Universidad de Costa Rica
Facultad de Ingeniería

Escuela de Ingeniería Eléctrica

Tarea # 2

Jose Pablo Laurent Chaves Carné: B63761

21 de septiembre de 2024

Resumen

La presente tarea pretende realizar una descripción estructural a partir del programa de síntesis Yosys utilizando la librería de elementos lógicos proporcionada en clase para hacer el mapeo tecnológico correspondiente, para ello se utiliza el diseño de la tarea #1.

Para ello se pretende realizar una descripción estructural genérica (RTLIL) que no depende de una tecnología en particular, luego verificar que la descripción estructural genérica funciona usando la infraestructura de pruebas diseñada para la tarea #1. Posteriormente se selecciona la biblioteca cmos_cells.lib para realizar el mapeo tecnológico del diseño, con su respectiva verificación. Finalmente se modifican los archivos de la biblioteca para tomar en cuenta retardos, evaluando su funcionamiento y contabilizando la cantidad de componentes.

Índice

1. Descripción Arquitectónica

En esta sección se detalla el funcionamiento del controlador de estacionamiento.

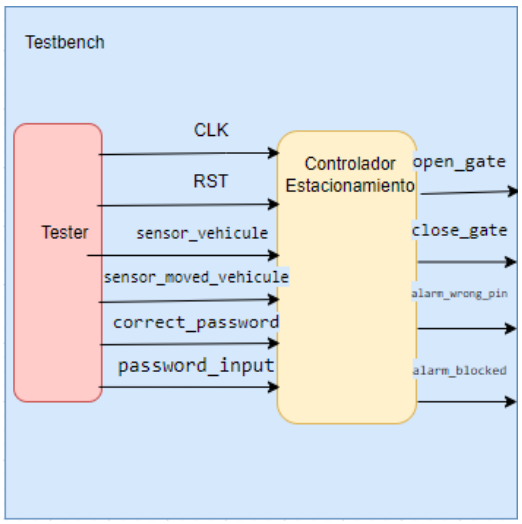

Figura 1: Diagrama de bloques del controlador.

Con base en la imagen de la Figura 1, la estructura del proyecto consta de un archivo denominado “testbench.v” el cual importa e instancia los archivos “tester.v” y “controlador_estacionamiento.v” de manera respectiva.
Con respecto al archivo “tester.v”, este alimenta al controlador del estacionamiento mediante las señales clk (señal de reloj), rst (señal de reinicio del controlador), sensor_vehicule (señal que detecta si llego un vehículo), sensor_moved_vehicule (señal que indica si el vehículo se movió), password_input (señal de entrada con el pin que ingresa el usuario) y correct_password (señal con el pin correcto para abrir la compuerta).
Por otro lado, “controlador_estacionamiento.v” recibe como entrada las señales previamente mencionadas, luego realiza lógica combinacional para obtener 4 señales de salida denominadas como: open_gate (señal que se da cuando el usuario ingresa la contraseña correcta y se abre la compuerta), close_gate (señal que se da cuando el vehículo terminó de moverse y cierra la compuerta), alarma_wrong_pin (señal que se activa cuando el usuario realiza 3 intentos fallidos de ingreso de pin) y por último, alarm_blocked (señal que se da cuando se activa el sensor de llegada de vehículo y sensor de que el vehículo ya terminó de pasar).

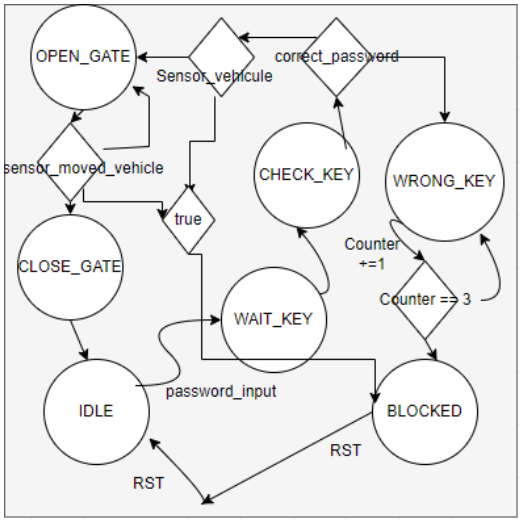

Figura 2: Diagrama de estados.

En la imagen de la Figura 2 se observa el comportamiento a nivel interno del controlador, en el se establece un total de 7 estados.

1. IDLE: este se define como el estado inicial del controlador, donde se inicializan en cero las siguientes variables open_gate, close_gate, alarm_wrong_pin, alarm_blocked y counter, además se establece como estado siguiente WAIT_KEY.

2. WAIT_KEY: este estado se encarga de esperar una contraseña por parte del usuario y establece como estado siguiente CHECK_KEY.

3. CHECK_KEY: este estado verifica que la contraseña sea correcta, en caso de serlo se establece como siguiente estado OPEN_GATE, caso contrario el siguiente estado será WRONG_KEY.

4. WRONG_KEY: este estado cuenta la cantidad de intentos fallidos, en el caso de que el usuario llegue a un total de 3 intentos fallidos activa una alarma denominada como alarm_wrong_pin que bloquea el controlador, por consiguiente se establce como siguiente estado BLOCKED.

5. BLOCKED: es el estado que bloquea el sistema, puede recibir activar tanto alarm_wrong_pin como alarm_blocked como señales de salida de controlador de manera respectiva, en caso de querer volver a utilizar el sistema se activa la señal de RST para volver al estado IDLE.

6. OPEN_GATE: Este estado se encarga de abrir la compuerta del estacionamiento, para ello se necesita que se active la señal sensor_vehicule.

7. CLOSE_GATE: Este estado se encarga de cerrar la compuerta del estacionamiento, para ello debe activarse la señal sensor_moved_vehicule, posteriormente se cierra la compuerta. Nota: en caso de que las señales sensor_vehicule y sensor_moved_vehicule este activas al mismo tiempo se activará la señal alarm_blocked.

2. Plan de Pruebas

En esta sección se establece un total de 4 pruebas.

2.0.1. Prueba #1, funcionamiento normal básico.

Llegada de un vehículo, ingreso del pin correcto y apertura de puerta, sensor de fin de entrada y cierre de compuerta.

2.1. Prueba #2, ingreso de pin incorrecto menos de 3 veces.

Llegada de un vehículo, ingreso de pin incorrecto (una o dos veces), la compuerta permanece cerrada. Ingreso de pin correcto, funcionamiento normal básico. Revisión de contador de intentos incorrectos

2.2. Prueba #3, ingreso de pin incorrecto 3 o más veces.

Revisión de alarma de pin incorrecto. Revisión del contador de intentos incorrectos. Ingreso de pin correcto, funcionamiento normal básico. Revisión de limpieza de contadores y alarmas.

2.3. Prueba #4, alarma de bloqueo.

Ambos sensores encienden al mismo tiempo, encendido de alarma de bloqueo, ingreso de clave incorrecta, bloqueo permanece. Ingreso de la señal reset para el desbloqueo. Funcionamiento normal básico.

3. Instrucciones de utilización de la simulación

3.1. Makefile

En la terminal de Linux ejecute el comando *make*, este desplegara 3 formas de onda.

```
all: yosys controladorRTLIL controladorSintetizado controladorRetardos

yosys: controlador.ys
	yosys -s controlador.ys

controladorRTLIL: testbench_RTLIL.v
	iverilog testbench_RTLIL.v -o RTLIL.out
	vvp RTLIL.out
	gtkwave ondas_RTLIL.gtkw &

controladorSintetizado: testbench.v
	iverilog testbench.v -o sintetizado.out
	vvp sintetizado.out
	gtkwave ondas.gtkw &

controladorRetardos: testbench_con_retardos.v
	iverilog testbench_con_retardos.v -o sintetizado_retardos.out
	vvp sintetizado_retardos.out
	gtkwave ondas_con_retardos.gtkw &

mostrar_ondas:
	gtkwave ondas_RTLIL.gtkw ondas.gtkw ondas_con_retardos.gtkw &

clean:
	rm -f *.out
	rm .*.swp
```

Figura 3: Makefile.

- La tarea yosys crea dos archivo denominados controlador_RTLIL.v y controlador_synth.v este último contiene el condigo sintetizado sin retardos.
- La tarea controladorRTLIL lee el archivo testbench_RTLIL.v y este incluye los archivos controlador_RTLIL.v y tester_sin_retardos.v.
- La tarea controladorSintetizado incluye el archivo testbench.v que a su vez incorpora controlador_synth.v, cmos_cells.v y tester_sin_retardos.v.
- La tarea controladorRetardos incluye los archivos controlador_synth.v, cmos_cells_con_retardos.v (se modifica archivo para tener retardos de 1 unidad de tiempo)y tester.v (contiene un periodo de reloj de 20 unidades de tiempo y modificaciones en el tiempo de los planes de pruebas para que el controlador actúe adecuadamente).

4. Ejemplos de los resultados de la Tarea #1

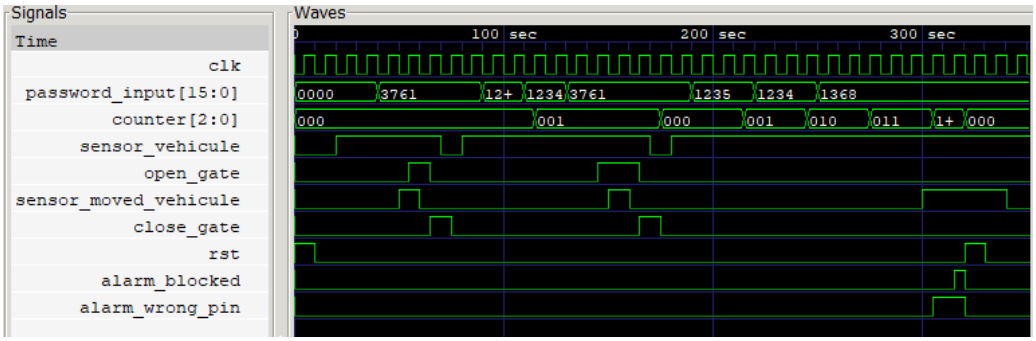

Figura 4: Resultados.

En la imagen de la figura 4 se observan los resultados obtenidos de la simulación.

4.0.1. Prueba #1, funcionamiento normal básico.

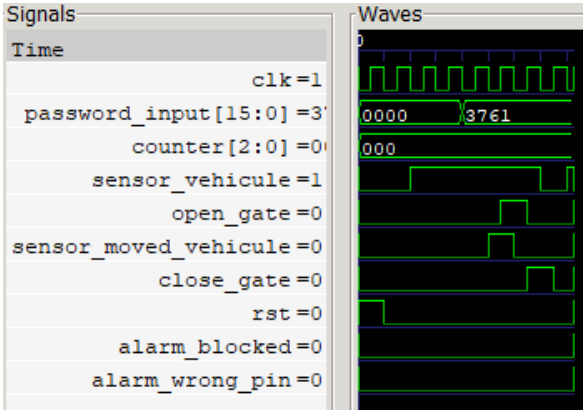

Figura 5: Prueba #1.

De la imagen de la Figura 5 se observa la llegada de un vehículo, el ingreso del pin correcto, luego la apertura de puerta, el sensor de fin de entrada detecta que el vehículo terminó de pasar y cierre de compuerta.

4.1. Prueba #2, ingreso de pin incorrecto menos de 3 veces.

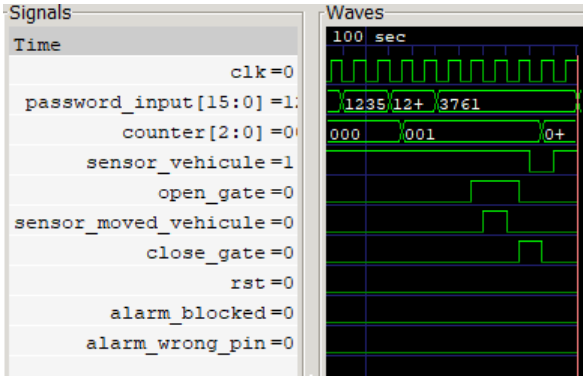

Figura 6: Prueba #2.

De la imagen de la Figura 6 se detecta la llegada de un vehículo, ingreso de pin incorrecto dos veces, la compuerta permanece cerrada, luego se ingresa el de pin correcto, se de el funcionamiento normal básico. El contador detecta únicamente un error, debido al tiempo de la señal de reloj no fue suficiente para contar el segundo error, luego de que se ingresa el pin correcto el contador se re inicia.

4.2. Prueba #3, ingreso de pin incorrecto 3 o más veces.

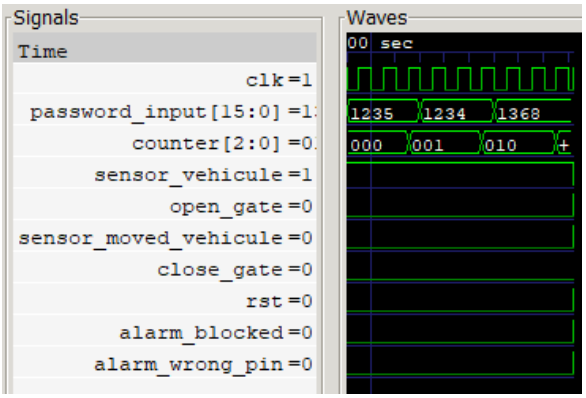

Figura 7: Prueba #3.

De la imagen de la Figura 7 se observa que el usuario ingreso el pin incorrecto un total de 3 veces por tal motivo no se abre la compuerta ni deja pasar el vehículo, en esta captura de pantalla no se observa la activación de la alarma de pin incorrecto, sin embargo en la imagen de la Figura 8 sí se muestra.

4.3. Prueba #4, alarma de bloqueo.

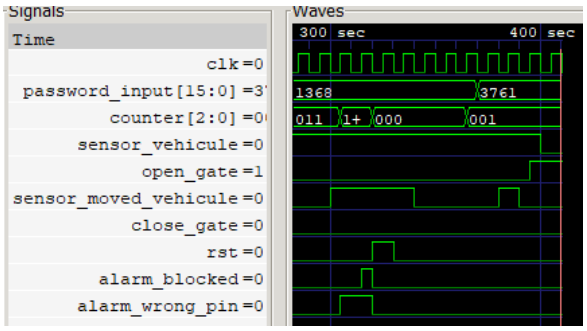

Figura 8: Prueba #4.

En la imagen de la Figura 8 se observa como luego de que el usuario realizará un total de 3 intentos fallidos se activa la alarma “alarm_wrong_pin”, además se activa la alarma “alarm_blocked” cuando se detecta un vehículo y además se detecta que el que vehículo se está moviendo, finalmente para volver a utilizar el controlador se activa la señal de “rst”.

5. Síntesis

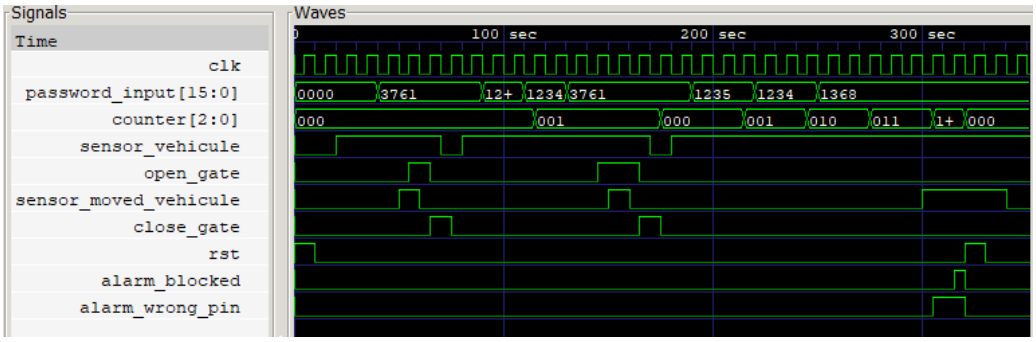

Figura 9: Resultados código conductual.

De la imagen de la Figura 9 se obtienen los resultados del código conductual del circuito de la Tarea #1 ante el plan de pruebas previamente descrito.

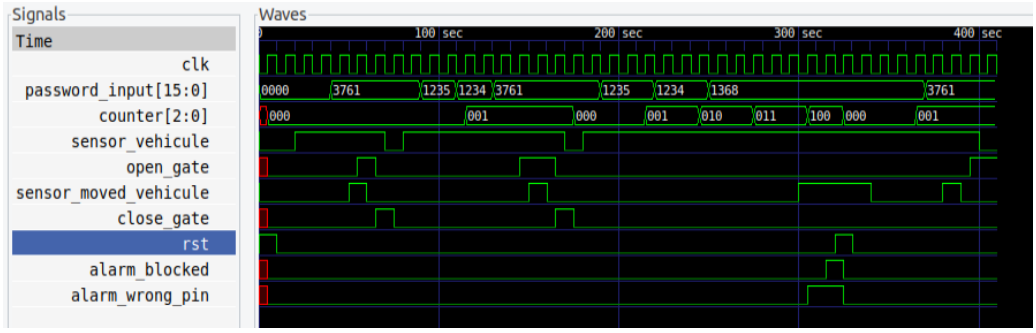

Figura 10: Resultados código con RTLIL.

De la imagen de la Figura 10 se obtienen los resultados del circuito sintetizado RTLIL.

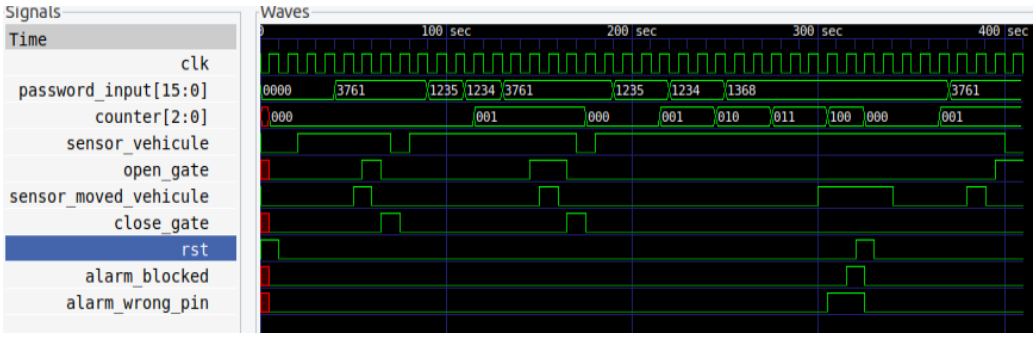

Figura 11: Resultados código con cmos_cells.lib.

De la imagen de la Figura 11 se obtienen los resultados del circuito sintetizado utilizando la biblioteca cmos_cells.lib.

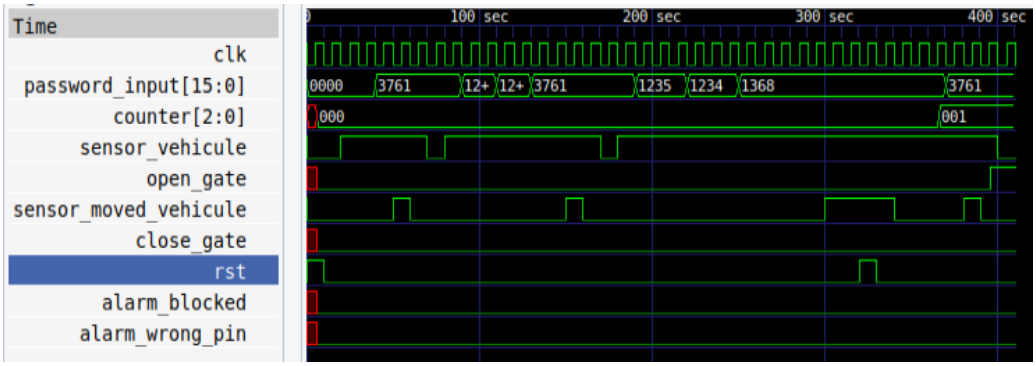

Figura 12: Resultados con retardos

En la imagen de la Figura 12 se ingresan retardos de #1 para las compuertas NAND, NOR, NOT y FFs, se observa que ante los retardos de las compuertas y FFs el circuito no posee el tiempo de reacción suficiente, por tal motivo el circuito no realiza correctamente sus funciones.

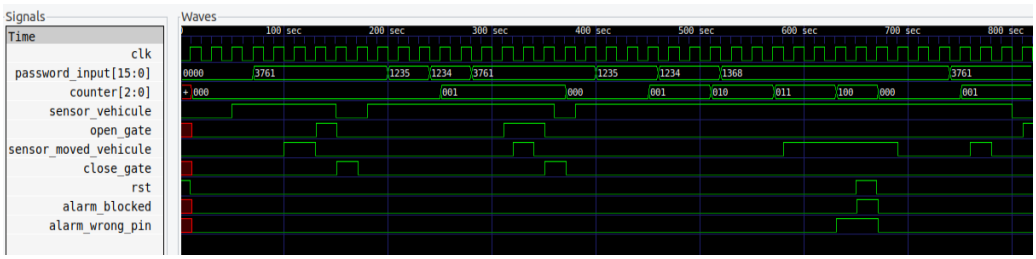

Figura 13: Resultados con retardos corregidos

De la imagen de la Figura 13 se agrega retardos de #1 para las compuertas NAND, NOR, NOT y FFs, además el ciclo del reloj pasa de 10 a 20 unidades de tiempo para que el el circuito pueda tener el tiempo necesario para procesar el retardo de las compuertas, por último se modifican e incrementan los tiempos de las señales de entrada en tester.v para que el circuito tenga el tiempo suficiente para mostrar los resultados deseados.

6. Cantidad NAND, NOR, NOT y FFs

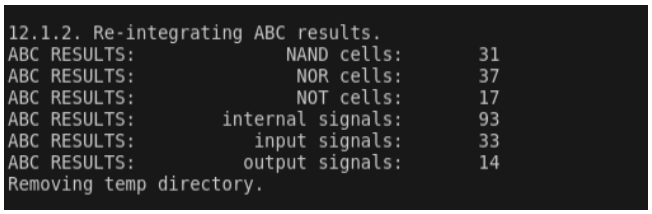

Figura 14: NAND, NOR, NOT

De la imagen de la Figura 14 se obtienen NAND 31, NOR 37 y NOT 17.

Figura 15: FFs

De la imagen de la Figura 15 se obtienen 14 Flip Flops. El retardo total del circuito es de 99 unidades de tiempo por tal motivo de incrementa el periodo del reloj a 20 unidades de tiempo con el fin de disminuir la frecuencia del reloj que en cada flanco positivo de esta ocurra un cambio y el circuito tenga el suficiente tiempo para responder de manera adecuada.

7. Conclusiones y recomendaciones

Para poder hacer sintetizable el circuito de la tarea #1, se debe añadir los 4 últimos dígitos en el dut conocido como controlador.v, por ejemplo:

```
parameter correct_password =16'b0011_0111_0110_0001;
// contraseniacorrecta 3761
```

La instrucción anterior no es sintetizable por Yosys al enviar la contraseña correcta desde el testbench. De manera adicional al ejecutar el comando "abc -liberty ./cmos_cells.lib" se deben evitar agregar espacio en la ruta que alberga el comando abc dentro de controlador.ys, para poder vincular los flip flops por defecto con la librería ./cmos_cells.lib, se debe verificar que el código conductual posee únicamente "@(posedge clk)" ya que si se agrega otro argumento no será posible su vinculación.

Se concluye que es posible sintetizar un codigo conductual en estructural utilizando Yosys como herramienta, además al agregar retardos en el circuito sintetizado se deben disminuir la frecuencia del reloj para que tenga tiempo de realizar las operaciones deseadas y es posible que se deben realizar modificaciones en el temporizador del plan de pruebas para obtener los resultados deseados.