PRÁCTICAS EN LABORATORIO – MEMORIA

"Diseño digital con esquemáticos y FPGA"

**GRUPO** H

**FECHA** Febrero 2019

**MIEMBRO**

1

ÍNDICE

2. ENCENDER UN LED

3. PUERTAS LOGICAS

4. DECODIFICADOR 4 A 2

5. DECODIFICADOR 3 A 8

6. DECODIFICADOR 4 A 16

7. CONVERTIDOR DE HEXADECIAMAL A 7 SEGMENTOS

8. CODIFICADORES

9. EXTENSIÓN DE LA CAPACIDAD DE UN CODIFICADOR

10. COMPARADORES

11. COMPARADORES EN CASCADA

12. EL MULTIPLEXOR

13. EL SUMADOR

14. SUMADOR / RESTADOR

15. UNIDAD ARITMÉTICO LÓGICA

16. ELEMENTOS DE MEMORIA

17. INTERRUPTOR CON PULSADOR

18. PARPADEO DE UN LED

2

**Para realizar las prácticas debes además seguir este guión y realizar los ejercicios propuestos a continuación. Se deberán entregar al profesor por el campus virtual en forma y fecha estipulada por el profesor. Si necesitas más espacio para los ejercicios al ser un Word sólo tienes que incluir saltos de línea o tabuladores. Respeta el formato de este fichero para la entrega.**

**Para completar los ejercicios debes apoyarte de la bibliografía aportada por el profesorado y el libro de la asignatura colgado en el campus. Debes completar todos los ejercicios para que se califique el bloque.**

**1. INTRODUCCIÓN A LA PLACA Y LA INSTALACIÓN**

Para estas prácticas trabajamos con el kit Basys 2.

Podéis seguir este guión de laboratorio sin problemas, pero debéis prestar atención al pin-out de la placa porque es diferente de la del guión.

El pin-out lo podéis obtener del siguiente enlace: https://www.digilentinc.com/data/products/basys2/basys2\_rm.pdf

Después, para instalar el software como en el laboratorio debéis instalar:

(1) Entorno ISE de Xilinx (versión 9.2 mínimo): http://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/design-tools/archive.ht ml - Nota: para descargarte el xilinx de modo gratuito debes registrarte. Recuerda instalar la 9.2 o la 10.1 para poder seguir el guión del laboratorio.

(2) Software para grabar el programa de la placa Bays 2 , se llama Adept: Para windows, tenéis que pinchar en la sección de Windows del siguiente enlace: https://reference.digilentinc.com/digilent\_adept\_2#software\_downloads ENLACE NUEVO

Nota: debéis descargados los 3 enlaces que vienen e instalar las 3 cosas (sdk, utilities y system). Básicamente, son los 3 enlaces que os dice la web de Adept para windows:

▪ Adept 2.16.1 System, 32/64-bit Windows – Download

▪ Adept 2.2.1 Utilities, 32/64-bit Windows – Download

▪ Adept 2.3.1 SDK, 32/64-bit Windows – Download Después, os descargáis los 3 archivos y ejecutáis el .EXE de System y Utilities. No hay que hacer más, si tenéis cualquier problema llevar vuestro portátil al laboratorio de clase.

(3) Para grabar la placa, a diferencia de lo que pone en el guión debéis de no abrir el IMPACT, si no el ADEPT. Pasos: - Sintetizar el proyecto para generar el archivo .bit (ver guión). - Para subirlo a la nueva placa debes ir al menú Adept. Para ello, debes Pulsar Inicio --> Windows --> Digelent --> Tools --> Adept - Conectar la placa Basys 2 (la primera vez se instalará el driver). - Aparecerá en el cuadro de texto de la derecha arriba el nombre de la placa. - Después en el primer campo de texto, pulsamos a la derecha en browser y seleccionamos el archivo .bit generado por el ISE una vez que hayamos sintetizado el proyecto correctamente.

Con este guión y el libro de las prácticas de la asignatura podréis seguir el laboratorio.

3

**2. ENCENDER UN LED**

Para este primer hito, cuyo objetivo es entrar en contacto con el software Xilinx y comenzar con el diseño digital a través del uso de esquemáticos, estableceremos como primer objetivo encender uno de los leds de la placa. Si la observamos con atención, vemos que la placa Spartan 3E tiene 8 Leds disponibles para utilizar, cada uno de ellos conectados a distintos pines de la FPGA. Para la mayor sencillez, encenderemos el Led0. En este primer caso, el esquemático a utilizar resulta muy sencillo, dado que únicamente conectaremos alimentación a la salida ld0 para que se encienda el Led. Incluimos una etiqueta ld1 conectada a tierra de manera que el Led1 nunca se encenderá. Una vez hecho, procedemos a compilar el resultado y, dado que no hay errores, asignamos los pines de la siguiente manera: [Design / User Constraints / Assign Package Pins] ld0 M5 ld1 M11 De esta manera, al programar nuestro diseño en la FPGA el Led0 recibe un1 lógico y se enciende; mientras que el Led1, al estar conectado a tierra, se queda apagado.

**EJERCICIO 1:**

**- Incluye a continuación la Tabla de la Verdad. - Demostrarle al profesor que funciona.**

**LDO LD1**

1 0

**3. PUERTAS LÓGICAS, INTERRUPTORES Y PULSADORES NOTA: SOFTWARE 14**

● **Para la asignación de los pines es en la pestaña Design -> User constrains -> IO Pin planning, se os abrirá el programa Ahead. Dentro de él, esperar la primera vez porque tarda mucho en abrirse, tenéis que buscar en la parte inferior donde pone "Ports". Ahí, tenéis que desplegar para que os aparezcan los pines de vuestro proyecto. Pues simplemente, en la ventana "site", nada más, ponéis el número de vuestro PIN.** 4

En este apartado, se busca introducir en nuestras fuentes tipo esquemático la utilización de puertas lógicas, utilizando los botones e interruptores (switches) y leds de nuestra placa como entradas y salidas respectivamente. Para ello, utilizaremos puertas AND y OR, además de una puerta inversora NOT y un buffer, que electrónicamente nos retrasará un poco la señal, pero que a efectos prácticos nos sirve para poder conectar un puerto de entrada que ya tiene una salida a otra salida distinta. El resultado del esquemático se muestra a la derecha. Como se puede observar en nuestro circuito lógico, este diseño tiene el siguiente funcionamiento: Para el caso de la puerta AND, que realiza la operación lógica multiplicación (S=A**·**B), podemos ver que su salida LD0 únicamente será un 1 lógico en el caso de que sus dos entradas (SW0 y SW1) tengan el valor 1. Esto ocurre cuando se encienden los switches 0 y 1 a la vez, lo que encenderá el Led0 de nuestra placa. La puerta OR realiza la suma lógica (S=A+B), de manera que la salida LD1 sólo recibirá un 0 en el caso en que ambos switches permanezcan en off, y por tanto se activará tanto si se enciende el switch 2, como el switch 3 o los dos a la vez. En el último caso, siempre tendremos una salida activada. El Botón0 siempre estará enviando un 0 lógico hasta que se pulse en la FPGA. Si no se pulsa, el Led2 permanece encendido y el Led3 apagado. Cuando se pulse el botón sucederá lo contrario. OBSERVAMOS TODOS LOS COMPORTAMIENTOS DESCRITOS EN LA SIGUIENTE SIMULACIÓN:

**Fig. 3**

Podemos comprobar de esta manera que nuestro diseño funciona correctamente. Asignamos los pines correspondientes y procedemos a la implementación en la placa a través de Adept.

Incluye aquí el video demostrativo utilizando una herramienta como Youtube o vimeo, aquí tiense un ejemplo de compañeros del año pasado: https://www.youtube.com/watch?v=67FsF\_OM8h0

**EJERCICIO 2: Incluye aquí el enlace de tu video demostrativo (no debe ser superior a 50 MB):**

**VISTO Y VERIFICADO POR LA PROFESORA EN EL AULA DE PRÁCTICAS.**

5

**4. DECODIFICADOR 2 A 4.**

Para este apartado, empezaremos ya a aplicar la funcionalidad del software para crear y adaptar bloques combinacionales, en este caso un decodificador de 2 a 4, para formar futuros diseños más complejos. Un decodificador de 2 a 4 recibe dos entradas que están codificadas en binario y activa la salida que se corresponde con el número binario introducido (si se introduce 10 se activa la salida 2 del decodificador). El diseño con esquemáticos quedaría de la siguiente manera:

El decodificador tiene el siguiente funcionamiento:

El diseño consta de dos entradas, por las cuales se introducen dos cifras binarias que conforman un número del 0 al 3. Estas entradas son A0 y A1 siendo el bit 0 y el bit 1 del número binario respectivamente. Por ello, obtenemos 4 salidas (S0, S1, S2 y S3) que se activan dependiendo del número que haya en las entradas. Además, disponemos de una entrada ENABLE (E) que hace la función de señal de habilitación que, en caso de no estar activa, todas las salidas estarán también inactivas independientemente del valor de las entradas.

**EJERCICIO 3 A REALIZAR: Rellena la siguiente Tabla de Verdad para el decodificador de 2 a 4:**

**E A1 A0 S3 S2 S1 S0** 0 X X 0 0 0 0 1 0 0 0 0 0 1 1 0 1 0 0 1 0 1 1 0 0 1 0 0 1 1 1 1 0 0 0

6

Si lo tienes bien, que las salidas son minitérminos de las entradas siempre y cuando la señal E este activada. Es por eso que implementamos el circuito con puertas AND, introduciendo en las entradas de estas las señales A0 y A1 como corresponda (EJ: S0 = A1 **·** A0 ; S1 = A1 **·** A0 ). La segunda línea de puertas AND se incluye para lograr que si la señal de habilitación se corresponde con un 0 lógico, ninguna de las salidas se active, dado que las entradas quedan multiplicadas por 0. Pasamos ahora a realizar la simulación, con la que veremos que cada una de las combinaciones posibles de entradas activa una sola salida con la señal ENABLE activa. También comprobaremos tanto la funcionalidad de deshabilitar el circuito poniendo un 0 en E como que todo funciona correctamente. Así queda la simulación:

**Fig. 5**

**EJERCICIO 4 A REALIZAR: Incluye en el siguiente hueco la simulación de tu circuito pero que el orden de las variables que figuran en la parte izquierda sea E, S0, S1, S2, S3, A0, A1**

Por último asignamos los pines, de manera que las entradas A0 y A1 se corresponden con los switches 0 y 1, y de la misma manera las salidas Si se asocian a los leds (esto es: S1 al Led1, de forma análoga asignamos el resto). En este caso la señal E la asignaremos al switch7. De esta manera, con las distintas combinaciones de los 2 interruptores encenderemos uno de los 4 leds. El símbolo quedará del siguiente modo. Sigue las pautas del documento extenso para guardar el símbolo. Comprueba que te queda como el siguiente símbolo.

7

Fig. 6: *Símbolo del decodificador 2 a 4*

8

**5. DECODIFICADOR 3 A 8.**

Hemos realizado el decodificador de 2 a 4, utilizando dos interruptores encendíamos uno de los cuatro primeros leds. Ahora vamos a construir un decodificador que encienda uno de los 8 leds que tiene nuestra placa, utilizaremos para ello los tres primeros interruptores. Para comenzar con el siguiente decodificar que se plantea en este ejercicio, necesitamos añadir a nuestro proyecto el bloque creado en el apartado anterior, dado que realizaremos el diseño de este decodificador de 3 a 8 a partir del decodificador de 2 a 4 de manera que nos quedará un diseño mucho más simple que si eligiésemos realizarlo a través de puertas lógicas. Por ello, lo primero es *Project / Add Copy of Source...* para añadir el bloque que se muestra en la última imagen del apartado anterior. Tendremos que volver a utilizar el Wizard para darle forma.

Una vez hecho esto, si observamos la tabla de verdad verás que faltan valores:

**EJERCICIO 5 A REALIZAR: completa los valores de la tabla de verdad teniendo en cuenta que como ya habíamos hecho en el hito anterior, que si E = 0, independientemente de lo que valgan las entradas las salidas están inactivas.**

**Por otro lado, podemos dividir la tabla en dos, siempre y cuando E = 1:**

**● Si A2 = 0, vemos como en ese trozo de la tabla S0, S1, S2 y S3 se comportan igual que el decodificador 2 a 4. Por su parte de S4 a S7 permanecen inactivas.**

**● Si A2 = 1, son ahora S4, S5, S6 y S7 las que se van activando, mientras que las anteriores no se activan. También se comportan como un decodificador 2 a 4.**

**E A2 A1 A0 S7 S6 S5 S4 S3 S2 S1 S0** 0 X X X 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 1 0 1 1 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 1 1 0 1 0 0 1 0 0 0 0 0 1 1 1 0 0 1 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0

9

Observamos

Visto esto, podemos pasar al diseño del esquemático del decodificador 3 a 8, sabiendo ya que necesitaremos 2 decodificadores 2 a 4 debidamente conectados. Este es el resultado:

Para el decodificador situado más arriba, como habíamos deducido en la tabla de verdad, metemos como señal de activación E **·** A2.

Para el de abajo, la señal de habilitación valdrá E **·** A2.

Introducimos las señales A0 y A1 como entradas de ambos decodificadores y el primero nos dará las salidas de S0 a S3 y el segundo de S4 a S7.

**EJERCICIO 6 A REALIZAR: Incluye una imagen de cómo ha quedado el esquemático del presente bloque con el mismo formato que el de la figura** Fig. 1**.**

10

**EJERCICIO 8 A REALIZAR: Incluye en el siguiente hueco la simulación incluyendo las señales de entrada y salida teniendo en cuenta que debes meter los mismos valores de la siguiente figura:**

El último paso será la asignación de pines para poder comprobar el funcionamiento en la FPGA. Para ello, sabemos que las entradas A0, A1 y A2 se corresponden con los switches del 0 al 2 respectivamente (SW0, SW1 y SW2), y que la señal E la asociaremos a SW7. Las salidas en esta ocasión serán los 8 leds, de manera que los asignamos en su orden natural (S0 con LED0, S1 con LED1, S2 con LED2,...). Utilizando los switches como un número en binario puro, obtendremos encendido a la salida un led cuyo identificador se corresponde con dicho número binario. Compruébalo.

**EJERCICIO 8.2: Incluye aquí el enlace de tu video demostrativo (no debe ser superior a 50 MB):**

**VISTO Y VERIFICADO POR LA PROFESORA EN EL AULA DE PRÁCTICAS.**

11

**6. DECODIFICADOR 4 A 16**

Este caso es muy similar al anterior, ya que se nos pide hacer un decodificador de 4 a 16 implementándolo a través de los decodificadores 2 a 4 del apartado 4. Este nuevo decodificador de 4 a 16 activa una sola de las 16 líneas de salida de acuerdo con el código binario de las 4 entradas. Las salidas son mutuamente exclusivas ya que solamente una de las salidas es igual a 1 en cualquier momento, como ya veíamos en los dos casos anteriores.

**EJERCICIO 9 A REALIZAR: Rellena la tabla de la Verdad para un decodificador de 4 a 16**

**E A3 A2 A1 A0 S15 S14 S13 S12 S11 S10 S9 S8 S7 S6 S5 S4 S3 S2 S1 S0** 0 x x x x 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

En la tabla de verdad deberías comprobar que en el caso donde E='0', se desactiva todas las salidas, con independencia de las entradas. Deberías ver que de nuevo la tabla para E='1' se puede dividir en 4 grupos de forma análoga al apartado anterior, y que para cada grupo se puede utilizar un decodificador de 2 a 4 como el que ya hemos diseñado.

12

Por lo tanto, a la hora de realizar la implementación de la tabla de la verdad, todos los decodificadores tendrán las mismas entradas A1 y A0. La única diferencia está en las salidas a las que se conecta cada decodificador y la habilitación. Por ejemplo, el primer decodificador tendrá como salidas S0, S1, S2 y S3 y sólo estará habilitado cuando A3='0' y A2='0'; el segundo decodificador estará habilitado cuando A3='0' y A2='1'. Y de manera similar para el tercer y cuarto decodificador. Por ello, lo único necesario es crear la lógica para habilitar cada decodificador: Si nos fijamos en como las señales A3 y A2 habilitan a los distintos decodificadores, vemos que se comportan también como un decodificador 2 a 4, esto es: A3 **·** A2 habilita el decodificador1, A3 **·** A2 habilita el decodificador 2, A3 **·** A2 habilita el decodificador3 y A3 **·** A2 habilita el decodificador4. Por todo ello, utilizaremos también un quinto decodificador 2 a 4 para gestionar las señales de ENABLE.

**EJERCICIO 10 A REALIZAR: Incluye la imagen del esquemático que has creado teniendo en cuenta lo explicado anteriormente.**

13

**EJERCICIO 11 A REALIZAR: Incluye la simulación.** *Nota: Dado que en la placa sólo tenemos 8 leds disponibles para usar, y sin embargo nuestro decodificador tiene 16 salidas, no se puede probar directamente. Lo que sí puedes hacer es una simulación para comprobar si el esquemático es correcto. Mostramos en la siguiente imagen la parte final de la simulación (no nos cabía todo en la imagen porque no podíamos hacer más Zoom Out en la ventana de simulación):*

Una vez que hayas comprobado que la simulación es correcta, crea el símbolo como la siguiente figura:

Fig. 7: *Símbolo del decodificador 4 a 16*

14

**7. CONVERTIDOR DE HEXADECIMAL A 7 SEGMENTOS.**

Para este nuevo proyecto, nos proponemos mostrar un número hexadecimal (4 bits) en un visualizador o display de siete segmentos. Dado que es un número codificado en hexadecimal utilizando 4 bits (desde 0 hasta 15), podemos tener cifras desde el 0 hasta la F. Como puedes ver en la documentación de la placa, nuestro display consta de 4 partes o 4 sub-displays, de manera que se pueden tener a la vez 4 cifras representadas. Si sólo se quiere activar un display (por simplicidad usaremos el de la derecha) debemos poner la señal AN\_0='0'. Si se quisieran encender los demás displays para mostrar también cifras en ellos, deberíamos poner su correspondiente señal AN\_X a 0, si no lo dejamos a 1, como haremos nosotros en esta ocasión.

Con todo ello, trataremos de mostrar en un display el número hexadecimal codificado en los cuatro primeros interruptores (SW0, SW1, SW2 y SW3). Las salidas, por tanto, serán los segmentos que se quieran encender. Otras salidas adicionales son estos 'habilitadores' de los displays de los que se ha hablado anteriormente (AN\_X). Como siempre, también dispondremos de una señal de Enable (E) que permitirá habilitar/deshabilitar el circuito.

A la hora de diseñar y realizar este diseño, se han de entender varias cosas:

- Los segmentos, al igual que las señales AN\_X, se activarán si reciben un '0' lógico, y por tanto

permanecerán apagados mientras reciban un '1'. - Para implementar nuestro diseño, utilizaremos el decodificador de 4 a 16 creado en el apartado anterior, dado que tenemos esa posibilidad, pero también podría realizarse a través de puertas lógicas. - El mismo segmento se puede encender para varios números distintos.

Lo primero que debes realizar es completar la siguiente tabla de la Verdad, para comenzar a idear el esquemático.

15

**EJERCICIO 12 A REALIZAR: Rellena la siguiente tabla de la Verdad**

**E I3 I2 I1 I0 A B C D E F G** 0 X X X X 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 1 1 1 1 1 0 0 1 0 0 0 1 0 0 1 0 1 0 0 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 1 1 0 0 1 0 1 0 1 0 1 0 0 1 0 0 1 0 1 1 0 0 1 0 0 0 0 0 1 0 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 1 0 0 1 1 0 1 0 0 0 0 1 0 0 0 1 1 0 1 1 1 1 0 0 0 0 0 1 1 1 0 0 0 1 1 0 0 0 1 1 1 1 0 1 1 0 0 0 0 1 0 1 1 1 1 0 0 1 1 0 0 0 0 1 1 1 1 1 0 1 1 1 0 0 0

De la tabla deberías deducir que:

Como viene siendo habitual en los últimos proyectos, disponemos de la señal E, con la que independientemente de las entradas, todos los segmentos permanecen apagados (reciben un '1' lógico).

Con E='1', en la tabla de verdad se le dará un 0 a aquellos segmentos que conformen el número (o letra) que tenemos representado en las entradas I0 a I3 en binario.

Así, vemos que si juntamos los minitérminos para sacar la función de cada segmento en su primera forma canónica (1FC) tendremos precisamente las ocasiones en las que dicho segmento no debe encenderse. Por ello, deberías utilizar puertas OR en las distintas salidas del decodificador, para realizar los minitérminos. En el caso de que se tenga que encender un segmento dado, la suma lógica que se lleva a cabo en la puerta OR dará '0' y por lo tanto el segmento se encenderá. De esta manera habrá que crear la lógica a la salida del decodificador para formar el minitérmino (mx) de cada segmento. Esto queda del siguiente modo:

16

**EJERCICIO 13 OPCIONAL A REALIZAR: Incluye la versión del esquemático con puertas AND.**

Sería como este, pero con puertas AND.

Nota: mx como se ha dicho representa los minitérminos que llevan a la salida correspondiente (segmento). Para no tener un lío de cables excesivo, renombraremos cada cable con el minitérmino al que esté representando. Como podemos ver, tanto AN\_1, AN\_2 y AN\_3 como SEG\_DT se conectan a Vcc para que estén permanentemente apagados. El decodificador actúa de la misma manera que en el apartado anterior.

17

**EJERCICIO 14 A REALIZAR: Incluye la simulación. Explica la simulación en un párrafo de 5 líneas máximo.**

**(Aquí tu simulación)**

**La simulación anterior es un convertidor hexadecimal a 7 segmentos. Mostrará los números del 0 al 9, y las letras A-F respectivamente en un display 7 segmentos de acuerdo al valor que tome según la tabla de verdad realizada en el ejercicio 12. Por ejemplo, si el número que queremos mostrar es el 3 solamente conectaremos los segmentos a, b, c, d y g.**

**EJERCICIO 15 A REALIZAR: Prueba en la placa para realizar el video** *Nota:* Como siempre, procedemos a la asignación de pines, PERO IMPORTANTE esta vez a través del fichero .ucf (*"Edit Constraints (Text)")* porque tenemos más volumen de pines, mirando en esta ocasión los que están en la parte inferior y superior del display. Asignamos la entrada E al SW7 y las entradas I0, I1, I2 e I3 a los switches SW0, SW1, SW2 y SW3 respectivamente.

**Incluye aquí el enlace de tu video demostrativo (no debe ser superior a 50 MB):**

**VISTO Y VERIFICADO POR LA PROFESORA EN EL AULA DE PRÁCTICAS.**

18

**8.CODIFICADOR 4 A 2.**

Pasamos a realizar otro de los bloques combinacionales que vimos en clase, esta vez se trata de un codificador 4 a 2 con prioridad. Este tipo de codificadores permiten que varias entradas de las 4 disponibles estén activadas simultáneamente, y que se siga cumpliendo un comportamiento lógico. Además, incluiremos una señal visual (A) que nos aporte información acerca de si se está utilizando alguna de las entradas (si alguno de los botones ha sido pulsado).

**EJERCICIO 16 A REALIZAR: Rellena la siguiente tabla de la Verdad**

**EI I3 I2 I1 I0 S1 S0 A E0** 0 X X X X 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 1 0 1 0 0 1 X 0 1 1 0 1 0 1 X X 1 0 1 0 1 1 X X X 1 1 1 0

Como ya sabemos, si la entrada de Enable (EI) tiene un '0' lógico, las salidas son 0 independientemente de las entradas. De resto, el codificador muestra a la salida encendidos los leds que representarán el número de botón que se pulse a la entrada, es decir, si se pulsa el botón 3 estarán encendidos los dos leds, si se pulsa el botón 2 sólo se encenderá el led 2.

Para determinar las prioridades de nuestro codificador, lo que debemos hacer es incluir puertas AND, cuyas entradas sean las señales con más prioridad y la señal en cuestión. Pondremos la prioridad más alta para el bit más significativo, de modo que si I2 e I1 estuviesen pulsados, el codificador mostrase un 10 y no un 11. De esta manera, para la entrada I0 (BTN0), pondremos una puerta AND cuyas entradas sean I0, I1, I2 e I3, de manera que si hay un '1' lógico en alguna de las superiores multiplicamos por 0 en la puerta lógica. De forma análoga se obtienen las demás puertas AND. Adjuntamos la tabla de verdad con la prioridad incluida: describir.

Utilizaremos los leds 0 y 1 como salidas S0 y S1 (las que se encienden dependiendo de los botones que pulsemos). La señal que nos indica si se ha pulsado algún botón (A), se corresponde con el led6, que como ya habíamos dicho se trata de una señal visual. Además, los codificadores con habilitación incluyen una nueva señal de salida (EO: enable out), que nos permitirá conectar varios codificadores y extender su capacidad. Esta señal nos indica si el codificador está habilitado pero no hay ninguna señal de entrada activa, nos diferencia de las ocasiones en que nuestro codificador no se encuentra habilitado. Con todas estas especificaciones:

19

**EJERCICIO 17 A REALIZAR: Incluye la imagen del esquemático que has creado teniendo en cuenta lo explicado anteriormente.**

Por último se conecta como ejercicio adicional el codificador que acabamos de diseñar con el conversor a siete segmentos que hemos realizado anteriormente. Así, además de ver el número resultante codificado con los leds, lo veremos en el display de siete segmentos. Para ello, primero creamos el símbolo del codificador y añadimos una copia de la fuente del conversor. Simplemente tendremos que conectar la salida A del codificador a la entrada E del decodificador, ya que dicha salida es la que nos dice si se está codificando. Utilizamos como entradas del conversor S0 y S1 y dejamos las dos entradas restantes inutilizadas, conectando ambas a tierra. Con esta descripción:

20

**EJERCICIO 18 A REALIZAR: Incluye la imagen del segundo esquemático que has creado teniendo en cuenta lo explicado anteriormente.**

Repetimos los pasos finales de todos los proyectos: guardamos y chequeamos para detectar posibles fallos. Si no los hay, procedemos a introducir los pines correspondientes en el fichero .ucf, y por último pasamos a programar nuestro diseño en la FPGA.

21

**EJERCICIO 20 A REALIZAR: Incluye aquí el enlace de tu video demostrativo (no debe ser superior a 50 MB):**

**VISTO Y VERIFICADO POR LA PROFESORA EN EL AULA DE PRÁCTICAS.**

22

**EJERCICIO 19 A REALIZAR: Incluye la simulación**

**9.EXTENSIÓN DE LA CAPACIDAD DE UN CODIFICADOR.**

En ocasiones ocurre que necesitamos ampliar la capacidad de nuestro codificador y resulta que no tenemos acceso a uno de las dimensiones requeridas. Por ello, en este apartado vamos a realizar un codificador de 8 a 3 a partir de dos codificadores 4 a 2 como los que se hizo anteriormente.

En la extensión (codif8a3), vemos como será necesario poner el switch 0 (SW0) en ON para que el sistema pueda funcionar (señal ENABLE). En nuestro diseño, el codificador 4 a 2 situado en la parte superior es el que se encargará de los bits menos significativos, quedando así los bits más significativos (los de la izquierda) para el codificador que está debajo. También se podría haber diseñado a la inversa. Las salidas S0 y S1 del codificador de 8 a 3 se forman con la OR de las S0 y S1 de los codificadores de 4 a 2, respectivamente. Para obtener la tercera salida, señal a la que llamaremos S2, basta con tomar la señal A (activo) del codificador de los bits más significativos. De esta manera, la función de este nuevo codificador ampliado es la de coger el número de entrada que se este pulsando y codificarlo, es decir, pasarlo a binario, de manera que se tendrá a la salida encendidos aquellos leds que formen dicho número en binario. La A del codificador de 8 a 3 se obtiene realizando la suma lógica (OR) de las señales de Activo de los codificadores de 4 a 2, ya que estará activo si cualquiera de ellos está activo.

23

**EJERCICIO 21 A REALIZAR: Incluye la tabla de la Verdad basándote en el esquemático de la figura anterior.**

**E1 I7 I6 I5 I4 I3 I2 I1 I0 S2 S1 S0 A E0**

0 X X X X X X X X 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 1

1 0 0 0 0 0 0 0 1 0 0 0 1 0

1 0 0 0 0 0 0 1 X 0 0 1 1 0

1 0 0 0 0 0 1 X X 0 1 0 1 0

1 0 0 0 0 1 X X X 0 1 1 1 0

1 0 0 0 1 X X X X 1 0 0 1 0

1 0 0 1 X X X X X 1 0 1 1 0

1 0 1 X X X X X X 1 1 0 1 0

1 1 X X X X X X X 1 1 1 1 0

Como se podrás observar, debes mantener tanto la señal A como la señal E= descritas en el apartado en el que se realizó el codificador de 4 a 2. Además, mantenemos también las prioridades de las señales de entrada, dándole más prioridad a los valores más altos, siendo la de menor prioridad I0, que sólo entrará en juego si es la única señal de entrada en ON, siempre y cuando EI='1'.

De la misma manera que en el hito anterior, añadiremos el convertidor a siete segmentos para tener una salida más visual. Habrá por ello que copiar las fuentes y crear un símbolo para éste y para el codificador de 8 a 3. Como única diferencia tenemos que ahora la entrada I2 del convertidor a siete segmentos si se usa y no se conecta a tierra, sino que se corresponde con la salida S2 del codificador de 8 a 3. Pues bien, realiza el siguiente ejercicio antes de continuar.

24

**EJERCICIO 22 A REALIZAR: Incluye la imagen del esquemático que has creado teniendo en cuenta lo explicado anteriormente.**

**EJERCICIO 23 A REALIZAR: Incluye la simulación. Explica la simulación en un párrafo de 5 líneas máximo.**

**En la anterior simulación tenemos un codificador de 8 a 3 con prioridad, el codificador sólo permitirá que una de las entradas tome como estado lógico 1, y si hay más de una entrada con estado lógico 1 simultáneamente el codificador seleccionará la entrada de mayor prioridad.**

25

**EJERCICIO 24 A REALIZAR: Incluye aquí el enlace de tu video demostrativo (no debe ser superior a 50 MB):**

**VISTO Y VERIFICADO POR LA PROFESORA EN EL AULA DE PRÁCTICAS.**

**10. COMPARADORES**

Siguiendo con los bloques vistos en clase, en esta ocasión diseñaremos un comparador de 2 bits. La función básica de un circuito comparador consiste en comparar las magnitudes de dos cantidades binarias para determinar la relación entre ellas ( >, < ó = ).

**A B IGUAL AM BM** 0 0 1 0 0 0 1 0 0 1 1 0 0 1 0 1 1 1 0 0 Para un comparador tan simple ( de sólo 2 bits) lo único que necesitamos es observar la tabla de verdad para obtener las funciones lógicas de cada salida, e implementarlas con puertas. Siendo esta tabla de verdad, hemos denominado AM a la salida que se produce cuando A>B y, por consiguiente, BM a la salida que avisa de que B>A.

En esta ocasión, utilizaremos como entradas A y B los switches 0 y 1 (SW0, SW1) y como salidas el led0 para A>B, el led1 para A=B y el led2 para B>A.

Pasamos ahora al diseño de un comparador más complejo, que compare dos señales de dos bits. Se ha de tener en cuenta esta vez el bit más significativo de cada señal a la hora de realizar los cálculos y el diseño con esquemáticos. Creando una nueva tabla de verdad de forma análoga a la anterior, sacamos las siguientes conclusiones:

✔ A es igual a B cuando: A1=B1 y A0=B0 ✔ A es mayor que B cuando A1>B1 o cuando (A1=B1) y A0>B0 ✔ A es menor que B cuando A1<B1 o cuando (A1=B1) y A0<B0

26

Sabiendo esto, ya podemos implementar nuestro circuito...

Aunque también se puede hacer de igual manera que el anterior, utilizando las simplificaciones de Karnaugh al tener 4 entradas, nosotros hemos elegido hacerlo a través del comparador de 1 bit. Vemos que con las conclusiones anteriores resulta muy sencillo de hacer, y hay menos lio de cables y puertas. Únicamente quedaría asignar los pines antes de probar el diseño en la placa. Lo hacemos de la siguiente manera: Vídeo demostrativo de cómo debería funcionar en:

https://www.youtube.com/watch?v=PYjYWKjmn\_g

**11. COMPARADORES EN CASCADA.**

El mecanismo para hacer un comparador de mayor capacidad (más bits) es el mismo que el que se ha seguido en el punto anterior a la hora de crear el comparador de señales de 2 bits. Por ejemplo, si a partir del comparador de dos bits quisiésemos realizar un comparador de 4 bits, los conectaríamos de manera similar a como se hizo. Una vez se tenga esto, también se podría realizar un comparador de 8 bits. Dado que la lógica que hay que poner para conectar unos comparadores con otros es un poco compleja, nos disponemos a añadir algunas entradas de manera que nos sea más sencillo conectarlo todo.

Como se puede ver, el esquemático anterior se ha diseñado para poder ampliar nuestro comparador de 2 bits para tener otras 3 nuevas entradas (IGUAL\_LSB, AM\_LSB Y BM\_LSB), las cuales se corresponden con la comparación resultante de los bits menos significativos [LSB = *Least significant bit*]. Guardamos este diseño, dado que es el que utilizaremos para conectar en cascada y obtener un comparador de 4 bits.

27

**EJERCICIO 25 A REALIZAR: Incluye la imagen del esquemático que debes crear para obtener el comparador de 4 bits:**

***Nota: Debes tener en cuenta que*** *estas nuevas entradas que debes añadir se corresponden con las salidas del primer comparador, por lo que podemos deducir que el de la izquierda será el comparador de los bits menos significativos, por el que meteríamos en paralelo los dos bits de menos importancia de cada señal [A(0) y A(1) ; B(0) y B(1)], lo cual nos dará una medida comparativa en forma de '0' ó '1' lógico en las 3 salidas que se utilizarán en el comparador de bits más significativos (MSB) donde se tendrán también como entradas los dos bits situados más a la izquierda de cada señal, también introducidos en paralelo. Las entradas AM\_LSB y BM\_LSB del primer comparador valen '0', mientras que IGUAL\_LSB vale '1', dado que al no haber más bits antes los vamos a considerar iguales para que no influyan en la comparación posterior. Opcional: Podrías utilizar buses en los cables para simular que hay 4 cables con sólo una línea (3:0), de manera que obtenemos un diseño más limpio, menos enrevesado y más fácil de entender. Los buses nos permitirán escoger el bit que queramos de entre los 4 cables que representa la línea.*

Una vez obtenido, guardamos el diseño, comprobamos el esquemático (Tools Check Schematic) y la jerarquía (Tools Hierarchical Check). Si todo es correcto, pasamos a organizar los puertos del diseño con los pines de la FPGA para poder programarlo: el bus A será el primero compuesto por los 4 primero Switches (SW0, SW1, SW2 Y SW3) y el bus B será el segundo compuesto por los cuatro Switches restantes

28

(SW4, SW5, SW6 Y SW7), además para mostrar cuál de los dos números es el mayor usaremos 3 leds (LD0, LD1 y LD2).

**EJERCICIO 26 A REALIZAR: Incluye aquí el enlace de tu video demostrativo (no debe ser superior a 50 MB):**

**VISTO Y VERIFICADO POR LA PROFESORA EN EL AULA DE PRÁCTICAS.**

29

**12. MULTIPLEXOR**

El bloque que nos disponemos a realizar permite dirigir la información binaria procedente de diferentes fuentes a una única línea de salida para ser transmitida, a través de ella, a un destino común. Aquí te dejamos un Vídeo demostrativo : https://www.youtube.com/watch?v=ENKEVEdF8DY

Para ampliar la funcionalidad del bloque anterior, en este caso trataremos de que, al compararse dos número binarios A y B, no sólo se enciendan los leds indicando cual de los dos es mayor, sino que también aparezcan dichos números en los displays, distribuidos de la siguiente forma:

En caso de que sea A el número mayor, lo mostrará por el display de la derecha (AN\_0), en caso de que A sea menor lo mostrará por el display de la izquierda (AN\_3), y en caso de que sean iguales se mostrará por los dos displays anteriores.

**TABLA DE VERDAD**

**SEL Z**

0 D0

1 D1

30

Como podemos observar en el siguiente esquemático, para este caso utilizaremos multiplexores de los que ya vienen prediseñados en Xilinx, conectándolos adecuadamente entre sí para lograr nuestro multiplexor de señales de 4 bits. Cada uno de los 4 recibe uno de los bits de las señales de datos (D0 y D1), y dependiendo de la señal de selección que se introduzca (SEL), tendremos una salida u otra (A ó B).

Una vez realizado este multiplexor, al que llamamos *mux* simplemente nos queda añadir el convertidor a 7 segmentos y un comparador, puedes escoger el compm4 de xilinx. Sólo restaría unir todos los bloques para tener un diseño con la funcionalidad que se explicó al principio del apartado para tener el esquemático final.

**EJERCICIO 27 A REALIZAR: Incluye la imagen del esquemático final**

**EJERCICIO 28 A REALIZAR: Describe brevemente qué ocurre cuando la señal A > B, A < B**

*Vemos que la señal de selección se corresponde con la señal de salida GT del comparador, que no es más que "A>B". Cuando esto se cumple, se escoge la señal D1 del multiplexor, que se corresponde con los 4 bits de A. cuando no se cumpla, tendremos un 0 como señal SEL y se escogerá la entrada 0 (los 4 bits de B). Si son iguales, nos da igual cual escoger, dado que las dos entradas contienen el mismo número. Sin embargo, en el encendido de los displays si que lo tenemos que tener en cuenta, dado que se mostraría por los dos. Una vez hecho el checkeado del esquemático y el check jerárquico*

31

Por último, tenemos que realizar el último paso que es bajarlo a placa, para ello, como sabes debes asignar las señales correspondientes a los puertos de la placa. Simplemente nos queda decir que los bits de A y B se corresponden con los switches de 1 al 7.

**EJERCICIO 30 A REALIZAR: Pega aquí el contenido del fichero ucf**

**EJERCICIO 29 A REALIZAR: Incluye la simulación de 2 números (los que quieras)**

32

**13. EL SUMADOR**

Siguiendo con la temática de los bloques combinacionales, realizaremos en esta ocasión un sumador de dos números de 4 bits. El circuito sumará dos números sin signo de 4 bits (A y B) que estarán codificados por medio de los interruptores de la placa. Se mostrará en el display de la derecha el resultado de la suma y en caso de que haya desbordamiento encenderá el LED situado más a la izquierda (LD7), además encenderá el punto decimal del display cuando haya desbordamiento. Por último, el sumador recibirá el acarreo de entrada por medio del pulsador de la derecha (BTN0).

Para ello, empezaremos realizando el diseño de un sumador de 1 bit, para luego ir juntando estos sumadores y lograr uno de mayor capacidad. Creamos en primer lugar el bloque 'ha' (semisumador de un), que nos da la suma y el acarreo, y crearemos su símbolo para poder realizar el 'fa' (full-adder o sumador completo) del cual también crearemos símbolo, y que será el bloque del que partiremos para hacer nuestro sumador de 4 bits. Debes seguir el mismo esquemático que las siguientes figuras.

Como se puede ver en los símbolos, la diferencia entre estos dos (ha y fa) es si se tiene o no acarreo de entrada.

**Fig. 10**

**Fig. 11**

33

Ahora se procede a la creación del sumador de 4 bits. Una vez que tenemos un sumador completo de 1 bit, la realización de un sumador de 4 bits es bastante fácil. Simplemente tenemos que conectar los acarreos consecutivamente del bit menos significativo al más significativo; y conectar las entradas de A y B en paralelo, en sus sumadores correspondientes. **EJERCICIO 31 A REALIZAR: Pega la imagen del diseño del esquemático del sumador de 4 bits a continuación**

Ahora vamos a implementar el circuito completo, incluyendo la parte de visualización. Para esto tendréis que añadir al proyecto el convertidor a siete segmentos que hemos realizado en el apartado 7. Para este diseño, tendremos que eliminar del convertidor varias cosas:

-Se ha de quitar la etiqueta y circuitería referente a SEG\_DT -Se quitará también las señales AN\_0, AN\_1, AN\_2 y AN\_3 de ese diseño.

34

**EJERCICIO 32 A REALIZAR: Pega la imagen del diseño del esquemático del sumador de 4 bits conectado al display de 7 segmentos**

Este sería el resultado final, Simplemente restaría la parte de asignación de pines, síntesis e implementación y programar el esquemático realizado en la placa. Sabemos para ello que los 4 bits de las entradas A y B se corresponden de nuevo con los switches del 0 al 7, de la misma manera que anteriormente. Mostraremos el acarreo de salida en el led 7 (LD7) e introduciremos el correspondiente acarreo de entrada pulsando el primer botón de la FPGA (BTN0). La salida la obtendremos tanto a través de los leds como en el display de 7 segmentos.

35

**EJERCICIO 33 A REALIZAR: Incluye la simulación de dos sumas (4 + 5, 4+8), puedes incluir otras adicionales. Explica la simulación en un párrafo de 5 líneas máximo.**

**Hemos incluido la simulación de la suma de 1+3 esta simulación la vemos al principio de 0 ns a 300 ns aproximadamente. También podemos ver la suma de 1 + 1 y por último, a partir de 630ns aproximadamente vemos la suma de 3+1. BTN0 es un botón que cuando está a cero no lleva acarreo y no suma nada, cuando es btn0, suma uno a nuestra suma ,es decir, la primera en lugar de su resultado ser 4 será 5. En resumen nos incrementa uno porque conlleva acarreo.**

36

**14. SUMADOR / RESTADOR**

Para este bloque, queremos añadir además de la suma la funcionalidad de poder realizar una resta, escogiendo la opción deseada a través de una señal de entrada. Usando el complemento a 2 (Ca2) podemos realizar una resta sumando, lo cual nos sirve para poder aprovechar nuestro diseño del sumador de 4 bits realizado anteriormente.

El circuito debe sumar o restar dos números codificados en complemento a 2 con 4 bits (OPA y OPB) y cuyos valores estarán determinados por la posición de los interruptores de la placa. El circuito mostrará en el display de la derecha el resultado de la suma o la resta.

El resultado de la operación (suma o resta) se mostrará por el primer display de 7 segmentos, que añadiremos al esquemático a continuación. Para mostrar el signo negativo se utilizará el punto decimal, es decir, si el punto decimal está luciendo significa que el resultado es el número mostrado pero negativo. En caso de que haya desbordamiento el display mostrará la letra E. Además los cuatro LED de la derecha (de LD0 a LD3) mostrarán el resultado directo (en complemento a 2). Utilizaremos el LD7 para el acarreo de salida y LD6 para el desbordamiento en la resta. El pulsador BTN0 se usará para indicar que se realiza la suma o la resta. Cuando BTN0 esté pulsado se mostrará la suma, cuando no esté pulsado se mostrará la resta. El desbordamiento, señal OV, vemos que se obtiene cuando las entradas tienen el mismo signo, pero se tiene un resultado con el signo cambiado; a esto se deben las puertas AND con algunas entradas negadas.

Una vez hecho esto, será necesario aplicar el complemento a 2 del número B en caso de que se quiera realizar una resta. Para ello, creamos el bloque Ca2: Como ya sabemos, simplemente invertimos todos los bits de OPB (con lo que obtenemos el Ca1) y utilizamos un sumador para sumar 1 al número y tener nuestro resultado en complemento a 2. Añadimos este bloque al diseño.

37

Este es el circuito (al que hemos llamado visualiza) que nos permitirá ver el resultado correctamente en los displays, y tal y como se especificó al principio. Si observamos el primer multiplexor, vemos que recibe el resultado de la operación, y también este mismo resultado pero una vez calculado su Ca2. La entrada de selección es el bit de signo del resultado, ya que sabemos que si el número es positivo, este bit será un 0, y si es negativo tendremos un 1 y habrá o no que realizar la conversión a complemento a 2. En el segundo multiplexor recibimos el resultado ya correcto, o bien el número 14 codificado en binario, dado que la salida de este multiplexor será la entrada del convertidor a 7 segmentos. La entrada de selección es OV (desbordamiento)y, como se dijo antes, si lo hay queremos que nuestro display muestre una E (no 14 en binario puro). Ya tenemos el bloque que nos permitirá reconvertir el resultado si es necesario, y mostrarlo a través de los segmentos de la placa. Así queda el esquemático final:

Como siempre, asignamos pines (puertos de la FPGA). Para ello, ponemos los 4 bits de A y B en los interruptores (switches del 0 al 7), teniendo en cuenta que los primeros bits de cada número (SW3 y SW7) son de signo! La disposición de leds ya se explicó en la introducción, así que sólo queda probar el diseño en la placa.

38

**EJERCICIO 34 A REALIZAR: Incluye la simulación de dos sumas (4 + 5, 4+8), y dos restas (5 – 3, 10 – 3), puedes incluir otras adicionales con acarreo. Explica la simulación en un párrafo de 5 líneas máximo.**

**Hemos realizado varias operaciones distintas a las indicadas. Cuando res\_sum no está activado se realizará la suma, y cuando res\_sum está activo, es decir, a 1 se realizará la resta. En el primer caso vemos la suma de 0+7, en el segundo caso vemos la resta de 13-7 y en el último caso vemos una resta con resultado negativo, ya que es 0-7.**

39

**15. UNIDAD ARITMÉTICO LÓGICA. HASTA AQUÍ ES EL BLOQUE OBLIGATORIO DE LA ASIGNATURA, ESTE BLOQUE 15 TAMBIÉN SE DEBE REALIZAR PARA APROBAR EL LABORATORIO JUNTO CON LOS EJERCICIOS TANTO DE LOS BLOQUES ANTERIORES COMO DEL BLOQUE 15.**

Vamos a realizar un bloque ALU (Unidad Aritmético Lógica), con ella lo que conseguiremos será poder realizar 4 operaciones distintas con dos números de 4 bits (suma, multiplicación por 2, mayor y la operación lógica AND). La ALU suele constar de unos bloques que realizan las operaciones aritméticas y lógicas, y de un multiplexor que escoge las salidas de esos bloques según la operación seleccionada. Lo primero que haremos será realizar cada uno de los bloques que contendrán nuestras operaciones.

15.1 Multiplicador x2:

Tenemos que multiplicar el operando OPA por 2. Simplemente hay que desplazar los números hacia la izquierda y en el bit menos significativo se pone un cero. Si hubiese un uno en el bit más significativo habría que indicar que ha habido desbordamiento.

15.2 Circuito comparador, "mayor": Simplemente usamos un comparador con la salida A>B como entrada de selección de un multiplexor, para en caso de que valga '1' escoger A y en caso contrario B:

Operación lógica AND: Simplemente hay que realizar la operación lógica AND, entre los operadores. Se realiza bit a bit, esto es, R(0)=OPA(0) AND OPB(0); R(1)=OPA(1) AND OPB(1); y de forma análoga obtenemos los demás.

40

15.3 Mutiplexor de 4 entradas: Ahora queremos realizar un multiplexor de cuatro entradas de datos y por tanto, dos de selección. Cada dato tiene cuatro bits. Además, incluiremos una señal de habilitación para que no saque ningún dato en el caso de que no se haya seleccionado ninguna operación. Para ello, utilizamos dos multiplexores de 4 bits y dos entradas de los que hicimos en apartados anteriores.

**EJERCICIO 35 A REALIZAR: Pega el diseño del esquemático a continuación:**

41

15.4 Codificador de 4 a 2 con prioridad:

Por último, haremos un codificador de 4 entradas con prioridad, para el caso en el que se pulsen dos botones a la vez. En estas ocasiones, en nuestro diseño prevalecerá el botón pulsado que este situado más a la derecha en nuestra placa. El diseño es igual al codificador 4 a 2 de 2 bits que ya hicimos en su momento, pero en esta ocasión tenemos que añadir la entrada de habilitación.

**EJERCICIO 36 A REALIZAR: Pega el diseño del esquemático a continuación:**

Llegados a este punto, ya tenemos todos los componentes necesarios para implementar la ALU, así que vamos a crear el diseño que los integra. Sólo tenemos que juntar todos los símbolos en torno al multiplexor, y utilizar las salidas del codificador con prioridad como entradas de selección del mismo.

42

**EJERCICIO 37 A REALIZAR: Pega el diseño del esquemático final a continuación:**

Es hora de implementar. Los operandos vendrán dados por los interruptores. El operando OPA se codificará mediante los 4 interruptores de la derecha (de SW0 a SW3), y el operando OPB con los cuatro interruptores de la izquierda (de SW4 a SW7). Para realizar una operación se deben de pulsar uno de los 4 pulsadores:

x BTN0: realiza la suma S=OPA+OPB x BTN1: multiplica por 2 el operando OPA. S=2xOPA x BTN2: devuelve el número mayor entre OPA y OPB x BTN3: aplica la función lógica AND entre los bits de OPA y OPB El resultado de la operación se mostrará por los cuatro LED de la derecha (de LD0 a LD3), además se mostrará por el display de la derecha (AN\_0). Según qué operación se esté realizando LD7 mostrará:

x En la suma: el acarreo de salida x En la multiplicación: el desbordamiento x En la operación "mayor que": se mantendrá apagado x En la operación lógica AND, se mantendrá apagado Cuando no haya ningún botón pulsado, todos los LED permanecerán apagados, y también el display. Con esta información, asignamos los pines, sintetizamos y podemos programar nuestro diseño.

Ahora debes bajar tu diseño a placa pero antes:

43

**EJERCICIO 38 A REALIZAR: Incluye la simulación para los siguientes casos, explicando para cada caso brevemente lo que ocurre en la simulación, intenta poner colores o formas sobre la simulación utilizando las herramientas del Word o power point para que resulte más sencillo entenderlo:**

**- 4 de sumas (elige los números que quiere, al menos 1 con desbordamiento). - 4 de multiplicación (elige los números que quiere, al menos 1 con desbordamiento). - 4 comparaciones. - 4 de AND.**

**NOTA: También debes incluir un video, el video debe ser de cada uno de los casos que se piden en este ejercicio.**

**VISTO Y VERIFICADO POR LA PROFESORA EN EL AULA DE PRÁCTICAS.**

**Podemos ver en la simulación que cada rectángulo es una operación. En primer lugar, realizamos las cuatro sumas mediante BTN0 ya que lo tenemos activado (está a 1), y eso indica que se realizan la sumas. Estas cuatro sumas empiezan en 0ns y terminan en los 220ns aproximadamente. Posteriormente observamos que BTN1 se activa (es 1), y BTN0 deja de ser 1. Por lo que se realizaría otra operación, la multiplicación y así sucesivamente hasta completar los 1000ns de la simulación y realizar las 16 operaciones (suma, multiplicación, comparación y AND).**

44

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////// O P C I O N A L ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

**Incluye la descripción, tablas de Verdad, simulación y video si lo consideras oportuno de los siguientes bloques opcionales. Puedes hacer todos, o casi todos.**

**El bloque que permite sacar hasta un 9 será el del reloj digital.**

**Como bloque adicional y opcional se propone realizar “El coche fantástico” con los leds de la placa y conectando leds adicionales, si añades sonido podrás llegar al 10. Deberás entregar el borrador del diseño al profesor 2 semanas antes de terminar el laboratorio.**

**16. ELEMENTOS DE MEMORIA**

**17. INTERRUPTOR CON PULSADOR.**

**18. SEGUNDERO (PARPADEO DE UN LED)**

45