Apellidos, Nombre: _____

Diseño de una cerradura digital

Estructura de Computadores
Ingeniería Informática. Tecnologías Informáticas
Dpto. de Tecnología Electrónica
Jorge Juan Chico, Septiembre, 2020

1 Material

- Ordenador con el entorno ISE de Xilinx instalado.
- Placa de desarrollo FPGA Digilent Basys 2 y documentación.
- Archivos base del diseño (kit del laboratorio).

2 Descripción

El objetivo de esta práctica es diseñar e implementar un circuito digital básico para controlar una puerta automática mediante una máquina de estados finitos. El circuito tiene cuatro entradas conectadas a botones (*b0*, *b1*, *b2* y *b3*) y una salida *z* que controla la puerta. El circuito debe abrir la puerta (z=1) cuando se pulsan tres botones en la siguiente secuencia: *b0* → *b2* → *b1*.

Si pensamos en los botones como en un teclado numérico, esta secuencia correspondería a un código de entrada "021". Una vez que la puerta está abierta se cerrará al pulsar cualquier botón del teclado.

El dispositivo debe ser un circuito secuencial síncrono, por lo que tendrá una señal de reloj *clk* y una entrada de puesta a cero *reset*. El esquema de entradas y salidas completo es el de la Figura 1:

En los sistemas que están controlados por pulsadores (botones) es muy importante tener en cuenta que cuando un botón se pulsa, la pulsación *no* dura sólo un ciclo de reloj: el reloj del sistema en la placa es de 50MHz por lo que el ciclo de reloj es de 20ns. Si un botón se pulsa durante 1 segundo, la entrada correspondiente al circuito estará activa durante 50 millones de ciclos de reloj. Por tanto, para que el sistema considere que se ha introducido una cifra del código, debe detectar la pulsación del botón y esperar a que la pulsación finalice. La máquina de estados que controla la cerradura debe considerar estos factores. Una posible estrategia es la siguiente:

- Moverse a un nuevo estado cuando se pulsa un botón determinado.
- Permanecer en ese estado mientras el mismo botón esté pulsado.
- Moverse a un nuevo estado cuando se suelte el botón.

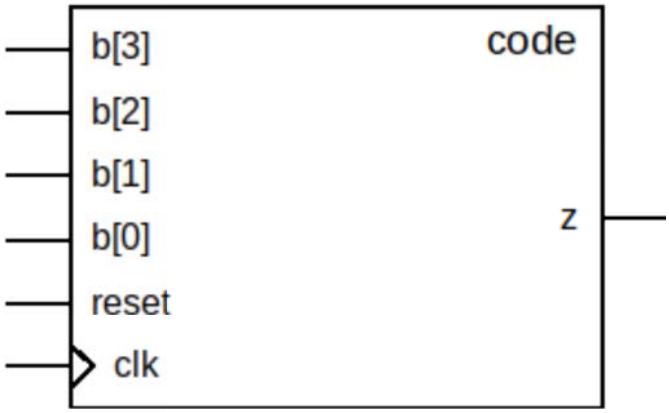

Figura 1: Entradas y salidas del módulo "code".

2.1 Resultados del aprendizaje

- Representar problemas secuenciales mediante máquinas de estados finitos.
- Describir máquinas de estado en Verilog.
- Preparar bancos de prueba y simular máquinas de estados.
- Conocer la utilidad de los detectores de flanco.
- Conocer la utilidad de los filtros de rebotes.

3 Trabajo previo

1. Dibuje una máquina de estados finitos que implemente el circuito de cerradura digital descrito.
2. Descargue los archivos base del diseño de la práctica que encontrará junto a este documento. Los archivos son:
   - `code.v`: diseño parcial de la máquina de estados.
   - `code_tb.v`: banco de pruebas.
   - `edge_detector.v`: filtro de rebotes y detector de flancos.
   - `Basys2_100_250_code.ucf`: archivo de restricciones de diseño genérico para la placa Basys 2.
3. Complete el diseño de la cerradura digital que se encuentra en el archivo `code.v`.
4. Use el banco de pruebas en `code_tb.v` para verificar el diseño. Puede simular el banco de pruebas con cualquier simulador Verilog como EDA playground (en la WEB) o Icarus Verilog. Para simularlo con Icarus Verilog puede hacer:

```
$ iverilog code.v code_tb.v
$ vvp a.out
$ gtkwave test.vcd
```

5. Lleve todos los archivos incluyendo sus modificaciones a la sesión de laboratorio.

6. ¿Cómo de difícil sería abrir la puerta con esta cerradura?
   a) Calcule el número de combinaciones de 3 cifras que se pueden hacer con la cuatro cifras (botones) disponibles. Este es el número de combinaciones de nuestro diseño. La probabilidad de abrir la puerta escribiendo una combinación aleatoria sería 1 entre este número.
   b) ¿Cuál sería la probabilidad de abrir la cerradura aleatoriamente si usamos códigos de 4, 5 o *n* cifras?

4 Trabajo en el laboratorio

4.1 Implementación y pruebas del sistema

1. Copie los archivos del diseño a una carpeta del ordenador del laboratorio.
2. Cree un proyecto en ISE con nombre **code**. Use las propiedades del proyecto adecuadas para la placa Basys 2:
   - General Purpose
   - Family: Spartan 3E
   - Device: XC3S100E
   - Package: CP132
   - Speed grade: -5
3. Añada los archivos `code.v`, `code_tb.v` y el archivo UCF al proyecto.
4. Simule el diseño con ISIM (el simulador del entorno ISE). Compruebe que la simulación es correcta.
5. Edite el archivo UCF y realice las conexiones adecuadas de las señales del diseño con los periféricos de la placa según la siguiente tabla:

Señal circuito	**Periférico**
clk	Reloj del sistema (MCLK)
reset	Interruptor 7
b[3]	Botón 3
b[2]	Botón 2
b[1]	Botón 1
b[0]	Botón 0
z	LED 0

6. Ejecute el proceso de síntesis e implementación del diseño para obtener el archivo de configuración de la FPGA (*bitstream*). Haga las correcciones necesarias en el caso de que haya errores.

7. Programe el diseño en la placa y compruebe que el circuito funciona correctamente tal como se indica el la descripción inicial. Compruebe que la señal *reset* asociada al interruptor 7 funciona correctamente.
8. Modifique el diseño para que la salida se active con una secuencia diferente de 3 cifras, pero sin repetir cifras iguales. Pida a otros compañeros que no conozcan la nueva clave que intenten abrir la cerradura probando códigos al azar. ¿Lo consiguen?

4.2 Mejora del sistema: detección de flancos

Hemos visto que cuando queremos controlar una acción de una máquina de estados mediante la pulsación de un botón, a menudo la máquina de estados tiene que detectar cuando se pulsa el botón y cuando se suelta. En general, al pulsar un botón queremos hacer una acción (cambio de estado) una sola vez, por lo que sería muy interesante que una pulsación hiciera que la señal conectada al botón se activara sólo durante un ciclo de reloj.

Ésta es precisamente la tarea de los llamados *detectores de flanco*: circuitos que activan su salida durante un solo ciclo cuando detectan un cambio en la entrada. El pulso puede ser generado en el flanco de subida, de bajada o en ambos, según el diseño del detector. En la Figura 2 se muestra un detector de flanco activado solo en el flanco de subida.

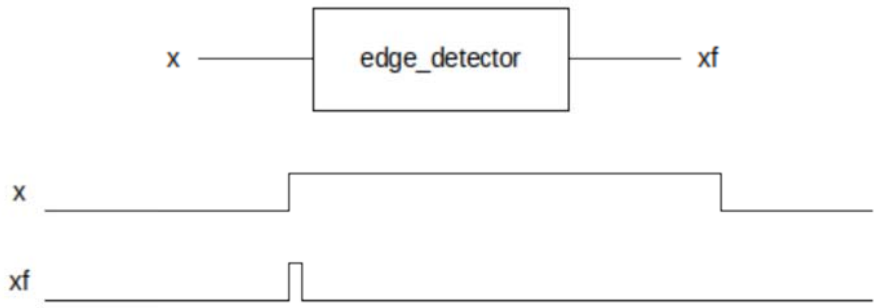

Figura 2: Detector de flancos.

De esta forma podemos simplificar mucho la máquina de estados, puesto que sólo tenemos que detectar cuando un botón se pulsa, pero no cuando se suelta, y un solo estado basta para detectar una pulsación, en vez de dos. Esto nos permitirá hacer que el sistema detecte códigos de 4 cifras usando menos estados.

1. Cree un proyecto en ISE con nombre **code_ed**. Use las propiedades del proyecto adecuadas para la placa Basys 2 como en el proyecto anterior.
2. Añada los archivos `code.v`, `code_tb.v`, `edge_detector.v` y el archivo UCF al proyecto. En el archivo `edge_detector.v` se encuentra ya diseñado un detector de flancos (módulo **edge_detector**).
3. Modifique el diseño del módulo detector del código (**code**) insertando un detector de flanco en cada entrada de botón *b[i]*, generando las correspondientes señales ya filtradas *bf[i]*. Por ejemplo:

```
edge_detector ed0(.ck(clk), .x(b[0]), .z(bf[0]));
...
```

4. Diseñe una nueva máquina de estados del módulo **code** para que emplee las nuevas entradas filtradas *bf[i]*, de forma que detecte una secuencia de 4 cifras, por ejemplo: 0221. Dado que ahora las entradas sólo se activan durante un ciclo con cada pulsación, sólo debe necesitar un estado por cada pulsación detectada. Si lo necesita, dibuje un diagrama de estados previamente.
5. Implemente y pruebe el diseño modificado. En caso de que la operación no sea correcta, modifique el banco de pruebas original para adaptarlo a la nueva secuencia e intente localizar el error mediante la simulación.