Electrónica Digital
Grado de Robótica

Prof: Juan J Pombo

Sesión 3. Prácticas de HDL (Verilog)

Práctica 6

Título: Simulación básica de proyectos Verilog con EDA Playground
Tiempo estimado : 45min

Objetivos:

Manejar Verilog como herramienta para describir hardware de circuitos digitales.
Comprender el procedimiento elaboración y verificación de diseños usando un lenguaje de descripción de hardware como Verilog.

Introducción:

Como primera actividad vamos a probar varios aspectos del lenguaje Verilog tanto a nivel de diseño del circuito como del *testbench* para testeo. Los desarrollaremos en primera instancia utilizando el simulador de Verilog online EDA Playground. Empezaremos por circuitos y pruebas de funcionamiento muy simples para conocer el entorno.

Tareas :

1.- Probar, para empezar, exclusivamente el módulo de testeo (panel izquierdo). Escribir código en él utilizando funciones del sistema como $display, asignaciones a variables y retardos de tiempo. Revisar el ejemplo https://www.edaplayground.com/s/example/352. Verifica que se graban los diseños (opción "save").

2.- Abrir y analizar el ejemplo incorporado del flip-flop tipo D incluído en los ejemplos EDA Playground.
Entender para que se usa `$dumpfile("dump.vcd");` (ver documentación incorporada en EDA)
Fijarte en la estrategia que se usa para tener un "log" adecuado por pantalla llamando a una "task" y así verificar el funcionamiento del diseño.
Probar la salida de formas de onda usando el módulo EPWAVE.

3.- Modificar el código del ejemplo para hacer un flip-flop tipo JK y probar el resultado elaborando un *testbench* adecuado.

Documentación de apoyo:

Antes de comenzar:
Si aún no se ha hecho, para comenzar, ver el vídeo introductorio al uso del entorno EDA recomendado en clase.

Ver la documentación para utilizar las funciones de sistema de depuración: $display, $write, $strobe, $monitor.

Notas

Como herramientas de simulación el entorno EDA tiene múltiples posibilidades. Aquí utilizaremos **Icarus Verilog** (seleccionable en el panel de la izquierda), que es un compilador freeware que además podemos instalar en nuestro PC personal como herramienta de aprendizaje.

Práctica 7

Título: Utilizar diseño modular en Verilog.
Tiempo estimado : 1h

Objetivos:

Utilizar las características de Verilog para definir hardware en forma de bloques funcionales (*module*) y ver cómo estos bloques se instancian múltiples veces y se reconectan para hacer circuitos más complejos.

Introducción :

Para ilustrar este caso diseñaremos y verificaremos un Multiplexor 4:1 pero partiendo de un bloque multiplexor más elemental, el mux 2:1.

Un multiplexor 4:1 es capaz de seleccionar 1 de 4 casos utilizando 2 bits de selección. De forma intuitiva, esta operación de selección podría interpretarse de otra manera. Al dividir el conjunto de 4 casos seleccionables en 2 grupos de 2 casos cada uno, primero podríamos seleccionar en qué grupo se encuentra el caso que queremos conectar a la salida. En una segunda fase seleccionaríamos cual de los 2 casos de cada grupo es el implicado.

Esta forma de ver el multiplexor 4:1 equivale a una conexión de multiplexores en cascada.

Tareas :

1.- Diseñar sobre el papel, a nivel de bloques, el circuito mux 4:1 a partir de bloques 2:1.

2.- Crear y verificar en EDA un bloque funcional MUX 2:1. El bloque ha de ser genérico, con sus variables de entrada y salida, de forma que posteriormente pueda ser insertado como subcircuito en otras partes del diseño.
Existe la opción de crearlo como *task* pero en este caso es preferible que simplemente sea un módulo separado.

Nota: En un fichero pueden convivir varias definiciones de módulo. Aunque así es una práctica habitual definirlos en ficheros diferentes. Si se desea pueden añadirse más módulos a un proyecto EDA, añadiendo pestañas tanto en el lado de diseño como en el de test.

3.- Crear un segundo diseño donde probaremos el módulo 4:1. Ver las notas generales de apoyo a esta práctica más abajo para conocer el procedimiento de instanciado.
Generar un fichero de ondas y analizarlo para asegurar el buen funcionamiento del mismo.

Notas generales de apoyo.

Múltiples módulos pueden ser "instanciados" con una aproximación como la siguiente:

```
1
2  module one (input I, output O);
3    assign O = I;
4  endmodule
5
6  module two (input I, output O);
7    assign O = I;
8  endmodule
9
10 module top (input I, output O);
11
12   wire W;
13
14   one inst1 (.I(I), .O(W));
15   two inst2 (.I(W), .O(O));
16
17 endmodule
18
```

Práctica 8

Título: Máquinas de estados (FSM) en Verilog. Aplicación a una cerradura electrónica.
Tiempo estimado : 2h

Objetivos:

Expresar en Verilog un circuito secuencial correspondiente a una máquina de estados.

Introducción :

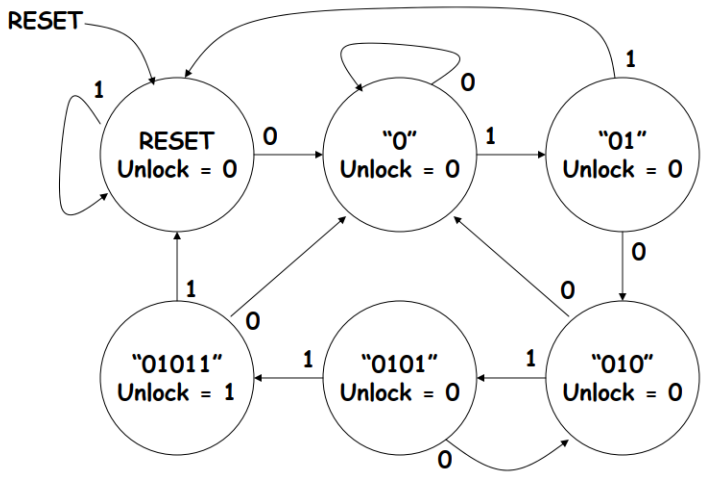
En primer lugar analizaremos el ejemplo de una FSM incluído en EDA (https://www.edaplayground.com/x/B) para comprender una posible aproximación al problema. Después aplicaremos el ejemplo a un diseño particular.

Tareas :

1.- Analizar el ejemplo indicado y comprender su funcionamiento.

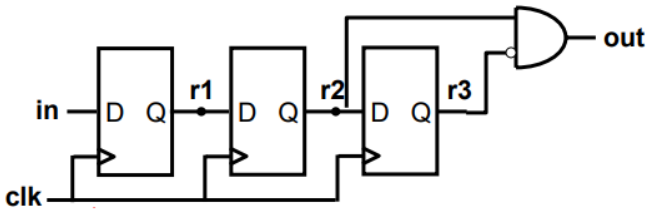
2.- Como ejemplo vamos a construir una cerradura electrónica simplificada, capaz de abrir una puerta con un código introducido por teclado.
Supondremos que el número de dígitos posibles del código es solamente 2 (un teclado binario que solo tiene las teclas 0 o 1), y que la longitud total del codigo es 5. Tambien tendremos una tecla RESET para borrar el código introducido en caso de error. Si se teclea el código correcto se dará un señal para abrir la puerta.
Lo realizaremos para un código fijo. En la máquina de estados que se presenta aquí se utiliza el código 01 011.

a) Analizar el sistema secuencial a partir de la siguiente máquina de estados



b) Esbozar la arquitectura del sistema con los bloques funcionales necesarios

c) El teclado debe funcionar de una manera tal que al mantener pulsado un botón no introduzca constantemente ese código. Esto requiere de algunos trucos a nivel electrónico. Se propone para ello el siguiente circuito. Probar su funcionamiento en Logisim e interpretar qué operación hace (incluir la explicación en la memoria de prácticas).



d) Una vez tengamos clara la arquitectura comenzar a diseñarlo en EDA Playground basándonos en el ejemplo que hemos analizado anteriormente, junto con su banco de pruebas.

Sugerencia: Además de la salida de desbloqueo de pestillo sería interesante inicialmente incorporar un indicador en el *testbench* de si se han teclado 1, 2, 3... números desde la última solicitud de aceptación. Esto nos facilitará además la depuración.