Departamento de Tecnología Electrónica
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Introducción a Verilog y XILINX

Enunciados de Prácticas de Laboratorio
Estructura de Computadores

1. Introducción y objetivos

Uno de los objetivos generales de la asignatura Estructura de Computadores es llegar a conocer la metodología de diseño de sistemas digitales a nivel de transferencia entre registros. Para ello en las clases de teoría y prácticas, que se desarrollan en clases de aula, se introduce dicha metodología y se aplica a un amplio número de casos de diseño de sistemas digitales. Para completar la formación en este tema es importante no quedarnos exclusivamente en la parte más teórica sino que es necesario complementarla con un conocimiento de implementación real de dichos sistemas digitales.

Los objetivos de esta práctica son:

- Familiarizarse con el lenguaje Verilog-HDL
- Conocer el entorno de diseño sobre FPGA, en concreto el entorno de diseño de XILINX[1].
- Conocer las herramientas de verificación del diseño desarrollado.
- Desarrollar el proceso de diseño y simulación. Para ello se han elegido dos circuitos muy simples, uno combinacional y otro secuencial. Concretamente un convertidor de código y un contador.

2. Estudio teórico

Para realizar la sesión de laboratorio es necesario realizar el estudio teórico previo consistente en describir dos componentes en Verilog:

1. Un convertidor de código de binario a siete segmentos

1 Xilinx Inc.: Compañía de desarrollo de FPGAs (www.xilinx.com)

2. Un contador ascendente de 4 bits con varias señales de control

Para ambos circuitos se han preparado unas plantillas de código que se incluidas en este documento. Además, para el desarrollo de la prácticas dispondrá de los siguientes ficheros de texto:

Nombre del fichero	**Contenido**	**Descripción**
convertidor.v	Módulo con el código del convertidor 7 segmentos	Debe completarlo el alumno antes de asistir a la sesión de laboratorio
convertidor_tb.v	Testbench para el convertidor 7 segmentos	Se debe utilizar sin modificar para realizar la simulación
contador.v	Módulo con el contador modulo 16	Debe completarlo el alumno antes de asistir a la sesión de laboratorio
contador_tb.v	Testbench para el contador módulo 16	Se debe utilizar sin modificar para realizar la simulación
lab2.v	Descripción estructural con la unión de los 2 módulos	Debe completarlo el alumno durante la sesión de laboratorio
lab2_tb.v	Testbench del sistema completo	Se debe utilizar sin modificar para realizar la simulación

Tabla 1. Ficheros necesarios durante la sesión de laboratorio.

A continuación se detalla cada uno de los circuitos que se deben desarrollar en Verilog.

2.1. Diseño del convertidor a 7 segmentos

Para poder visualizar números se utilizan displays 7 segmentos. Estos displays tienen 7 entradas, uno por cada segmento que se puede iluminar, de forma que, al iluminar algunos de ellos se puede componer visualmente un número. En la figura 1 se muestran los números desde el 0 al 9.

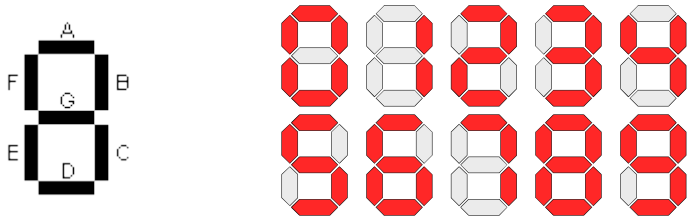


Figura 1. Ejemplo dígitos en un display de 7 segmentos.

Como primer paso para el diseño de este convertidor se presenta la tabla de verdad del circuito combinacional que hay que diseñar (tabla 2). Los nombres de los segmentos en la tabla 2 corresponden a los nombres asignados en la figura 1. La activación de los segmentos es en nivel **bajo,** es decir, para que un segmento se ilumine el valor de la señal debe colocarse a cero. En la tabla se representan los dígitos del 0 al 9 a partir de su valor en binario de 4 bits; también aparecen algunos números mayores de 9 para lo cual se iluminan ciertos segmentos de forma que aparezcan los dígitos hexadecimales.

bin_3	bin_2	bin_1	bin_0	A	B	C	D	E	F	G
0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0
0	0	1	1	0	0	0	0	1	1	0
0	1	0	0	1	0	0	1	1	0	0
0	1	0	1	0	1	0	0	1	0	0
0	1	1	0	0	1	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	1	0	0
1	0	1	0	0	0	0	1	0	0	0
1	0	1	1	1	1	0	0	0	0	0
1	1	0	0	0	1	1	0	0	0	1
1	1	0	1	1	0	0	0	0	1	0
1	1	1	0	0	1	1	0	0	0	0
1	1	1	1	0	1	1	1	0	0	0

Tabla 2: Tabla de verdad de conversión de binario a 7 segmentos.

Para diseñar el código Verilog se propone utilizar la estructura “case” de Verilog y usar como plantilla el siguiente fragmento de código (fichero *convertidor.v*):

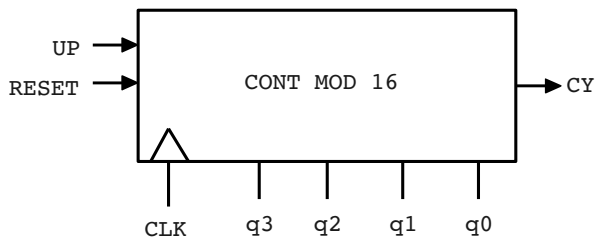
```
module convertidor_bin7seg(
    input  [3:0] bin_in,      // entrada binaria 4 bits
    output reg a,b,c,d,e,f,g);   // salida 7-segmentos

    // Escriba aqui el código
    // Se recomienda utilizar un proceso always con
    // una sentencia case

    ...


endmodule
```

Código 1. Fichero convertidor.v, plantilla de código Verilog para el convertidor 7 segmentos.

2.2. Diseño del contador de 4 bits

Dicho contador será disparado por el flanco de subida del reloj, tendrá una señal RESET síncrona de puesta a cero. Además, incluirá una señal *carry* (CY) que se activará cuando el contador llegue al último estado de cuenta.

El contador debe cumplir la tabla comportamiento de la figura 2b y, para diseñar el código Verilog se propone utilizar como plantilla el fragmento de código 2 (fichero contador.v).



(a)

CLK	RESET	UP	CONT ←
↑	1	X	CONT ← 0
↑	0	1	CONT ← CONT+1
↑	0	0	CONT ← CONT

CY	
0	[CONT] ≠ 1111
1	[CONT] = 1111

(b)

Figura 2. Descripción del contador módulo 16: (a) Descripción estructural, (b) Descripción funcional.

```
module contador_mod16(
  input  clk,up,reset,
  output reg [3:0] q,
  output cy);

  // Escriba su código aqui
  // Cuidado con la señal cy, tiene que
  // activarse durante el ultimo estado de cuenta

   ...

endmodule
```

Código 2. Fichero contador.v, plantilla de código Verilog para el contador módulo 16.

3. Estudio experimental

En esta sesión de laboratorio se va a realizar la simulación del sistema digital diseñado con el paquete de herramientas de Xilinx ISE. Se utilizarán unos ficheros con los *testbench* ya preparados. Los pasos a seguir son los siguientes:

1. Siga los pasos del tutorial de uso de Xilinx ISE, incluido en la siguiente sección, para crear un nuevo proyecto *PracticaEdC2*, incluyendo los ficheros Verilog que ha realizado en el estudio teórico.

2. Realice la simulación del convertidor binario a 7 segmentos siguiendo los pasos descritos en el apéndice, con ayuda del testbench facilitado (fichero *convertidor_tb.v*). Compruebe si el bloque convertidor funciona correctamente.

3. Realice ahora la simulación del contador de 4 bits, con ayuda del testbench facilitado (fichero *contador_tb.v*). Compruebe si el contador funciona correctamente.

4. Fíjese que al simular el contador no aparecen todos los estados ni la activación de la señal de *Carry*. Haga los cambios oportunos para poder visualizar al menos un ciclo completo de cuenta y vuelva a realizar la simulación para comprobarlo.

5. Añada una descripción estructural en Verilog que interconecte ambos bloques: *convertidor* y *contador*, de forma que ambos aparezcan incluidos por este nuevo fichero. Utilice la plantilla de código (véase Código 3) suministrada (fichero *lab2.v*).

6. Realice por último la simulación de este nuevo bloque, con ayuda del testbench facilitado (fichero *lab2_tb.v*). Compruebe si el sistema completo funciona correctamente.

```
module lab2(
   input clk, up, reset,
   output [0:6] seg,
   output cy);

   // Declare un cable para
   // interconectar la salida del contador
   // con la entrada del convertidor

   ...

   // Instancie el contador modulo 16
   // y realice las conexiones correctamente

   ...

   // Instancie el convertidor binario a 7 segmentos
   // y realice las conexiones correctamente con el
   // modulo anterior

   ...

endmodule // lab2
```

Código 3. Fichero lab2.v, plantilla de código Verilog para el contador módulo 16.

4. Tutorial de Xilinx ISE

Esta sección describe el entorno ISE del fabricante XILINX. Este entorno, entre otras características, incluye un simulador para el lenguaje Verilog que será utilizado en esta sesión de laboratorio.

4.1. Creación de un proyecto en Xilinx ISE

Tras iniciarse el sistema operativo, el primer paso es arrancar el entorno ISE y crear un nuevo proyecto. En el menú File hay que utilizar la opción New Project, obteniéndose la ventana mostrada en la figura 3 donde hay que escribir un nombre para el proyecto, por ejemplo *PracticaEdC2*. La herramienta creará una carpeta con ese mismo nombre y guardará en su interior todo lo que se va generando a medida que vamos trabajando en ese proyecto.

Tras escribir el nombre se activa el botón Next y aparece el cuadro de diálogo mostrado en la figura 4, donde hay que establecer todas las opciones indicadas en la figura. Concretamente hay que asegurarse de establecer las siguientes opciones a su valor correcto:

- **Family:** Spartan 3E

- **Device:** XC3S100E
- **Package:** CP132
- **Preferred Language**: Verilog

El resto de opciones deberían estar por defecto a los mismos valores que los mostrados en la figura 4. Tras establecer las opciones correctas, utilizando el botón Next aparece una última ventana con información y un botón Finish que se pulsa para crear el proyecto. La figura 5 muestra el proyecto recién creado, sólo se muestra el nombre del proyecto y el tipo de FPGA "xc3s100e-cp132".

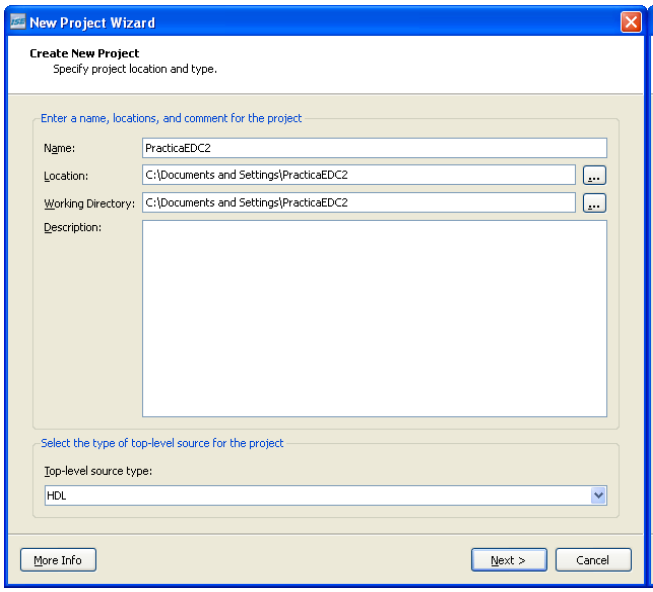


Figura 3. Ventana de creación del proyecto.

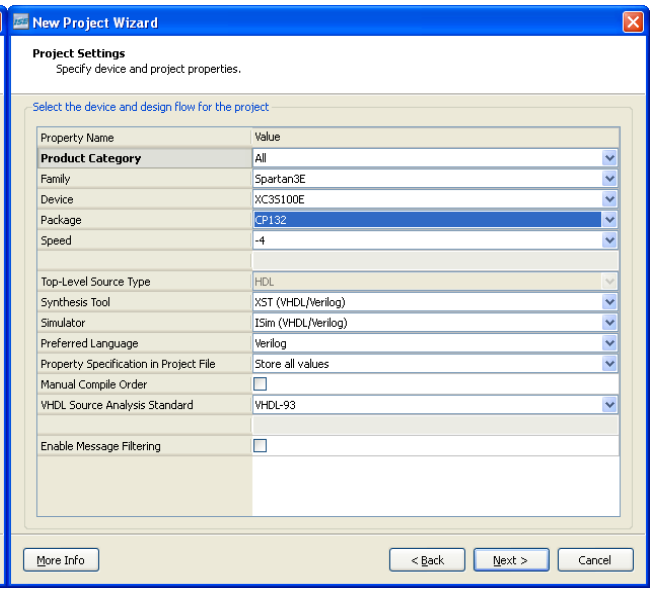


Figura 4. Ventana de propiedades del proyecto.

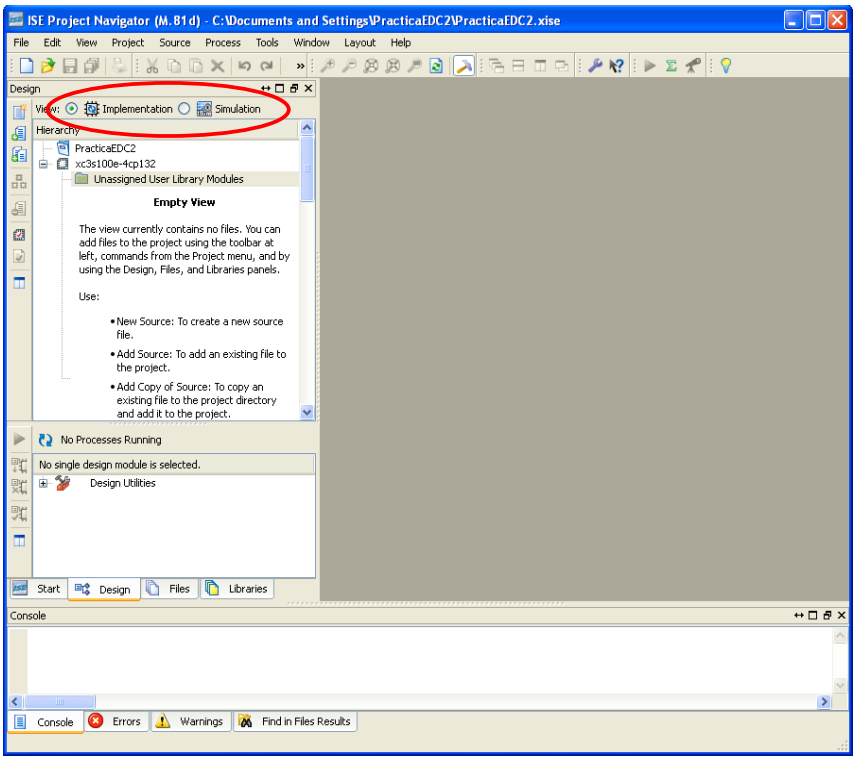


Figura 5. Vista general de un proyecto en el entorno Xilinx.

Nótese que encima del nombre del proyecto aparecen dos iconos *Implementation* y *Simulation*, (ver figura 5) cada uno con distintas vistas del mismo proyecto. Debemos procurar estar siempre en modo de **simulación** para evitar problemas con el entorno ISE. También se recomienda utilizar la entrada de menú Layout → Restore Default Layout en caso de no ver correctamente las ventanas o los controles de *ISE Project Navigator* (o del entorno de Simulación *Isim* que usará más adelante*).*

4.2. Añadir ficheros al proyecto

El primer paso tras la creación del proyecto es añadir los ficheros Verilog (diseños y testbenchs) al proyecto. Para añadir ficheros al proyecto se puede puede utilizar la opción de menú Project o, pulsar el botón derecho del ratón en la zona en blanco de la vista del proyecto, eligiendo entre Add Source o bien Add copy of source teniendo en cuenta que estas dos opciones son algo diferentes:

- Add Copy of Source crea un nuevo fichero dentro del proyecto que inicialmente será una copia del fichero fuente seleccionado. Así las modificaciones serán propias a este proyecto y el fichero fuente original no se modificará. La copia residirá dentro de la carpeta del proyecto.

- Add Source no crea un nuevo fichero dentro del proyecto, utiliza el propio fichero fuente seleccionado sin crear ninguna copia. Así las modificaciones afectarán al fichero fuente en su ubicación original, es decir, el fichero residirá en la misma carpeta dónde esté almacenado previamente.

Otra posible opción sería New source creándose un nuevo fichero fuente vacío donde habría que escribir el código.

La mejor opción es Add Source seleccionando uno o más ficheros a añadir al proyecto. Hay que confirmar los ficheros que son para implementación y cuales son exclusivamente para simulación (cómo los ficheros de *testbench* suministrados). En la figura 6 se muestran los ficheros a añadir y la asociación realizada en cada uno de ellos para que la simulación opere correctamente (elija *All* y *Simulation* según se indica).

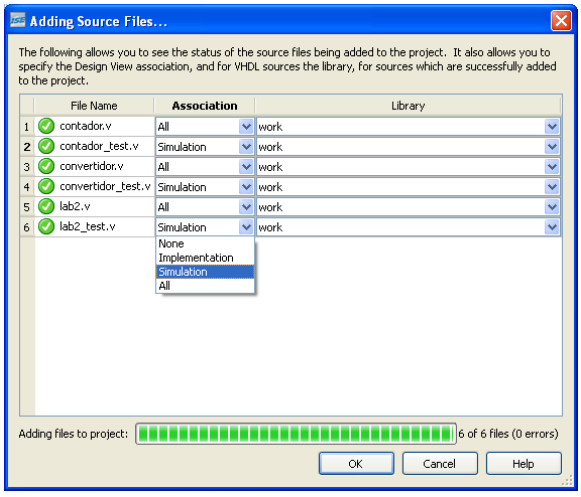


Figura 6. Ventana de inclusión de ficheros al proyecto.

Una vez añadidos los ficheros, éstos son mostrados en la vista del proyecto de forma jerárquica y,

componiendo el árbol de proyecto de ficheros en función de que un fichero necesite de otro fichero, esto se puede visualizar en la figura 6. El fichero que incluye a los demás será el primer fichero del árbol de proyecto.

Para editar o ver cualquiera de los ficheros del proyecto, sólo hay que seleccionarlo con el ratón en el árbol de proyecto y pulsar el botón izquierdo del ratón dos veces.

4.3. Simulación y verificación de un diseño

La simulación nos permite verificar el correcto funcionamiento de una unidad/módulo diseñado, para los casos que se plantean en el fichero de testbench. Éstos podrán ser más o menos completos según el caso y si encontramos algún problema, nos permite indagar la causa del mismo antes de pasar a modificar el código. Efectuando correcciones y simulaciones se consigue solucionar todos los problemas que pueda tener el diseño.

Tras añadir un fichero de testbench, éste no se muestra en la vista de *implementación*, únicamente aparece en la vista *simulación*. Esto es debido a que dichos ficheros contienen información para realizar una simulación/verificación del diseño lógico, pero no se usa para realizar una implementación de la unidad (la última etapa del proceso de diseño). También puede comprobar como en la vista de simulación aparecen las unidades en un orden jerárquico distinto al de implementación. Concretamente, los ficheros de testbench aparecen en primer lugar, puesto que estos serán los que van a guiar la simulación, como se muestra en la figura 7.

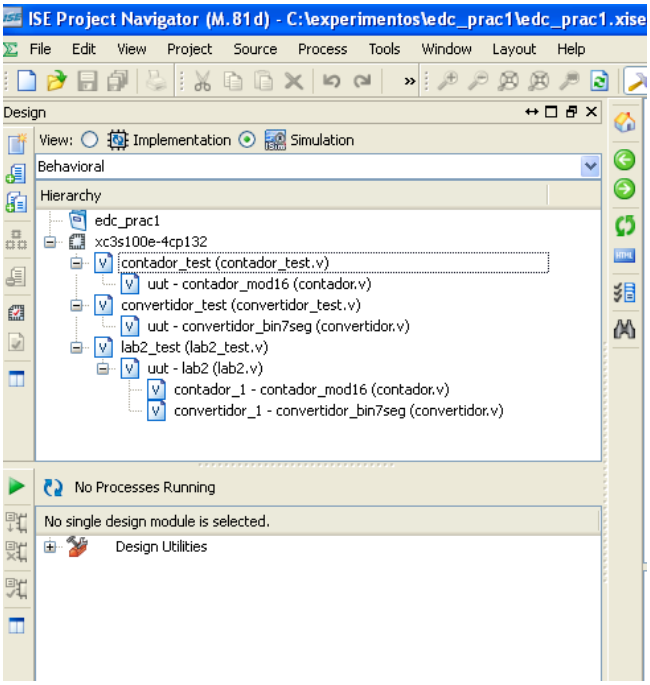


Figura 7. Vista jerárquica de un proyecto.

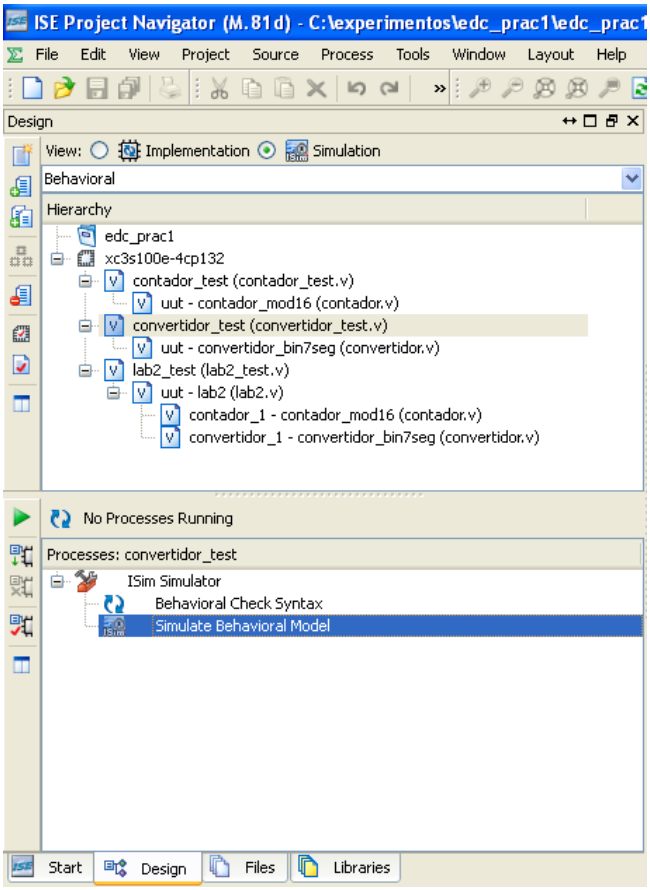


Figura 8. Selección de simulación para el convertidor binario 7 segmentos.

Así, para simular una unidad debemos resaltar el nombre de la unidad/fichero de testbench a simular y abajo en la caja titulada *Processes* desplegar la entrada *ISim Simulator* para ejecutar la orden *Simulate Behavioral Model* pulsando el ratón dos veces, como muestra la figura 8.

Si no hay errores en el diseño, se abrirá una nueva ventana con el simulador *ISim* y ejecutará una simulación durante un corto periodo de tiempo (habitualmente 1 μs), deteniéndose la simulación en ese punto, salvo que el testbench detenga la simulación con antelación (con la orden *$finish*).

Si no hay errores en el código se abrirá el simulador *ISim* donde, para ver las formas de onda, hay que utilizar la pestaña DEFAULT.WCFG . Es aconsejable utilizar en este momento el icono  (ZOOM TO FULL VIEW) para tener una vista completa de toda la simulación.

En esta vista, mostrada en la figura 9, se dispone de una ventana con las formas de onda a la derecha en fondo negro y varias señales representadas con sus valores simulados, que deben ser las entradas y salidas de nuestra unidad (al menos aquellas que aparecen en el testbench). Si hacemos pulsamos el ratón sobre el gráfico de formas de ondas, se sitúa un cursor amarillo indicando datos exactos en ese instante de tiempo (nótese como cambian los valores de las señales al situarse en distintos instantes). Para las señales de varios bits podemos cambiar la codificación pulsando el botón derecho del ratón sobre el nombre de la señal y, accediendo en el menú flotante a la opción *RADIX* (binario, hexadecimal, decimal, etc.).

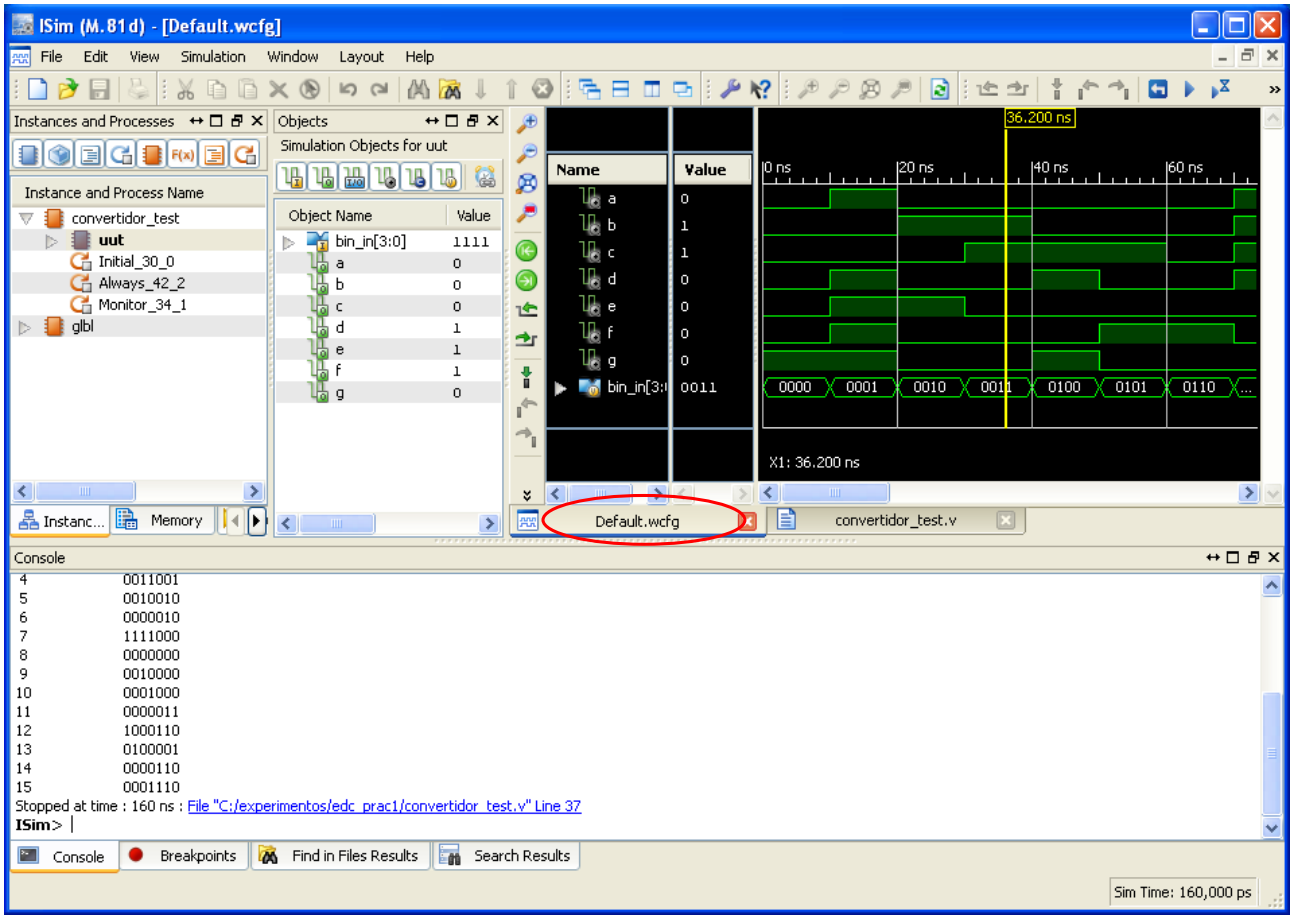


Figura 9. Simulación del convertidor binario 7 segmentos con ISim.

Otros controles importantes de *ISim* se encuentra en dos bloques situados a la izquierda con los que se puede navegar por todas las unidades que componen el diseño. Utilizando estos dos bloques se pueden buscar señales y componentes internos de cualquier módulo para poder mostrar sus formas de ondas, pero, para ello habría que reiniciar la simulación tras añadirlas a la vista de simulación.

Por último, en la barra de iconos superior hay varios iconos que permiten hacer ampliaciones y reducciones de escala en la imagen. Además de estos iconos, varios iconos verdes que hay a continuación sirven para navegar por la forma de onda y, los iconos azules, controlan la simulación utilizándose para continuar o detener el proceso.

Los iconos verdes se deben usar para buscar o centrarse en una parte concreta de las formas de onda, por ejemplo, *Previous Transition* y *Next Transition* nos permiten ir al anterior/siguiente flanco de una señal seleccionada previamente en la ventana de formas de onda.

Los iconos azules controlan el flujo de ejecución de la simulación permitiendo: borrar la simulación actual volviendo al instante cero (*Restart*), continuar la simulación indefinidamente (*Run All*), continuar un tiempo de simulación determinado deteniéndose automáticamente (*Run*), ejecutar la simulación línea a línea de Verilog (*Step*) y, por último, detener una simulación en ejecución (*Break*).