

# Laboratorio #4: Verilog Estructural

Escuela de Ingeniería Eléctrica

Universidad de Costa Rica

## 1 Introducción

### 1.1 Objetivos del Laboratorio

Este laboratorio está diseñado para que las personas estudiantes implementen circuitos digitales utilizando Verilog estructural, una metodología que permite describir sistemas digitales a nivel de compuertas lógicas. A través de esta práctica, se busca que las personas estudiantes comprendan la conexión entre las funciones lógicas, su implementación en Verilog y la programación en hardware mediante FPGAs.

A través de esta actividad, los estudiantes lograrán:

- Implementar circuitos digitales básicos en Verilog estructural.
- Simplificar funciones lógicas utilizando álgebra booleana para su implementación eficiente.
- Crear módulos en Intel Quartus Prime Lite para verificar su funcionamiento lógico.
- Programar una FPGA para probar el diseño en hardware físico.
- Diseñar sistemas digitales a partir de problemas utilizando módulos en Verilog

## 2 Conceptos teóricos necesarios

### 2.1 Lenguajes de Descripción de Hardware

Los lenguajes de descripción de hardware (HDL, por sus siglas en inglés) son herramientas fundamentales en el diseño digital moderno. Estos lenguajes permiten describir el comportamiento y la estructura de circuitos digitales, desde sistemas simples como compuertas lógicas hasta diseños complejos como microprocesadores o sistemas embebidos. A diferencia de los lenguajes de programación tradicionales, los HDLs no solo especifican cómo se ejecuta una tarea, sino cómo se interconectan los componentes de hardware para realizar esa tarea.

Algunas de las principales características de los lenguajes de descripción de hardware son las siguientes:

- **Concurrencia:** Los HDLs permiten modelar circuitos que operan de manera concurrente (en paralelo, todo al mismo tiempo), reflejando el comportamiento del hardware real.
- **Jerarquía:** Los circuitos pueden describirse utilizando módulos independientes que luego se interconectan, facilitando la reutilización y la organización del diseño.
- **Simulación y Síntesis:** Los HDLs permiten verificar el comportamiento del diseño mediante simulaciones, donde mediante observaciones de las formas de onda obtenidas al ejercitar el circuito con cierto estímulo se puede validar si el comportamiento es esperado. Además, a partir de las descripciones de circuitos en HDL, es posible generar la descripción a nivel de compuertas para su implementación física en hardware programable, como FPGAs, o en hardware fijo, como ASICs.

Uno de los principales lenguajes de descripción de hardware, y el más común en la industria, es Verilog. Este lenguaje tiene un enfoque similar a lenguajes como C, y permite describir tanto la estructura como el comportamiento de los circuitos digitales. Más recientemente, Verilog fue extendido a lo que hoy se conoce como SystemVerilog, el cual incorpora características avanzadas como mejores capacidades de verificación y orientación a objetos.

Como se mencionó anteriormente, Verilog se puede escribir de dos formas:

1. **De forma estructural:** Donde se describen los componentes lógicos que constituyen el diseño, así como su interconexión.
2. **De forma conductual:** Donde, de forma más abstracta, se describe el comportamiento del circuito, sin entrar en detalles de su estructura. Esto permite el desarrollo de circuitos más complejos, pues facilita la descripción de circuitos complejos.

En este laboratorio, el enfoque será **únicamente** en Verilog Estructural.

## 2.2 Verilog Estructural

Es un paradigma de diseño dentro del lenguaje Verilog que describe los circuitos digitales a través de sus componentes básicos (como compuertas lógicas, memorias, etc) y sus interconexiones, imitando la manera en que se construiría físicamente el hardware. Este enfoque es ideal para modelar sistemas jerárquicos y comprender cómo interactúan los bloques de hardware en un diseño más complejo.

En este paradigma los diseños se construyen a partir de módulos básicos que representan componentes específicos, como compuertas AND, OR, y XOR, y son conectados para formar estructuras más grandes. De esta forma, la descripción del circuito diseñado es completamente explícita.

La principal ventaja de este paradigma es que permite visualizar cómo se implementará el circuito real en hardware, pues las conexiones son explícitas. Sin embargo, este método es más complejo para diseños con comportamientos complejos.

Para más detalles sobre la sintaxis y semántica de Verilog estructural, refiérase al material compartido por su profesor sobre el tema, disponible en Mediación Virtual.

## 3 Ejemplo introductorio - Divisor por 2

**Recuerde que el dispositivo que se usará es el Max 10 10M50DAF484C7G.**

En esta sección se presenta el desarrollo de un módulo sencillo para mostrar la clase de situaciones que pueden ser modeladas por medio de un sistema digital y la secuencia de pasos para implementar dicho sistema. También se presentarán los pasos requeridos para incorporar uno o más módulos de Verilog en los proyectos de Quartus Prime para utilizar dichos módulos en la FPGA de la placa de desarrollo y cómo simular los módulos en Quartus Prime, previo a programar la FPGA.

### 3.1 Planteamiento del Problema

Se desea desarrollar un circuito que, por medio de los switches de la tarjeta de desarrollo, reciba un número entre el 0 y el 15. El módulo deberá detectar si el número es par o no. En caso de ser par, se deberá encender uno de los LEDs de la tarjeta, caso contrario deberá mantenerse apagado.

Como la entrada del módulo será un número del 0 al 15, se tienen 16 posibles casos para las entradas de este módulo y por ende se requiere de 4 bits para codificar cada una de estas entradas ( $2^4 = 16$ ).

Los números del 0 al 15 que cumplen ser divisibles por 2 son 0, 2, 4, 6, 8, 10, 12 y 14 por lo que estas entradas serán las que produzcan un estado alto en la salida.

Se define la entrada al módulo como los bits ABCD, donde A es el bit más significativo y D el menos significativo. Además, las entradas se codificarán de manera idéntica al número que representan, es decir, el cero será ABCD = 0000, el uno será ABCD = 0001 y así sucesivamente.

A partir de esto se puede definir una tabla de verdad (Tabla 1) para el problema, donde la columna de más a la izquierda representa el número decimal de la entrada al módulo, las columnas “A”, “B”, “C” y “D” la

entrada de 4 bits correspondiente a ese número decimal y la columna “Salida” contiene el valor que deberá adoptar la salida del módulo para cada entrada.

Número Decimal	A	B	C	D	Salida
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	0
14	1	1	1	0	1
15	1	1	1	1	0

Tabla 1: Tabla de verdad del sistema digital correspondiente al ejemplo

De la tabla anterior se puede llegar a que la función lógica que representa este sistema es la siguiente:

$$Salida(A, B, C, D) = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}C\bar{D} + \bar{A}B\bar{C}\bar{D} + \bar{A}BC\bar{D} + A\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D} + AB\bar{C}\bar{D} + +ABC\bar{D}$$

Que se puede simplificar a la siguiente función:

$$Salida(A, B, C, D) = \bar{D}$$

### 3.2 Desarrollo del módulo de Verilog del sistema a implementar

Teniendo claro la función lógica que representa la situación que se desea resolver, se procede a implementarla en un módulo de Verilog. Utilice para esto el editor de texto de su preferencia.

Para este ejemplo, la función lógica se implmentó en un archivo de nombre `div2.v` que contiene un módulo de Verilog denominado `div2` en el cual la salida está asignada como el negado de la entrada `D`, lo cual concuerda con la función lógica obtenida tras la simplificación.

A continuación se presenta el contenido del módulo desarrollado.

```
div2.v
1 //Declaracion del modulo div2
2 module div2 (A, B, C, D, Out);
3 //Declaracion de entradas y salidas
4 input A, B, C, D;
5 output Out;
6
7 //Instanciacion de bloques logicos
8 //Un inversor con Out en la salida y D en la entrada
9 not inv1(Out, D);
10 //Termina modulo
11 endmodule
```

### 3.3 Creación del proyecto de Quartus Prime

En esta sección se presentan los pasos para crear el proyecto de Intel Quartus Prime e incorporarle el archivo que contiene el módulo `div2` desarrollado en la sección anterior.

1. Siga la secuencia de creación de un proyecto de Intel Quartus Prime, tal y como se indicó en la guía anterior. En el paso 7, deberá realizar las siguientes selecciones para la fila “*Simulation*”:
  - “Tool Name”: “*Questa Intel FPGA*”.
  - “Format(s)”: “*Verilog HDL*”.

La configuración requerida también se muestra en la Figura 1.

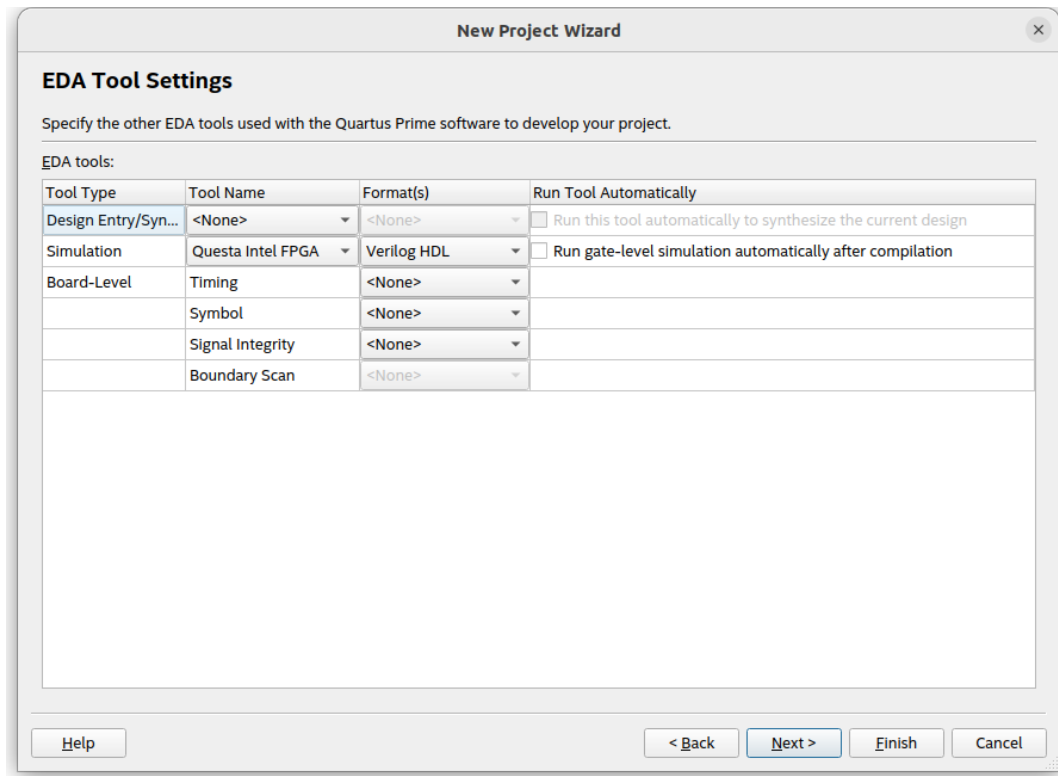


Figura 1: Selección de la opción “Files” en el “Project Navigator”.

Tras esto, finalice la creación del proyecto con normalidad.

2. Seleccione la opción “Files” en el “Project Navigator”, como se muestra en la Figura 2.

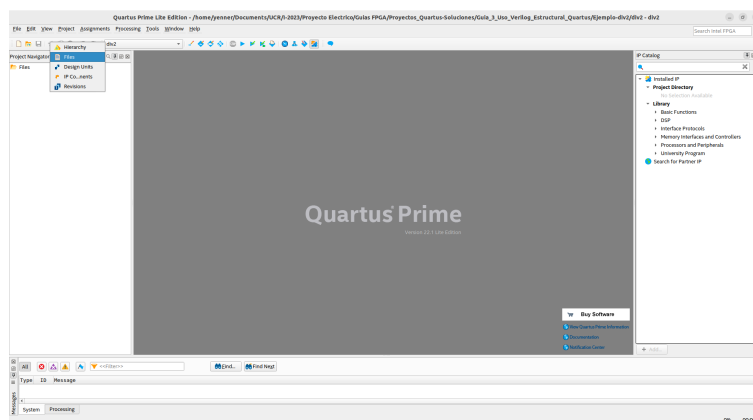



Figura 2: Selección de la opción “Files” en el “Project Navigator”.

3. Haga clic derecho en el ícono  y seleccione la opción “Add/Remove Files in Project...”. Esto hará que se abra el menú mostrado en la Figura 3.

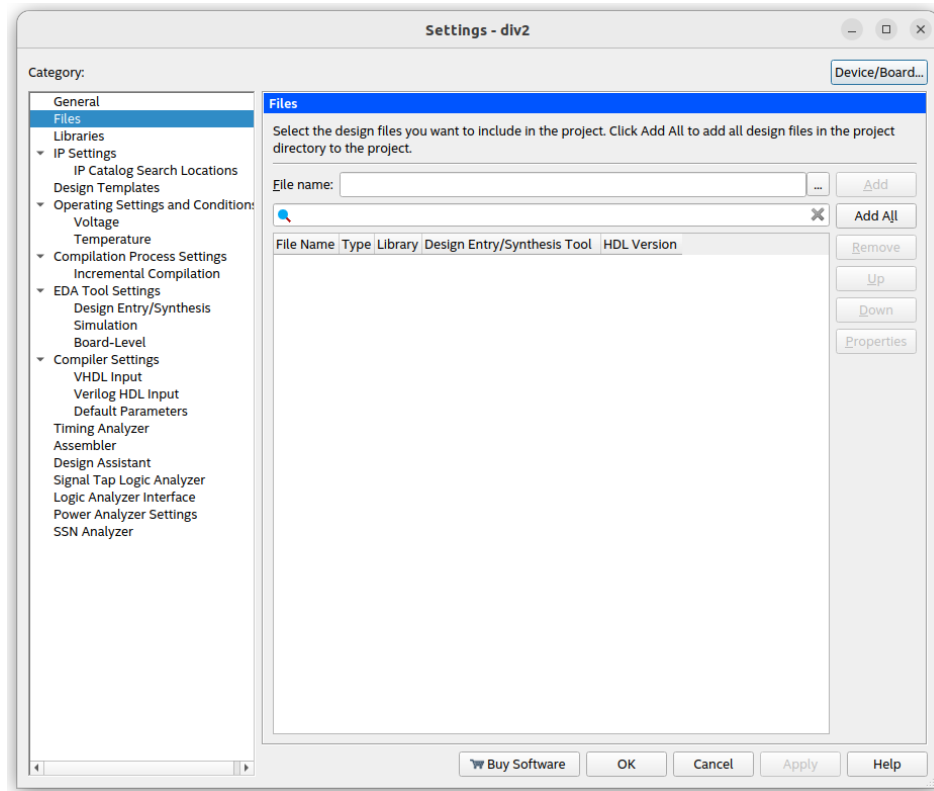


Figura 3: Menú para modificar los archivos de un proyecto de Intel Quartus Prime.

4. Haga clic en el ícono de los tres puntos ubicado a la derecha del cuadro “File name”.
5. Utilice el navegador de archivos para buscar y seleccionar el archivo de Verilog que desea añadir al proyecto. Seleccione “OK” tanto en el navegador como en el menú para modificar los archivos del proyecto.

Tras la realización de los pasos anteriores los archivos seleccionados se deberán haber incluido a su proyecto de Quartus Prime. Esto se muestra en la Figura 4 para un archivo con nombre `div2.v`.

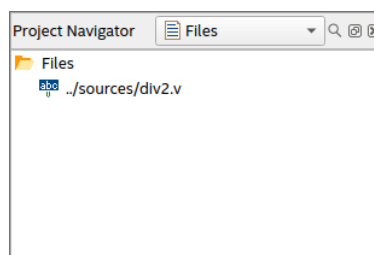



Figura 4: Sección de archivos del “Project Navigator” tras incluir un archivo al proyecto.

Compile el proyecto para verificar que el archivo incluido funciona correctamente, siguiendo el paso 10 de la sección 3.3 de la Guía 2 (Presionando el ícono  o mediante la combinación de teclas **Ctrl+L**). Si la compilación concluye exitosamente el archivo no presenta errores (recuerde que esto no implica necesariamente que el archivo funciona de la manera deseada).

### 3.4 Programación de la FPGA del módulo de Verilog diseñado

6. Programe la FPGA, para ello utilice la siguiente asignación de pines:

Señal	Componente
A	SW3
B	SW2
C	SW1
D	SW0
Salida	LEDRO

Tabla 2: Señales y componentes para el módulo.

7. Modifique el estado de los interruptores asignados y observe los cambios en los LEDs para verificar que el diseño funcione correctamente.

## 4 Trabajo en Clase

Considere los siguientes problemas y desarrolle una representación de la situación descrita que se pueda implementar en la placa de desarrollo para cada uno.

1. Desarrolle un sistema que reciba un número del 0 al 15 por medio de los interruptores de la placa de desarrollo e indique a través de un LED si el número ingresado es divisible por 3. Utilice la misma asignación de pines para el ejemplo.
2. Utilice el ejercicio anterior en conjunto con el ejemplo de esta guía para realizar un sistema que detecte si un número del 0 al 15 es divisible por 6. Utilice la asignación de entradas del problema anterior, para las salidas utilice los LEDs LEDRO, LEDR1 y LEDR2 para indicar si el número ingresado es divisible por 2, por 3 o por 6, respectivamente.
3. Las condiciones de iluminación en una sala de hospital son críticas para el bienestar de los pacientes y el trabajo del personal médico. En una sala se implementó un sistema de control de iluminación que cuenta con los siguientes sensores:
  - Sensor de luz ambiental (L): Produce una salida en alto si la iluminación natural es suficiente.
  - Sensor de presencia (P): Produce una salida en alto si hay personas en la sala (pacientes o personal médico).
  - Control manual (M): Permite activar (salida en alto) o desactivar (salida en bajo) las luces independientemente de las condiciones anteriores.

El sistema debe funcionar bajo las siguientes condiciones:

- Las luces deben encenderse si no hay suficiente luz ambiental y hay personas en la sala.
- Si el control manual está activado, las luces deben encenderse sin importar las condiciones de los sensores.
- Si no hay personas en la sala, las luces deben permanecer apagadas, salvo que el control manual esté activado.

Describa el sistema de iluminación utilizando Verilog estructural. Implementelo en la tarjeta de desarrollo y verifique su funcionamiento. Utilice el LED **LEDRO** para indicar si la iluminación está encendida o no, según las entradas definidas.

**Para la DEMOSTRACIÓN** deberá mostrar los diseños descritos funcionando de forma adecuada. Apenas finalice cada diseño, puede solicitar a la persona docente hacer el demo. La calificación se repartirá de la siguiente manera:

- Divisibilidad por 3: 35%
- Divisibilidad por 6: 25%
- Sistema de Iluminación: 40%