ELO212: Laboratorio de Sistemas Digitales
Guía de Actividades Sesión 2

10 al 14 de abril de 2023

1. Requisitos de entrada.

- Haber completado las actividades de la sesión anterior y dominar lenguaje técnico mínimo asociado a diseño de sistemas digitales y las primitivas lógicas que se utilizaron en dicha actividad.
- Haber estudiado las slides de referencia de SystemVerilog disponibles en el repositorio de material (ver enlace asociado en Aula). Idealmente, contar con material complementario sobre detalles más específicos de sintaxis y ejemplos de diseño con SystemVerilog (quedará material de referencia en el mismo repositorio).
- Entender a la perfección como funciona un flip-flop tipo D y un latch tipo D, y la diferencia fundamental entre estos elementos.
- Previo a trabajar en la guía, revisar la cápsula de video *capsula_sesion_2.mp4*, disponible en el repositorio del curso. Esta cápsula contiene información crítica para entender algunos conceptos que se usarán a lo largo del semestre y que profundizaremos en las siguientes sesiones.
- Leer el datasheet con información técnica sobre la tarjeta de desarrollo Nexys4DDR/NexysA7 de Digilent.

2. Objetivos.

- Reforzar los conceptos fundamentales sobre descripción y simulación de hardware revisados en la sesión anterior.
- Entender la diferencia entre distintos niveles de abstracción en el diseño usando SystemVerilog (aunque aplicables a HDLs en general).
- Familiarizarse con el proceso de síntesis lógica y análisis e interpretación de sus resultados.
- Diseño de componentes acorde a especificaciones y requerimientos técnicos.
- Desarrollar un sistema complejo utilizando los conceptos de modularidad, jerarquía y regularidad.

3. Actividades guiadas.

3.1. Tarjeta de desarrollo Nexys4DDR/NexysA7.

Revise la documentación técnica sobre la tarjeta de desarrollo Nexys4DDR/NexysA7 [1] en base al siguiente enlace: `https://digilent.com/reference/programmable-logic/nexys-a7/start`. También quedará información en el repositorio del curso.

La tarjeta de desarrollo indicada es fabricada por la empresa Digilent, y está basada en un chip FPGA fabricado por AMD-Xilinx. El término tarjeta de desarrollo se refiere a la integración en una placa del chip principal de procesamiento, en este caso una FPGA, con diversos periféricos orientados a aplicaciones específicas.

Si bien durante las primeras sesiones no tendrán acceso físico a la tarjeta, todos los diseños estarán orientados a este hardware, pues es necesario establecer restricciones para dirigir a la herramienta en el proceso de diseño. Durante la segunda mitad del curso tendrán la tarjeta a disposición para implementar y validar físicamente los diseños que ya fueron sintetizados, depurados, y verificados funcionalmente mediante simulación de comportamiento.

3.2. Introducción a síntesis lógica.

El video de cápsula introductoria a la sesión explica en forma general el proceso de síntesis lógica y como interpretar y analizar sus resultados usando Vivado. Utilizando este material como base, investigue y discuta con su grupo (y en lo posible con otros grupos) la diferencia entre el esquemático obtenido mediante **Elaborated Design** (technology agnostic) y **Logic Synthesis** (technology/platform dependant).

El profesor dará una breve introducción sobre este tema para complementar lo mencionado en el video. **Entender estos procesos será crítico para un paso exitoso por el curso. Haga todas las consultas sobre este tema que considere necesarias**.

3.3. BCD y display 7-segmentos.

Estudie el datasheet o manual de referencia de la tarjeta de desarrollo Nexys4DDR/NexysA7 para determinar como operar uno de los displays de 7 segmentos. Luego de entender las especificaciones y requerimientos, describa la funcionalidad de un circuito que reciba como entrada un número de 4 bits codificado en BCD 8421 y muestre en el display su símbolo equivalente en decimal. **Fíjese cuidadosamente en los requerimientos de señales eléctricas, en particular las polaridades, para hacer funcionar el display.** La Figura 1 muestra un diagrama de entradas y salidas para el bloque requerido, y un snippet de código en SystemVerilog para la funcionalidad se muestra a continuación:

```
1 module BCD_to_sevenSeg (
2     input  logic [3:0] BCD_in,
3     output logic [6:0] sevenSeg
4     );
```

[1] Ambas tarjetas son equivalentes. Solo hubo un cambio de nombre y algunas actualizaciones de hardware que no son relevantes para el curso. Posteriormente en el curso cada grupo tendrá acceso a una de ellas.

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

```
5
6       always_comb begin
7           case(BCD_in)
8               4'd0:     sevenSeg = 7'b1111110; // output is abcdefg
9               4'd1:     sevenSeg = 7'b0110000;
10                .
11                .
12                .
13          endcase
14      end
15  endmodule
```

Complete el código anterior para que se pueda visualizar en el display los números decimales ingresados en codificación BCD 8421. Una vez haya entendido y completado su descripción, realice un esquemático en papel y lápiz para el circuito que esperaría implementara la función. Siga los pasos para ver lo que infiere la herramienta mediante Elaborated Design, y compare lo que infirió Ud, con lo que entrega la herramienta de diseño. **El contar con diagramas de alto nivel hechos a mano son un paso crucial de diseño, y de ahora en adelante se le solicitarán como requisito para responder consultas.**

Realice la sintesis lógica de su diseño y verifique todos los mensajes que entrega la herramienta en términos de *info*, *warnings*, y *errors*. Hay muchos warnings que son completamente aceptables y solo son informativos o *bugs de la herramienta de diseño*, mientras que otros pueden ser fatales para su diseño (y los posibles usuarios de su sistema si este ejecuta una actividad crítica, como el sistema de autodestrucción de un misil o el freno automático de un automóvil).

Revise cuidadosamente los reportes de síntesis. Una regla de oro en este curso será: **su diseño nunca deberá inferir latches!** Los latches salen claramente reportados en el resultado de síntesis, y para efectos de este curso serán considerados como veneno. Si su diseño contiene un latch, no se le proveerá ayuda ni se le revisarán las actividades evaluadas. En el caso de que una actividad evaluada infiera un latch, esta se considerará automáticamente incorrecta sin mayor análisis.

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

3.4. Bloques `always` en SystemVerilog.

Utilice como base las slides sobre descripción de hardware usando SystemVerilog para estudiar la descripción de comportamiento utilizando los bloques `always_comb` y `always_ff`. Complemente lo indicado en las slides buscando referencias adicionales sobre la sintaxis y uso de dichos bloques.

3.5. Contador de 4 bits.

Cree un nuevo archivo en su proyecto y tipee el siguiente código. Verifique que su código no contiene errores de sintaxis.

```
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
```

Observe y analice el esquemático obtenido a partir del Elaborated Design. Este circuito es fundamentalmente distinto a todo lo que hemos hecho hasta ahora. Para entender su funcionamiento es necesario que **entiendan a la perfección como funciona un flip-flop tipo D**.

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
```

Luego de entender la simulación, vuelva a revisar el esquemático con Elaborated Design, y compárelo con el esquemático generado a partir de la síntesis lógica.

3.6. 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. El circuito a diseñar consta de un contador del tipo *free-run* de cuatro bits, el cual incrementa su cuenta con cada canto positivo de la señal de reloj que proviene del exterior. El valor del contador se pasa como entrada a dos bloques: uno que convierte el valor instantáneo del contador de su representación binaria de 4 bits (notar que en este caso coincide con la representación en códigos BCD 8421) a su representación en 7-segmentos; y un

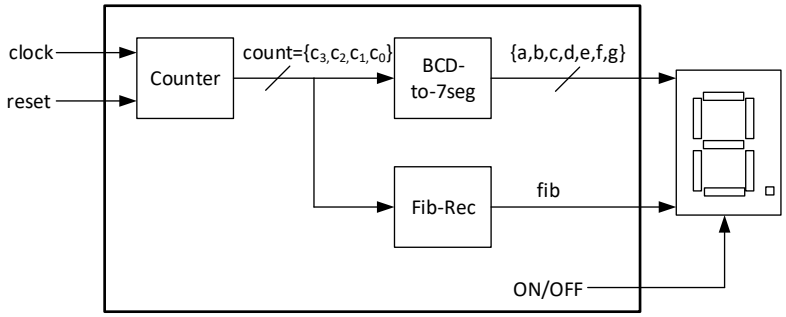

Figura 2: Diagrama del sistema a implementar durante la sesión.

segundo circuito que determina si el mismo valor instantáneo del contador corresponde a un número fibinario o no. Las salidas de estos dos circuitos se conectan a uno de los displays de la tarjeta de desarrollo, el cual mostrará el símbolo numérico del valor del contador, prendiendo el punto decimal del display en el caso que el valor actual corresponde a un número fibbinario. Además, se agrega una señal para encender o apagar el display completo. Notar que es un diagrama genérico que no considera los detalles eléctricos del display, los cuales debe observar para su tarjeta en particular.

Notar que implícitamente, **mediante las actividades anteriores que ya tiene resueltas y entendidas**, 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. En sesiones posteriores, este será un proceso que Uds. deberán realizar por su cuenta.

Se enfatiza que antes de molestarse en escribir una línea de código, se realice 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 en *papel y lápiz* es un camino seguro a dolores de cabeza, frustración, y peleas entre integrantes de grupo 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 la herramienta con el diagrama entregado como base. Sintetice su diseño, revise los reportes de uso de recursos, y estudie el esquemático generado post-síntesis.

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 simulación para visualizar el comportamiento de señales externas e internas a los módulos.

Haga todas las consultas necesarias. Recuerde que arrastrar dudas a sesiones posteriores puede acarrear problemas y retrasos.

3.7. Contador de N bits (módulos parametrizables).

Busque información sobre el uso de módulos parametrizables en SystemVerilog. En base a lo estudiado, parametrice el código del contador de la actividad anterior para que el número de bits del contador sea N, donde N es un valor ingresado al momento de instanciar. Implemente un testbench en el cual debe instanciar múltiples instancias de un mismo módulo utilizando distintos valores para N en cada instancia (considere al menos N=4, N=5, N=8). 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).

¿Cuáles son las ventajas asociadas al uso de módulos parametrizados?

3.8. 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 volverá cada vez más relevantes a medida que nos movamos en capas de abstracción superiores y los sistemas se vuelvan 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.