Apellidos, Nombre: _____

Circuito Secuencial Síncrono

Circuitos Electrónicos Digitales
Ingeniería Informática. Tecnologías Informáticas
Dpto. de Tecnología Electrónica
Diciembre, 2022

1 Descripción y objetivos

En esta práctica tiene como principal objetivo el diseño de circuitos secuenciales síncronos.

Se trabajan los siguientes conceptos y competencias con respecto a los mismos

- Diseño y análisis
- Descripción en lenguajes de descripción de hardware
- Simulación lógica
- Implementación en lógica reconfigurable

La práctica se estructura a partir de una descripción en inicial que es proporcionada por el profesor a modo de ejemplo. A partir de es ejemplo, se proponen varios proyectos adicionales que amplían o se basan en el ejemplo inicial. El alumno debe realizar el proyecto basado en el ejemplo inicial y realizar uno o varios de los proyectos adicionales.

2 Sistema digital de ejemplo

Se desea que un circuito secuencial síncrono que controle una tira de luces de un árbol de navidad. La tira tiene ocho luces numeradas de 0 a 7. Para ello el circuito tiene un bus de salida *Z* con una línea por cada una de las luces. Cada luz se enciende si y sólo si la el bit correspondiente de *Z* vale 1. El circuito dispone de una entrada inicialización asíncrona denominada *reset*. También dispone de otra entrada denominada *enable*. Si *enable* vale 0 el circuito no cambiará de estado de manera síncrona. Se proporciona una descripción inicial del circuito en Verilog.

3 Material y documentación

3.1 Software y hardware

- Instrumentación del laboratorio de Electrónica Digital.
- Software para simulación de circuitos electrónicos a a partir de su descripción en Verilog (EDA Playground o Icarus Verilog)
- Ordenador con entorno Xilinx ISE instalado.
- Placa de desarrollo Digilent Basys2.
- Archivos iniciales de diseño (kit de laboratorio).

3.2 Documentación

- Vídeos explicativos.
- Manual del instrumental básico de laboratorio de la asignatura
- Documentación del tema de circuitos secuenciales

4 Primera parte: Estudio y simulación

Actividades a realizar:

1. Descargue los ficheros de diseño y simulación proporcionados.
2. El fichero '`generator.v`' contiene la descripción inicial del circuito. Lea e interprete su contenido.
3. Indique razonadamente si el circuito descrito implementa una máquina de Moore.
4. Escriba su tabla de transición escribiendo los valores de los estados en binario natural, es decir, no los escriba en el orden del código Gray. Para ello no considere la señal de inicialización asíncrona *reset* como una entrada.
5. Escriba su tabla de salida escribiendo los valores de los estados en binario natural, es decir, no los escriba en el orden del código Gray.
6. Utilice el testbench contenido en el fichero '`generator_tb.v`' para simular el comportamiento de la descripción inicial del circuito y compruebe que funciona de acuerdo a las tablas que ha escrito.

5 Segunda parte: Implementación del circuito en FPGA

5.1 Creación de un proyecto en Xilinx ISE

- Inicie el entorno Xilinx ISE. La ventana del entorno se divide en tres secciones: la zona izquierda es el panel del proyecto y sirve para gestionar los archivos y módulos del proyecto y las acciones que se pueden realizar sobre los mismos; la zona derecha es el panel principal

que da acceso a la edición de los elementos del proyecto, como los archivos Verilog; la zona inferior es la consola, donde podemos ver los mensajes de error y avisos de las herramientas del entorno conforme se realizan las operaciones sobre el diseño.

- Es posible que al iniciar ISE se abra automáticamente el último proyecto editado. Si es así, cierre el proyecto desde File→Close Project.
- Cree un nuevo proyecto desde File→New Project. Se abrirá el asistente para creación de proyectos. Escriba un nombre para el proyecto como 'luces'. Puede elegir la carpeta para el nuevo proyecto si lo desea. Pulse NEXT.
- Ahora introduzca los detalles del tipo de chip FPGA en que se sintetizará el proyecto. Para la placa Basys2 los detalles son:
  - General Purpose
  - Family: Spartan3E
  - Device: XC3S100E
  - Package: CP132
  - Speed grade: -5
  - El resto de opciones del proyecto no deben modificarse: XST synthesis, ISIM simulator,Verilog preferred language, etc.). Pulse NEXT.
  - Revise el resumen de opciones del proyecto y si no hay errores pulse FINISH.

5.2 Añadir archivos al proyecto

- Para añadir una copia de los archivos previamente elaborados al proyecto haga click derecho sobre el panel de la jerarquía del diseño (Hierarchy) y elija "Add Copy of Source".
- Seleccione los archivos '`generator.v`', '`generator_tb`' y '`Basys2_100_250General.ucf`'. Puede elegir varios archivos a la vez con `Ctrl`+*click*. ISE usará estos archivos para implementar y/o simular el diseño. Puede elegir la finalidad de cada archivo manualmente, pero normalmente ISE detectará la función de cada archivo automáticamente y sólo tendremos que confirmar.
- Ahora, el panel de diseño muestra los archivos que forman el proyecto. Hay dos vistas: implementación (Implementation) y simulación (Simulation). La vista de implementación muestra los archivos con el diseño que se configurará en la FPGA. La vista de simulación muestra, además, los archivos con los bancos de prueba que sólo se emplean en la simulación.
- Haciendo doble click sobre cualquier archivo, éste se abre en el panel central y puede ser editado.
-

5.3 Simulación del banco de pruebas

Aunque en la primera parte de esta práctica ya se ha realizado una simulación del circuito, no está demás realizar una simulación en el entorno ISE previa a la implementación. Para ello haga lo

siguiente:

Seleccione la vista de simulación y seleccione el módulo *test* que se encuentra en el archivo ‘`generator_tb.v`’.

- Observe como el panel jerárquico organiza el diseño en función de los módulos que contiene, independientemente de en qué archivo se definen (aunque indica el nombre del archivo entre paréntesis).
- En el panel de procesos (Processes) aparecen las acciones que se pueden ejecutar sobre el módulo seleccionado en el panel de vistas del diseño. Al seleccionar el módulo *test* aparecen las acciones asociadas al simulador ISim. Aquí puede seleccionar comprobar la sintaxis del código Verilog (Behavioral Check Syntax) o directamente simular el módulo del banco de pruebas (Simulate Behavioral Model). Haga doble click sobre esta última opción. Si hay errores en el diseño tendrá que editar el código y volver a ejecutar la simulación.
- Si el código es correcto, tras unos segundos se abrirá la ventana del simulador ISim. En el panel principal verá el código Verilog simulado y una marca donde se ha detenido el simulador. Elija la pestaña con las formas de onda, busque los botones de selección de ampliación (zoom) y use el botón que muestra todo el tiempo simulado. Observe las señales y compruebe que el resultado es correcto.
- Por defecto, ISim simula un tiempo de 1000ns o bien hasta que encuentre la directiva Verilog $finish. Desde el menú “Simulation” o mediante los botones de control de la simulación sobre el panel principal puede continuar la simulación, detenerla o reiniciarla, pero no es necesario para este ejercicio.
- El formato en que se muestran las señales puede modificarse desde el menú contextual que aparece al hacer click derecho sobre cualquier señal. Por ejemplo, pruebe a mostrar la señal de 4 bits *q* en formato decimal sin signo: Radix → Unsigned Decimal.
- Es muy útil guardar la configuración de la visualización de las señales para futuras simulaciones. Elija en el menú File → Save As... y guarde la configuración con un nombre significativo como ‘`generator.wcfg`’.
- Salga del simulador. De vuelta en la ventana de ISE, vamos a configurar el proceso de simulación para que cargue automáticamente la configuración de ondas que hemos salvado. Para ello, haga click derecho sobre el proceso “Simulate Behavioral Model” y elija “Process Properties.”. En la ventana que aparece, busque la opción “Custom Waveform Configuration File” y busque el archivo de configuración de ondas ‘`generator.wcfg`’. Esto facilitará mucho ver los nuevos resultados de simulación cuando volvamos a ejecutar el simulador.

5.4 Editar el fichero UCF para poder realizar la implementación

El chip de la FPGA de la placa Basys2 tiene sus pines conectados a varios periféricos: botones, interruptores, LEDs, display 7 segmentos, etc. Para poder implementar el diseño es necesario mapear las entradas y salidas de nuestro diseño a los pines adecuados del dispositivo FPGA de manera que queden en periféricos útiles de la placa. Un mapeado correcto enlazaría entradas con interruptores o botones y salidas con LEDs. El fichero ‘`Basys2_100_250General.ucf`’ contiene la información de este mapeado y es necesario editarlo de acuerdo a las necesidades de

nuestro diseño y la placa con la FPGA que se esté utilizando.

Para este circuito utilizaremos el interruptor 7 para controlar la entrada *reset* y el botón 3 para controlar la entrada *enable*. Conectaremos las 8 líneas de salida a los LED. También debemos controlar la entrada de reloj. La placa tiene generadores de reloj que en principio podrían utilizarse, pero su frecuencia es muy alta y haría imposible percibir el apagado y encendido de los LED. Por ese motivo controlaremos la entrada *clk* con el botón 0. En general, las señales de reloj deben suministrarse al sistema por pines (IOB) especiales para señales de reloj, pero nosotros conectaremos esta señal con un botón de la placa que está conectado a un IOB convencional. Como esto es en general un error de diseño, la herramienta se detiene en la fase de implementación. Nuestro diseño es muy simple y no hay problema en tener el reloj en un pin convencional. Podemos indicar a la herramienta que ignore esta restricción en el archivo UCF. Para ello edite el archivo UCF y copie la siguiente línea tras la línea de asignación de la señal *clk*:

```
NET "clk" CLOCK_DEDICATED_ROUTE = FALSE;
```

5.5 Síntesis e Implementación

- Seleccione el módulo a implementar en el panel de diseño. En el panel de procesos aparecen las tres acciones principales del proceso completo de síntesis: *Synthesize - XST*, *Implement Design* y *Generate Programming File*.
- Haga click derecho sobre *Generate Programming File* y seleccione *Run*. Esto generará el archivo de configuración de la FPGA, pero también ejecutará todos los procesos anteriores que sean necesarios. Observe el icono animado que indica la ejecución de cada proceso.

  Si todo va bien, aparecerá una marca verde o amarilla junto a cada proceso. Si algún proceso falla, aparecerá una marca roja y se detendrá la síntesis. En este caso, puede consultar los errores en la consola inferior, en la pestaña *Errors*. Mire si hay errores, lea los mensajes de error e intente interpretarlos.

5.5.1 Implementación del diseño en la FPGA

Como resultado de los procesos anteriores, se ha generado un archivo con la configuración (bitstream) que hay que cargar en el dispositivo FPGA. Esta configuración está un el archivo con extensión ‘`.bit`’ guardado en la carpeta del proyecto. El proceso de configuración consiste simplemente en grabar el contenido de ese fichero en la memoria de configuración de la FPGA. El proceso es sencillo, pero existen varias alternativas para hacerlo, dependiendo del sistema operativo donde estemos usando ISE:

1. Emplear la herramienta de configuración del entorno ISE (iMPACT). Disponible en MS-Windows y GNU/Linux.
2. Emplear la herramienta independiente *Adept* suministrada por el fabricante de la placa Basys2. Disponible en MS-Windows.
3. Emplear el comando de configuración `djtgcfg` suministrado por el fabricante de la placa. Disponible en GNU/Linux.

Con cualquiera de las opciones, tendremos el diseño programado en la placa y podremos probarlo.

En caso de que la operación no sea la correcta debemos comprobar todo el proceso de diseño:

- Revise si las herramientas han dado errores o avisos en algún momento del proceso.
- Compruebe que la asignación de señales a pines de la placa son correctos en el archivo UCF.
- Compruebe si los resultados de la simulación son correctos. Si no hemos hecho un banco de pruebas y simulado previamente el diseño, quizá sea hora de hacerlo.

> ☞ **Avise al profesor para mostrarle el funcionamiento del circuito.**

5.6 Modificaciónes

5.6.1 Habilitación controlada por interruptor

Modifique el fichero UCF para que la señal *enable* esté controlada por el interruptor 0 en lugar de por un botón. Implemente dicha modificación en la FPGA y pruébela.

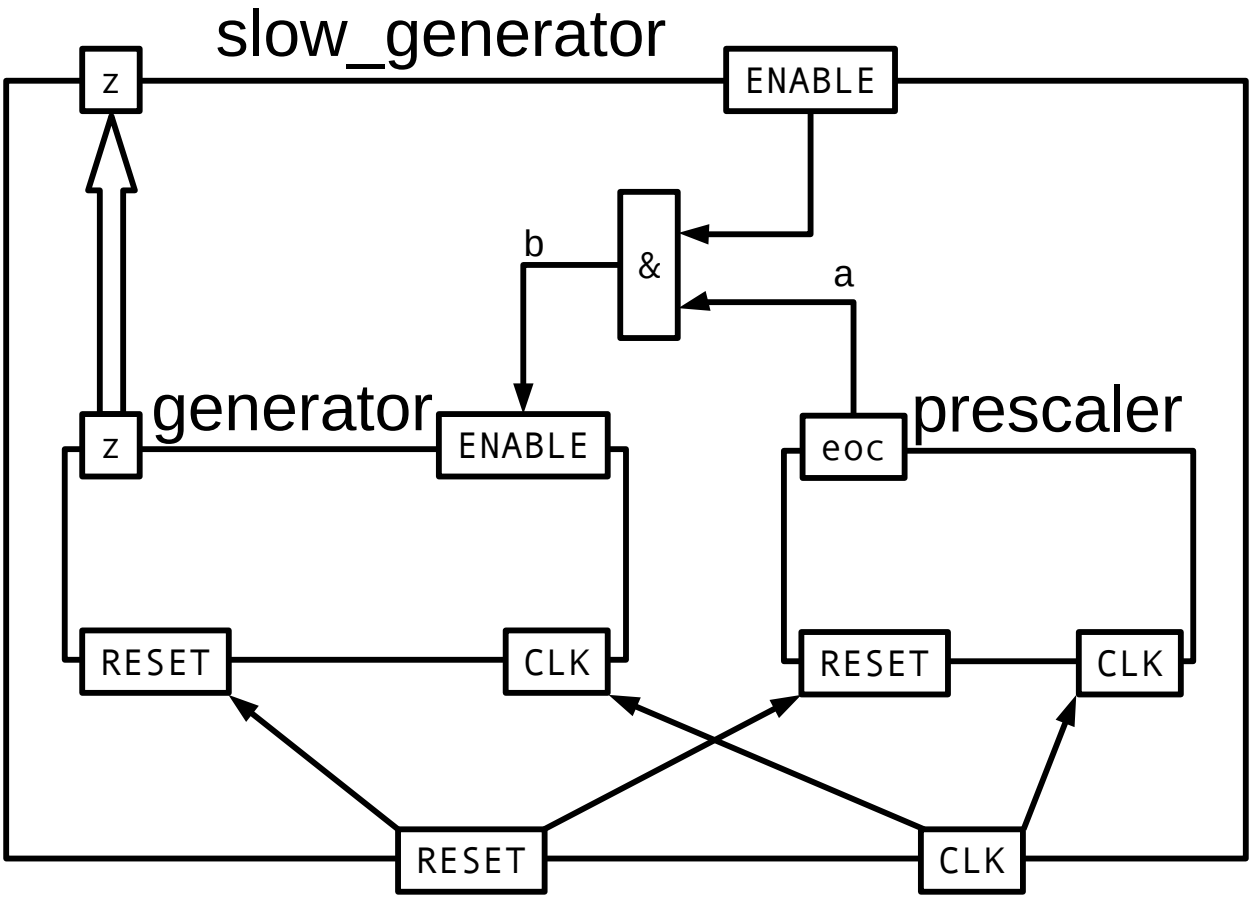
> ☞ **Avise al profesor para mostrarle el funcionamiento del circuito.**

5.6.2 Reducción de la frecuencia de operación haciendo uso de un contador

Como indicamos anteriormente, si conectamos la entrada de reloj *clk* al generador de la placa las luces parpadearán tan rápido que no podremos apreciarlo. Para que el estado de las luces cambie una vez por segundo haremos que la señal *enable* se active durante un sólo ciclo de reloj por segundo. Con ese propósito usaremos el diseño del contador descrito en el fichero '`prescaler.v`'. A parte de la entrada de reloj *clk* dispone de una entrada de inicialización asíncrona *reset* y una salida de fin de cuenta *eoc* que se activa únicamente en el último estado del contador. Su número de estados de cuenta el igual a la frecuencia de la señal cuadrada generada por la placa en hercios. Por ello, si conectamos su entrada de reloj a dicha señal cuadrada su salida *eoc* se activará durante un sólo ciclo y sólo una vez por segundo, es decir, tendrá el comportamiento que deseamos para la entrada *enable* de nuestro circuito de control de los LED. Así, nuestro nuevo diseño -al que denominaremos *slow_generator*- contendrá dos subsitemas:

- Una instancia del módulo de control de LED que probamos anteriormente denominado *generator*.
- Una instancia del contador que usaremos para generar la entrada *enable* de la instancia anterior denominado *prescaler*.

El sistema completo tendrá la siguiente estructura:



Para implementarlo haga lo siguiente:

- Añada al proyecto los ficheros '`prescaler.v`' y '`slow_generator.v`' que contienen respectivamente la descripción del contador y el módulo principal.
- Edite el fichero '`slow_generator.v`' y completelo.
- Edite el archivo UCF para conectar la entrada *clk* al IOB de la FPGA que está conectado al generador de señal cuadrada de la placa. Dicho IOB, denominado *B8*, sí está preparado para recibir y distribuir internamente señales de reloj, por lo que puede eliminar o comentar la siguiente línea:

  ```
  NET "clk" CLOCK_DEDICATED_ROUTE = FALSE;
  ```

- Implemente y pruebe el diseño en la FPGA.

> ☞ **Avise al profesor para mostrarle el funcionamiento del circuito.**

6 Cuarta parte: proyectos adicionales

En esta parte el alumno deberá realizar la simulación e implementación en la FPGA de un circuito secuencial síncrono que genere una secuencia de encendido de luces distinta a la propuesta.