ELO212: Laboratorio de Sistemas Digitales
Semestre I-2020
Guía de Actividades Sesión 2

12, 13, 15 de Mayo de 2020

1. Requisitos de entrada

Para las actividades de esta sesión considerar lo siguiente:

- Haber terminado las actividades de la sesión previa y tener dominio sobre tareas basicas de descripcion y simulacion utilizando SystemVerilog y Vivado.
- Estudiar como funciona un flip-flop tipo D y un latch. Entender a la perfección la diferencia entre uno y otro elemento.
- Tener dominio de representaciones de números en distintas bases, en particular bases binarias y hexadecimal con números fijos de bits.

2. Objetivos

Los objetivos para esta sesión son:

- Reforzar los conceptos fundamentales sobre descripción y simulación de hardware revisados en la sesión anterior.
- Estudiar la diferencia entre distintos niveles de abstracción en el diseño.
- Estudiar las asignaciones continuas y procedurales en SystemVerilog.
- Desarrollar un sistema complejo utilizando los conceptos de modularidad, jerarquía y regularidad.

3. Actividades Guiadas

3.1. Revisión de reconocedor de números fibinarios de 4 bits.

En Piazza se encuentra un link a un repositorio Github que contiene diseños base y ejemplos para el curso. En ese repositorio, busque la carpeta `fib_example_project`, la cual contiene un

Figura 1: Diagrama de alto nivel de conversor BCD a 7 segmentos.

projecto de Vivado con dos descripciones del reconocedor de números fibinarios de 4 bits diseñado y simulado en la sesión anterior. Ambas descripciones son funcionalmente equivalentes (para una misma entrada generan los mismos resultados), pero ofrecen un distinto grado de abstracción en la descripción, inferirán distinto hardware y tendrán distinto desempeño una vez implementados.

Descargue el proyecto del repositorio y analice y simule las descripciones incluidas. Asegúrese de que entiende todo sobre la descripción, simulación, estructura del proyecto, e interpretación de los resultados[1]. Discuta con el staff y haga todas las preguntas que considere necesarias. Se enfatiza que en esta etapa no hay preguntas demasiado simples o demasiado tontas.

El profesor dará una breve introducción sobre como visualizar, leer, e interpretar los mensajes y reportes de la herramienta para distintas etapas del diseño.

Recuerden que SystemVerilog es un estándar que cumple dos propósitos que son fundamentalmente distintos: **(i)verificación y (ii) descripción para síntesis lógica**. En la simulación o testbenches se provee un entorno virtual para instanciar y **simular** la funcionalidad del sistema descrito. El entorno de simulación se ejecuta en el computador y permite usar gran parte del lenguaje. Sin embargo, la descripción de hardware para síntesis lógica es mucho más restrictiva. En este punto, el profesor explicará las diferencias fundamentales entre los entornos de descripción para síntesis lógica y simulación.

3.2. BCD y display 7-segmentos.

Lea el datasheet de la tarjeta de desarrollo para determinar como operar uno de los displays de 7 segmentos. Luego de enteder el funcionamiento, describa la funcionalidad de un circuito que reciba como entrada un número de 4 bits codificado en BCD y muestre en el display su símbolo equivalente en decimal.

El diagrama de entradas y salidas de alto nivel se muestra en la Figura 1, y un snippet de código en SystemVerilog se muestra a continuación:

```
module BCD_to_sevenSeg (
    input  logic [3:0] BCD_in,
    output logic [6:0] sevenSeg
    );

    always_comb begin
        case(BCD_in)
            4'd0:    sevenSeg = 7'b1111110; // output is abcdefg
            4'd1:    sevenSeg = 7'b0110000;
            default: sevenSeg = 7'b0000000;
        endcase
    end
```

[1]Saber leer, generar e interpretar diagramas de tiempo es una de las habilidades fundamentales de cualquier cualquier ingeniero electrónico o telemático que se digne a llamar ingeniero electrónico o telemático.

```
endmodule
```

Complete el código anterior para que se pueda visualizar en el display los números decimales ingresados en codificación BCD 8421, y siga los pasos para ver lo que infiere la herramienta mediante el *Elaborated Design*.

Realice la sintesis lógica de su diseño y verifique todos los mensajes que entrega la herramienta en terminos de *info*, *warnings*, y *errors*. Hay muchos warnings que son completamente aceptables y solo son informativos, mientras que otros pueden ser fatales para su diseño (y los posibles usuarios de su sistema si este ejecuta una actividad crítica).

Un regla de oro en este curso será: **su diseño nunca deberá inferir latches!**. Si su diseño contiene un latch, no se le proveerá ayuda ni se le revisarán las actividades evaluadas.

Elabore un testbench que permita una verificación funcional exhaustiva de su diseño.

3.3. Contador de 4 bits.

Cree un nuevo archivo en su proyecto y copie el siguiente código. Verifique que su código no contiene errores de sintáxis, y observe lo que infiere la herramienta utilizando el Elaborated Design. Este circuito es fundamentalmente distinto a todo lo que hemos hecho hasta ahora. Para entender su funcionamiento es muy importante que entiendan a la perfección como funciona un flip-flop tipo D.

```
module counter_4bit(
    input  logic        clk, reset,
    output logic [3:0]  count
    );

    always_ff @(posedge clk) begin
        if (reset)
            count <= 4'b0;
        else
            count <= count+1;
    end
endmodule
```

Simule el comportamiento utilizando el siguiente testbench.

```
module test_counter();

    logic       clk, reset;
    logic [3:0] count;

    counter_4bit DUT(.clk(clk), .reset(reset), .count(count));

    always #5 clk = ~clk;  //Generacion senal de reloj

    initial begin
        clk = 0;
        reset = 1;
        #10 reset = 0;
    end
endmodule
```

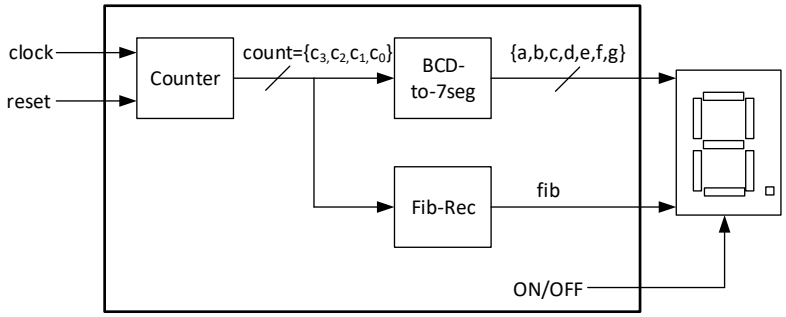

Figura 2: Diagrama del sistema a implementar durante la sesión.

3.4. Contador de N bits.

Busque información sobre como parametrizar el código del contador para describir un número de N bits utilizando parámetros de SystemVerilog. Estudie las ventajas asociadas al uso de módulos parametrizables.

Implemente un testbench en el cual debe instanciar multiples instancias de un mismo módulo utilizando distintos valores para N en cada instancia. Visualice los resultados en diagrama de tiempo, y explore las opciones de la herramienta de simulación para facilitar la visualización de los resultados (por ejemplo, agrupación de señales).

3.5. Diseño modular de reconocedor de número fibinarios.

La Figura 2 presenta un diagrama de alto nivel del sistema a implementar en esta actividad, el cual sigue una estructura modular y jerárquica que integra en un solo diseño los módulos diseñados y probados individualmente hasta la actividad anterior. Implícitamente se forzó la estrategia (*divide and conquer*), en la cual se resuelve un problema por medio de la resolución de problemas más pequeños. En esta ocasión, ya teníamos resueltos los problemas más pequeños, por lo que nuestro gran problema se reducirá a conectar piezas ya diseñadas y probadas.

Se enfatiza que antes de molestarse en escribir una línea de código, es necesario realizar al menos un diagrama de alto nivel del sistema que se quiere describir e implementar, incluyendo los módulos principales, su funcionalidad, y su interconexión (pines de entrada/salida). Es absolutamente necesario realizar un diagrama de alto nivel del sistema, identificando cuales serán los módulos principales a diseñar. Sin embargo, no es absolutamente necesario que este sea perfecto y definitivo. En general, el proceso de diseño es iterativo, y a partir de la implementación pueden encontrar detalles que les permitan ir refinando la descripción de su sistema. Omitir el paso previo de planear, analizar y discutir sus diseños es un camino seguro a dolores de cabeza y frustración en etapas posteriores.

Ponga en práctica lo aprendido para describir un *top_module* que instancie e integre cada uno de los módulos descritos y probados anteriormente (como el sistema de la Figura 2). Use la visualización del Elaborated Design para que vea la estructura del proyecto y como la entiende la herramienta. Compare lo que entrega le herramienta con el diagrama entregado como base.

Genere un testbench para probar la funcionalidad del sistema. Discuta con el staff y responda las

preguntas que se le harán. Explore las opciones de la herramienta para visualizar señales externas e internas a los módulos.

3.6. Discusión y actividades adicionales.

Las actividades desarrolladas en esta sesión son extremadamente simples y a la vez extremadamente fundamentales e ilustrativas para facilitar su trabajo en el resto del semestre. Es muy importante que clarifiquen todas sus dudas cuanto antes, por muy simples que estas parezcan.

Se vuelve a enfatizar que aplicar los conceptos de modularidad, regularidad y jerarquía se volverán cada vez más relevantes a medida que nos movemos en capas de abstracción superiores y los sistemas se vuelven más complejos.

Cualquier diseño digital, por muy complejo que sea, se basa en los mismos principios planteados y evaluados en esta sesión. En general, después de adquirir un poco de experiencia, la mayor dificultad se encontrará en como plantear un buen diagrama de alto nivel para identificar los módulos necesarios, y la descripción en sí se vuelve un proceso más mecánico, donde deberá maximizar el uso de módulos ya probados.

También es muy importante documentar el proceso de diseño. Si durante la implementación encuentran que es necesario hacer algún cambio al diagrama inicial (faltó una señal, hay que dividir un módulo en más componentes, etc.), este debe quedar documentado. Esto les permitirá aprender de sus errores e ir acumulando experiencia.

Dependiendo del tiempo que tomen las actividades anteriores, se utilizará el resto de la sesión para discusión y ver actividades adicionales.