



**DEPARTAMENTO DE TECNOLOGÍA ELECTRÓNICA**  
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

**Enunciados de Laboratorios  
Estructura de Computadores**

**1º de Grado en Ingeniería Informática -  
Ingeniería del Software**

*Paulino Ruiz de Clavijo Vázquez <[paulino@dte.us.es](mailto:paulino@dte.us.es)>*

# Índice de contenido

<b>Lab1. Uso del analizador lógico para testear un dispositivo de lectura de ROMs de Atari 2600.</b>	<b>4</b>
Introducción.....	4
Objetivos.....	6
Estudio teórico.....	6
Estudio experimental.....	7
1.Montaje y conexión con el instrumental.....	7
2.Configuración del analizador LA-2132 para chequear el circuito.....	8
3.Comprobación del estudio teórico.....	10
<b>Lab2. Introducción a Verilog y Xilinx.....</b>	<b>12</b>
Introducción y objetivos.....	12
Estudio teórico.....	12
1.Diseño del convertidor binario a 7 segmentos.....	13
2.Diseño del contador de 4 bits.....	14
Estudio experimental.....	15
Tutorial de Xilinx ISE.....	16
1.Creación de un proyecto en Xilinx ISE.....	16
2.Añadir ficheros al proyecto.....	18
3.Simulación y verificación de un diseño.....	19
<b>Lab3. Multiplicador Secuencial 4x4.....</b>	<b>24</b>
Introducción y objetivos.....	24
Estudio teórico.....	25
Descripción del multiplicador secuencial.....	26
1.Algoritmo de multiplicación.....	28
2.Sistema completo en la placa de desarrollo.....	30
Estudio Experimental.....	32
1.Verificación del multiplicador.....	33
2.Implementación del multiplicador en FPGA.....	35
Producto de $1011 \times 0001$ .....	38
Producto de $0010 \times 1101$ .....	39
<b>Lab4. El Computador Simple 2010 (CS2010).....</b>	<b>40</b>
Introducción y objetivos.....	40
Estudio teórico.....	42
Estudio experimental.....	43
<b>Lab5. Introducción a AVR-STUDIO.....</b>	<b>48</b>
Introducción y objetivos.....	48
Estudio teórico.....	49
Estudio experimental.....	50
1.Introducción a AVR-STUDIO.....	50
1.1.Ejecución en el simulador del programa.....	52
1.2.Programación del microcontrolador.....	54
2.Realización de diversos programas de control E/S.....	59
2.1.Programa para controlar los conmutadores.....	59
2.2.Programa contador de pulsaciones.....	60
2.3.Opcional: Mejoras en el programa contador de pulsaciones.....	63

# Uso del analizador lógico para testear un dispositivo de lectura de ROMs de Atari 2600

## *Estructura de Computadores*

IMPORTANTE: La presentación del estudio teórico escrito es obligatoria para la realización de la prácticas. Se presentará un estudio por persona y se entregará al profesor durante la sesión de prácticas. El estudio ha de dar respuesta a cada una de las cuestiones y ha de ser detallado, completo, claro y bien presentado. El profesor podrá realizar preguntas o pedir aclaraciones sobre el estudio teórico realizado. Además cada alumno debe llevar impreso el enunciado.

## **Introducción**

El analizador lógico es un instrumento de laboratorio que facilita enormemente el testado de circuitos y sistemas digitales que poseen un alto número de entradas y salidas como, por ejemplo, las memorias de acceso directo. Al igual que el osciloscopio, el analizador lógico mide señales de tensión, pero entre ambos existen importantes diferencias:

- El osciloscopio puede medir simultáneamente un número reducido de señales (usualmente solo 2), mientras que analizador puede leer multitud de ellas (del orden de decenas).
- El osciloscopio mide niveles continuos de tensión, mientras que el analizador lógico solo establece si las señales están por encima o por debajo de un determinado umbral. De esta forma el analizador puede indicar el valor lógico de una señal (0 o 1), pero no puede concretar su nivel exacto de tensión.
- El osciloscopio monitoriza y representa continuamente las señales. El analizador, por el contrario, solo examina las señales en ciertos instantes de tiempo para registrar su valor lógico en una memoria interna (la acción de leer y registrar el valor lógico de las señales se denomina

captura). El analizador solo puede informarnos del valor lógico que tienen las señales en los instantes de captura.

Los analizadores pueden realizar las capturas espaciadas en el tiempo en un periodo seleccionable, para lo que disponen de un reloj interno. No obstante, también pueden configurarse para que realicen las capturas cuando ocurra un flanco en una entrada especial (la entrada de reloj externo). Además pueden configurarse para que empiecen a registrar los datos al detectar un determinado patrón en los canales de datos (a este patrón se le denomina palabra de disparo). Todo esto convierte al analizador en una herramienta ideal para analizar el funcionamiento de buses digitales.

En nuestro laboratorio disponemos de analizadores lógicos sobre PC, esto es, una tarjeta de adquisición y un programa sobre PC que funcionan como tal analizador, concretamente el modelo LA-2132. En la presente práctica se usará el dicho analizador para estudiar el proceso de lectura de una ROM, en concreto la ROM de un cartucho de la videoconsola Atari 2600. Dichas ROMs tienen una capacidad de 4KBytes, esto es,  $2^{12} \times 8$  bits. En el laboratorio se realizará una lectura de todas sus 4K posiciones de memoria. Esto se consigue generando todas las posibles direcciones de memoria, desde 0 hasta 4095 (\$000 a \$FFF en hexadecimal). Para ello se usará un contador de 12 bits construido conectando en cascada tres contadores comerciales 74191 tal como se muestra en la Error: No se encuentra la fuente de referencia. Dichos contadores son disparados por el flanco de subida (véase el anexo de la última página).

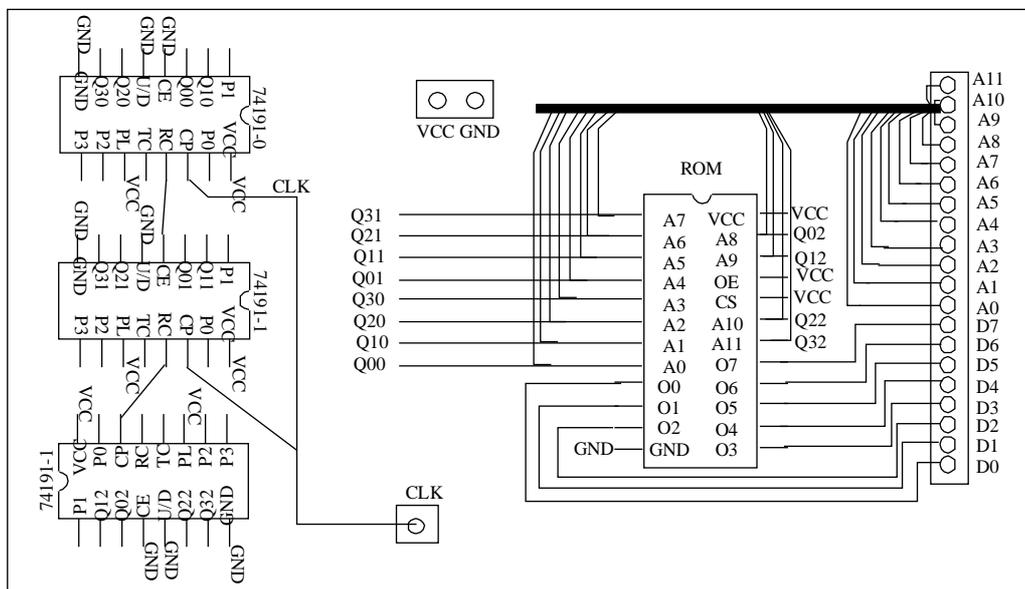


Figura 1: Conexión de la placa

Usaremos el analizador lógico para observar las señales digitales del circuito. La conexión del analizador a la placa del circuito se hace a través de unas sondas o cables. El analizador lógico opera en dos fases: primero, adquiere el valor lógico de las señales y los almacena en su memoria interna; después, representa estos valores en un monitor mediante una herramienta que facilita los recursos de visualización y medida de tiempos. Además, el analizador permite salvar los datos capturados en un fichero.

## Objetivos

- Introducir el manejo del analizador lógico.
- Realizar el testado a nivel lógico de un sistema digital sencillo, en concreto una placa diseñada para leer ROMs de juegos de Atari 2600.
- Realizar un volcado de la ROM y comprobar experimentalmente la corrección del procedimiento mediante el uso de un emulador de Atari 2600.
- Medir, de forma aproximada, tiempos de propagación y frecuencias máximas de operación del circuito.

## Estudio teórico

1. Estudie y comprenda el funcionamiento del circuito de la Error: No se encuentra la fuente de referencia (el componente 74191 se describe en el anexo de la última página).
2. Suponga que los contenidos de las tres posiciones de ROM más bajas (\$000, \$001 y \$002) son respectivamente \$37, \$A9 y \$7E, y que el contenido de la posición de memoria más alta es \$00. Considere \$FFF como estado inicial del contador. Suponiendo que tanto la ROM como los contadores tienen un retraso nulo, complete el siguiente cronograma:
3. En el cronograma anterior, suponga que analizador lógico captura las señales a una frecuencia cuatro veces superior a la del reloj de la placa y que la primera se realiza en el instante marcado por la flecha. Marque sobre el cronograma cuando se realizarían el resto de las capturas e indique los valores de las líneas de datos y dirección en cada una de ellas.
4. Vuelva a completar el cronograma siguiente suponiendo ahora que el conjunto de contadores tiene un retraso  $R_c > 0$ , la ROM un retraso  $R_r > 0$  y asumiendo que la suma de ambos retrasos es menor a la mitad del periodo de reloj de la placa (es decir, debe suponer  $R_c + R_r < T_p/2$ , donde  $T_p$  es el periodo de reloj de la placa). No olvide marcar los retrasos sobre el cronograma.
5. En el cronograma anterior, indique los valores de las líneas de dirección y datos que capturaría el analizador lógico si realiza las capturas en los flancos de bajada del reloj de la placa.
6. Vuelva a completar el cronograma siguiente suponiendo ahora que ambos retrasos son menores que la mitad del periodo, pero que su suma no lo es ( $R_c < T_p/2$ ,  $R_r < T_p/2$ ,  $R_c + R_r > T_p/2$ ).
7. En el cronograma anterior, indique los valores de las líneas de dirección y datos que capturaría el analizador lógico si realiza las capturas en los flancos de bajada del reloj de la placa.
8. En el cronograma anterior, indique los valores de las líneas de dirección y datos que capturaría el analizador lógico si realiza las capturas en los flancos de subida del reloj de la placa.

## Estudio experimental

La realización del estudio experimental requerirá los siguientes componentes: Analizador lógico, generador de ondas, osciloscopio, fuente de continua, placa de circuito impreso (el lector de ROMs de Atari 2600) y de cartucho de Atari 2600. El osciloscopio se usará únicamente para chequear la corrección de las señales de reloj y la alimentación.

### 1. Montaje y conexión con el instrumental.

La disposición de los distintos componentes y conectores de la placa con los contadores se muestra en la Error: No se encuentra la fuente de referencia. En ella puede observarse:

- Un conector donde ha de conectarse la placa con la ROM de Atari 2600. La ROM estará siempre activa, de manera que en su salidas (señales de D0 hasta D7) aparece siempre el contenido de la palabra direccionada.
- Tres contadores de 4 bits (C.I. 74191) que están conectados en cascada formando un contador módulo 4096. Sus 12 bit de salida constituyen la entrada de dirección de la ROM.
- Un conjunto de puntos de conexión para la alimentación, la señal de reloj, las de fin de ciclo (FC2, FC1 y FC0) y las líneas de dirección y datos de la ROM. Con ellos se pueden conectar con facilidad las sondas del analizador lógico a las señales de interés.

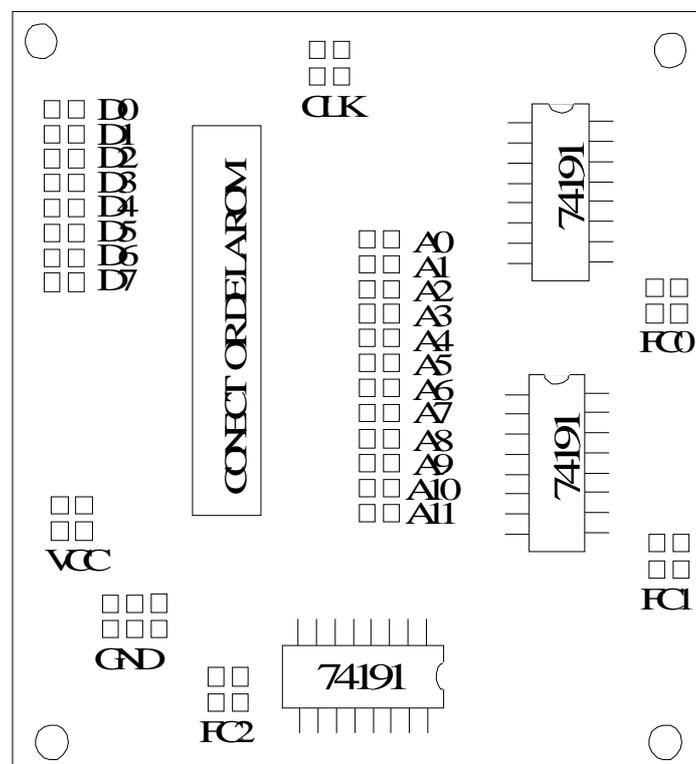


Figura 2: Distribución de los componentes y conectores

**Antes** de alimentar la placa con la fuente de continua debe hacerse lo siguiente:

- 1) En primer lugar se extraerá la placa con la ROM del cartucho y se insertará en el zócalo del circuito de lectura. ES MUY IMPORTANTE NO INSERTARLA AL REVÉS: La cara con el chip de

la ROM debe estar orientada hacia los pines de datos D0-D7.

- 2) A continuación se conectarán los terminales del circuito bajo test a las sondas del analizador que, por otro lado, deberá conectarse al ordenador a través del puerto USB. El orden de las conexiones de las terminales del analizador a la placa se muestra en la tabla siguiente:
- 3) Use el generador de ondas para generar la señal de reloj. Debe ser una señal cuadrada de 25 Khz de frecuencia que varíe entre 0 y 5 voltios. Además deberá tener un duty cycle del 50%, es decir, durante cada ciclo deberá encontrarse la mitad del tiempo en nivel alto y la otra mitad en nivel bajo. Tendrá que emplear el osciloscopio para chequear que la señal es correcta.
- 4) Use el generador de continua para generar la señal de alimentación de 5 voltios. **No use el generador de tensión fija (el de la derecha).**
- 5) Baje el limitador de intensidad (sin situarlo completamente a la izquierda).
- 6) Use el osciloscopio para chequear que la alimentación es de 5 voltios.

**Solo después** de haber realizado los pasos anteriores deberá conectar la alimentación y la señal de reloj a la placa. Tras esto la ROM estará poniendo cíclicamente en su salida el contenido de todas sus posiciones de memoria.

## **2. Configuración del analizador LA-2132 para chequear el circuito**

Hay que tener en cuenta que antes de ejecutar el software del LA-2132 debemos conectar el analizador al puerto USB del PC. Este software, disponible en Windows XP, se ejecuta pulsando sobre el icono LA-2132 disponible en el escritorio.

A grandes rasgos, la ventana de aplicación presenta un marco superior que contiene los distintos menús (File, View, Timing,...), una fila de comandos u opciones que se pueden ejecutar (Go, Stop, ...) y una ventana dividida en dos partes:

- La parte superior contiene los controles de los cursores/marcadores del sistema, los controles de zoom y una barra para desplazarnos a lo largo de las capturas.
- La parte inferior está dedicada a presentar los datos capturados en los distintos canales de adquisición.

Antes de adquirir y visualizar las señales de la placa debemos establecer las condiciones de trabajo. Para ello pulsamos con el botón derecho sobre la ventana inferior y, en la ventana recién aparecida, seleccionamos las siguientes opciones:

- En la entrada 'Trig Word' (en la que aparecen los canales en el orden 31, 30, 29, ..., 0), escribimos '0' en las posiciones correspondientes a los canales del 8 al 19, es decir, los canales correspondientes a las líneas de dirección. El analizador no comenzará a capturar los datos hasta que el valor de los contadores coincida con esta entrada (palabra de disparo), es decir, hasta que la salida de los contadores no sea \$000.

- Para establecer que la adquisición de datos vaya sincronizada con el flanco de subida del reloj externo (el de nuestra placa), en la entrada 'Source' seleccionamos 'External rising'.
- Para que el tamaño de la captura sea de 8K muestras seleccionamos '8K' en la entrada 'Memory'.
- Para que realice una sola captura seleccionamos 'Single' en la entrada 'Acquire'.

Tras cerrar la ventana emergente establecemos la agrupación de canales en el menú 'View'->'Group edit'. Para cada grupo de canales hay que establecer su nombre (entrada 'Name'), base en la que se visualizará el valor de sus canales (entrada 'Base'), número de canales que lo forman (entrada 'Number') y cuáles son dichos canales. Para cambiar de grupo se pulsa sobre las flechas de la parte superior de la ventana. Se crearán tres grupos. El primero, llamado 'contenido', estará constituido por los 8 bits correspondientes a la salida de la ROM y se mostrará en hexadecimal. El segundo, llamado 'dirección' estará constituido por los 12 bits correspondientes a las entradas de dirección de la ROM y se mostrará también en hexadecimal. El tercero, llamado 'Fin Cuenta' contendrá las señales de fin de cuenta de cada uno de los tres contadores y se mostrará en binario.

Una vez modificados todos estos menús se puede pasar a la adquisición de los datos. Para ello pulse la tecla 'Go' y comenzará la adquisición al ritmo marcado por el reloj. Tras finalizar la adquisición aparecerán actualizados los datos capturados en la pantalla. Con la configuración señalada se capturaran las señales en cada flanco de subida del reloj externo. Para chequear que ha ido todo bien, compruebe lo siguiente:

- El valor de las líneas de dirección, al ser la salida de un contador controlado por el reloj de la placa, debe incrementarse en cada captura. Sus formas de onda deben ser señales cuadradas, cada una de ellas de periodo doble a la anterior.
- Las tres señales de fin de cuenta se pondrán simultáneamente a uno únicamente cuando el contador esté en el último estado de cuenta.

Además hay que testear que en las líneas de datos se pone, de forma secuencial, el contenido de cada posición de la ROM. Dado que se trata de una ROM de Atari 2600, esto puede comprobarse guardando los datos capturados y pasándolos a un emulador tal y como se describe a continuación.

1. Para guardar las capturas del analizador en un fichero debemos hacer lo siguiente:
  - 1.1. En el menú del software del analizador seleccionamos 'File'->'Save data'->'Save data as'.
  - 1.2. Elegimos un directorio (por ejemplo 'c:\roms') y elegimos un nombre de archivo. Conviene que el nombre tenga menos de 8 caracteres para acceder a él directamente desde la línea de comandos.
2. Ahora debemos extraer de dicho archivo los valores correspondientes a las líneas de datos y grabarlos en un fichero binario legible por el emulador de la forma siguiente:

- 2.1. abrimos una ventana de línea de comandos accediendo al menú de inicio de Windows ('Inicio'->'Todos los programas'->'Accesorios' ->'Símbolo del sistema').
- 2.2. ejecutamos esto:

```
cd directorio_con_el_fichero_salvado
dso2bin archivo_original.dso fichero_salida.bin
```

Es muy importante dar al fichero de salida un nombre que termine en la extensión '.bin' para que pueda leerlo el emulador Stella

3. Finalmente comprobaremos el volcado con el emulador Stella. Para ello:
  - 3.1. Se pulsa sobre su icono en el escritorio.
  - 3.2. Dentro del emulador se hace doble click sobre el fichero previamente creado.
  - 3.3. Se comprueba que en efecto el programa de Atari funciona. Las teclas básicas son F2 (start, comienzo de partida), espacio (disparo), y los cursores.

### 3. Comprobación del estudio teórico

- 1) Apunte algunas de las direcciones de memoria que aparecen en la pantalla (en particular la \$000), así como el contenido correspondiente.
- 2) Cambie el modo de adquisición de datos en el analizador lógico para que realice las capturas a una frecuencia cuatro veces superior a la del reloj de la placa ('Source'->'Internal'->'Rate'->'100Ks'). Verifique experimentalmente su respuesta al punto 3 del estudio teórico (mueva el cursor 'Trigger' para ver cuantas veces se repiten las muestras).
- 3) Cambie la frecuencia del reloj interno del analizador a 100Mhz y vuelva a capturar. Use los cursores para medir aproximadamente el retraso del contador de 12 bits (Rc) y de la ROM (Rr).
- 4) Cambie de nuevo el modo de adquisición de datos para que la captura se realice en los flancos de bajada del reloj de la placa. Use los datos anotados en el apartado 1 para verificar experimentalmente su respuesta al punto 5 del estudio teórico.
- 5) Para verificar experimentalmente las respuestas a los puntos 6 y 7 haga lo siguiente:
  - 5.1)Cambie el modo de adquisición para que el analizador realice cíclicamente capturas sin detenerse (modo de disparo 'Auto') y pulse 'Go' para iniciar las capturas.
  - 5.2)Aumente poco a poco la frecuencia del reloj de la placa y deténgase justo cuando los datos anotados en el apartado 1 no se capturen correctamente.
  - 5.3)Calcule el periodo del reloj de la placa y compruebe si se cumple la desigualdad del apartado 6 del estudio teórico.
- 6) Cambie de nuevo el modo de adquisición de datos para que la captura se realice en los flancos de subida del reloj de la placa. Use los datos anotados en el apartado 1 para verificar experimentalmente su respuesta al punto 8 del estudio teórico.

### **Anexo: descripción del contador comercial 74191**

El componente 74191 es un contador de 4 bits. A parte de sus salidas de estado (Q) y sus entradas de

datos para carga en paralelo (P) dispone de las siguientes señales:

- CE (Count Enable, entrada): es la señal de habilitación de cuenta (activa en bajo).
- U/D (Up/Down, entrada): indica si la cuenta será ascendente (valor 0) o descendente (valor 1).
- CP (Clock Pulse, entrada): es la señal de cuenta y es activa en el flanco de subida.
- TC (Terminal Count, salida): indica el fin del ciclo de cuenta (se activa en el estado 1111).
- RC (Ripple Clock, salida): En el último estado de cuenta es igual a CP. En los demás vale 1.
- PL (Parallel Load, entrada): Cuando vale cero el contador realiza una carga en paralelo.

# **Introducción a Verilog y Xilinx**

## *Enunciados de Prácticas de Laboratorio Estructura de Computadores*

### **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<sup>1</sup>.
- Conocer las herramientas de simulación usadas para la verificación de sistemas digitales.
- 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.

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

Para ambos circuitos se han preparado unas plantillas de código que, además de estar incluidas en este documento, se proporcionan a través 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 módulo 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.

### 1. Diseño del convertidor binario a 7 segmentos

Para poder visualizar números se utilizan displays 7 segmentos. Estos displays tienen 7 entradas, una por cada segmento que se puede iluminar, de forma que, al iluminar algunos de ellos se puede componer visualmente un número. En la figura Error: No se encuentra la fuente de referencia se muestran los números desde el 0 al 9.

<sup>1</sup> Xilinx Inc.: Compañía de desarrollo de FPGAs ([www.xilinx.com](http://www.xilinx.com))

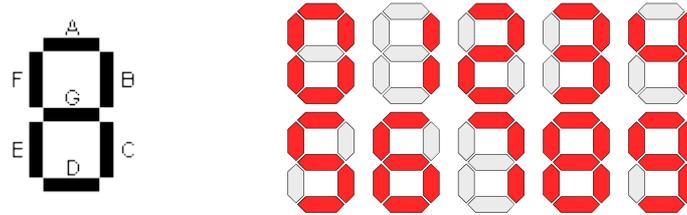


Figura 3. Ejemplo de representación de dígitos en un display de 7 segmentos.

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 Error: No se encuentra la fuente de referencia. 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.

<b>bin<sub>3</sub></b>	<b>bin<sub>2</sub></b>	<b>bin<sub>1</sub></b>	<b>bin<sub>0</sub></b>	<b>A</b>	<b>B</b>	<b>C</b>	<b>D</b>	<b>E</b>	<b>F</b>	<b>G</b>
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 aquí 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 binario a 7 segmentos.

## 2. Diseño del contador de 4 bits

Dicho contador, disparado por el flanco de subida del reloj, tendrá dos señales síncronas: una señal de puesta a cero (RESET) y otra de incremento (UP). 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 Error: No se encuentra la fuente de referenciab y, para diseñar el código Verilog se propone utilizar como plantilla el fragmento de código Error: No se encuentra la fuente de referencia (fichero *contador.v*).

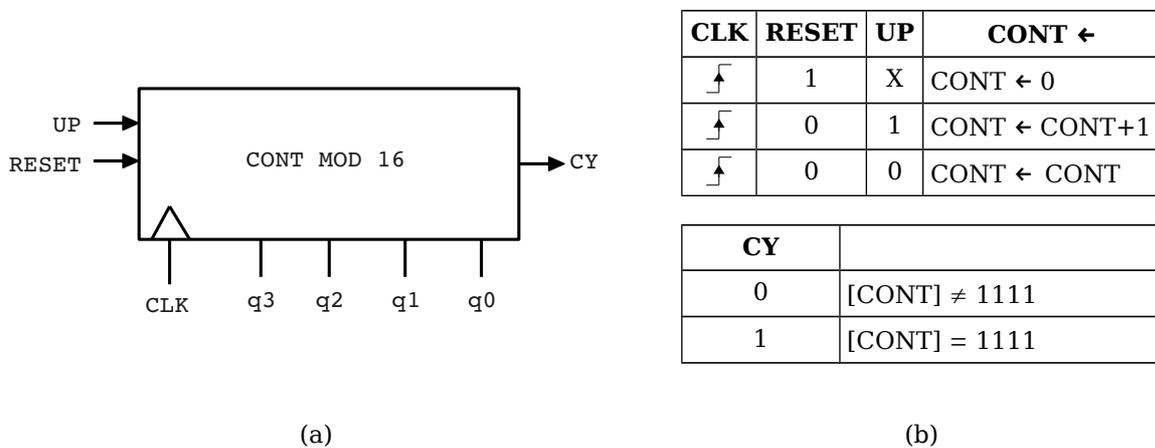


Figura 4. Descripción del contador módulo 16: (a) Descripción estructural, (b) Descripción funcional.

```
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.

## 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 la sección 4, 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 Error: No se encuentra la fuente de referencia) 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.

## 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.

## 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 Error: No se encuentra la fuente de referencia 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 Error: No se encuentra la fuente de referencia, 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
- **Speed:** -4
- **Preferred Language:** Verilog

El resto de opciones deberían estar por defecto a los mismos valores que los mostrados en la figura Error: No se encuentra la fuente de referencia. 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 Error: No se encuentra la fuente de referencia muestra el proyecto recién creado (sólo se muestra el nombre del proyecto y el tipo de FPGA "xc3s100e-cp132").

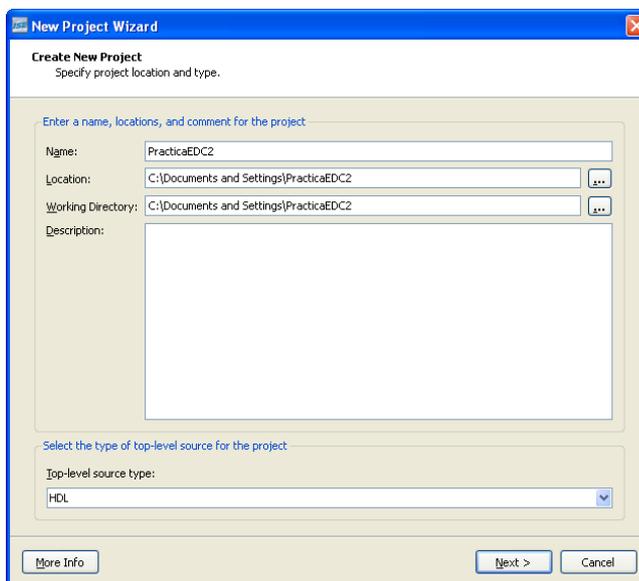


Figura 5. Ventana de creación del proyecto.

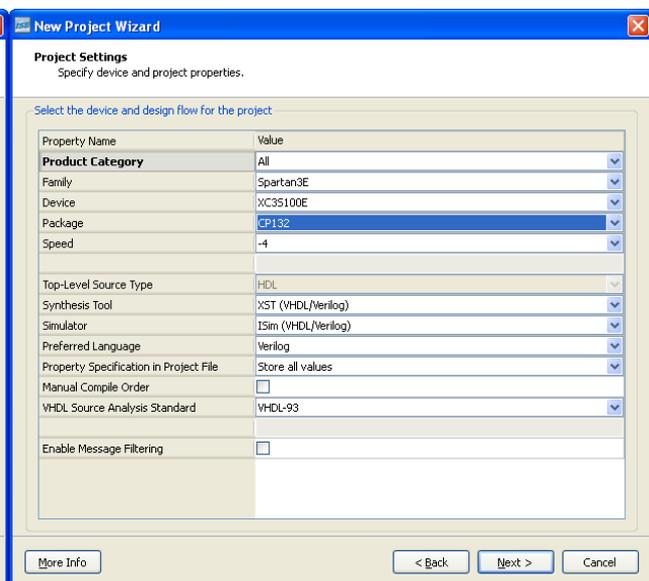


Figura 6. Ventana de propiedades del proyecto.

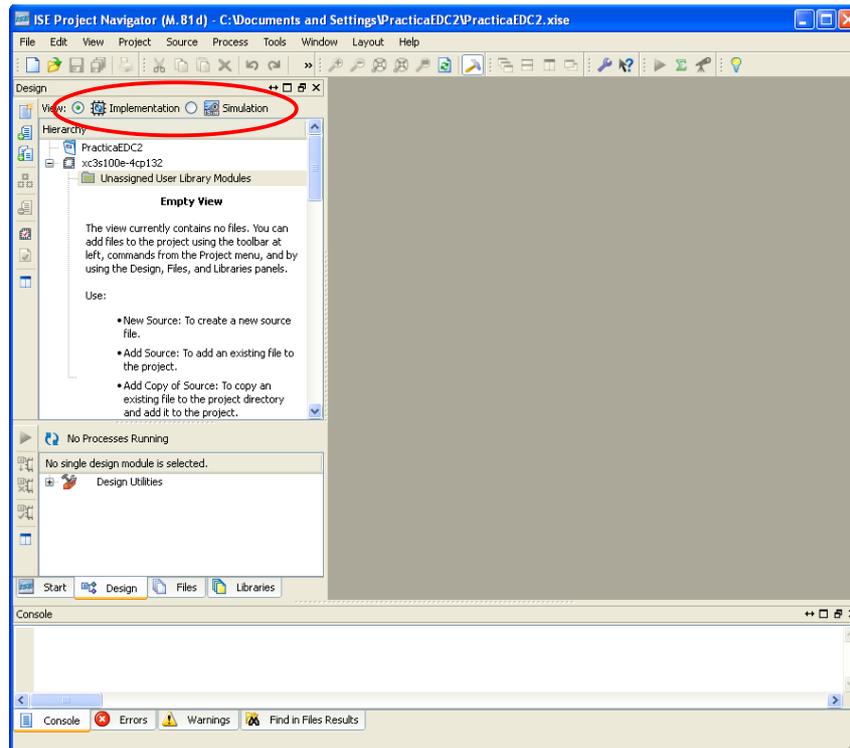


Figura 7. Vista general de un proyecto en el entorno Xilinx.

Nótese que encima del nombre del proyecto aparecen dos iconos *Implementation* y *Simulation*, (ver figura Error: No se encuentra la fuente de referencia) 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).

## 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 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 cuáles son exclusivamente para simulación (como los ficheros de *testbench* suministrados). En la figura Error: No se encuentra la fuente de referencia 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).

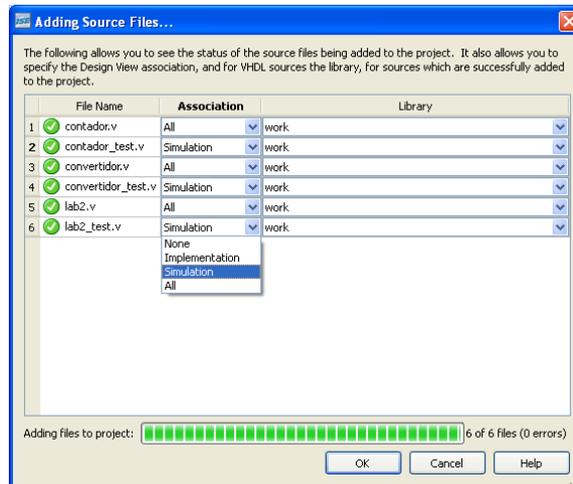


Figura 8. Ventana de inclusión de ficheros al proyecto.

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 Error: No se encuentra la fuente de referencia. 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.

### 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 Error: No se encuentra la fuente de referencia.

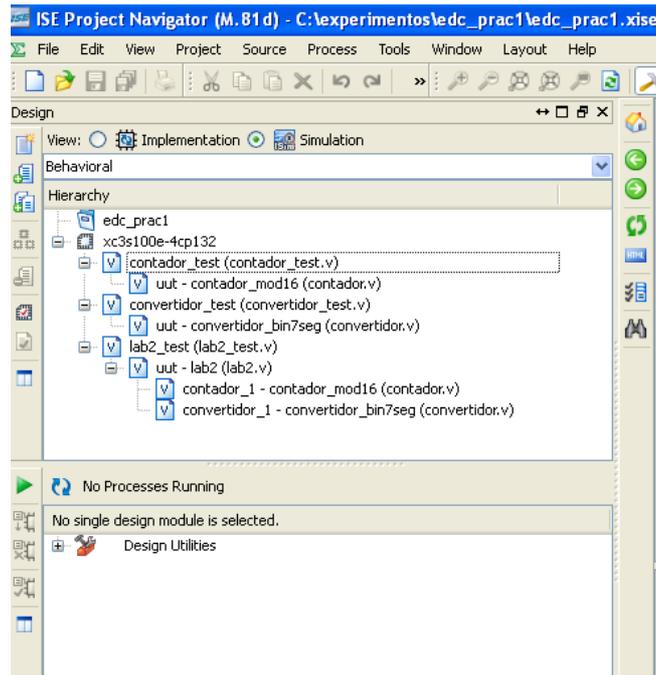


Figura 9. Vista jerárquica de un proyecto.

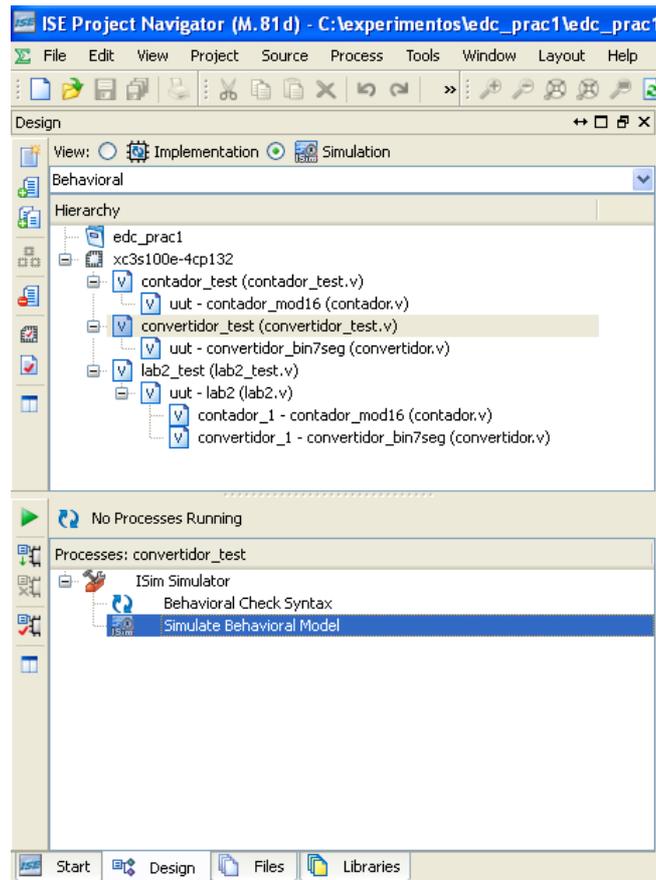


Figura 10. Selección de simulación para el convertidor binario 7 segmentos.

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 Error: No se encuentra la fuente

de referencia.

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  $1\mu\text{s}$ ), 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 Error: No se encuentra la fuente de referencia, 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 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.).

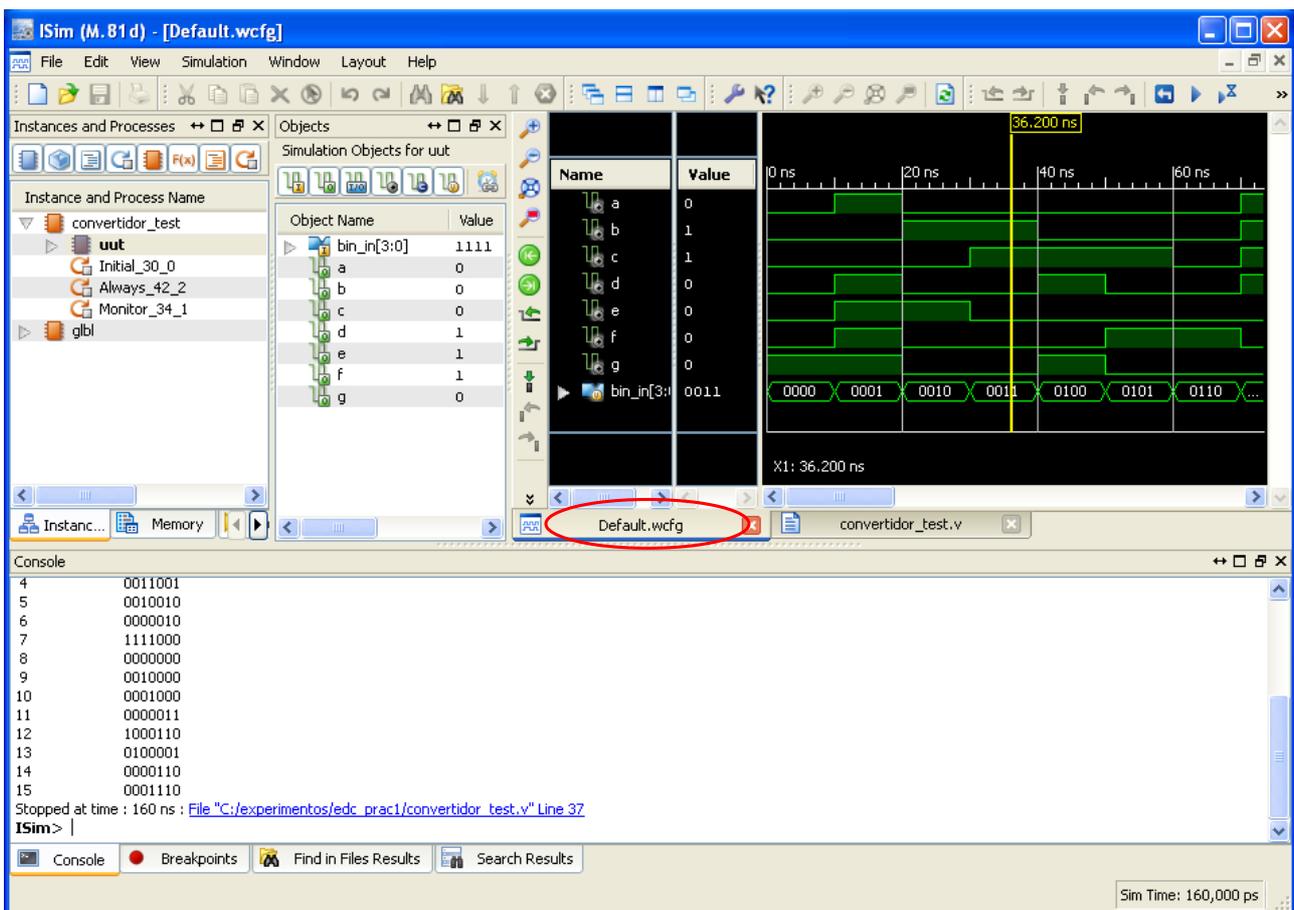


Figura 11. Simulación del convertidor binario 7 segmentos con *ISim*.

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*).

## Anexo para profesores

 Solución propuesta para el convertidor binario a 7-segmentos

```

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

    always @*
    begin
        case (bin_in)
            4'h0: {a,b,c,d,e,f,g} = 7'b0000001; //--0
            4'h1: {a,b,c,d,e,f,g} = 7'b1001111; //--1
            4'h2: {a,b,c,d,e,f,g} = 7'b0100100; //--2
            4'h3: {a,b,c,d,e,f,g} = 7'b0110000; //--3
            4'h4: {a,b,c,d,e,f,g} = 7'b0011001; //--4
            4'h5: {a,b,c,d,e,f,g} = 7'b0010010; //--5
            4'h6: {a,b,c,d,e,f,g} = 7'b0000010; //--6
            4'h7: {a,b,c,d,e,f,g} = 7'b1111000; //--7
            4'h8: {a,b,c,d,e,f,g} = 7'b0000000; //--8
            4'h9: {a,b,c,d,e,f,g} = 7'b0010000; //--9
            4'hA: {a,b,c,d,e,f,g} = 7'b0001000; //--A
            4'hB: {a,b,c,d,e,f,g} = 7'b0000011; //--b
            4'hC: {a,b,c,d,e,f,g} = 7'b1000110; //--C
            4'hD: {a,b,c,d,e,f,g} = 7'b0100001; //--d
            4'hE: {a,b,c,d,e,f,g} = 7'b0000110; //--E
            default: {a,b,c,d,e,f,g} = 7'b0001110; //--F
        endcase
    end
endmodule

```

 Solución para el contador

```

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

    always @(posedge clk)
    begin
        if (reset == 1)
            q <= 0;
        else if (up == 1)
            q <= q + 1;
        end

        assign cy = (q == 4'b1111);
    endmodule // contador_mod16

```

 Solución de la interconexión de los dos componentes

```
module lab2(
    input clk, up, reset,
    output [0:6] seg, // el orden de las señales importa, sobre todo para implementación.
    output cy);

    wire [3:0] del_q_al_d; // bus para conectar la salida Q con la entrada D

// Se dan 2 soluciones, la "corta" lista para usar, y la "larga" va comentada.
// Solución "corta", preservando el orden de los puertos de entrada/salida de cada modulo

    contador_mod16 contador_1(clk, reset, up, del_q_al_d, cy);
    convertidor_bin7seg convertidor_1(del_q_al_d, seg[0], seg[1], seg[2], seg[3], seg[4], seg[5],
seg[6]);

// Solución "larga", nombrando cada puerto del modulo y la señal que se asigna.
// Si se quiere probar, descomantarla completa y comentar la otra solución.
//
// contador_mod16 contador_1(
//     .clk(clk),
//     .reset(reset),
//     .up(up),
//     .q(del_q_al_d),
//     .cy(cy));
//
// convertidor_bin7seg convertidor_1(
//     .bin_in(del_q_al_d),
//     .a(seg[0]),
//     .b(seg[1]),
//     .c(seg[2]),
//     .d(seg[3]),
//     .e(seg[4]),
//     .f(seg[5]),
//     .g(seg[6]));
//

endmodule // lab2
```



**DEPARTAMENTO DE TECNOLOGÍA ELECTRÓNICA**  
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

## **Multiplicador Secuencial 4x4**

*Enunciados de Prácticas de Laboratorio*  
*Estructura de Computadores*

### **Introducción y objetivos**

El propósito general de esta sesión de laboratorio es operar con sistemas digitales modernos. Este propósito se concreta en los siguientes objetivos:

- Conocer cómo opera un sistema digital con la estructura “Unidad de Datos y Unidad de Control”. En particular, el sistema propuesto realiza la multiplicación de dos números binarios A y B de cuatro bits mediante el algoritmo basado en sumas y desplazamientos a la derecha.
- Desarrollar las habilidades de diseño e implementación. Para ello, tendrá que realizar la unidad de control del multiplicador secuencial, contando previamente con la especificación completa de la unidad de datos y con una descripción del microprograma de control. Además, se implementará el multiplicador para lo que se volcará el diseño sobre una FPGA.
- Comprobar la implementación del multiplicador realizada sobre FPGA tanto a nivel de microoperación como a nivel de macrooperación.

Para la realización de este laboratorio se facilitan un conjunto de ficheros, algunos de ellos deben ser completados antes de la sesión de laboratorio. La tabla 4 resume el contenido y el objetivo de cada uno de ellos.

<b>Nombre del fichero</b>	<b>Contenido</b>	<b>Descripción</b>
u_control.v	Unidad de control del multiplicador	Debe completarlo el alumno antes de asistir a la sesión de laboratorio
u_datos.v	Unidad de datos del multiplicador	Debe utilizarlo sin modificaciones para

Nombre del fichero	Contenido	Descripción
		conectarlo a la unidad de control.
multiplicador.v	Módulo para el multiplicador	Debe completar la descripción estructural durante la sesión de laboratorio.
multiplicador_tb.v	Testbench para multiplicador	Debe utilizarlo para realizar la simulación
sistema_completo.v	Sistema completo para implementar en FPGA	Se utilizará sin modificaciones durante la implementación
basys2.ucf	Fichero con la descripción de pads de la placa Basys2	Se utilizará sin modificaciones durante la implementación

Tabla 3. Ficheros necesarios durante la sesión de laboratorio.

## Estudio teórico

Estudie adecuadamente la descripción del multiplicador secuencial que aparece en la sección y realice dos tareas:

3. A partir de la carta ASM propuesta para el multiplicador secuencial (figura Error: No se encuentra la fuente de referencia), realice la unidad de control en Verilog mediante una descripción canónica. Para ello utilice el código del fichero *u\_control.v* (ver código Error: No se encuentra la fuente de referencia mostrado al final de esta sección). Este apartado es **absolutamente imprescindible** para realizar la práctica.
4. Usando las tablas proporcionadas en las últimas páginas, muestre la secuencia de datos y de señales de control del sistema en cada ciclo de reloj CLK, para las dos siguientes multiplicaciones :

4.1. A = 1011 B = 0001

4.2. A = 0010 B = 1101

```

module u_control(
  input xs,clk,reset,busc0,cycont,
  output reg clsincr,fin,
  output reg wa,wd,wsuml,wsumh,shrsum,upcont
);

// Defina aqui la lista de estados con la sentencia parameter
...

// Defina aqui dos variables tipo reg llamadas:
// estado_actual y siguiente_estado
// Dimensione ambas variables con el tamaño correcto en función
// del número de estados que definió previamente
...

// Proceso de cambio de estado, utiliza la variables
// definidas previamente
always @(posedge clk,posedge reset)
begin
  if(reset)
    estado_actual <= S0;

```

```

else
    estado_actual <= siguiente_estado;
end
// Proceso combinacional de calculo de proximo estado
// debe obtenerlos a partir de la carta ASM
always @(*)
begin
    clsincr=0; // Por defecto se establecen todas las señales a cero
    wa=0;
    wd=0;
    wsuml=0;
    wsumh=0;
    upcont=0;
    shrsum=0;
    fin=0;
    case(estado_actual)
        // Rellene todos los estados y active las señales de control siguiendo la carta ASM
        S0:
            if(xs)
                begin
                    clsincr=1;
                    siguiente_estado = S1;
                end
            else
                siguiente_estado = S0;

        // Aqui deben ir el resto de estados
        ...
        ///

        default: // No elimine esta sentencia para evitar problemas
            siguiente_estado=S0; // Cuidado aqui, hay que cubrir todos los casos
            // de lo contrario aparece un latch

    endcase
end
endmodule

```

Código 4. Fichero u\_control.v, plantilla de código para la unidad de control.

## Descripción del multiplicador secuencial

El multiplicador secuencial es un sistema digital con dos bloques bien diferenciados: Unidad de datos y Unidad de control (ver figura Error: No se encuentra la fuente de referencia).

El funcionamiento global de este multiplicador es el siguiente. La primera vez que se dispone a multiplicar, el usuario realizará un RESET del sistema activando la señal destinada a ello (RESET). Posteriormente, activará la señal XS y la unidad de control irá proporcionando, ciclo a ciclo, las señales de control que necesita cada elemento de la unidad de datos para que se realice la multiplicación de los números A y B que se hayan fijado como entrada al sistema.

El circuito de la unidad de datos se muestra en la figura Error: No se encuentra la fuente de referencia. Éste surge del algoritmo de multiplicación que se describe en la sección 1. Puede ver que está compuesto por tres registros (A, SUMH y SUML), un sumador paralelo de 4 bits y un contador módulo 4 (CONT). Cada uno de ellos tiene su tabla de descripción RT representada en la misma figura Error: No se encuentra la fuente de referencia. Las salidas de estos registros son incondicionales.

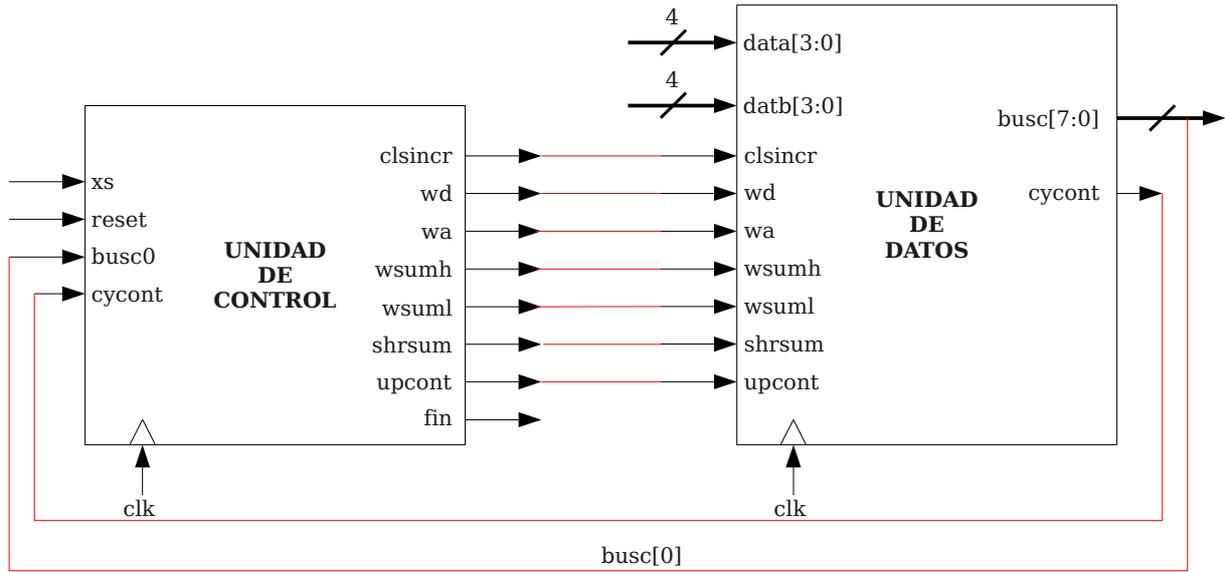


Figura 12. Representación a nivel de bloques del multiplicador secuencial.

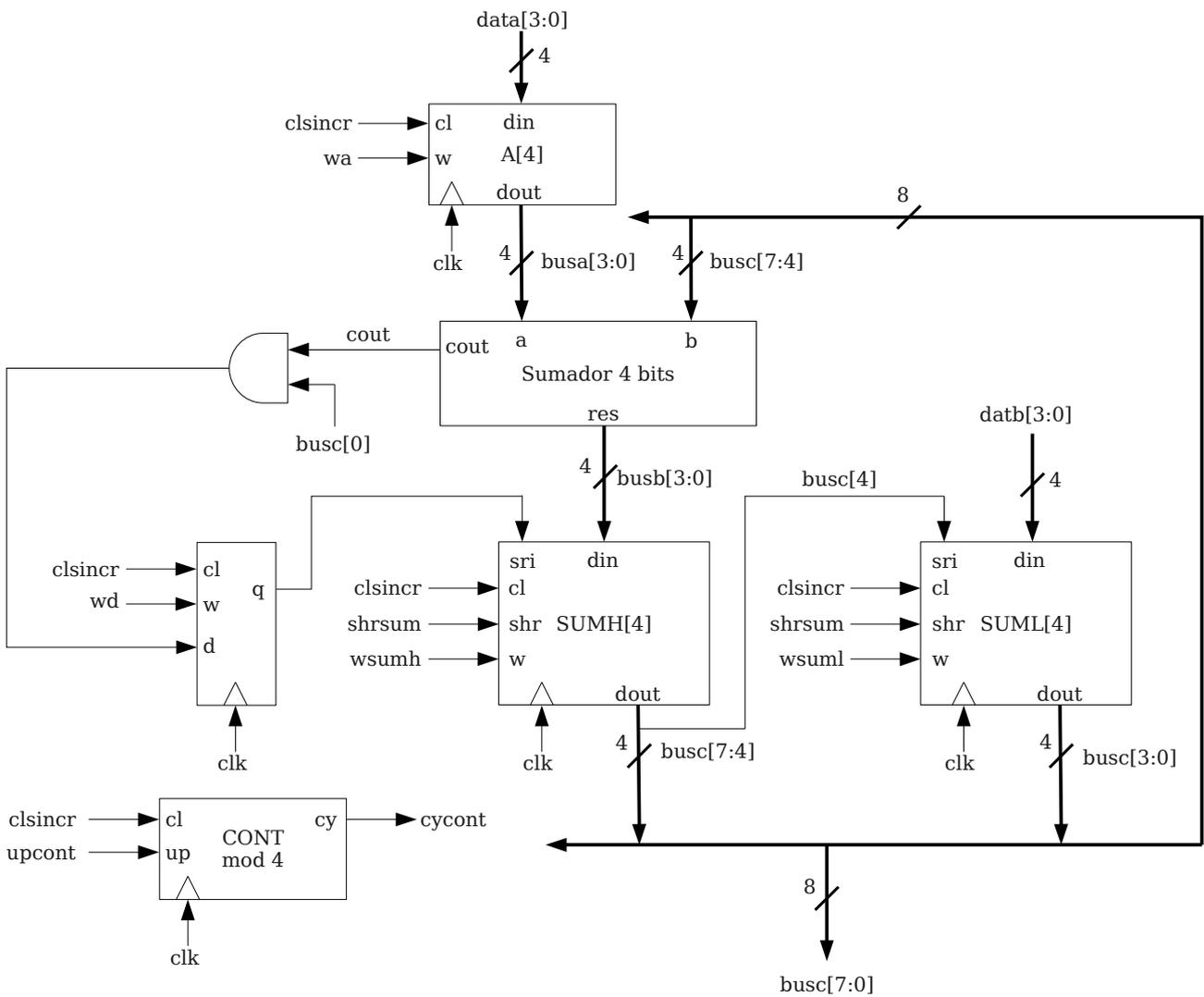


Figura 13. Representación estructural de la ruta de datos.

clk	cl	w	Operación
$\uparrow$	1	-	$A \leftarrow 0$
$\uparrow$	0	1	$A \leftarrow \text{DATA}[3:0]$
$\uparrow$	0	0	$A \leftarrow A$

Registro A

clk	cl	up	Operación
$\uparrow$	1	-	$\text{CONT} \leftarrow 0$
$\uparrow$	0	1	$\text{CONT} \leftarrow \text{CONT} + 1$
$\uparrow$	0	0	$\text{CONT} \leftarrow \text{CONT}$

Contador

clk	cl	w	shr	Operación
$\uparrow$	1	-	-	$R \leftarrow 0$
$\uparrow$	0	1	-	$R \leftarrow \text{Din}[3:0]$
$\uparrow$	0	0	1	$R \leftarrow \text{SHR}(R, \text{sri})$
$\uparrow$	0	0	0	$R \leftarrow R$

Registro de desplazamiento

clk	cl	w	Operación
$\uparrow$	1	-	$D \leftarrow 0$
$\uparrow$	0	1	$D \leftarrow \text{Din}$
$\uparrow$	0	0	$D \leftarrow D$

Biestable

Figura 14. Descripción RT de los componentes de la ruta de datos.

### 1. Algoritmo de multiplicación

Para realizar la multiplicación de dos números binarios de 4 bits, A y B, se ha seguido el algoritmo de sumas y desplazamientos a la derecha. A continuación se describe el procedimiento.

Cuando se realiza la multiplicación de dos números binarios mediante el procedimiento tradicional de multiplicaciones se puede observar como la multiplicación consiste en realizar sumas desplazadas a la izquierda. Al ser los datos binarios, los sumandos sólo pueden ser el primer operando de la multiplicación o cero.

El algoritmo de multiplicación por sumas y desplazamientos propuesto consiste en realizar cada una de estas sumas parcialmente. La idea básica es sumar a un resultado parcial un operando cada vez que un bit del otro operando sea 1. En cada paso se realiza una suma parcial y se guarda en un registro separado el bit menos significativo del resultado obtenido. Esto se debe a que en la siguiente suma parcial este bit ya no interviene. Realizando sucesivamente la operación suma y desplazamiento a la derecha de un bit se obtiene el resultado de la multiplicación.

En la figura Error: No se encuentra la fuente de referencia se muestra un ejemplo de multiplicación siguiendo dicho algoritmo paso a paso. Observe como en cada paso se analiza un bit del operando B. Siempre que encuentre un 1 realiza una suma y un desplazamiento de un bit (2 pasos). En caso de ser este bit 0, se realiza únicamente el desplazamiento de un bit (un único paso).

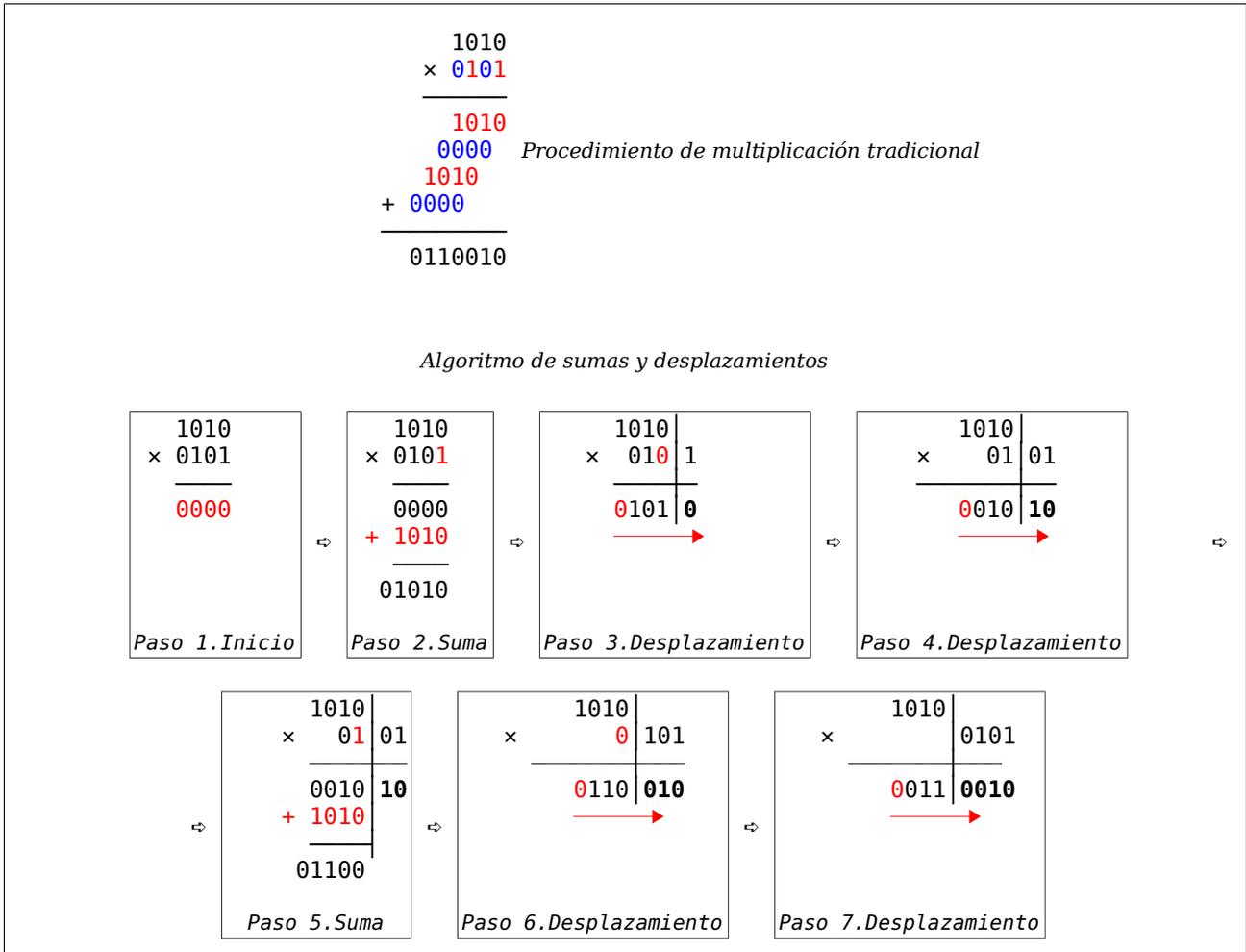


Figura 15. Algoritmo de multiplicación por suma y desplazamiento.

Para la ruta de datos propuesta el algoritmo se realizará de la siguiente forma:

1. Se realiza una carga en paralelo de los datos A y B en los registros A y SUML respectivamente.
2. Se ha dispuesto de un contador módulo 4 para contar los 4 desplazamientos necesarios. Será el bit de acarreo de este contador el que indique cuándo finaliza la multiplicación. Debe iniciarse este contador a cero.
3. Se procede a analizar cuál es el bit menos significativo de B (registro SUML0). Si es 0 no se hace nada, en cambio, si es 1 se realiza una suma del dato de A con el dato presente en un tercer registro (SUMH) destinado a almacenar los productos parciales de la multiplicación. Esta suma se guarda en el mismo registro SUMH.
4. Tras el paso anterior se realiza una operación de desplazamiento a la derecha de los registros SUMH y SUML conjuntamente.
  - 4.1. SUMH desplaza su bit LSB a SUML así, en la próxima operación de suma este bit no se suma.
  - 4.2. Este bit desplazado pasa a convertirse en el siguiente bit del resultado y queda guardado en SUML.

- 4.3. Se decreenta el contador módulo 4.
- 5. Se realizarán iterativamente los pasos 3 y 4 cuatro veces, ya que el dato B posee 4 bits. Cuando se active el bit de acarreo del contador módulo 4 finaliza la multiplicación.
- 6. El resultado final de la operación quedará guardado en los registros SUMH y SUML.

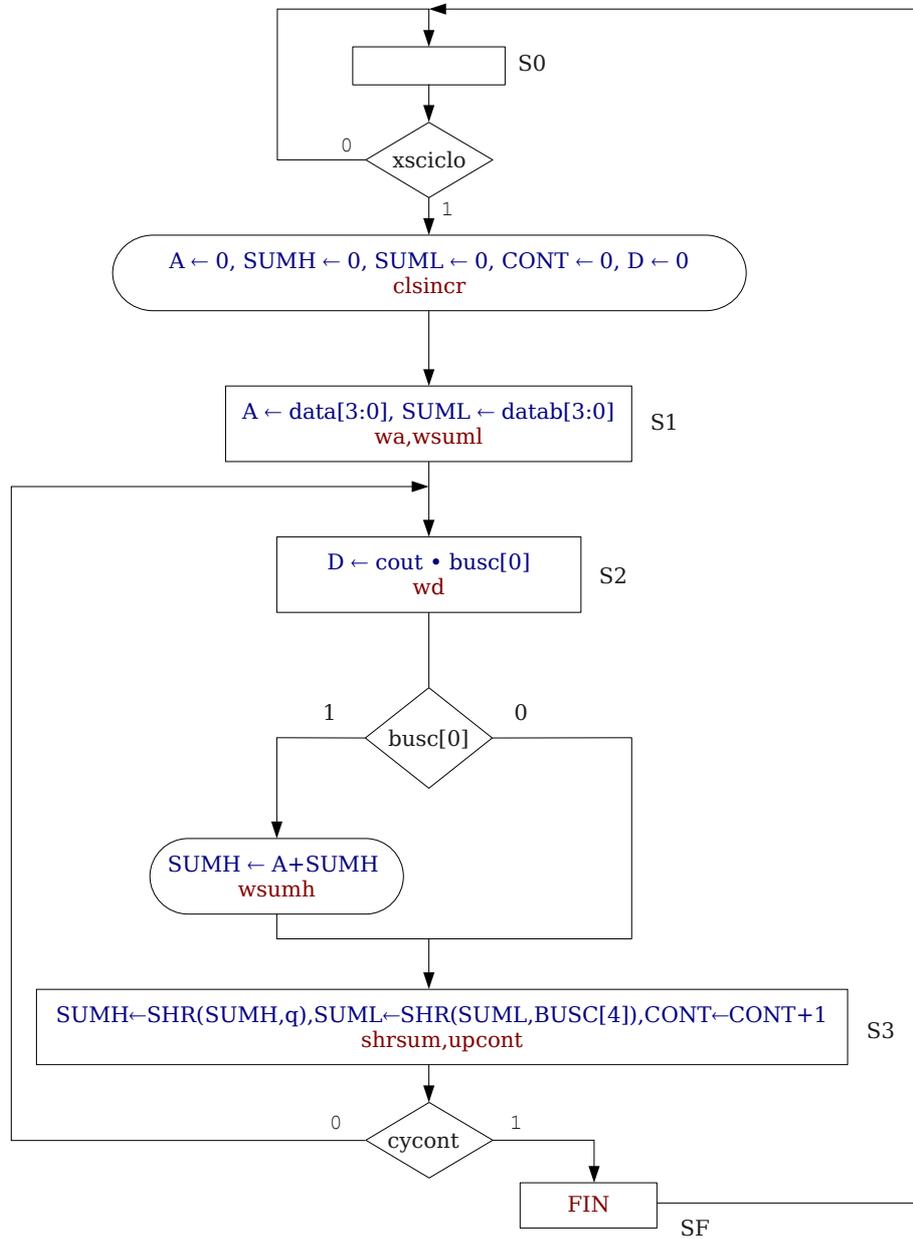


Figura 16. Carta ASM de datos y de control del multiplicador secuencial.

Una carta ASM que da respuesta al algoritmo y Unidad de Datos anteriores es la que se muestra en la figura Error: No se encuentra la fuente de referencia. De ella se obtiene directamente la Unidad de Control, la cual deberá ser obtenida por el alumno usando una descripción canónica en Verilog.

## 2. Sistema completo en la placa de desarrollo

Para probar el diseño digital realizado se utilizará una placa desarrollo del fabricante Digilent<sup>2</sup>

<sup>2</sup> Empresa dedicada al diseño en tecnologías basadas en FPGA y microcontroladores. Web: <http://www.digilentinc.com>

llamada *Basys2* que incluye un FPGA de 500.000 puertas y algunos componentes electrónicos para entrada y salida básica. Para poder interactuar con el sistema utilizando la placa de desarrollo, es necesario añadir al diseño un procedimiento para introducir los datos a multiplicar, poder visualizar los resultados y consultar el estado del sistema. Esta placa de desarrollo dispone de algunos elementos de entrada y salida como son: 8 conmutadores, 4 displays de 7 segmentos, 8 leds y 4 pulsadores entre otros.

Mediante varios módulos descritos en Verilog y ya preparados (archivo *sistema\_completo.v*) se construye un sistema digital como el indicado de manera esquemática en la figura Error: No se encuentra la fuente de referencia. Esta figura muestra el módulo multiplicador construido por el alumno, conectado a un módulo capaz de controlar el display 7 segmentos. Este controlador consta de 4 entradas de 4 bits cada una de ellas ( $d_0$ ,  $d_1$ ,  $d_2$  y  $d_3$ ) correspondiéndose cada una de las entradas con uno de los dígitos mostrados en el display. Además, se ha conectado el reloj, la señal *xs* y la señal *reset* a pulsadores para controlar el funcionamiento del sistema manualmente. También, la señal *fin* y el reloj se interconecta a los *leds* para poder visualizar el efecto de pulsación del reloj e indicar la finalización de la multiplicación. Por último, los datos A y B se pueden introducir en binario mediante conmutadores.

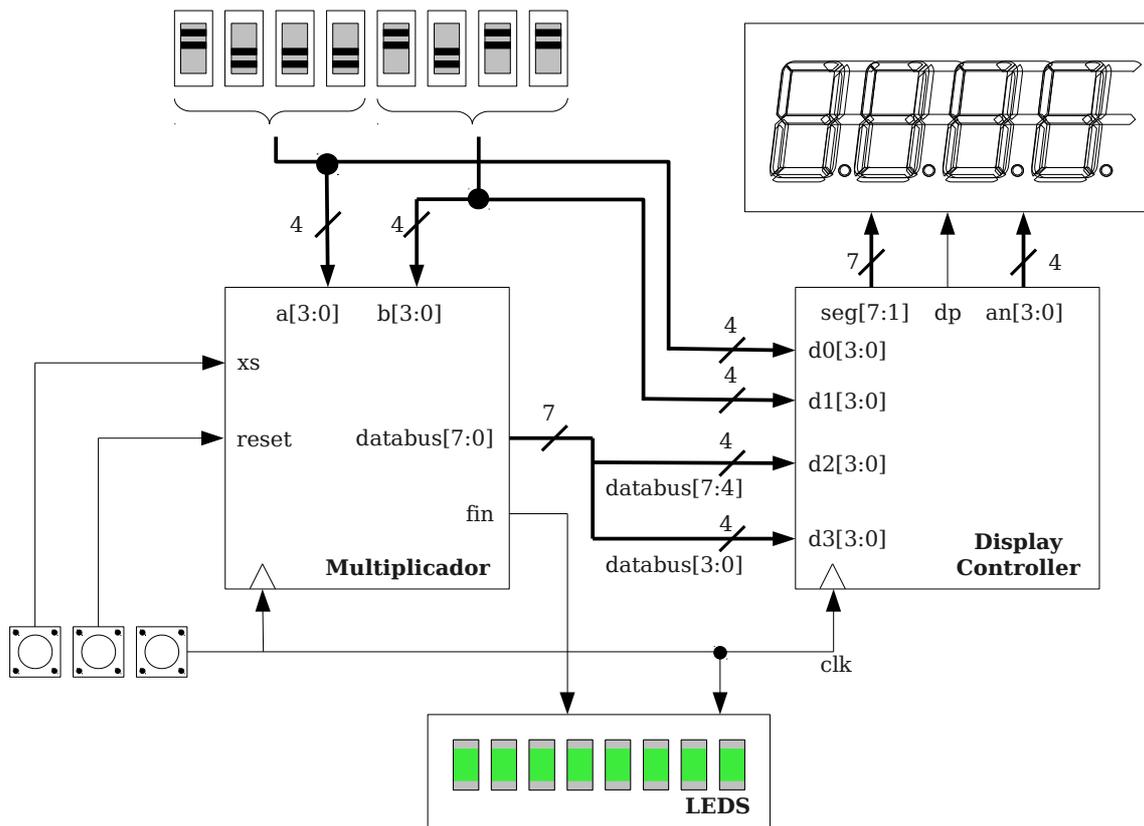


Figura 17. Esquema del sistema digital completo para el multiplicador.

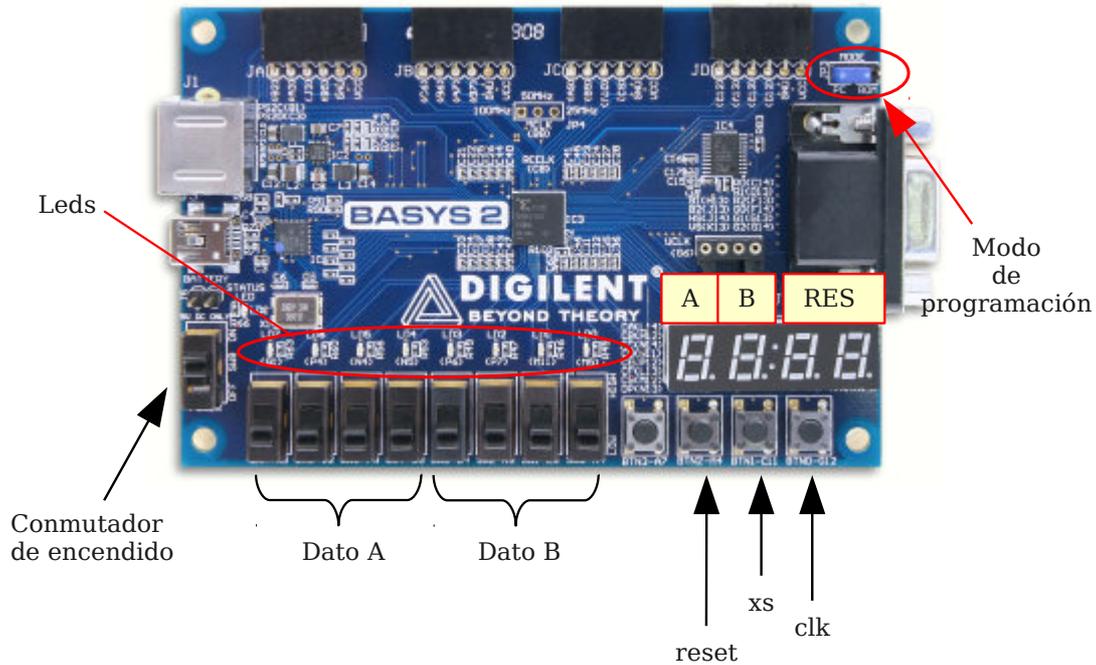


Figura 18. Fotografía de la placa de desarrollo Basys2

La figura Error: No se encuentra la fuente de referencia muestra una fotografía de la placa de desarrollo. Para este diseño digital los 4 primeros conmutadores permiten introducir en binario el dato A. Simultáneamente el primer dígito del display mostrará en hexadecimal el número establecido según se cambien estos conmutadores. Con el dato B sucede lo mismo, sólo que con los últimos cuatro conmutadores y el segundo dígito del display. El resultado (formado por SUMH y SUML) se irá mostrando ciclo a ciclo de reloj en los dos últimos dígitos del display.

Para operar, primero se configuran los valores de A y B con los conmutadores, tras esto, es necesario generar la señal *xs* de comienzo y el reloj utilizando los botones. Los botones señalados en la figura introducen un 1 lógico cuando se pulsan y están conectados a las señales *reset*, *xs*, y *clk*. El procedimiento consiste en pulsar en primer lugar el botón *xs* y tras esto pulsar repetidamente el reloj (*clk*).

Si el sistema está operando correctamente, por cada ciclo de reloj un led diferente se iluminará indicando que se ha detectado la pulsación y, los dos últimos dígitos irán mostrando el resultado parcial. Cuando se active la señal de fin se iluminarán todos los leds indicando que ha terminado la multiplicación. En este momento el resultado de la multiplicación estará en los dos últimos dígitos del display.

## Estudio Experimental

En la sección anterior se ha descrito el multiplicador como Sistema Digital. Éste consta de dos partes: Unidad de Datos (archivo *u\_datos.v*) y Unidad de Control (archivo *u\_control.v*). La primera se proporciona al alumno ya diseñada y la segunda, el alumno debe traerla hecha. En la parte experimental de la práctica hay que realizar las siguientes tareas:

1. Completar la descripción estructural del multiplicador y simular el multiplicador para verificar el

correcto funcionamiento.

2. Realizar la implementación del sistema completo en FPGA y verificar su funcionamiento.

A continuación se detalla cada tarea.

## 1. Verificación del multiplicador

Para verificar el correcto funcionamiento del multiplicador se va a realizar una simulación de 10 multiplicaciones mediante un testbench proporcionado en el fichero *multiplicador\_tb.v*. Para conseguir este objetivo siga los siguientes pasos:

1. Cree un nuevo proyecto vacío en el entorno ISE siguiendo los mismos pasos de la sesión de laboratorio anterior.

2. Establezca el modo de simulación en el proyecto (figura Error: No se encuentra la fuente de referencia). Añada al proyecto los 4 ficheros mediante la opción de menú **Add Source** del menú **Project**. Los ficheros son los indicados a continuación teniendo en cuenta los siguientes aspectos en cada uno de ellos:

2.1. Fichero *u\_datos.v*: Debe añadirlo sin modificarlo y como *Implementation*.

2.2. Fichero *u\_control.v*: Debe haber completado el fichero en la realización de estudio teórico. Añádalo como *Implementation*.

2.3. Fichero *multiplicador.v*: Se completará posteriormente. Debe añadirlo como *Implementation*.

2.4. Fichero *multiplicador\_tb.v*: Debe añadirlo como *Simulation*. Tras añadir este fichero debe aparecer como raíz del árbol de ficheros del proyecto, corresponde al llamado *test*.

3. El siguiente paso es editar el fichero *multiplador.v* abriéndolo desde el árbol de proyecto de ISE. Debe completar los fragmentos indicados en el mismo fichero, concretamente hay que realizar la interconexión de la unidad de datos y de control tal y como se mostraba en la figura Error: No se encuentra la fuente de referencia. Es recomendable visualizar el contenido del fichero *u\_datos.v* y tomar como ejemplo el módulo *u\_datos* donde se realiza el conexionado de la unidad de datos. Debe completar en el fichero *multiplicador.v* siguiendo los pasos indicados a continuación:

3.1. Realizar el conexionado de la unidad de datos con los cables internos ya definidos: *wa*, *upcont*, *wsuml*, *wsumh*, *cycont*, *clsincr* y *shrsum*. Debe realizar la conexión directa (sin usar cables internos) de las entradas *a*, *b*, *databus* y *clk* del módulo *multiplicador* con la unidad de datos. Tenga en cuenta que *databus* se refiere al bus de salida de la unidad de datos, llamada *busc* en la unidad de datos.

3.2. Realizar la instancia de la unidad de control del mismo modo que se ha instanciado la unidad de datos. Debe interconectar todas las entradas y salidas de la unidad de control, tanto a los cables internos como las entradas/salidas del módulo *multiplicador*.

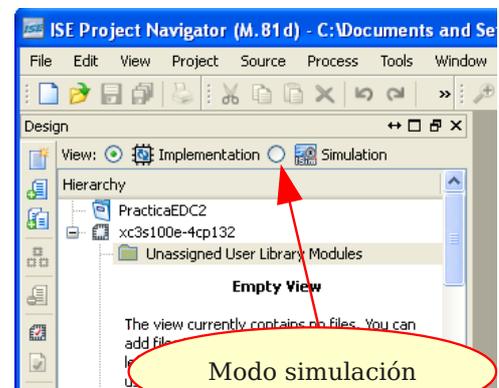


Figura 19. Modo simulación

```

module multiplicador(
wire wa,wd,upcont,wsuml,wsumh,cycont,clsincr,shrsum;

// Instancia de la unidad de datos debe completar las conexiones
u_datos U_DATOS(
    .data(a)
    ...
    // Complete aqui todas las conexiones que faltan
    // Para ello puede abrir el fichero incluido: "u_datos.v"
    // y mirar la definicion del modulo "u_datos" que sirve
    // como ejemplo de interconexión de módulos.
);

/* Aqui debe instanciar la unidad de control del mismo
modo que se ha instanciado la unidad de datos.

Debe interconectar todas las conexiones
con la unidad de datos: tanto las entradas como las salidas. */

    ...
endmodule

```

Código 5. Fichero *multiplicador.v*, plantilla de código interconexión de unidad de datos y de control.

4. Por último se va a realizar la simulación. Seleccione el testbench en el árbol de proyecto, es el módulo superior en el árbol de proyecto (ver figura Error: No se encuentra la fuente de referencia), con el nombre *test* (*multiplicador\_tb.v*).

4.1. Tras esta selección debe utilizar la entrada *Behavioural Check Syntax* para comprobar si hay errores en su código. Debe corregir todos los errores (ver indicación en la figura Error: No se encuentra la fuente de referencia).

4.2. Una vez corregidos los errores, realice la simulación mediante la opción *Simulate Behavioural Model*. Debe comprobar todas las señales que obtuvo en la tabla del estudio teórico, para ello, añada a la simulación las señales internas de los módulos de la ruta de datos tal y como se muestra en el figura Error: No se encuentra la fuente de referencia. Busque las multiplicaciones del estudio teórico y amplíe la forma de onda hasta que se visualicen correctamente las señales.

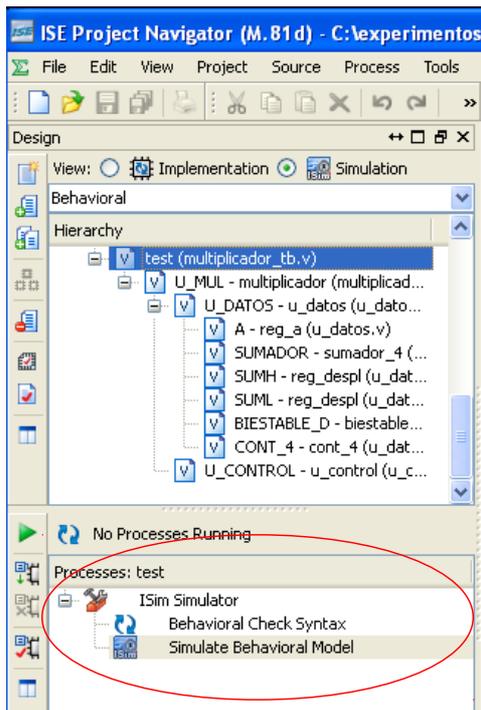


Figura 20. Simulación del tesbench.

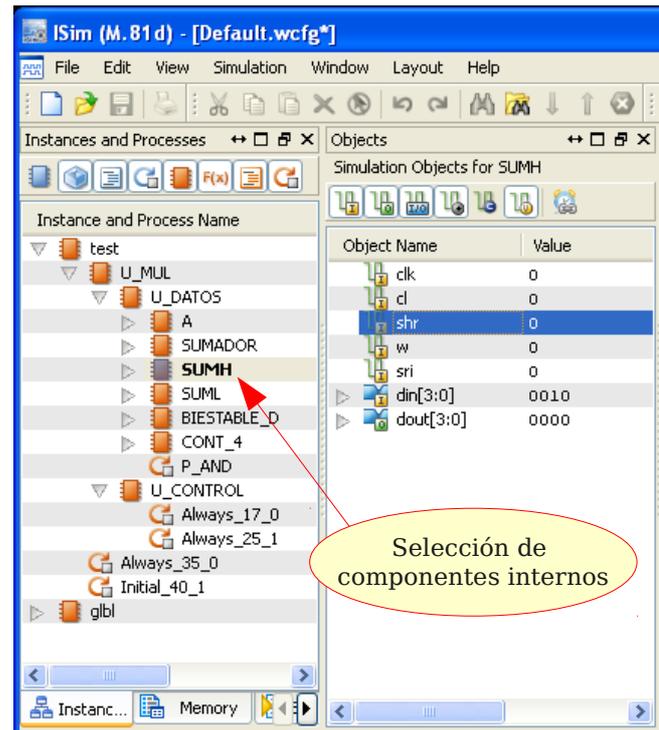


Figura 21. Acceso a módulos internos.

## 2. Implementación del multiplicador en FPGA

Finalmente se implementará el sistema digital realizado en un dispositivo programable tipo FPGA incluida en la placa de desarrollo Basys2.

Tal y como se detalló en la sección 2 el multiplicador necesita conectarse a los conmutadores, botones y el display de la placa de desarrollo para poder interactuar con él. Los módulos que controlan estos componentes se encuentran ya diseñados y testados en el fichero *sistema\_completo.v*. Además del sistema completo, es necesario utilizar otro fichero donde se indican las conexiones entre los pads del chip FPGA y los componentes de la placa. Este fichero se llama *basys2.ucf* y ya contiene la asignación de las conexiones del sistema a los componentes de la placa.

Los pasos a seguir son los siguientes:

1. Cambiar el proyecto ISE del modo simulación al modo implementación tal y como se mostraba en la figura Error: No se encuentra la fuente de referencia
2. Añadir dos ficheros al proyecto:
  - 2.1. Fichero *sistema\_completo.v* como implementación. Este fichero aparecerá ahora como raíz del proyecto.
  - 2.2. Fichero *basys2.ucf*. Debe comprobar que también aparece en el árbol de proyecto.

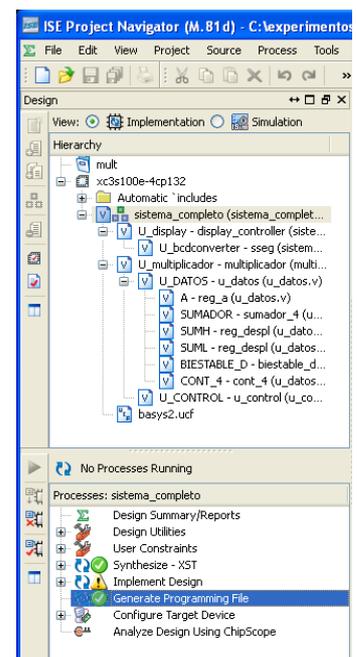


Figura 22. Implementación

3. La implementación se realiza seleccionando el módulo *sistema\_completo* en el árbol de proyecto. Debe seleccionar la opción **Generate Programming File** (figura Error: No se encuentra la fuente de referencia) y seguir los siguientes pasos:

3.1. Pulsando el **botón derecho** del ratón sobre la opción **Generate Programming File** aparecerá un menú flotante como el mostrado en la figura Error: No se encuentra la fuente de referenciaa, seleccionando la opción de menú de **Process Poperties** aparecerá el diálogo mostrado en la figura Error: No se encuentra la fuente de referenciab.

3.2. En el diálogo hay que elegir la categoría **Startup Options** y cambiar el valor del primer campo **FPGA Start-Up Clock** de **CCLK** a **Jtag-Clock**. Tras aceptar los cambios con el botón **OK** se volverá a la ventana principal de ISE.

3.3. Pulsando dos veces botón izquierdo del ratón sobre **Generate Programming File** se ejecuta el proceso completo y se genera el fichero de programación. Tal y como se muestra en la figura Error: No se encuentra la fuente de referencia, si el proceso ha terminado con éxito aparecerá un indicador verde, en caso contrario un aspa rojo indicando la existencia de errores. En caso de existir errores debe corregirlos y repetir el proceso.

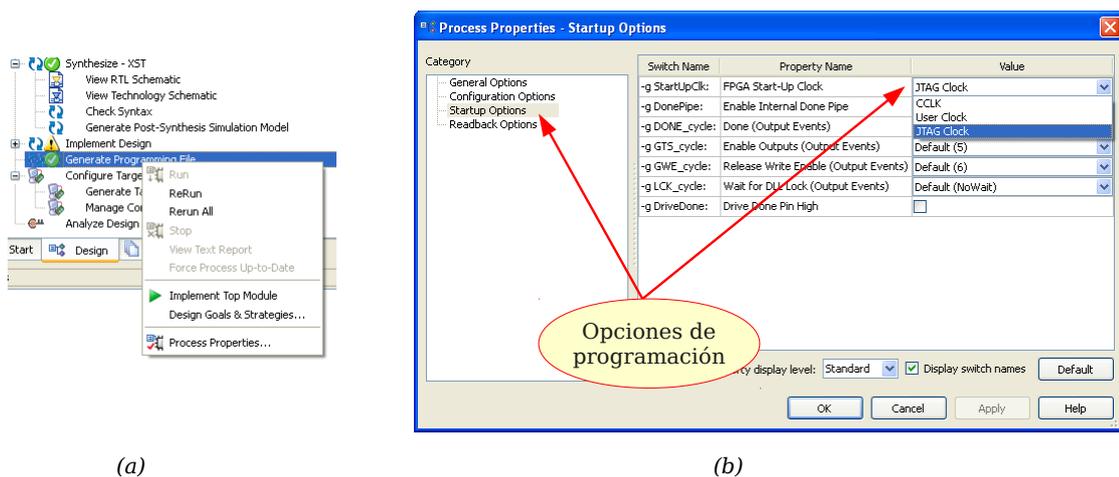


Figura 23. Opciones de generación del fichero de programación:  
(a) menú desplegable, (b) diálogo con opciones

4. El último paso consiste en programar la placa de desarrollo con un fichero que se ha generado tras el proceso del paso anterior. Concretamente, el fichero debería llamarse *sistema\_completo.bit* y hay que transferirlo por la conexión USB a la FPGA. Para ello siga los siguientes pasos:

4.1. Compruebe en la placa el modo de programación establecido, para ello fíjese en la figura Error: No se encuentra la fuente de referencia donde está situado el conmutador *Modo de Programación*. Asegúrese que está en modo *PC*.

4.2. Conecte el puerto USB de la placa de desarrollo y conmute la alimentación de la placa. Por defecto debe cargarse un programa de test de la propia placa que cuenta dígitos BCD en el Display. Además, puede comprobar el correcto funcionamiento de los conmutadores y los botones antes de proceder con la programación.

4.3. Inicie el programa **Adept** desde el menú de inicio (menú **Digilent** → **Adept**, icono ) y aparecerá el programa mostrado en la figura Error: No se encuentra la fuente de referencia. Con este programa se puede transferir el fichero de programación (*bitstream*) a la FPGA. El programa

*Adept* permite programar los componentes de la placa Basys2, estos son, una FPGA y una PROM. Se programará la FPGA, por tanto, se debe utilizar el botón **Browse** indicado en la figura y seleccionar el fichero *.bit*. Hay buscar la carpeta del proyecto ISE en el que está trabajando, allí encontrará el resultado de la síntesis en un fichero *.bit*, concretamente, *sistema\_completo.bit*. Una vez seleccionado este fichero se activará el botón **Program** y bastará con pulsarlo para la placa se programe.



Figura 24. Programación de placas Digilent con Adept.

5. En la figura Error: No se encuentra la fuente de referencia se mostraba la interconexión de los componentes de la placa de desarrollo con el módulo multiplicador desarrollado. Fijándose en esta figura, utilice los conmutadores y los botones para realizar la multiplicación de varios números y compruebe si el resultado es correcto. El procedimiento es:
  - 5.1. Colocar los números con los conmutadores.
  - 5.2. Pulsar el botón *xs* para que comience la multiplicación.
  - 5.3. Generar el reloj manualmente pulsando el último botón repetidamente hasta que se iluminen todos los Leds, lo que indica que se ha activado la señal *fin* y ha terminado la multiplicación.





# El Computador Simple 2010 (CS2010)

*Enunciados de Prácticas de Laboratorio  
Estructura de Computadores*

## Introducción y objetivos

Los objetivos de esta práctica son:

- Ejercitar el uso del CS2010 y de su programación
- Comprobar el funcionamiento de la unidad de control
- Comprobar el funcionamiento de la unidad de datos
- Realizar y comprobar el funcionamiento de un programa sencillo para el CS2010 que realice operaciones de entrada-salida



Con estos objetivos se ha implementado<sup>3</sup> un sistema basado en el procesador CS2010 sobre la placa de prototipado Digilent Basys2 que se muestra en la figura Error: No se encuentra la fuente de referencia. El sistema implementado incluye los siguientes componentes:

- El procesador
- La memoria de código (instrucciones o programa)
- La memoria de datos
- Dispositivos de entrada-salida mapeados en memoria
- Una unidad de depuración

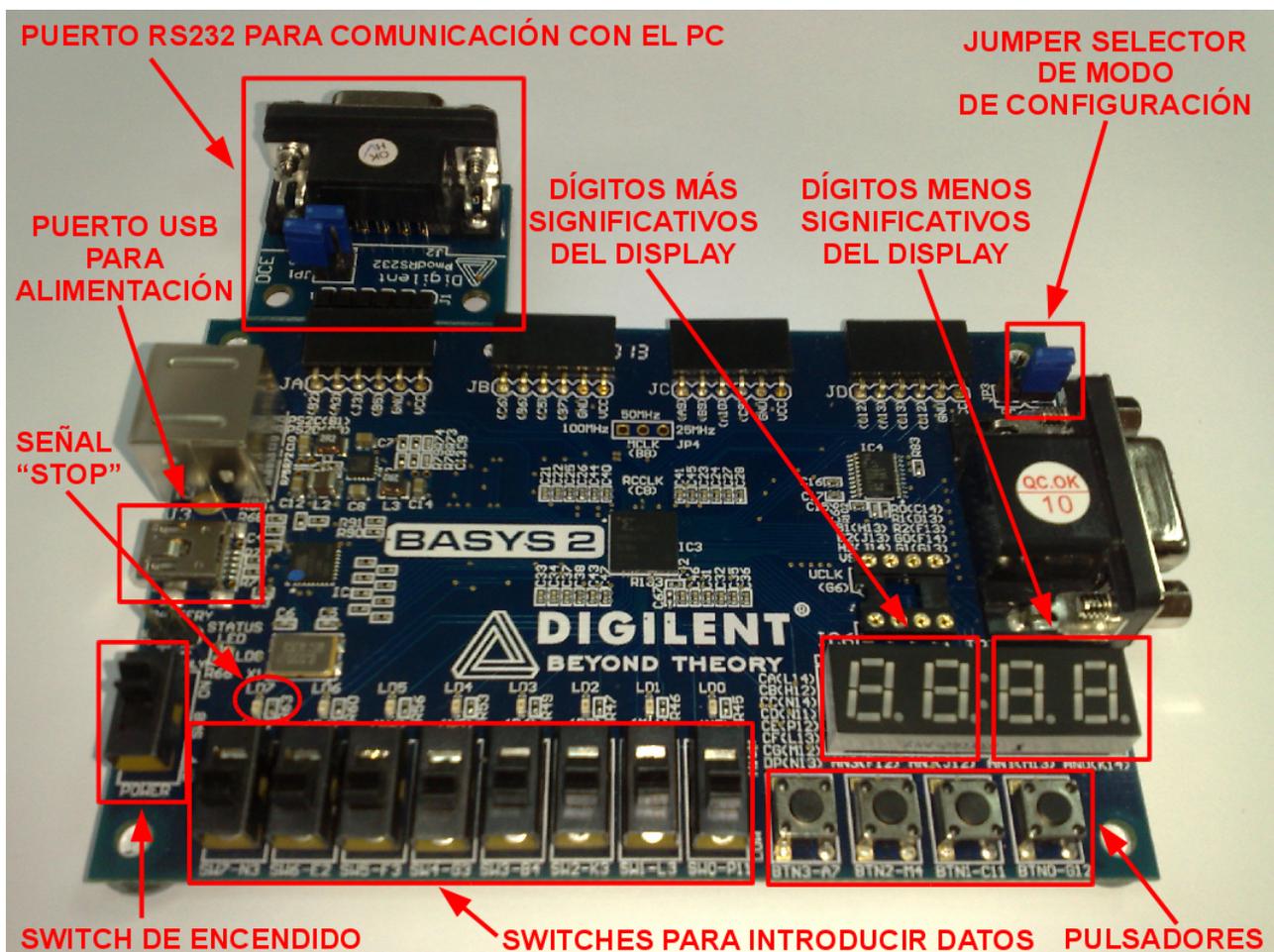


Figura 25. Placa de prototipado empleada en la práctica.

Los dispositivos de entrada-salida se han mapeado en el espacio de direccionamiento de la memoria de datos tal y como muestra la tabla siguiente:

Dirección	Sentido	Dispositivo	Descripción
-----------	---------	-------------	-------------

<sup>3</sup> La implementación ha sido realizada por Jonathan Ruiz Páez como proyecto fin de carrera

\$80	entrada	Switches	Cada uno de los 8 bits de esta posición de memoria indica si el switch correspondiente está activo. Así, el bit 7 valdrá uno si y sólo si sw7 está activo, el bit 6 valdrá uno si y sólo si sw6 está activo, etc.
\$81	entrada	Pulsadores	Los 4 bits menos significativos de esta posición de memoria indican si los botones correspondientes están pulsados. Así, el bit 3 valdrá uno si y sólo si btn3 está pulsado, el bit 2 valdrá uno si y sólo si btn2 está pulsado, etc. Los 4 bits más significativos de la posición siempre valen 0.
\$82	salida	dígitos menos significativos del display	El contenido de esta posición de memoria se muestra en hexadecimal en la derecha del display
\$83	salida	dígitos más significativos del display	El contenido de esta posición de memoria se muestra en hexadecimal en la izquierda del display

Tabla 4. Dispositivos mapeados en memoria.

La unidad de depuración es un componente que se comunica con el PC a través del puerto RS232 y permite a éste leer y escribir la memoria de código y datos. Además, la unidad de depuración hace posible depurar los programas descargados mediante:

- Ejecución de ciclos de reloj individuales.
- Ejecución de instrucciones individuales.
- Inspección del contenido de la memoria y los registros.
- Inspección del estado de las líneas de control.
- Habilitación/inhibición del reloj del procesador.

La interacción con la unidad de depuración se realiza a través del software descrito en los apéndices.

## Estudio teórico

6. Lea atentamente los apéndices y aprenda el formato de los programas en ensamblador del CS2010.
7. Escriba una subrutina del CS2010 que introduzca en el registro R7 el producto de R5 y R6 asumiendo notación sin signo.
8. Escriba un programa que, usando la subrutina anterior, calcule el producto de 22 por 10 y almacene el resultado en la posición de memoria 4.
9. Escriba una tabla que indique qué señales se activan y qué transferencias entre registros se realizan en cada ciclo de reloj durante la ejecución de las dos primeras instrucciones del programa del apartado anterior.
10. Usando la subrutina anterior, escriba un programa que implemente el siguiente algoritmo:

```
HACER
  MIENTRAS EL BOTÓN 0 (BTN0) NO ESTÉ PULSADO
    NO HACER NADA
  FIN MIENTRAS
R5<-NÚMERO CODIFICADO EN LOS SWITCHES
  MIENTRAS EL BOTÓN 3 (BTN3) NO ESTÉ PULSADO
    NO HACER NADA
  FIN MIENTRAS
R6<-NÚMERO CODIFICADO EN LOS SWITCHES
R7<-R5*R6
  MUESTRA R7 EN EL DISPLAY
POR SIEMPRE
```

11. Compruebe que la sintaxis de los programas escritos es correcta usando el software disponible en la página de la asignatura.

**NOTA IMPORTANTE:** Es imprescindible traer a la sesión de laboratorio los ficheros de los programas ya escritos en soporte electrónico y con la sintaxis ya chequeada.

## Estudio experimental

1. Cambie el jumper de selección de modo de configuración de la placa (JP3) para que conecte el pin central con el etiquetado con "ROM".
2. Conecte la placa al puerto RS232 del PC (vea la figura Error: No se encuentra la fuente de referencia para ver como conectar la extensión RS232 a la placa de prototipado).
3. Alimente la placa de prototipado a través del puerto USB del PC.
4. Cargue en la memoria de código el programa que realiza la multiplicación de 22 por 10 tal y como se describe en los apéndices.
5. Ejecute ciclo a ciclo las dos primeras instrucciones comprobando que en cada ciclo se activan las señales de control que esperaba y se realizan las transferencias entre registros que esperaba.
6. Ejecute instrucción a instrucción las dos líneas siguientes y compruebe que tienen el efecto esperado.
7. Habilite el reloj para ejecutar las instrucciones restantes y vuelva a deshabilitarlo para comprobar el resultado final.
8. Cargue en la memoria de código el programa que implementa el algoritmo del apartado 5 del estudio teórico y haga que se ejecute de forma autónoma habilitando el reloj del CS2010. Compruebe que el programa funciona correctamente multiplicando distintos valores.

**NOTA:** Si el software indica que no puede realizarse la conexión a pesar de haber conectado correctamente la extensión RS232, es posible que la placa no esté aun configurada para implementar el CS2010. En tal caso pulse sobre el icono "configurar BASYS2 para CS2010" y, cuando la grabación termine, apague y vuelva a encender la placa.

## Apéndice 1: Formato de los programas en ensamblador del CS2010

El código binario que ejecuta el procesador se denomina “código máquina”. Escribir directamente en código máquina no resulta práctico. En lugar de eso escribiremos los programas en un lenguaje más legible para los humanos denominado “lenguaje ensamblador”. Los programas en ensamblador pueden ser traducidos al código binario ejecutable por la máquina mediante una herramienta denominada también “ensamblador”. A esta acción se la denomina “ensamblar”. Un programa en ensamblador del CS2010 puede contener 3 tipos de líneas:

- Línea en blanco: contiene únicamente caracteres de espacio, tabuladores y/o comentarios. Estas líneas son ignoradas por el ensamblador.
- Línea de instrucción de programa: describe una instrucción del CS2010 mediante los nemónicos vistos en teoría. Si una instrucción requiere un operando “numérico”, este puede especificarse de cuatro formas:
  - Usando un numeral decimal, en cuyo caso puede ir precedido por un signo.
  - Usando un numeral binario, en cuyo caso debe ir precedido por el prefijo “0B” y tener exáctamente 8 bits.
  - Usando un numeral hexadecimal, en cuyo caso debe ir precedido por el prefijo “0x” o el prefijo “\$” y tener uno o dos dígitos hexadecimales.
  - Usando una etiqueta: una etiqueta es una cadena que tiene asociado un valor de 8 bits. Deben empezar por una letra y pueden contener un número arbitrario de letras y dígitos. Más adelante veremos la forma de asociar valores a etiquetas.

Son numerales válidos “1234”, “+2”, “-1”, “0b11000110”, “\$4F”, “0x32”, “\$3”. No son numerales válidos “12F3”, “+0b10101010”, “0b000000000000001”, “0b10”, “0x023”.

- Línea de directiva de ensamblado: le indica al ensamblador que realice determinadas acciones. En el CS2010 son de dos tipos:
  - Directiva EQU: permite asociar un valor a una etiqueta. Su sintaxis es:

```
.EQU <etiqueta> = <numeral>
```
  - Tras encontrar una línea como ésta, el ensamblador cambiará cualquier ocurrencia de la etiqueta que encuentre en las líneas de instrucción por el valor asociado. Nótese que estas líneas no se corresponderán con ninguna instrucción máquina del programa ensamblado.
  - Directiva OP CODE: permite insertar instrucciones indicando explícitamente su código máquina. Su sintaxis es:

```
.OPCODE <byte_más_significativo> , <byte_menos_significativo>
```

donde <byte\_más\_significativo> debe ser un numeral y <byte\_menos\_significativo> puede ser un numeral o una etiqueta.

Existe una segunda forma de asignar valores a etiquetas: tanto a las líneas de instrucciones de programa como a las líneas con la directiva OP CODE puede añadirse un prefijo en la forma:

```
<etiqueta> :
```

Esto hará que el ensamblador asocie a la etiqueta la dirección de la posición de memoria que ocupará la instrucción máquina correspondiente a esta línea. Esto es útil para hacer más claros los programas que usan instrucciones de salto.

Otra forma de hacer más legibles los programas es añadir comentarios al final de las líneas. Los comentarios comienzan por el carácter ';'. Todo el texto situado entre ese carácter y el final de la línea es ignorado por el ensamblador.

Por último hay que hacer notar que el ensamblador no distingue entre mayúsculas y minúsculas.

A continuación se muestra un ejemplo de programa en ensamblador del CS2010:

```
;esto es un comentario y será ignorado por el ensamblador

LDI R0,15; esto mete en R0 el número 15
    ;los espacios en blanco y tabuladores son ignorados
    LDI R1,$0F ;esto mete en R1 el número 15
LDI R2,-1 ;esto mete en R2 el número 255
;esto es otro comentario seguido de líneas en blanco

    .EQU menos1=0b11111111 ; esto no genera código máquina
.equ uno=1
    LDI R3,0
bucle:SUBI R0, uno
BRCS termina
    st (r3), r0 ; no importa si usamos minúsculas
    ADDI R3, UNO

jmp bucle
    TERMINA:      STOP

    ;al ejecutarse esto se rellenarán las 14 primerasposiciones de memoria
;con los enteros entre el 14 y el 1 ordenados de forma descendente
```

## Apéndice 2: Software de programación y depuración

Para interactuar con la unidad de depuración usaremos el software instalado en los PC del laboratorio. Dicho software está disponible en la página web de la asignatura. Para lanzarlo simplemente pulsamos sobre el icono correspondiente en el escritorio. Al hacerlo nos aparecerá la siguiente ventana:



Figura 26: Software de interfaz con la unidad de depuración del CS2010.

La parte superior de la ventana contiene una serie de controles para interactuar con la unidad de depuración del CS2010 que se describen a continuación:

- Campo para la ruta del puerto serie. Aparece relleno por defecto con la correspondiente al puerto serie de los ordenadores del laboratorio, por lo que en principio no debe cambiarse.
- Campo de fichero de código binario (BIN): este campo debe contener la ruta del fichero binario cuyo contenido se escribirá en la memoria de código cuando se inicialice el procesador. Al rellenarse se mostrará en la parte superior derecha de la ventana el contenido de la memoria desensamblado. Normalmente no tendremos que preocuparnos de rellenar este campo ya que usaremos un fichero en formato ensamblador para especificar el programa que deberá escribirse.
- Campo de fichero de código ensamblador (ASM): este campo debe contener la ruta de un fichero de código ensamblador correspondiente al programa que será escrito en la memoria de código. Al rellenar este campo la herramienta ensamblará el fichero informando de los posibles errores y escribirá la ruta del fichero binario resultante en el campo comentado anteriormente. Rellenar este campo no es obligatorio, pero es la forma habitual de trabajar con la herramienta.
- Campo de fichero de datos: si deseamos inicializar la memoria de datos con el contenido de un fichero binario, podemos indicar su ruta en este campo.
- Botón de establecimiento de conexión e inicialización: al pulsarlo el software establece una conexión a través del puerto RS232 con la unidad de depuración, ordena inicializar el CS2010 y escribe las memorias de código y datos con el contenido de los ficheros especificados. Únicamente tras esto podremos interactuar con la unidad de depuración.

- Botón “Mostrar Unidad De Datos”: al pulsar este botón aparecerá la ventana siguiente:

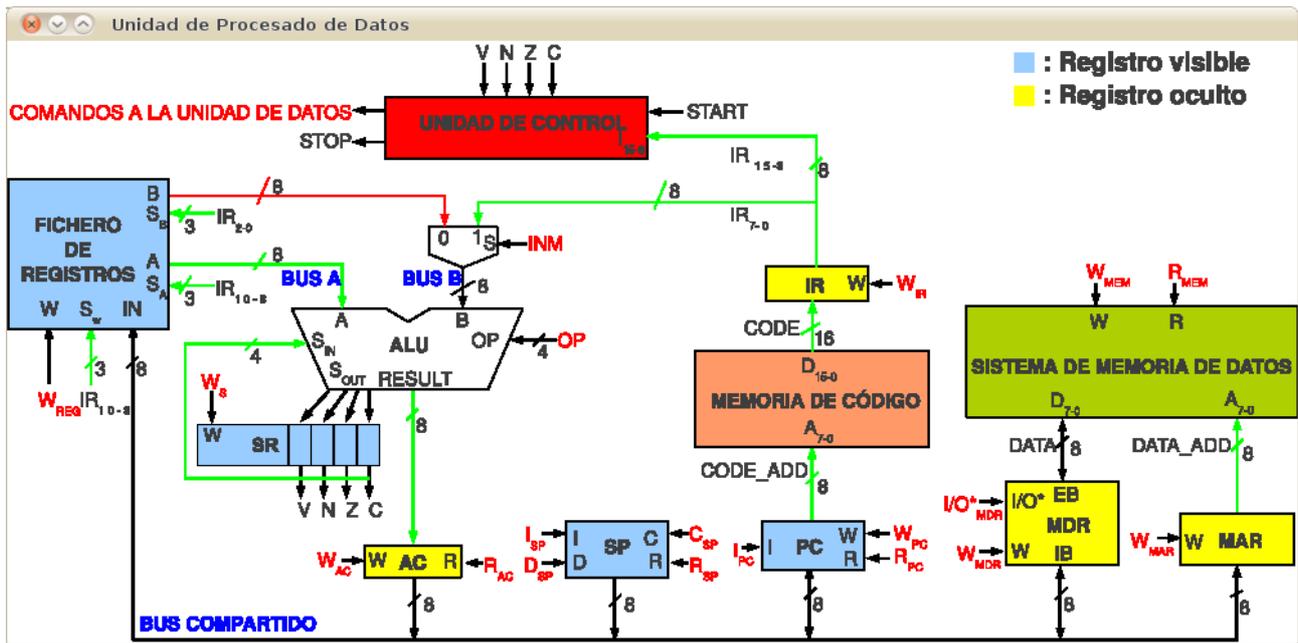


Figura 27: Unidad de datos del CS2010.

Durante las ejecuciones ciclo a ciclo se resaltarán en esta ventana las señales que se vayan activando.

- Botón “Habilitar Reloj”/“Deshabilitar Reloj”: Si el reloj del CS2010 está detenido, este control permite ponerlo en marcha para que se ejecute el programa de forma autónoma hasta encontrar una instrucción de STOP. Si el reloj está ya en marcha permite detenerlo para inspeccionar el estado del procesador y la memoria.
- Botón “Activar Start”: Le indica a la unidad de depuración que genere un pulso alto en la señal de inicio del CS2010.
- Botón “Ejecutar Un Ciclo”: Le indica a la unidad de depuración que habilite el reloj durante un solo ciclo.
- Botón “Ejecutar Una Instrucción”: Le indica a la unidad de depuración que habilite el reloj hasta que se ejecute la instrucción en curso o hasta que se llegue al estado de espera.

Cuando el reloj del CS2010 está detenido se muestra el estado del procesador incluyendo:

- Contenido de la memoria de datos.
- Contenido de los registros visibles.
- Contenido de los registros ocultos.
- Estado de las líneas de control.
- Valor de los buses de entrada y salida de la ALU y del bus común.

Además, la instrucción en curso aparecerá resaltada. El contenido de las posiciones memoria de datos puede ser editado manualmente. También es posible guardar el contenido de la memoria de datos en un fichero binario pulsando sobre el botón "Guardar Memoria Datos".



**DEPARTAMENTO DE TECNOLOGÍA ELECTRÓNICA**  
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

## **Introducción a AVR-STUDIO**

*Enunciados de Prácticas de Laboratorio  
Estructura de Computadores*

### **Introducción y objetivos**

Los objetivos de la sesión de laboratorio son los siguientes:

- Introducir el entorno de programación y depuración de microcontroladores de ATMEL<sup>4</sup> llamado AVR-STUDIO.
- Realizar la simulaciones de programas escritos en lenguaje ensamblador para el microcontrolador ATMEGA328P.
- Realizar la programación de un microcontrolador mediante AVR-STUDIO mediante la plataforma AVR-DRAGON.
- Realizar la programación, depuración de programas y control del microcontrolador desde AVR-STUDIO.

En esta sesión de laboratorio se utilizará el entorno de desarrollo AVR-STUDIO para programar el microcontrolador ATMEGA328P que se encuentra en una placa de desarrollo llamada Arduino<sup>5</sup> Duemilanove. La programación se realiza mediante la plataforma de depuración/programación AVR-DRAGON también del fabricante ATMEL.

AVR-STUDIO puede descargarse gratuitamente de desde las páginas del fabricante de ATMEL en <http://www.atmel.com>. Respecto a los Arduinos, éstos están diseñados para ser programados en un lenguaje de programación propio, transfiriéndose los programas a través de su puerto USB. En esta sesión de laboratorio no se utilizarán estas características, es decir, se programarán directamente en

---

<sup>4</sup> Fabricante de microcontroladores, más información en <http://www.atmel.com>

<sup>5</sup> Plataforma opensource de prototipado electrónico, mas información en <http://www.arduino.cc>

ensamblador. Por ello, se han realizado modificaciones en dichas placas. Aunque no es relevante para esta sesión de laboratorio, se puede consultar toda la información adicional sobre esta placas en <http://www.arduino.cc>

Durante la sesión de laboratorio se debe disponer de los ficheros indicados en la tabla 5. Algunos de los ficheros deben ser completados en el estudio teórico y otros se completarán durante la sesión de laboratorio.

Nombre del fichero	Contenido	Descripción
contador_0_10.asm	Programa contador de 0 a 10	Debe completarlo el alumno antes de asistir a la sesión de laboratorio.
contador_0_1000.asm	Programa contador de 0 a 1000	Debe completarlo el alumno antes de asistir a la sesión de laboratorio.
conmutadores.asm	Programa de control de conmutadores y leds	Debe completarlo durante la sesión de laboratorio.
contador_bcd.asm	Programa contador de pulsaciones en BCD	Debe completarlo durante la sesión de laboratorio.

Tabla 5. Ficheros necesarios durante la sesión de laboratorio.

Es **obligatorio** traer los programas del estudio teórico preparados para utilizarlos durante el desarrollo de la sesión de laboratorio.

## Estudio teórico

Se deben realizar dos programas en lenguaje ensamblador que incrementen un valor almacenado desde 0 hasta un valor determinado. Dichos programas cuando terminen la cuenta volverán a empezar de nuevo la cuenta desde 0.

A continuación se detallan los programas:

9. Programa contador 0 a 10: Realizar un programa en ensamblador que cuente de 0 a 10 utilizando un registro del microcontrolador. Cuando termine la cuenta el programa debe invertir el valor del PINC0 y volver a empezar, es decir, volverá a contar de 0 a 10 e invertirá el PINC0. Así indefinidamente.

Para realizar el programa correctamente se debe configurar el puerto C como salida, para ello se propone comenzar el programa utilizando el siguiente fragmento de código (*fichero contador\_0\_10.asm*):

```
; Programa contador de 0 a 10
; Cada vez que se pase por 10 se debe invertir el PINC0

.INCLUDE "m328pdef.inc"
.DEF  TMP=R19

        LDI    TMP,$FF
        OUT    DDRC,TMP      ; Configura el puerto C completo como salida
```

Código 6. Fichero contador\_0\_10.asm, plantilla de código para el programa contador 0 a 10.

10. **Programa contador 0 a 1000:** Realizar un segundo programa similar al anterior donde ahora la cuenta debe ser desde 0 a 1000. Tenga en cuenta que los registros del micro controlador son de 8 bits y solo pueden contar de 0 a 255. El fichero debe llamarse *contador\_0\_100.asm* y puede utilizar la plantilla de código suministrada.

## Estudio experimental

El estudio experimental se divide en dos bloques, el primero consiste en utilizar el programa realizado en el estudio teórico para familiarizarse con el entorno de desarrollo AVR-STUDIO. El segundo consistirá en completar fragmentos de código de algunos programas donde se controlará la entrada y salida del microcontrolador.

Antes de comenzar el estudio experimental asegúrese de disponer de todos los ficheros indicados en la tabla 5.

### 1. Introducción a AVR-STUDIO

Se utilizarán los programas realizados en el estudio teórico en el entorno de desarrollo AVR-STUDIO. Los pasos para crear un proyecto nuevo y poder escribir el código del programa se detalla a continuación en esta sección.

Una vez iniciado AVR STUDIO aparece un asistente para creación o apertura de un nuevo proyecto tal y como se muestra en la figura Error: No se encuentra la fuente de referencia. Si no apareciera el asistente hay que acceder al menú **Project** y seleccionar la opción **Project Wizard**.

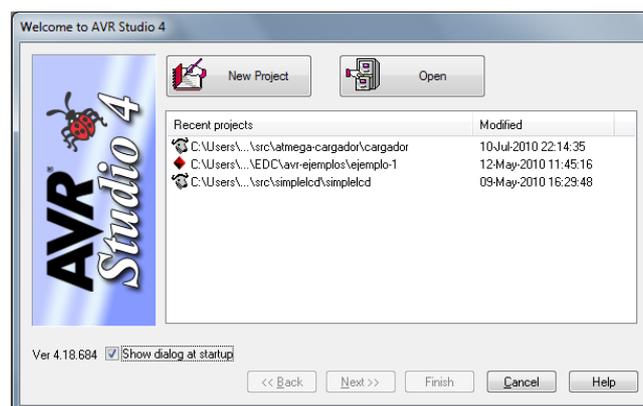


Figura 28. Asistente para creación o apertura de un proyecto.

Se debe seleccionar un nuevo proyecto (botón *New Project*) y aparecerá el siguiente diálogo del asistente (figura Error: No se encuentra la fuente de referencia) donde, habrá que indicar el nombre del proyecto, el directorio y seleccionar la opción *Atmel AVR Assembler*. Antes de pulsar el botón *Next* seleccione adecuadamente la opción *Create initial file*, tiene dos opciones, seleccionar o no seleccionar dicha opción (observe la marca roja en la figura Error: No se encuentra la fuente de referencia):

1. Si no selecciona esta opción, el proyecto se creará sin ningún archivo de texto asociado. Esto le

permite posteriormente utilizar un fichero de texto que tenga en si disco con el programa ya escrito. De esta forma evita tener que teclear el programa de nuevo

2. Si lo selecciona, se creará un nuevo fichero vacío en el que deberá teclear el programa. Si ya trae el programa escrito en otro fichero tendrá que copiar y pegar el código desde el bloc de notas a AVR-STUDIO.

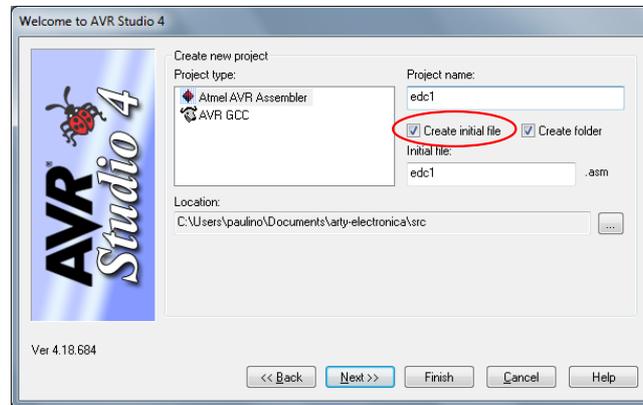


Figura 29. Selección de tipo y nombre de proyecto.

Tras escoger la opción que le interese en cada caso, tras pulsar el botón *Next* aparecerá la última ventana de asistente. Aquí debe seleccionar las opciones *AVR Simulator* y *ATMega328P* tal y como se muestra en la figura Error: No se encuentra la fuente de referencia.

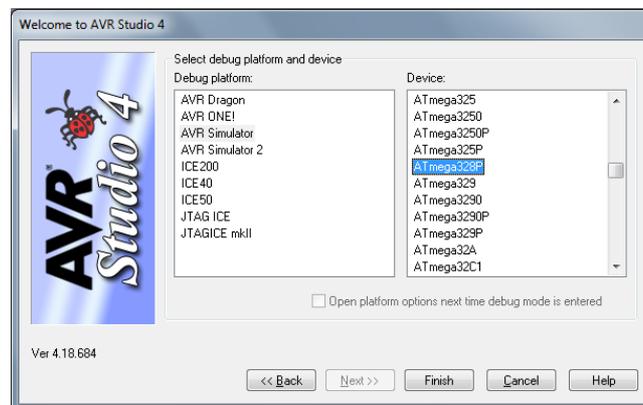


Figura 30. Selección de tipo y nombre de proyecto.

En caso de no haber seleccionado la opción *Create initial file* tendrá un proyecto vacío al que hay que añadir un programa previamente escrito en un fichero. Para realizar esto, hay que utilizar el botón derecho del ratón en la raíz del árbol de proyecto y aparecerá un menú flotante como el mostrado en la figura Error: No se encuentra la fuente de referencia. Con la opción **Add files to project** podemos seleccionar del disco el fichero con el programa en ensamblador que se desee.

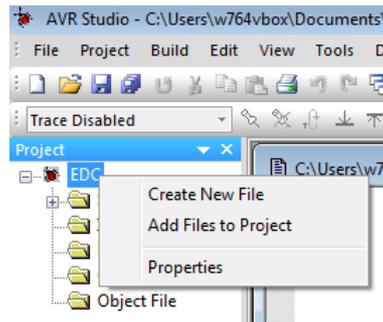


Figura 31. Añadir ficheros al proyecto.

Tras estos pasos aparece en el árbol de proyecto el nombre del fichero ensamblador sobre el que trabajar. Pulsando con el ratón dos veces sobre este nombre del fichero aparece una ventana en la que puede editar el código ensamblador del programa.

Una vez se ha terminado de escribir el programa hay que realizar el ensamblado del código. Este paso se realiza en menú **Build**, opción **Build**. También puede utilizar el icono  de la barra de herramientas. Si el código es correcto debe aparecer en la parte inferior información referente al programa ensamblado:

```
C:\Users\paulino\Documents\edc1\edc1.asm(4): Including file 'C:\Program Files (x86)\Atmel\AVR
Tools\AvrAssembler2\Appnotes\m168DEF.INC'

C:\Users\paulino\Documents\edc1\edc1.asm(23): No EEPROM data, deleting
C:\Users\paulino\Documents\edc1\edc1.eep

ATmega168 memory use summary [bytes]:
Segment  Begin    End      Code  Data   Used   Size  Use%
-----
[.cseg]  0x000000  0x000022    34    0     34   16384  0.2%
[.dseg]  0x000100  0x000100     0    0     0    1024  0.0%
[.eseg]  0x000000  0x000000     0    0     0     512  0.0%

Assembly complete, 0 errors. 0 warnings
```

Código 7. Salida de la construcción del programa contador.

En caso de producirse errores, en la ventana inferior aparecerá el número de línea del programa donde está el error.

## 1.1. Ejecución en el simulador del programa

AVR-STUDIO incluye un simulador con el cual se puede visualizar el estado del microcontrolador durante la ejecución de un programa. Entre las diversas opciones que ofrece el simulador nos centraremos en la posibilidad de ejecutar instrucción a instrucción un programa y la posibilidad de ejecutar un programa hasta que llegue a una instrucción determinada.

Para comenzar la simulación del programa hay que acceder al menú **Debug** y utilizar la opción **Start Debugging**. Tras esto aparecen diferentes ventanas (ver figura Error: No se encuentra la fuente de referencia) que componen el simulador:

- **Ventana del Procesador:** Situada en la parte izquierda, muestra el estado interno de procesador

(Frecuencia, contador de ciclos del reloj) y el contenido de los registros: PC, SP, X, Y, Z, SREG y los 32 registros internos.

- **Ventana de dispositivos de E/S:** Situada en la parte superior derecha, muestra en forma de árbol todos los dispositivos que tiene el microcontrolador seleccionado. En esta primera práctica de debe seleccionar el puerto C, de igual modo que se ha seleccionado la en la figura Error: No se encuentra la fuente de referencia. De esta forma se visualizan los tres registros que forman el puerto.
- **Ventana de visualización de Memoria:** Situada en la parte inferior derecha, permite ver en tiempo real el contenido de la memoria del microcontrolador. Se puede seleccionar entre memoria de programa, memoria SRAM y EEPROM. Principalmente interesará ver el contenido de la memoria SRAM a partir de la dirección \$100. Se debe recordar que hasta la dirección \$99 están mapeados los periféricos, por lo que no se debe usar como espacio de almacenamiento para los programas.

Para comprobar el funcionamiento del programa se debe realizar la ejecución paso a paso observando como cambian los valores de los registros y el PINC0. Hay que desplegar los registros en la ventana del procesador y el puerto C en la ventana de E/S para visualizar los registros del puerto.

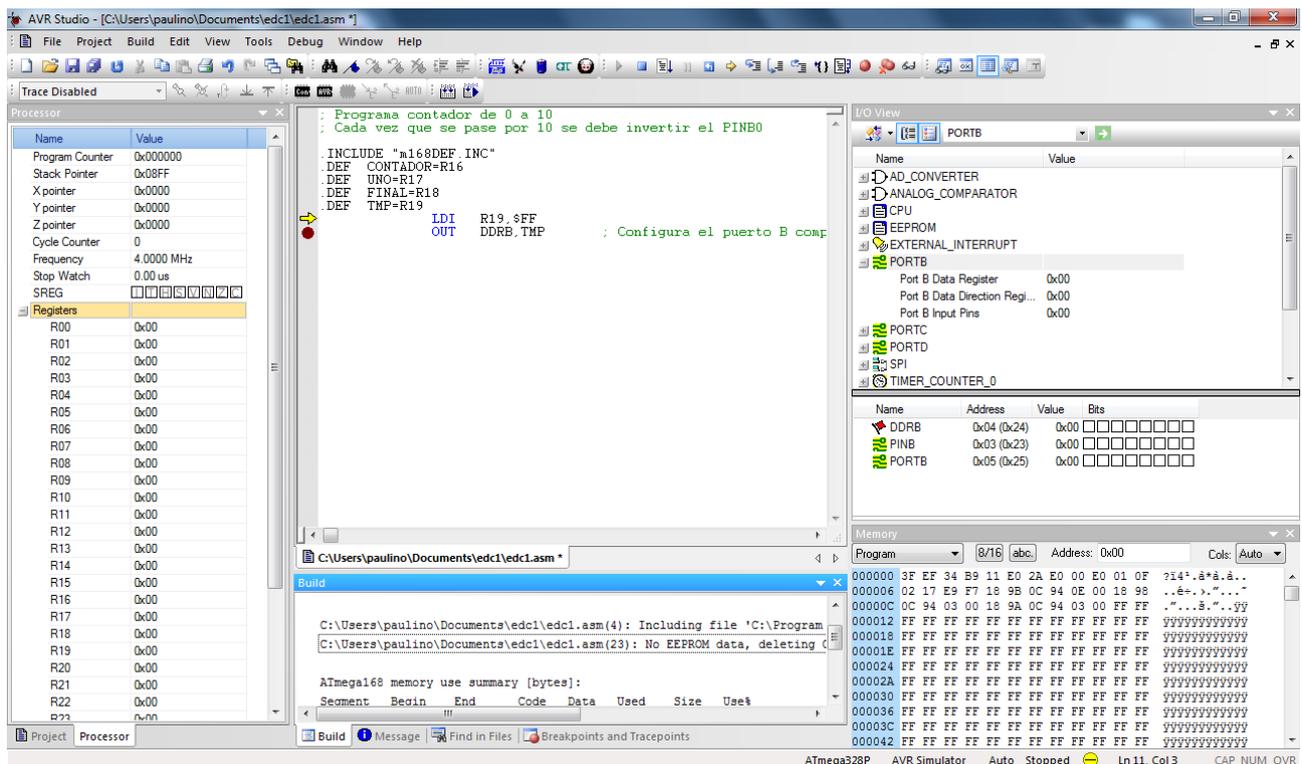


Figura 32. Visión global del modo de depuración de AVR-STUDIO.

El simulador permite la ejecución instrucción a instrucción del programa. El indicador  situado en la parte izquierda indica la siguiente instrucción que se ejecutará. En el menú **Debug** (figura Error: No se encuentra la fuente de referencia) se pueden encontrar diversas acciones útiles durante la simulación. En la ejecución paso a paso las funciones más utilizadas son:

- **Step Over:** (icono ) Ejecuta instrucciones hasta la siguiente línea, en caso de ser una llamada a

una subrutina, la ejecuta completamente para avanzar a la siguiente línea de código.

- **Step Into:** (icono ) Ejecuta una instrucción, en caso se existir una llamada a subrutina, realiza la llamada y se sitúa en la primera instrucción de la subrutina.
- **Step Out:** (icono ) Ejecuta instrucciones hasta encontrar una instrucción de retorno de subrutina.
- **Reset:** (icono ) Reinicia la simulación y sitúa la ejecución en la primera instrucción del programa.
- **Toggle Breakpoint:** (icono ) Establece un punto de ruptura de ejecución. Cuando se ejecute el comando *Run* (icono ) el programa se ejecutará hasta encontrar algún punto de ruptura.
- **Run to Cursor:** (icono ) Ejecuta instrucciones hasta la instrucción en la que está el cursor.

A continuación inicie la simulación con **Start Debuggin** y ejecute paso a paso el programa del estudio teórico. Puede utilizar la tecla F10 para no tener que utilizar los menús. Compruebe que su programa opera correctamente desplegando el puerto C en la ventana E/S y realice las siguientes tareas:

1. Ensamble y simule el primer programa del estudio teórico (*contador\_0\_10.asm*) programa y obtenga el número de ciclos que tarda el su programa en conmutar de valor el PINC0.
2. Calcule la frecuencia del señal cuadrada que se generará sabiendo que el procesador funciona a una frecuencia de 1Mhz. A partir de la frecuencia y el número de ciclos que tarda su programa en conmutar el PINC0 puede realizar este cálculo.
3. Utilice la opción **AutoStep** (icono ) y compruebe la ejecución animada que realiza el simulador del programa.

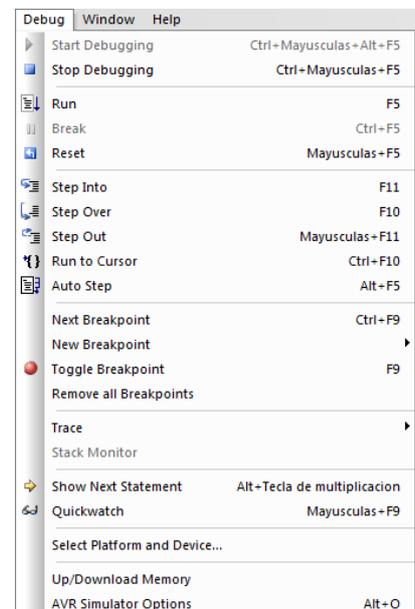


Figura 33. Menú de depuración.

## 1.2. Programación del microcontrolador

El siguiente paso consiste en la programación con el programador AVR-DRAGON (figura Error: No se encuentra la fuente de referenciaa) de un microcontrolador ATMEGA328P en una placa Arduino (figura Error: No se encuentra la fuente de referenciab). El entorno de pruebas utilizado en esta sesión de laboratorio está formada por tres componentes: programador AVR-DRAGON, placa de prototipo Arduino Duemilanove y placa de expansión para Arduino con componentes E/S.

La placa de expansión mostrada en la figura Error: No se encuentra la fuente de referencia está conectada a la placa Arduino, quedando ocultos todos los componentes del Arduino. En la placa de expansión están disponibles todos los puertos del microcontrolador en los laterales de la placa, además, estos puertos también están conectados a diversos componentes como son, leds, displays, conmutadores, etc. Estos componentes se utilizarán posteriormente para realizar programas que

controlen la entrada/salida.

En primer lugar se deben conectar ambas placas a los conectores USB. No es necesaria ninguna alimentación adicional ya utilizan los 5V suministrados por el Bus USB. Tras la conexión USB puede aparecer en el ordenador algún cuadro de diálogo indicando que se ha encontrado nuevo hardware. Si esto ocurriera, debe instalar los controladores, no cancele la instalación o tendrá problemas de programación del microcontrolador.

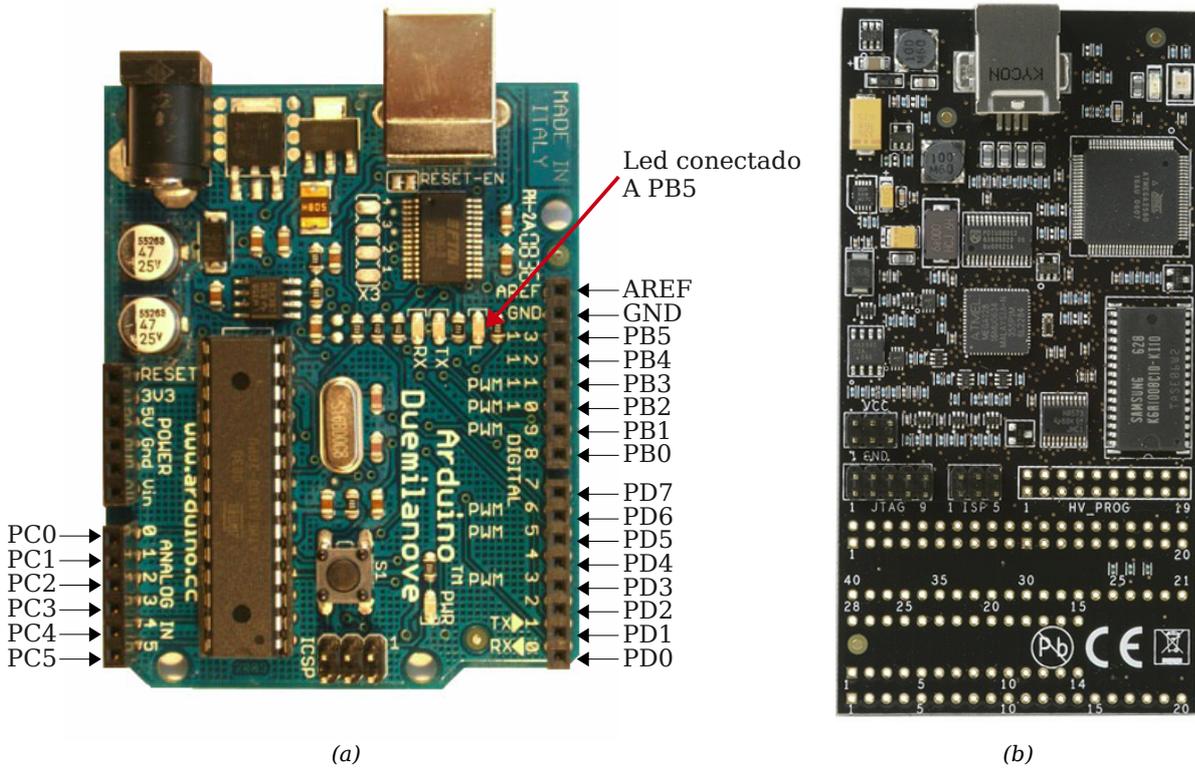


Figura 34. a) Placa de desarrollo Arduino b) Programador/Depurador AVR-DRAGON.

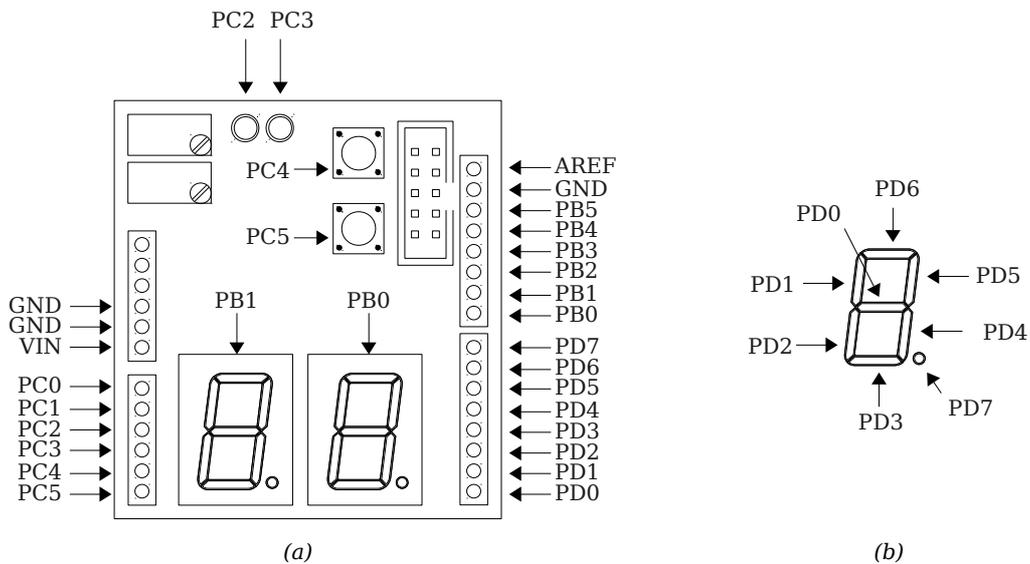


Figura 35. a) Placa de expansión E/S para Arduino. b) Detalle de conexión de los segmentos a los puertos.

La placa AVR-DRAGON dispone de dos leds, inicialmente se iluminan uno en verde y otro en rojo. El led de color rojo cambiará de color indicando el estado de la comunicación con AVR-STUDIO. La tabla Error: No se encuentra la fuente de referencia muestra el significado de los diferentes colores de dicho led, debemos observarlo durante los siguientes pasos para detectar posibles problemas en la programación del microcontrolador.

Color	Descripción
Rojo	En reposo, no hay conexión con AVR Studio
Apagado	En reposo, conectado a AVR Studio
Verde	Transfiriendo datos
Amarillo	Inicialización o actualización del firmware

Tabla 6. Indicaciones del led de AVR-DRAGON.

Antes de realizar la programación se debe verificar la correcta configuración de AVR-STUDIO realizando una prueba de conexión con el microcontrolador. Accediendo al menú **Tools** hay que usar el submenú **Program AVR** y, opción **Connect**. Aparecerá el diálogo mostrado en la figura Error: No se encuentra la fuente de referencia. Alternativamente, dicho diálogo se puede obtener de manera directa utilizando el botón **Con** de la barra de herramientas.

En este diálogo hay que establecer la configuración indicada en la figura Error: No se encuentra la fuente de referencia: plataforma AVR-DRAGON y puerto USB. Tras pulsar el botón **Connect**, si la conexión es correcta, debe aparecer automáticamente el diálogo mostrado en la figura Error: No se encuentra la fuente de referencia y el led rojo de AVR-DRAGON se apagará.

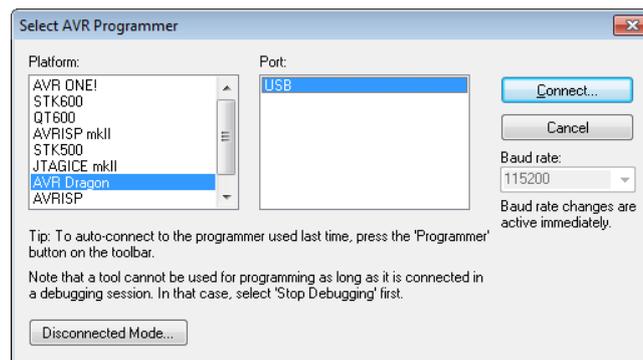


Figura 36. Selección del programador y el puerto

En caso de no aparecer automáticamente el diálogo de la figura Error: No se encuentra la fuente de referencia se puede utilizar el botón de la barra de herramientas **AVR** o, la opción de menú **Tools** submenú **Program AVR**. Tras esto finalmente aparecerá la ventana mostrada en la figura Error: No se encuentra la fuente de referencia.

De las múltiples pestañas que contiene sólo utilizaremos la primera y segunda: *Main* y *Program*. En primer lugar se realizará una prueba de comunicación siguiendo estos pasos:

1. Seleccionar la pestaña *Main*.
2. Seleccionar el microcontrolador correcto del cuadro desplegable indicado con *Device and Signature Bytes*. En estas placas disponemos del microcontrolador ATmega328P.
3. Pulsar el botón *Read Signature*. El programa debe responder con el texto *Signature matches selected device*. Si respondiera con un error, se debe volver a desplegar el cuadro selector de microcontrolador, seleccionar el correcto, y volver a realizar el test de comunicación.

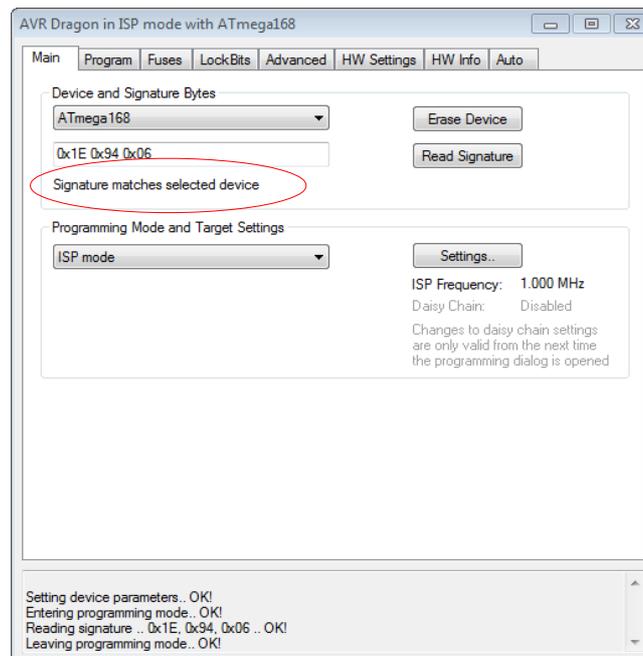


Figura 37. Pestaña principal de la ventana de programación del microcontrolador.

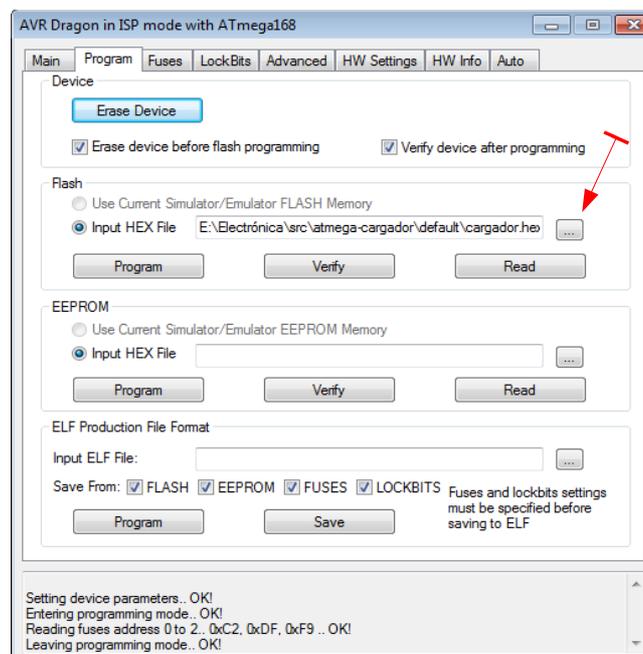


Figura 38. Pestaña de programación de la ventana de programación del microcontrolador.

El siguiente paso consiste en realizar la programación del microcontrolador con el código que se ha ensamblado, para ello, seleccione la pestaña *Program*. Si el ensamblado se realizó con éxito habrá

generado un fichero con extensión *.hex* dentro del directorio del proyecto y con el nombre del proyecto. En la figura Error: No se encuentra la fuente de referencia se muestra el diálogo de programación donde hay que seleccionar el fichero *.hex*. Este diálogo tiene tres cuadros donde se puede seleccionar un fichero: *Flash*, *EEPROM* y *ELF*. Hay que utilizar la sección *Flash* y el botón de selección de fichero (indicado con la flecha roja en la figura Error: No se encuentra la fuente de referencia). Tras esto basta con pulsar el botón *Program* para realizar la programación del microcontrolador.

Una vez realizada la programación hay que comprobar si el programa se está ejecutando correctamente. Para ello, debe conectar un canal del osciloscopio en el PINC0 y comprobar si se visualiza una señal cuadrada entre 0 y 5 voltios (no olvide conectar la tierra del osciloscopio). Utilice el esquema de la figura Error: No se encuentra la fuente de referencia para realizar las conexiones y realice las siguientes tareas:

1. Programe en el microcontrolador con el primer programa (*contador\_0\_10.asm*) y compruebe con el osciloscopio la frecuencia de la señal cuadrada generada en el PINC0.
2. Utilizando el segundo programa del estudio teórico (*contador\_0\_1000.asm*), compruebe con el osciloscopio la frecuencia de la señal cuadrada generada en el PINC0.
3. De forma experimental y ayudado por el resultado anterior, modifique el valor de final de cuenta del programa hasta conseguir que la frecuencia de dicha señal sea aproximadamente 5HZ. Programe el microcontrolador según modifique el programa hasta conseguir visualizar la frecuencia de 5Hz en el osciloscopio.
4. Modifique el programa de forma que el PIN que conmute sea el PINC2, de esta forma debe observar el LED amarillo parpadear 5 veces por segundo.

## **2. Realización de diversos programas de control E/S**

En esta sección se van realizar una serie de programas para controlar los componentes de entrada salida de la placa de desarrollo: conmutadores, leds y displays 7 segmentos.

Debe utilizar las plantillas de código preparadas para cada programa y escribir el fragmento de código indicado en cada uno de ellos.

### **2.1. Programa para controlar los conmutadores**

Se pretende realizar un programa que permita manejar los puertos de entrada salida. En concreto, se trata de activar los leds cuando se pulsa un conmutador.

En el esquema de la figura Error: No se encuentra la fuente de referencia aparecen dos leds y dos conmutadores que debe operar de la siguiente forma: cuando se pulse el conmutador conectado a PC5 debe encenderse el led conectado a PC2 y mantenerse encendido hasta que se vuelva a pulsar el conmutador. Además, al pulsar el conmutador conectado a PC4 se encenderá el led conectado a PC3 y permanecerá encendido hasta que se pulse nuevamente el conmutador. La tabla Error: No se encuentra la fuente de referencia muestra los puertos y los bits asociados a los componentes así como la configuración necesaria para que operen correctamente.

Puerto	Bit	Componente	Configuración	Funcionamiento
PORTC	2	Led	Como salida DDRC2=1	PINC2=0 ⇒ apagado PINC2=1 ⇒ encendido
PORTC	3	Led	Como salida DDRC3=1	PINC3=0 ⇒ apagado PINC3=1 ⇒ encendido
PORTC	4	Conmutador	Como entrada DDRC4=0	PINC4=0 ⇒ pulsado PINC4=1 ⇒ no pulsado
PORTC	5	Conmutador	Como entrada DDRC5=0	PINC5=0 ⇒ pulsado PINC5=1 ⇒ no pulsado

Tabla 7. Configuración de los puertos e/s de los leds y conmutadores

Utilizando el la plantilla de código Error: No se encuentra la fuente de referencia (fichero *conmutadores.asm*) debe realizar la siguientes tareas:

1. Cree un nuevo proyecto utilizando el código suministrado en el fichero *conmutadores.asm* y complete el programa.
2. Utilice el simulador para comprobar que funciona correctamente. Debe conmutar manualmente los pines PC4 y PC5 desde el simulador. Esto se consigue desplegando el puerto C en árbol de dispositivos que muestra el AVR-STUDIO en la parte derecha durante la simulación y pulsando el botón del ratón sobre el cuadro que representa el bit correspondiente. Cuando el cuadro está relleno de color negro significara que el bit está a 1, si está en blanco es 0.
3. Una vez comprobado en el simulador el correcto funcionamiento, repita los pasos realizados en la sección 1.2 para programar el microcontrolador con este nuevo programa. Compruebe que funciona correctamente pulsando los conmutadores.

```
.include "m328pdef.inc"
.def temp = r16          /* Define un registro para uso temporal se ha utilizado el r16
                          para poder emplear los modos con direccionamiento indirecto. */

ldi    temp,$c
out    ddrc,temp        // Configura portc[3:2] como salidas y el resto como entradas

ldi    temp,$30
out    portc,temp       // Activa las resistencias de pull-up del portc[1:0]

// Debe completar un bucle que lea continuamente PC5 y PC4.
// Cuando cambie el pin PC5 a 1 debe invertir el valor del pin PC4
// Cuando cambie el pin PC4 a 1 debe invertir el valor del pin PC3
// El bucle debe ser infinito
bucle: // Escriba el código a partir de aquí

...
```

Código 8. Fichero *conmutadores.asm*, plantilla de código para el controlador de pulsadores y leds.

## 2.2. Programa contador de pulsaciones

El nuevo programa a completar debe contar el número de pulsaciones de un conmutador y mostrarlo en el display 7 segmentos. Habrá que completar tres fragmentos de código; el primero es la inicialización correcta de los puertos, el segundo es una subrutina que crea una tabla en memoria con el código 7 segmentos y, el tercero es el programa principal.

La tabla Error: No se encuentra la fuente de referencia muestra la información de los componentes de entrada/salida que se usarán. Se incluyen los puertos, los bits asociados a los componentes así como la configuración necesaria para que operen correctamente.

Puerto	Bit	Componente	Configuración	Funcionamiento
PORTD	0-7	Segmentos de los displays	Como salida DDRD=0xFF;	PORTDX=0 ⇒ apagado PORTDX=1 ⇒ encendido
PORTB	0	Display 0	Como salida DDRB0=1	PORTB0=0 ⇒ encendido PORTB0=1 ⇒ apagado
PORTB	1	Display 1	Como salida DDRB1=1	PORTB1=0 ⇒ encendido PORTB1=1 ⇒ apagado

Tabla 8. Configuración de los puertos e/s de los displays 7 segmentos.

Utilizando el fichero *contador\_bcd.asm* mostrado en el listado de código Error: No se encuentra la fuente de referencia debe realizar las siguientes tareas:

1. Completar la subrutina de inicialización de puertos llamada *inicializa\_puertos*. Puede utilizar como ejemplo de inicialización la utilizada en el programa de la sección anterior (listado de código Error: No se encuentra la fuente de referencia). Debe inicializar los puertos con la siguiente configuración:
  - 1.1. En el puerto C los pines 3 y 2 deben ser salidas, el resto deben ser entradas.
  - 1.2. El puerto D está conectado a los segmentos del display, deben ser todos salida.
  - 1.3. El puerto B controla en encendido o apagado completo de cada uno de los dos displays. Debe configurarlo como salida, así, poniendo un 0 en PORTB0 se activará el display 0 y poniendo un 0 en PORTB1 se activará el display 1.
2. Completar la subrutina que crea una tabla para el convertidor de 7 segmentos llamada *inicializa\_tabla7seg*. Esta tabla contiene los códigos 7 segmentos de los dígitos 0 - 9. Al escribir un elemento de esta tabla en el puerto D aparecerá un número BDC en los displays. Como ejemplo se muestran 2 números, donde se puede observar la correspondencia de los bits a uno con la activación de los segmentos mostrados en la figura Error: No se encuentra la fuente de referencia. Complete los números que faltan, del 2 al 9.
3. El bucle principal del programa comienza a partir de la etiqueta *bucle*. Aquí debe escribir el programa que cuente las pulsaciones detectada en un conmutador. El programa se puede realizar siguiente estos pasos:
  - 3.1. Escribir un bucle que espere hasta detectar que el conmutador se ha pulsado. Un valor 1 en el pin correspondiente al conmutador indica que se ha pulsado.
  - 3.2. Tras detectar la pulsación hay que incrementar el contador en 1.
  - 3.3. Comprobar si el contador ha llegado a 10 para ponerlo de nuevo a cero.
  - 3.4. Esperar en un bucle hasta que se suelte en botón, fíjese que este fragmento de código ya está hecho y corresponde a la etiqueta *espera*.

4. Construya el programa y programe el microcontrolador para comprobar si funciona. Si no opera correctamente puede utilizar el simulador para detectar los errores. Tenga en cuenta que a veces existen problemas de rebotes en los conmutadores, esto significa que, al pulsar una vez el conmutador se detectan varias pulsaciones y el valor mostrado en el display se incrementa en más de una unidad.

```

.include "m328pdef.inc"
.def temp = r16      /* Define un registro para uso temporal se ha utilizado el r16 para
                    poder emplear los modos con direccionamiento indirecto.*/
.def contador = r17 // Cuenta el número de pulsaciones
.def cero= r18

.dseg
.org $100

TABLA7SEG: .byte 10 // Se reservan 10 bytes para una tabla de valores del convertidor bin7seg
.cseg
.org $0

rcall inicializa_puertos    // Rutina que inicializa los puertos
rcall inicializa_tabla7seg // Rutina que inicializa la tabla del convertidor

bucle:                // Programa principal
rcall display              // Representamos el valor de contador en el display

/* Aquí debe escribir el programa que haga lo siguiente:
1. Esperar hasta que se pulse un pulsador
2. Si se pulsa el pulsador incrementar el contador
3. Si el contador llega a 10 hay que ponerlo a cero */

...

espera:              // Este fragmento de código esperamos a que se suelte sw1
sbis   pinc,4          // si no se pone se incrementaría muchas veces el contador
rjmp   espera         // en el momento que se pulse
rjmp   bucle

inicializa_puertos:
// Aquí debe configurar portc[3:2] como salidas y el resto como entradas

...

// Aquí debe configurar puerto D y el puerto B completo como salida

out    portc,temp     // Se Activan las resistencias de pull-up del portc[1:0] y apaga leds
ldi    temp,$ff
ret

/* La siguiente rutina inicializa la tabla de 7 segmentos. Esta rutina sería innecesaria si se
hubiera utilizado la memoria de programa para almacenarla */

inicializa_tabla7seg:
ldi    zh,high(TABLA7SEG) // Utilizamos Z para apuntar a la tabla
ldi    zl,low(TABLA7SEG)  // low() high() son directivas que devuelven el byte bajo o
                          // el byte alto de la dirección que se le pasa como
                          // argumento respectivamente

ldi    temp,0b01111110    //Código 7 segmentos del 0
st     z+,temp
ldi    temp,0b00110000    //Código 7 segmentos del 1
st     z+,temp

/* Aquí debe completar los códigos de los dígitos que faltan: del 2 al 9 */

...

ret

/* La siguiente rutina permite representar un número en el display de 7 segmentos. Utiliza para
ello el registro Z, que inicialmente apunta a la tabla de 7 segmentos. A este registro se le suma
Cont que es una variable entre 0 y 9 y, después, mediante acceso indirecto se carga el código 7

```

```
segmentos correspondiente en el puerto.*/  
display:  
  ldi    zh,high(TABLA7SEG)  
  ldi    zl,low(TABLA7SEG)  
  add    zl,contador    // El registro Z es de 16 bits, mientras que contador es de 8  
  adc    zh,cero        // No olvidar sumar el acarreo que se genera del byte bajo a ZH  
  ld     temp,z  
  out    portd,temp  
  sbi    portb,1        // Apaga el display 1  
  cbi    portb,0        // Activa el display 0  
  ret
```

Código 9. Fichero contador\_bcd.asm, plantilla de código para el contador BCD.

### 2.3. Opcional: Mejoras en el programa contador de pulsaciones

Opcionalmente se proponen hacer algunas mejoras en el programa contador, estas son: utilizar los dos conmutadores para incrementar/decrementar y realizar un control de rebote del conmutador

Se propone realizar las siguientes tareas donde debe partir del programa realizado en la sección anterior:

1. Modifique el bucle principal del programa para que se comprueben las pulsaciones de los dos conmutadores. Si se pulsa el primero, el contador debe incrementarse, si se pulsa el segundo, el contador debe decrementarse. No olvide comprobar antes el decremento si es cero para establecerlo a 9, si no el decremento fallará.
2. Para realizar el control de rebotes se propone paralizar la ejecución del programa entre 5-10mseg. Para ello realice las siguientes modificaciones en el programa:
  - 2.1. Cree una subrutina llamada *no\_rebote* que consista en un bucle que espere un tiempo determinado. Sabiendo que el procesador funciona a 1Mhz utilice un contador que mantenga un bucle contando durante unos 10ms aproximadamente. Esta subrutina es similar al programa realizado en el estudio teórico.
  - 2.2. Modifique el programa principal incluyendo una llamada a esta nueva subrutina cuando detecte que el conmutador se ha pulsado. Además, debe llamar de nuevo a esta subrutina cuando detecte que el conmutador se ha soltado, ya que los rebotes pueden ocurrir tanto en al pulsar como al soltar el conmutador.
  - 2.3. Pruebe su nuevo programa en la placa de desarrollo.