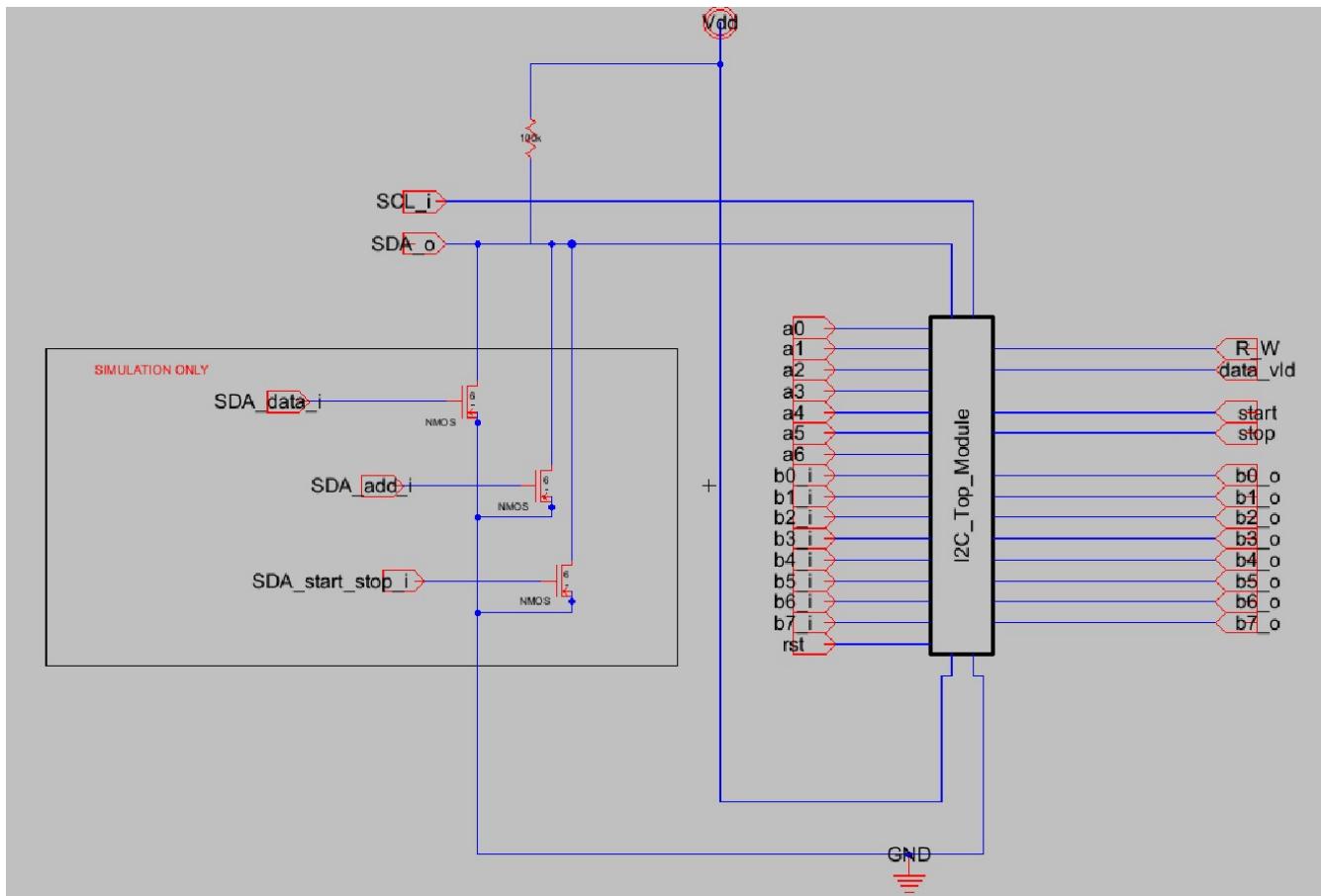


Figura 30: Simulacion de Top Module - Esquemático.

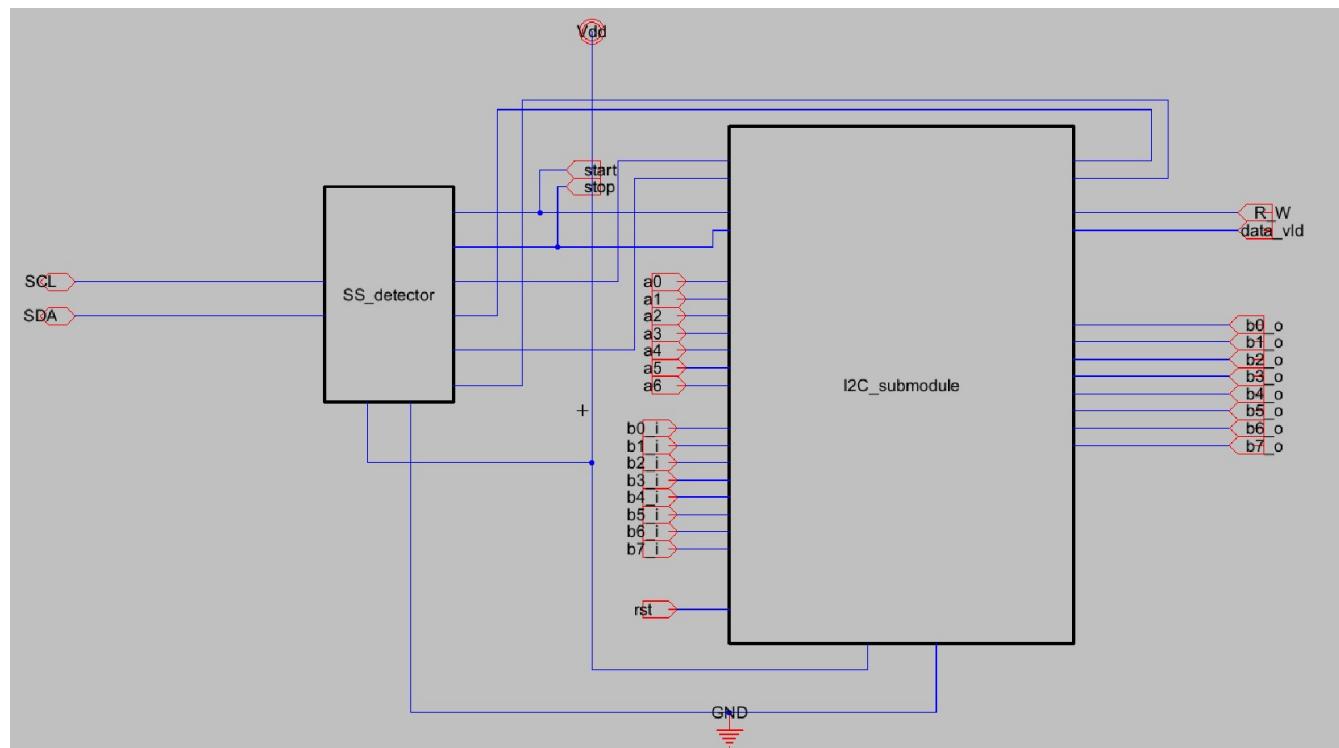


Figura 31: Top Module - Esquemático.

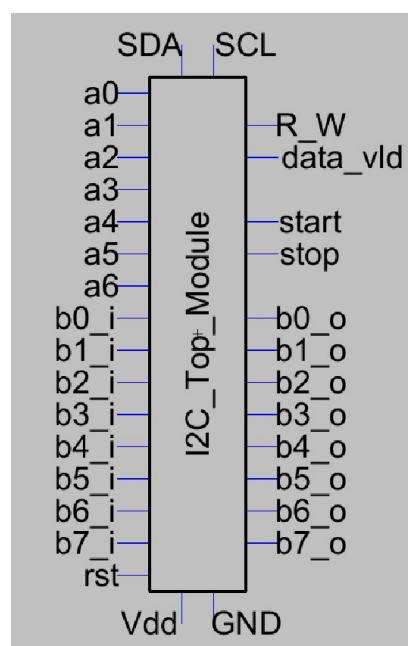


Figura 32: Top Module - Símbolo.

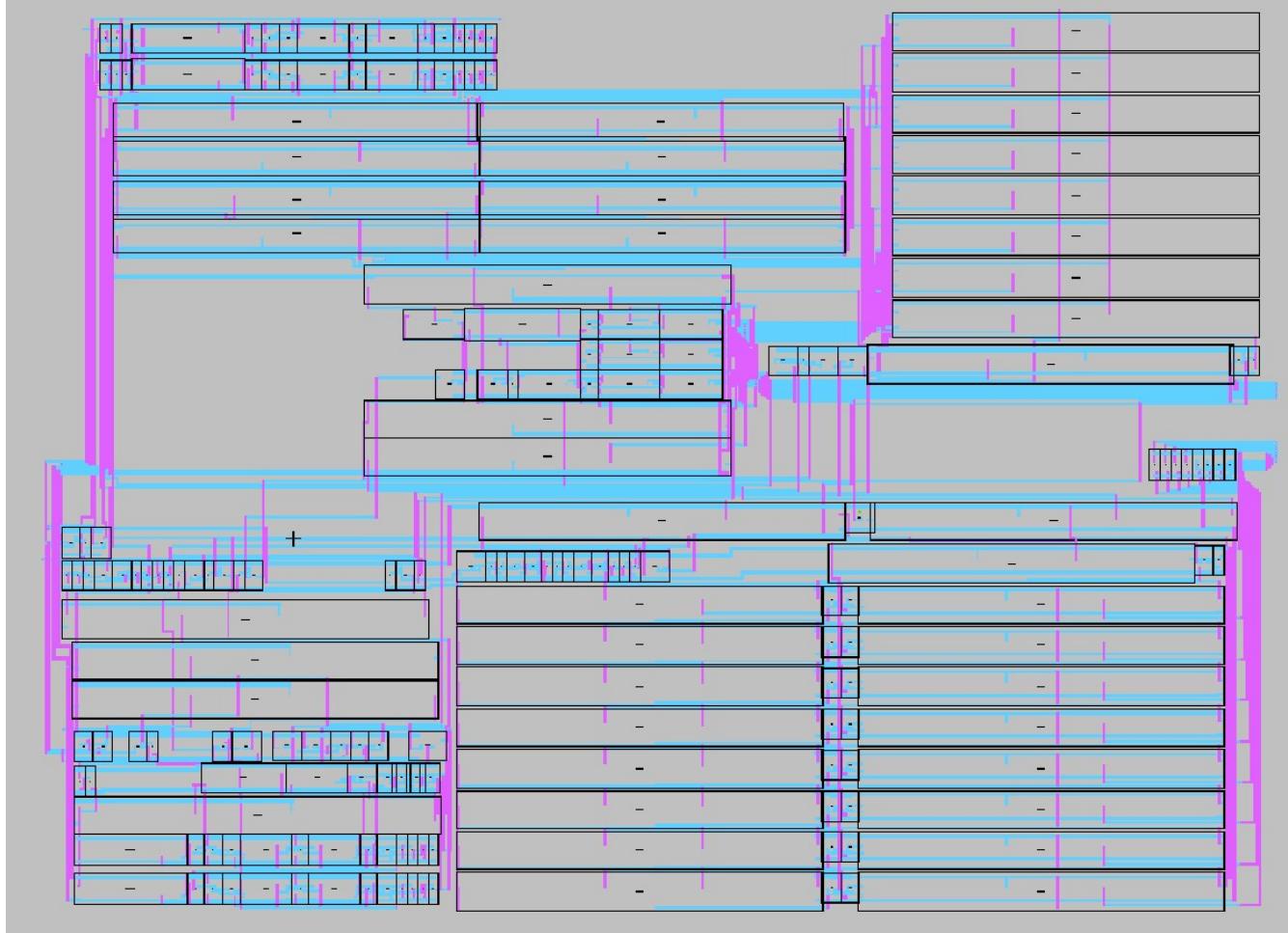


Figura 33: Top Module - Layout.

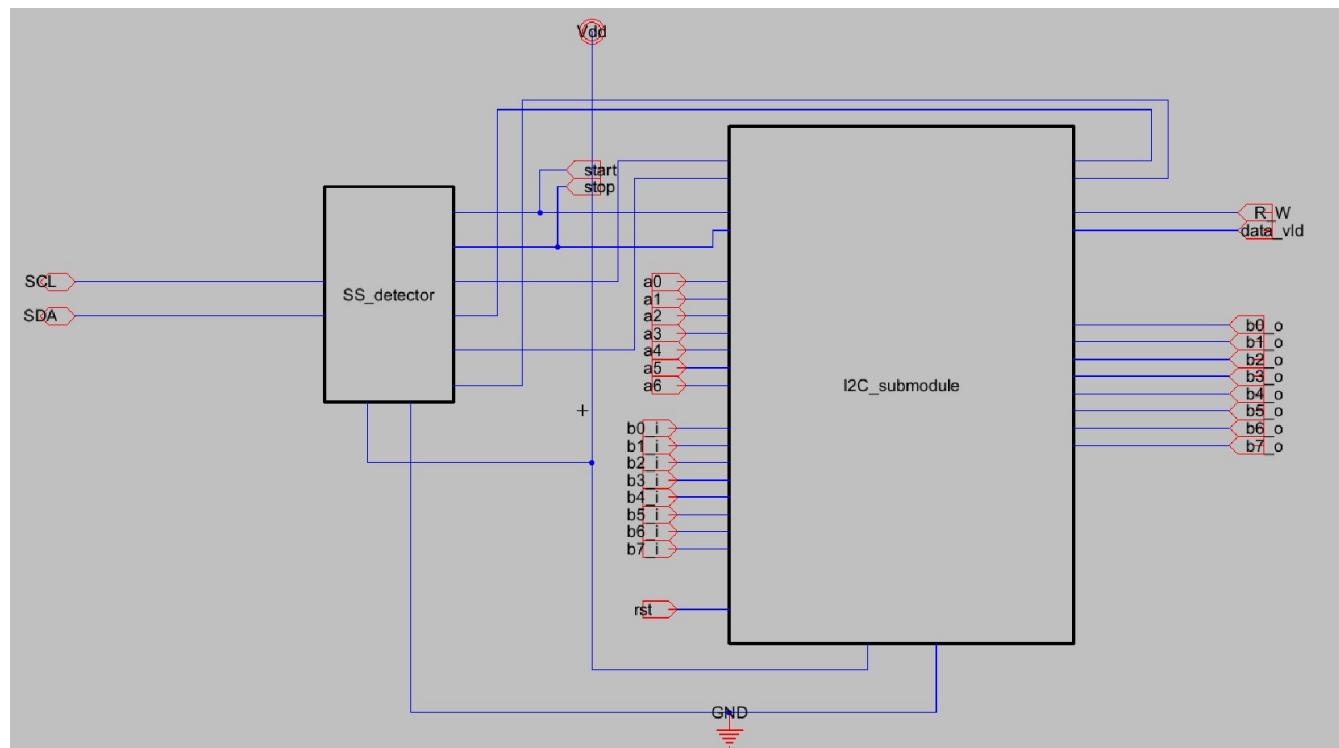


Figura 34: Top Module - Esquemático.

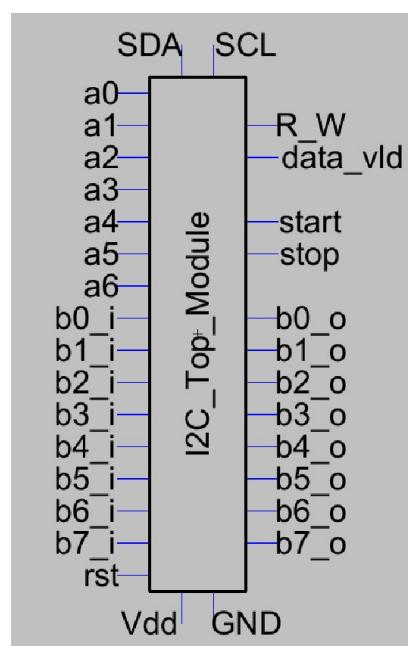


Figura 35: Top Module - Símbolo.

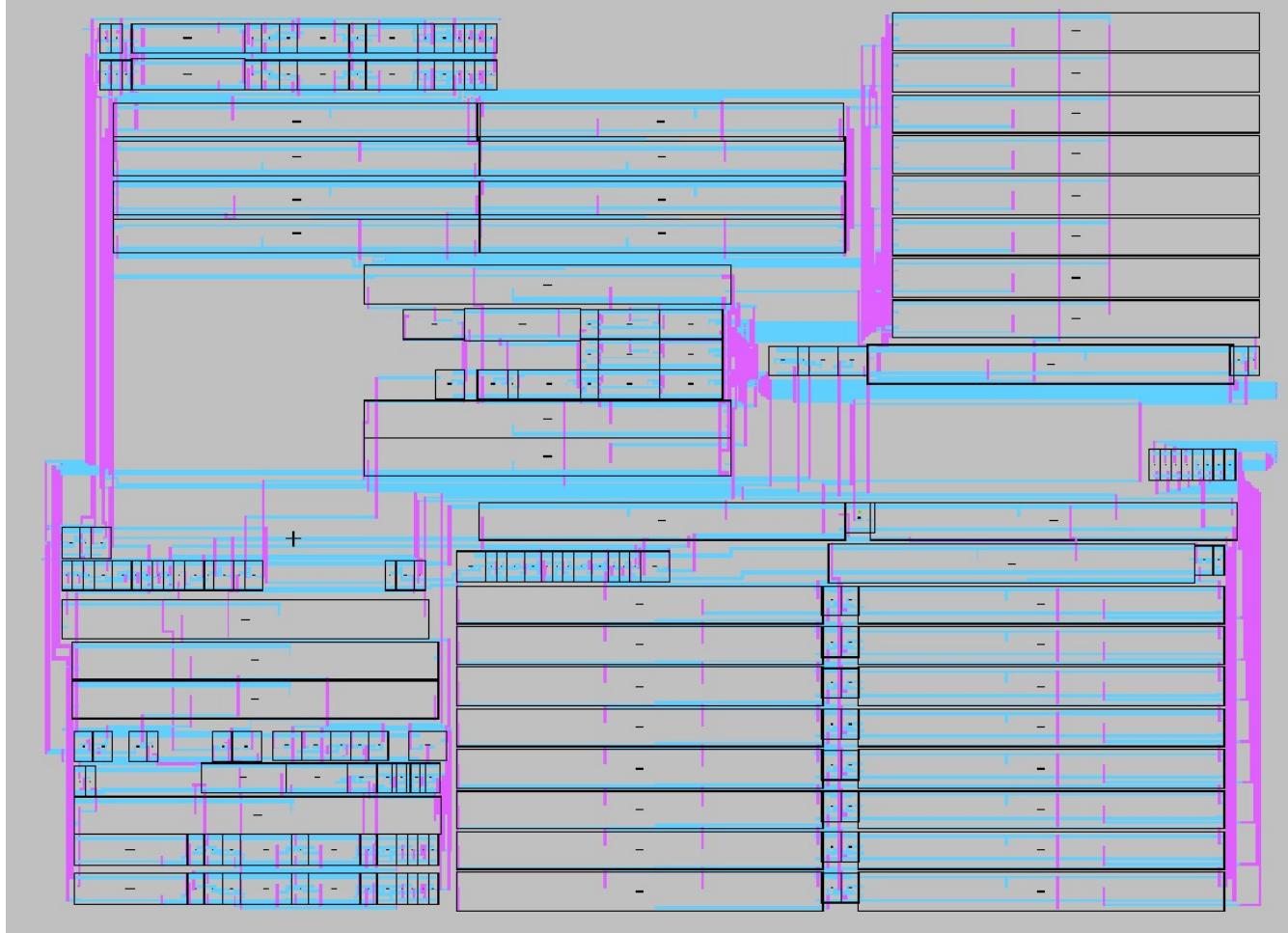


Figura 36: Top Module - Layout.

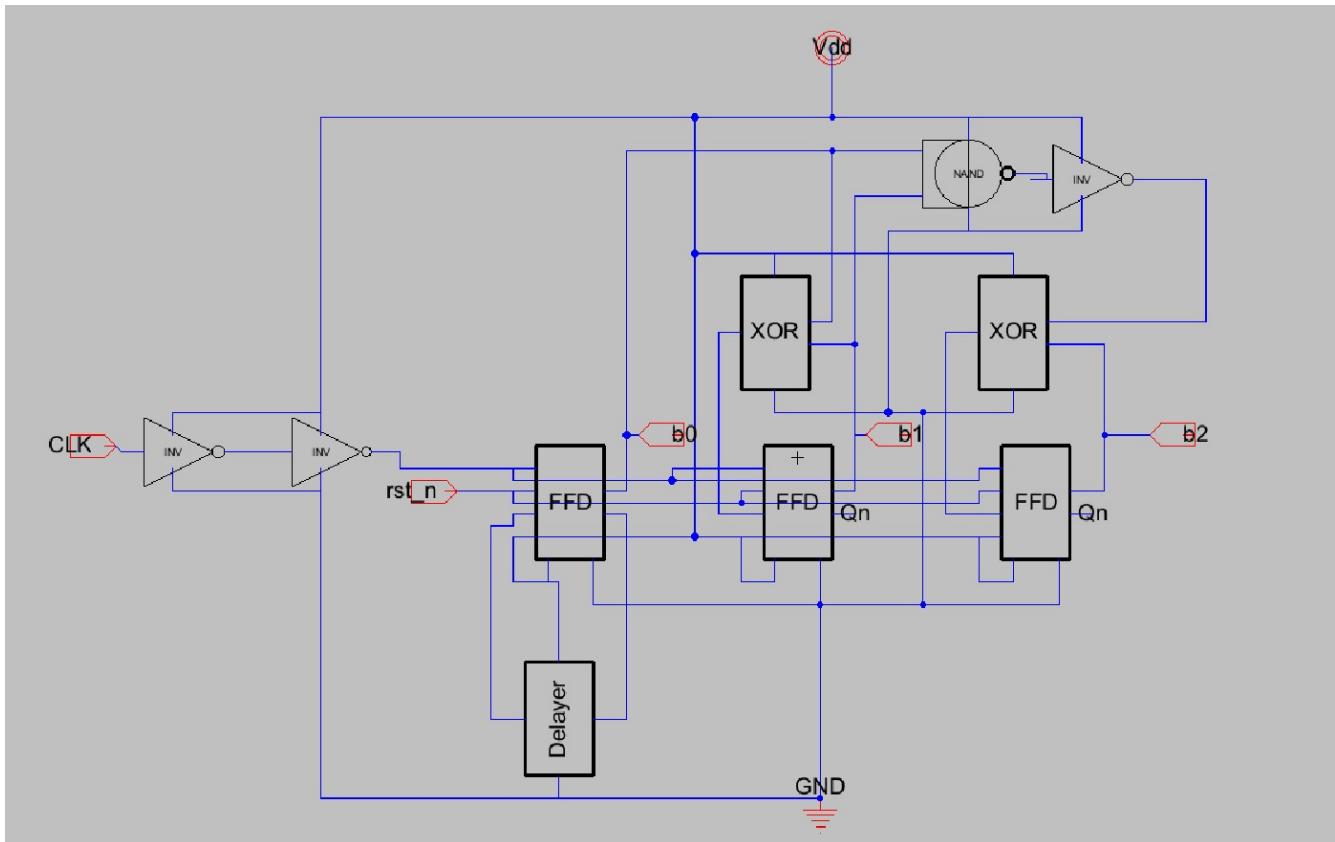


Figura 37: Counter 3b - Esquemático.

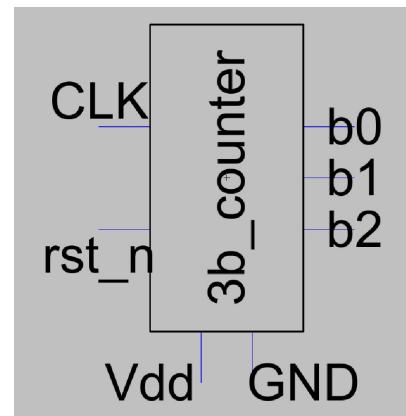


Figura 38: Counter 3b - Símbolo.

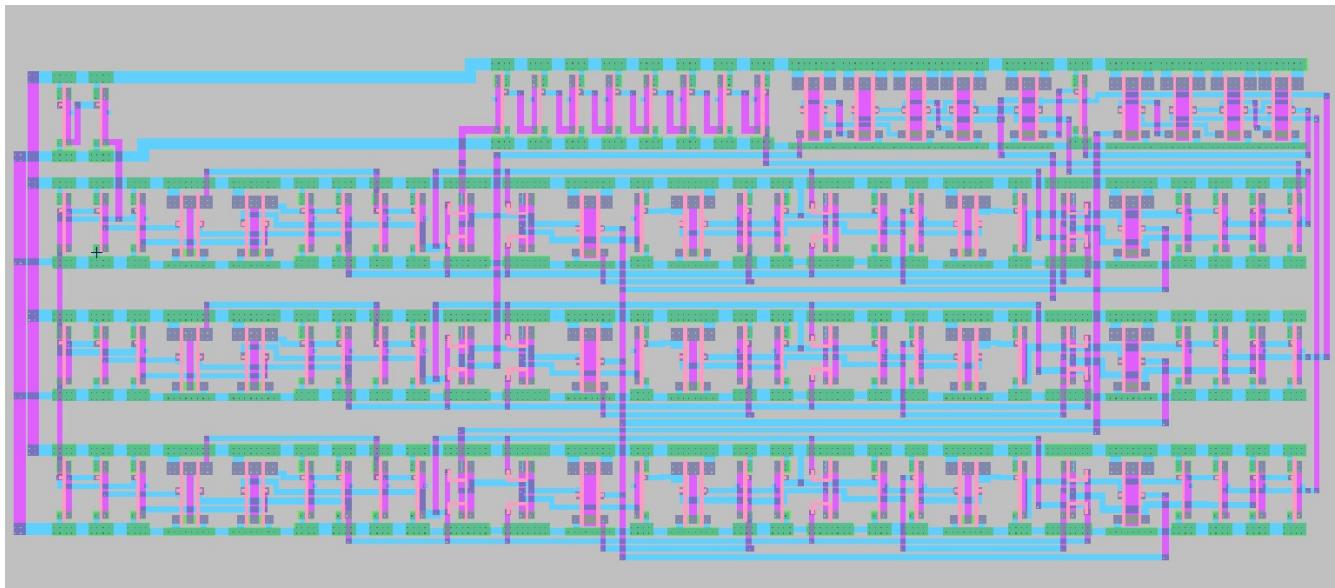


Figura 39: Counter 3b - Layout.

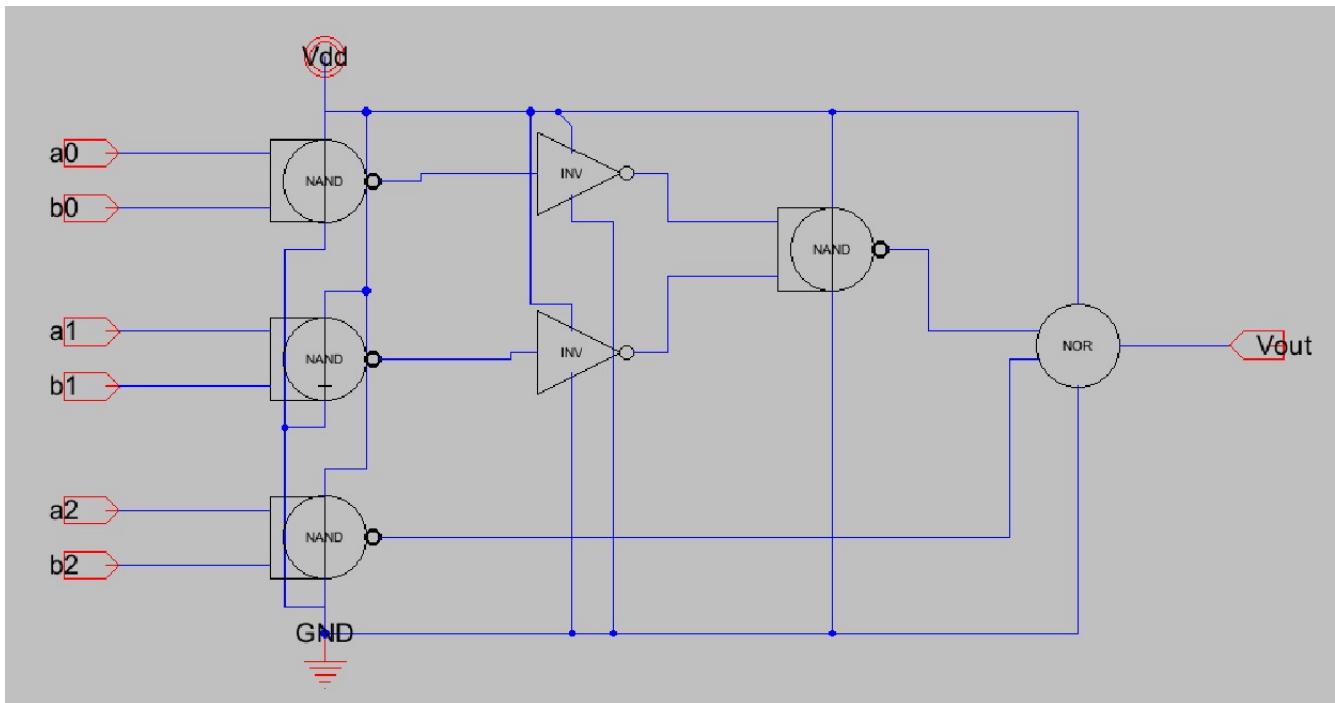


Figura 40: AND 6b - Esquemático.

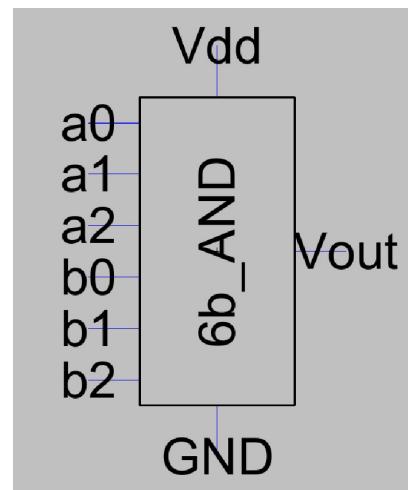


Figura 41: AND 6b - Símbolo.

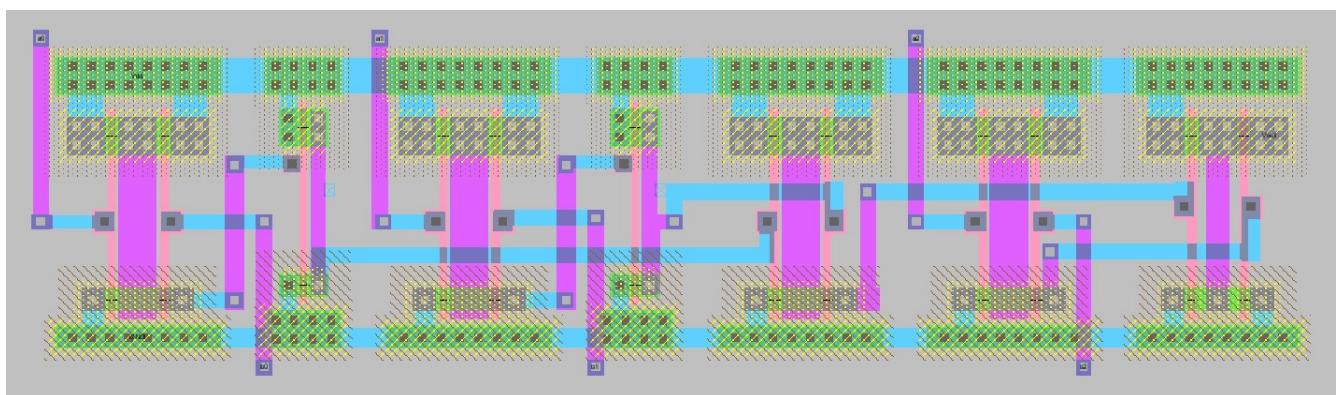


Figura 42: AND 6b - Layout.

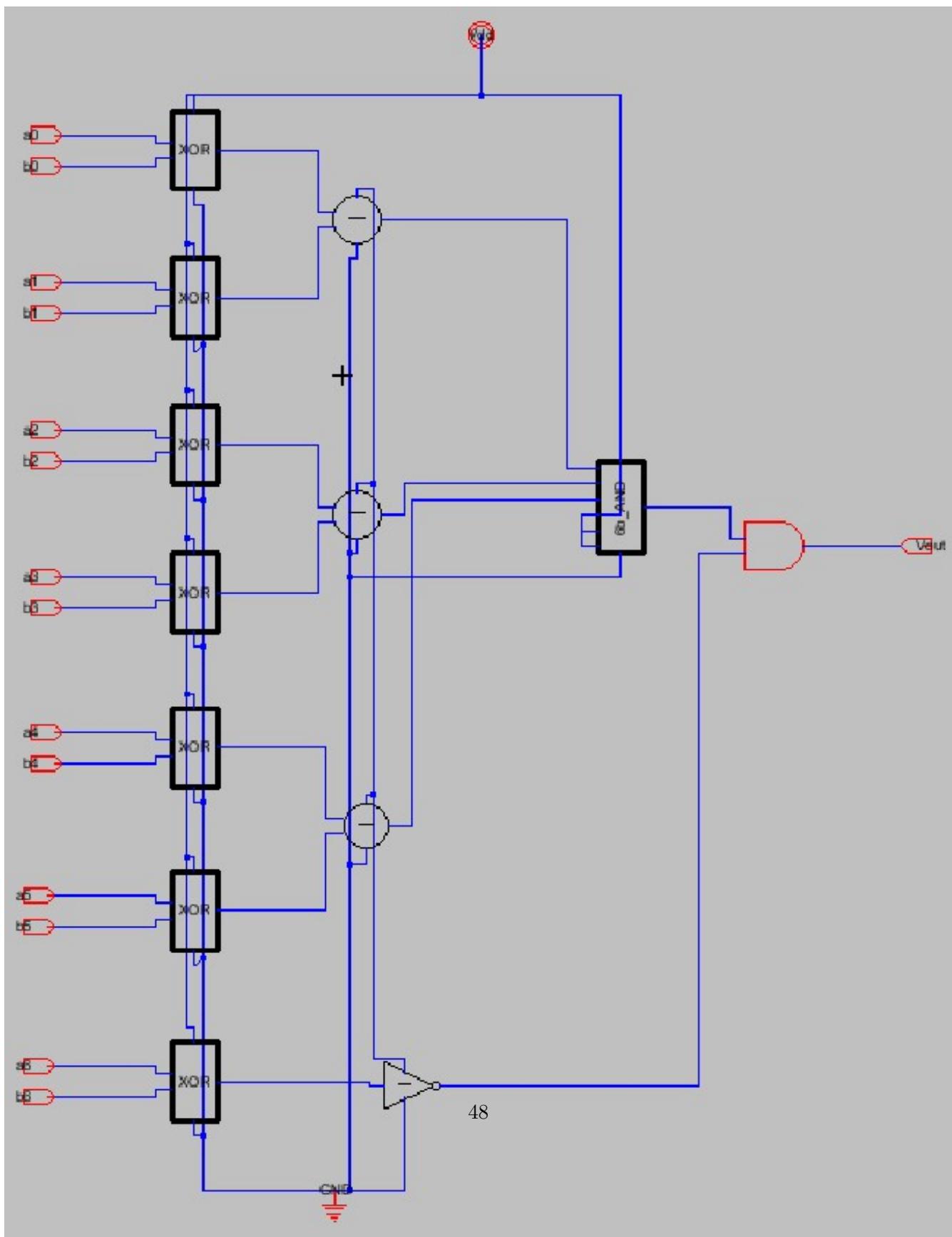


Figura 43: EQUAL 14b - Esquemático.

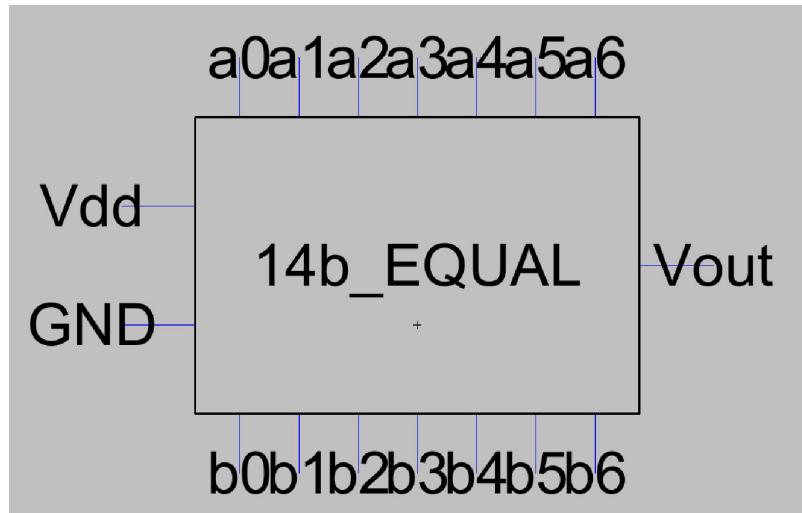


Figura 44: EQUAL 14b - Símbolo.

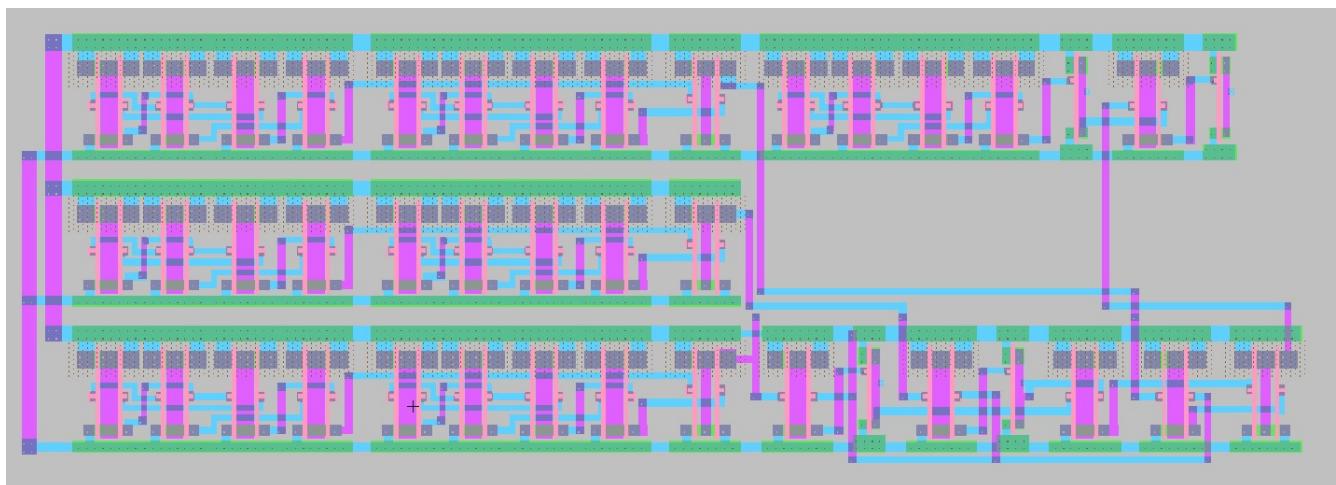


Figura 45: EQUAL 14b - Layout.

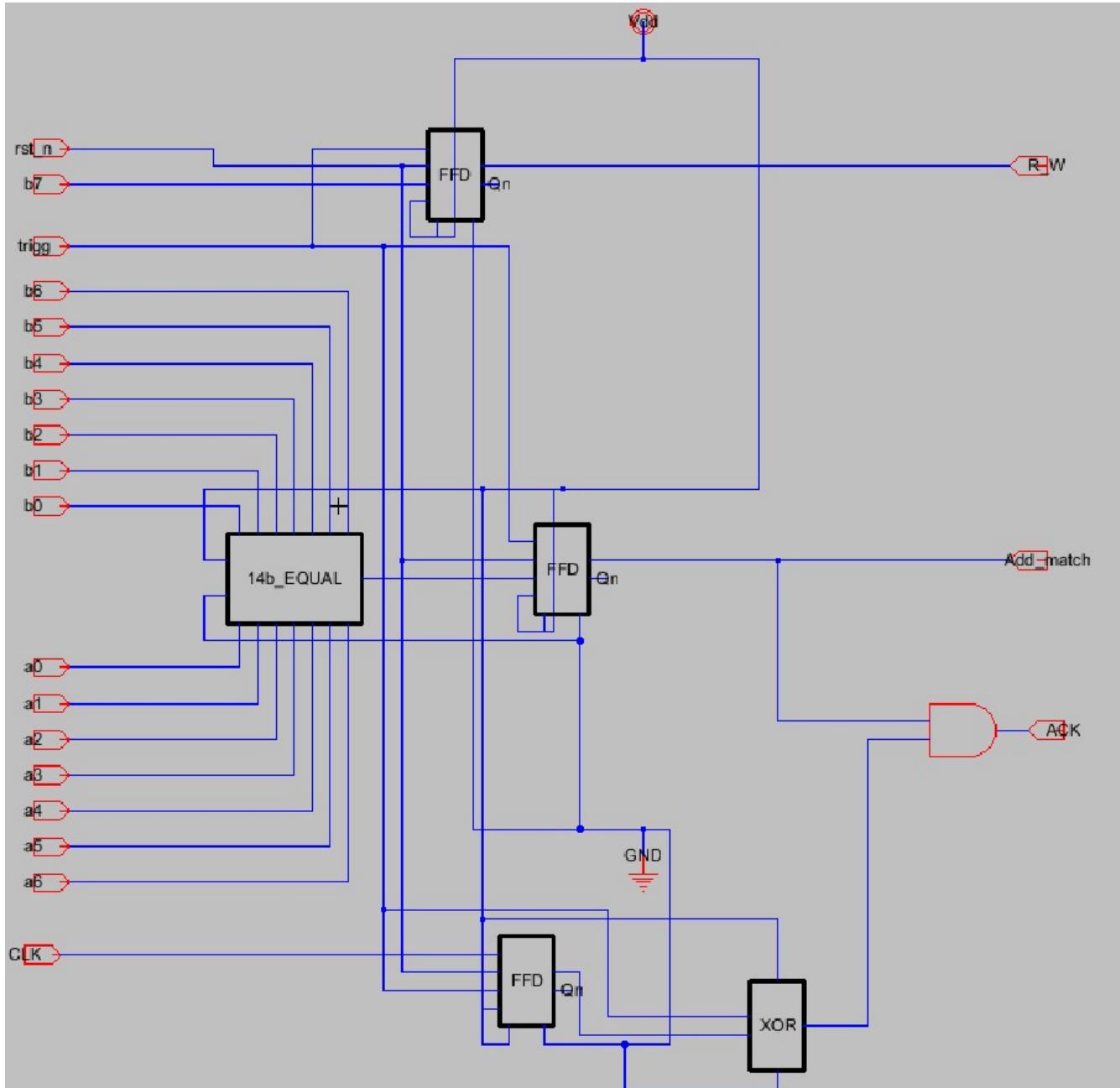


Figura 46: Add Detect Module - Esquemático.

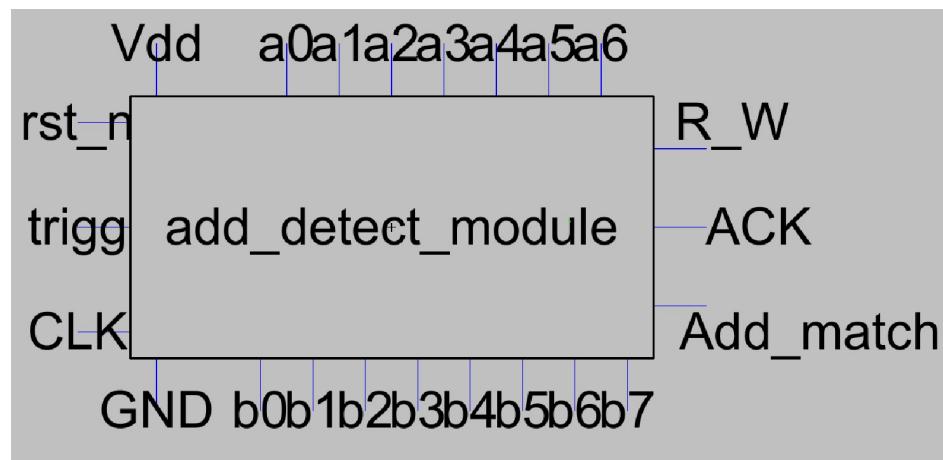


Figura 47: Add Detect Module - Símbolo.

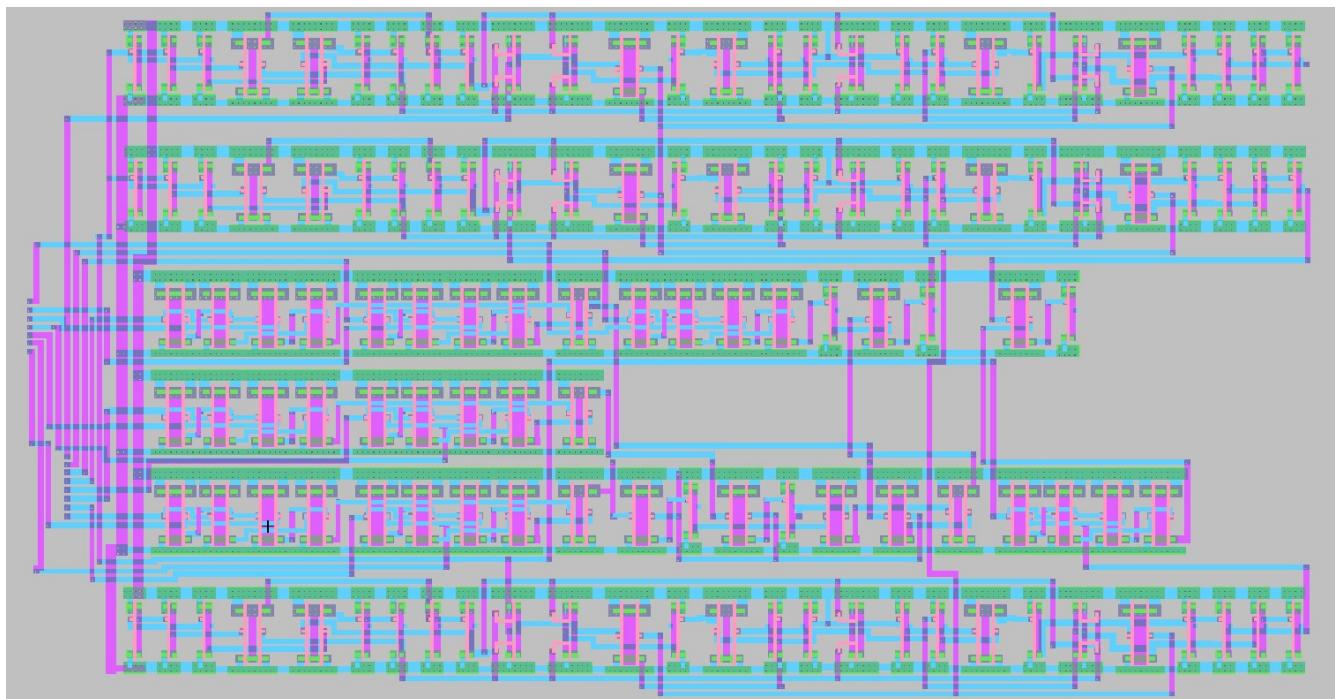


Figura 48: Add Detect Module - Layout.

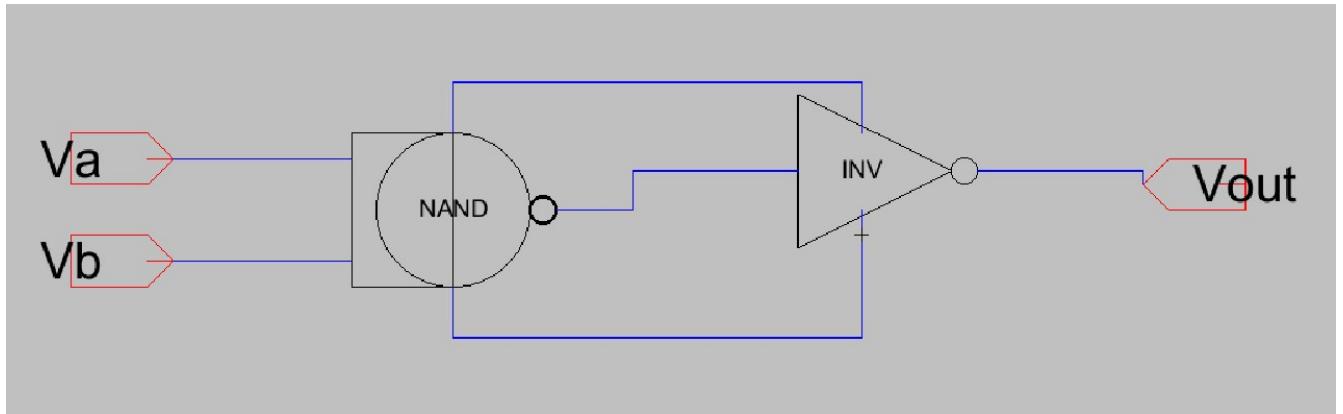


Figura 49: AND - Esquemático.

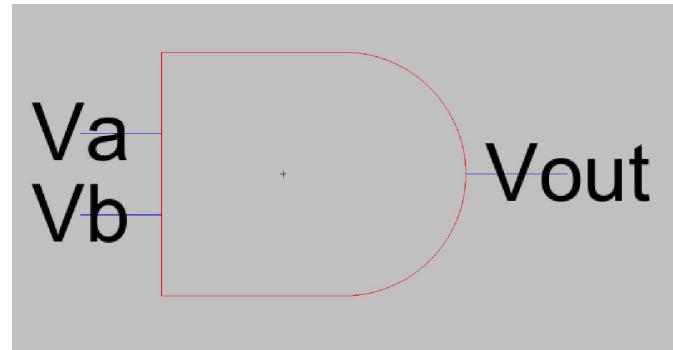


Figura 50: AND - Símbolo.

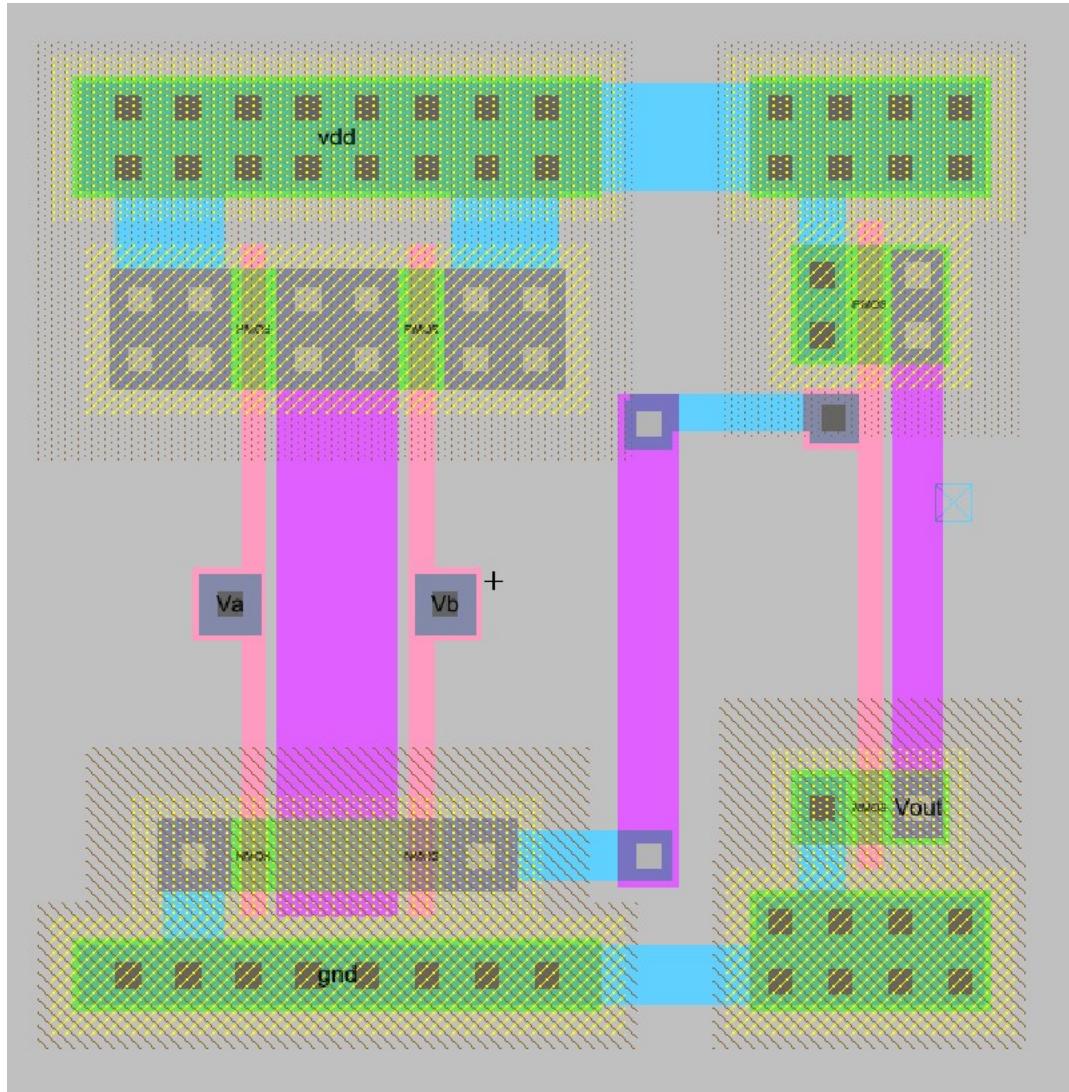


Figura 51: AND - Layout.

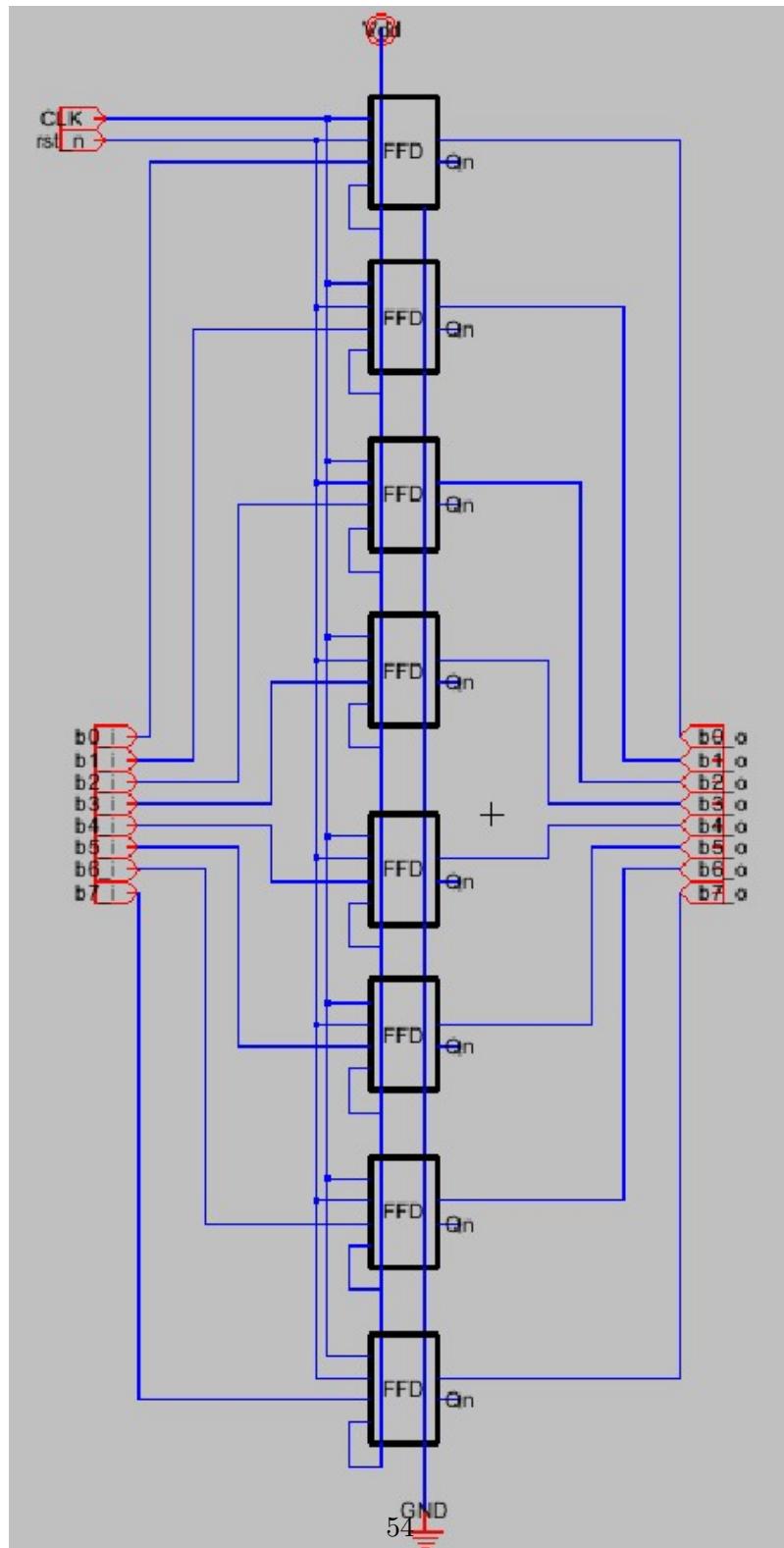


Figura 52: Buffer 8bit - Esquemático.

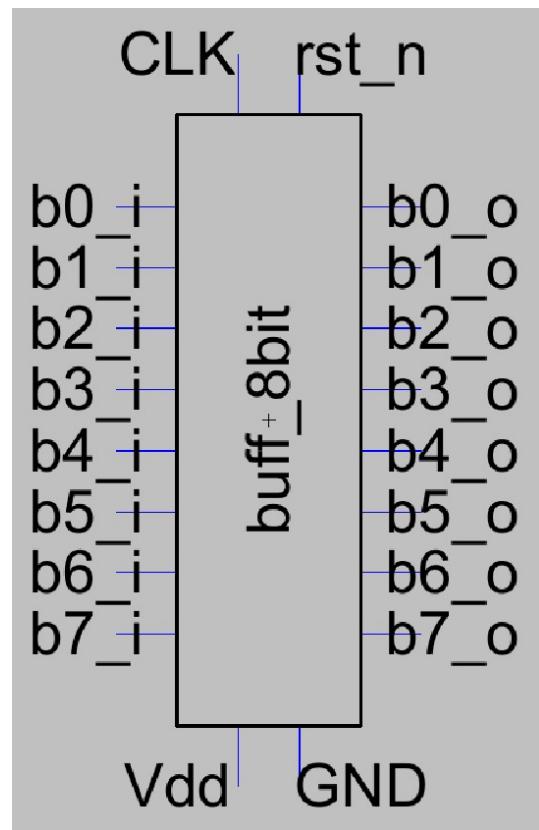


Figura 53: Buffer 8bit - Símbolo.

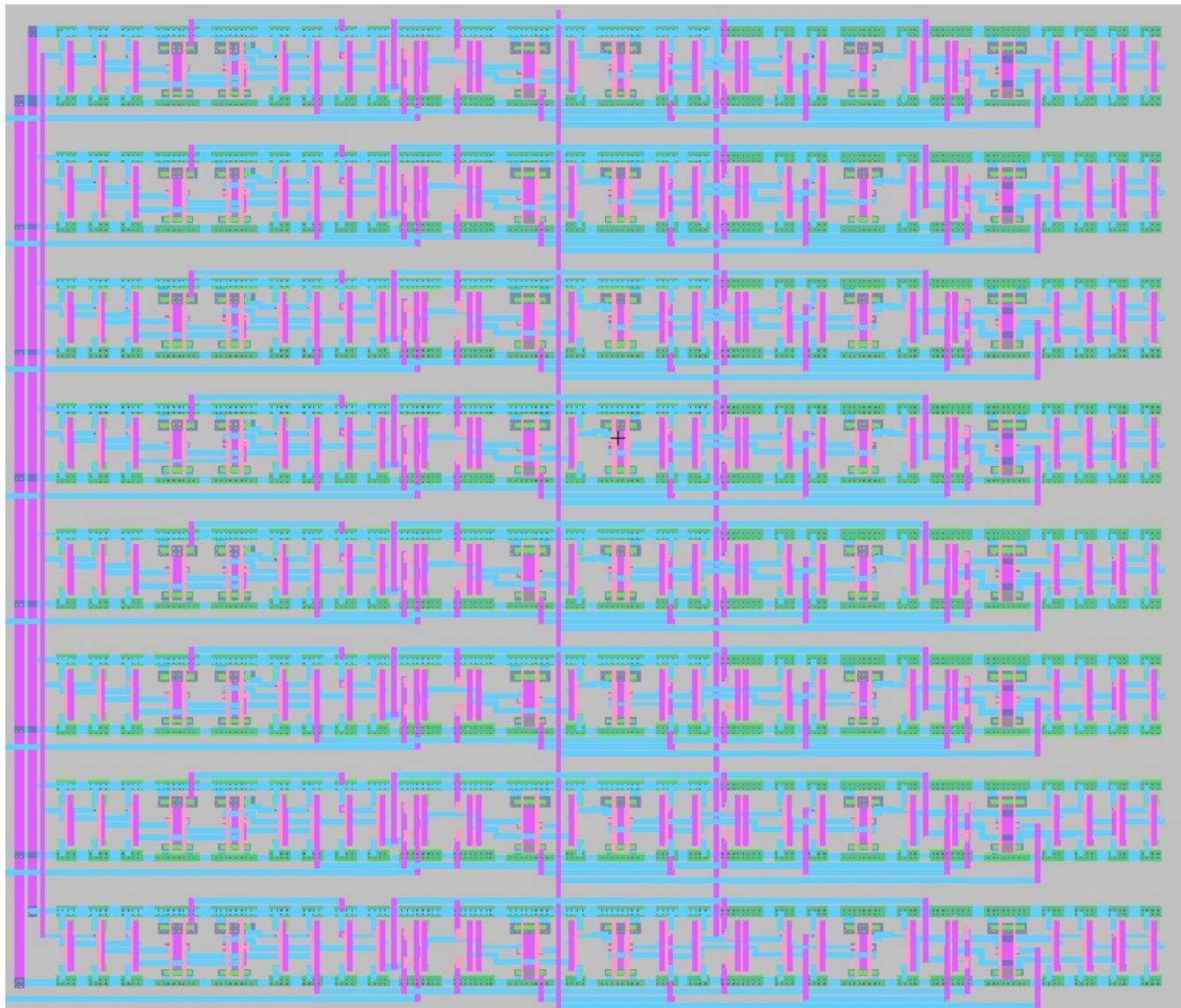


Figura 54: Buffer 8bit - Layout.

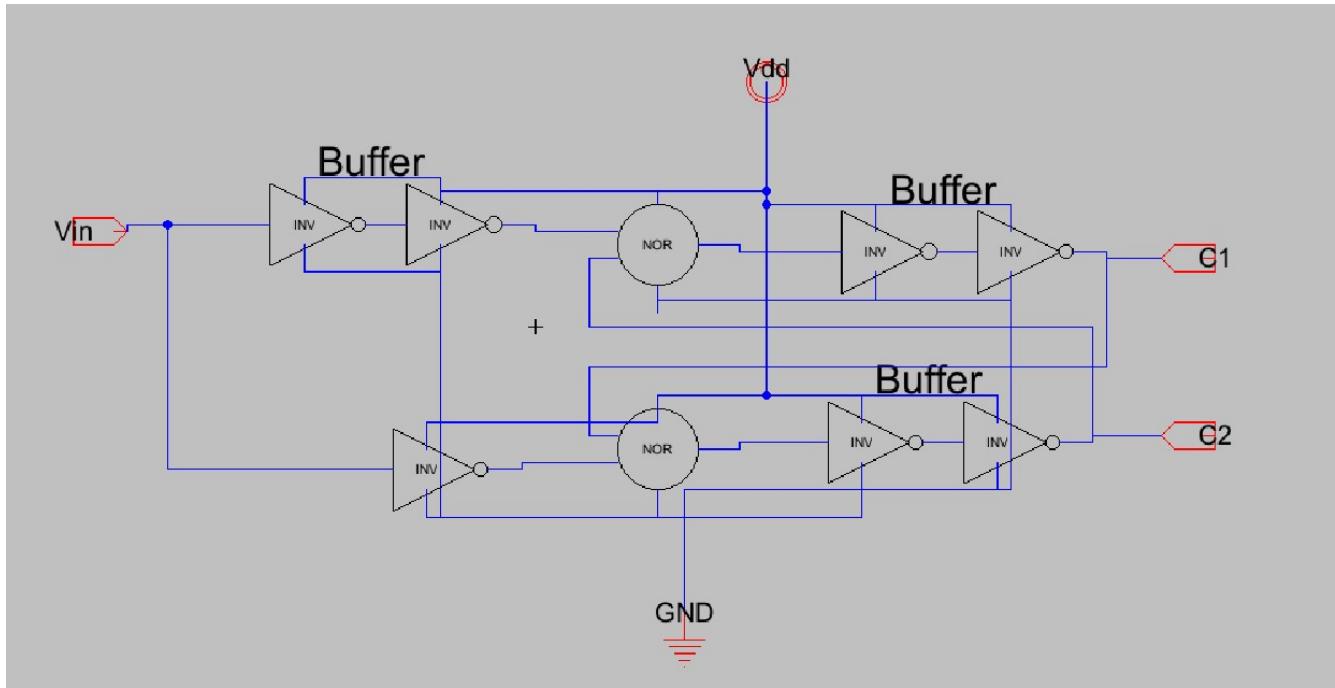


Figura 55: Clock No Slop - Esquemático.

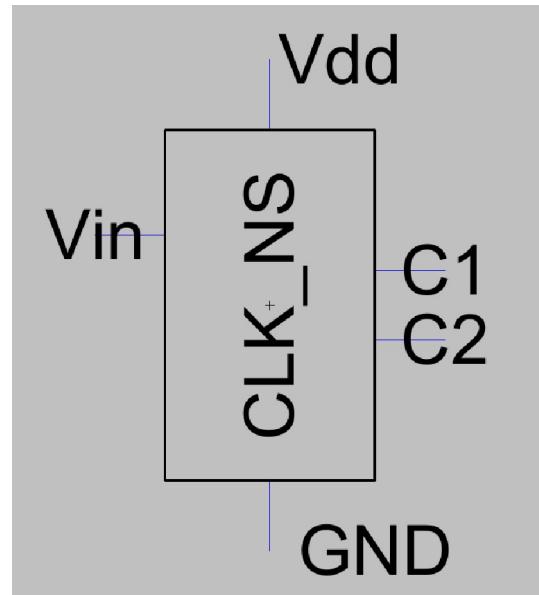


Figura 56: Clock No Slop - Símbolo.

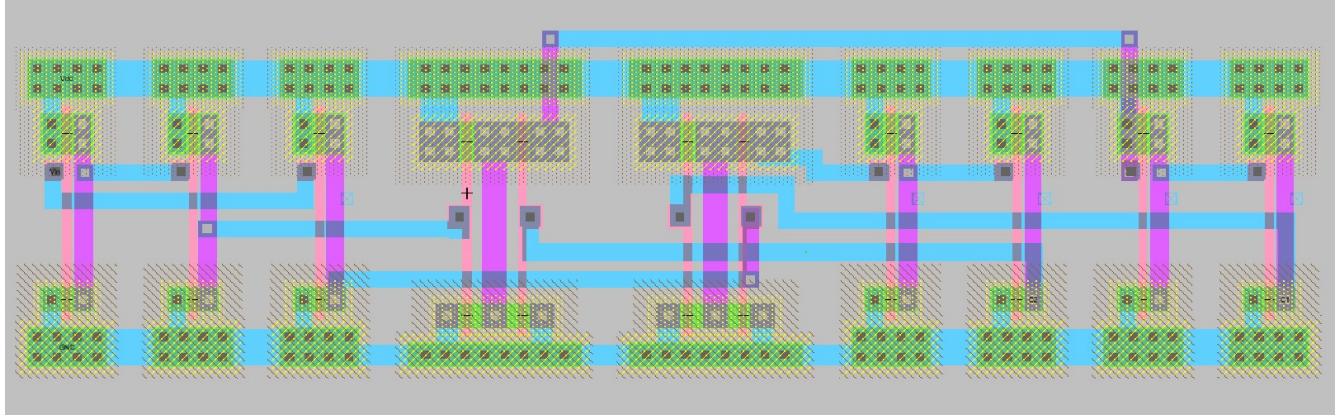


Figura 57: Clock No Slop - Layout.

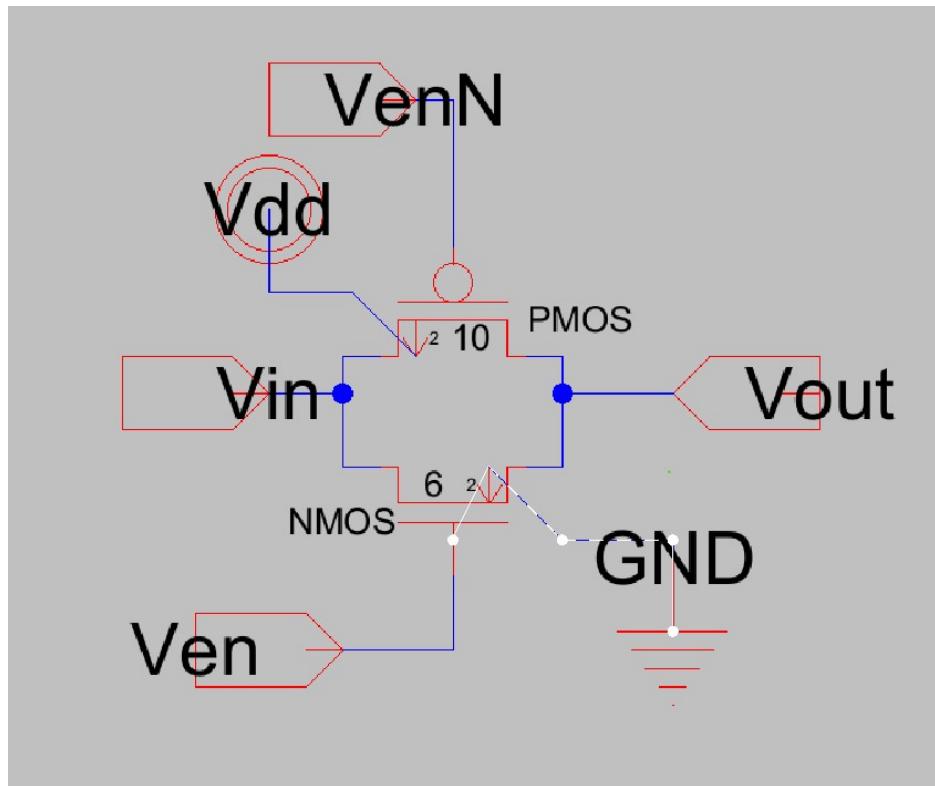


Figura 58: Comp Paso - Esquemático.

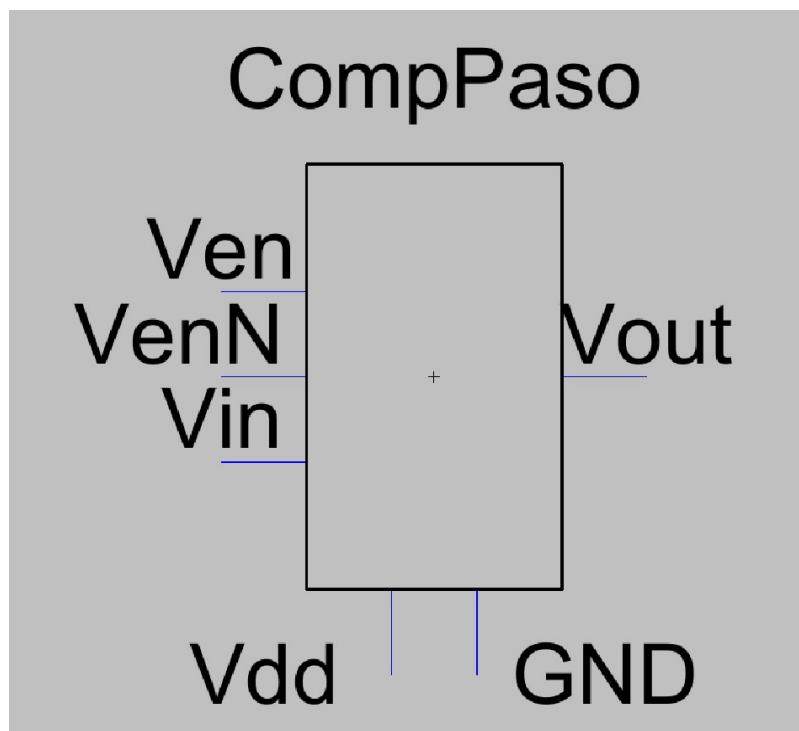


Figura 59: Comp Paso - Símbolo.

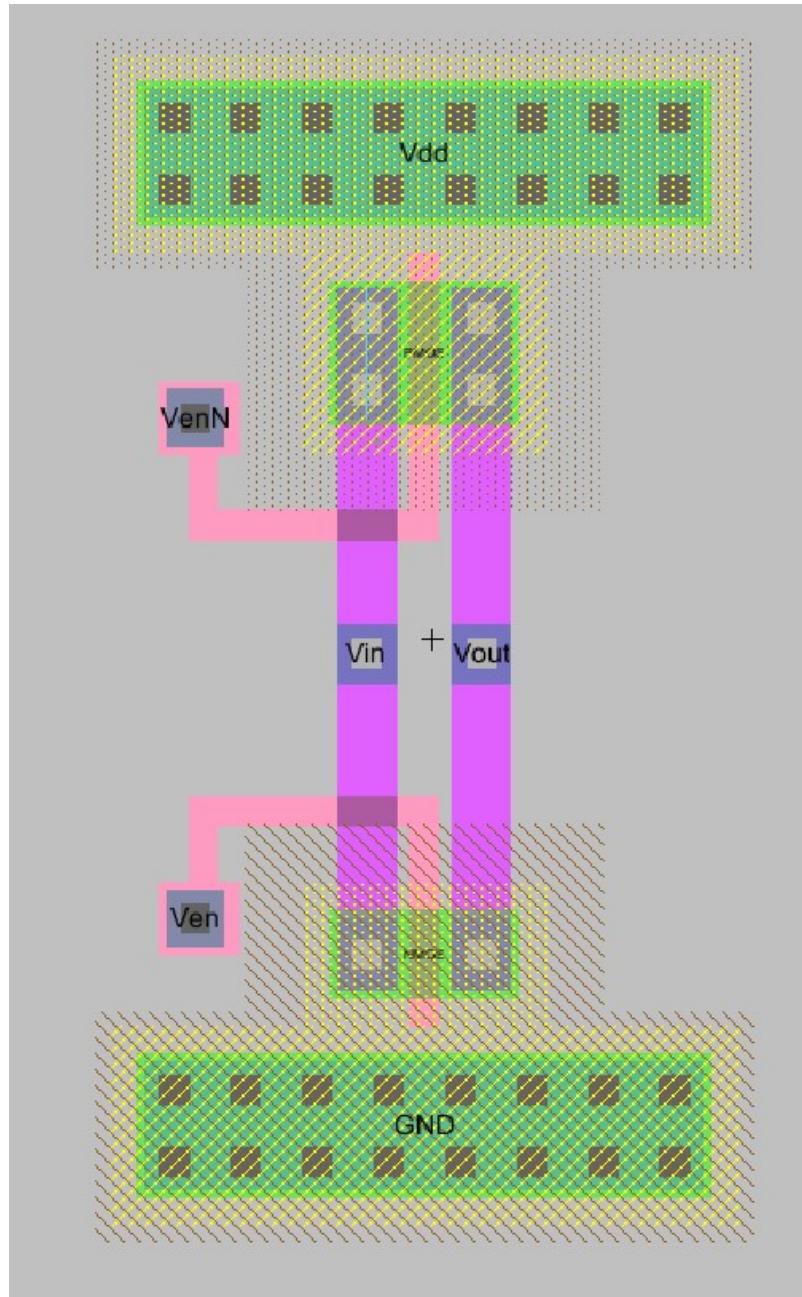


Figura 60: Comp Paso - Layout.

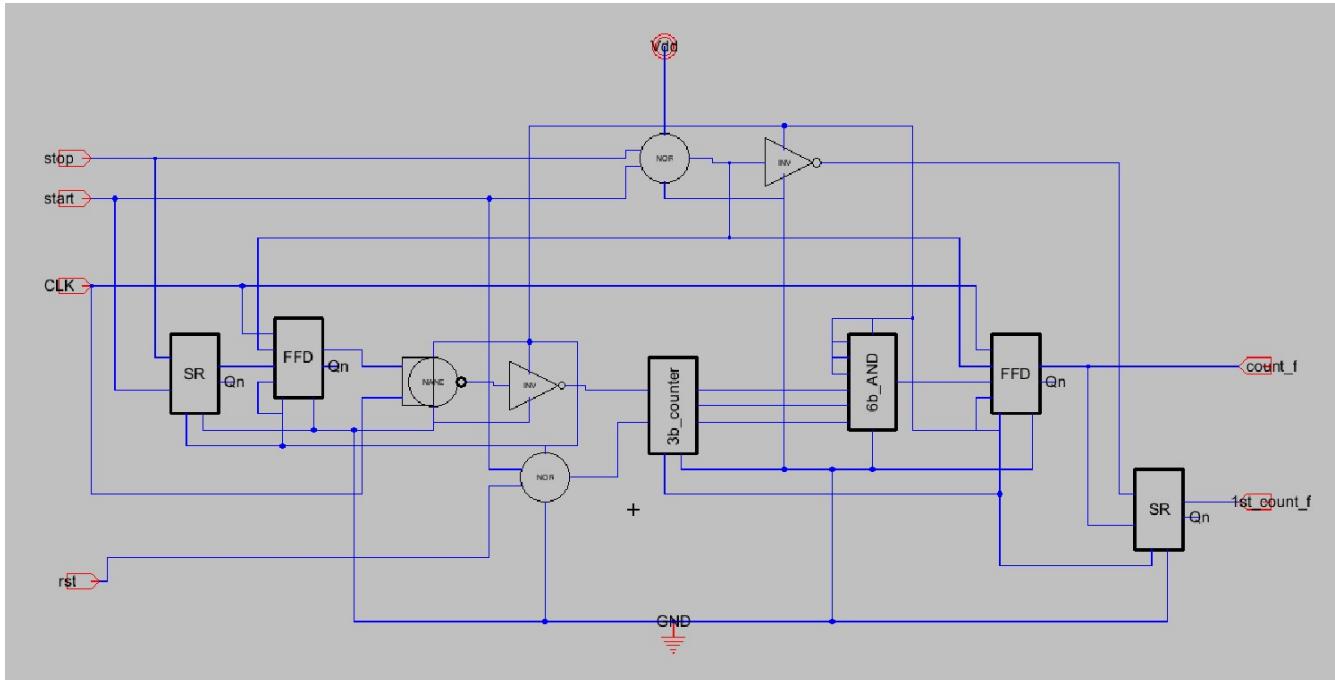


Figura 61: Counter Module - Esquemático.

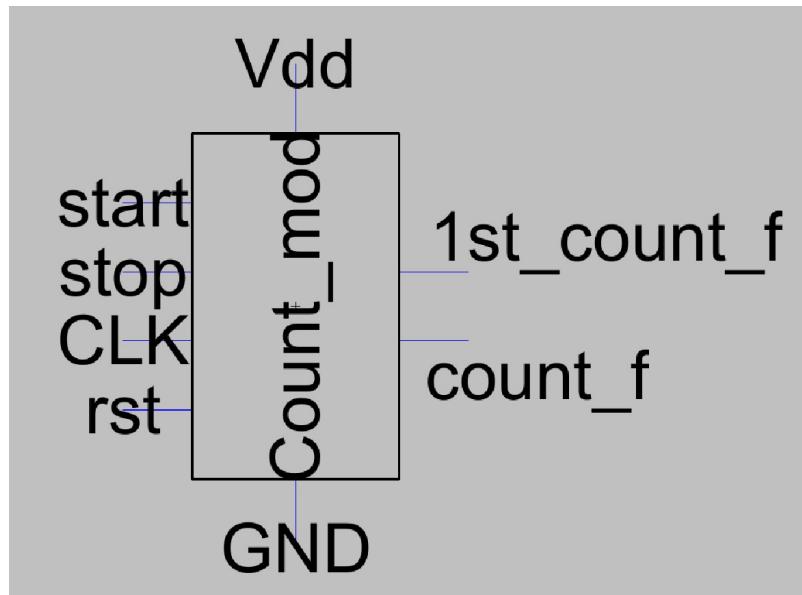


Figura 62: Counter Module - Símbolo.

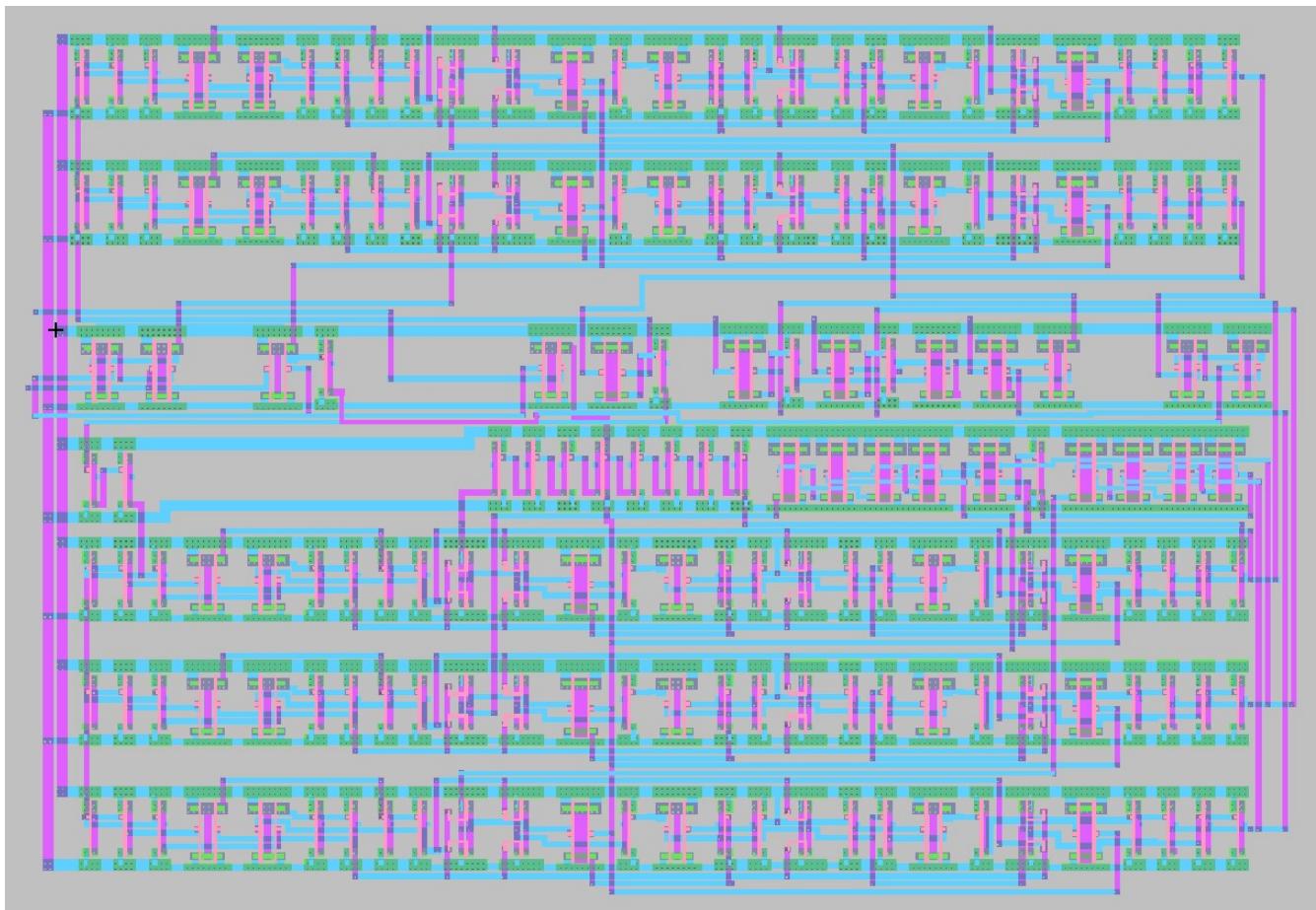


Figura 63: Counter Module - Layout.

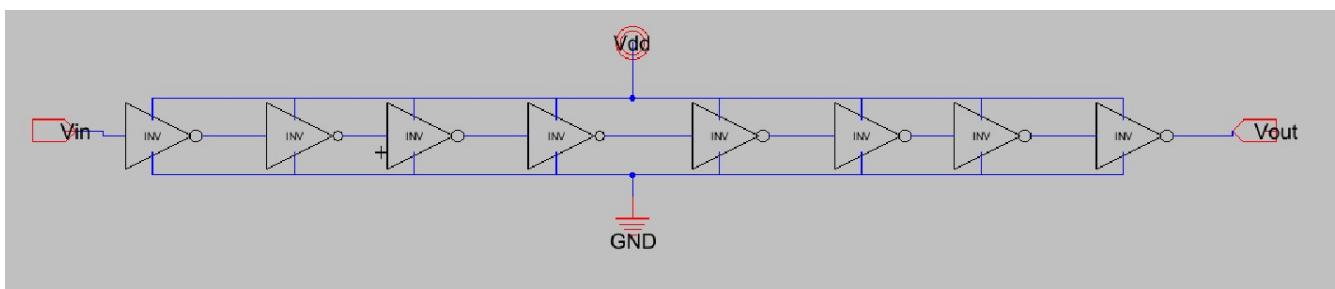


Figura 64: Delayer - Esquemático.

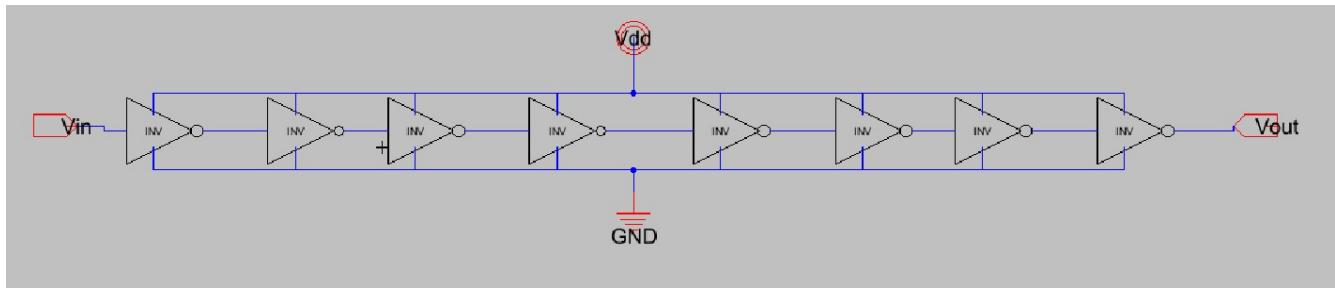


Figura 65: Delayer - Símbolo.

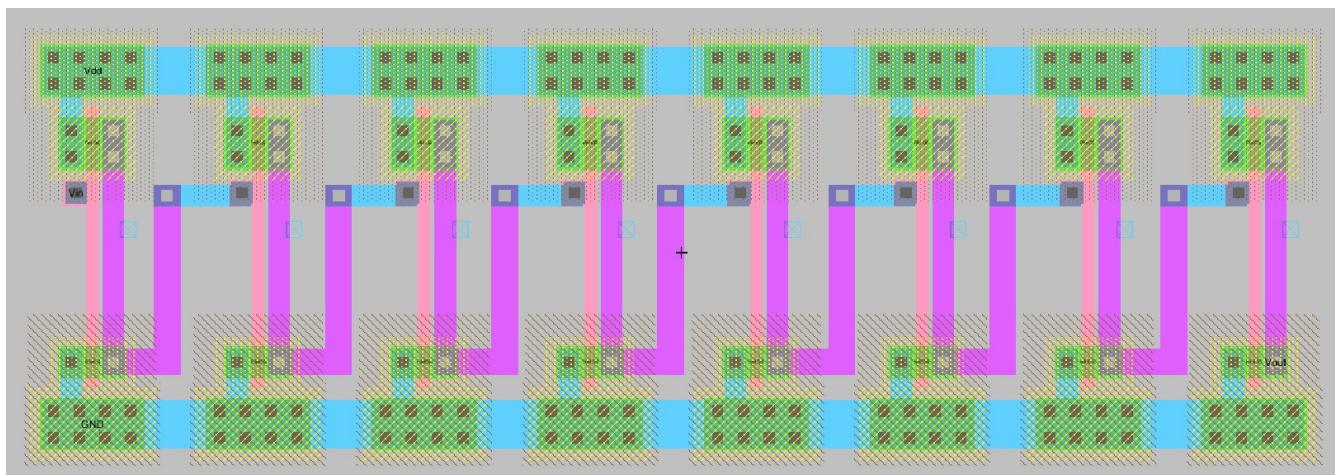


Figura 66: Delayer - Layout.

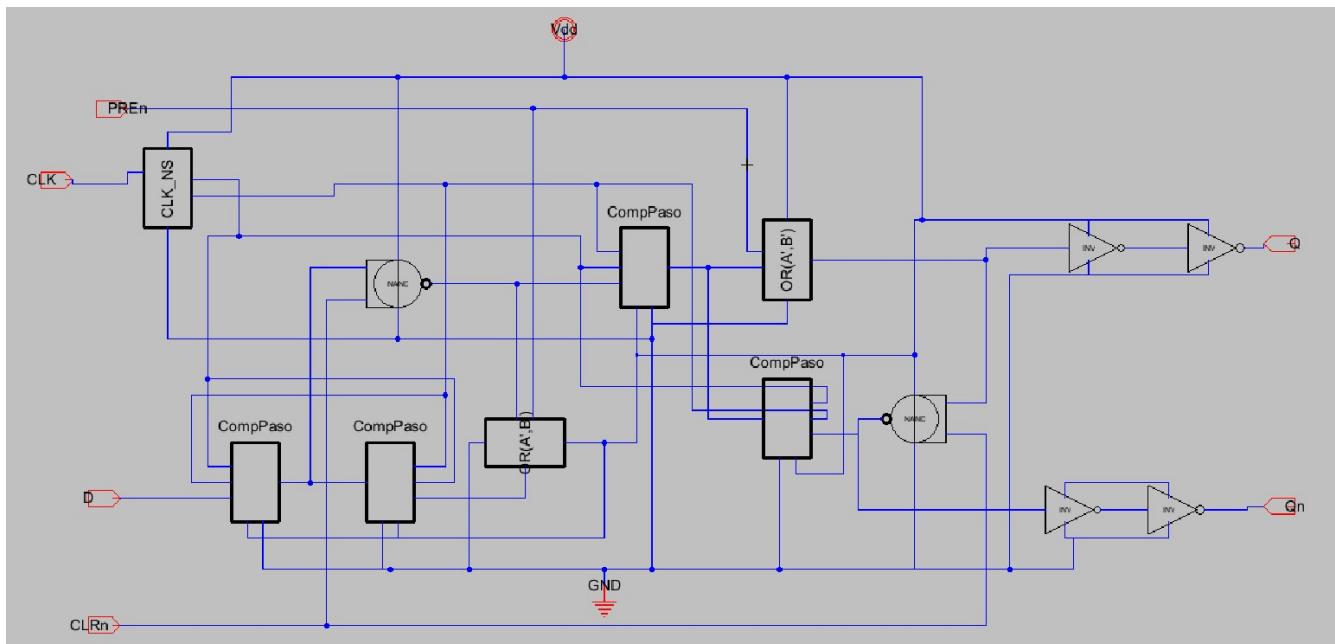


Figura 67: FFD Clrn - Esquemático.

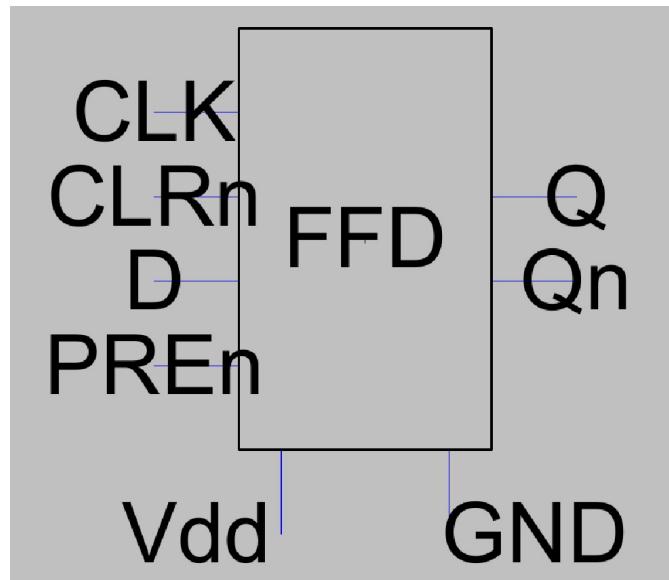


Figura 68: FFD Clrn - Símbolo.

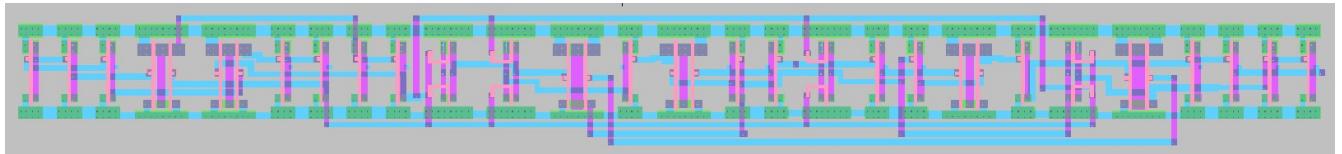


Figura 69: FFD Clrn - Layout.

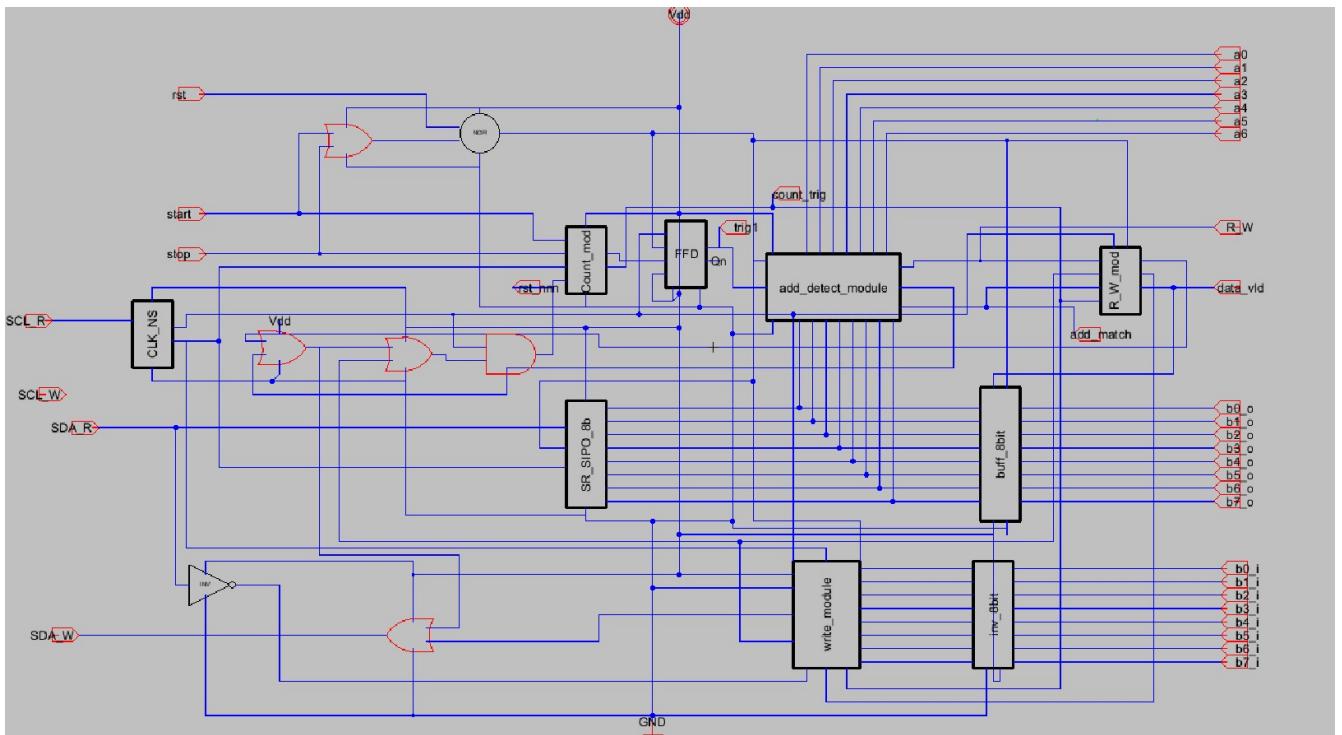


Figura 70: I2C Submodule - Esquemático.

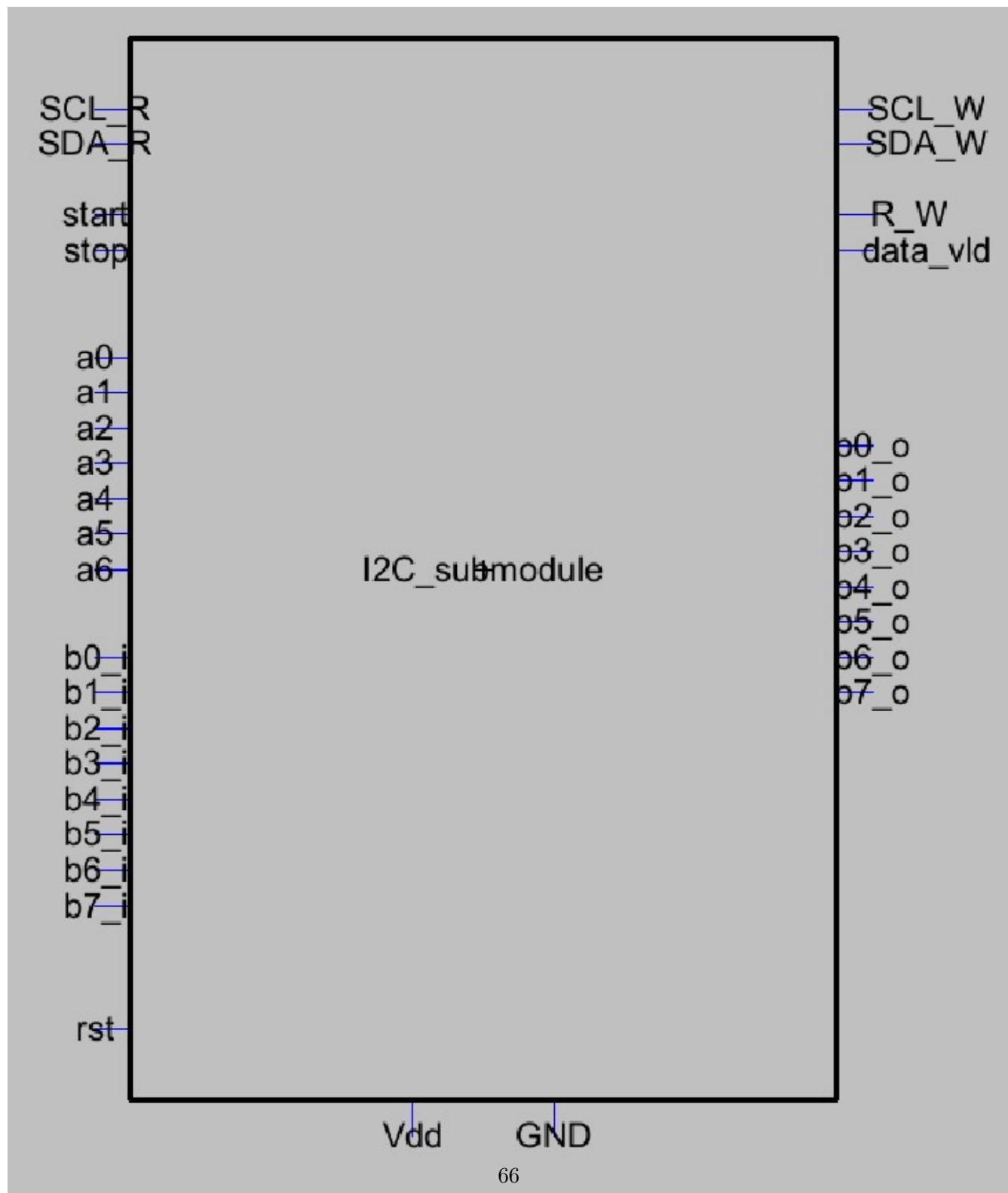


Figura 71: I2C Submodule - Símbolo.

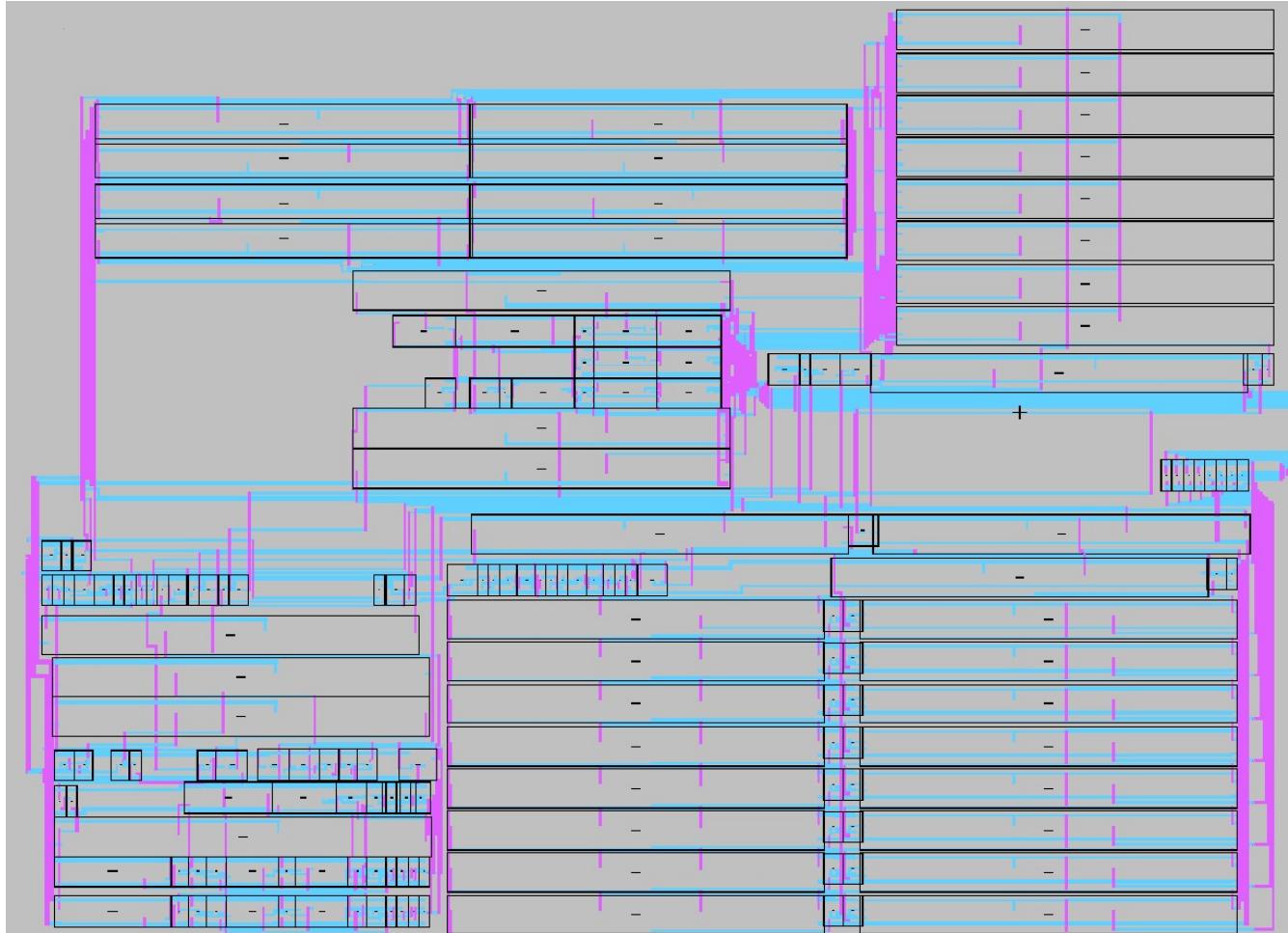


Figura 72: I2C Submodule - Layout.

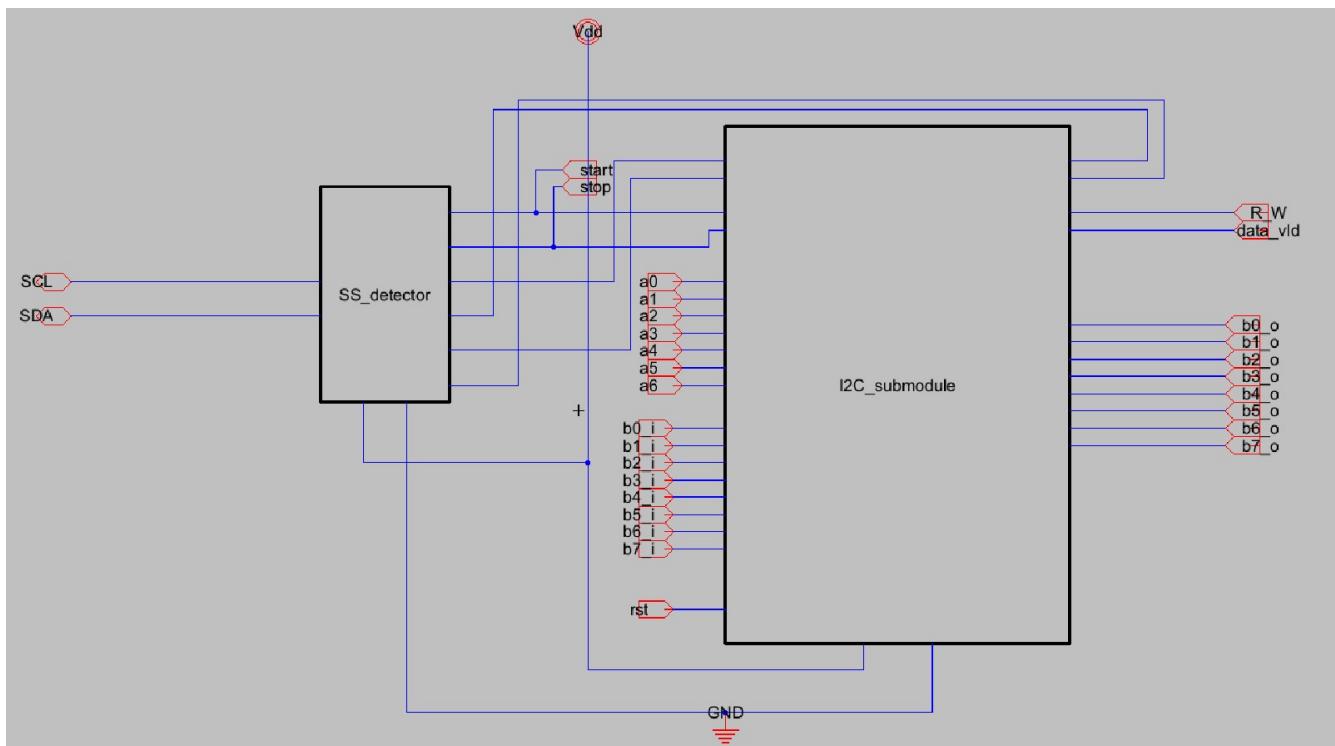


Figura 73: I2C Top Module - Esquemático.

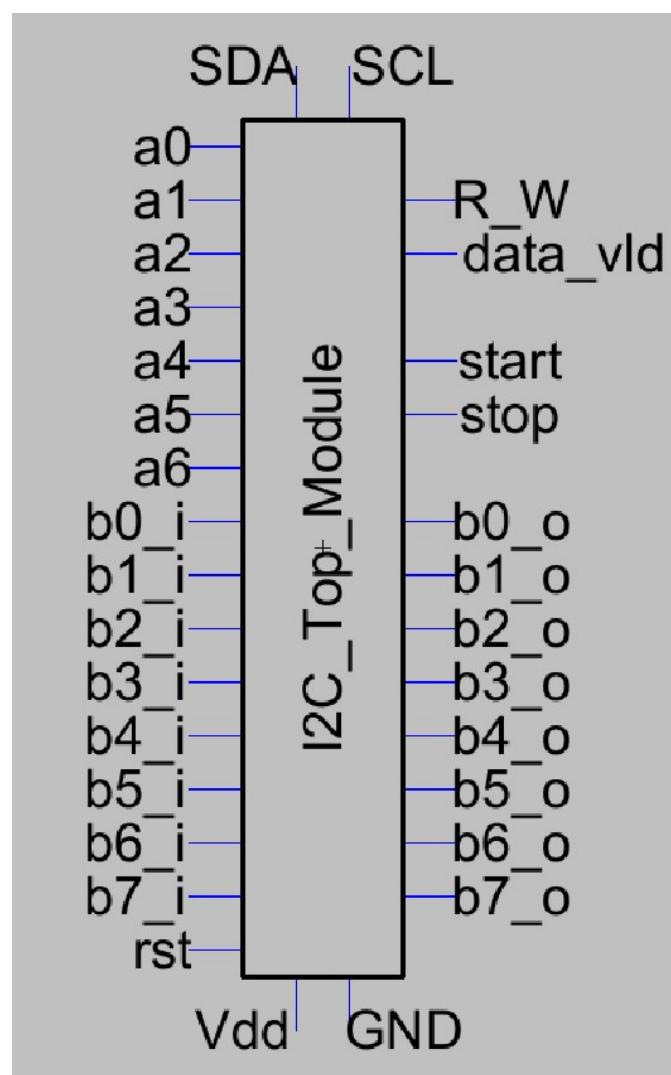


Figura 74: I2C Top Module - Símbolo.

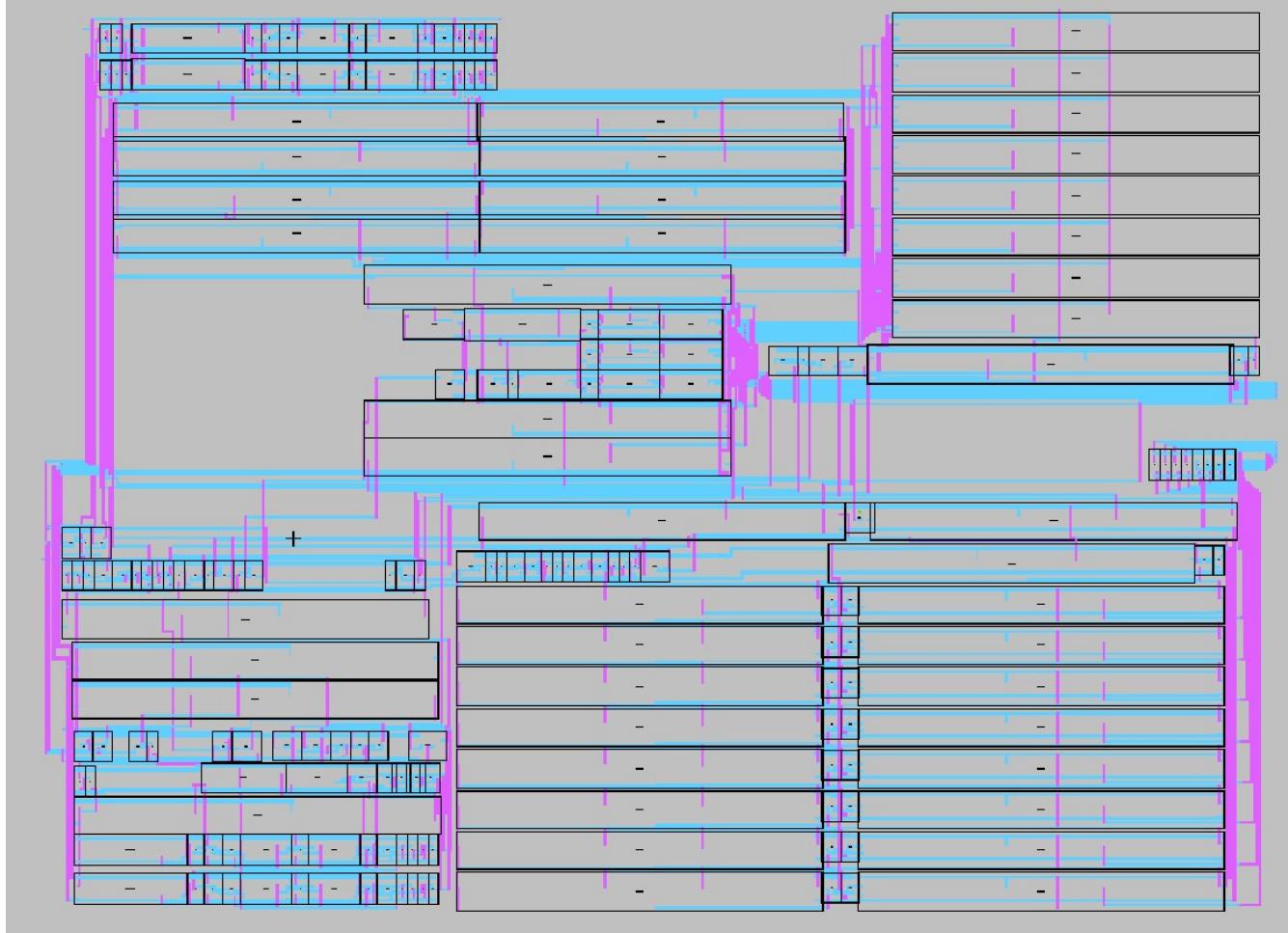


Figura 75: I2C Top Module - Layout.

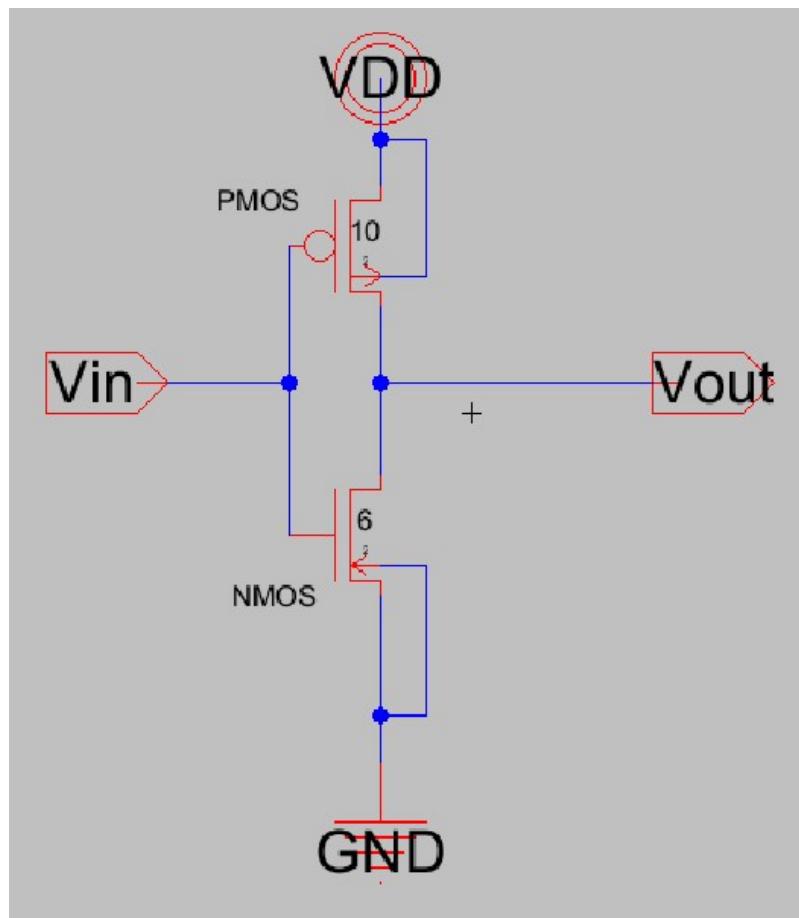


Figura 76: INV - Esquemático.

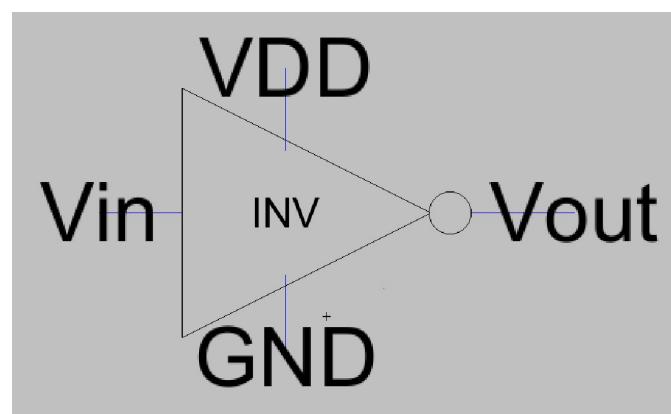


Figura 77: INV - Símbolo.

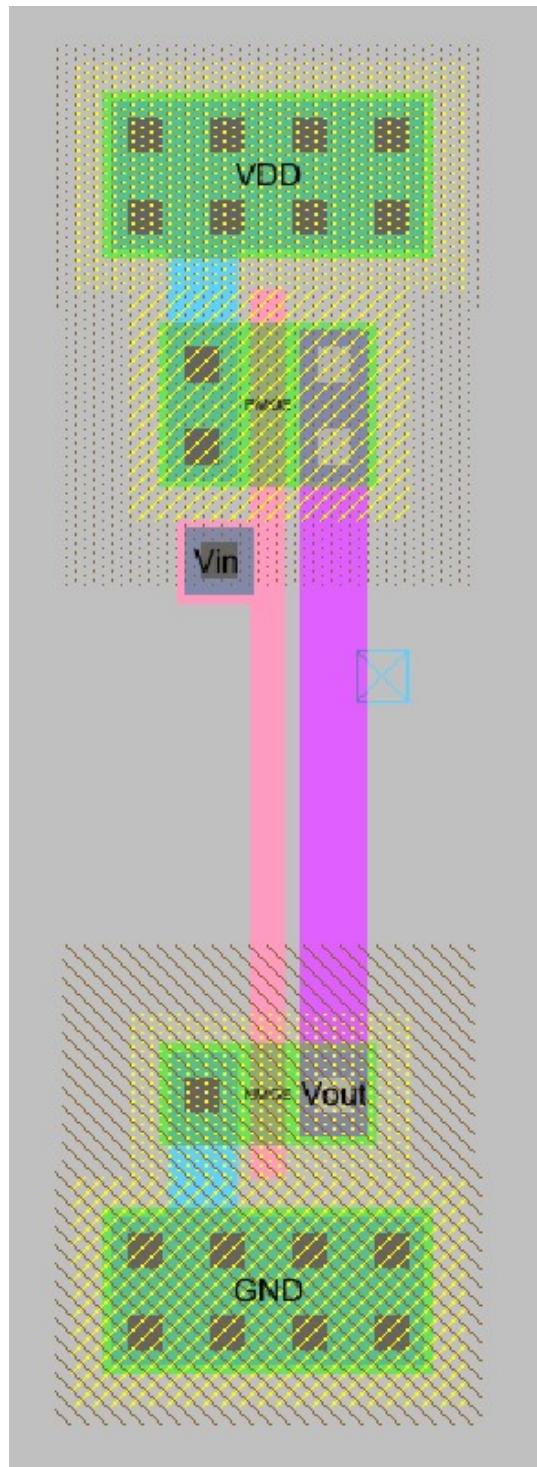


Figura 78: INV - Layout.

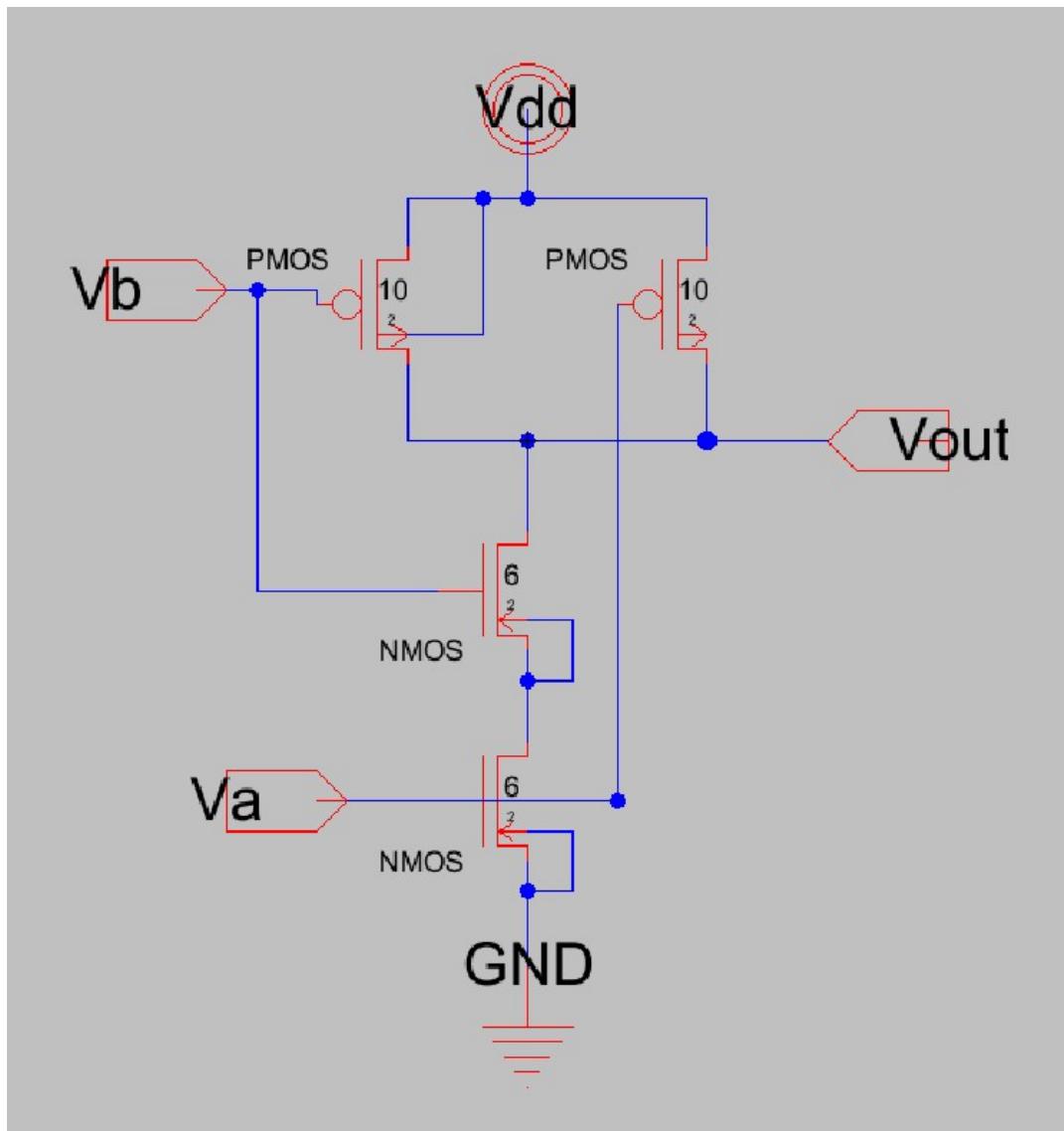


Figura 79: NAND - Esquemático.

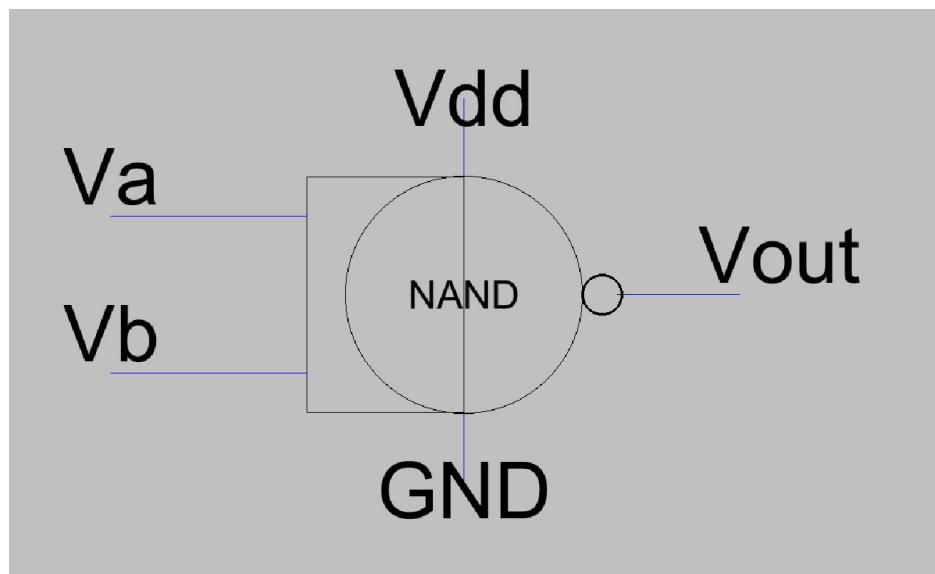


Figura 80: NAND - Símbolo.

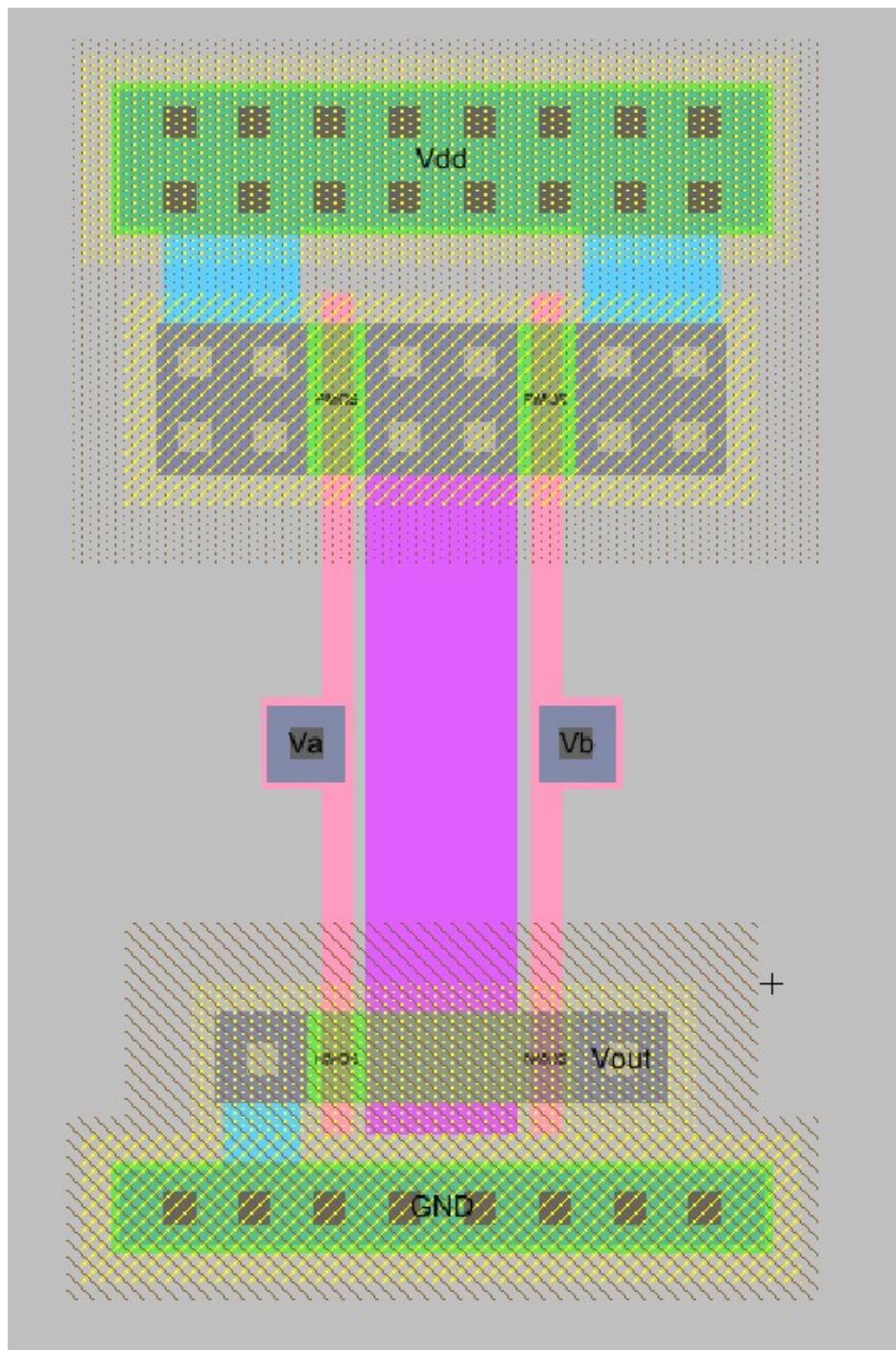


Figura 81: NAND - Layout.

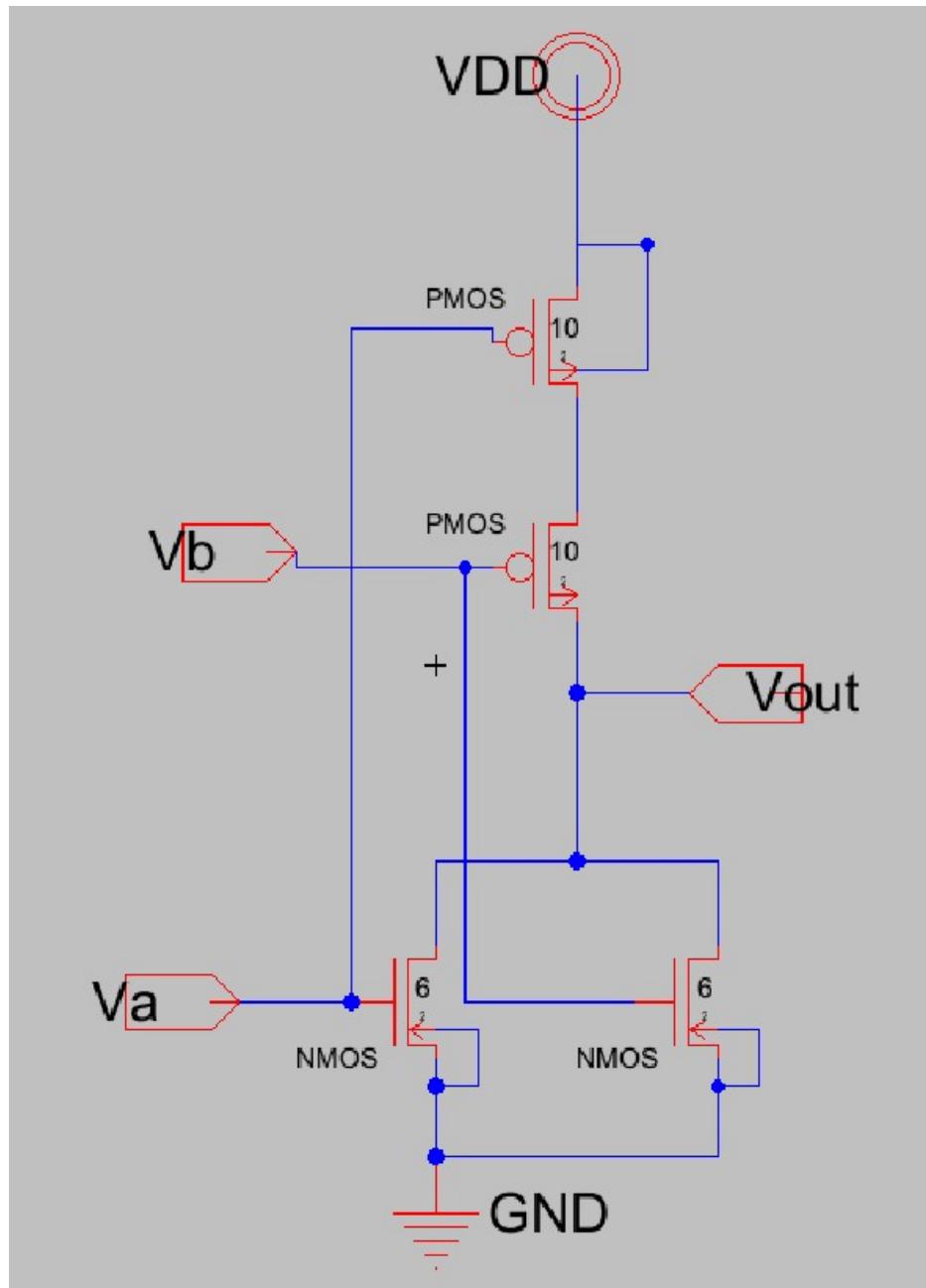


Figura 82: NOR - Esquemático.

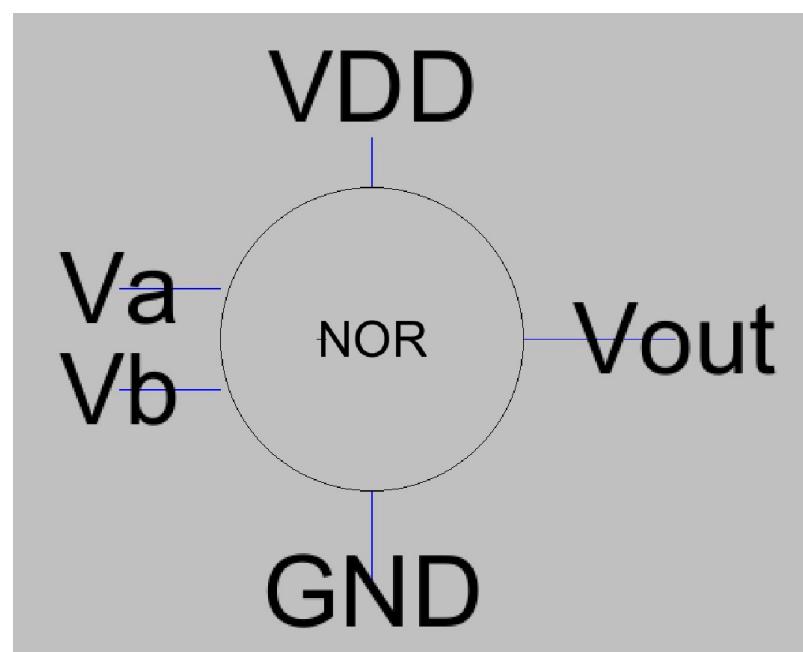


Figura 83: NOR - Símbolo.

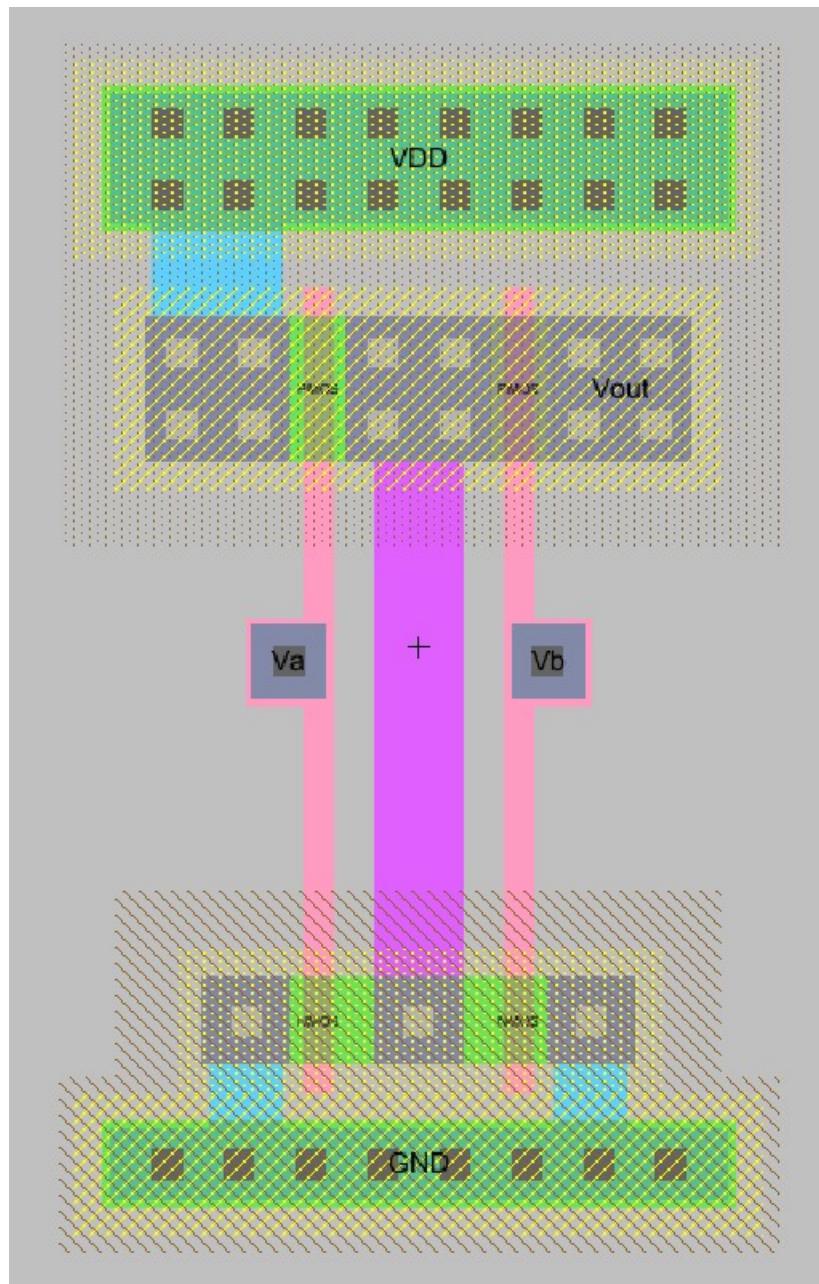


Figura 84: NOR - Layout.

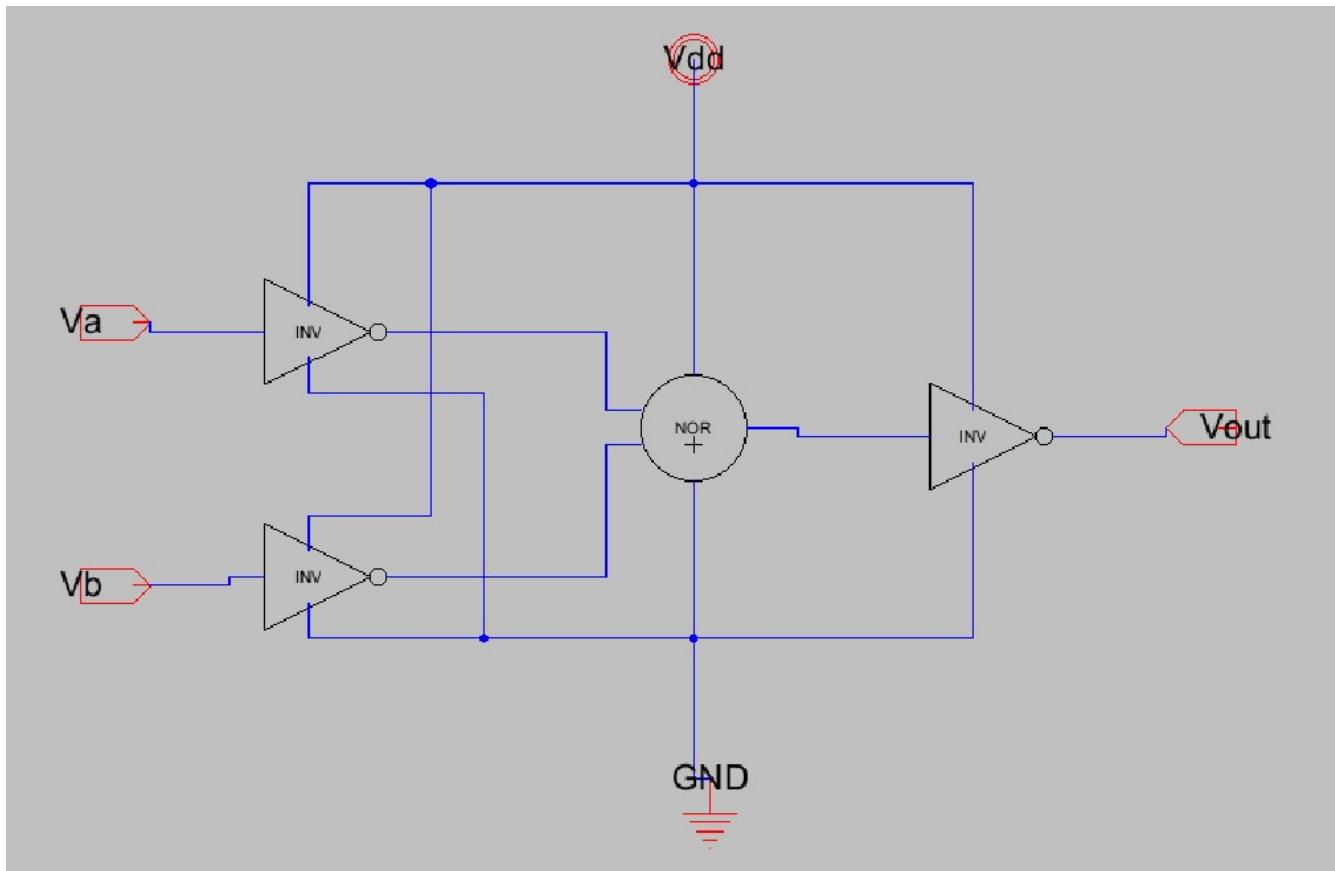


Figura 85: OR NOTANOTB - Esquemático.

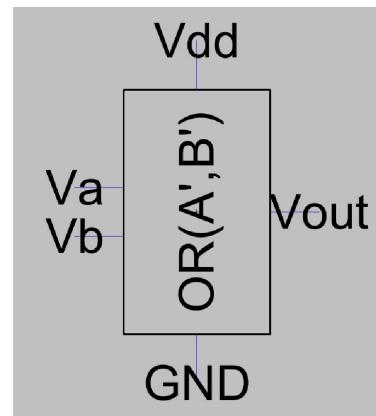


Figura 86: OR NOTANOTB - Símbolo.

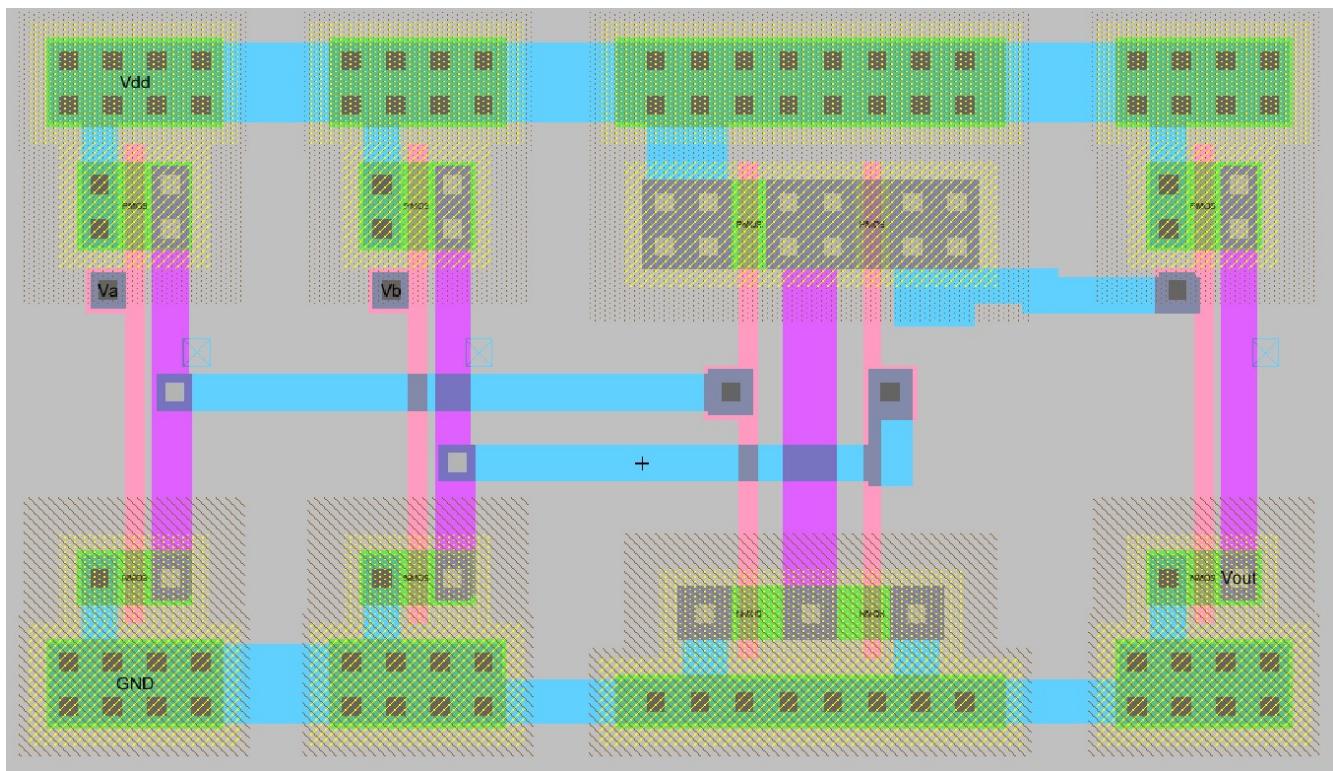


Figura 87: OR NOTANOTB - Layout.

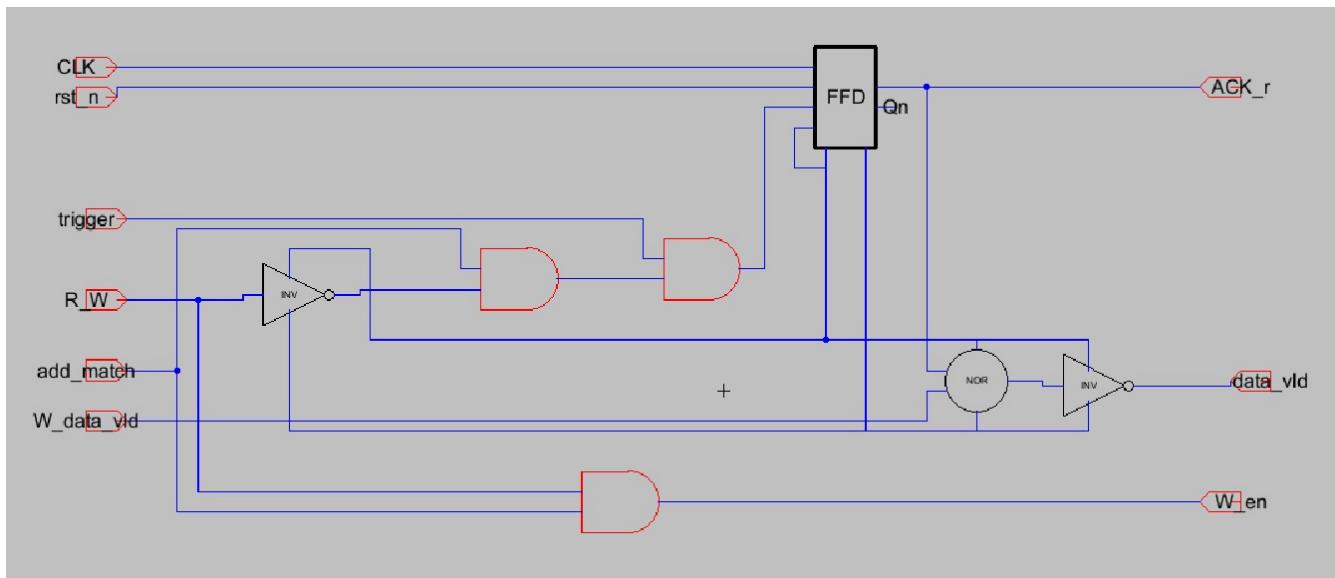


Figura 88: Read Write Module - Esquemático.

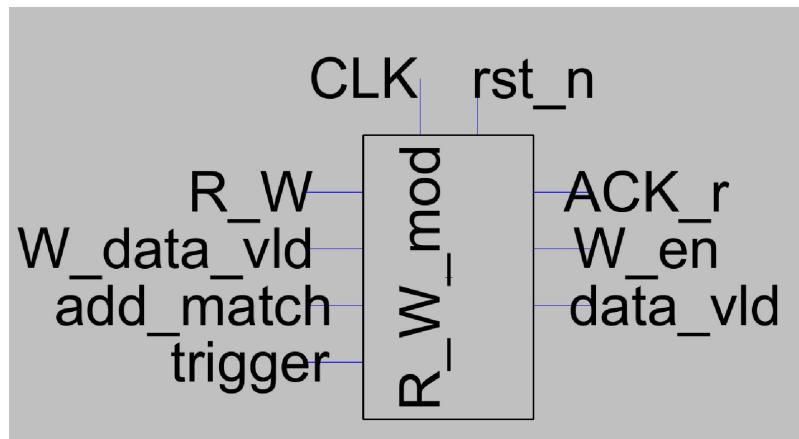


Figura 89: Read Write Module - Símbolo.

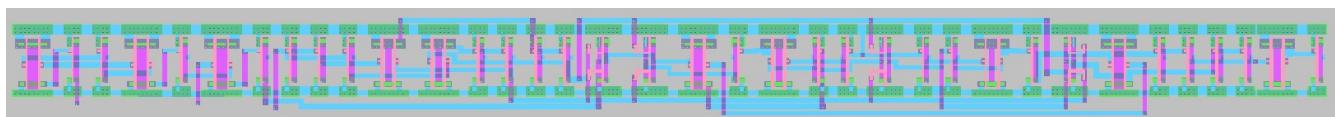


Figura 90: Read Write Module - Layout.

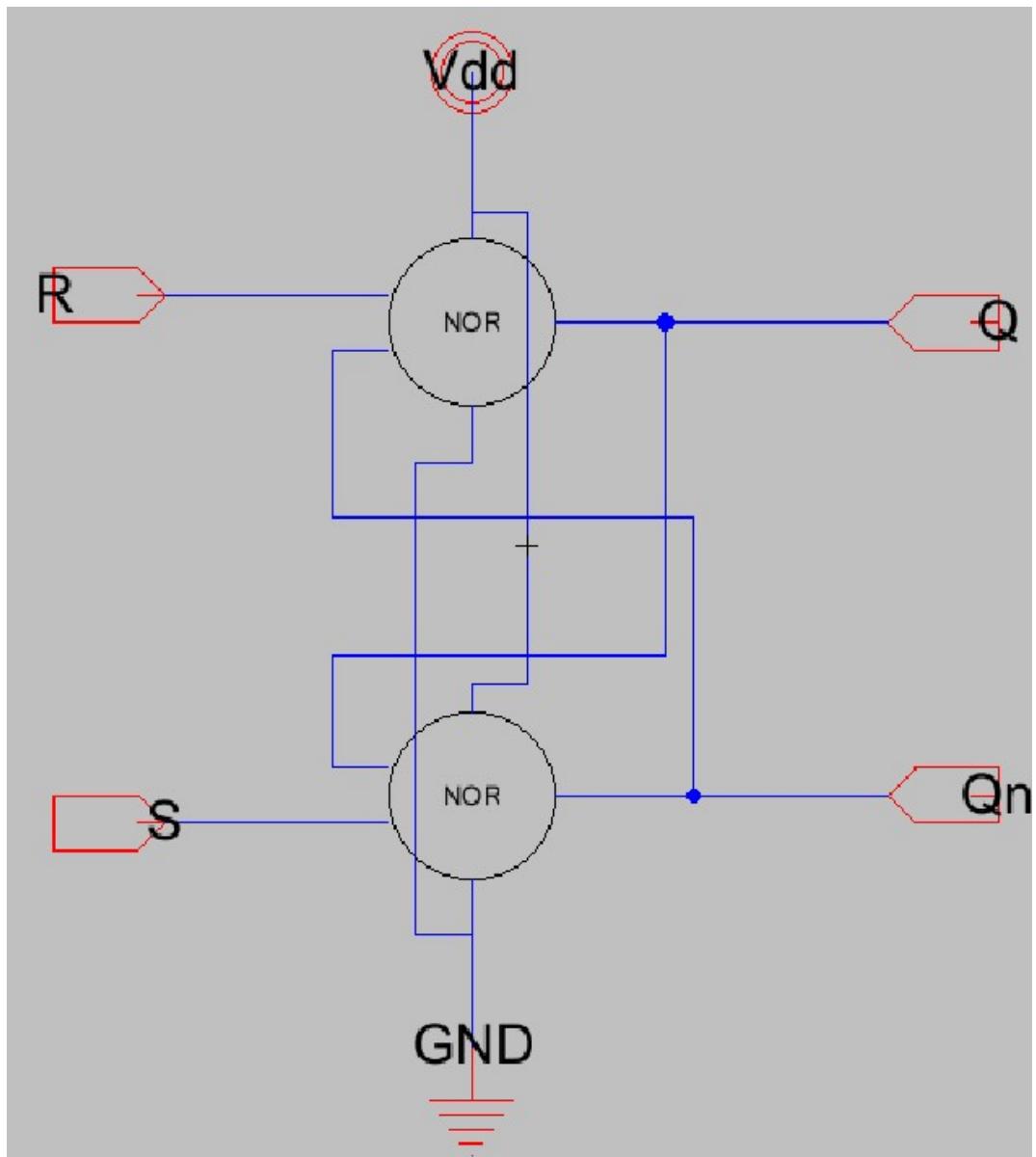


Figura 91: Set Reset - Esquemático.

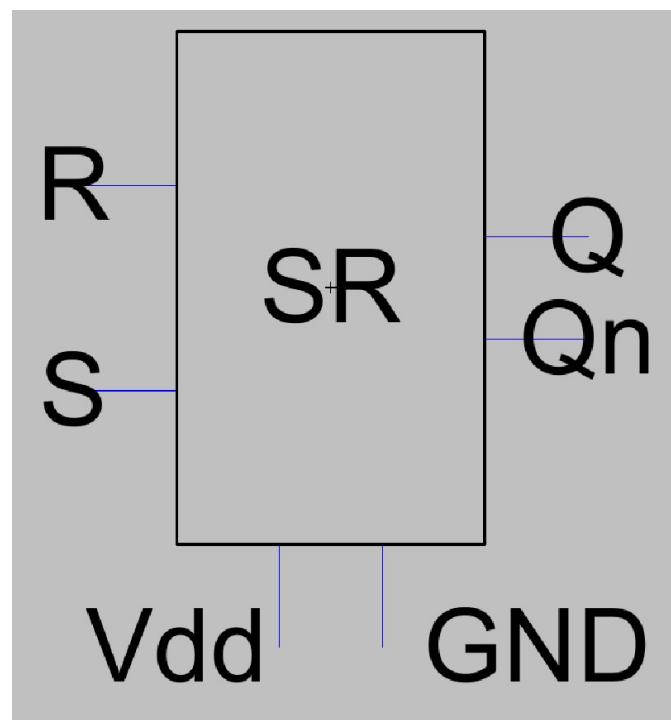


Figura 92: Set Reset - Símbolo.

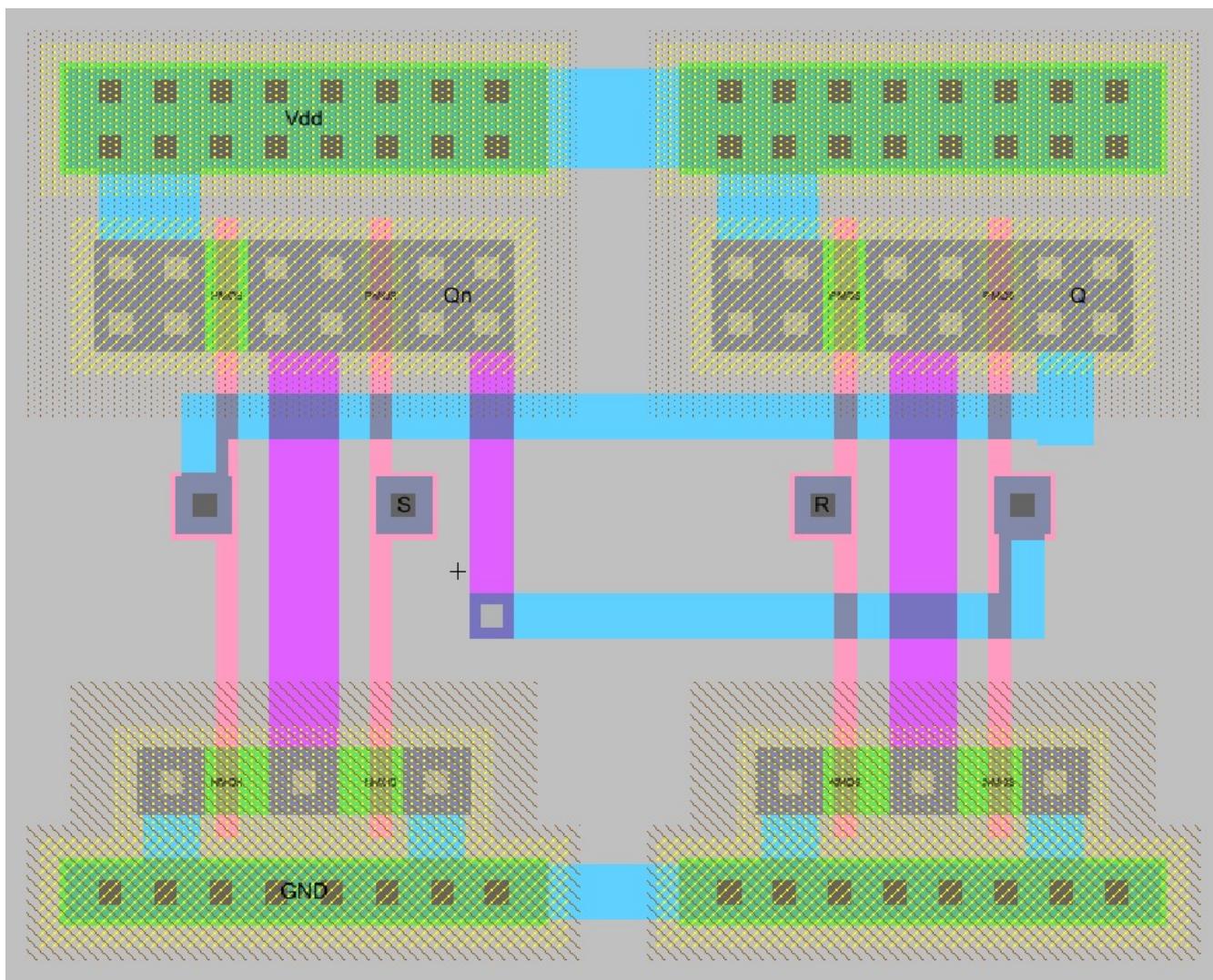


Figura 93: Set Reset - Layout.

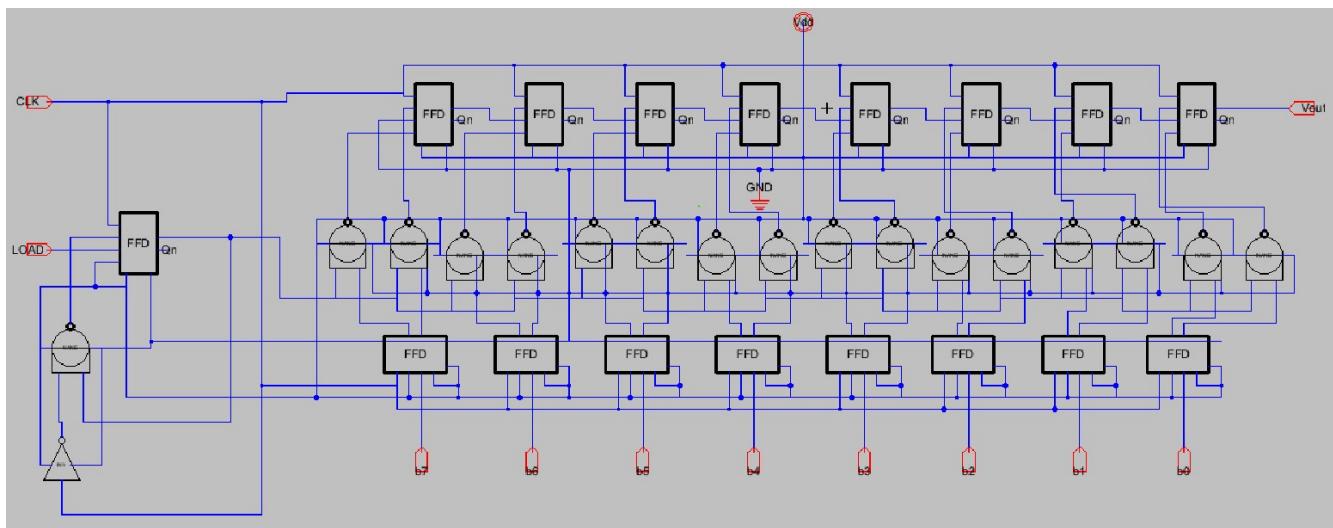


Figura 94: Shift Register PISO - Esquemático.

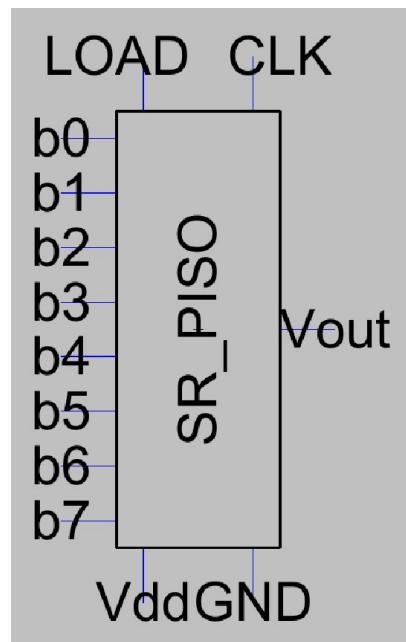


Figura 95: Shift Register PISO - Símbolo.

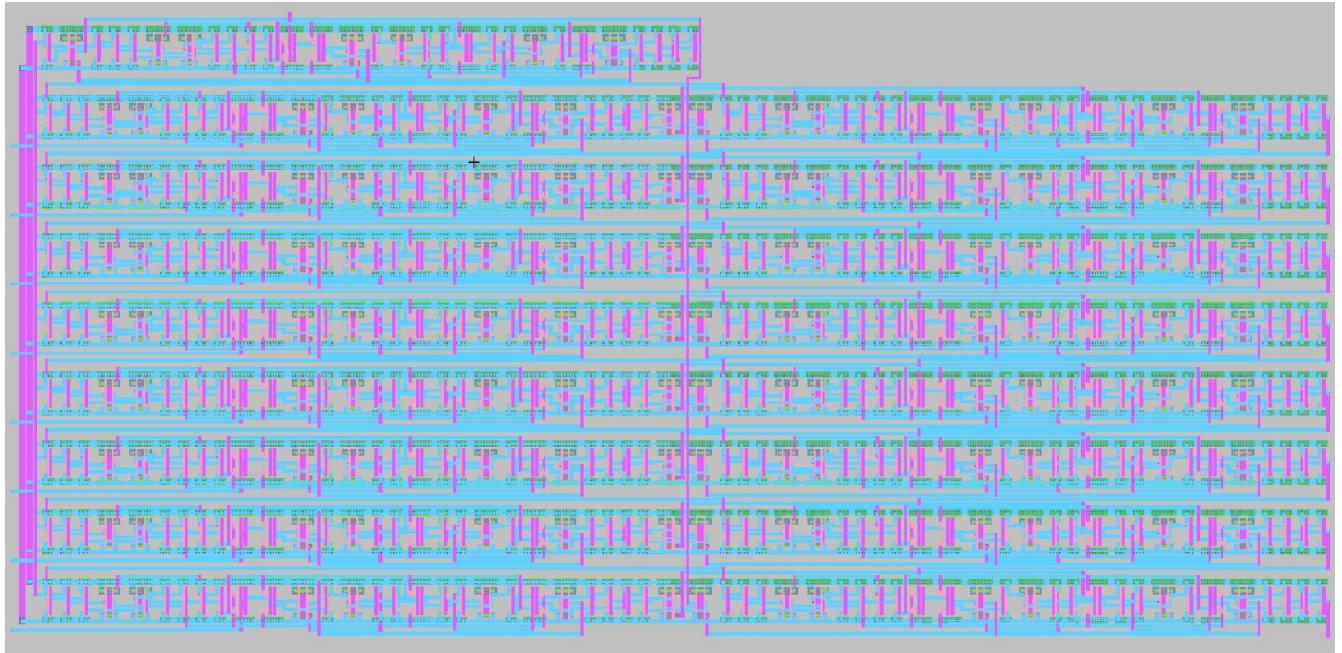


Figura 96: Shift Register PISO - Layout.

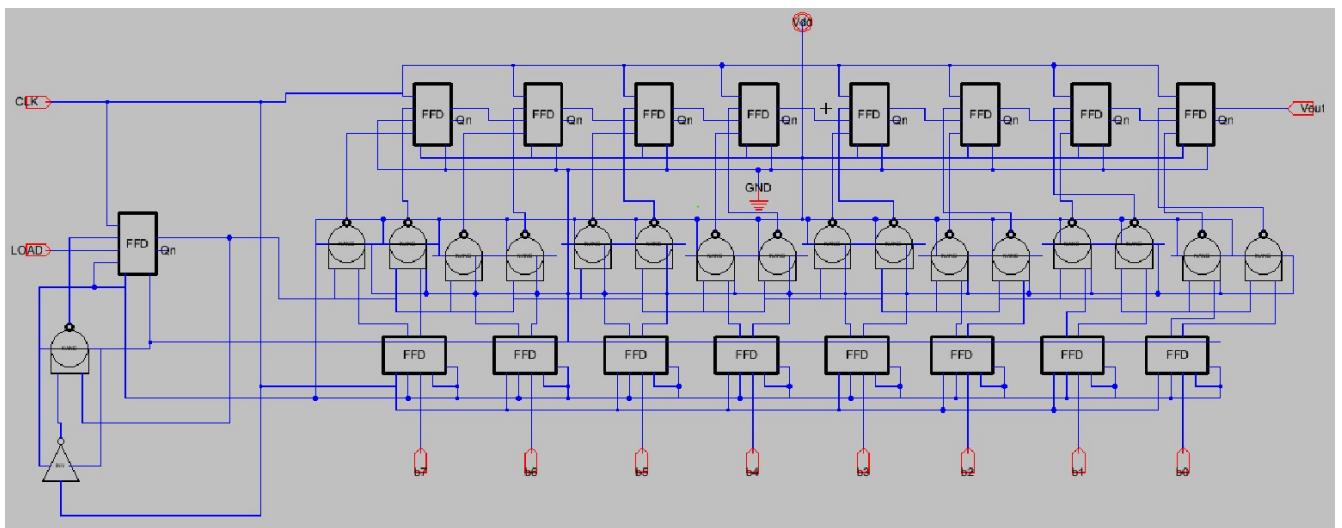


Figura 97: Shift Register PISO 8b - Esquemático.

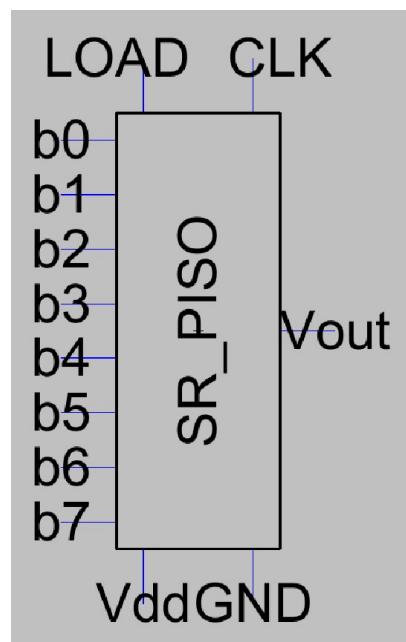


Figura 98: Shift Register PISO 8b - Símbolo.

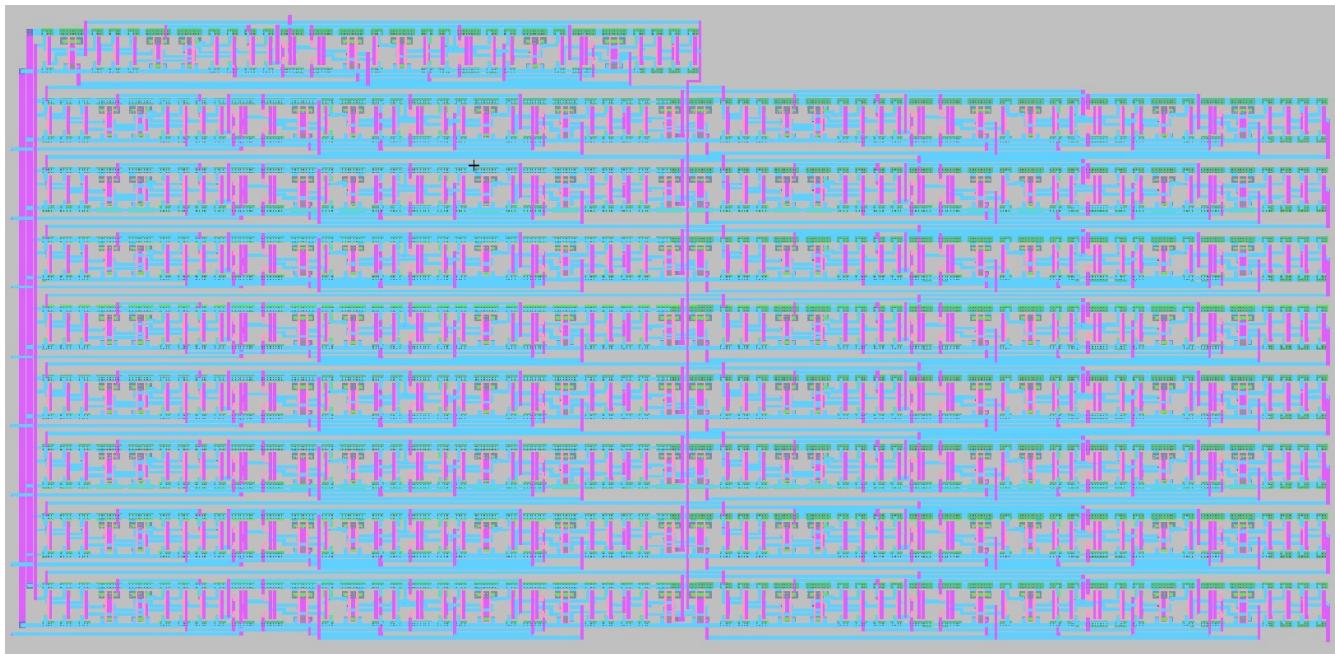


Figura 99: Shift Register PISO 8b - Layout.

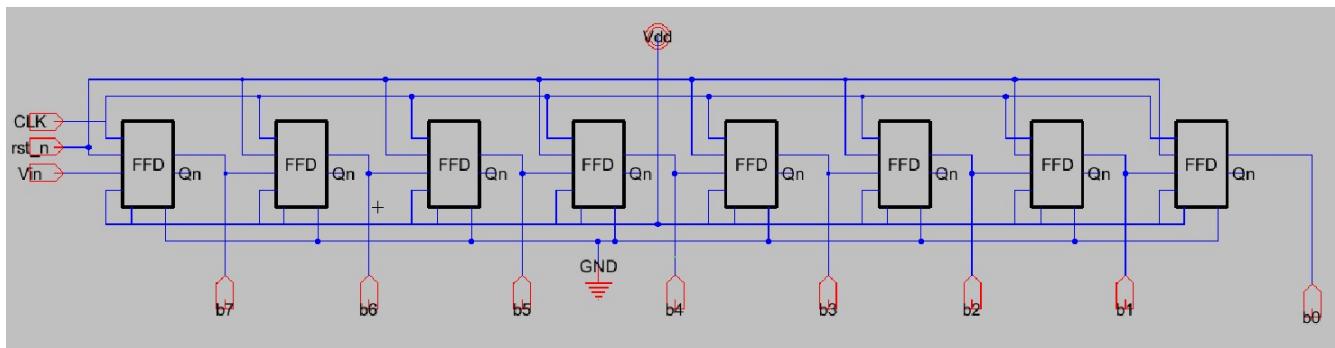


Figura 100: Shift Register SIPO 8b - Esquemático.

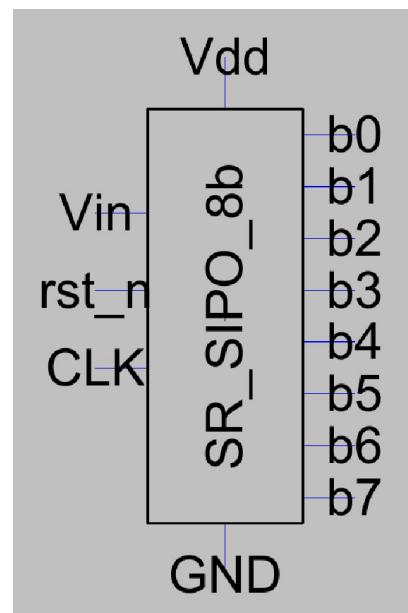


Figura 101: Shift Register SIPO 8b - Símbolo.

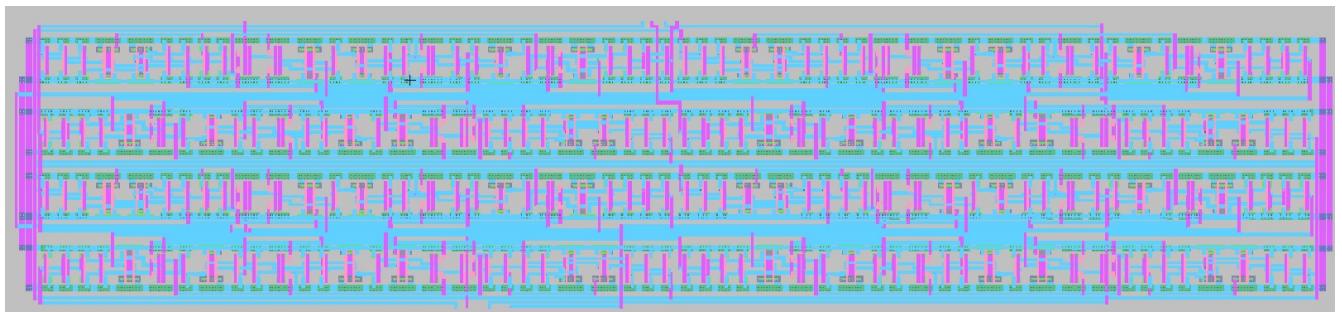


Figura 102: Shift Register SIPO 8b - Layout.

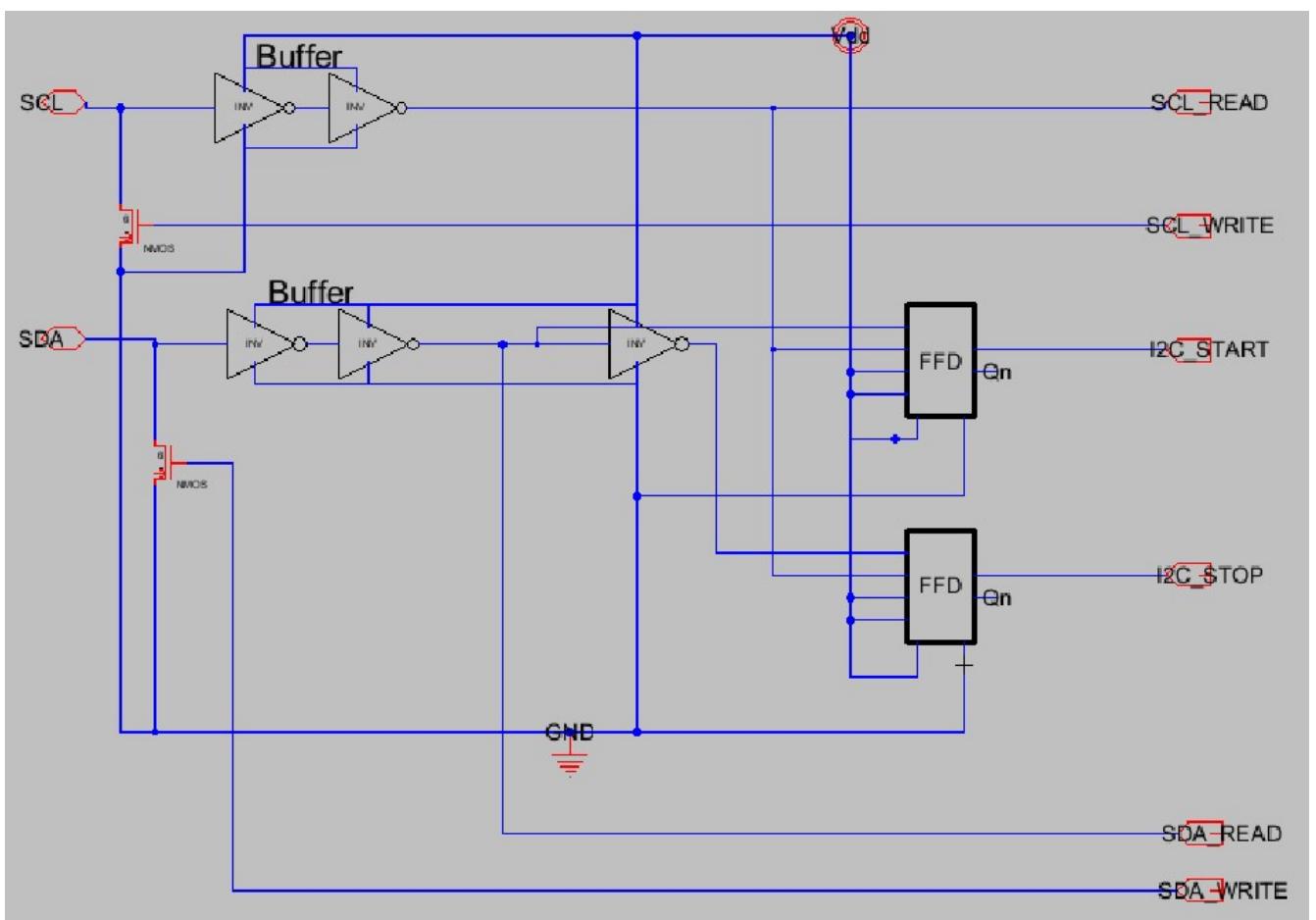


Figura 103: SS Detector - Esquemático.

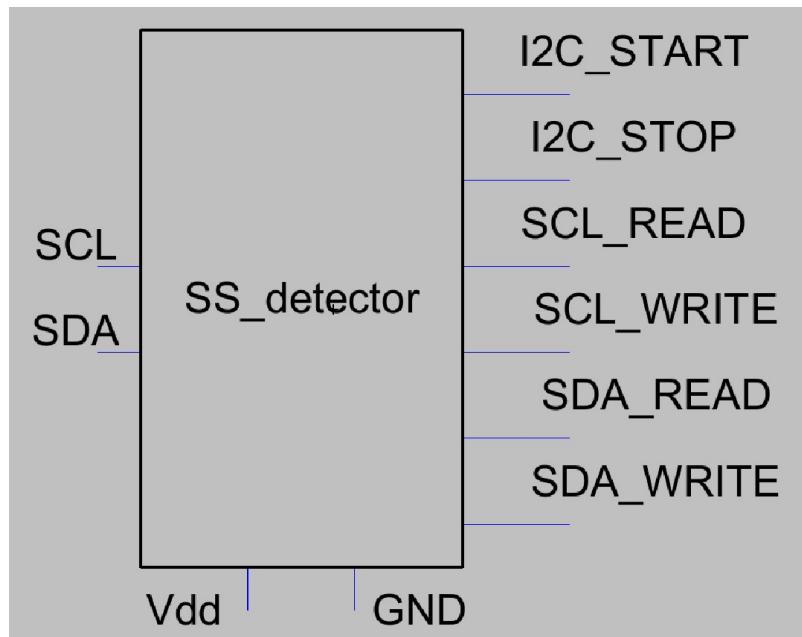


Figura 104: SS Detector - Símbolo.

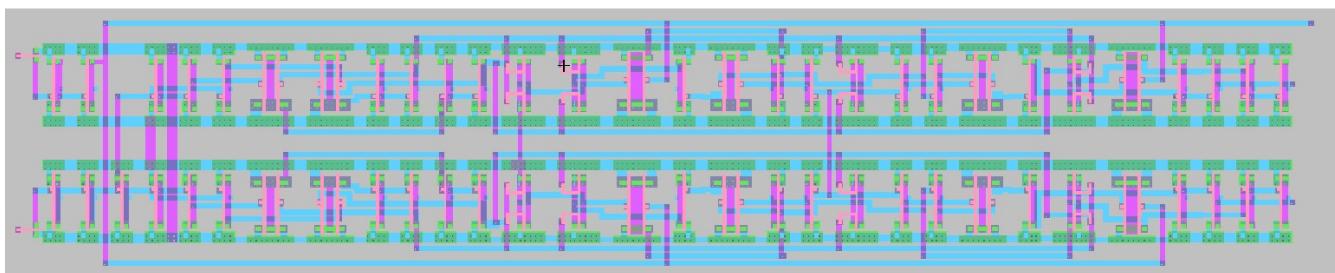


Figura 105: SS Detector - Layout.

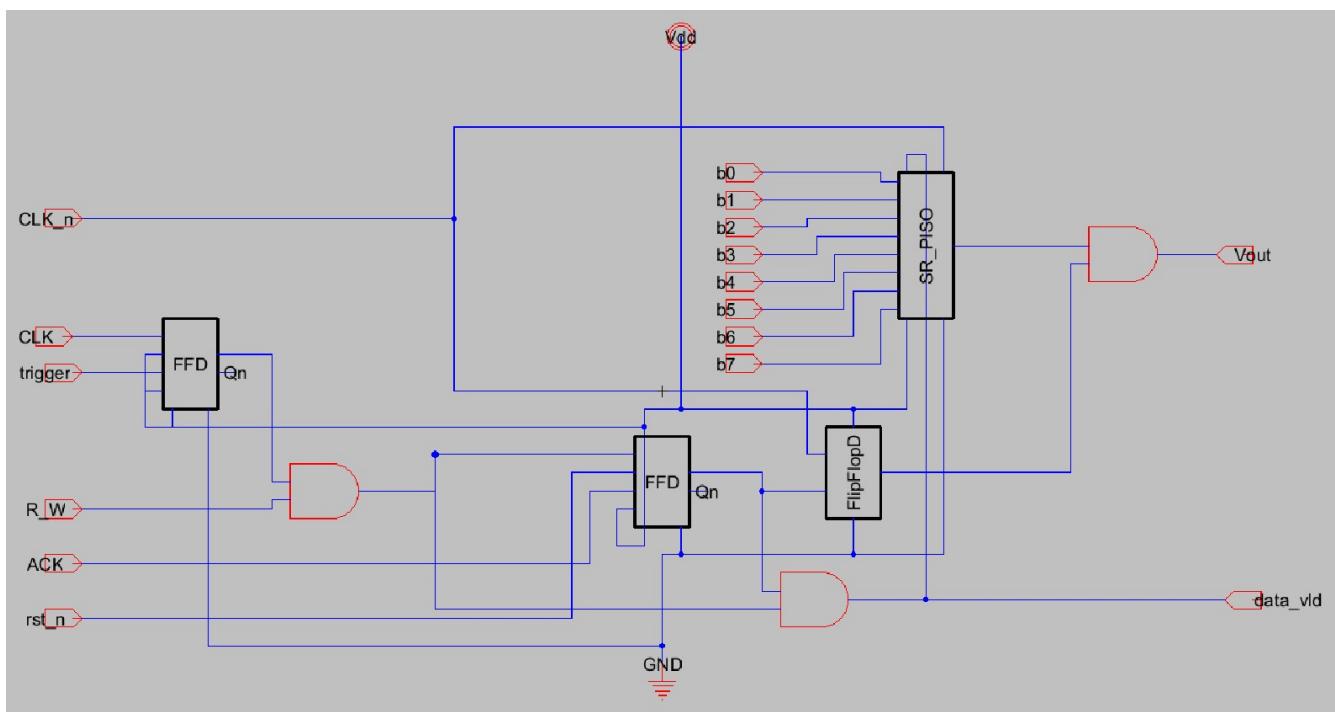


Figura 106: Write Module - Esquemático.

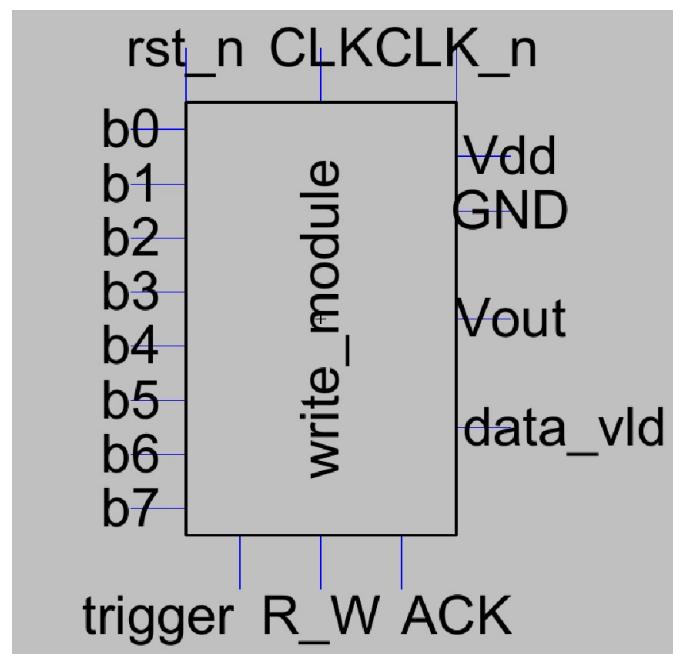


Figura 107: Write Module - Símbolo.

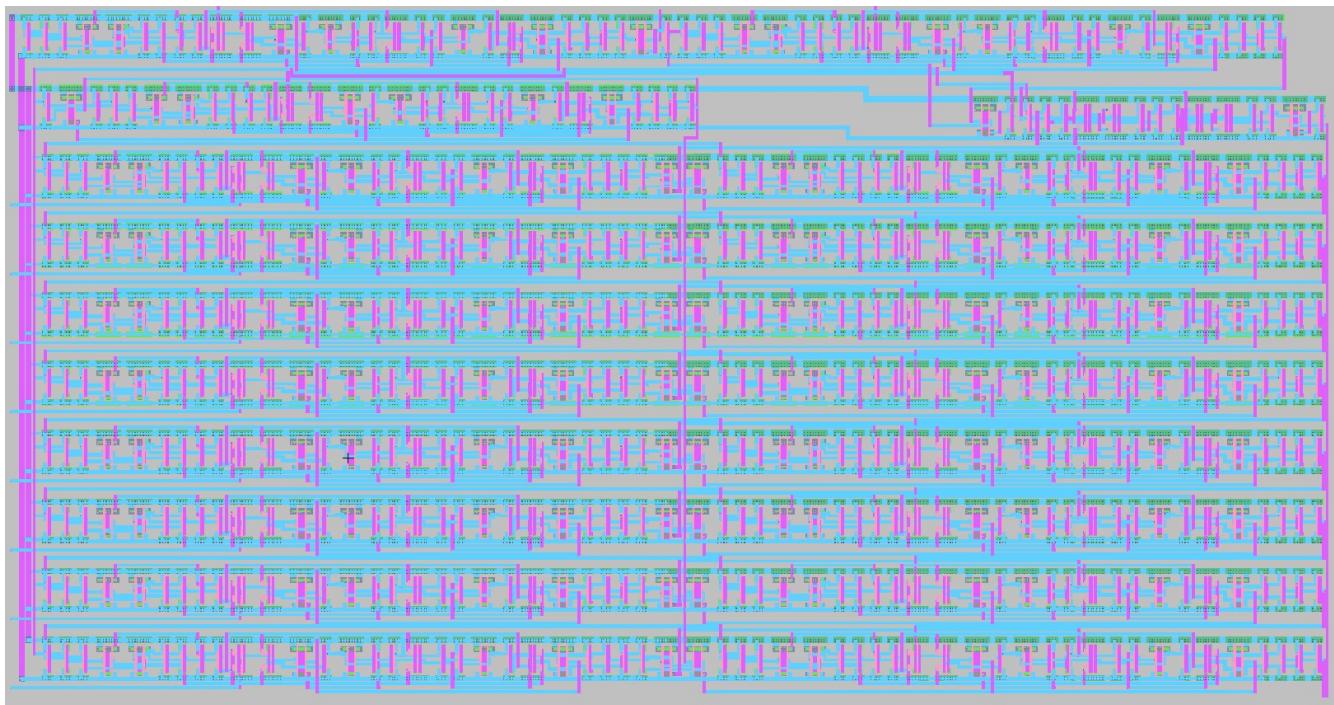


Figura 108: Write Module - Layout.

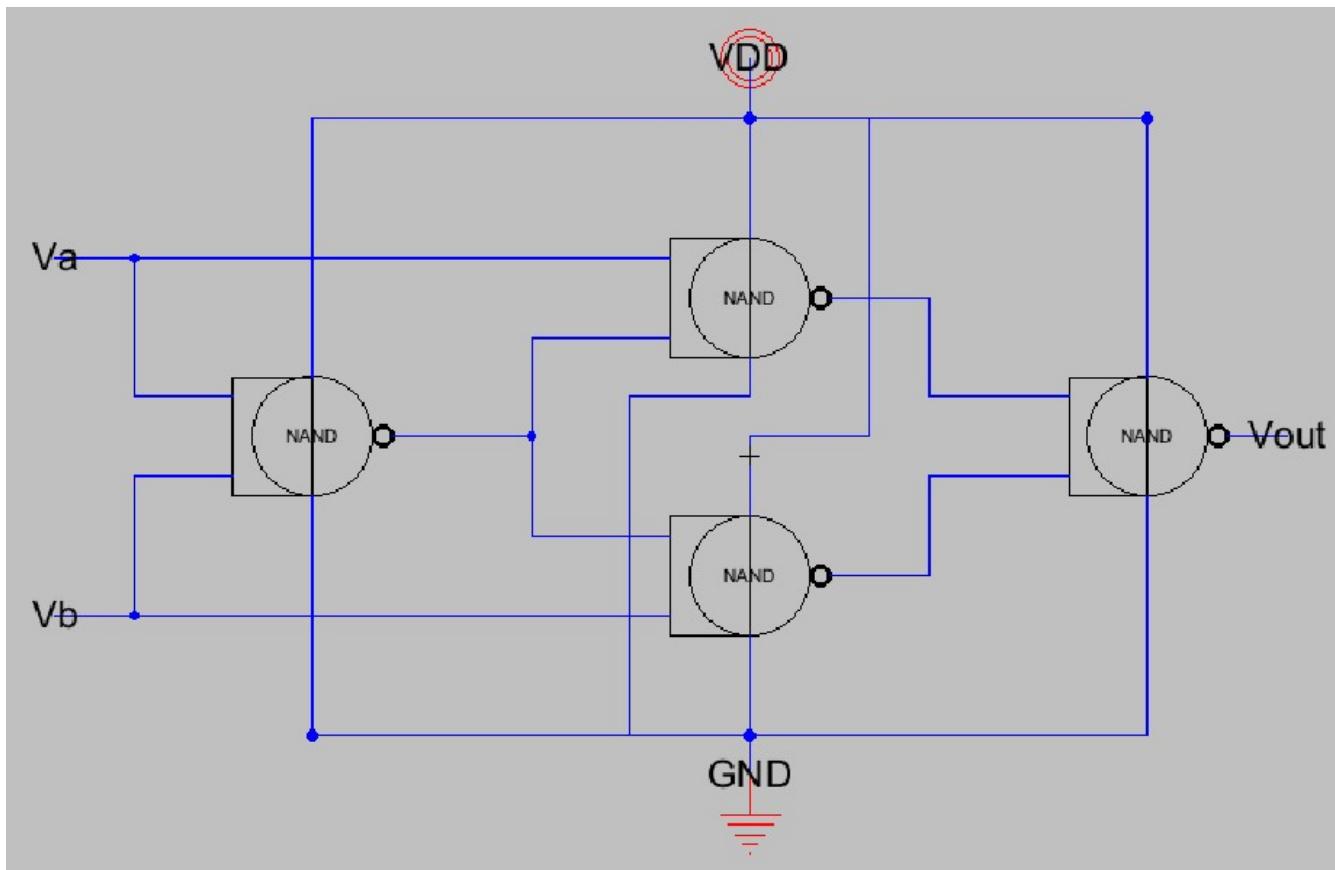


Figura 109: XOR - Esquemático.

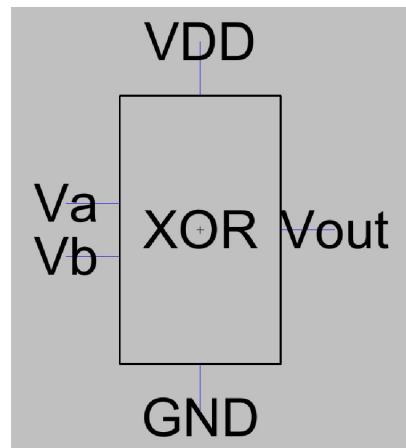


Figura 110: XOR - Símbolo.

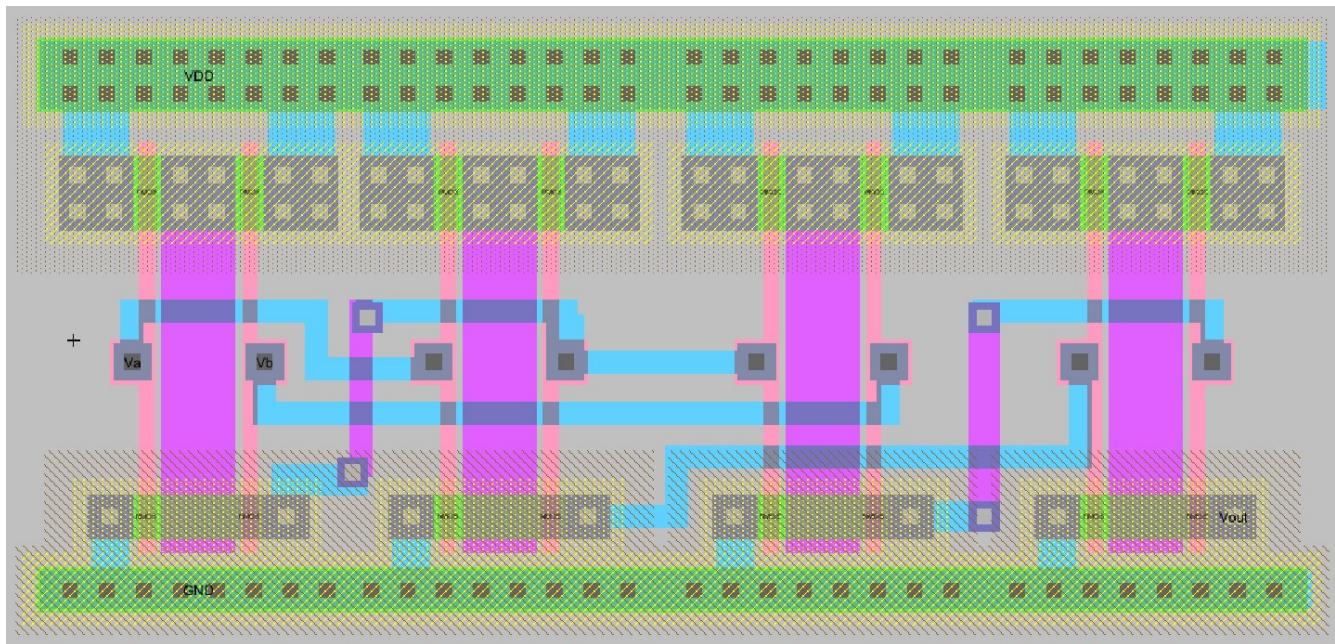


Figura 111: XOR - Layout.