DISEÑO DIGITAL I

Modelado y simulación de un multiplicador parametrizable

Universidad Politécnica de Madrid
ETSI Sistemas de Telecomunicación
Dpto. Ingeniería Telemática y Electrónica

DD1-A4

Introducción

Esta actividad está dividida en dos partes. La primera parte describe la arquitectura de un multiplicador de números naturales. En la segunda parte se proponen diversos ejercicios relacionados con los conocimientos adquiridos en las actividades ya realizadas en el curso.

Para la realización de la actividad, dispone de ficheros adjuntos a este documento.

PARTE I: Arquitectura de un multiplicador

La figura 1 muestra la arquitectura de un circuito multiplicador para números naturales de n bits.

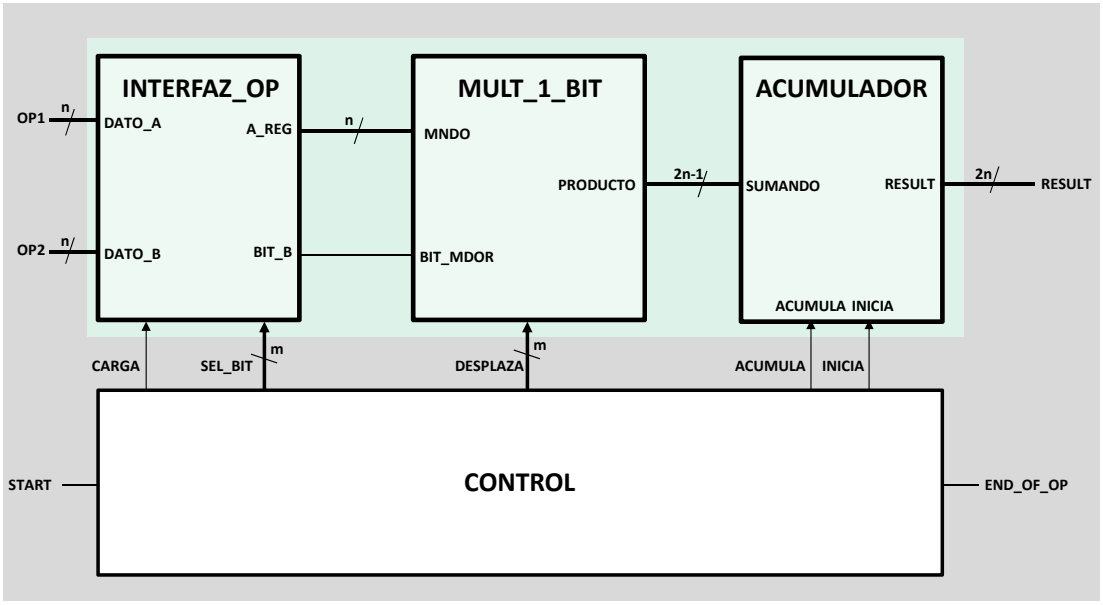


Figura 1.- Arquitectura para el diseño de un multiplicador

El módulo **INTERFAZ_OP** de la figura 1 registra el multiplicador y el multiplicando (**OP1** y **OP2**), cuando se activa la entrada **CARGA**, y entrega por la salida **BIT_B** el bit del multiplicador seleccionado por la entrada **SEL_BIT**. La salida **A_REG** se corresponde con el valor registrado del multiplicando.

El módulo **MULT_1_BIT** realiza el producto aritmético del multiplicando (**MNDO**) y el bit del multiplicador de entrada (**BIT_MDOR**), considerando su peso, codificado por la señal de control **DESPLAZA**, y entrega el resultado de esta operación por la salida **PRODUCTO**.

El módulo **ACUMULADOR** suma los productos generados por **MULT_1_BIT** para obtener el resultado de la multiplicación. Las sumas se realizan cuando la entrada **ACUMULA** está activa. La entrada **INICIA** resetea el registro de acumulación.

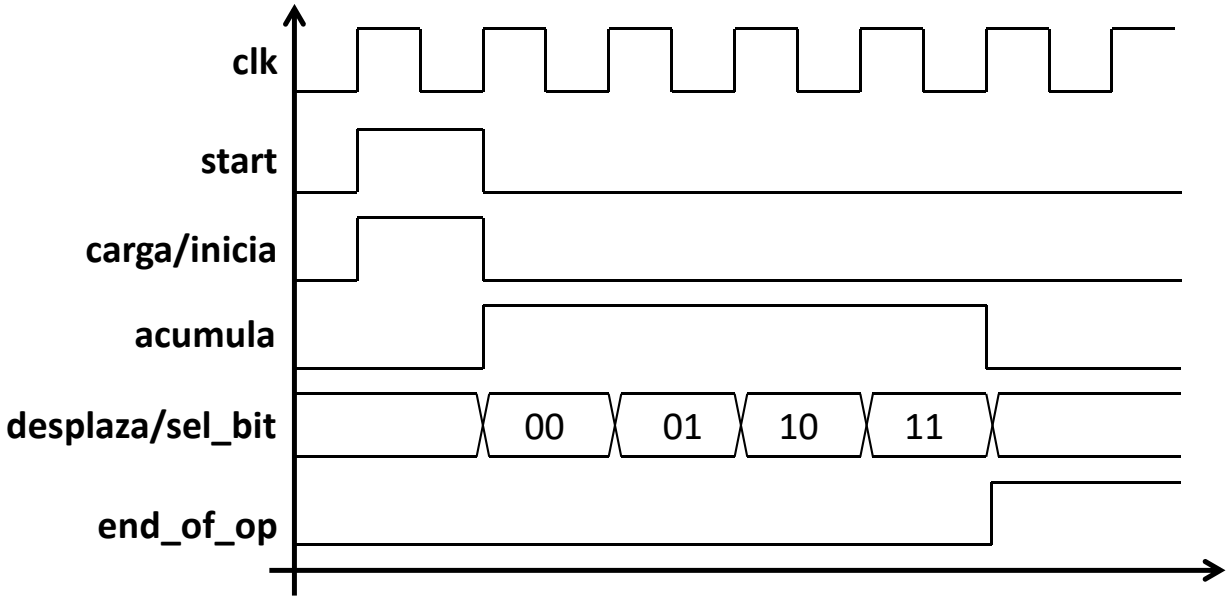
El módulo **CONTROL** genera las señales de control para los tres módulos de la ruta de datos del circuito; por ejemplo, para la realización de un multiplicador de 4 bits:

a) Las salidas **CARGA** e **INICIA** se activan cuando, estando el sistema preparado para realizar una multiplicación, se activa la entrada **START**.

b) La salida **ACUMULA** se activa durante los cuatro ciclos de reloj que siguen a la activación de la entrada **START**. Durante estos mismos ciclos, las salidas **SEL_BIT** y **DESPLAZA** toman, consecutivamente, los valores 0, 1, 2 y 3.

c) La salida **END_OF_OP** se activa una vez transcurridos los cuatro ciclos de reloj en que **ACUMULA** está activa.

El cronograma adjunto muestra la secuencia de señales generada por el módulo **CONTROL**.



PARTE II: Ejercicios

Descomprima, en su directorio de trabajo, el fichero **DD1_A4.zip** disponible en la documentación adjunta.

Arranque Questa y cree el proyecto de trabajo *A4* en el directorio *sim* de la actividad.

Añada al proyecto los ficheros: **mult_8bits.vhd**, **mult_nbits.vhd**, **auxiliar.vhd**, **test_mult_nbits.vhd**.

Ejercicio 1.

En la documentación adjunta dispone del fichero **mult_8bits.vhd** que incluye un modelo VHDL sintetizable de un multiplicador de 8 bits que se ha realizado en base a la arquitectura de la figura 1.

a) La ruta de datos del multiplicador se modela mediante tres procesos y una sentencia concurrente. Identifique qué bloque (o bloques), o la parte de qué bloque (o bloques), modela cada uno de ellos:

- Proceso de la línea 39.
- Sentencia concurrente de la línea 54.
- Proceso de la línea 56.
- Proceso de la línea 67.

b) Identifique las innovaciones añadidas a VHDL-2008 que se emplean en el modelado de la ruta de datos.

c) ¿Le parece correcto el estilo de codificación empleado para modelar el *barrel-shifter*? Analícelo y compárelo con otras alternativas de codificación.

d) El control de la arquitectura del multiplicador ha sido modelado como un autómata, empleando atributos para especificar las transiciones y los valores de las salidas. Dibuje el diagrama de estados del autómata.

e) Considerando las posibles alternativas para la codificación de autómatas que podrían haberse empleado, en lugar de la que se ha elegido, realice un análisis que permita obtener conclusiones sobre las ventajas e inconvenientes de ésta última.

Ejercicio 2.

Partiendo del modelo del multiplicador de 8 bits (**mult_8bits**), realice, en un fichero de nombre **mult_16bits.vhd** (que debe añadir al proyecto creado en esta actividad), el modelo de un multiplicador para números de 16 bits. Compile el modelo realizado.

Ejercicio 3.

El fichero **mult_nbits.vhd** contiene un modelo sintetizable y parametrizable para la realización de multiplicadores con una arquitectura como la de la figura 1.

Nota: El análisis del modelo parametrizable que va a realizar en este ejercicio le resultará más sencillo sabiendo que se han realizado los siguientes cambios respecto al modelo analizado en el ejercicio 1:

- El bloque de control se realiza empleando un contador
- Se ha simplificado el número de señales de control (se han eliminado **alias** y la señal **acumula** se ha sustituido por **not eop**).
- Se emplea el paquete **ieee.std_logic_arith**.

a) La ruta de datos del modelo parametrizable se describe con dos procesos y una sentencia concurrente. Analice la sentencia concurrente para identificar qué elementos de la ruta de datos modela y cómo lo hace.

b) Calcule, para n = 8, la correspondencia entre los estados de cuenta del contador de control y los estados del autómata del modelo **mult_8bits**.

Ejercicio 4.

El fichero **test_mult_nbits.vhd** contiene un *test_bench* que permite verificar el funcionamiento del modelo parametrizable del multiplicador.

a) El código del *test-bench* emplea la función **To_String** del paquete **std_developerskit.std_iopak**. Identifique qué función realiza y analice por qué es necesario utilizarla para la materialización de este *test-bench*.

b) Analice el conjunto de pruebas que se realizan y cómo se verifican los resultados de manera automática. Identifique el procedimiento de verificación automática de resultados.

c) Ejecute una simulación (con n = 4) y compruebe los resultados de la misma.

Ejercicio 5.

En determinados diseños puede resultar complejo formular en un *test-bench* los resultados que debe proporcionar el modelo VHDL, como se ha hecho con la sentencia *assert* en el ejercicio 4.

En estos casos, una alternativa es escribir en un fichero los datos de entrada y los resultados esperados, empleando otras herramientas (Matlab, C, Python…). El *test_bench* lee el fichero, aplica los estímulos a las entradas y verifica, empleando de nuevo sentencias *assert*, que el resultado proporcionado por el modelo coincide con el esperado presente en el fichero.

El fichero **test_mult_nbits_file.vhd** contiene un *test_bench* que emplea esta alternativa para simular el multiplicador parametrizable. Aunque en este caso concreto sería más adecuada la solución mostrada en el ejercicio 4, se aprovecha este mismo ejemplo para ilustrar esta otra alternativa. Incluya este fichero en el proyecto.

a) En la documentación adjunta dispone de un script de Python, **mult_generator.py**, que genera el fichero de estímulos y resultados para operandos de **n** bits. Ejecute el script y observe el fichero que se genera en **sim/estímulos.txt**.

b) Analice el fichero **test_mult_nbits_file.vhd** para comprender su funcionamiento.

c) Ejecute una simulación (con n=4) y compruebe los resultados de la misma. Para ello deberá modificar el parámetro **n** en el script de Python y volver a ejecutarlo, así como el fichero **test_mult_nbits_file.vhd** para establecer el valor de **n**.

d) Introduzca errores en el modelo o en el fichero y observe el resultado de la simulación.