Escuela Técnica Superior de Ingeniería Industrial y de Telecomunicación. Grado en Ingeniería de Tecnologías de Telecomunicación. Curso 2021/22. Electrónica Digital I.

Se valorará la corrección del trabajo, así como la presentación de este. La presentación puede hacerse a mano o por ordenador. Deben entregarse también (via e-mail, por ejemplo) los ficheros de Circuito Maker o Quartus II (todos los ficheros del proyecto) generados al realizar la práctica (por ejemplo, comprimidos en un .zip o en un .rar).

Incluir una referencia al tiempo utilizado para hacer todo el trabajo (resolución y presentación), y el trabajo realizado por cada miembro del grupo indicando el tanto por ciento del trabajo total y las tareas realizadas. Última fecha de entrega: 7-Enero-2022.

Trabajo nº 5:

1º. Diseñar un circuito aritmético combinacional que tenga dos entradas, X e Y, y una salida Z, todas ellas de cuatro bits para números descritos en complemento-2 (valores entre -8 y +7), que realice las tres siguientes operaciones aritméticas:

(VER HOJAS DE PRACTICA POR GRUPO)

La estructura del diseño (con variaciones según las operaciones a realizar) debe estar basada en la estructura de un circuito sumador/restador con selección de los operandos. Se deben utilizar dos entradas de control C1 y C0 para determinar la operación aritmética que se realiza. El diseño debe disponer también de una salida Ov de desbordamiento que indique cuando el resultado de la operación excede del rango válido de las salidas Z. El cálculo del desbordamiento se puede obtener del siguiente razonamiento (ver clases teóricas del tema I): si al hacer una suma los signos de ambos operandos son iguales y el signo del resultado es distinto es que se ha producido desbordamiento.

a) Diseño mediante módulos combinacionales y puertas lógicas.

- Diseñar el circuito tomando como base las operaciones de sumas y restas para números binarios en complemento a 2, usando únicamente los siguientes dispositivos de la familia 74 para realizar el circuito aritmético:

* Un único circuito sumador de cuatro bits 74LS83 para realizar las operaciones de suma (y/o resta).
* Un único circuito con cuatro puertas EXOR 74LS86 para realizar la complementación de un operando al realizar la operación de complemento-2.
* Uno o dos circuitos (según el trabajo, usar uno solo cuando sea posible) con cuatro multiplexores de dos entradas 74LS157 para realizar la selección de los operandos de entrada del sumador. También se puede dejar un operando a 0 usando la entrada de *enable* del multiplexor.

En la mayoría de los casos definiendo adecuadamente el valor lógico de las señales de control C1 y C0 para cada operación aritmética, la lógica de control se podrá implementar sin usar ninguna puerta lógica. Si fuese necesario hay que utilizar el menor número de puertas lógicas de la familia 74LS.

Incluir además los circuitos necesarios para el cálculo del desbordamiento Ov. El circuito que genere la salida OV de desbordamiento tiene que ser el más pequeño posible.

- Simular el circuito introduciendo 20 pares de datos de entrada aleatorios mediante un *Data Sequencer* (*Data Seq.*, *hotkey* G). Las señales de control C1 y C0 se pueden aplicar mediante *switches* (*Logic Switch*, *hotkey* s). Las salidas Z se pueden visualizar mediante un *display* hexadecimal (*Hex Display*, *hotkey* h), y Ov mediante un led (*Logic Display*, *hotkey* 9). Simular el circuito para las tres operaciones aritméticas (seleccionadas mediante los valores lógicos de C1 y C0), comprobando que el circuito opera correctamente, en caso contrario rehacer el diseño del circuito y volver a simular. Indicar en la memoria del trabajo los valores aplicados en las entradas y los resultados obtenidos, comprobando que el circuito opera bien.

- Utilizando las hojas de características de los dispositivos encontrar el tiempo de propagación máximo del diseño.

b) Diseño a partir de una descripción VHDL.

- Realizar una descripción VHDL del problema. Se sugiere que la descripción se adapte a la estructura del circuito desarrollada en el apartado a) y que se tenga en cuenta que las operaciones aritméticas son para números con signo (utilizar el paquete *ieee.std_logic_signed*).

- Crear un proyecto en Quartus II y compilar la descripción del circuito sobre el dispositivo CYCLONE II EPC235F672C6, eliminando los errores que se produzcan. Obtener las características físicas del diseño: número de celdas y tiempo de propagación máximo.

- Simular en Quartus II el circuito utilizando un conjunto de 20 estímulos aleatorios para las entradas X e Y, que varíen más o menos cada 5 veces el tiempo de propagación máximo del circuito: programar adecuadamente el campo *End Time* y el valor del periodo de intervalo fijo de asignación para hacer una asignación aleatoria (icono ) en X e Y. Fijar sucesivamente los valores de las entradas de control a 1 o a 0 según lo necesario para cada operación aritmética y comprobar que el funcionamiento del circuito es correcto para todas las operaciones. Se sugiere que X, Y y Z se muestren en tipo *signed decimal*.

- ***OPCIONAL***. Implementar el circuito sobre la tarjeta DE2. Las entradas deben aplicarse mediante *switches*. Las entradas (X, Y, C1, C0) deben aplicarse mediante *switches*. Modificar la descripción VHDL del circuito de forma que el valor de las entradas y salidas numéricas se visualicen en *displays* de 7 segmentos, usando dos *displays* por dato de entrada (X, Y) y de salida (Z), de forma que uno corresponda al signo del dato (todos los leds apagados para positivos, - para negativos) y el otro al valor absoluto del dato (de 0 a 8 como máximo). La salida de desbordamiento Ov debe aparecer en un led rojo. La implementación debe hacerse mediante una única programación, después de haber simulado concienzudamente el circuito para comprobar que funciona correctamente. La implementación del circuito puede hacerse desde el fichero .sof generado por la compilación en Quartus II.

IMPORTANTE: Para evitar problemas que puedan estropear la tarjeta, al asignar las entradas y salidas del proyecto hay que dejar los pines sin usar del dispositivo como entradas en alta impedancia. Para ello antes de compilar el dispositivo hay que ejecutar el comando *Device* del menú **Assignment**, pulsar en el botón *Device & Pin Options…*, y en la pestaña *Unused Pins* en su campo *Reserve all unused pins:* seleccionar *As input tri-stated*.

2º. Diseñar un circuito realice varias operaciones aritméticas entre dos dígitos X e Y de un código BCD A con peso (valores entre 0 y 9), correspondientes al código BCD A del apartado 1 del trabajo 4, pero suponiendo que tienen añadido un bit de signo (Sx o Sy), de forma que si el bit de signo es 0 el número es positivo, y si el bit el de signo es 1 el número es negativo. Los números así codificados están entre +9 y -9, con dos ceros +0 y -0, pero en este diseño se debe suponer que el -0 no es válido. El resultado Z de la operación aritmética debe obtenerse en complemento-2 en 6 bits.

El circuito debe realizar operaciones aritméticas con los datos (SxX) y (SyY) en función de los valores de dos señales C1 y C0 del circuito. Las operaciones a realizar son:

(VER HOJAS DE PRACTICA POR GRUPO)

que se implementarán con circuitos ALU 74LS18, que realiza operaciones aritméticas para números de 4 bits descritos en complemento-2, y operaciones lógicas para datos de 4 bits. Para realizar el circuito se debe convertir los valores de cada operando a complemento-2 de 5 bits, y luego realizar las operaciones aritméticas usando dos circuitos 74LS181 conectados en cascada para conseguir los 6 bits necesarios para hacer la operación.

Además, si las entradas no son válidas (alguno de los datos no corresponde con alguno de los dígitos del código BCD correspondiente) debe encenderse una señal de error E y fijarse las salidas Z a 0 (el circuito ALU 74LS181 que se debe usar en este trabajo tiene al menos una operación que fija las salidas de datos a 0).

a) Diseño mediante módulos combinacionales y puertas lógicas.

- Realizar el diseño utilizando principalmente módulos combinacionales de la familia 74LS, usando el menor número de puertas lógicas posible.

Como sugerencia el circuito puede realizarse primero mediante una conversión de los dígitos X e Y a complemento-2 de 5 bits mediante un sistema decodificador-codificador formado por dos circuitos 74LS138 (decodificador 3 a 8) o un circuito 74LS154 (decodificador 4 a 16), y circuitos 74LS147 (codificador con prioridad de 10 a 4, aunque también se podría usar un codificador sin prioridad hecho con puertas lógicas). El bit de signo del número en complemento-2 es directamente el bit del signo del operando de entrada.

Convertir los números positivos es relativamente sencillo (ver problema 7_2 del tema IIIb), ya que la salida del codificador 74LS147 genera los datos binarios de 0 a 9 en polaridad negativa, que corresponde al [+0, +9] en complemento-2 en polaridad negativa, que se puede pasar a positiva con puertas NOT. Para convertir los números negativos hay que usar otro codificador 74LS147 con la idea de que su salida, vista en polaridad positiva, genera los datos binarios de 1111 (salida a 0) a 0110 (salida a 9), que se pueden utilizar para codificar de -1 (1 1111 en complemento-2) a -9 (1 0111 en complemento-2); una idea parecida a esta se puede ver en el problema 6_3 del tema IIIb. Por último, en función de que el bit de signo sea positivo o negativo se seleccionan las salidas de un codificador o las del otro (se puede hacer con el multiplexor de dos entradas 74LS157).

La señal de error de cada dígito (Ex o Ey) se puede realizar operando en una OR (hecha con NAND) las salidas del decodificador que no corresponden a la codificación y la combinación que tiene el signo a 1 y el dato a 0 (-0, que no es válida).

El circuito requiere dos copias del convertidor a complemento-2, una para cada dígito X e Y, y se puede generar la salida de error E final como la OR de Ex y Ey. En Circuit Maker es sencillo duplicar un circuito seleccionándolo, y usando el comando *Duplicate* del menú *Edit* (o copiando y pegando).

Con los datos ya convertidos a complemento-2, se realizan las operaciones usando la ALU 74LS181. Las hojas de características del dispositivo describen su funcionamiento en dos casos, según se elija la polaridad (activo alto o activo bajo) de las entradas A, B y de las salidas F del 74S181. Para facilitar la simulación de los circuitos se recomienda considerar la polaridad positiva (activo alto), aunque en el esquema del circuito en Circuit Maker aparece en polaridad negativa (activo bajo). Para realizar el diseño hay realizar una decodificación que convierta los valores de entrada de las señales de control C1C0 y la señal de error E (produce 0 en la salida Z) en los valores adecuados de las entradas S3-S0, M y Cn (acarreo de entrada) de la ALU 74LS181 para que haga la operación pedida. Hay que intentar asociar a las entradas C1C0 y E una codificación de las operaciones que permitan reducir la lógica de decodificación. Como los datos de entrada a la ALU son de 5 bits en complemento-2, y hay operaciones del tipo PLUS y doble de (con incremento en 1 o sin él), o MINUS (con decremento en 1 o sin él), los valores de Z pueden estar entre [-19, +19], lo que indica que el número de bits de la salida Z será de 6. Para obtener 6 bits de salida se deben usar dos ALUs en conexión *ripple* y considerar una extensión de signo de los operandos de entrada de la ALU a 6 bits. La conexión *ripple* se realiza conectando la salida de acarreo de una ALU (C_{n+4}) a la entrada de acarreo de la siguiente ALU (Cn), manteniendo iguales los valores de las entradas de control S y M. Los bits no usados de los operandos de entrada de la ALU más significativa se deben fijar a 0 y los bits de salida no utilizados deben quedar desconectados.

- Simular el circuito con Circuit Maker y comprobar que el circuito opera correctamente. Los módulos combinacionales pueden localizarse en la clase *Digital by Number*. Introducir los datos con ayuda de llaves hexadecimales (*Hex Key*, *hotkey* H) para X e Y, *switches* para Sx y Sy (*hotkey* s), o mediante dos *Data Sequencers* (*hotkey* G), uno para (SxX) y otro para (SxY). Utilizar *switches* para C1 y C0. Leer los datos usando por ejemplo displays hexadecimales (*Hex Display*, *hotkey* h) o displays lógicos (*Logic Diplay*, *hotkey* 9) y un display lógico (*Logic Display*, *hotkey* 9) para E. Indicar en la memoria del trabajo los valores aplicados en las entradas y los resultados obtenidos, comprobando que el circuito opera bien.

- Utilizando las hojas de características de los dispositivos encontrar el tiempo de propagación máximo del diseño.

b) Diseño a partir de una descripción VHDL.

- Realizar una descripción VHDL del problema. Se sugiere que se tenga en cuenta que las operaciones aritméticas son para números con signo (utilizar el paquete *ieee.std_logic_signed*).

Se sugiere editar la descripción de la operación dentro de un *process*. En él, realizar primero la conversión del código BCD con signo de entrada a código en complemento-2 mediante una sentencia **case** (o un **if-else** según el signo y dos **case**, uno para números positivos y otro para negativos). Se puede hacer primero para X (generando la variable XC2, por ejemplo), y luego con *copy-paste* para Y (generando la variable YC2). Además, dentro del **case** se puede controlar la existencia de error sobre una variable de 1 bit (por ejemplo, EI). EI se debe inicializar antes de las sentencias **case** al valor de no hay error (‘0’), y si hay error se le debe asignar ‘1’ (en **when others**). EI se cargará finalmente en la salida E.

Obtenidos XC2 e YC2, en función de EI, C1 y C0 se deben realizan las operaciones aritméticas para generar Z, seleccionándolas con sentencias **case** o sentencias **if-else**, sin olvidar que debe hacerse una extensión de signo en XC2 e YC2 a 6 bits.

- Crear un proyecto en Quartus II y compilar la descripción del circuito sobre el dispositivo CYCLONE II EPC235F672C6, eliminando los errores que se produzcan. Obtener las características físicas del diseño: número de celdas y tiempo de propagación máximo.

- Simular en Quartus II el circuito utilizando al menos 50 combinaciones distintas de valores en las entradas (SxX) e (SyY), que varíen más o menos cada 5 veces el tiempo máximo del circuito: programar adecuadamente *End Time* para hacer una asignación en el rango de X e Y con el icono  . y Comprobar que el funcionamiento del circuito es correcto para cada una de las operaciones (cambiar los valores de C1 y C0 en cada simulación). Se sugiere que C1, C0, Sx y Sy se muestren como un bit y X e Y se muestren en formato hexadecimal. Por último, se sugiere que E se muestre como un bit y que Z se muestre como *signed decimal*. Es posible que al cargar los valores de entrada aleatorios muchos de ellos produzcan error, al ser códigos no válidos en X o Y; en ese caso probar a cambiar los valores de X o de Y usando otra vez, el icono  sobre las entradas

- ***OPCIONAL***. Implementar el circuito sobre la tarjeta DE2. La implementación debe hacerse mediante una única programación, después de haber simulado concienzudamente el circuito para comprobar que funciona correctamente. Las entradas (SxX, SyY, C1, C0) deben aplicarse mediante *switches*. Modificar la descripción VHDL del circuito para que el valor de las entradas y salidas numéricas se visualicen en *displays* de 7 segmentos, usando dos *displays* para las entradas (SxX) y (SyY), de forma que uno corresponda al signo del dato (todos los leds apagados para positivos, - para negativos) y el otro al valor absoluto del dato (de 0 a 9 como máximo). La salida Z debe mostrarse en tres *displays* que correspondan uno al signo del resultado (todos los leds apagados para positivos, - para negativos) y los otros dos muestren el valor absoluto del número (entre 0 y 19) en formato decenas (0 => todos los leds apagados o 1) y unidades (entre 0 y 9). La implementación del circuito puede hacerse desde el fichero .sof generado por la compilación en Quartus II.

IMPORTANTE: Para evitar problemas que puedan estropear la tarjeta, al asignar las entradas y salidas del proyecto hay que dejar los pines sin usar del dispositivo como entradas en alta impedancia. Para ello antes de compilar el dispositivo hay que ejecutar el comando *Device* del menú **Assignment**, pulsar en el botón *Device & Pin Options*…, y en la pestaña *Unused Pins* en su campo *Reserve all unused pins:* seleccionar *As input tri-stated*.