Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
IE0523 – Circuitos Digitales II
I ciclo 2024

Reporte Laboratorio 1

Coordinación Semáforo / Peatonal

Alexa López Marcos B94353

Profesor: Ana Eugenia Sánchez Villalobos

7 de Abril del 2024

1. Resumen

El problema implica coordinar semáforos y peatonales en dos calles, Calle A y Calle B. Los carros que cruzan por estas calles solo pueden seguir en línea recta y no pueden cambiar a otra calle. Por lo tanto, es crucial una coordinación entre los semáforos, las calles y los peatones para evitar accidentes, asegurando que los peatones puedan cruzar de manera segura cuando corresponda según el estado de los semáforos de las calles. Como parte del funcionamiento:

- Los semáforos de las calles A y B tienen tres estados: rojo, amarillo y verde.
- Cuando el semáforo A está en verde, el peatonal correspondiente (Apeatonal) debe estar en rojo, y viceversa.
- Lo mismo se aplica para el semáforo B y su peatonal (Bpeatonal).
- El circuito debe tener en cuenta la sincronización con el reloj (CLK) y el habilitador (ENB), así como la posibilidad de reinicio (RST).

2. Ejecución

Para realizar una correcta ejecución del programa se facilita el siguiente archivo tipo makefile. Para poder ejecutar este archivo desde la terminal de Linux se debe de :

- Abrir la terminal de Linux.
- Posicionarse en la carpeta en la que se encuentra el archivo makefile con los archivos de los módulos del programa.
- Colocar en la terminal **make verilog**

Este archivo tipo makefile se utilizara para automatizar la ejecución del programa y contiene las líneas necesarias para ejecutar el programa desde la terminal, se colocan dentro de una bandera del archivo tipo makefile, para que al ejecutar en la terminal se generen los archivos necesarios para ejecutar y abrir GTKWave para ver el comportamiento de las señales.

Dicho makefile contiene:

```
verilog:
        iverilog -o tb.vvp testbench.v
        vvp tb.vvp
        gtkwave tb.vcd
        rm -rf tb.vvp tb.vcd

clean:
        rm -rf tb.vvp tb.vcd
```

Con esto, ya no es necesario ejecutar una a una las líneas necesarias para ejecutar el programa, solamente se ejecutaría **make verilog** desde la terminal.

3. Análisis

El programa está compuesto de 3 módulos distintos:

- **DUT**
- **TESTER**
- **TESTBENCH**

3.1. DUT

El archivo llamado **SemaforoDUT.v** corresponde al módulo DUT, en el cual se especifican las condiciones que deben de cumplir las entradas para ejecutar las salidas.

3.1.1. Entradas:

En este caso, este módulo contiene como entradas las señales:

- **CLK**: Se utiliza para sincronizar las señales.
- **ENB**: Esta señal es fundamental para que el circuito funcione y tiene las siguientes condiciones:
  - ***ENB=1***: Esta señal debe de estar en alto(1) para que todo el circuito funcione.
  - ***ENB=0***: En caso de estar en bajo(0) el circuito debe de apagarse.
- **RST**: Se utiliza para colocar las señales de los semáforos peatonales en su estado inicial, en este caso, ese estado inicial es 0.
- **Semaforo_A**: Esta señal condiciona el semáforo peatonal A, cuando esta en alto(1) el peatonal A debe de estar en bajo(0) y tiene 3 estados:
  - ***ROJO***: Cuando la señal es *2'b00*.
  - ***AMARILLO***: Cuando la señal es *2'b01*.
  - ***VERDE***: Cuando la señal es *2'b10*.
- **Semaforo_B**: Esta señal condiciona el semáforo peatonal B, cuando esta en alto(1) el peatonal B debe de estar en bajo(0) y tiene 3 estados:
  - ***ROJO***: Cuando la señal es *2'b00*.
  - ***AMARILLO***: Cuando la señal es *2'b01*.
  - ***VERDE***: Cuando la señal es *2'b10*.

3.1.2. Salidas:

En este caso, este módulo contiene como salida las señales:

- **A_Peatonal**: Corresponde a la señal del semáforo peatonal A, este coincide con el semáforo A y tiene que cumplir la condición de que si el semaforo A esta en verde o amarillo, el peatonal A va a estar en rojo. Los estados del peatonal A son:
  - ***ROJO***: Cuando la señal es 0.

- ***VERDE***: Cuando la señal es 1.

- **B_Peatonal**: Corresponde a la señal del semáforo peatonal B, este coincide con el semáforo B y tiene que cumplir la condición de que si el semaforo B esta en verde o amarillo, el peatonal B va a estar en rojo. Los estados del peatonal B son:

  - ***ROJO***: Cuando la señal es 0.
  - ***VERDE***: Cuando la señal es 1.

3.1.3. Código:

Para cumplir con que el código este sincronizado por el CLK, activado solo con la señal ENB y que coloque las salidas en 0 cuando se aplica la señal RST, se crea el suguiente bloque de código dentro del DUT:

```
always @(posedge CLK)
begin
    if (ENB) begin
    // Si el enable esta encendido entonces todo debe funcionar
        if (Semaforo_A == VERDE && (Semaforo_B == ROJO || Semaforo_B
           ↪ == AMARILLO)) begin
            // Si el semaforo A esta en verde y el semaforo B esta en
               ↪ amarrillo o rojo, entonces...
            A_Peatonal <= 0;    // El peatonal A va a estar en rojo.
            B_Peatonal <= 1;    // El peatonal B va a estar en verde.

        end else if (Semaforo_B == VERDE && (Semaforo_A == ROJO ||
           ↪ Semaforo_A == AMARILLO)) begin
            // Si el semaforo B esta en verde y el semaforo A esta en
               ↪ amarrillo o rojo, entonces...
            A_Peatonal <= 1;    // El peatonal A va a estar en verde.
            B_Peatonal <= 0;    // El peatonal B va a estar en rojo.

        end
    end else if (RST) begin // Todo vuelve a su condicion inicial
            B_Peatonal <= 0; // En rojo
            A_Peatonal <= 0; // En rojo
        end
end
```

3.2. TESTER

El archivo llamado **tester.v** corresponde a módulo TESTER, el cual se utiliza para definir el comportamiento de las entradas del circuito, en este módulo en específico las entradas y salidas se definen inversas a como se encuentran definidas en el DUT.//

3.2.1. Código:

En este módulo se define la señal de CLK como una señal periodica en el que cada flanco tarda 1ns, esto se realizo con el siguiente código:

```
1  `timescale 1ns/1ns // Escala de tiempo
2  .
3  module tester(
4      .
5      .
6  );
7  .
8  .
9  always begin
10     #1 CLK = !CLK;
11 end
12 .
13 .
14 endmodule
```

Los cambios de las señales se realizaron de la sigiente forma:

Tiempo	Semaforo_A	Semaforo_B	ENB	RST
0ns	x	x	1	0
+2ns	ROJO	VERDE	1	0
+2ns	ROJO	AMARILLO	1	0
+2ns	VERDE	ROJO	1	0
+2ns	AMARILLO	ROJO	1	0
+2ns	ROJO	VERDE	1	0
+2ns			0	1
+2ns			1	0
+2ns	ROJO	AMARILLO	1	0
+2ns			0	1
+2ns			1	0
+2ns	VERDE	ROJO	1	0
+2ns	AMARILLO	VERDE	1	0
+2ns	ROJO	VERDE	1	0
+2ns	ROJO	AMARILLO	1	0
+2ns	VERDE	ROJO	1	0
+10ns			1	0

3.3. TESTBENCH

El archivo llamado **testbech.v** corresponde al módulo TESTBENCH, en el que se importan los módulos DUT y TESTER para poder realizar las conexiones por medio de cables de las salidas del DUT con las entradas del TESTER y las salidas del TESTER con las entradas el DUT, esto hace que se cierre el circuito y se pueda realizar la simulación.

3.4. Resultados

Como resultados de la ejecución en conjunto de los 3 módulos y con uso de la herramienta GTKWave obtenemos la figura 1:

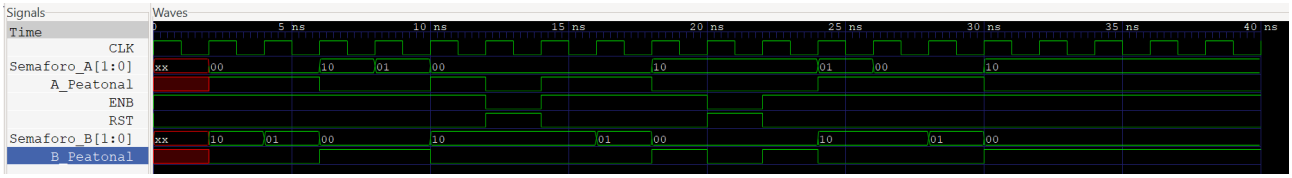

Figura 1: Imagen de la simulación realizada en GTKWave.

4. Conclusiones

- Al realizar el cambio de los semáforos se coordina que estos no permitan el paso vehicular del carril A, mientras en el carril B aun se encuentren circulando vehículos, y viceversa, por medio del propio diseño del módulo de pruebas.
- Se cumple la condición de que mientras el semáforo A se encuentre en verde, el peatonal A no se encuentre en verde. Esta condición también se cumple para el semáforo B y el peatonal B.
- Se logra realizar los 3 módulos para resolver el problema planteado, el módulo DUT, el módulo de pruebas TESTER y el módulo de conexiones TESTBENCH.
- Se logra realizar un archivo tipo makefile para automatizar la ejecución del programa desde una terminal de Linux.

5. Aspectos

5.1. ¿Qué aspectos puede mejorar y cuales le costaron?

Creo que en el código de los módulos se podría mejorar un poco la documentación. Al inicio al leer la instrucción sí me costó un poco comprender cuales eran las condiciones que se debían de cumplir pero con un poco de lectura y análisis más detenido logré comprender las especificaciones del programa a realizar. Por mejorar a nivel de conocimiento me gustaría conocer como sería un programa que necesite tener más de los 3 módulos para funcionar, y en ese caso como sería la estructura y lógica de programación a seguir.

5.2. ¿Aprendió cosas nuevas aparte de lo visto en clase?

Sí, con este laboratorio logré comprender mejor que se realizaba en cada uno de los módulos. Que en el modulo de DUT se condiciona el comportamiento de las salidas con respecto al comportamiento de las entradas. Que en el módulo de TESTER se genera el comportamiento que se quiere que tengan las entradas. Y que en el módulo de TESTBENCH es el que realiza las condiciones "físicas.^{en}tre el módulo de DUT y TESTER.