ELO212: Laboratorio de Sistemas Digitales
Semestre I-2020
Guía de Actividades Sesión 6

7, 8, 10 de Julio de 2020

1. Objetivos

Los objetivos para esta sesión son:

- Revisar y experimentar con los conceptos de descripción e implementación de máquinas de estado utilizando SystemVerilog y Vivado (ya visto en cátedra).
- Experimentar con la descripción de circuitos sincronizadores y anti-rebotes.

2. Actividades

Al comenzar las actividades de esta sesión se asume que ya ha completado todo el trabajo (actividades y estudio asociado) de las sesiones anteriores.

El trabajo de esta sesión estará enfocado en los aspectos prácticos de la descricpión de máquinas de estado utilizando SystemVerilog. El material base asociado ya se revisó en horario de cátedras y se encuentra disponible en Piazza. Utilice este material como guía para estudiar en forma independiente y definir estrategias de descripción que le acomoden. Para un repaso general, se recomienda revisar el capítulo 4 del libro *"Digital Design and Computer Architecture"* de Harris & Harris. **Recuerde siempre aplicar lectura crítica**.

2.1. Revisión de códigos de ejemplo y templates de máquinas de estado.

En el repositorio Github del curso se encuentran disponibles distintos ejemplos de descripción de máquinas de estado, incluyendo ejemplos completos y templates con *snippets* de código en SystemVerilog que puede utilizar como base para describir sus propias máquinas de estado. Revise cuidadosamente estos códigos y haga las consultas que considere necesarias sobre su utilización.

2.2. FSM para conversor nivel a pulso y controlador de semáforo.

En la cátedra asociada a descripción de máquinas de estado usando SystemVerilog se utilizaron dos ejemplos: (i) un detector de canto o conversor de nivel a pulso, cuyo código base está descrito en las slides; y (ii) un semáforo básico, cuyo código base esta disponible en la carpeta `semaforo_FSM` del repositorio Github del curso (este esta basado en un ejemplo descrito en el libro de Harris).

Para cada uno de estos ejemplos, realice las siguientes actividades:

- Analice, describa y simule la funcionalidad básica de ambas máquinas de estado. En particular, familiaricese con la forma de verificar y depurar el comportamiento de las máquinas de estado visualizando los diagramas de tiempo generados mediante testbenches. En este contexto considere que el reloj base del sistema opera en *unidades de tiempo* arbitrarias.
- Revise los reportes de síntesis y analice cuidadosamente todos los warnings e informaciones asociadas al paso de su descripción HDL al netlist. En particular, ponga especial atención a los mensajes asociados a la inferencia de sus máquinas de estado, e identifique la codificación exacta (la cadena de bits asignada) que la herramienta definió para sus estados. Modifique el código para ver los efectos de los cambios en su descripción en este contexto. Averigue distintas formas de forzar una codificación específica de sus estados utilizando SystemVerilog y opciones de configuración específicas de Vivado[1].
- Revise el datasheet de la tarjeta e identifique periféricos de entrada/salida que puedan servir para implementar la funcionalidad requerida por sus máquinas de estado. En el caso del semáforo, identifique que periféricos le permitirían representar los colores estandar de un semáforo, y estudie su funcionamiento.
- Modifique sus máquinas de estado para incorporar salidas que se conecten a los periféricos identificados en el punto anterior. Añada los constraints necesarios para conectar las salidas del módulo a los periféricos correspondientes en la tarjeta de desarrollo.

2.3. Sincronizador y debouncer para pulsadores mecánicos.

Antes de realizar esta actividad, investigue sobre los fenómenos asociados a la captura de las señales generadas por pulsadores mecánicos que se utilizan como entradas a sistemas digitales. En general, la captura de estas señales (normalmente activadas por usuarios humanos) requiere de etapas de sincronización y filtro de ruido o rebotes.

Una vez entendidos los conceptos asociados, obtenga desde el repositorio del curso los archivos de la carpeta `PushButton_debouncer`, la cual contiene un proyecto con módulos que implementan la función de sincronización y filtro de señal para las señales digitales generadas por un pulsador mecánico típico.

Analice cuidadosamente el código basado en descripción por FSM. A partir del código, identifique las entradas y salidas, obtenga el diagrama de estados (el diagrama debe estar formalmente correcto), y analice la funcionalidad de los distintos parámetros del módulo. Considerando que el reloj base de la máquina de estados es de 100 Mhz, determine un valor adecuado del parámetro `DELAY`

[1] ¿Cuáles son las opciones de codificación que ofrece Vivado?

para que pueda capturar correctamente la presión de un botón de la tarjeta de desarrollo (el valor por defecto de `DELAY` es intencionalmente pequeño para facilitar la simulación funcional, pero no funcionará en la práctica).

La máquina de estados para este sistema incorpora el concepto de *timed FSM*, en la cual la transición de estados se realiza luego de cumplirse un tiempo predefinido. Esto también muestra el concepto de *factorizacion de máquinas de estado*, con el cual el diseño de una máquina de estados potencialmente compleja se puede simplificar interconectando dos máquinas mucho más simples. En este caso se tiene una máquina de estados principal que determina la secuencia a seguir en base a valores de los pulsos de entrada, y un contador secundario (que también puede representarse por una máquina de estados) cuya salida se conecta como entrada a la máquina principal y permite especificar tiempos en los que se debe mantener cierto estado. Tratar de integrar ambas funcionalidades en un solo diagrama de estados se vuelve bastante complejo (haga la prueba). La carpeta de templates de FSM provista en el repositorio también incluye un snippet genérico que sirve como base para describir este tipo de máquinas.

Como referencia, también se provee una descripción alternativa funcionalmente equivalente, pero que no utiliza máquinas de estado. Analice y compare ambos códigos. Realice una simulación funcional para verificar que ambas descripciones son equivalentes (mismas salidas para mismos valores de entrada). Notar que no necesariamente una forma de describir la funcionalidad requerida es mejor que otra en términos de la implementación final, pero si hay una que es más intuitiva y auto-descriptiva que la otra.

2.4. Modificación controlador de semáforo.

En la descripción de la funcionalidad para el semáforo entregada como ejemplo en el repositorio, se asume que el valor de los sensores de entrada se mide en cada ciclo de reloj para decidir las transiciones de estado. Por simplicidad, se trabaja en unidades de tiempo, las cuales pueden ser arbitrariamente largas (por ejemplo, puede considerar que el reloj tendrá un período en el orden de varios segundos).

Para esta actividad, asuma que el reloj de la máquina de estados opera a 100 MHz. Utilizando como base la descripción de una *timed FSM*, modifique la máquina de estados que describe el semáforo para que el valor de los sensores se samplee cada 5 segundos. Es decir, la máquina debe permanecer en cada estado un mínimo de 5 segundos.

2.5. Base de máquinas de estado para notación polaca inversa.

A partir del diseño de la ALU con circuitos retentores obtenido en la guía anterior, describa una máquina de estados que permita realizar el ingreso de datos siguiendo la notación polaca inversa. Considere que el usuario solo puede manipular el bus de datos de entrada (mapeado a los 16 switches de la tarjeta de desarrollo) y cuenta con un único botón ENTER para el ingreso de información (el reset y reloj van por defecto). El botón ENTER se usa para moverse en secuencia por una máquina de estados, la cual setea las señales correspondientes para la ALU.

Luego de describir y validar su diseño mediante simulación funcional, considere las siguientes especificaciones técnicas para la implementación física:

- La máquina de estados opera con un reloj de 100MHz.
- La condición para el cambio de estados será la detección de un **pulso de presión del botón BTNC limpio (sin rebotes) y sincronizado con el reloj de la máquina de estados**. Reconfigure y adapte el módulo revisado en la actividad anterior si lo considera necesario.
- En todo momento, los LEDs deben indicar el valor codificado en binario del estado en el que se encuentra la máquina. Revise los resultados de síntesis para ver si la codificación de estados concuerda con lo esperado.
- Como señal de reset utilice un pulso limpio y sincronizado que indique que el botón *CPU_Reset* ha sido presionado. Revise cuidadosamente la documentación de la tarjeta para utilizar este botón.