Práctica 7

Decodificador rotatorio (FSM, registros y contadores)

En la práctica 7 realizaremos un diseño que combinará el uso de una máquina de estados con registros y contadores. La práctica se plantea para ser implementada en la placa Basys3. Si las condiciones extraordinarias de confinamiento en que se desarrolla nos permiten acceder al laboratorio antes de final de curso, haréis una demo de su funcionamiento. En cualquier caso, debéis superar las fases de simulación y síntesis.

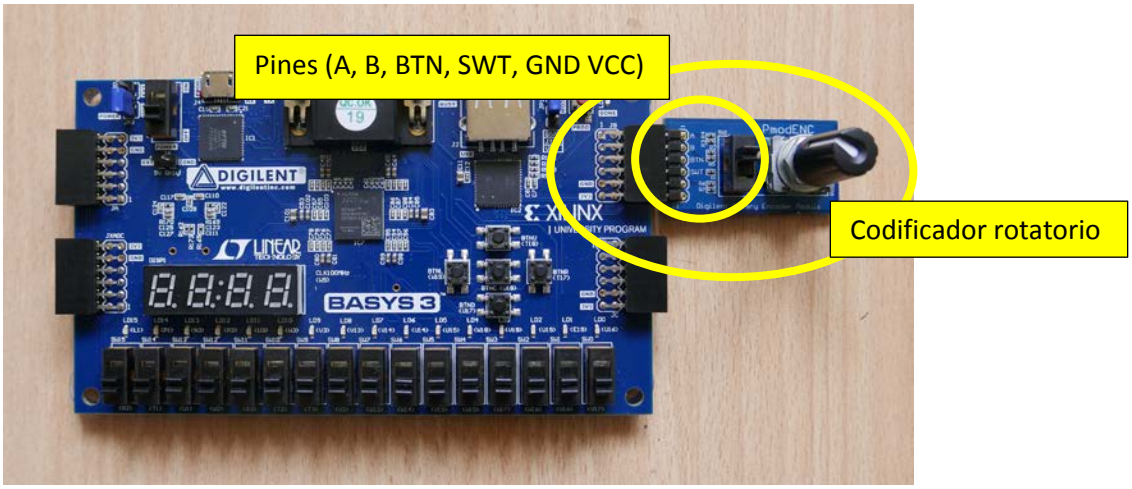
Dado que no tenéis acceso a la placa, la buena calidad de la simulación se hace más necesaria. Esta es la situación a la que se enfrentan los diseñadores, por ejemplo, cuando la tecnología destino es de tipo ASIC. No hay dispositivo físico sobre el que hacer pruebas. Las simulaciones deben ser lo más exhaustivas posible.

En Moodle tenéis un video con la práctica finalizada y funcionando (o en este enlace: https://unican-my.sharepoint.com/:v:/g/personal/fernandv_unican_es/EbziFunfU4BLiiv6LiDabh0BrAhb9ow9kJQ7vywrfm1Ehg?e=VzHRJI).

Contenido

Codificador rotatorio

En la siguiente imagen podéis ver el codificador rotatorio montado en la placa Basys3. Se ha montado en el conector PMOD JB (en la fila inferior del conector pues tiene dos filas de 6 pines).



La pequeña placa Pmod tiene los siguientes pines (su ubicación se distingue también en la imagen anterior):

Pin	Signal	Description
1	A	Output of button A in the encoder shaft
2	B	Output of button B in the encoder shaft
3	BTN	Output of the integral push button in the encoder shaft
4	SWT	Output of the on board switch
5	GND	Power Supply Ground
6	VCC	Positive Power Supply (3.3/5V)

La salida BTN se pone a '1' cuando el mando del rotatorio es apretado como un botón. Adicionalmente, la placa Pmod dispone de un Switch cuyo valor puede leerse en la salida SWT.

Hay varios tipos de codificadores rotatorios. El que se usará en el laboratorio se basa en la generación de dos señales A y B que provienen de dos contactos internos. Cuando se hace girar el mando del codificador, las señales A y B generan una secuencia cuyo orden depende del sentido del giro. Cuando se gira en sentido anti-horario, la primera señal que se activa es la A. En sentido horario será la B.

En el siguiente enlace podéis ver una explicación de funcionamiento. Centraos, fundamentalmente, en la parte que va desde el minuto 1 al 2:40. El que manejaréis en el laboratorio es muy similar, salvo que el paso por los contactos A y B provoca un '0' y, cuando no se está girando están a '1'. La posición de los contactos también es la contraria (en el del vídeo, el giro horario activa primero el pin A)

https://www.youtube.com/watch?v=v4BbSzJ-hz4

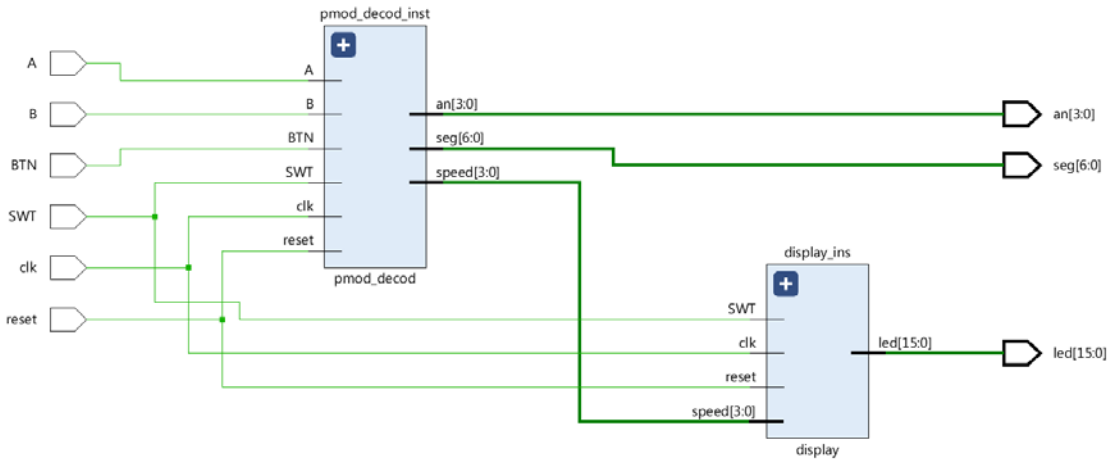
En Moodle tenéis un video explicativo con el rotatorio montado en la placa Basys3 (también en este enlace: https://unican-my.sharepoint.com/:v:/g/personal/fernandv_unican_es/EfPtbPYegt9KjwbQ1pi1GT8B-IG9Enk0Hcl99VcIK1znLw?e=CIc2e8).

Hay dos aspectos que debemos tener en cuenta a la hora de computar pasos de giro:

- El giro en cualquier sentido se puede abortar antes de completarlo (volviendo a la posición de partida) y no debería ser computado. Esto se muestra en el vídeo anteriormente referenciado.
- Las señales A y B son susceptibles de tener rebotes en sus cambios de valor. Ahora bien, cuando una de las señales tiene rebotes la otra está estable. Por lo tanto, evitando el fenómeno explicado en el punto anterior se evita también éste.

Diseño a realizar

En la siguiente figura se puede ver un esquema del circuito a realizar.



Está compuesto por dos bloques: un bloque decodificador para el rotatorio y bloque encargado de representar el desplazamiento en los leds a velocidad variable.

Decodificador del rotatorio (pmod_decod en la figura)

Este módulo será el encargado de leer las señales provenientes del rotatorio y detectar si se ha producido un giro en sentido horario o antihorario. Estas señales son A, B, BTN y SWT. Son asíncronas para nuestro circuito, así que debemos ponerlas una doble FF como protección frente a metaestabilidad.

El módulo debe generar una señal *speed(3:0)* que especifique el grado de velocidad al que queremos que se mueva el led iluminado. Su valor más bajo ("0000") querrá decir la velocidad más baja y su valor más alto ("1111") la velocidad más alta. Las velocidades concretas se programarán en el otro componente.

A, B: Debemos usarlas como entradas a una máquina de estados que, en función de la secuencia que sigan, detecte si se ha producido un giro horario o antihorario. Si se produce un giro horario se incrementará el valor de *speed*. Se decrementará en caso de giro antihorario. El valor *speed* se debe guardar en un registro.

BTN: Realizará una inicialización síncrona del módulo.

Las salidas *an* y *seg* son opcionales. En el video en que se muestra el funcionamiento del rotario se han usado para ver los valores de A, B, BTN y SWT.

Generador de secuencia en leds (display en la figura)

Este módulo debe iluminar un led y hacerle girar en un sentido u otro dependiendo del valor de SWT. La velocidad de giro debe depender de la entrada *speed*. A mayor valor, mayor velocidad. Las velocidades concretas se programan en este módulo. Se recomienda almacenarlas en un array de constantes para que puedan ser fácilmente modificables.

Se recomienda un diseño basado en dos contadores. Un primer contador llevará una cuenta de ciclos de reloj como base temporal. Cada vez que dicho contador llegue a su valor máximo habrá pasado una cantidad conocida de tiempo. A modo de ejemplo, en la demo que tenéis en el vídeo, se ha puesto una unidad temporal base de 1/100 s. Recordad que el reloj de la placa Basys3 es de 100 MHz.

Un segundo contador debe llevar la cuenta del número de veces que el primero ha llegado a su valor máximo. Para el ejemplo de 1/100 s, estaría contando cuántas centésimas de segundo transcurren. Cuando llegue a su valor máximo desplazaremos el led iluminado. El valor máximo le haremos depender de la velocidad a la que queremos que se produzcan los desplazamientos. Os recomiendo que el valor máximo al que el contador ha de llegar se actualice (si ha cambiado *speed*) en el momento en que llega a su valor máximo para que no se cambie mientras está incrementándose (y no ocurra que se le ponga un nuevo valor máximo que ya haya pasado y se desborde).

Para la prueba en placa son razonables valores que ronden el desplazamiento/segundo. Para las simulaciones, estos valores pueden llevarnos a tiempos de simulación elevados. Podéis, para la fase de simulación, trabajar con tiempos escalados mucho más bajos. Para la versión definitiva en placa habría que recordar poner los valores reales.

Simulación Funcional

Crear un *testbench* para la simulación del diseño desarrollado. Realizar una simulación y comprobar el correcto funcionamiento.

Síntesis, implementación y generación del *bitstream*

Seguir el procedimiento habitual. No olvidar el fichero de restricciones *.xdc*. Debe incluir la asignación de pines y el requerimiento de periodo para la señal de reloj.

Recordad que tenéis que reportar los datos de ocupación y frecuencia máxima de vuestra implementación.

Programación y prueba en placa

Seguir el procedimiento habitual para la programación de la FPGA. Comprobar el correcto funcionamiento en la placa.