Departamento de Tecnología Electrónica
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Multiplicador Secuencial 4x4

Enunciados de Prácticas de Laboratorio
Estructura de Computadores

1. 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 micro-operación 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 1 resume el contenido y el objetivo de cada uno de ellos.

Nombre del fichero	**Contenido**	**Descripción**
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 1. Ficheros necesarios durante la sesión de laboratorio.

2. Estudio teórico

Estudie adecuadamente la descripción del multiplicador secuencial que aparece en la sección 3 y realice dos tareas:

1. A partir de la carta ASM propuesta para el multiplicador secuencial (figura 5), realice la unidad de control en Verilog mediante una descripción canónica. Para ello utilice el código del fichero *u_control.v* (ver código 1 mostrado al final de esta sección). Este apartado es **absolutamente imprescindible** para realizar la práctica.
2. 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 :
   2.1. A = 1011 B = 0001
   2.2. 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 1. Fichero u_control.v, plantilla de código para la unidad de control.

3. 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 1).

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 2. Éste surge del algoritmo de multiplicación que se describe en la sección 3.1 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 3. Las salidas de estos registros son incondicionales.

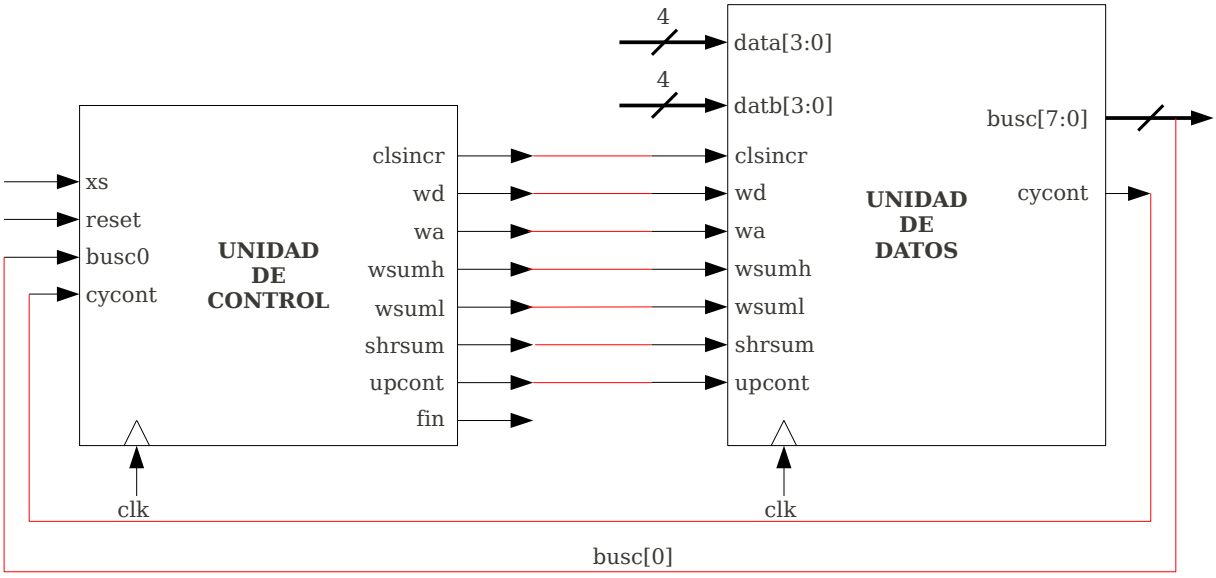


Figura 1. Representación a nivel de bloques del multiplicador secuencial.

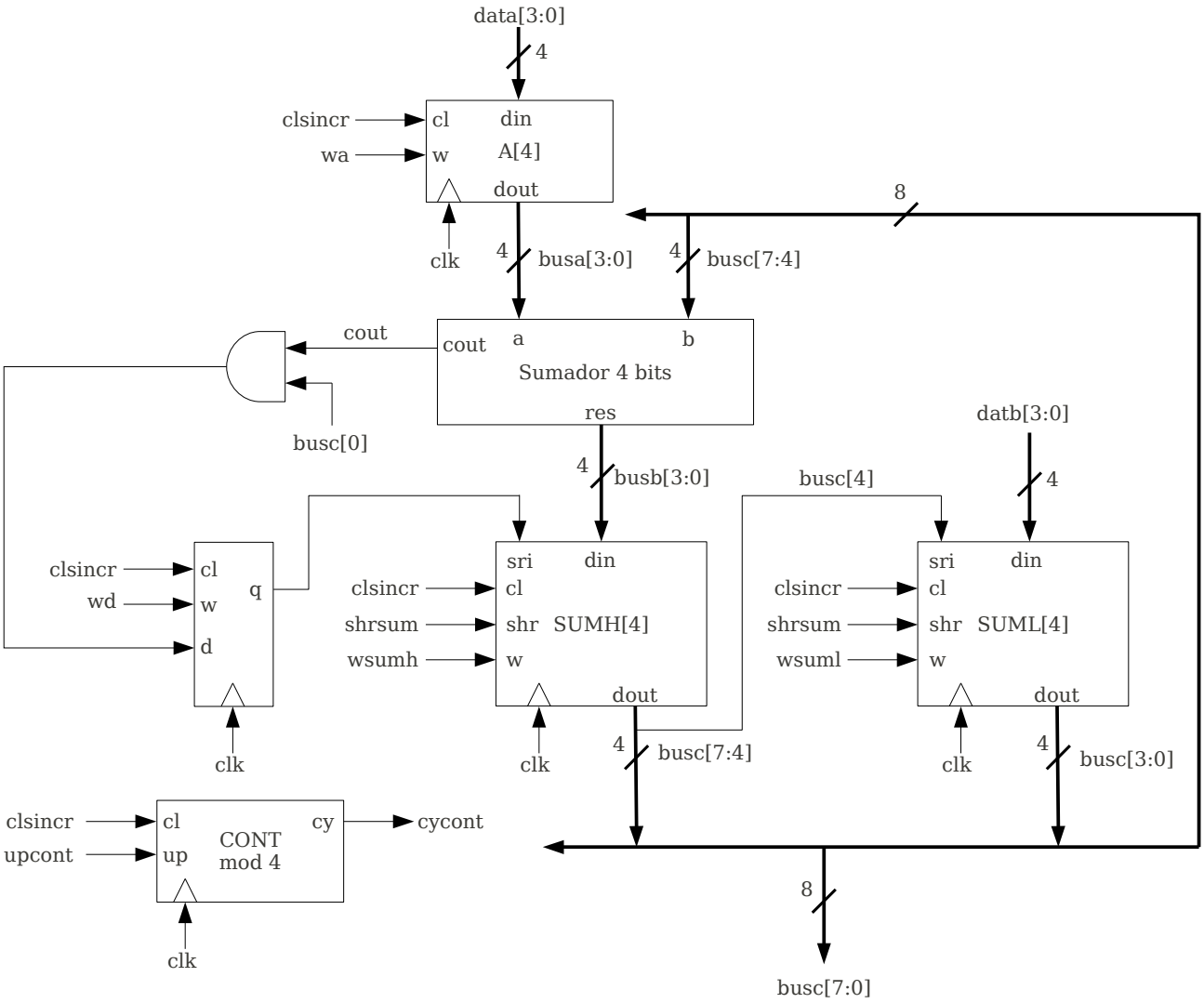


Figura 2. Representación estructural de la ruta de datos.

clk	cl	w	Operación
↑	1	-	A ← 0
↑	0	1	A ← DATA[3:0]
↑	0	0	A ← A

Registro A

clk	cl	up	Operación
↑	1	-	CONT ← 0
↑	0	1	CONT ← CONT+1
↑	0	0	CONT ← CONT

Contador

clk	cl	w	shr	Operación
↑	1	-	-	R ← 0
↑	0	1	-	R ← Din[3:0]
↑	0	0	1	R ← SHR(R,sri)
↑	0	0	0	R ← R

Registro de desplazamiento

clk	cl	w	Operación
↑	1	-	D ← 0
↑	0	1	D ← Din
↑	0	0	D ← D

Biestable

Figura 3. Descripción RT de los componentes de la ruta de datos.

3.1. 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 4 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 se ser este bit 0, se realiza únicamente el desplazamiento de un bit (un único paso).

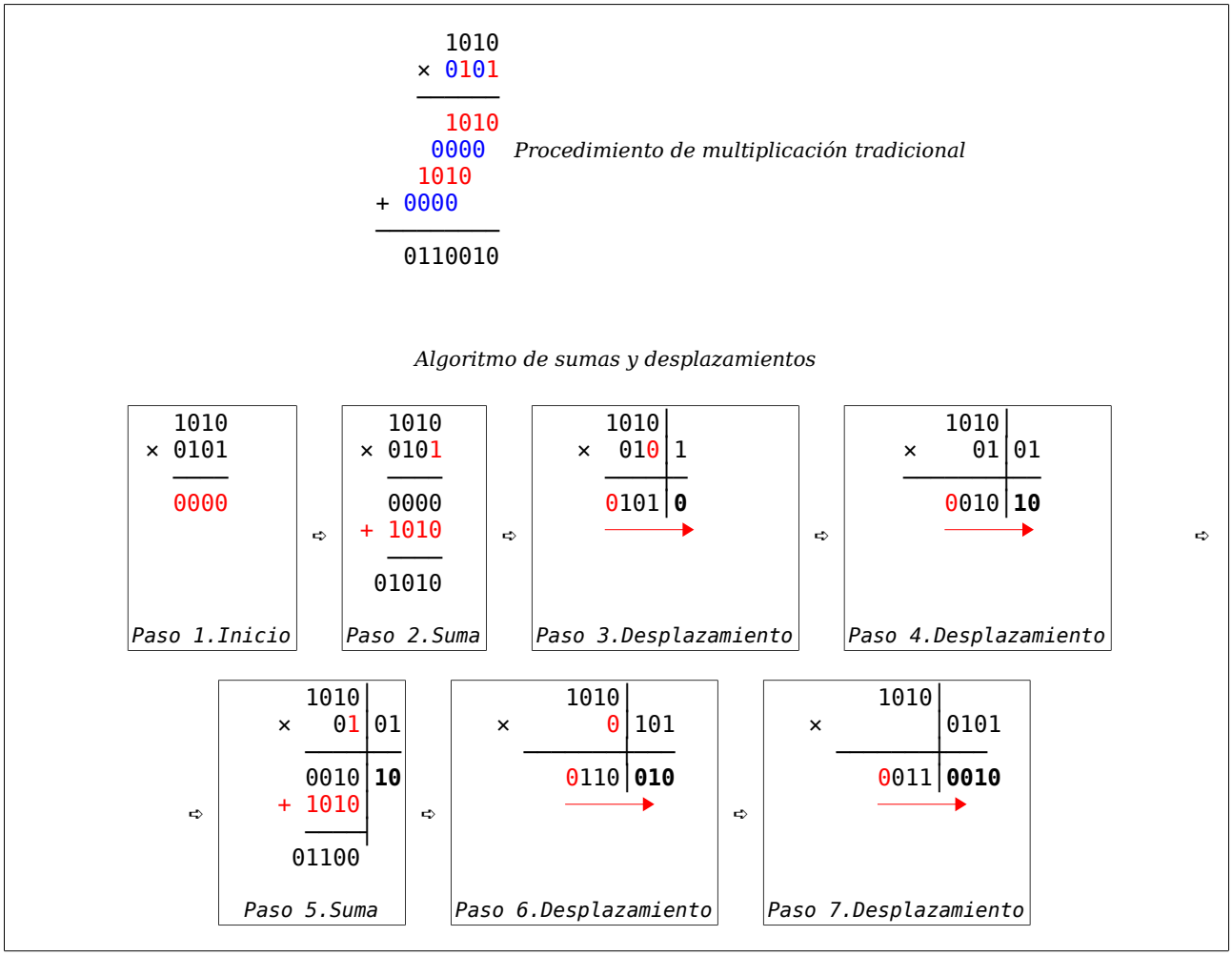


Figura 4. Algoritmo de multiplicación por suma y desplazamiento.

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 decrementa 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.

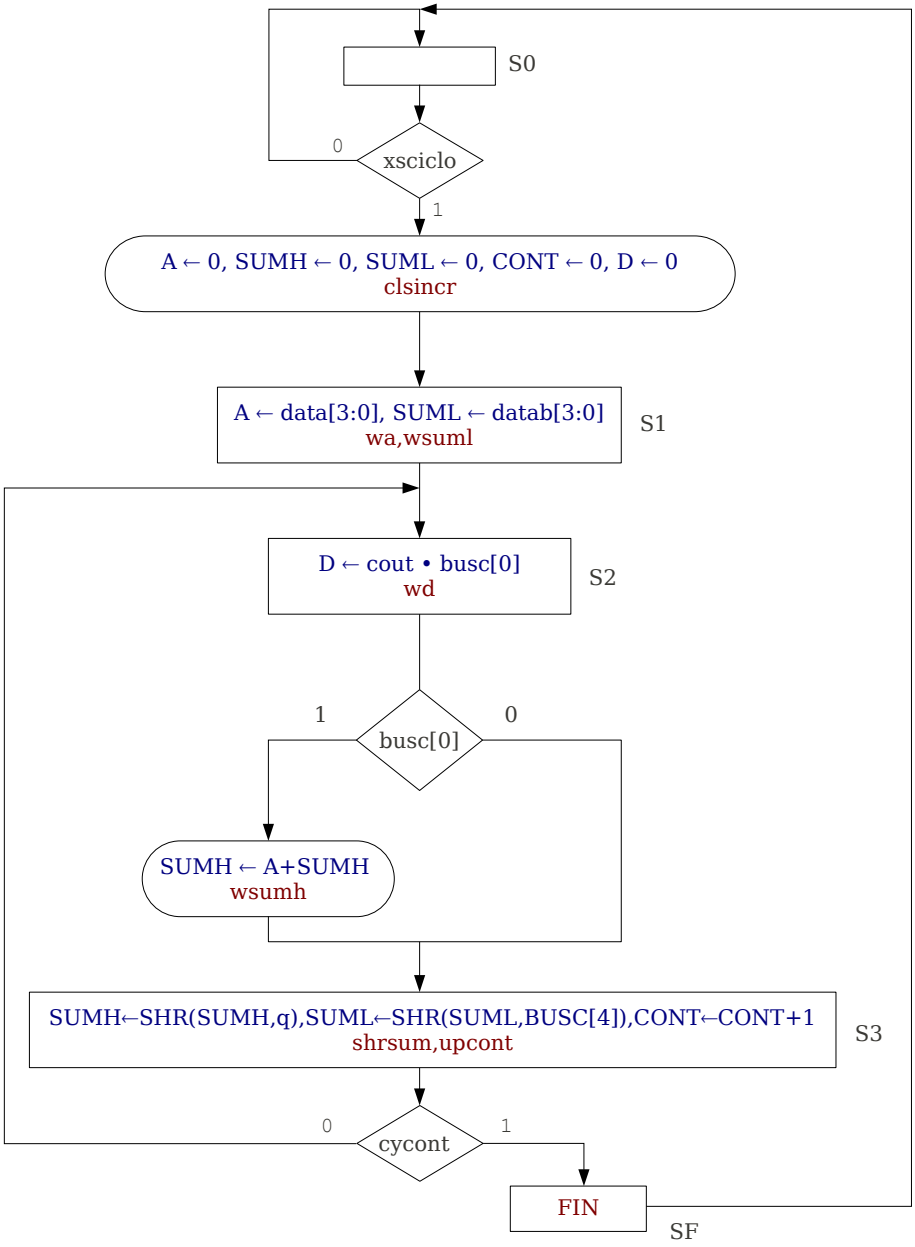


Figura 5. Carta ASM de datos y de control del multiplicador secuencial.

Una carta ASM que da respuesta al algoritmo y Unidad de Datos anteriores es la que se muestra en la figura 5. De ella se obtiene directamente la Unidad de Control, la cual deberá ser obtenida por el alumno usando una descripción canónica en Verilog.

3.2. Sistema completo en la placa de desarrollo

Para probar el diseño digital realizado se utilizará una placa desarrollo del fabricante Digilentic[1]

1 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 (fichero *sistema_completo.v*) se construye un sistema digital como el indicado de manera esquemática en la figura 6. 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 (d0, d1 d2 y d3) 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.

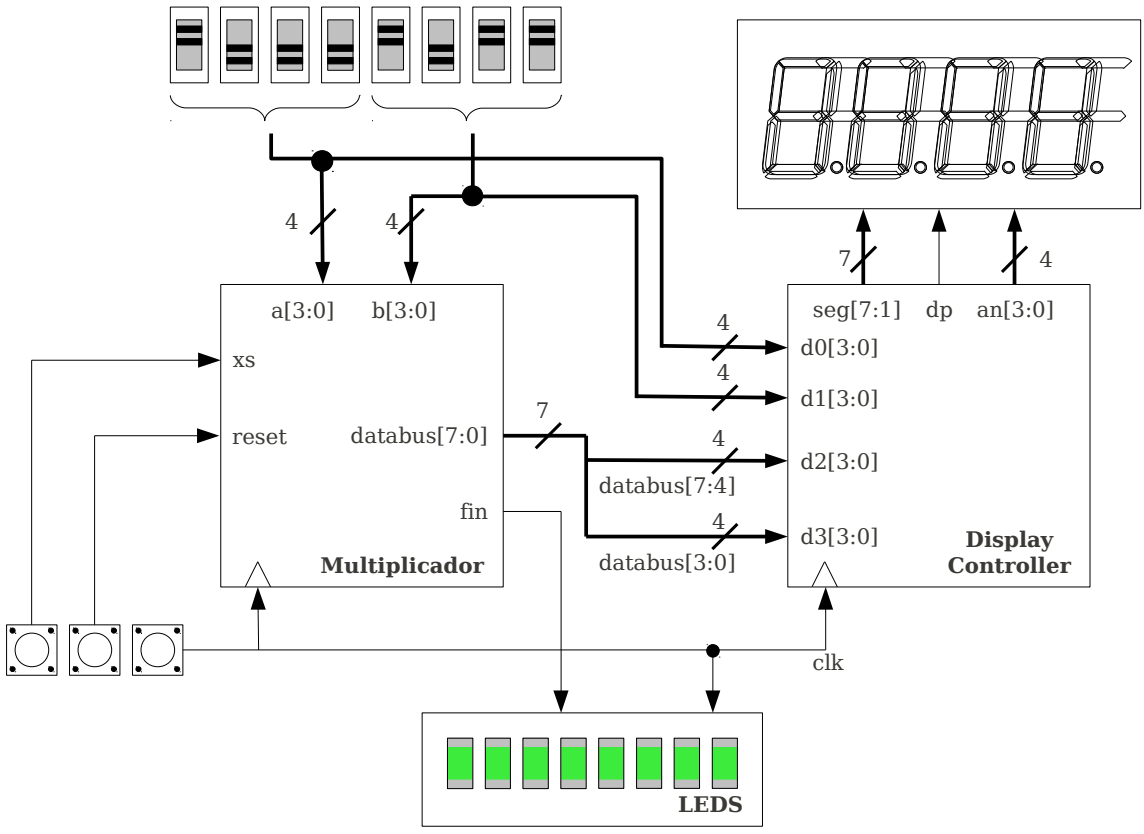


Figura 6. Esquema del sistema digital completo para el multiplicador.

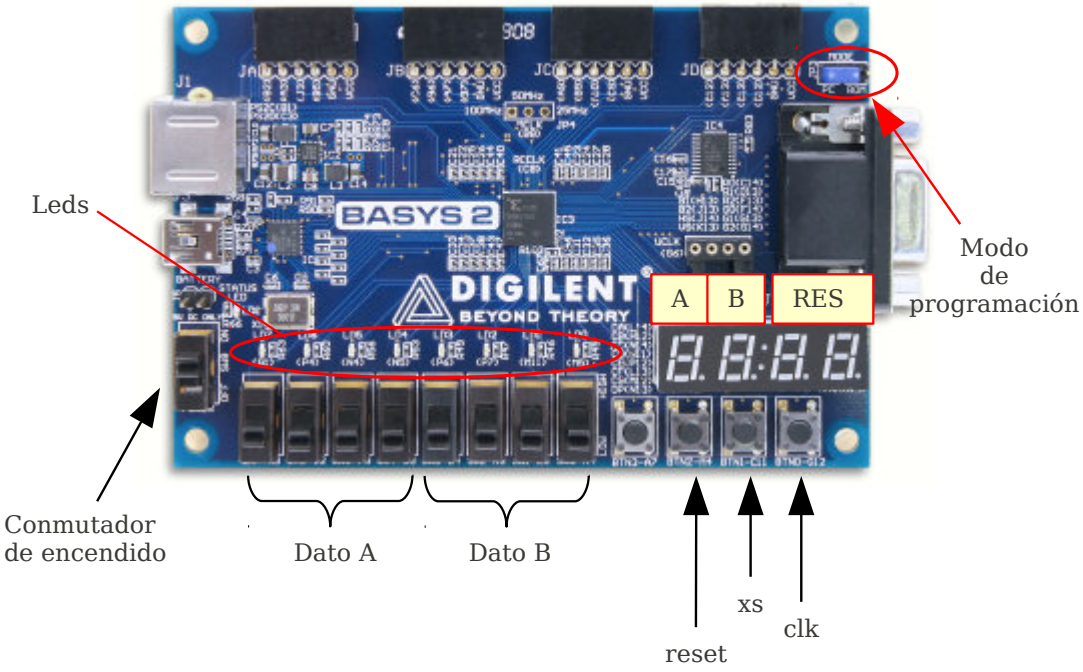


Figura 7. Fotografía de la placa de desarrollo Basys2

La figura 7 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.

4. Estudio Experimental

En la sección anterior se ha descrito el multiplicador como Sistema Digital. Éste consta dos partes: Unidad de Datos (fichero *u_datos.v*) y Unidad de Control (fichero *u_control.v*). La primera se proporcionan 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.

4.1. 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 8). 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*.

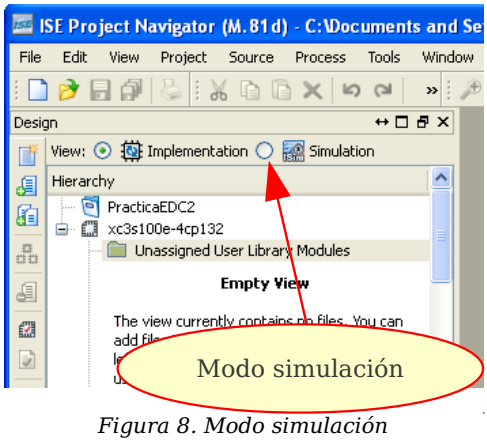


Figura 8. Modo simulación

   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 1. 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*.

```
module multiplicador(
 input  [3:0] a,      // Entrada del dato A
 input  [3:0] b,      // Entrada del dato B
 input  clk,          // Reloj del sistema
 input  xs,           // Señal de comienzo
 input  reset,        // Reset del sistema
 output fin,          // Señal de fin de operacion
 output [7:0] databus // Bus de datos de resultado (parcial y final)
 );

// Conexiones internas para interconectar los dos módulos
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 mimso
   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 2. 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 9), 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 9).

   4.2. Una vez corregidos los errores, realice la simulación mediante la opción *Simulate Behavioural Model*. Debe comprobar todas la 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 10. Busque las multiplicaciones del estudio teórico y amplíe la forma de onda hasta que se visualicen correctamente las señales.

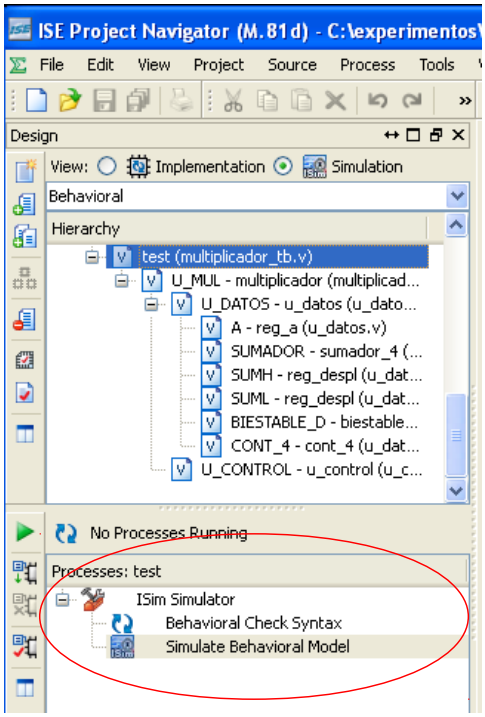


Figura 9. Simulación del tesbench.

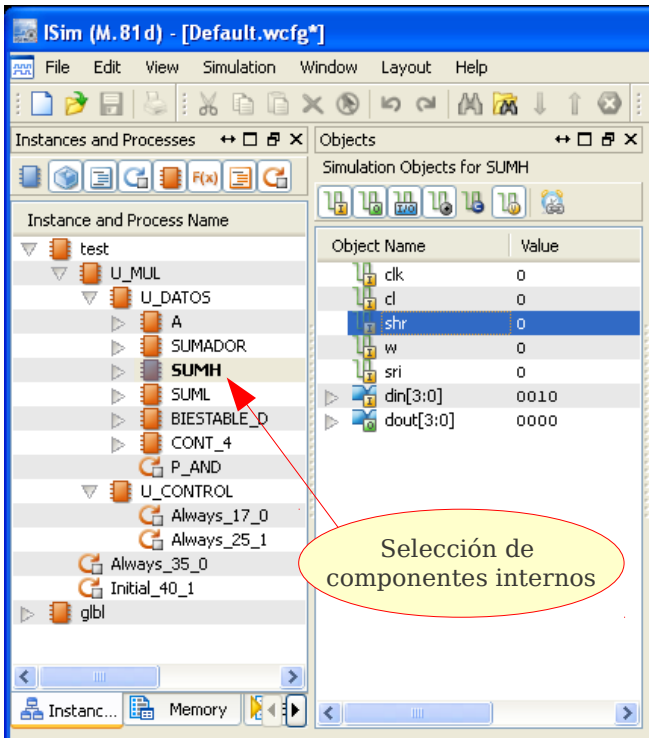


Figura 10. Acceso a módulos internos.

4.2. 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 3.2 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.

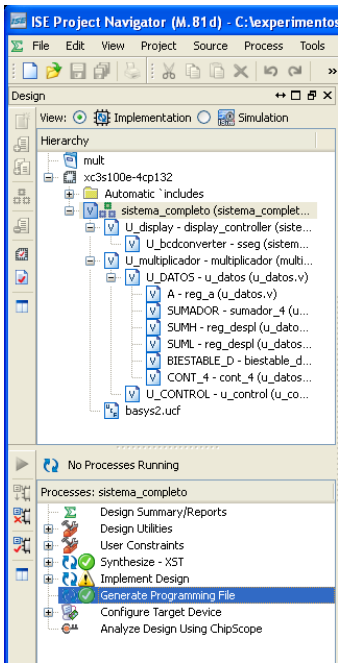


Figura 11. Implementación

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 8
2. Añadir dos ficheros al proyecto:

   2.1. Fichero *sistema_completo.v* como implementación. Este fichero aparecerá ahora como raíz del proyecto.

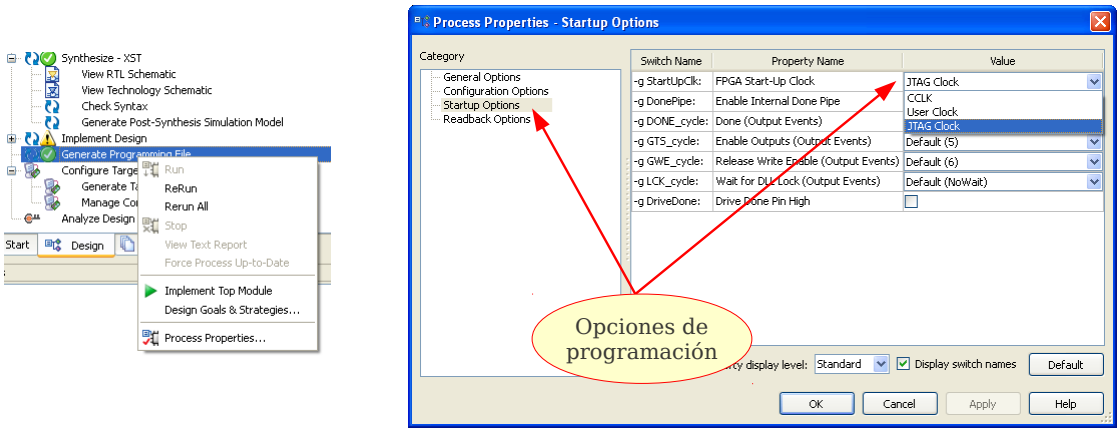
   2.2. Fichero *basys2.ucf*. Debe comprobar que también aparece en el árbol de proyecto.

3. La implementación se realiza seleccionado el módulo *sistema_completo* en el árbol de proyecto. Debe seleccionar la opción Generate Programming File (figura 11) 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 12a, seleccionando la opción de menú de Process Poperties aparecerá el diálogo mostrado en la figura 12b.

   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 11, 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.



(a) *(b)*

Figura 12. 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 7 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 Adept) y aparecerá el programa mostrado en la figura 13. 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.

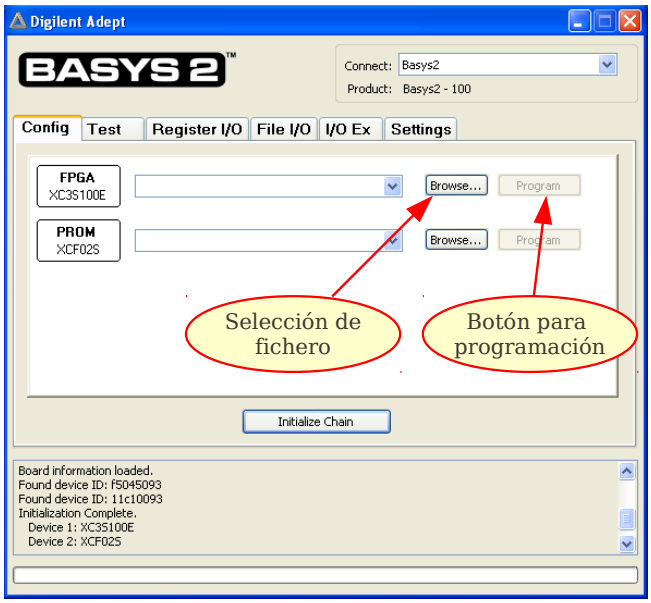


Figura 13. Programación de placas Digilent con Adept.

5. En la figura 7 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.

Producto de 1011 × 0001

Ciclo	Señales de control									estado_actual	Contenido de los registros					micro-operaciones
clk	xs=clsincr	wa	wd	wsuml	wsumh	shrsum	upcont	cycont	fin		[A]	[CONT]	[D]	[SUMH]	[SUML]	
1	1	0	0	0	0	0	0	0	0	S0	-	-	-	-	-	A ← 0, SUMH ← 0, SUML ← 0, CONT ← 0, D ← 0
2																
3																

Producto de 0010 × 1101

Ciclo	Señales de control									estado_actual	Contenido de los registros					micro-operaciones
clk	xs=clsincr	wa	wd	wsuml	wsumh	shrsum	upcont	cycont	fin		[A]	[CONT]	[D]	[SUMH]	[SUML]	