UNIVERSIDAD NACIONAL DE TUCUMÁN

FACULTAD DE CIENCIAS EXACTAS Y TECNOLOGÍA

ELECTRÓNICA II

Actividad 6: Controlador de Semáforo

Autor: Chaves, Gonzalo Efraín - Estudiante de Ing. Electrónica

Resumen

En esta actividad se desarrolló una descripción de hardware de un controlador de semáforo digital. Se contempla entradas de solicitud para cruce peatonal como detección para vehículos de emergencia.
El objetivo principal fue implementar un sistema secuencial capaz de controlar una semaforo de manera automatica, en secuencia normal (verde, amarillo, rojo), modificandolo dependiendo de pedidos externos como pedido peatonal y de emergencia.

Para lograr un diseño eficiente y modular, se aplicó el concepto de máquina de estado factorizada, dividiendo el sistema principal en submódulos más simples e interconectados. De este modo permitió reducir la complejidad lógica y mejorar la comprensión del sistema. En particular, se implementaron tres máquinas: un prescaler para obtener una base de tiempo de 1 Hz a partir del reloj de 12 MHz, un temporizador (Timer) encargado de generar los intervalos de duración de cada tic del reloj, y la máquina principal del semáforo, responsable de coordinar los estados de tránsito y las condiciones de prioridad.

Finalmente, se verificó el funcionamiento mediante un testbench automático, que permitió aplicar estímulos y comparar los resultados obtenidos con los patrones esperados. Las simulaciones confirmaron el comportamiento correcto del sistema en las secuencias normales, los pedidos peatonales y los modos de emergencia.

Introducción

Enunciado:

Diseñar un control de semáforo para una intersección de calles. La secuencia normal del semáforo será rojo durante 60 s, verde durante 50 s y amarillo durante 10 s. Cuando una calle está en verde o amarillo la otra calle estará en rojo. Cada dirección tiene un sensor detector de servicios de emergencia y un pulsador de cruce peatonal.

El detector de servicio de emergencia inicia un modo de emergencia.

- Si una dirección está en rojo y se activa el detector de servicios de emergencia en dicha dirección, la otra dirección pasará de inmediato a amarillo, para luego dar paso a la dirección de emergencia.
- Si una dirección está en verde y se activa el detector de servicios de emergencia en dicha dirección, permanecerá en verde hasta que el detector se desactive.
- Si una dirección está en modo de emergencia, la otra no puede entrar a dicho modo hasta que se despeje la primera.

El pulsador de cruce peatonal inicia el modo de cruce peatonal.

- El modo de cruce peatonal extiende al doble el tiempo verde de la dirección correspondiente.
- Si la otra dirección entra en modo de emergencia durante el verde de cruce peatonal, esperará igualmente a que finalice normalmente el verde del cruce peatonal (no inicia el amarillo de inmediato).

Realizar la descripción de hardware del diseño y un banco de pruebas apropiado para verificar su normal funcionamiento. Incluir, además de las entradas necesarias para satisfacer los requerimientos, una entrada de reset activa baja (nreset). Realizar la síntesis lógica y configuración de la EDU-CIAA-FPGA con el agregado de entrada/salida utilizado en las actividades anteriores. Para simular los indicadores del semáforo se utilizarán los segmentos del display y para simular pulsadores de cruce peatonal y sensores de emergencia se utilizarán los bit de entrada según se indica en la figura 1. La fuente de reloj será el reloj de 12 MHz integrado a la placa edu-ciaa-fpga.

Especificaciones:

Entradas:
Solicitud_peaton_a, Solicitud_peaton_B, Solicitud_emergencia_a, Solicitud_emergencia_b.

Estados:
Verde_a, Verde_b, Amarillo_a, Amarillo_b, Verde_a_ad, Verde_b_ad, emerg_a, emerg_b.

Salidas:
transito_a, transito_b, peaton_a, peaton_b, confirmacion_peaton_a, confirmacion_peaton_b, confirmacion_emergencia_a, confirmacion_emergencia_b.

Lógica sincrónica, así que tiene una entrada de clk, nreset(reset activo bajo).

Materiales y Métodos

Para esta actividad se investigó sobre dos Máquinas de Estado Finito (Moore y Mealy), al final se decidió implementar una máquina de Moore.
Si la máquina se realiza como una máquina única, su complejidad sería muy superior a hacerlo como una máquina factorizada, es decir derivando procesos a otras máquinas de estado más sencillas.

Durante el desarrollo se utilizó el método de realizar y controlar por etapas. Para ello se realizó una TestBench automática, donde se lee dos archivos de textos con vectores, donde en uno se leen los estímulos de las entradas, y en el segundo se lee el patrón esperado de las señales de salida.
Una vez realizado se irá complejizando la máquina agregando la lógica combinacional y revisando que la misma tenga la respuesta esperada, y así etapa por etapa, hasta tener la máquina completa.

Resultados

Se utilizaron tres máquinas de estado en esta implementación:

- Prescaler: máquina que recibe un reloj de 12Mhz, y devuelve un tic cada 1 segundo, es decir pasa de 12 MHz a 1 Hz(cambio de escala del reloj).

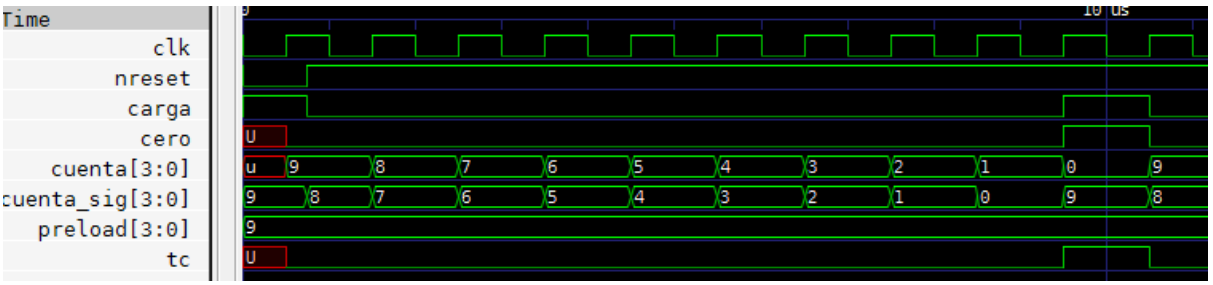


Figura 1.1 - Test Bench del Prescaler.

En la figura 1.1, se observa el funcionamiento del prescaler, donde tc, es el tic que devuelve cada ciclo, en este ejemplo, cada 10 ciclos de reloj devuelve un tc.

- Timer: esta máquina recibe una cuenta desde la cual debe contar hacia atrás cada un segundo, también recibe la salida del Prescaler para trabajar a 1 Hz. En su salida devuelve un T(time) cuando la cuenta es igual a ‘1’ y devuelve un Z(zero) cuando la cuenta es igual a ‘0’.

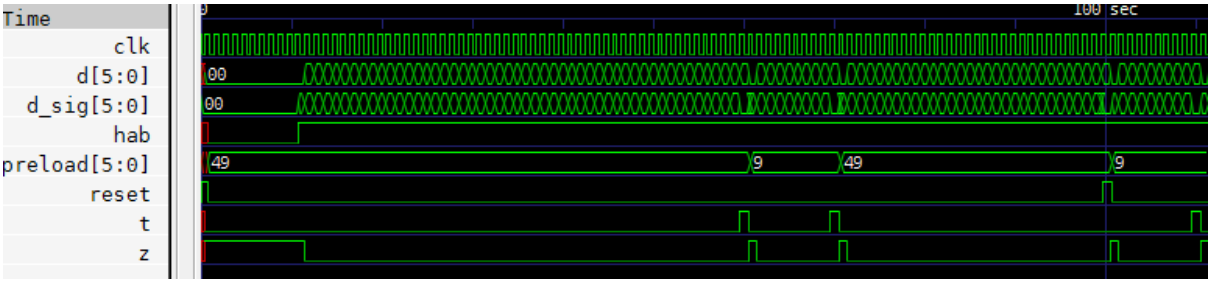


Figura 1.2 - Test Bench del Timer

En la Figura 1.2, se observa el funcionamiento del Timer, donde se le da un valor de recarga (preload) y cada ciclo o habilitación cuenta atrás, restando uno (-1) a la cuenta ingresada, devuelve un t (time) cuando la cuenta llega a ‘1’ y devuelve un z (zero) cuando la cuenta llega a ‘0’, además en ese momento carga la siguiente recarga, comenzando a contar nuevamente.

- Semáforo: máquina principal, la cual implementa las dos anteriores para su funcionamiento, de acuerdo a las especificaciones arriba mencionadas. Su diagrama de estados se muestra en la Figura 1.3, donde se muestra de forma de alto nivel, algunas entradas y salidas codificadas.

Diagrama de Estados General

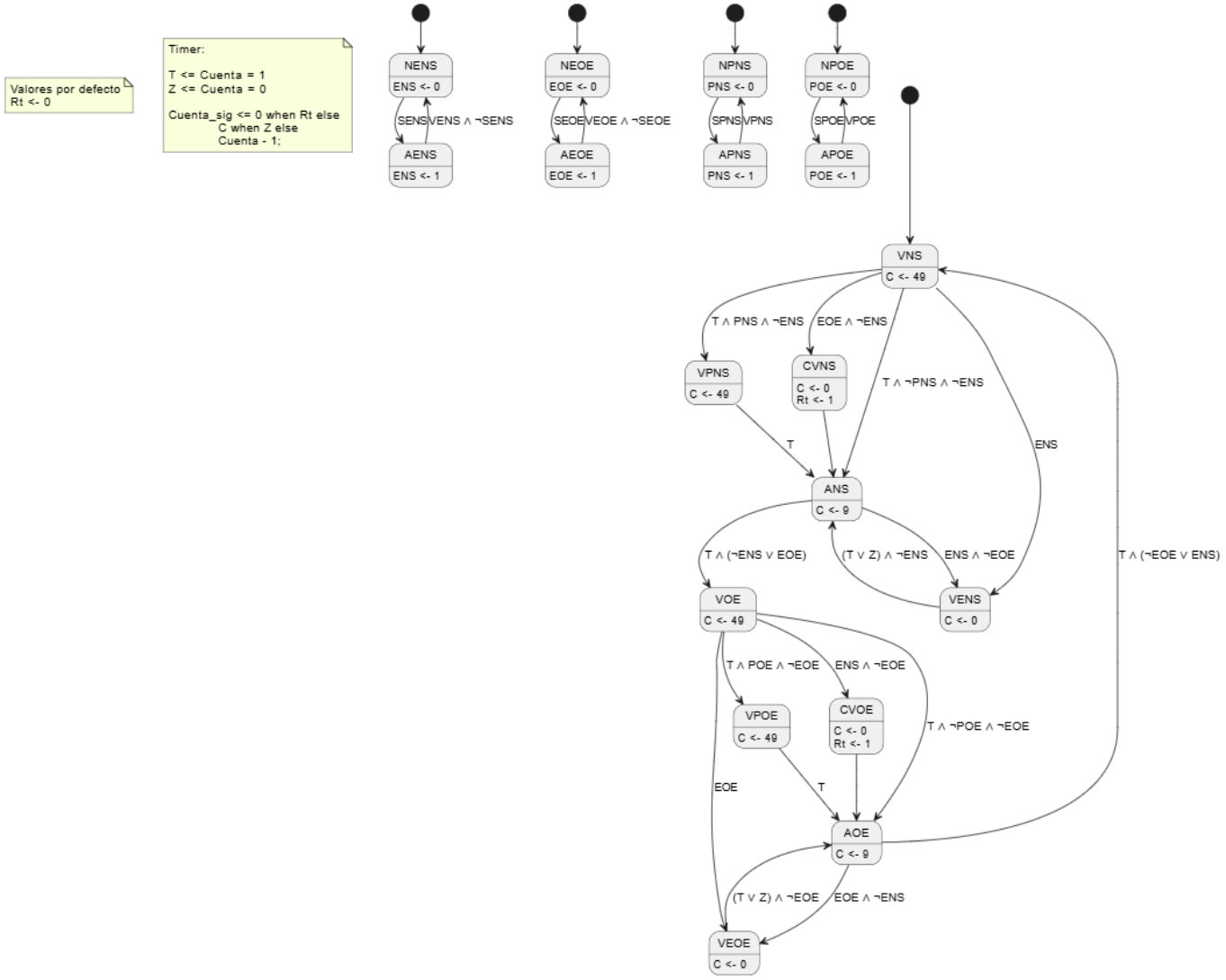


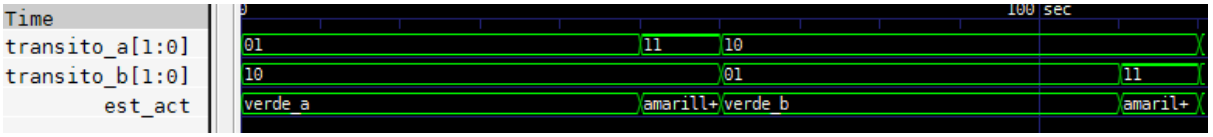
Figura 1.3, diagrama general de estados del controlador de semaforo

Test Bench del Controlador de Semaforo:

Recordar que se utilizó un Test Bench Automatico, donde lee vectores de estimulo formados como:
solicitud_peaton_a&solicitud_peaton_b&solicitud_emergencia_a&solicitud_emergencia_b y un entero que indica el tiempo de estimulo.
Y codificacion para los colores: Verde : “01”, Amarillo : “11”, Rojo : “10”.

Primera Implementación, y revisión de funcionamiento, Secuencia normal:

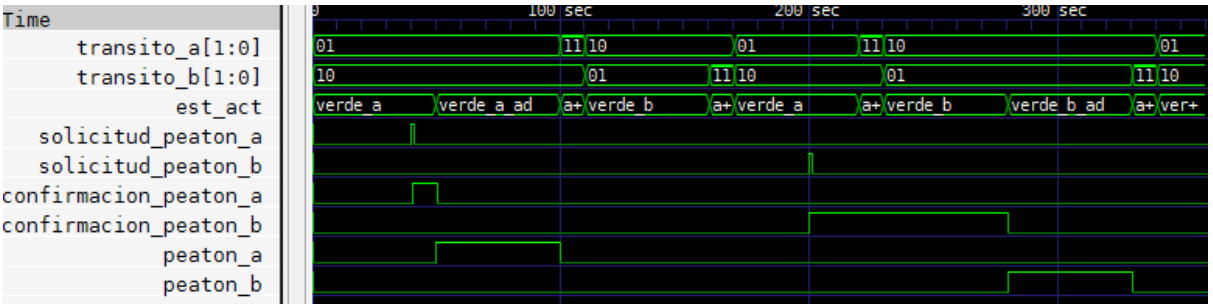
Estímulo recibido:
0000 120



Se observa como el transito_a pasa de verde → amarillo → rojo.
Y el transito_b pasa de rojo → verde → amarillo
Secuencia normal correcta.

Segunda implementación, Pedido peatonal:

Estímulo recibido:
0000 40
1000 1 - - solicitud_peaton_a por 1 segundo
0000 159 - - espera 159 sin estimular nada
0100 1 - - solicitud_peaton_b por 1 segundo
0000 100

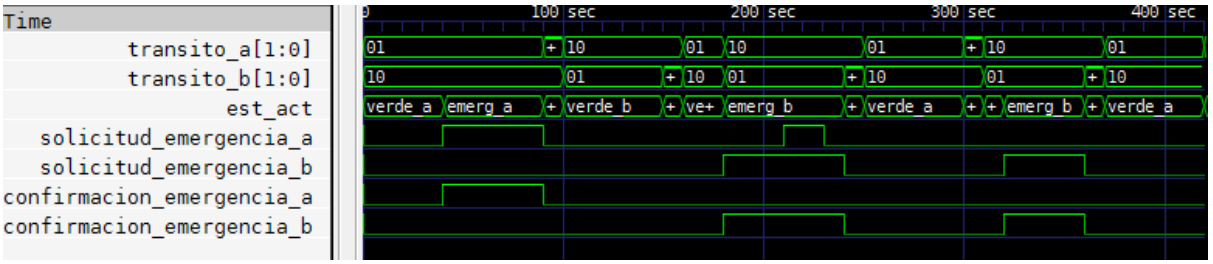


Se observa como luego de pedir la solicitud se guarda en la confirmación hasta ser atendido el pedido de tiempo de verde adicional, luego sigue la secuencia normal de funcionamiento, mientras no reciba una nueva estimulación.

Tercera Implementación, Estado de Emergencia:

Estímulo recibido:
0000 40
0010 50 - - solicitud de emergencia a durante verde a
0000 90

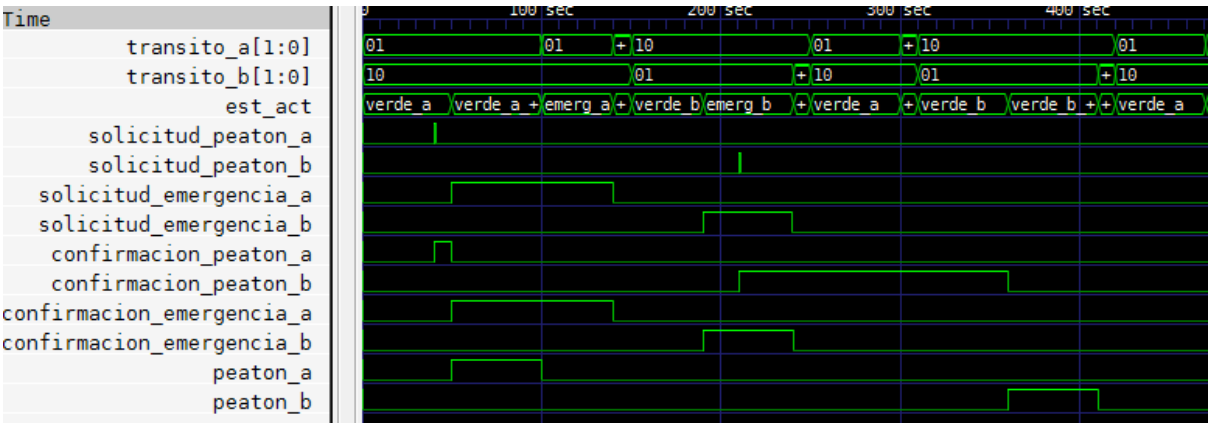
0001 30 - - solicitud de emergencia b
0011 20 - - solicitud de emergencia a y b a la vez durante verde b
0001 10 - - solicitud de emergencia b
0000 80
0001 40 - - solicitud de emergencia b durante un verde de a
0000 10



Observamos que la implementación funciona correctamente para los estímulos dados de acuerdo al estado de emergencia.
Cuando está en verde en una dirección y hay un pedido de emergencia en esa dirección, continua en verde hasta que termina el pedido.
Si ya está en modo emergencia en una dirección no habilita la señal de la otra dirección.
Si está en verde de una dirección y hay un pedido de emergencia en otra dirección, pasa inmediatamente al estado de emergencia del pedido.

Caso donde hay pedido peatonal y pedido de emergencia:

Estímulo recibido:
0000 40
1000 1 - - pedido peatonal a
0000 9
0010 90 - - solicitud emergencia a, durante tiempo verde adicional a
0000 30
0100 1 - - pedido peatonal b
0000 19
0010 50 - - solicitud emergencia a
80000 20



Primero hay un pedido de emergencia durante el tiempo de verde adicional en a, pero no responde, pues debe esperar a que termine el tiempo adicional y recién responder al pedido de emergencia.
Luego en el segundo caso, cuando recibe el pedido del peatón y se encuentra en estado de emergencia, debe esperar otro ciclo de semáforo antes de que el mismo responda al pedido.

El comportamiento observado coincide con las especificaciones. Las máquinas auxiliares (Prescaler y Timer) simplificaron la implementación del controlador principal y facilitaron la verificación modular.

Conclusiones

De acuerdo con lo desarrollado en la actividad y al comparar los distintos métodos de implementación de máquinas de estado finito, se concluye que no siempre resulta conveniente emplear una única máquina compleja. Integrar toda la lógica en una sola entidad puede aumentar significativamente la dificultad de diseño y generar posibles conflictos o errores difíciles de detectar y depurar.

En cambio, el uso de una máquina factorizada permite distribuir el trabajo en varias sub máquinas o máquinas esclavas más simples, de modo que la máquina principal solo gestiona las señales de entrada y salida de las demás. Esta estrategia facilita el desarrollo modular, la comprensión del funcionamiento y la posibilidad de reutilizar componentes.

En síntesis, esta actividad permitió comprender la importancia del diseño jerárquico en sistemas digitales y la utilidad de las máquinas factorizadas para mejorar la legibilidad, depuración y escalabilidad de los proyectos.