ELO212: Laboratorio de Sistemas Digitales
Semestre I-2020
Guía de Actividades Sesión 3

26, 27, 29 de Mayo de 2020

1. Requisitos de entrada.

Para las actividades de esta sesión debe cumplir con los siguientes requerimientos:

- Haber completado las actividades de las sesiones anteriores.
- Entender el funcionamiento de los multiplexores y decodificadores binarios.
- Entender el funcionamiento e implementación de contadores sincrónicos simples.
- Haber revisado las slides y material asociado a las reglas de uso de asignaciones bloqueantes y no bloqueantes en los bloques `always_comb` y `always_ff` de SystemVerilog.

2. Objetivos.

Los objetivos para esta sesión son:

- Definir templates de código para describir funcionalidad de bloques combinacionales genéricos.
- Entender la diferencia entre bloques procedurales `always_comb` para lógica combinacional y `always_ff` para lógica secuencial sincrónica.
- Experimentar con la descripción de contadores sincrónicos con distinta funcionalidad.
- Experimentar con la descripción y uso de módulos parametrizados.
- Descripción de sistemas basados en *Time Division Multiplexing (TDM)* para compartir recursos limitados.

3. Actividades Guiadas.

3.1. Uso de bloques procedurales para lógica combinacional y secuencial sincrónica.

El profesor hará una breve revisión sobre el uso de los bloques procedurales `always_comb` y `always_ff`, junto con entregar algunas indicaciones sobre su uso en la descripción de código sintetizable utilizando asignaciones bloqueantes y no bloqueantes. Para efectos de las actividades de la sesión, es importante que entiendan y apliquen las reglas de uso de estos bloques. Los aspectos formales se discutirán más en detalle en la cátedra común del día Jueves.

3.2. Multiplexor y decodificador binario.

Escriba un código en SystemVerilog basado en descripción procedural que describa la funcionalidad de un multiplexor y un decodificador binario. Para el decodificador binario, considere la configuración de N entradas y 2^N salidas, en donde el valor en las entradas especifican cual de las salidas tendrá un valor alto, mientras que todas las demás salidas se mantienen en bajo (configuración one-hot)[1].

Describa la funcionalidad de ambas componentes en un solo archivo de diseño. Prepare un testbench que permita realizar una simulación exhaustiva para verificar la funcionalidad. Muestre sus resultados y discuta con el staff (profesor/ayudantes).

3.3. Contadores.

Describa un módulo parametrizado para un contador *free-run* de N-bits, donde N es un parámetro que es posible especificar al momento de instanciar el módulo en un módulo de más alto nivel. La idea es que utilizando el mismo código, sea posible instanciar contadores con distinto ancho de bits. El módulo debe recibir una señal de reloj y una señal de reset, y dar como salida el valor actual de un contador interno de N-bits. El contador interno se incrementa en una unidad con cada canto positivo de la señal de reloj. Si la señal de reset se encuentra en alto al momento de ocurrir un canto de reloj, entonces el valor del contador se vuelve a 0 (reset sincrónico).

Una vez descrito el módulo, verifique su funcionalidad mediante un testbench. Muestre sus resultados y discuta con el staff. Manifieste todas sus dudas, hallazgos fundamentales, u observaciones.

Una vez entendido el funcionamiento y depurada la descripción del contador básico, úselo como template para agregar las variaciones indicadas a continuación. Para cada caso, analice el resultado del *Elaborated Design*, realice simulaciones para verificar funcionalidad, y recuerde verificar los warnings y errores, asegurándose que no hayan latches inferidos.

- Averigue como implementar un contador con un reset asincrónico y compárelo con el contador de reset sincrónico. Investigue y analice las ventajas y desventajas de cada implementación.
- Implemente un nuevo contador con una entrada adicional `enable`. Si `enable` está en alto al momento de llegar un canto positivo de reloj, entonces el contador se incrementa normalmente; en caso contrario, el contador no se incrementa y mantiene el valor anterior.

[1] Pueden revisar el libro guia de Harris o la descripción en Wikipedia como referencia rápida.

- Al contador del punto anterior, agregue las entradas `load` de 1 bit y `load_value` de N bits. Si la entrada `load` esta en alto al llegar un canto positivo de reloj, entonces el contador se actualiza con el valor indicado en la entrada `load_value`. Si la entrada `load` esta en bajo, al llegar un canto positivo de reloj, entonces el contador funciona normalmente.

- Al contador del punto anterior, agregue una nueva señal `dec` de un bit para especificar el modo de operación del contador. Si la entrada `dec` está en alto al momento de llegar un canto de reloj, entonces el contador opera en forma decremental (restando una unidad al valor actual). Si la entrada `dec` está en bajo al momento de llegar un canto de reloj, entonces el contador opera en forma incremental.

3.4. Revisión diseños de referencia.

En la sección Resources de Piazza, esta disponible un enlace llamado *Material de referencia SystemVerilog*, el cual dirige a una página de material educacional de Xilinx con buenos templates de diseño.

En el enlace, revise la seccion *Modeling registers and counters*, y compare los resultados con los diseños realizados por Ud. en la actividad anterior.

En general, se recomienda revisar en detalle esa página y tenerla como referencia, sobretodo para las primeras experiencias. Se invita además a que se adelanten en los contenidos.

3.5. Multiplexación temporal.

Revise en el datasheet de la tarjeta Nexys4DDR/NexysA7 los detalles de configuración y operación de los displays de 7 segmentos.

Una vez entendido el detalle del funcionamiento, diseñe y describa un sistema basado en un contador de 16 bits que muestre en los displays el valor instantáneo del contador en base hexadecimal. Notar que un número de 16 bits se traduce en cuatro símbolos hexadecimales, uno por cada grupo de cuatro bits.

Para lograr el objetivo, debe investigar sobre el principio de *Time Division Multiplexing* (TDM). Una vez entienda el principio, haga un diagrama de alto nivel que muestre las componentes principales de su sistema y como estarían interconectadas. Como base, su sistema necesita multiplexores, decodificadores, y contadores. Muestre y discuta el diagrama con el staff, explicando lo que asumen y las consideraciones de diseño para su propuesta. Una vez que su diseño haya sido aprobado por alguien del staff, proceda a describir su circuito y a verificar su comportamiento funcional.

Se enfatiza que a partir de ahora en adelante, el diagrama de alto nivel de un diseño será requisito para su revisión. En general, no deberían molestarse en escribir una línea de código si no tienen un diagrama previo. Por parte del staff, no se responderán consultas ni se entregará calificación en actividades evaluadas si no muestran este diagrama.

4. Actividades adicionales.

4.1. Clock divider.

Describa un módulo basado en contadores que a partir de un reloj de entrada genere como salida un reloj (o tren de pulsos) de una frecuencia menor a la del reloj de entrada. La frecuencia de salida debe estar especificada por medio de un parámetro que se puede definir al momento de instanciar el módulo.