EIE
Escuela de
Ingeniería Eléctrica

**Universidad de Costa Rica**
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica

UNIVERSIDAD DE COSTA RICA

`IE0411` Microelectrónica -G01-

# Proyecto final

Jorge Muñoz Taylor (A53863) - `jorge.munoztaylor@ucr.ac.cr`

II-2020

# Índice

# 1. Desarrollo

Para la obtención de la frecuencia máxima y las medidas de los circuitos en Qflow se utilizó el comando[1] del código 1.

Código 1
```
qflow sta ARCHIVO
```

Donde ARCHIVO corresponde al nombre del archivo que se va a analizar, cabe resaltar que dependiendo de la versión de Qflow que se tenga instalada se debe colocar o no la extensión del archivo verilog.

Para generar el layout se utilizará el programa magic, con este mismo software se calculará el área de cada circuito. Los comandos necesarios para ello se muestran en el código 2, donde el área se utiliza multiplicando el valor de la longitud horizontal por la longitud vertical del circuito.

Código 2
```
lef read /usr/share/qflow/tech/osu035/osu035_stdcells.lef
def read ./../layout/CIRCUITO.def
measure horizontal
measure vertical
```

## 1.1. Tecnologías

Para mapear los diseños creados en verilog se utilizan bibliotecas que contienen el tipo de compuertas que se utilizarán junto con sus características físicas, para el proyecto se usarán las bibliotecas *OSU* [1] propiamente las OSU035 y OSU050. El número indica el tamaño de los transistores que componen las compuertas de la biblioteca.

## 1.2. Señales de verificación

Para analizar la información obtenida de las simulaciones se crearon 4 señales orientadas a ello, las llamaré señales de **fallo**:

- enable_reset_fallo
- Q_fallo
- load_fallo
- rco_fallo

Estas señales se ven reflejadas en las simulaciones en *GTKwave* en la parte inferior de la imágen, cuando algún error ocurre en la señal de salida o *reset/enable* las señales de fallo pasarán al estado

[1] Acrónimo de *Oklahoma Satate University*, que corresponde al nombre de la universidad donde fueron desarrolladas estas bibliotecas.

ALTO de lo contrario permanecerán en BAJO.

Cada simulación está subdividida en 4 imágenes, la primera figura muestra la parte **no** aleatoria de la simulación mientras que la segunda figura muestra la parte aleatoria, por razones de espacio cada parte de la simulación de dividió en dos y se colocaron una sobre otra, pero recuerde que forma parte de la misma simulación.

Bastará con un fallo para considerar que esa característica no funciona adecuadamente.

### 1.3. AOI22

Se trata de una compuerta que lleva a cabo la función lógica AND seguida de una NOR, ambas de 2 entradas. Esta compuerta es muy utilizada en CMOS debido a las ventajas en espacio y velocidad que ofrece, ya que su conteo de transistores es menor a que si se implementará directamente la función con compuertas AND y NOR. En la figura 1 se puede ver su representación simbólica.

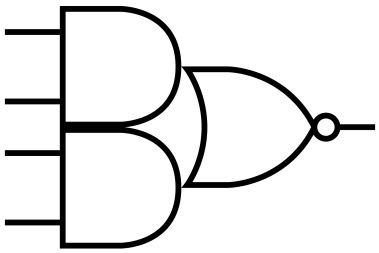

Figura 1: Diagrama de compuertas de una AOI22. Imágen tomada de `https://en.wikipedia.org/wiki/AND-OR-Invert#/media/File:AOI22Symbol.svg`.

# 2. Parte A: Diseño y verificación funcional

## 2.1. Pruebas realizadas sobre el contador

El contador tiene cinco entradas y tres salidas, para probar el funcionamiento de dicho contador se siguieron las pautas a continuación:

- Primero se resetea el contador colocando *enable* en 0 y *reset* en 1, para que inicialize en $Q = 0$, luego de 1 ciclo se pone *enable* en 1 y *reset* en 0 para comenzar el conteo. También en $56400ns$ se cambia los valores a $enable = 0$-$reset = 0$ y en $70400ns$ se cambian a $enable = 1$-$reset = 0$ para verificar el comportamiento de $Q$.
- Se toman uno a uno, los cuatro modos de operación del contador para verificar que cada uno cuenta como debe hacerlo: en 00 suma de 1 en 1, en 01 resta 1, en 10 resta de 3 en 3 y el 11 cada el valor de la entrada D en Q.
- A partir de cierto punto de la simulación y por medio de un loop *forever*, la entrada *mode* cambia aleatoriamente al igual que las entradas *reset* y *enable*, de esta forma no se tiene control sobre las posibles salidas y así comprobar los resultados obtenidos en la parte no aleatoria.
- La entrada D es aleatoria durante toda la simulación.

El diseño del *testbench* se realizó siguiendo la tabla de verdad que se muestra en el cuadro 1, note que la entrada nodo *no importa* (X) en tres casos, esto es así a pesar de que no se menciona en las especificaciones ya que, en el primer caso, al tener enable y reset en 0 la salida Q debe estar en 0 por lo que el valor que tenga la entrada modo no importa que valor tenga, de igual forma en los dos casos siguientes solo que con el reset, como es 1 todas las salidas deben valer 0.

| clk | enable | reset | modo | load | Q | rco |
|---|---|---|---|---|---|---|
| posedge | 0 | 0 | x | 0 | 0 | 0 |
| posedge | 0 | 1 | x | 0 | 0 | 0 |
| posedge | 1 | 0 | 00 | 0 | Q+1 | alto por medio ciclo |
| posedge | 1 | 0 | 01 | 0 | Q-1 | alto por medio ciclo |
| posedge | 1 | 0 | 10 | 0 | Q-3 | alto por medio ciclo |
| posedge | 1 | 0 | 11 | 1 | D | 0 |
| posedge | 1 | 1 | x | 0 | 0 | 0 |

Cuadro 1: Tabla de verdad que sigue el circuito contador de 4 bits con salida RCO en alto por medio ciclo de reloj tal como se describió en las especificaciones.

Los resultados de la simulación **en caso de errores** se guardan en un archivo de log *log_parteA.txt* en la carpeta llamada logs.

## 2.2. Resultados de la verificación

En la figura 2 se puede ver que el contador pone todas las salidas en 0 cuando la señal de reset está en alto, también puede apreciarse como el contador va restando 3 al valor actual de Q con cada flanco positivo del reloj. Y muy importante, en dicha figura se puede ver claramente que la señal RCO dura exactamente medio ciclo de reloj.

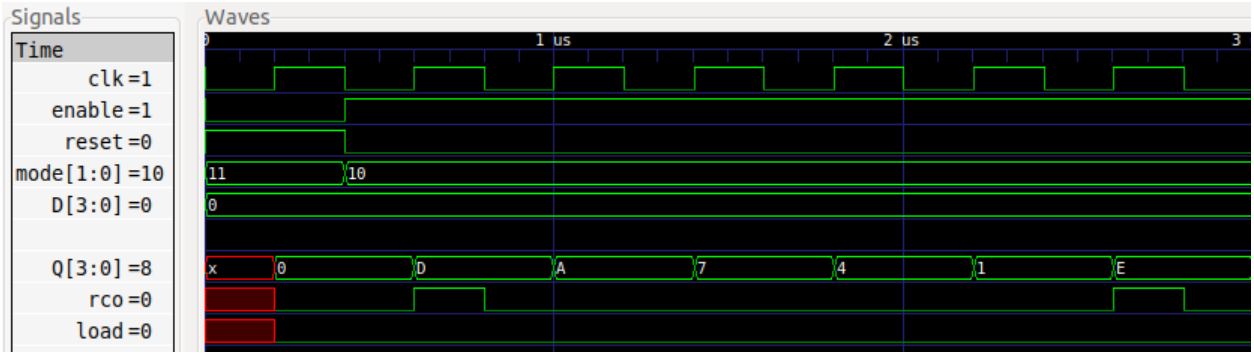


Figura 2: Resultado de la simulación del contador de 4 bits, note como la señal de RCO tiene una duración de medio ciclo cuando ocurre un rebase.

En la figura 3 se puede ver que el modo 01 se comporta como debe hacerlo, justo en el flanco positivo siguiente al cambio de modo se da el conteo descendente de 1 en 1.

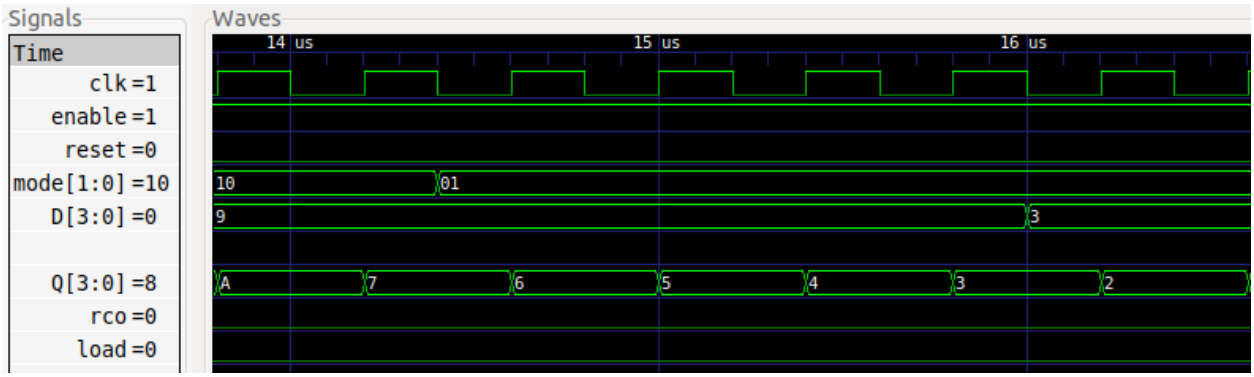


Figura 3: Resultado de la simulación del contador de 4 bits, el modo 01 (conteo descendente de 1 en 1) tiene el comportamiento esperado, note como el cambio de modo ocurre fluidamente es decir que el conteo se adecúa al modo y al valor inmediatamente anterior de Q.

En la figura 4 se prueba el modo de carga de la señal D en la salida Q, en la figura se cargan exitosamente dos valores de D en Q además note como la salida LOAD se pone en alto en el flanco positivo siguiente después de que la señal modo cambia a 11.

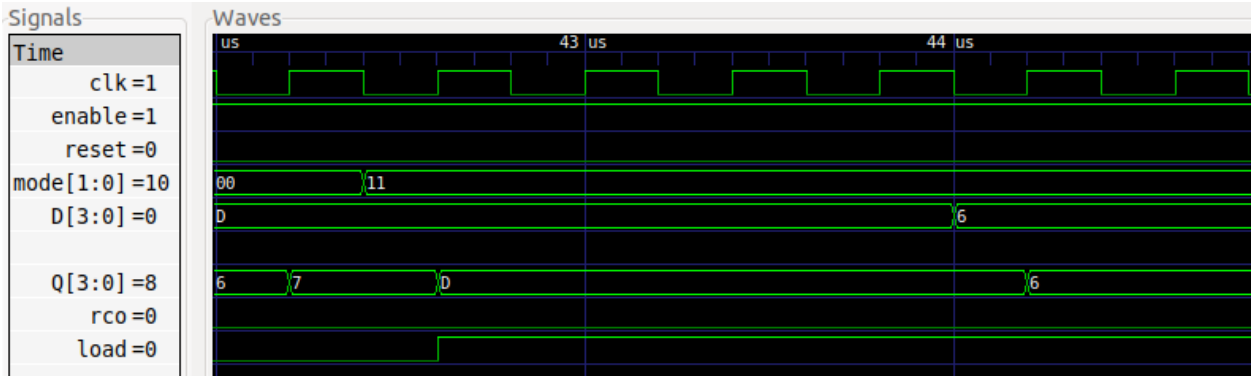


Figura 4: Resultado de la simulación del contador de 4 bits, aquí se verifica que el modo de funcionamiento 11 se comporta según lo esperado, puede notar como la salida Q toma el valor D cuando modo vale 11 y se da el flanco creciente del reloj.

En la figura 5 puede verse como las salidas del contador pasan a valer 0 cuando las señales de entrada ENABLE y RESET están en bajo.

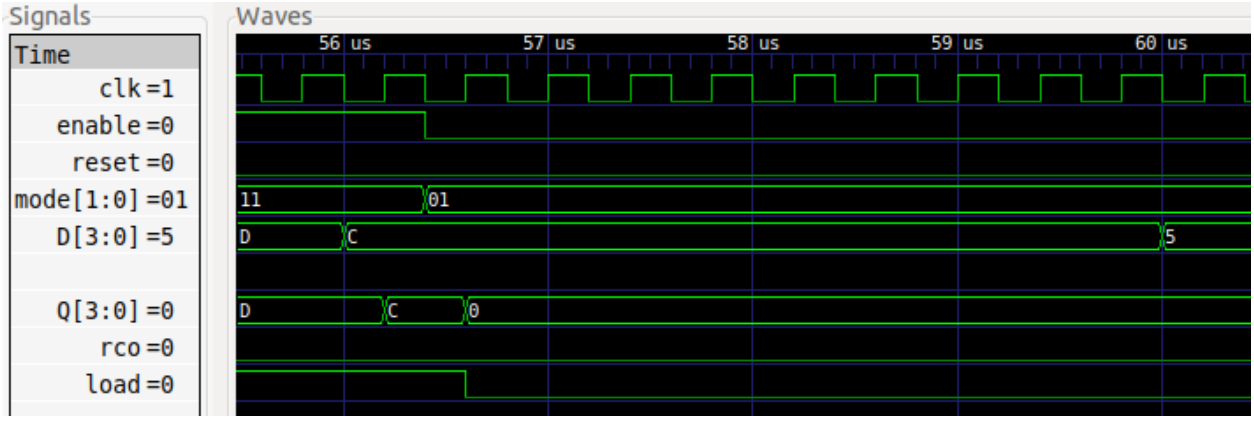


Figura 5: Resultado de la simulación del contador de 4 bits, las salidas del contador pasan a 0 cuando las señales de ENABLE y RESET tienen un valor BAJO.

En la figura 6 se puede notar que la señal LOAD identifica correctamente cuando el contador está configurado en el modo 11 (de carga directa) poniéndose en ALTO y cuando no lo está poniéndose en BAJO.

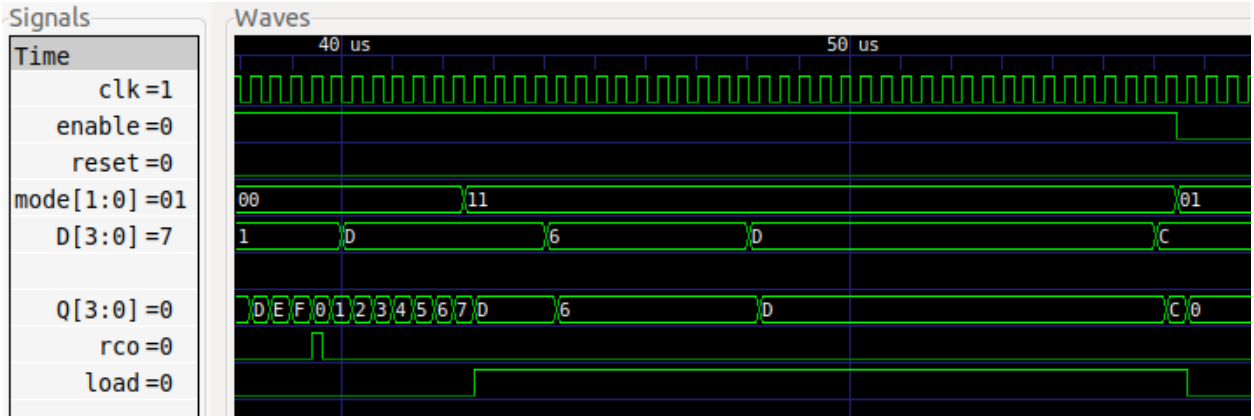


Figura 6: Resultado de la simulación del contador de 4 bits, la señal de salida LOAD está en ALTO cuando la señal de entrada MODO está en 11.

# 3. Parte B: Diseño y verificación estructural

Para construir el contador de 32 bits se utilizó con el contador de 4 bits diseñado en la parte A (con RCO activo durante medio ciclo de reloj) junto con el comando *generate* de verilog para simplificar la escritura de código y mejorar la legibilidad del mismo.

En la figura 7 se muestra que el modo 10 funciona correctamente, además note como cambian la salida Q a $FFFFFFFD$ cuando se le resta 3 a 00000000 en un sólo ciclo de reloj.

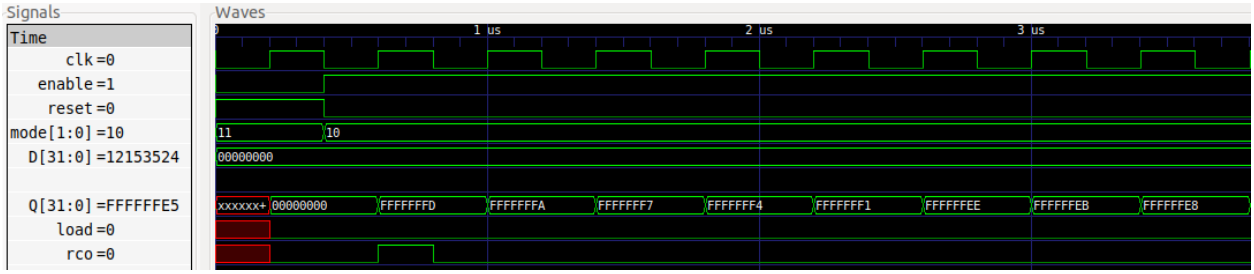


Figura 7: El modo 10 se comporta según las especificaciones, también se aprecia como la salida RCO se mantiene en ALTO durante medio ciclo de reloj cuando ocurre el rebase, esto ocurre porque la señal RCO del contador de 32 bits está conectada a la señal RCO del octavo contador de 4 bits que componen el contador de 32 bits, de ahí que sea de medio ciclo.

En la figura 8 se puede observar el modo de conteo descendente, en este caso sólo cambia el LSB hasta que ocurre un rebase mismo donde se activa el nibble siguiente, y así de forma sucesiva hasta el último nibble.

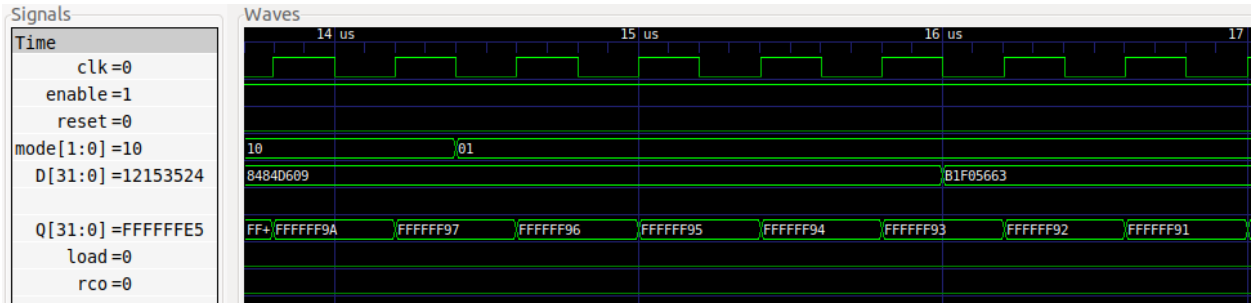


Figura 8: Conteo descendente del contador de 32 bits (modo 01).

En la figura 9 se observa el modo de conteo ascendente, de igual forma que en el conteo descendente sólo cambia el nibble menos significativo hasta que ocurre un rebase, cuando se da esta situación el RCO del nibble se pone el alto y activa el contador de 4 bits siguiente, sumándole 1 a ese nibble.

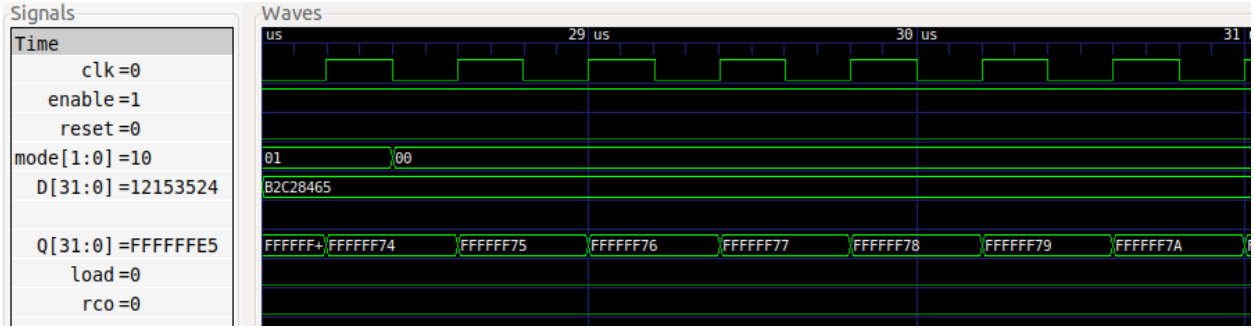


Figura 9: Conteo ascendente del contador de 32 bits (modo 00).

En la figura 10 puede apreciarse la carga del valor de la entrada D en el contador de 32 bits, mostrando su valor en la salida Q junto con la puesta en ALTO de la señal de salida LOAD. En la figura 11 se muestra un diagrama más claro de la salida LOAD.

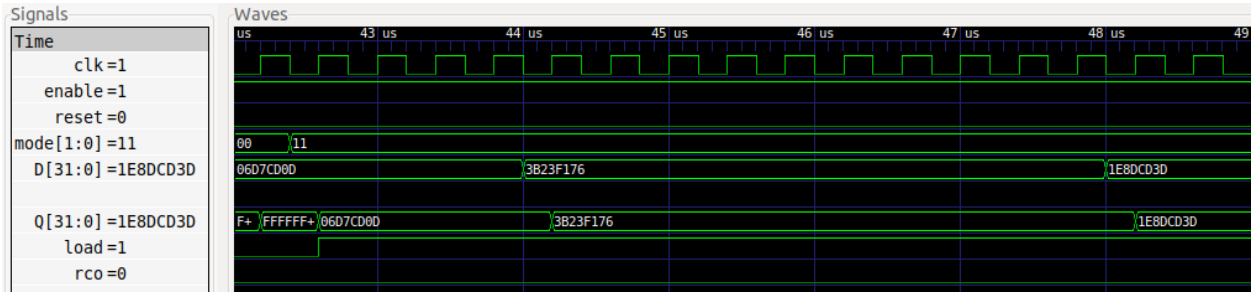


Figura 10: Carga directa en el contador de 32 bits (modo 11), también en este modo la salida LOAD se pone en ALTO que se cambie a otro modo de operación.

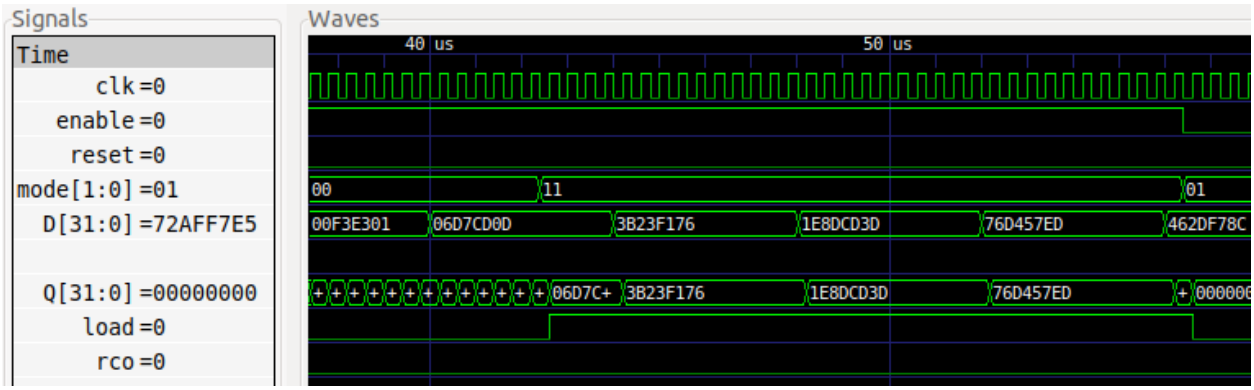


Figura 11: Representación más clara de la salida LOAD del contador de 32 bits cuando se da el modo 11.

La figura 12 tiene por objetivo mostrar el cambio de la salida Q cuando su valor es 00000000 y ocurre un rebase durante una resta de 1, produciéndose un valor de $FFFFFFFF$ en flanco creciente del próximo ciclo de reloj, este resultado indica que todos los contadores están sincronizados y conectados correctamente.

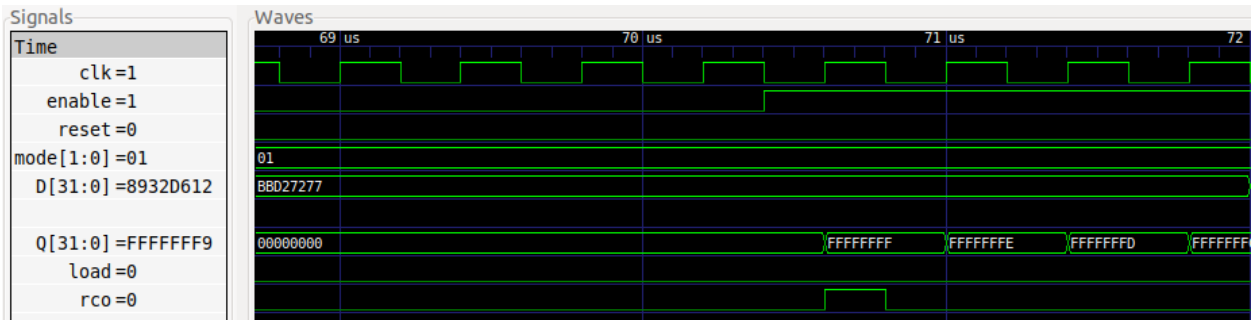


Figura 12: Cambio en la señal de salida Q cuando se da un rebase descendente.

# 4. Parte C: Verificación de errores

El contador de 4 bits se modificó de forma tal que ahora la salida RCO tiene una duración de un ciclo completo de reloj cuando se activa, por lo demás se comporta igual que el contador de 4 bits diseñado en la parte A por lo que sólo se mostrará como cambia la salida RCO cuando se da un rebase (ver la figura 13).

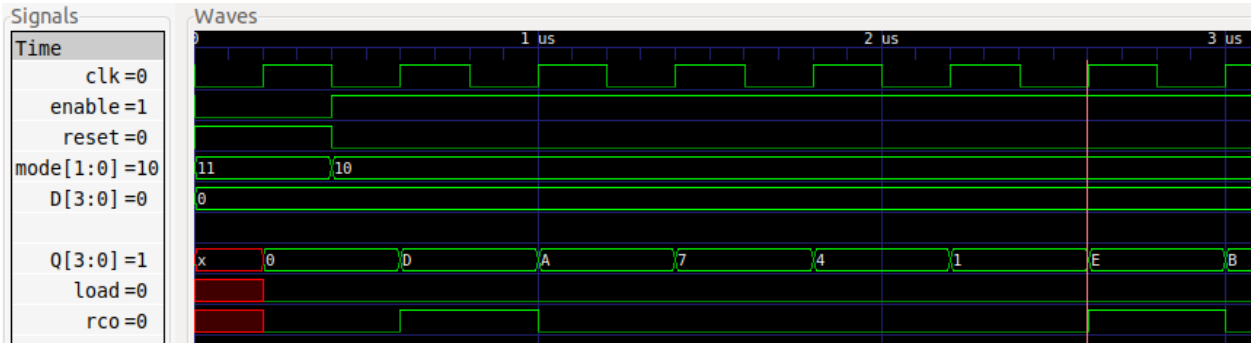


Figura 13: Comportamiento de la señal RCO del contador de 4 bits modificado, note que esta permanece durante un ciclo completo de reloj en alto cuando ocurre un rebase en el contador.

Cuando se utiliza este contador de 4 bits modificado en el contador de 32 bits **no ocurre ningún mal funcionamiento**, esto debido a la forma en la que se diseñó el contador de 32 bits.

El contador de 32 bits se diseñó de forma que sólo funcione permanente el primer contador de 4 bits (el correspondiente al primer nibble), los demás sólo se activan cuando el contador inmediatamente anterior tiene un rebase y capturando durante un ciclo entero de reloj la señal RCO de ese contador (por eso no le afecta la duración en alto del RCO de los contadores de 4 bits). **La única diferencia radica (como se explicó anteriormente) en que la señal RCO tendrá una duración de un ciclo entero de reloj, ya que como el RCO del contador de 32 bits está conectado al RCO del contador de 4 bits más significativo de los que lo conforman**. Este resultado se muestra en la figura 14.

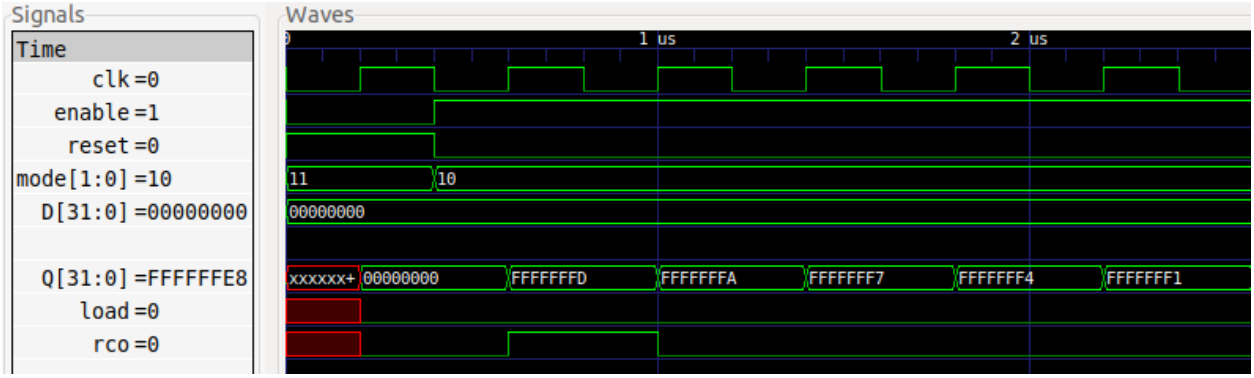


Figura 14: Contador de 32 bits hecho a partir de 8 contadores de 4 bits con señal RCO de un ciclo de reloj, note que el comportamiento no ha cambiado respecto al contador de 32 bits anterior, sólo se diferencia en el tiempo de sostenimiento de la señal RCO (que no afecta el funcionamiento del contador).

# 5. Parte D: Timing y layout

## 5.1. Contador de 4 bits

### 5.1.1. OSU035

Para los retardos máximos (descritos en el cuadro 11) se tiene, flop to flop: *path DFFPOSX1_1/CLK to DFFPOSX1_3/D*, pin to flop: *path input MODO[0] to DFFPOSX1_2/D*.

Para los retardos mínimos (descritos en el cuadro 11), flop to flop: *path DFFPOSX1_1/CLK to DFFPOSX1_1/D*, pin to flop: *pin CLK to DFFPOSX1_6/CLK*.

| Retardos | Máximo (ps) | Mínimo (ps) |
|---|---|---|
| Flop to flop | 1493,47 | 472,3 |
| Pin to flop | 914,32 | 0 |

Cuadro 2: Retardos flop-to-flop y pin-to-flop del circuito *counter4b.v* utilizando la biblioteca de componentes *osu035*.

Los valores de frecuencia máxima y medidas del circuito *counter4b.v* se muestran en el cuadro 12, el *layout* del mismo circuito con la biblioteca *osu050* es el presentado en la figura 15.

| | |
|---|---|
| $f_{max}$ [MHz] | 669,58 |
| Medida horizontal [$\mu m$] | 133 |
| Medida vertical [$\mu m$] | 87 |
| Area [$nm^2$] | 11.571 |

Cuadro 3: Área y frecuencia máxima del circuito *counter4b.v* usando la biblioteca osu035.

En el cuadro 13 se presenta el conteo de compuertas usadas durante la síntesis.

| Compuerta | Cantidad |
|---|---|
| AND | 7 |
| AOI3 | 7 |
| AOI4 | 2 |
| NAND | 7 |
| NOR | 6 |
| NOT | 10 |
| OAI3 | 9 |
| OAI4 | 2 |
| OR | 11 |
| XNOR | 1 |
| XOR | 7 |

Cuadro 4: Compuertas generadas por Qflow durante el proceso de síntesis para el circuito *counter4b.v* usando la biblioteca *OSU035*.

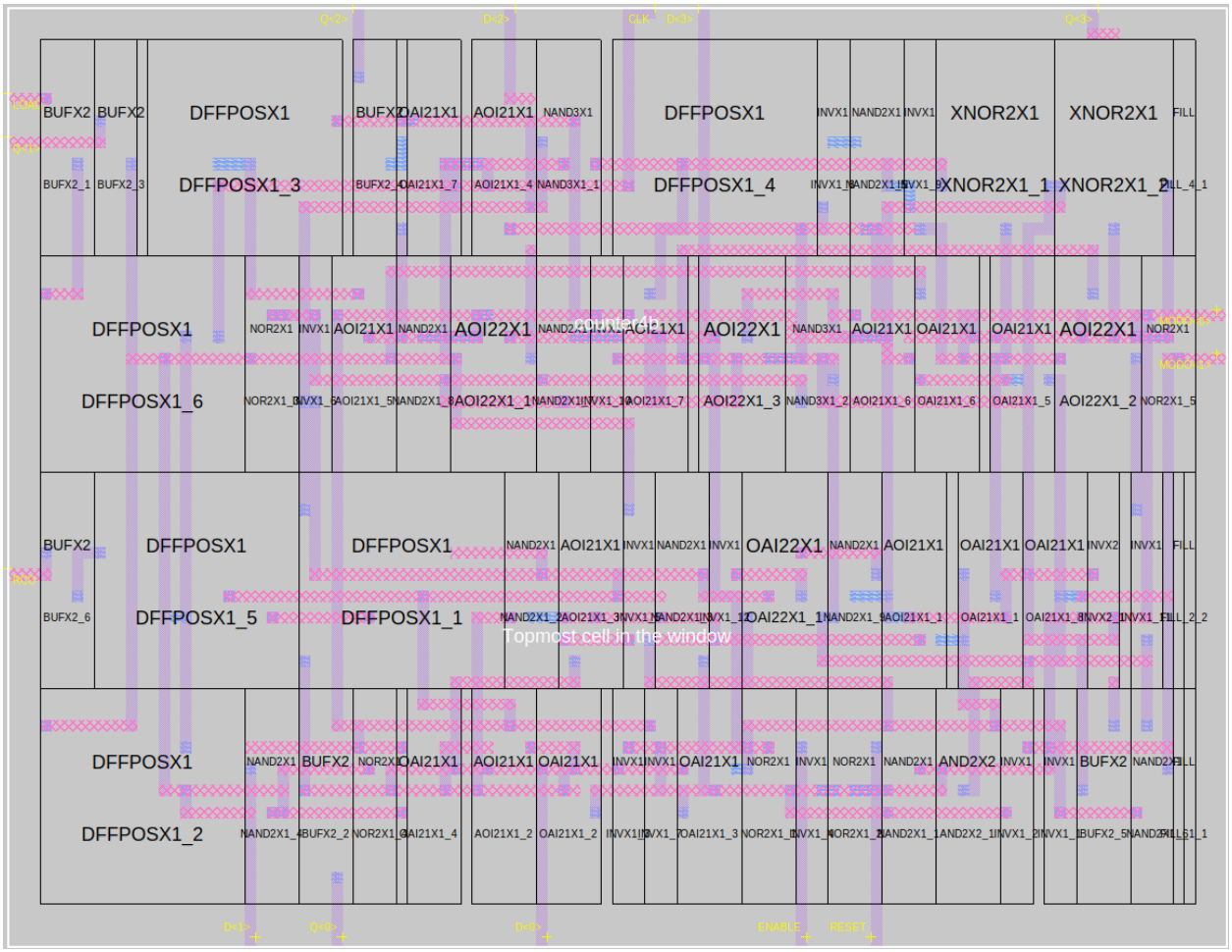


Figura 15: Layout generado del circuito *counter4b.v* utilizando la biblioteca de componentes osu035 de Qflow.

### 5.1.2. OSU050

Para los retardos máximos (descritos en el cuadro 11) se tiene, flop to flop: *path DFFPOSX1_1/CLK to DFFPOSX1_3/D*, pin to flop: *path input MODO[0] to DFFPOSX1_1/D*.

Para los retardos mínimos (descritos en el cuadro 11), flop to flop: *path DFFPOSX1_1/CLK to DFFPOSX1_1/D*, pin to flop: *pin CLK to DFFPOSX1_6/CLK*.

| Retardos | Máximo (ps) | Mínimo (ps) |
|---|---|---|
| Flop to flop | 1665,08 | 538,671 |
| Pin to flop | 1042,18 | 0 |

Cuadro 5: Retardos flop-to-flop y pin-to-flop del circuito *counter4b.v* utilizando la biblioteca de componentes *osu050*.

Los valores de frecuencia máxima y medidas del circuito *counter4b.v* se muestran en el cuadro 12, el *layout* del mismo circuito con la biblioteca *osu050* es el presentado en la figura 16.

| | |
|---|---|
| $f_{max}$ [MHz] | 600,573 |
| Medida horizontal [$\mu m$] | 173 |
| Medida vertical [$\mu m$] | 127 |
| Area [$nm^2$] | 21.971 |

Cuadro 6: Área y frecuencia máxima del circuito *counter4b.v* usando la biblioteca osu050.

En el cuadro 13 se presenta el conteo de compuertas usadas durante la síntesis.

| Compuerta | Cantidad |
|---|---|
| AND | 7 |
| AOI3 | 7 |
| AOI4 | 2 |
| NAND | 7 |
| NOR | 6 |
| NOT | 10 |
| OAI3 | 9 |
| OAI4 | 2 |
| OR | 11 |
| XNOR | 1 |
| XOR | 7 |

Cuadro 7: Compuertas generadas por Qflow durante el proceso de síntesis para el circuito *counter4b.v* usando la biblioteca *OSU050*.

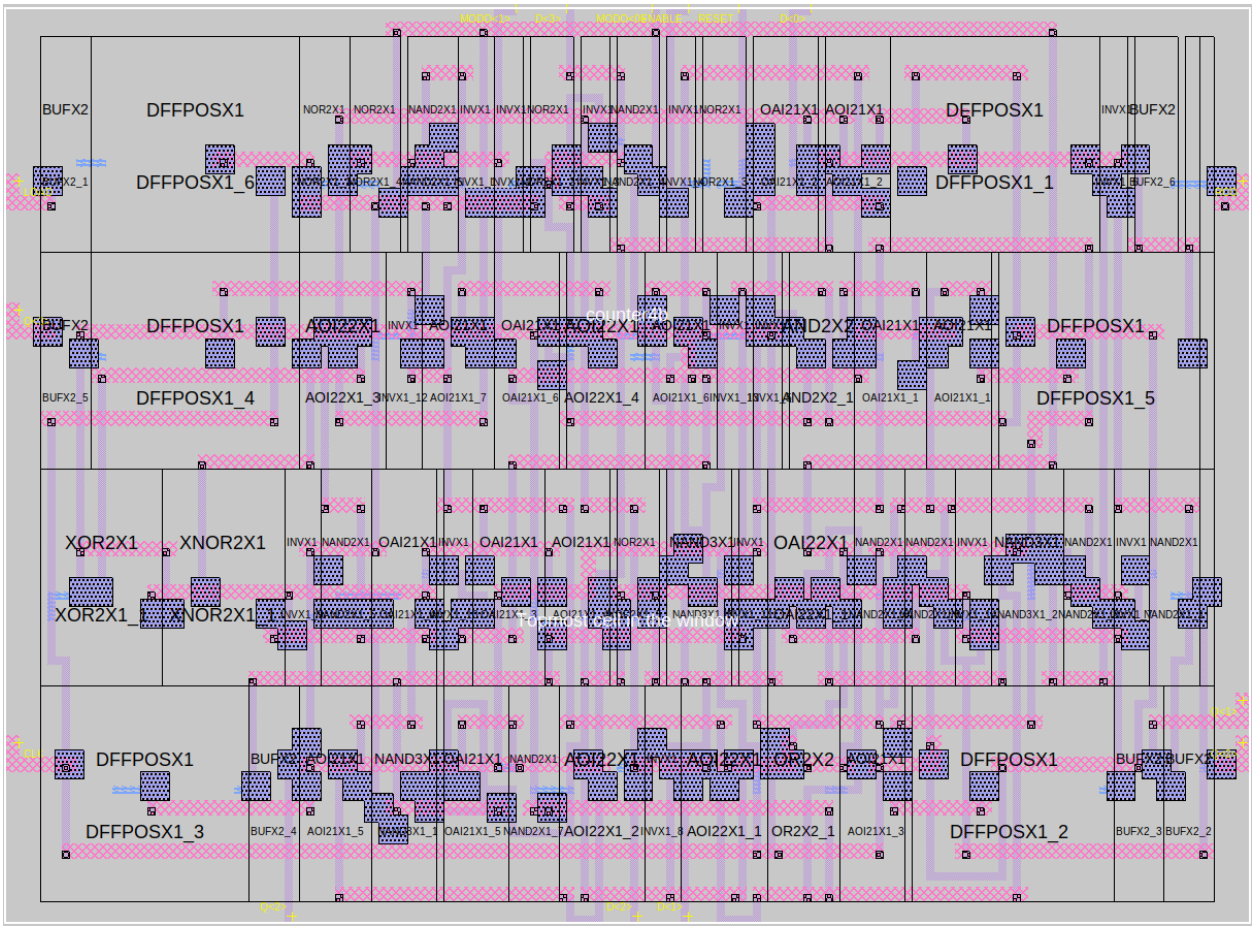


Figura 16: Layout generado del circuito *counter4b.v* utilizando la biblioteca de componentes osu050 de Qflow.

## 5.2. Contador de 32 bits

### 5.2.1. OSU035

Para los retardos máximos (descritos en el cuadro 11) se tiene, flop to flop: *path DFFPOSX1_1/CLK to DFFPOSX1_3/D*, pin to flop: *path input MODO[0] to DFFPOSX1_3/D*.

Para los retardos mínimos (descritos en el cuadro 11), flop to flop: *path DFFPOSX1_41/CLK to DFFPOSX1_1/D*, pin to flop: *pin CLK to DFFPOSX1_6/CLK*.

| Retardos | Máximo (ps) | Mínimo (ps) |
|---|---|---|
| Flop to flop | 1493,47 | 147,224 |
| Pin to flop | 873,233 | 0 |

Cuadro 8: Retardos flop-to-flop y pin-to-flop del circuito *counter32b.v* utilizando la biblioteca de componentes *osu035*.

Los valores de frecuencia máxima y medidas del circuito *counter32b.v* se muestran en el cuadro 12, el *layout* del mismo circuito con la biblioteca *osu035* es el presentado en la figura 15.

| | |
|---|---|
| $f_{max}$ [MHz] | 669,58 |
| Medida horizontal [$\mu m$] | 316 |
| Medida vertical [$\mu m$] | 227 |
| Area [$nm^2$] | 71.732 |

Cuadro 9: Área y frecuencia máxima del circuito *counter32b.v* usando la biblioteca osu035.

En el cuadro 13 se presenta el conteo de compuertas usadas durante la síntesis.

| Compuerta | Cantidad |
|---|---|
| AND2X2 | 1 |
| AOI21X1 | 7 |
| AOI22X1 | 3 |
| INVX1 | 13 |
| NAND2X1 | 9 |
| NAND3X1 | 2 |
| NOR2X1 | 5 |
| OAI21X1 | 8 |
| OAI22X2 | 1 |
| XNOR2X1 | 2 |

Cuadro 10: Compuertas generadas por Qflow durante el proceso de síntesis para el circuito *counter32b.v* usando la biblioteca *OSU035*.

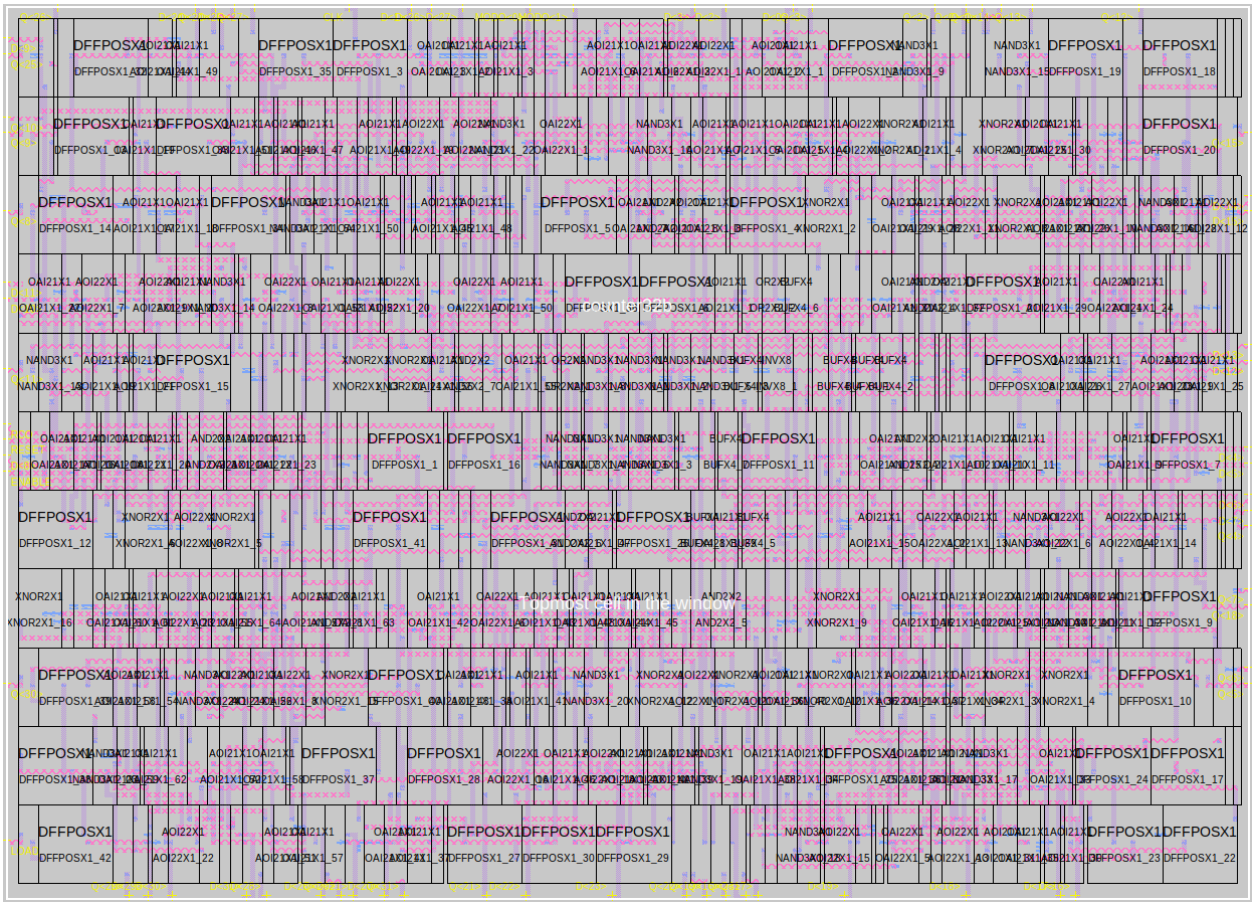

Figura 17: Layout generado del circuito *counter32b.v* utilizando la biblioteca de componentes osu035 de Qflow.

### 5.2.2. OSU050

Para los retardos máximos (descritos en el cuadro 11) se tiene, flop to flop: *path DFFPOSX1_37/CLK to DFFPOSX1_39/D*, pin to flop: *path input MODO[0] to DFFPOSX1_2/D*.

Para los retardos mínimos (descritos en el cuadro 11), flop to flop: *path DFFPOSX1_41/CLK to DFFPOSX1_1/D*, pin to flop: *pin CLK to DFFPOSX1_6/CLK*.

| Retardos | Máximo (ps) | Mínimo (ps) |
|---|---|---|
| Flop to flop | 1665,08 | 196,767 |
| Pin to flop | 968,339 | 0 |

Cuadro 11: Retardos flop-to-flop y pin-to-flop del circuito *counter32b.v* utilizando la biblioteca de componentes *osu050*.

Los valores de frecuencia máxima y medidas del circuito *counter32b.v* se muestran en el cuadro 12, el *layout* del mismo circuito con la biblioteca *osu050* es el presentado en la figura 16.

| | |
|---|---|
| $f_{max}$ [MHz] | 600,573 |
| Medida horizontal [$\mu m$] | 475 |
| Medida vertical [$\mu m$] | 336 |
| Area [$nm^2$] | 71.732 |

Cuadro 12: Área y frecuencia máxima del circuito *counter32b.v* usando la biblioteca osu050.

En el cuadro 13 se presenta el conteo de compuertas usadas durante la síntesis.

| Compuerta | Cantidad |
|---|---|
| AND2X2 | 1 |
| AOI21X1 | 7 |
| AOI22X1 | 4 |
| INVX1 | 14 |
| NAND2X1 | 8 |
| NAND3X1 | 3 |
| NOR2X1 | 5 |
| OAI21X1 | 6 |
| OAI22X2 | 1 |
| OR2X2 | 1 |
| XNOR2X1 | 1 |
| XOR2X1 | 1 |

Cuadro 13: Compuertas generadas por Qflow durante el proceso de síntesis para el circuito *counter32b.v* usando la biblioteca *OSU050*.

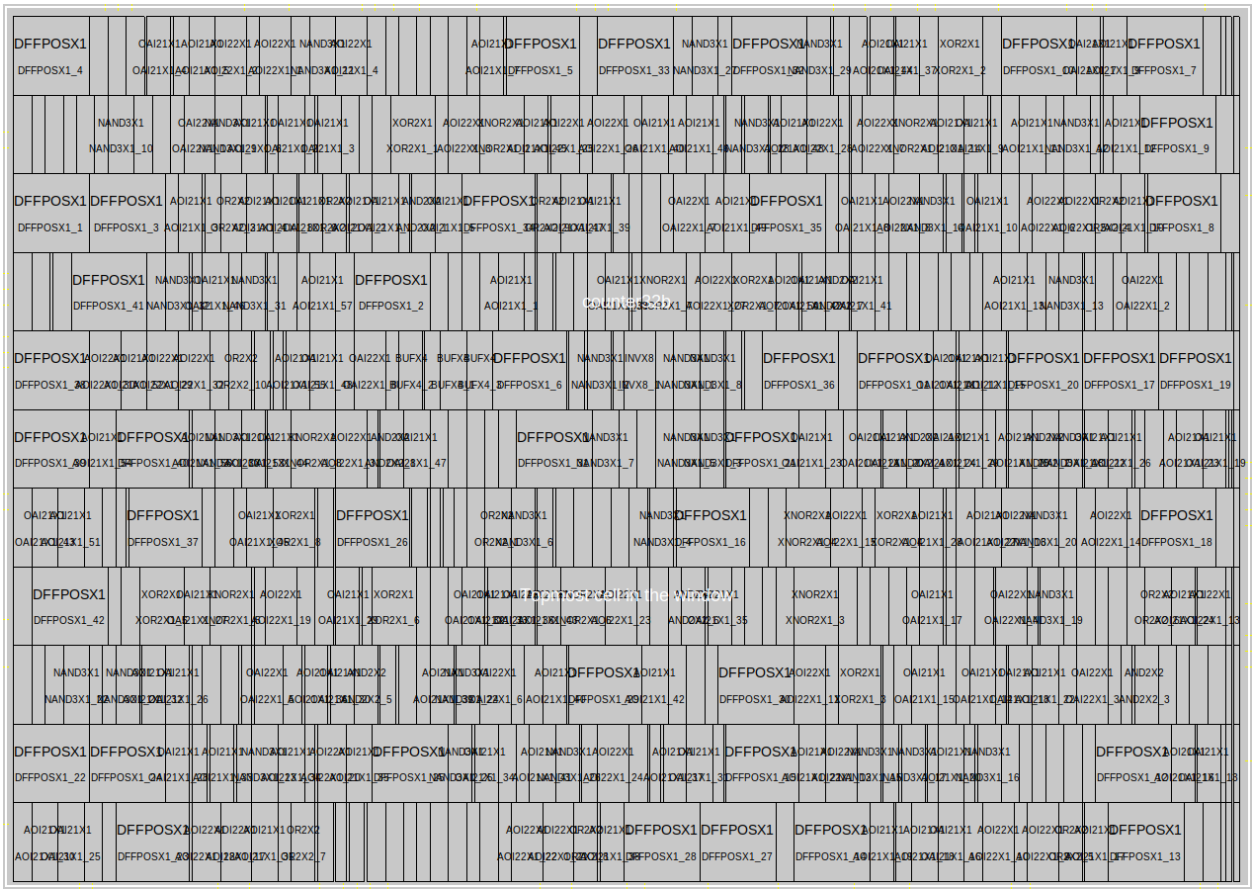

Figura 18: Layout generado del circuito *counter32b.v* utilizando la biblioteca de componentes osu050 de Qflow.

# 6. Parte E: Layout

## 6.1. NOR3

Se logró diseñar y simular correctamente una compuerta NOR de 3 entradas en el programa electric, en la figura 19 se muestra el layout generado por el programa, note que para generar este layout se siguió la tabla de verdad mostrada en la tabla 14.

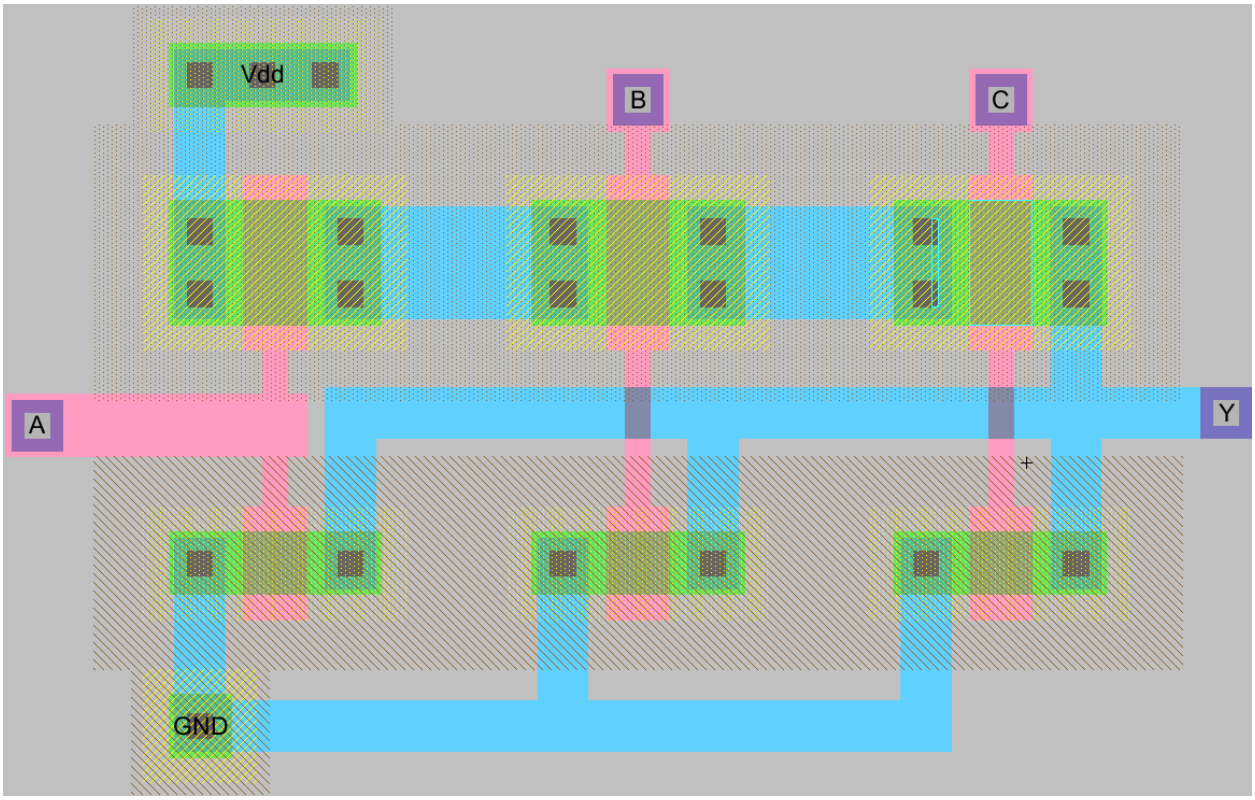


Figura 19: Layout generado del circuito *NOR de 3 entradas* en el programa electric vlsi.

| A | B | C | Y |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

Cuadro 14: Tabla de verdad de una compuerta NOR de 3 entradas.

En la figura 20 se muestra la simulación del layout realizado en LTSpice, puede notar que en el mismo está representada la tabla de verdad completa de la compuerta NOR.

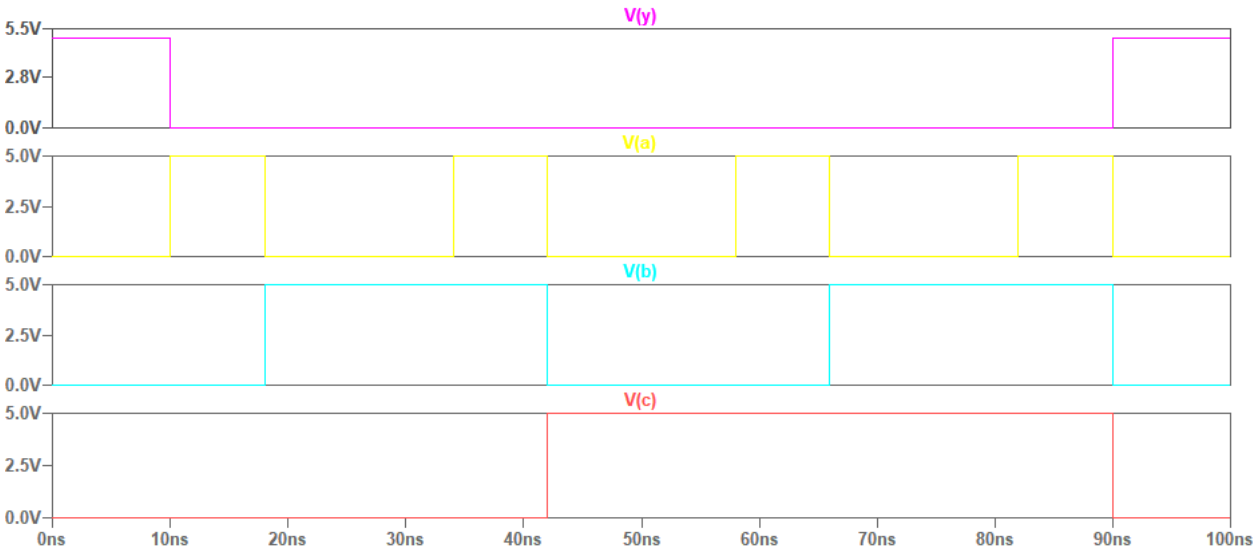


Figura 20: Diagrama de señales que representan la salida de la NOR de entradas en base a sus entradas, donde la morada corresponde a la salida a la salida y las demás a las entradas.

## 6.2. NAND3

De igual forma se diseño y simuló correctamente una compuerta NAND de 3 entradas en el programa electric, en la figura 21 se muestra el layout generado por el programa siguiendo como referencia la tabla de verdad de una NAND de 3 entradas (tabla 15).

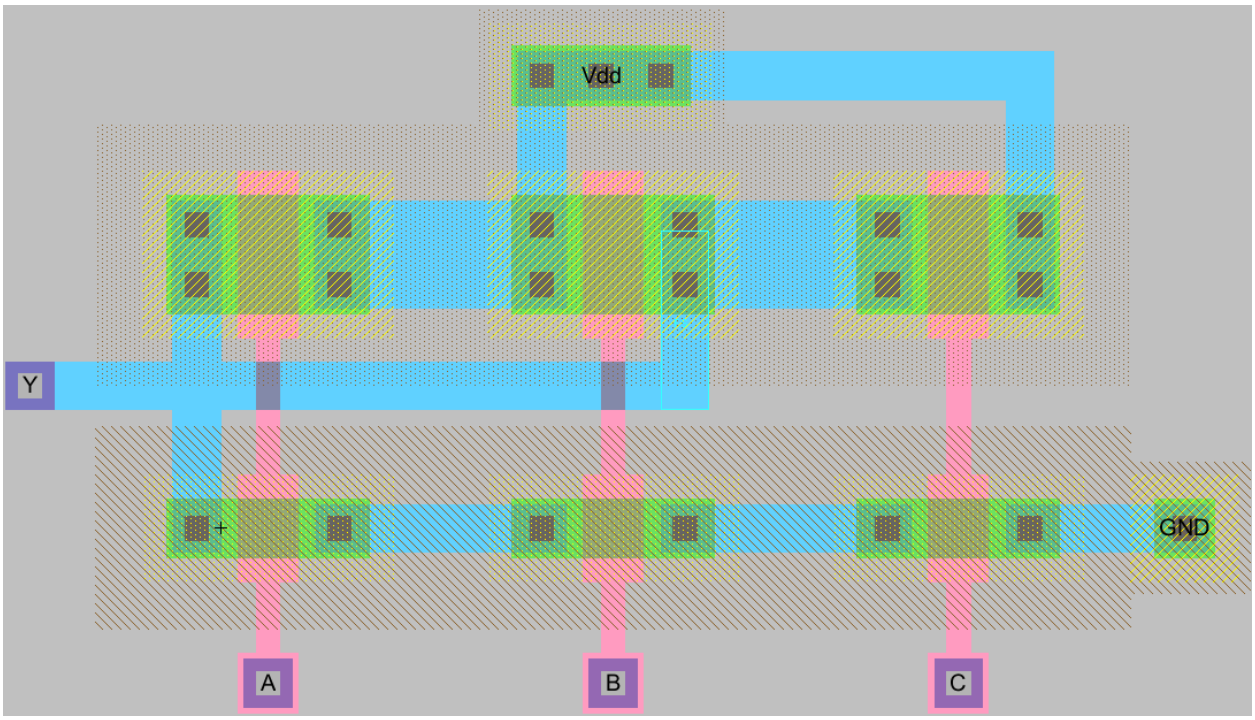


Figura 21: Layout generado del circuito *NAND de 3 entradas* en el programa electric vlsi.

| A | B | C | Y |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

Cuadro 15:

En la figura 22 se aprecia la simulación del layout en LTSpice, de igual forma que con la compuerta NOR en la simulación está representada toda la tabla de verdad para verificar si el circuito diseñado la cumple a cabalidad.

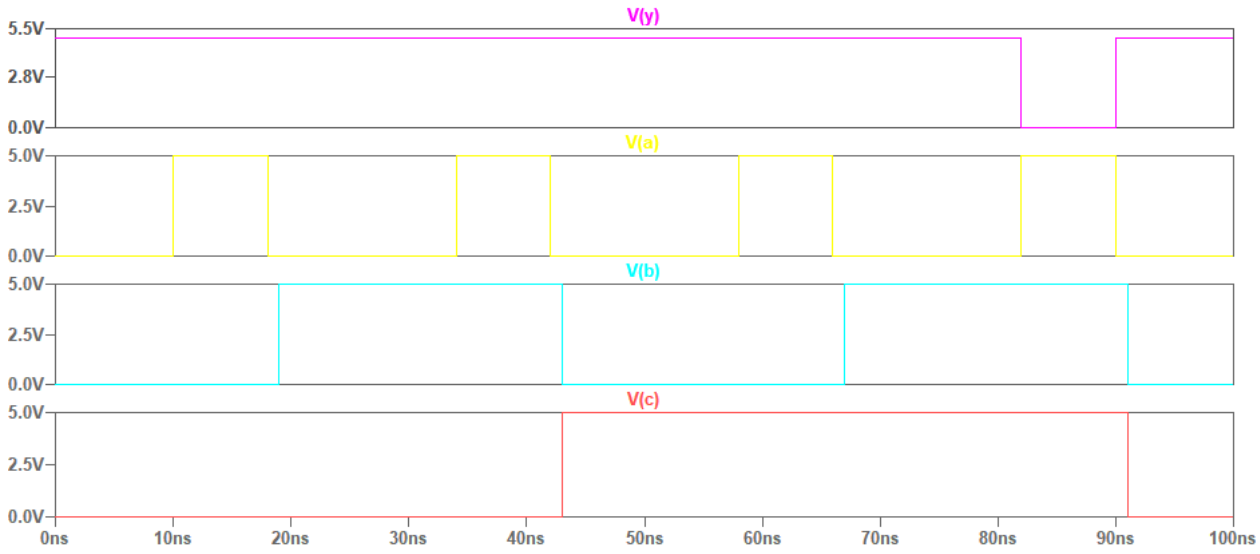


Figura 22: Diagrama de señales que representan la salida de la compuerta NAND de entradas en base a sus entradas, donde la morada corresponde a la salida a la salida y las demás a las entradas.

## 6.3. AOI22

El circuito AOI22 se muestra en la figura 21, este circuito es ligeramente más complejo que los otros en su cableado, y a pesar de que consiste en 2 etapas de compuertas su conteo de transistores es bastante reducido (esa es una de las razones por las que esta configuración es muy utilizada en el mercado). En la tabla 16 se muestra la tabla de verdad que sigue un circuito de estos.

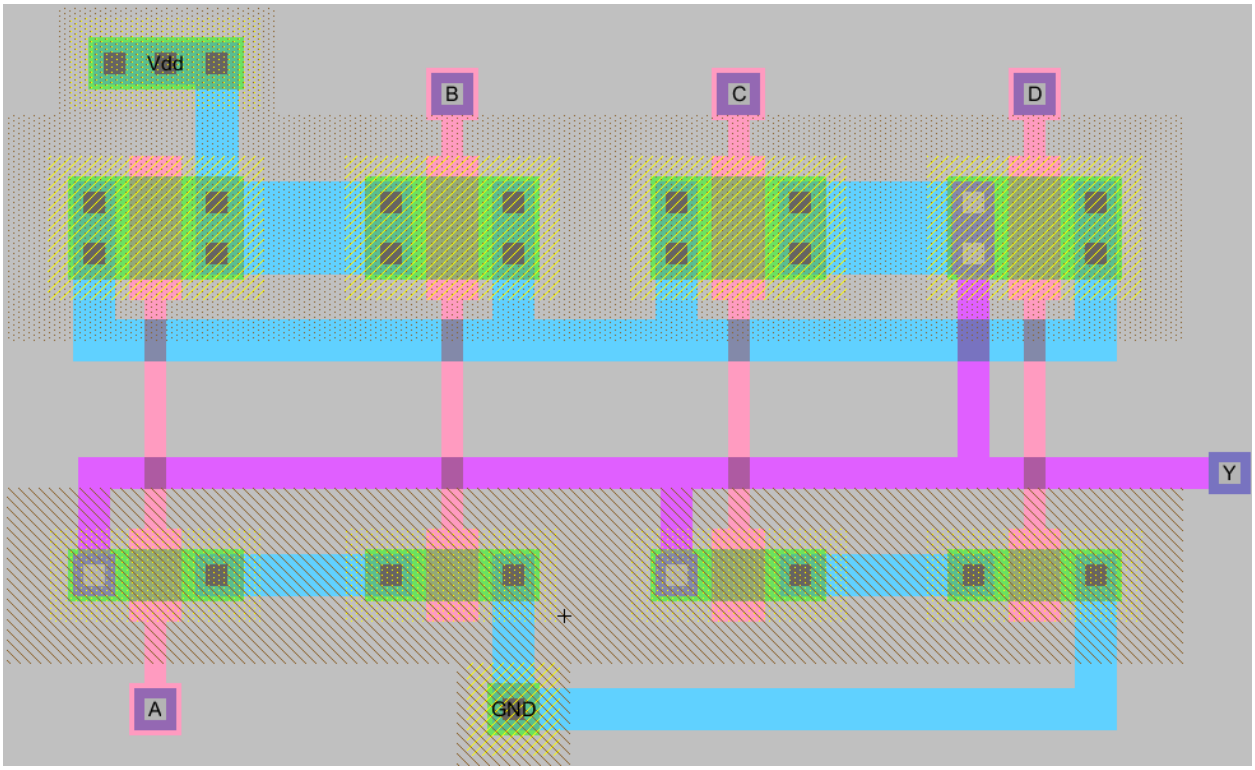


Figura 23: Layout generado del circuito *AOI22* generado en el programa electric.

| A | B | C | D | Y |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 |

Cuadro 16: Tabla de verdad de un circuito AOI22.

Para la simulación sólo se mostrarán 100ns debido a espacio (si desea verificar toda la tabla de verdad simule el circuito en LTspice), en la figura 24 puede notarse claramente que el comportamiento del circuito es el mismo que el representado en la tabla 16.

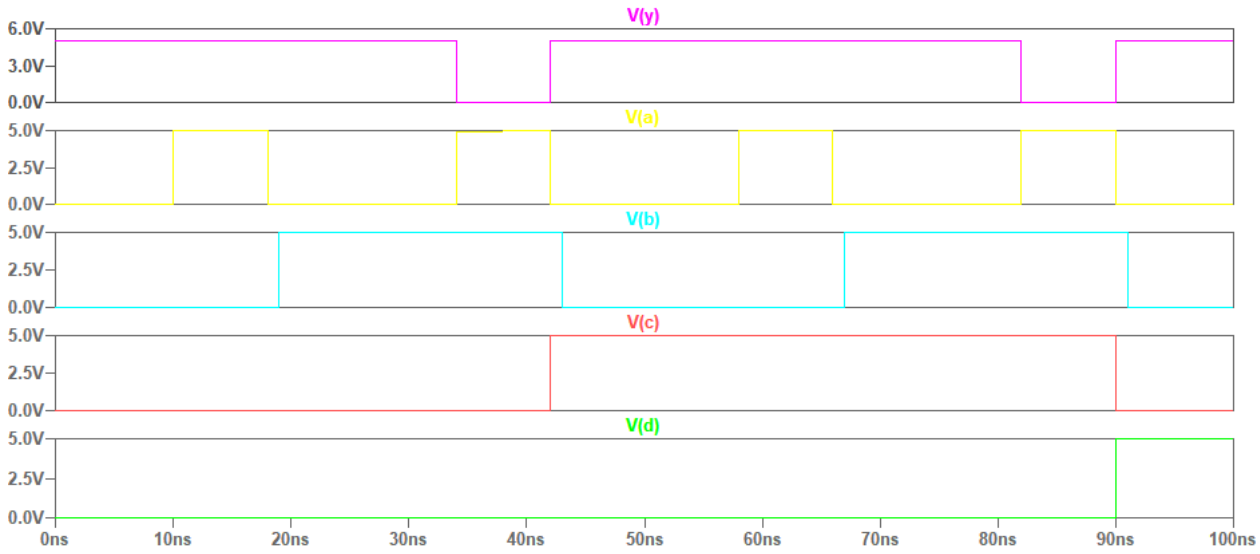


Figura 24: Simulación del circuito *AOI22* realizado en LTSpice, donde la señal morada corresponde a la señal de salida y las demás a las entradas.

# 7. Conclusiones

- Se pudo diseñar un contador de 32 al que no le afectará el tiempo de duración de la señal RCO de los contadores de 4 bits que lo componen.
- El tamaño de los semiconductores influye directamente en los tiempos de retardo de las compuertas lógicas.
- De igual forma la frecuencia del circuito se ve afectada con la tecnología que se implementa el mismo.
- La teoría estudiada en el curso microelectrónica sobre retardos fue fácilmente apreciable durante los resultados de las síntesis.
- Los circuitos diseñados con la biblioteca de componentes osu035 resultan ser más pequeños que con la biblioteca osu050, de igual forma tienen frecuencias más altas y retardos más pequeños.
- Se demostró que es posible llevar a cabo un ciclo de desarrollo completo utilizando herramientas de uso libre.
- Todas las simulaciones se llevaron a cabo satisfactoriamente y sin errores.

# 8. Recomendaciones

- Tener en cuenta la versión de *Qflow* que se tiene instalada en la máquina, ya que algunos comandos son ligeramente diferentes en base a la versión.
- Conocer las etapas seguidas por *Qflow* para sintetizar los circuitos ya que de esta forma se pueden entender claramente los mensajes desplegados en la terminal.
- Recordar que si no se especifica otra biblioteca de componentes *Qflow* utilizará de forma nativa la biblioteca *osu035*.
- Estar claro con que versión de electric se está trabajando, ya que algunas cosas son diferentes entre versiones y eso puede causar confusión.
- Al simular en Spice tener claro como funciona el generador de señales PULSE, ya que puede generar problemas si se quieren usar señales de distintas formas (cuadradas, triangulares, etc).

## Referencias

[1] TIM. (2019) Qflow tutorial. http://opencircuitdesign.com/qflow/.