Universidad de Costa Rica

Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
IE0523 - Circuitos Integrados Digitales
II ciclo de 2024

Tarea #2:
Descripción Estructural Controlador

Profesor:
Enrique Coen Alfaro

Estudiante:
Gabriel Siles Chaves - C17530

Grupo 01

22 de setiembre 2024

Índice

1. Resumen

En este proyecto se continúa en la elaboración del la descripción conductual del controlador realizado anteriormente por lo que a continuación se realiza una descripción estructural del mismo con la herramienta *yosys*. Al realizar un síntesis de alto nivel se tienen diferentes etapas la cual se realiza la estructura general. Se tiene la descripción estructural genérica (RTLIL) que no utiliza componentes de la librería interna del sintetizador. En este proyecto se realizará una verificación completa de la síntesis con las pruebas diseñadas anteriormente, realizando el proceso de síntesis paso por paso , observando el comportamiento de la descripción estructural.

Par esto se realizará paso por paso la aplicación de módulos como techmap, dfflibmap, abc entre otras. Se aplicará este proceso para aplicar compuertas únicamente NAND, NOR, NOT y FLIPFLOPS para realizar el diseño estructural ya sea con retardo y sin retardo. Este proyecto requiere un análisis completo para poder aplicar los tiempos necesarios para que toda la descripción funcione correctamente por lo que se podrá observar a continuación todo el desarrollo del diseño estructural.

2. Descripción Arquitectónica

Para la descripción arquitectónica se explicará la estructura que la herramienta de Yosys ejecuta y construye a la hora de realizar el proceso de síntesis, en donde se desarrolla 2 principales partes, la primera la descripción estructural genérica (rtlil) posteriormente la completa con las compuertas lógicas NAND, NOR, NOT y los componentes de flip flops ya sea sin o con retardo.

2.1. Descripción RTLIL

A la hora de realizar el proceso de síntesis se debe tener en consideración que se ajusta a las necesidades del diseñador, en este caso se ejecutan diferentes comandos para obtener únicamente la esta descripción por lo que será necesario repasar los elementos utilizados por yosys a la hora de generar esta descripción estructural. En la Figura 1 se puede observar los comandos para el RTLIL.

```
read_verilog controlador.v
hierarchy -check -top controlador
proc; opt; fsm; opt; memory; opt;
techmap; opt
show; write_verilog cntrl_rtlil.v
```

Figura 1: Código yosys para RTLIL]

Primero se realiza la lectura del archivo *controlador.v* con la descripción conductual, para poderle aplicar procesos que son los siguientes:

- **hierarchy -check -top controlador**: verifica y ajusta la jerarquía del diseño, verificando que todo esté instanciado correctamente generando un módulo de nivel superior llamado controlador.

- **proc**: Se encarga de realizar los procesos del código Verilog en lógica combinacional y secuencial, de manera que tenga la estructura para las compuertas para las puertas lógicas
- **fsm**: Optimiza la máquina de estados, simplificando la implementación
- **memory**: Cualquier elemento de construcción de memoria en componentes básicos para sintetizarlo.
- **techmap**: Se genera un mapeo al diseño aplicando componentes de celdas como FF y compuertas lógicas
- **opt**: Este comando se encarga de realizar la optimización de cada paso al diseño, es importante aplicarlo para la síntesis
- **show**: Muestra un diagrama en un archivo .dot con toda la ruta que se generó en la síntesis.

Una vez aplicado todos los pasos se escribe un archivo .v *write_verilog cntrl_rtlil.v* el cual posteriormente se le aplica el tester para observar el comportamiento en el gtkwave. En la Figura 2 se puede visualizar el diagrama del esquemático del RTLIL.

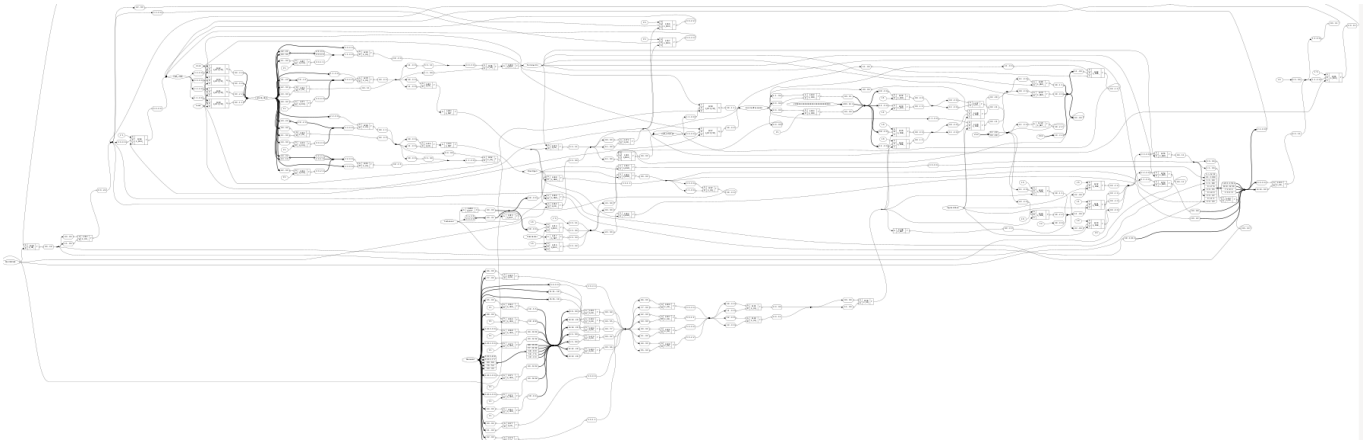

Figura 2: Esquemático RTLIL

Al ejecutar se puede observar todo el esquemático y en la terminal las acciones que se realizó. Si se quiere visualizar todo lo que se aplicó se puede correr el código como se indica en *Instrucciones*. Se generaron un total de 17 Muxes los cuales posteriormente al generar las compuertas se cambia el diseño para solo utilizar NANDs, NOTs, NORs y Flip Flops.

2.2. Diseño estructural

Es importante tener en cuenta que se eliminó la clave por default, realizando cambios en el controlador debido que en la primer Tarea no se aplicó correctamente las salidas para ser mapeadas como celdas de Flip Flops.

De igual manera como la descripción estructural del RTLIL tiene los mismo comandos únicamente que se agregan los siguientes:

- **dfflibmap -liberty ./cmos_cells.lib**: Se realiza un mapeo de flipflops en el diseño, con respecto a la biblioteca que se adjunta.

- **abc - liberty ./cmos_cells.lib**: Se aplica la síntesis lógica y optimización, convirtiendo diseño en términos de celdas lógicas con respecto a la librería cmos_cells.lib
- **clean**: Limpia cualquier dato qu ese haya generado durante las etapas mencionadas,no necesarias para la síntesis.

Producto de este se genera el esquemático de todas las compuertas y flip flops. En este al aplicar *dfflibmap* se generaron 6 Flip flops. Y al aplicarl *abc* en total se integraron 34 NANDS, 25 NORs, 11 NOTs, 77 señales internas, 25 señales de entrada y 9 señales de salida. Se debe tener en cuenta que posteriormente cada una de estas compuertas se les aplica un retardo. Todo esto se podrá observar al ejecutar la sintesis omo se explica en *Instrucciones*.

2.3. Retardos

Para la aplicación de los retardos a las compuertas lógicas se generó un archivo llamado delayed_cmos_cells.v en donde se agregan unidades de tiempo que retardan las salidas para poder simular lo que sucedería en la realidad. En la Figura 3 y Figura 4 respectivamente.

```
module BUF(A, Y);
  input A;
  output Y;
  assign Y = A;
endmodule

module NOT(A, Y);
  input A;
  output Y;
  assign Y = ~A;
endmodule

module NAND(A, B, Y);
  input A, B;
  output Y;
  assign Y = ~(A & B);
endmodule

module NOR(A, B, Y);
  input A, B;
  output Y;
  assign Y = ~(A | B);
endmodule

module DFF(C, D, Q);
  input C, D;
  output reg Q;
  always @(posedge C) begin
          Q <= D;
  end
endmodule
```

Figura 3: cmos_cells.v sin retardos

```
module BUF(A, Y);
  input A;
  output Y;
  assign #1 Y = A;
endmodule

module NOT(A, Y);
  input A;
  output Y;
  assign #1 Y = ~A;
endmodule

module NAND(A, B, Y);
  input A, B;
  output Y;
  assign #2 Y = ~(A & B);
endmodule

module NOR(A, B, Y);
  input A, B;
  output Y;
  assign #2 Y = ~(A | B);
endmodule

module DFF(C, D, Q);
  input C, D;
  output reg Q;
  always @(posedge C) begin
          #4 Q <= D;
  end
endmodule
```

Figura 4: delayed_cmos_cells.v con retardos

En la siguiente tabla se puede visualizar todos los componentes con sus retardos y cantidades:

Componente	Retardo	Cantidad
NAND	#2	34
NOT	#1	77
NOR	#2	11
DFF	#4	6

Tabla 1: Componentes con características del diseño

3. Plan de Pruebas

En esta sección se presenta la lista detallada de las pruebas de cada estado y corroborar el funcionamiento del sistema de acuerdo a las especificaciones.

3.1. Prueba para diseño RTLIL y descripción estructural sin retardo

A la hora de aplicar las pruebas se utiliza *tester.v* para aplicar las entradas y observar el comportamiento en el GTKwwave, en este tester a diferencia de la tarea anterior en vez de que la entrada dure 5 unidades de tiempo y luego se apague se realiza en 15 unidades de tiempo para que se genere una lectura de las entradas de manera correcta. A continuación se observan las pruebas que se aplican para la parte estructural del RTLIL y el diseño estructural sin retardo. Se aplica a ambos debido a que tanto como el RTLIL no cambia mucho con respecto al conductual, el estructural no se tiene en cuenta los retardos, por lo que se espera un comportamiento correcto del sistema. Ahora para las pruebas:

```
        initial begin
        reset = 0;
        clock = 0;
        CarS = 0;
        PassS = 0;
        Passw = 16'h0000;
        IngresoC = 0;
        #5 reset = 1;
        #5 reset = 0;

// Caso #01 Funcionamiento Normal Basico
        #15 CarS = 1;           // Carro detectado
        #15 CarS = 0;           // Carro no detectado
        #15 Passw = 16'h7530;   // Ingreso Password 7530
        IngresoC = 1;           // Se ingreso Password
        #15 IngresoC = 0;       // Se apaga Ingreso Password
        #15 PassS = 1;          // Se activa sensor de paso
        #15 PassS = 0;          // Se desactiva sensor de paso
        #20

// Caso #02 Ingreso de pin incorrecto menos de 3 veces
        #5 CarS = 1;            // Carro detectado
        #15 CarS = 0;           // Carro no detectado
        #15 Passw = 16'h7531;   // Ingreso Password 7531
        IngresoC = 1;           // Se ingresa Password
        #15 IngresoC = 0;       // Se apaga Ingreso Password
        #5 Passw = 16'h7534;    // Ingreso Password 7534
        IngresoC = 1;           // Se ingresa Password
        #15 IngresoC = 0;       // Se apaga Ingreso Password
        #5 Passw = 16'h7530;    // Ingreso Password 7530
        IngresoC = 1;           // Se INgresa Password
        #15 IngresoC = 0;       // Se apaga Ingreso Password
        #15 PassS = 1;          // Se activa sensor de Paso
        #15 PassS = 0;          // Se desactiva sensor de paso
        #20
```

Figura 5: Probador casos 1 y 2 para RTLIL y estructural sin retardo

```
// Caso #03 Ingreso de pin incorrecto 3 o mas veces
        #5 CarS = 1;            // Carro detectado
        #15 CarS = 0;           // Carro no detectado
        #5 Passw = 16'h2024;    // Ingreso Password 2024
        IngresoC = 1;           // Se ingresa Password
        #15 IngresoC = 0;       // Se apaga Ingreso Password
        #5 Passw = 16'h0509;    // Ingreso Password 0509
        IngresoC = 1;           // Se ingresa Password
        #15 IngresoC = 0;       // Se apaga Ingreso Password
        #5 Passw = 16'h1979;    // Ingreso Password
        IngresoC = 1;           // Se ingresa Password
        #15 IngresoC = 0;       // Se apaga ingreso Password
        #15 reset = 1;          // Se activa reset
        #15 reset = 0;          // Se desactiva reset
        #20
// Caso #04 Alarma de Bloqueo
        #10 CarS = 1;           // Carro detectado
        #15 CarS = 0;           // Carro no detectado
        #5 Passw = 16'h7531;    // Ingreso Password 7531
        IngresoC = 1;           // Se Ingresa Password
        #15 IngresoC = 0;       // Se apaga Ingreso Password
        #5 Passw = 16'h7530;    // Ingreso Password 7530
        IngresoC = 1;           // Se Ingresa Password
        #15 IngresoC = 0;       // Se apaga Ingreso Password
        #15 PassS = 1;          // Se activa sensor de paso
        CarS = 1;               // Carro detectado
        #15 PassS = 0;          // Se apaga Sensor de paso
        CarS = 0;               // Carro no detectado
        #20 reset = 1;          // Se activa reset
        #10 reset = 0;          // Se apaga reset
        #70 $finish;            // Se termina pruebas

        end

        always begin
                #5 clock = !clock;
```

Figura 6: Probador casos 3 y 4 para RTLIL y estructural sin retardo

Como se indicó todas las entradas se definen como 0 para poder así definirlas, se puede visualizar en la Figura 5 como se genera un reset 5 unidades de tiempo para iniciar sistema desde 0. Se aplica

el funcionamiento Normal Básico donde llega el carro, el sensor lo detecta, se ingresa la contraseña correcta y el carro pasa. Como mencionado anteriormente se agregó mas tiempos de subida a las entradas para tener una lectura eficaz.Para el Caso 2 la persona ingresa 2 veces incorrectamente la contraseña generando que el contador de intentos aumente y una vez que se ingresa correctamente se abra la puerta.

Para los casos de Ingreso de pin 3 veces incorrecto y alarma de bloqueo en el momento en el que un carro esté pasando y se detecto un en la entrada en la Figura 8 se muestra los tiempos de cada entrada y su duración

3.2. Descripción estructural con retardo

Para el caso del diseño estructural con retardo se debe utilizar un tester.v diferente al que se utilizó en el diseño conductual, RTLIL y el diseño estructural sin retardo debido a que a la hora de ingresar los retardos con el *delayed_ cmos_ cells.v* el controlador se desincroniza y genera errores. Por lo que se cambió el clock a 12 unidades de tiempo y se aplicó las siguientes pruebas:

```
        initial begin
        reset = 0;
        clock = 0;
        CarS = 0;
        PassS = 0;
        Passw = 16'h0000;
        IngresoC = 0;
        #12 reset = 1;
        #36 reset = 0;

// Caso #01 Funcionamiento Normal Basico
        #12 CarS = 1;           // Carro detectado
        #36 CarS = 0;
        #12 Passw = 16'h7530;   // Ingreso Password 7530
        IngresoC = 1;           // Se ingreso Password
        #36 IngresoC = 0;       // Se apaga Ingreso Password
        #36 PassS = 1;          // Se activa sensor de paso
        #36 PassS = 0;          // Se desactiva sensor de paso
        #48
// Caso #02 Ingreso de pin incorrecto menos de 3 veces
        #12 CarS = 1;           // Carro detectado
        #36 CarS = 0;           // Carro no detectado
        #36 Passw = 16'h7531;   // Ingreso Password 7531
        IngresoC = 1;           // Se ingresa Password
        #36 IngresoC = 0;       // Se apaga Ingreso Password
        #36 Passw = 16'h7534;   // Ingreso Password 7534
        IngresoC = 1;           // Se ingresa Password
        #36 IngresoC = 0;       // Se apaga Ingreso Password
        #36 Passw = 16'h7530;   // Ingreso Password 7530
        IngresoC = 1;           // Se INgresa Password
        #36 IngresoC = 0;       // Se apaga Ingreso Password
        #36 PassS = 1;          // Se activa sensor de Paso
        #36 PassS = 0;          // Se desactiva sensor de paso
        #48
```

Figura 7: Probador casos 1 y 2 para estructural sin retardo

```
// Caso #03 Ingreso de pin incorrecto 3 o mas veces
        #5 CarS = 1;            // Carro detectado
        #15 CarS = 0;           // Carro no detectado
        #5 Passw = 16'h2024;    // Ingreso Password 2024
        IngresoC = 1;           // Se ingresa Password
        #15 IngresoC = 0;       // Se apaga Ingreso Password
        #5 Passw = 16'h0509;    // Ingreso Password 0509
        IngresoC = 1;           // Se ingresa Password
        #15 IngresoC = 0;       // Se apaga Ingreso Password
        #5 Passw = 16'h1979;    // Ingreso Password
        IngresoC = 1;           // Se ingresa Password
        #15 IngresoC = 0;       // Se apaga ingreso Password
        #15 reset = 1;          // Se activa reset
        #15 reset = 0;          // Se desactiva reset
        #20
// Caso #04 Alarma de Bloqueo
        #10 CarS = 1;           // Carro detectado
        #15 CarS = 0;           // Carro no detectado
        #5 Passw = 16'h7531;    // Ingreso Password 7531
        IngresoC = 1;           // Se Ingresa Password
        #15 IngresoC = 0;       // Se apaga Ingreso Password
        #5 Passw = 16'h7530;    // Ingreso Password 7530
        IngresoC = 1;           // Se Ingresa Password
        #15 IngresoC = 0;       // Se apaga Ingreso Password
        #15 PassS = 1;          // Se activa sensor de paso
        CarS = 1;               // Carro detectado
        #15 PassS = 0;          // Se apaga Sensor de paso
        CarS = 0;               // Carro no detectado
        #20 reset = 1;          // Se activa reset
        #10 reset = 0;          // Se apaga reset
        #70 $finish;            // Se termina pruebas

        end

        always begin
                #5 clock = !clock;
```

Figura 8: Probador casos 3 y 4 estructural con retardo

4. Instrucciones de utilización de la simulación

Para ejecutar los comandos necesarios para realizar la simulación se debe contar con unos requisitos previos para la utilización del mismo, se debe tener instalado las siguientes aplicaciones:

- Icarus Verilog
- GTKWave
- Yosys

4.1. Linux

En caso de que se desee ejecutar en alguna extensión de Linux (Ubuntu, Debian, Xubuntu) se debe correr una terminal en la carpeta donde se encuentran todos lo archivos necesarios para la ejecución de cada parte.

Para ejecutar la síntesis del RTLIL se necesita correr el siguiente comando:

```
>> make techmap
```

Para ejecutar la síntesis estructural se corre el siguiente comando:

```
>> make estructural
```

Una vez realizadas los procesos de síntesis para poder observar los resultados en el GTKwave se debe correr los siguientes comandos.Para ejecutar los resultados del RTLIL:

```
>> make rtlil
```

Para ejecutar los resultados del diseño estructural sin retardo:

```
>> make sinretardo
```

Y por último para ejecutar los resultados del diseño estructural con retardo:

```
>> make conretardo
```

5. Ejemplos de los Resultados

A la hora de ejecutar el testbench se genera un archivo llamado *results.vcd* el cual al abrirse con el GTKWave se podrá visualizar las ondas y comprobar el funcionamiento del sistema.

5.1. Resultados RTLIL

Al generar los resultados del RTLIL se obtiene para el Caso #1 y para el Caso #2, como se observa en la Figura 9 donde se verifica el funcionamiento de manera correcta, que con diferencia al conductual este ya tiene conectado wires y generan un retraso al principio de los estados y los intentos incorrectos, las ondas para el Caso #3 y para el Caso #4 se observan en la Figura 10

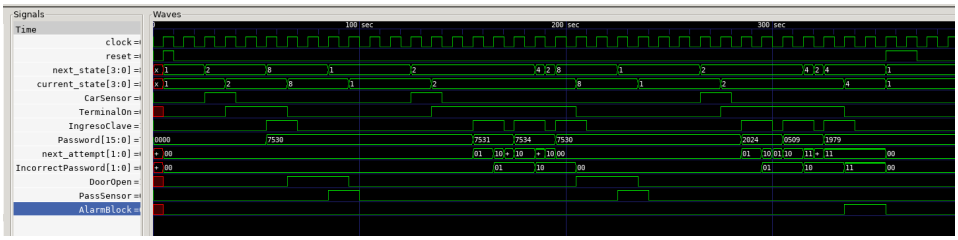

Figura 9: Ondas del GTKwave RTLIL Caso #1 y Caso #2

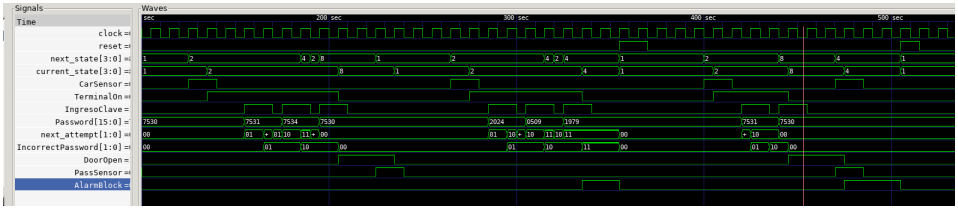

Figura 10: Ondas del GTKwave RTLIL Caso #3 y Caso #4

5.2. Resultados Diseño Estructural sin retardo

Al generar los resultados del estructural sin retardo se obtiene para el Caso #1 y para el Caso #2, como se observa en la Figura 11 y Figura 12 donde se verifica el funcionamiento de manera correcta, que se puede visualizar que es el mismo comportamiento que el RTLIL. Esto indica que al final desde el diseño RTLIL y el estructural sin retardo ambas no cambian debido a que tienen una estructura ya realizada desde el mapeo de jerarquía y estructura.

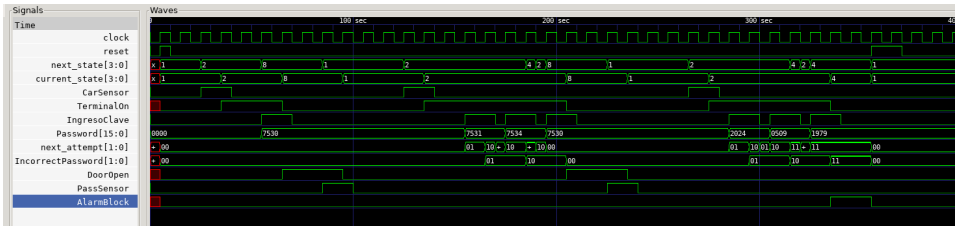

Figura 11: Ondas del GTKwave Diseño estructural sin retardo Caso #1 y Caso #2

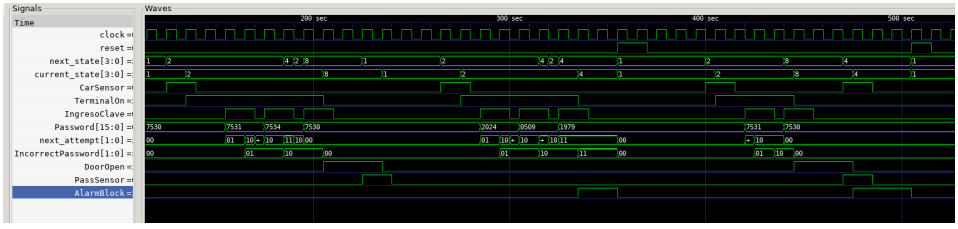

Figura 12: Ondas del GTKwave Diseño estructural sin retardo Caso #3 y Caso #4

5.3. Resultados Diseño Estructural con retardo

Al realizar el tester ajustado para el diseño estructural se tuvo un funcionamiento correcto debido a que el probador anterior con un clock de 10 unidades generaba errores esto gracias a la desincroni-

zación y que la diferencia entre los tiempos de las entradas y salidas tardaban en reaccionar por lo qu ese debía ajustar el clock de manera que se evitara un funcionamiento incorrecto.
Por lo que al cambiar el clock a 24 unidades en total se consiguió obtener un funcionamiento correcto del diseño estructural. Para el Caso #1 se aprecia en la Figura 13 y el Caso #2 en la Figura 14, se comprueba el funcionamiento de las entradas y salidas. Se puede observar el retardo que existe entre entradas y salidas que tardan ciertas unidades de tiempo para activarse, por lo que es un detalle que anteriormente cuando no se aplicaba el retardo las salidas se activaban de inmediato.

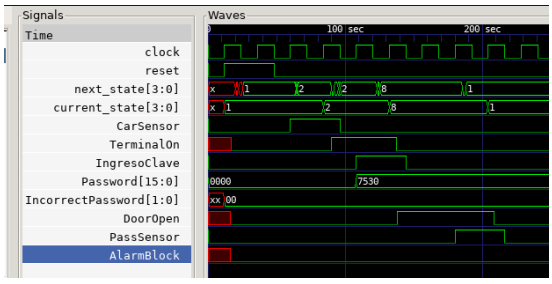

Figura 13: Ondas GTKwave Caso #1

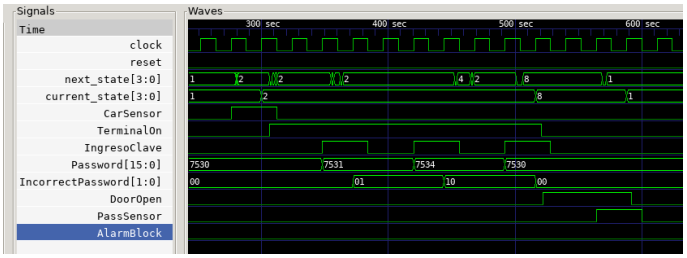

Figura 14: Ondas GTKwave Caso #2

A continuación se puede observar los retardos de propagación entre entradas y salidas en el primer caso y segundo, respectivamente:

Entrada	**Salida**	**Retardo** (s)
CarSensor	TerminalOn	6
IngresoClave	DoorOpen	6
PassSensor	DoorOpen	4

Tabla 2: Retardo propagación Caso #1

Entrada	**Salida**	**Retardo** (s)
CarSensor	TerminalOn	6
IngresoClave	IncorrectPassword	24
IngresoClave	DoorOpen	6
PassSensor	DoorOpen	6

Tabla 3: Retardo propagación Caso #2

Ahora se puede ver para el Caso #3 se aprecia en la Figura 15 y el Caso #4 en la Figura 16, se comprueba el funcionamiento de las entradas y salidas. De igual manera con los retardos de propagación entre entradas y salidas

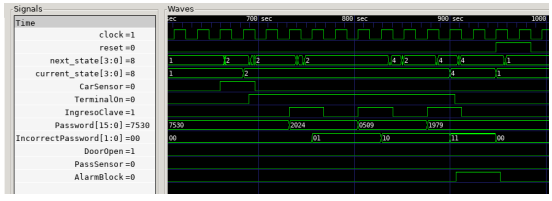

Figura 15: Ondas GTKwave Caso #3

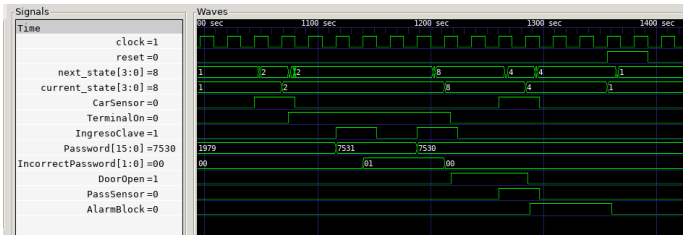

Figura 16: Ondas GTKwave Caso #4

Para todos los casos se obtuvieron esto comportamientos que tanto en el RTLIL y en estructural sin retardos no se presentaban, a la hora de agregar retardos las salidas se activaban en tiempos que no necesariamente se esperaba. Es importante tener cuidado debido a que un Flip Flop en este caso causaba bastantes errores el cual era IncorrectPassword debido a que necesitaba un clock realmente

preciso para que el contador funcionara.

A continuación se puede observar los retardos de propagación entre entradas y salidas en el primer caso y segundo, respectivamente:

Entrada	**Salida**	**Retardo** (s)
CarSensor	TerminalOn	6
IngresoClave	IncorrectPassword	24
IncorrectPassword	AlarmBlock	6
reset	AlarmBlock	4

Tabla 4: Retardo propagación Caso #3

Entrada	**Salida**	**Retardo** (s)
CarSensor	TerminalOn	6
IngresoClave	IncorrectPassword	24
IngresoClave	DoorOpen	6
PassSensor	DoorOpen	6

Tabla 5: Retardo propagación Caso #4

6. Conclusiones y recomendaciones

En conclusión a la hora de realizar la síntesis pueden existir cambios inesperados y fallos debido a que al generar un diseño estructural se convierte un diseño completamente nuevo, producto a esto se necesita tomar en consideración factores como el de mantener las señales de entrada por más tiempo para que exista una lectura correcta y activar la salidas de manera funcional y a tiempo.

Además, al no generar una descripción conductual correcta puede generar latches por lo que a la hora de aplicar las compuertas lógicas que uno desee puede que implique un cambio en el código para que se genere correctamente los FlipFlops y las compuertas. Al agregar los retardos se pudo observar como aparecieron retardos de propagación entre entradas y salidas que generan un cambio en las ondas y en el funcionamiento del sistema, por lo que por medio del esquemático de yosys se puede realizar un análisis para entender la reacción de las compuertas a los retardos.

En este proyecto se aprende como en la realidad se debe tener en consideración los tiempos en que las compuertas y la estructura tardan en generar las respuestas, y que son uno de los muchos problemas que existe en el diseño de circuitos digitales, en donde habrán factores que afecten y se distancien del diseño que uno genera en simuladores. Por lo que es importante tener en consideración realizar el diseño pensando en posibles errores en un futuro que le evitan tiempo, correcciones entre otras barreras en la industria. Mantener ciertos requisitos como por ejemplo la velocidad del clock son metas que requieren arduo trabajo y que necesitan un proceso ordenado de verificación y diseño.