

**Grado en Ingeniería de Tecnologías de Telecomunicación.
Escuela Técnica Superior de Ingeniería Industrial y de Telecomunicación.
Electrónica Digital I.**

Práctica nº 9. Diseño lógico con dispositivos programables.

En esta práctica se realizará el montaje de circuitos combinatoriales implementados mediante la tarjeta DE2. Los circuitos se diseñarán sobre el dispositivo programable de dicha tarjeta y se utilizarán como entradas y salidas los dispositivos de entrada (*switches*, botones, ...) y de salida (*leds*, *displays* de 7 segmentos, ...) de la tarjeta DE2.

Todo el trabajo de la práctica debe realizarse en el directorio (o carpeta) Pr9 (o similar), que debe crearse en el directorio de trabajo de cada alumno, donde deben guardarse los ficheros utilizados o generados durante la práctica, usando subdirectorios para cada apartado.

9.1. Diseño de un circuito convertidor de decimal a binario.

En este apartado se va a realizar el diseño, montaje y verificación de un circuito convertidor de un número de dos dígitos decimales D (décadas) y U (unidades) a un número binario B. D y U serán entradas de cuatro bits cuyo valor debe estar entre 0 y 9. En la tarjeta DE2 las entradas D y U se aplicarán usando 8 *switches*.

El número de entrada está en el rango de [0, 99] por lo que la salida B requerirá de 7 bits, que se observarán en la tarjeta DE2 en 7 *leds* ordenados adecuadamente (*leds* consecutivos donde el bit más significativo aparece a la izquierda y el menos significativo a la derecha). Además, se añadirá una salida de error E que indique si alguna de las entradas D y U toman un valor mayor de 9, con lo que la solución no es válida; en la tarjeta DE2 la salida E se mostrará en un *led* de color distinto al de los *leds* de B. La descripción del circuito puede hacerse mediante sumas sabiendo que:


$$B = 10 * D + U = 8 * D + 2 * D + U,$$

donde $8 * D = \text{“D000”}$ (D desplazado 3 dígitos a la izquierda, $D * 2^3$), y $2 * D = \text{“D0”}$ (D desplazado 1 dígito a la izquierda, $D * 2^1$).


La descripción VHDL puede hacerse mediante sumas de operandos de 7 bits sin signo (utilizar el paquete *ieee.std_logic_unsigned*), extendiendo a 7 bits los operandos que sean necesarios.


El diseño del circuito se hará mediante estos pasos:


- Ejecutar Quartus II. Se recuerda que toda la jerarquía del proyecto debería tener el mismo nombre (carpeta, proyecto, fichero y *entity*). Editar la descripción VHDL (comando *New* del menú **File**, eligiendo el tipo *VHDL File*), crear un proyecto con el dispositivo EP2C35F672C6

(al salvar el fichero VHDL) y compilar la descripción (pulsar en ). Una vez que la compilación no da error, indicar el número de celdas del dispositivo utilizadas al sintetizar el circuito (campo *Total logic elements* en la ventana *Compilation Report*) y el tiempo máximo de propagación (ejecutar el comando *Classic Timing Analyzer Tool* del menú **Processing**, pestaña *tpd*).

- Simular el circuito con un número reducido y aleatorio de muestras (20, por ejemplo) y comprobar que opera correctamente. Para ello hay que crear un fichero de formas de onda (comando *New* del **menú File**, eligiendo el tipo *Vector/Waveform File*).


Utilizar el comando *Insert Node or Bus* de la pestaña *Insert* del menú **Edit** para incluir (pulsar en *Node Finder*, luego pulsar en *List*, y seleccionar pulsando en ) las entradas D, U y las salidas B, E en la ventana de formas de onda, eligiendo inicialmente el formato decimal para D, U y B, y el formato binario para E.

Se pueden poner valores en las entradas fácilmente usando primero el comando *End Time* del menú **Edit** y fijando su valor 2.0 us. Seleccionar sucesivamente cada entrada D y U y pulsar en el icono  para aplicar valores aleatorios: elegir *At fixed intervals* fijando el campo *Interval period* a 100 ns desde teclado. Aparece un valor distinto aleatorio cada 100ns en la entrada seleccionada.

Pulsar en  para realizar la simulación. Comprobar que el circuito simula correctamente.

- Asignar las entradas y salidas a los *switches* y *leds* de la tarjeta DE2 con el comando *Assignment Editor* del menú **Assignments**, y, en la ventana *Category*: pulsar sobre Pin. Los pines asociados a los switches aparecen en la página 28 del fichero pdf *DE2_UserManual*, mientras que los leds aparecen en la página 29.

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*.

- Volver a compilar el circuito ().
- Programar el dispositivo programable desde el programador de Quartus II (comando *Programmer* del menú **Tools**).
- Comprobar que el circuito opera correctamente en la tarjeta DE2. Cambiar el formato de D, U y B a binario en la ventana de resultados de simulación (seleccionar la entrada, pulsar el botón derecho del ratón y seleccionar *Properties*). Introducir con los *switches* los valores que aparecen en algunas de las entradas y comprobar que el resultado correspondiente aparece en los leds.
- Guardar y cerrar el proyecto mediante los comandos *Save Project* y *Close Project* del menú **File**.

9.2. Diseño de un circuito de votación.

En este apartado se va a realizar el diseño, montaje y verificación de un circuito que calcule el resultado de una votación. Aunque la descripción del problema será común, alguna de las condiciones de la votación será diferente para cada grupo de prácticas, lo que se indicará al comenzar la sesión. La descripción común es la siguiente:

- Un grupo de V votantes I_1, I_2, \dots, I_v , realiza una votación en la que puede estar a favor ('1') o en contra ('0') de una moción. No se permiten las abstenciones. El número máximo de votantes es 10.
- Cada votante I_i tiene asignado un peso P_i o voto ponderado en la votación, de forma que además del número N de votos puros a favor, el resultado de la votación puede decidirse por el número de votos ponderados a favor $P = \sum_{i=1}^v P_i * I_i$. El total de los votos ponderados es 30 ($\sum_{i=1}^v P_i = 30$).
- El resultado de la votación siempre debe ser positivo o negativo. No puede haber empates. El resultado de la votación dependerá de N , de P y del voto de I_1 , según unas condiciones que se darán en la sesión de prácticas.
- El resultado de la votación se mostrará mediante varias salidas. Si la moción es aceptada se fijará a 1 la señal LV, y si es rechazada se fijará a 1 la señal LR (siempre habrá una y solo una de las luces a 1). Además, se deben visualizar el número de votos a favor y el número de votos ponderados a favor para los que se necesitarán tres *displays* de 7-segmentos: NV (para el número de votos positivos), PD y PU (decenas y unidades del voto ponderado positivo).


El diseño lógico del circuito podría hacerse de varias maneras, por ejemplo, mediante una tabla de verdad con todas las entradas y salidas. Como esta descripción es difícil de ampliar a más bits, y es difícil de editar en un tiempo limitado, se sugiere usar un sistema basado en una descripción VHDL del circuito mediante un *process* en los que se vayan haciendo en orden sucesivas tareas:

- Calcular el número de votos a favor NUMV, y el número de votos ponderados a favor NUMP. NUMV y NUMP pueden ser variables enteras con rango: NUMV entre 0 y V; NUMP entre 0 y 30. Se pueden inicializar estas variables a 0, e ir incrementándolas para cada votante: si el votante I_i está a favor NUMV se incrementa en 1 y NUMP en P_i ; si el votante I_i está en contra no se incrementa.
- En función de NUMV, NUMP, el voto de I_1 y las especificaciones dadas, calcular las salidas LV y LR.
- Mostrar el resultado de NUMV en un *display* de 7 segmentos mediante una sentencia **case** sobre NUMV, basada la descripción utilizada en el apartado 4 de la práctica 6 (copiar del fichero, pegar y modificar).
- Mostrar el resultado de NUMP en dos *displays* de 7 segmentos. Primero se obtienen las decenas PD. Un método posible es comprobar mediante una sentencia **if-elsif-else** el valor de NUMP, con ayuda de una variable entera NUMU de rango entre 0 y 9 que codificará las unidades:

- Si NUMP es 30 codificar un 3 en el *display* de 7 segmentos PD y poner NUMU := 0.
- Si NUMP está entre 20 y 30, codificar un 2 en el *display* de 7 segmentos PD y asignar a NUMU := NUMP – 20.
- Si NUMP está entre 10 y 20, codificar un 1 en el *display* de 7 segmentos PD y asignar a NUMU := NUMP – 10.
- Si NUMP es menor de 10, dejar apagado el *display* de 7 segmentos PD (mejor que mostrar 0) en el *display* de 7 segmentos PD y asignar NUMU := NUMP.


- Obtener el valor de las unidades PU en un *display* de 7 segmentos mediante una sentencia **case** sobre NUMU, basada la descripción utilizada en el apartado 4 de la práctica 6 (copiar del fichero, pegar y modificar).


El diseño del circuito se hará mediante estos pasos:

- Ejecutar Quartus II. Se recuerda que toda la jerarquía del proyecto debería tener el mismo nombre (carpeta, proyecto, fichero y *entity*). Editar la descripción VHDL (comando *New* del menú **File**, eligiendo el tipo *VHDL File*), crear un proyecto con el dispositivo EP2C35F672C6 (al salvar el fichero VHDL) y compilar la descripción (pulsar en ). Una vez que la compilación no da error, indicar el número de celdas del dispositivo utilizadas al sintetizar el circuito (campo *Total logic elements* en la ventana *Compilation Report*) y el tiempo máximo de propagación (ejecutar el comando *Classic Timing Analyzer Tool* del menú **Processing**, pestaña *tpd*).

- (Opcional) Simular el circuito con un número reducido y aleatorio de muestras (20, por ejemplo) y comprobar que opera correctamente. Para ello hay que crear un fichero de formas de onda (comando *New* del **menú File**, eligiendo el tipo *Vector/Waveform File*).

Utilizar el comando *Insert Node or Bus* de la pestaña *Insert* del menú **Edit** para incluir las entradas I y las salidas LV, LR, PD y PU en la ventana de formas de onda, eligiendo inicialmente el formato binario.

Se pueden poner valores en las entradas fácilmente usando primero el comando *End Time* del menú **Edit** y fijando su valor 2.0 us. Seleccionar sucesivamente cada entrada I y pulsar en el icono  para aplicar valores aleatorios: elegir *At fixed intervals* fijando el campo *Interval period* a 100 ns desde teclado. Aparece un valor distinto aleatorio cada 100ns en la entrada seleccionada.

Pulsar en  para realizar la simulación. Comprobar que el circuito simula correctamente


Al simular las salidas (NV, PD, PU) se muestran en la codificación de *displays* de 7 segmentos, por lo que es difícil comprobar de forma inmediata si los resultados están bien. Para comprobar más fácilmente el circuito se puede realizar una simulación funcional comprobando el valor de las variables internas NUMV y NUMP. Los pasos para realizar esta simulación son:

- En la ventana de formas de onda hay que insertar los nuevos nudos para la simulación (*Edit->Insert->Insert Node or Bus*) y al buscar nudos (*Node Finder*) situar Filter: al valor *Design Entry (all nodes)*. Al pulsar en List aparecen los nudos internos entre los que se pueden seleccionar NUMV y NUMP, fijando su valor al tipo *Unsigned Decimal*.

- Abrir la ventana de simulación mediante el comando Simulator Tool de Processing. Situar el modo de simulación al valor *Functional*. Pulsar en *Generate Funcional Simulation*. Pulsar en *Start* para simular y en *Report* para ver los resultados de la simulación.

- Asignar (usar *Assignment Editor* del menú **Assignments**, y, en la ventana *Category*: pulsar sobre Pin) las entradas I a los *switches*, la salida LV a un *led* verde y LR a un *led* rojo, y las salidas NV, PD, PU a displays de 7 segmentos. Los pines asociados a los *switches* aparecen en la página 28 del fichero pdf *DE2_UserManual*, mientras que los *leds* aparecen en la página 29 y los *displays* en la página 31.

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*.

- Volver a compilar el circuito ().
- Programar el dispositivo programable desde el programador de Quartus II (comando *Programmer* del menú **Tools**).
- Comprobar que el circuito opera correctamente en la tarjeta DE2.
- Guardar y cerrar el proyecto mediante los comandos *Save Project* y *Close Project* del menú **File**.