UNIVERSIDAD DE COSTA RICA

ESCUELA DE INGENIERÍA ELÉCTRICA

Curso: Microelectrónica: Sistemas en Silicio
IE0411

Tarea 3 - Reporte de temporización: Contador

Estudiante
Brandon J. Esquivel Molina **B52571**
Profesor: Javier Pacheco Brito

Índice

Índice de figuras

Índice de cuadros

1. Introducción

Los circuitos integrados digitales CMOS VLSI actuales y muchos otros sistemas, realizados con tecnologías submicrométricas, alcanzan enormes velocidades de operación. En sus puertas lógicas, los retrasos de propagación se hacen cada vez más pequeños, a la vez que la enorme densidad de integración hace cada vez más complejos los caminos de interconexión. Las consecuencias de estos hechos son múltiples: los retrasos de los circuitos resultan comparables a los de los caminos de interconexión, en contradicción con las hipótesis clásicas; el sistema tiene cada vez menos tiempo para estabilizar su operación antes de que llegue un nuevo ciclo de reloj; aumentan los conflictos entre señales como rebotes, pulsos cortos, clock skew entre otros. El resultado global es que aumenta constantemente la importancia de los aspectos temporales frente a otros más clásicos, como la reducción del área.[8] En este reporte se muestran los resultados de un análisis de temporización propuesto desde un modulo especificado, donde se busca comprobar el efecto de los retardos en modelos estructurales sintetizados con librerías predefinidas. Se crea y sintetiza un módulo contador con 4 modos de operación para comparar su funcionamiento sin y con retardos, además de diferentes periodos de reloj. Se utiliza una escala de referencia en ns con una precisión en ps en la implementación del código.

2. Diseño bajo verificación (DUV) de temporización

Se diseña un contador sincrónico de 4 bits, con los siguientes modos de funcionamiento:

- Cuenta hacia arriba.
- Cuenta hacia abajo.
- Cuenta de tres en tres hacia arriba.
- Carga en paralelo.

2.1. Especificaciones del DUV

2.1.1. Entradas

- CLK: Entrada de reloj del contador. El flanco activo de la señal CLK es el flanco creciente. Entonces, con cada flanco positivo del reloj el contador cambia de estado dependiendo del estado de las señales de MODE y si la señal ENABLE = 1.
- ENABLE: Entrada de habilitación del contador. Si ENABLE = 1, el contador funciona normalmente respondiendo a los flancos activos de CLK para cambiar de estado de acuerdo a la señal MODO. Si ENABLE=0 y RESET = 0, el contador tendrá una salida de alta impedancia.
- RESET: Entrada que pone todas las salidas del contador en cero.
- D[3:0]: Entrada de datos D consta de 4 líneas. El valor que tengan las entradas D[3:0] será almacenado en Q[3:0] en el flanco activo de CLK si ENABLE = 1 y MODO = 11.
- MODO[1:0]: Entrada de modo que consta de dos líneas y sirve para definir cuál será el próximo estado del contador al llegar el flanco activo del reloj en la entrada CLK. Si el contador se encuentra en el estado Q antes del flanco activo del reloj, luego del flanco activo, su estado será:
  - MODO = 00 ..... Q + 3
  - MODO = 01 ..... Q - 1
  - MODO = 10 ..... Q + 1
  - MODO = 11 ..... D

2.1.2. Salidas

- Q[3:0] : Salida Q que consta de cuatro líneas que indican el estado presente del contador. El estado del contador cambia con el flanco activo de la señal CLK mientras ENB=1 y de acuerdo con el modo seleccionado con las líneas MODO[1:0].

- RCO : Salida de llevo “Ripple-Carry Out” que indica cuando el contador llega a su cuenta límite para que la siguiente etapa, en contadores de más de 4 bits, se habilite para que realice su actualización de estado. Note que RCO se pone en dependiendo de la señal MODO, además, debe permanecer en bajo durante el MODO = 11.

- LOAD : Salida que indica cuando el contador esta en modo de carga debe permanecer en bajo para el resto de los modos.

3. Metodología de pruebas/verificación

Se mantiene una metodología Híbrida con UVM(Universal Verification Methodology) y VMM(Verification Manual Methodology) con esquemas de ambiente de diseño incorporando Generador de estímulos (específicos y aleatorios), driver controlador de señales/ impulsos, checker, modulo conductual vs Estructural y monitoreo enfocado de señales. El diseño general sigue el siguiente diagrama:

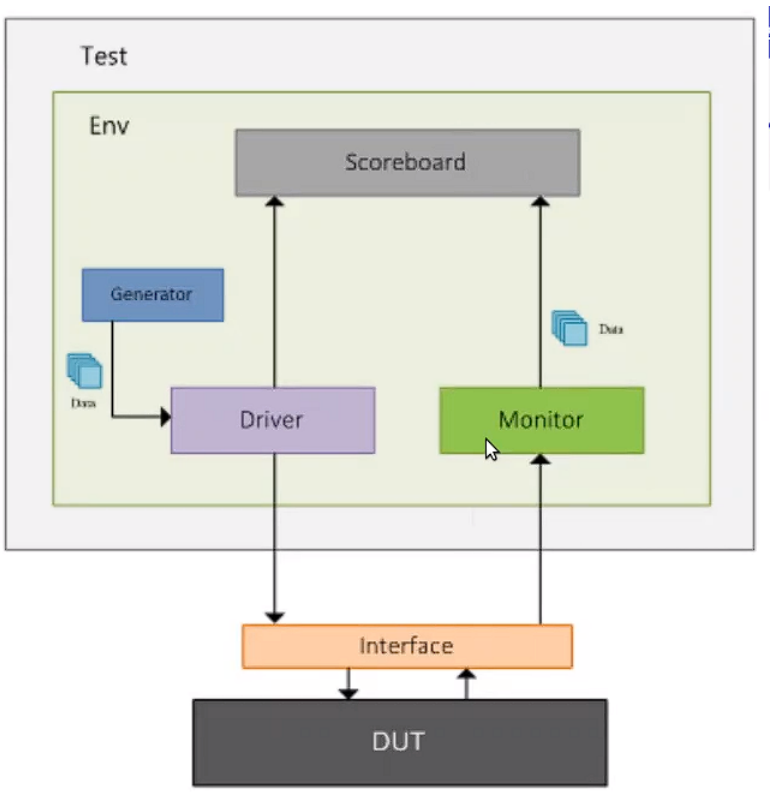

Figura 1: Ambiente de verificación. Tomado de ppt de clase

4. Mapeo de pruebas

Se diseñaron pruebas especificas y aleatorias con el fin de verificar todos los escenarios posibles, manteniendo controlabilidad en Verificación por jerarquía.

4.1. Driver.v Tasks

Se realizó una tarea que inicializa el contador con el modo de carga (11) y luego mantiene el modo 00 hasta que RCO se haga uno, contando así hasta el ultimo valor del contador, luego cambia MODO a 01 y realiza una cuenta regresiva desde 15 hasta cero. De nuevo se repite para recorrer todos los MODOS de los contadores y sus valores.

Se agregó una tarea randomizada para alcanzar valores aleatorios de MODO y D, con el fin de analizar los resultados en busca de comportamientos inadecuados. Finalmente se diseñaron pruebas individuales a cada MODO con cuenta normal y una prueba de RESET para verificar los casos extremos de timing.

4.2. modelo conductual del contador

El modulo contador diseñado se muestra a continuación:

```
*       Brandon Esquive Molina
        brandon.esquivel@ucr.ac.cr
   Counter module with 4-modes operation
*/

`ifndef COUNTER
`define COUNTER

`timescale 1 ns / 1 ps

module contador(
  // inputs
  input wire ENABLE, RESET, clk,
  input wire [3:0]  D,
  input wire [1:0]  MODO,
  //outputs
  output reg [3:0]    Q,
  output reg  RCO,  LOAD
);
reg [1:0] MODO_reg;
  // saving actual state of operation mode
  always@(*) begin
    MODO_reg =  MODO;
  end
  always @(posedge clk) begin
    if (RESET) begin
       Q <= 0;
       LOAD <= 0;
       RCO <= 0;
      end else begin
```

```
31          if(ENABLE) begin
32            case ({MODO})
33              2'b00: begin
34                if( Q == 15 ||  Q >=13 /*4'b1011*/ ) begin
35                   RCO <= 1'b1;
36                   Q <=  Q[3:0] + 4'b0011;
37                   LOAD <= 0;
38                end else begin
39                   Q <=  Q[3:0] + 4'b0011;
40                   LOAD <= 0;
41                   RCO <= 1'b0;
42                end
43              end
44              2'b01: begin
45                  if( Q == 4'b0000 ) begin
46                     RCO <=  1'b1;
47                     Q <=  Q[3:0] - 4'b0001;
48                     LOAD <= 0;
49                  end else begin
50                     Q <=  Q[3:0] - 4'b0001;
51                     LOAD <= 0;
52                     RCO <=  1'b0;
53                  end
54              end
55              2'b10: begin
56                  if( Q == 4'b1111 ) begin
57                     RCO <=  1'b1;
58                     Q <=  Q[3:0] + 4'b0001;
59                     LOAD <= 0;
60                  end else begin
61                     Q <=  Q[3:0] + 4'b0001;
62                     LOAD <= 0;
63                     RCO <=  1'b0;
64                  end
65              end
66              2'b11: begin
67                 RCO <= 0;
68                 Q <=  D;
69                 LOAD <= 1;
70              end
```

```
71              default: begin
72                 RCO <= 0;
73                 Q <= 0;
74                 LOAD <= 0;
75              end
76            endcase
77         end
78         else begin  // ENABLE == 0 & RESET == 0
79            Q <= 4'bzzzz;    //
                 ////////////////////////////////////////////////
                  H-I ZZZZZZZZZZZZ
80         end
81       end
82     end
83 endmodule
84
85 `endif
```

Todos los códigos se encuentran comentados en el repositorio del proyecto para consulta.

5. Resultados encontrados

Primeramente se sintetizó el el módulo conductual utilizando la herramienta YOSYS. Además se creó una librería de compuertas lógicas dentro del archivo cmoscells.v que incluye los siguientes módulos:

- Inversor
- NAND 2 entradas
- NOR 2 entradas
- NAND 3 entradas
- NOR 3 entradas
- DFF Flip Flop

Cada modulo mantiene un numero de parte, valores de retardo y una configuración especifica según su hoja del fabricante.

Las compuertas y sus retardos de propagación implementadas en la librería se resumen en la siguiente tabla:

GATE	tpd (ns)	thold (ns)	tsetup (ns)
NAND2	3.8		
NOR2	7.7		
NAND3	5.0		
NOR3	4.5		
BUF	3.5		
INV/NOT	6.4		
DFF	4.2	0.9	2.5

Cuadro 1: Tabla resumen de los valores de retardos por compuerta utilizados. Tomados de:[1],[2], [3],[4],[5],[6],[7].

Las hojas del fabricante seleccionadas se pueden ver en la carpeta docs del repositorio del proyecto. Las compuertas utilizadas en la síntesis por la librería elegida se muestran en la siguiente figura:

```
12.1.2. Re-integrating ABC results.
ABC RESULTS:              NAND cells:       34
ABC RESULTS:               NOR cells:       32
ABC RESULTS:               NOT cells:       15
ABC RESULTS:        internal signals:       98
ABC RESULTS:           input signals:       14
ABC RESULTS:          output signals:        6
Removing temp directory.
Removed 0 unused cells and 70 unused wires.
```

Figura 2: Resultados de síntesis: Elementos

Además, en el modulo sintetizado contadorsyn.v en la carpeta syn del proyecto se pueden apreciar las compuertas utilizadas (agregando los Flip Flops) y al ejecutar el programa se puede notar en la ventana de GTKWave que las compuertas no utilizadas son:

1. NAND 3 entradas (NAND3)

2. Buffer (BUF)

3. NOR 3 entradas (NOR3)

5.1. Síntesis ideal, sin retardos

Se sintetizó y simuló el diseño para comparar sus resultados sin retardos o tiempos de propagación (ideal) con un periodo de reloj de 2ns. El resultado fue el siguiente:

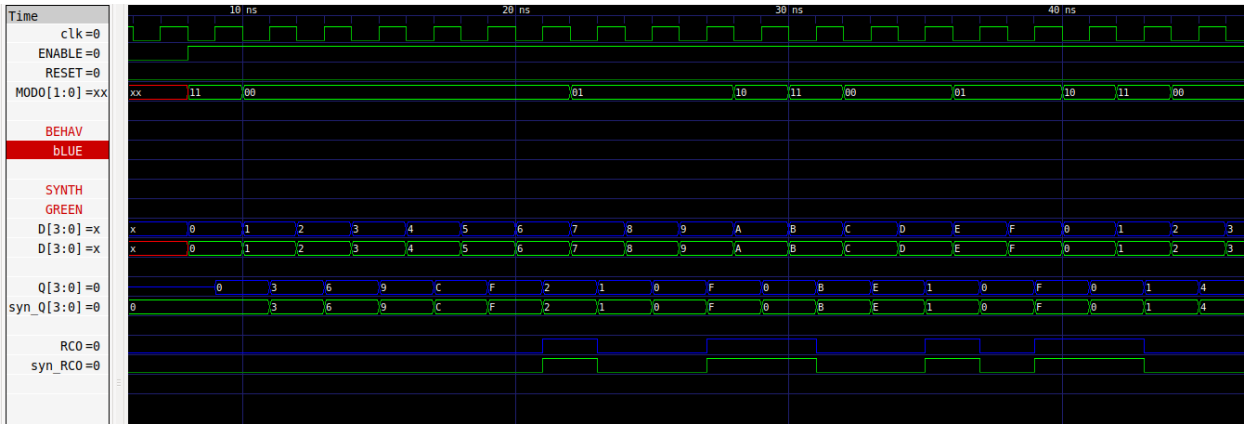

Figura 3: Resultados de simulación: esquema IDEAL (sin retardos). Periodo de 2ns.

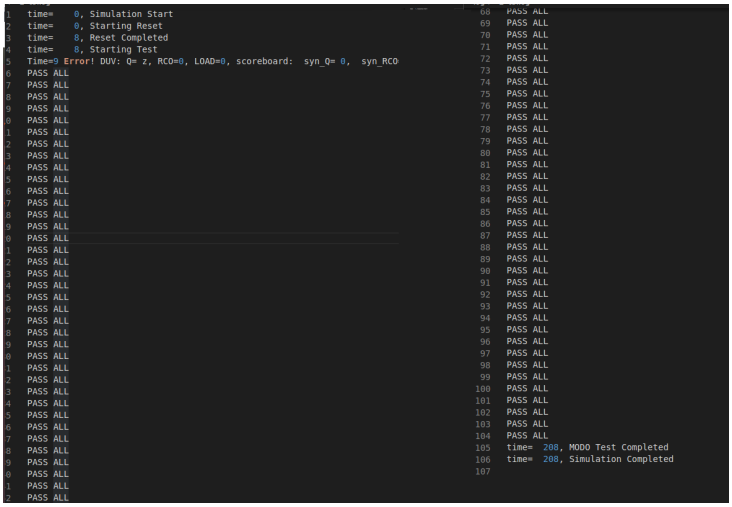

Figura 4: Resultados de simulación: log IDEAL

Se puede apreciar que el resultado es correcto y acertado, omitiendo el inicio del ciclo antes del enable y reseteo, que por defecto se setea a cero en el modelo estructural. Se aprecia que no se muestran problemas de temporización o violación de tiempos de propagación, como era de esperarse.

5.2. Síntesis con retardos, periodo de 2ns

Se agrega ahora la librería con retardos y se simula nuevamente el diseño estructural vs conductual. El resultado fue el siguiente:

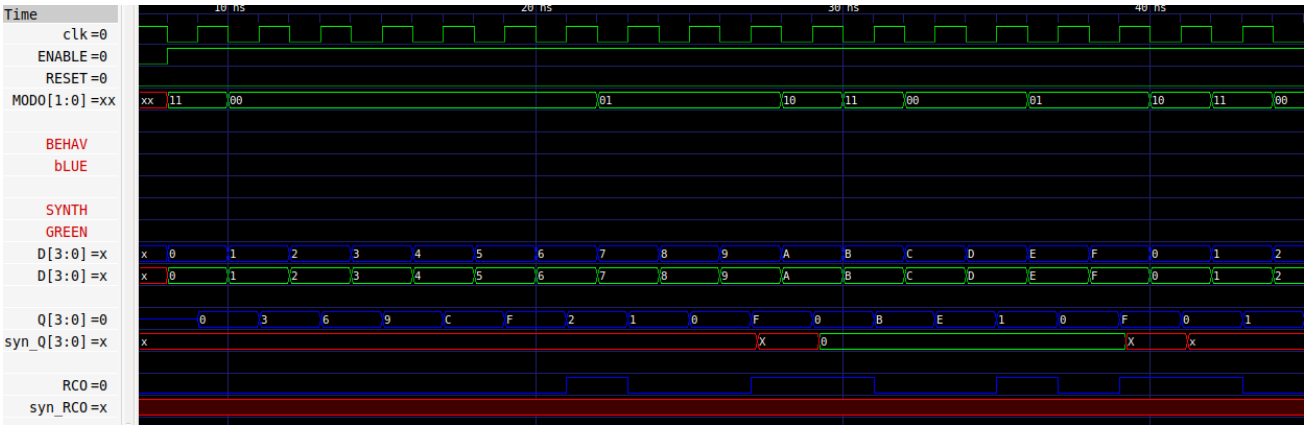

Figura 5: Resultado de simulación con retardos y periodo de 2ns

Se puede apreciar que ahora si se presentanb problemas de temporización y los valores se pierden entre las etapas, ya que el ciclo de reloj es muy rápido para los retardos y tiempos de propagación implementados. Se puede también consultar el log del checker:

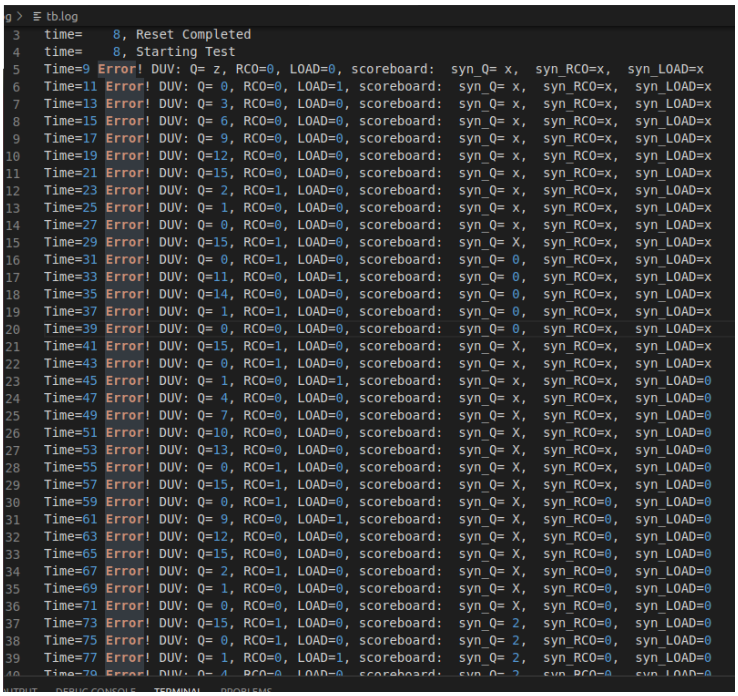

Figura 6: fragmento del archivo tb.log, resultado de la prueba con retardos y periodo de 2ns.

En el repositorio se encuentra la carpeta log donde encontrará el archivo tb.log que muestra los puntos de error en la simulación.

5.3. Síntesis con retardos, periodo de 4ns

Se duplica ahora el periodo de reloj para comprobar si se obtiene un comportamiento temporal adecuado, el cambio del ciclo y los resultados se puede observar en la siguiente figura:

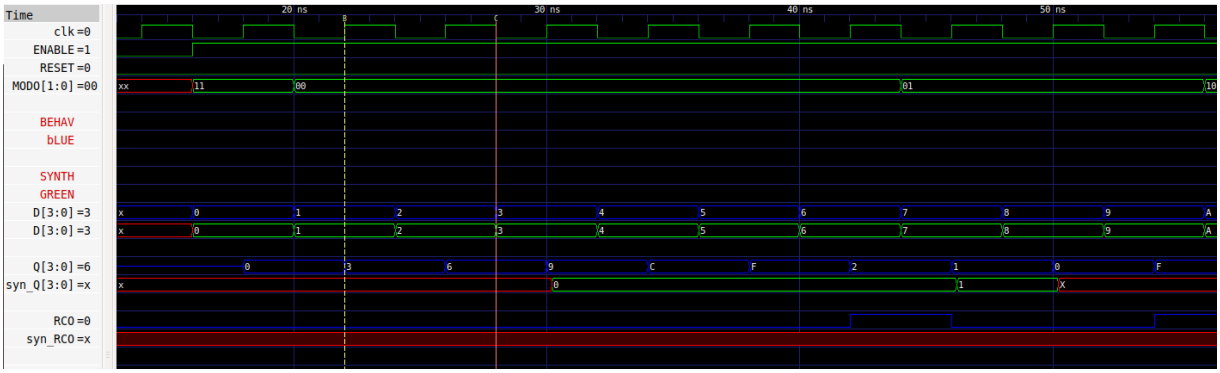

Figura 7: Resultado de simulación con retardos y periodo de 4ns

Se observa que los problemas de temporización se mantienen y las salidas no muestran el comportamiento adecuado por lo que se procede a aumentar el periodo al siguiente propuesto.

5.4. Síntesis con retardos, periodo de 20ns

Se configura ahora el periodo de reloj a 20ns para comprobar si se obtiene un comportamiento temporal adecuado, el cambio del ciclo y los resultados se puede observar en la siguiente figura:

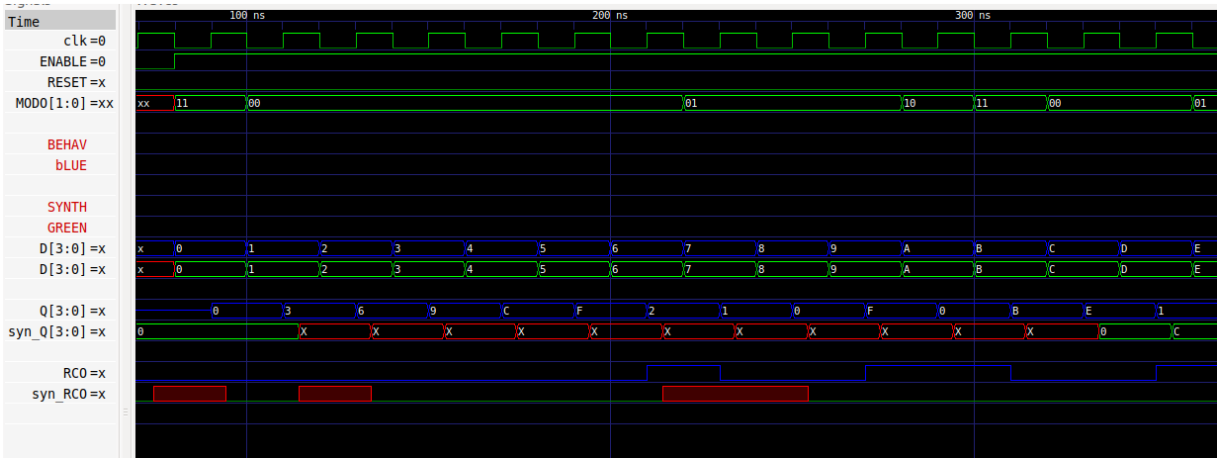

Figura 8: Resultado de simulación con retardos y periodo de 20ns

Se observa que los problemas de temporización se mantienen y las salidas

no muestran el comportamiento adecuado por lo que se procede a aumentar el periodo al siguiente propuesto.

5.5. Síntesis con retardos, periodo de 200ns

Se multiplica por diez ahora el periodo de reloj para comprobar si se obtiene un comportamiento temporal adecuado, el cambio del ciclo y los resultados se puede observar en la siguiente figura:

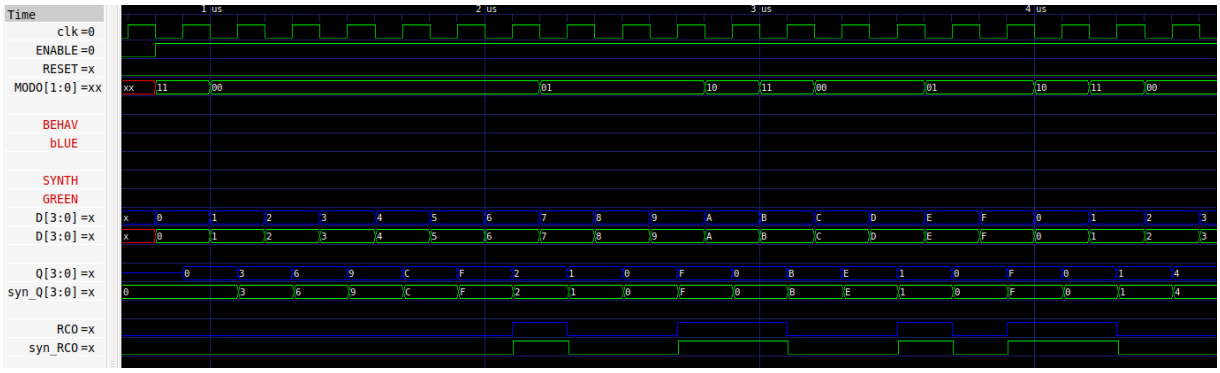

Figura 9: Resultado de simulación con retardos y periodo de 200ns

Se observa que los problemas de temporización no se mantienen y las salidas muestran el comportamiento adecuado, al igual que el comportamiento con el diseño ideal sin retardos, esto es el resultado deseado puesto que se utilizan muchas etapas de compuertas en los paths y esto genera un retardo de paths de orden mayor.

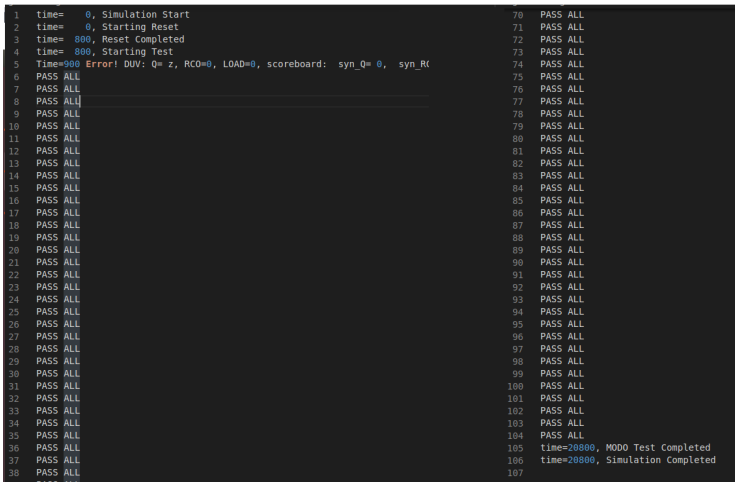

Figura 10: fragmento del archivo tb.log, resultado de la prueba con retardos y periodo de 200ns.

5.6. Diferencia de tiempos entre el modelo sin retardo y el modelo con retardo

En el siguiente zoom de una imagen de simulación se puede apreciar la diferencia de tiempos de propagación entre ambos modelos:

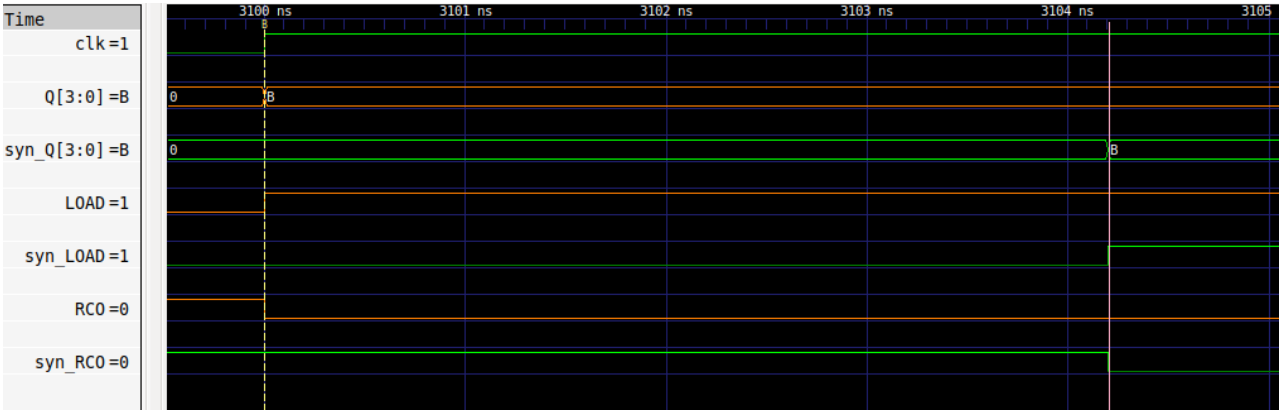

Figura 11: Comparación de modelo sin y con retardo para las salidas LOAD, RCO y Q. Periodo de 200ns

De los marcadores se puede apreciar que para estas salidas la diferen-

cia de tiempo es de 4.2ns en cada una, relación que se mantiene siempre para cada flanco de reloj. Esto concuerda adecuadamente con los retardos de propagación agregados al modulo secuencial para sincronizar: el Flip flop (DFF).

```
145  //  Using the maximun values for all parameters, ma
146  module DFF(C, D, Q);
147  specify
148      specparam tpd = 4.2;      You, 3 hours ago • sy
149      specparam thold = 0.9;
150      specparam tsetup = 2.5;
151      (C,D => Q) = (tpd, tpd);   // tRise,tFall
152      $setup(D, posedge C, tsetup);
153      $hold(posedge C, D, thold);
154  endspecify
155
156  input C, D;
```

Figura 12: Retardo configurado para módulo FF en la librería de celdas.

6. Enlace del repositorio

`https://github.com/brandonEsquivel/Paths_Timing_Verification`

6.1. Observaciones y Recomendaciones

Se fue comentando cuales complicaciones de temporización se fueron mostrando en cada configuración, también se comentó en resumen las celdas utilizadas y las sobrantes en el diseño sintetizado, por último se agregó un análisis de la diferencia de tiempo del modelo con retardo y del modelo sin retardo. Se nota entonces la importancia de la temporización de circuitos digitales y módulos, pues su análisis es fundamental para el correcto funcionamiento de los sistemas. El conocimiento claro y conciso del proceso de síntesis ayudar a mejorar la capacidad de análisis de los resultados y la solución de posibles errores presentes en la simulación y compilación. El uso adecuado de la herramienta YOSYS facilita la síntesis y comprender adecuadamente los script automatiza los procesos y ahorra tiempo de proyecto.

Referencias

[1] T. INSTRUMENTS., *Datasheet: Single Buffer Gate.* SN74LVC1G34, 2016.

[2] T. INSTRUMENTS., *Datasheet: Single Positive-Edge-Triggered D-Type Flip-Flop.* SN74LVC1G80, 2016.

[3] T. INSTRUMENTS., *Datasheet: Single Inverter Gate.* SN74LVC1G04, 2014.

[4] T. INSTRUMENTS., *Datasheet: Single 2-Input Positive-NAND Gate.* SN74LVC1G00, 2014.

[5] NEXPERIA., *Datasheet: Single 3-input NAND gate.* 74LVC1G10GF, 2016.

[6] T. INSTRUMENTS., *Datasheet: Single 2 Input Positive Nor Gate.* SN74AHC1G02-EP, 2008.

[7] T. INSTRUMENTS., *Datasheet: Single 3-Input Positive-NOR Gate.* SN74LVC1G27, 2013.

[8] A. Acosta and A. Jiménez, *Temporización en Circuitos Integrados Digitales CMOS.* ACCESO RÁPIDO, Marcombo, S.A., 2000.