Informe de laboratorio N°5

Fernando Guiraud

Universidad tecnológica de Panamá

Microprocesadores

1EE141

fjguiraud@gmail.com

1. **INTRODUCCIÓN**

En esta asignación de laboratorio, debe diseñar un sistema que contenga un pequeño módulo RAM y

probarlo llenándolo con valores, y leerlos y modificarlos repetidamente. Las entradas de datos a su

sistema son: (i) una dirección para seleccionar qué ubicación de memoria se está leyendo/escribiendo; y

(ii) datos a escribir en la ubicación de la memoria (si no está escrito). La salida de su sistema es el valor

de datos en la ubicación en la que se está abordando la memoria (es decir, el valor de lectura), que se

muestra en una pantalla de 7 segmentos.

El siguiente diagrama de bloques muestra el sistema completo.



Las entradas y salidas del sistema son las siguientes:

reloj: entregado a todos los componentes sincronizados dentro del sistema.

 Botones de presión: entradas de los cinco botones de la tarjeta, rotulados btnL, btnR, btnU, btnD y

btnC (centro).

 Addr: un valor de 4 bits que proporciona la dirección de la ubicación de memoria a la que se hace

referencia. Este valor se ingresa en la tarjeta a través de los cuatro controles deslizantes del extremo

izquierdo. Por lo tanto, esta asignación solo necesita una unidad de memoria con 16 ubicaciones.

 D: un valor de 4 bits que proporciona los datos para almacenar en la memoria. Por lo tanto, cada

palabra en la memoria es un único dígito hexadecimal [0..F]. Este valor se ingresa en la tarjeta a

través de los cuatro controles deslizantes a la derecha.

 segmentos: el valor leído de la memoria se codifica para la salida en la pantalla de 7 segmentos.

 digitselect: Un valor de salida constante que selecciona el extremo derecho de los cuatro caracteres

de 7 segmentos.

El comportamiento del sistema es el siguiente:

* El usuario selecciona la ubicación de la memoria (0 a 15) deslizando los interruptores correspondientes a Addr.
* Cuando el usuario proporciona Addr, la unidad de memoria genera el valor almacenado en esa ubicación, mem[Addr]. Este es un valor de 4 bits que se muestra como un carácter en la visualización del segmento.
* Si el usuario presiona uno de los botones: Se modifica el valor en la ubicación de la memoria de referencia. Su nuevo valor depende de qué botón se presionó, de la siguiente manera:

botón central: el valor de la memoria se establece en cero

botón arriba: el valor de la memoria se incrementa en 1

botón Abajo: el valor de la memoria se reduce en 1

botón izquierdo: el valor de la memoria es AND bit a bit con el valor D

botón derecho: el valor de la memoria es OR bit a bit con el valor D

NOTA: Para cada uno de estos escenarios, la operación de escritura de memoria debe

tener lugar cuando se presiona el botón; posteriormente, su máquina de estado debe

esperar hasta que se suelte el botón antes de volver a su estado inicial.

1. **OBJETIVOS**

* Diseñar una unidad de memoria simple (RAM)
* Más práctica con máquinas de estado (FSM)
* Más práctica con los interruptores anti-rebotes
* Más práctica con pantallas de 7 segmentos

1. **MATERIAL Y EQUIPO**

* Tarjeta Elbert V2 – Spartan 3A FPGA Development Board

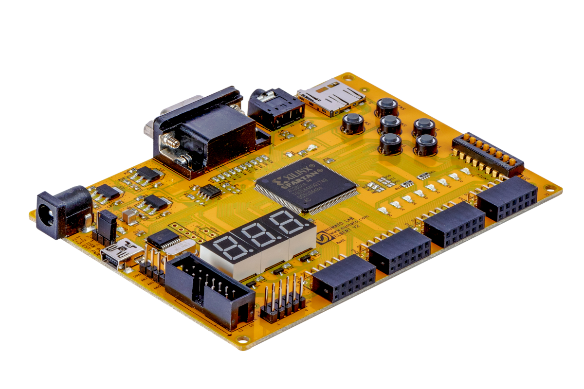


Figura 2: Tarjeta Spartan 3A FPGA

* Xilin ISE, 32-bit Project Navigator.

1. **DESARROLLO**

El primer paso para implementar el semisumador después de configurar la tarjeta Elbert V2, consiste en definir las entradas y salidas del proyecto principal:

Texto

Descripción generada automáticamente

El ISE Project Navigator inicializa las librerías básicas para comenzar a elaborar el algoritmo. Sin embargo, para implementar este algoritmo será necesario utilizar dos librerías numéricas adicionales.

Texto

Descripción generada automáticamente

En esta experiencia de laboratorio es necesario crear distintos componentes para cumplir la función del diagrama establecido. Primeramente, es necesario crear 5 bloques de anti-rebotes para cada botón de entrada de las 5 operaciones con los datos de la memoria. El código de un anti-rebote es el siguiente:

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente con confianza media

Texto

Descripción generada automáticamente

Antes de utilizar este antirebotes, hay que modificar el valor del escalón N en este caso con un reloj de 12MHz que corresponde al clock interno de la tarjeta a utilizar.

Después de esto se realiza la operación descrita en comentarios en la imagen anterior para encontrar un valor de N de 21 para una pulsación de 200ms.

Texto

Descripción generada automáticamente

Seguidamente, procedemos a mapear cada botón correspondiente a cada anti-rebote para generar un componente que describa a cada botón.

Es necesario crear una señal adicional por cada botón para invertir la señal ya que la tarjeta trabaja con lógica negativa.

Después de esto se agrega el módulo de la memoria RAM, que fue proporcionado.

Texto

Descripción generada automáticamente

A este módulo RAM también le haremos un mapeo que será descrito más adelante.

Posteriormente creamos la máquina de estado encargada de variar entre las operaciones al presionar cada uno de los anti-rebotes y enviar el estado de lectura o escritura al módulo de sincronización de RAM antes definido.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Después de implementar la maquina de estado, creamos un modulo que se encargue de procesar los datos binarios de las operaciones de los botones y los transforme a hexadecimal para después presentarlos en un display de 7 segmentos.

Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente

Después de esto creamos el componente encargado de procesar las operaciones lógicas y matemáticas de las ordenes enviadas por los botones anti-rebotes.

Texto, Carta

Descripción generada automáticamente

Tabla

Descripción generada automáticamente con confianza media

Por ultimo se hace un mapeo de las entradas y salidas de cada componente para que se ajusta a la forma del diagrama proporcionado.

Un conjunto de letras blancas en un fondo blanco

Descripción generada automáticamente con confianza media

En esta imagen podemos ver la maquina de estado (FSM), el modulo de sincronización de RAM (sync\_ram), el modulo de procesamiento lógico y matemático (logicomb), y por ultimo el modulo de conversión de binario a hexadecimal y presentación del digito en el display de 7 segmentos.

Para generar este mapeo fue necesario crear ciertas señales intermedias que toman valores de salidas y son usados de entradas en otros componentes.

Texto

Descripción generada automáticamente

Por ultimo como se puede ver en la imagen anterior, se habilita la variable Ena como un bus de datos de tres bits que definen la combinación de dígitos del display de 7 segmentos que se encuentran habilitados, en este caso como solo se trabaja con 4 bits en hexadecimal, se representan todos los elementos en un solo digito.

1. **RESULTADOS**

Para comprobar la funcionalidad de este código, procedemos a transferir el programa a la tarjeta, para esto tenemos que crear un archivo ucf que contenga la ruta de las entradas y salidas de la tarjeta que correspondan físicamente a el hardware deseado.

El primer pin usado es la ruta al clock de 12 MHz interno de la tarjeta, después las rutas a los 7 segmentos, la habilitación del digito del display.

La ruta de la variable D, corresponde a un bus de datos de 0 a 3 bits, que utiliza una combinación de 4 dip switchs, los cuales almacenan la combinación de entrada para el componente lógico.

Los pines de la variable Dis, corresponden a la salida de las operaciones lógicas que son enviadas al componente de sincronización de la ram, y sirven de referencia para comprobar que los datos que se presentan en el display son los correctos en binario. Corresponden a los primeros 4 leds de la tarjeta.

Addr corresponde a la dirección de 4 bits y se usan los restantes 4 dip switchs después de la entrada D.

Las ultimas 6 declaraciones corresponden a los botones que van hacia el componente de anti-rebotes, además del botón de reset que se encuentra de primero.

Tabla

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Ahora procedemos a transferir el código a la tarjeta con el ejecutable proporcionado por el fabricante, en este caso Numato.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Escogemos el puerto correspondiente y enviamos el programa.

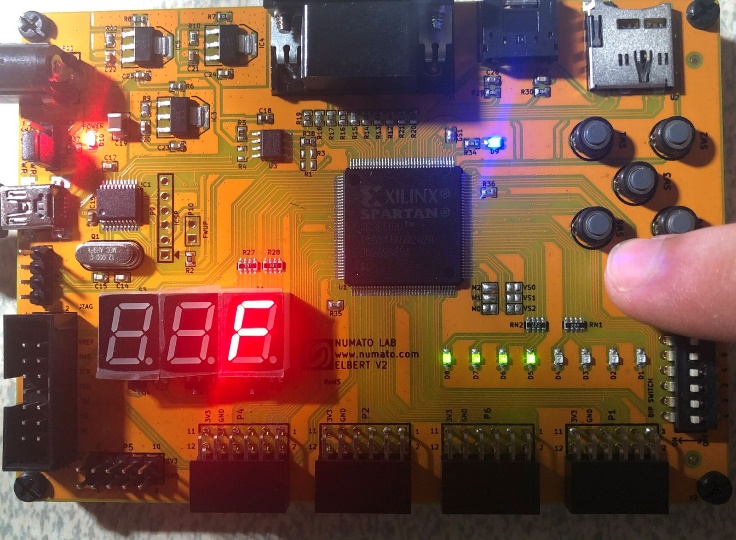
En las siguientes imágenes podemos ver el funcionamiento del módulo RAM.

En esta imagen describimos el hardware de cada una de las variables del sistema:

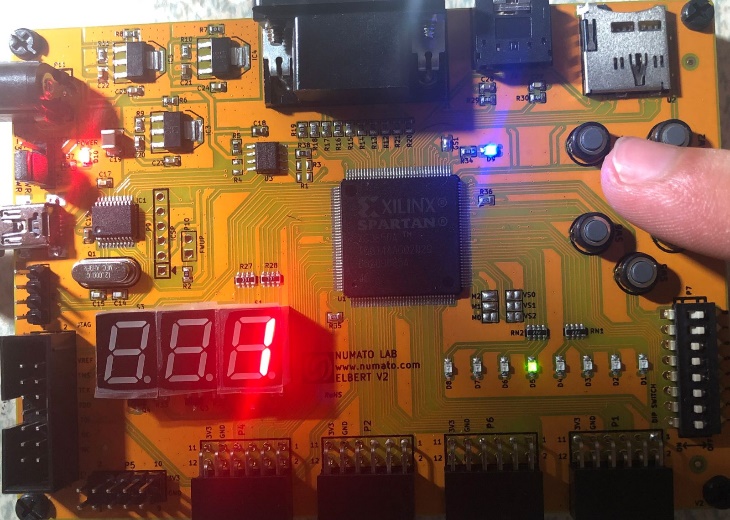
Un circuito electrónico

Descripción generada automáticamente

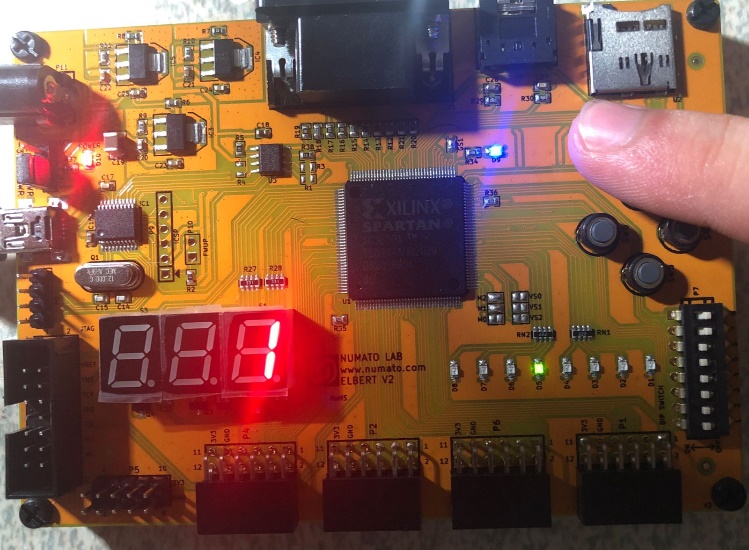
Restamos uno con el btnD:



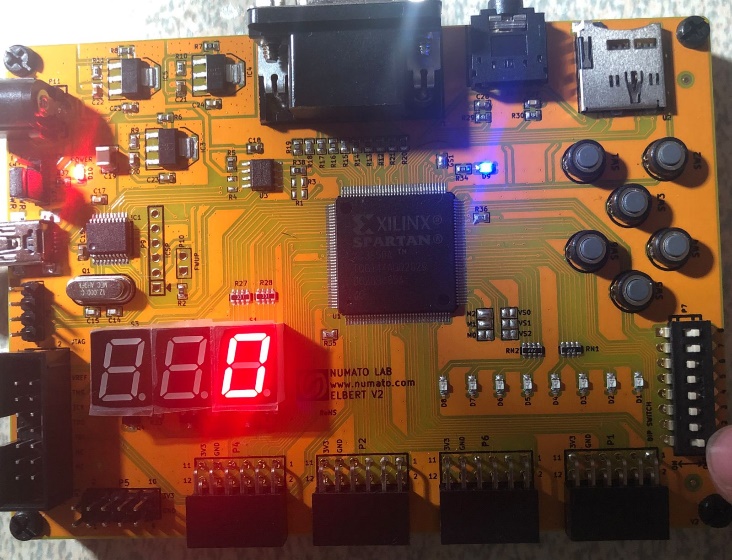
Sumamos uno con el btnU:



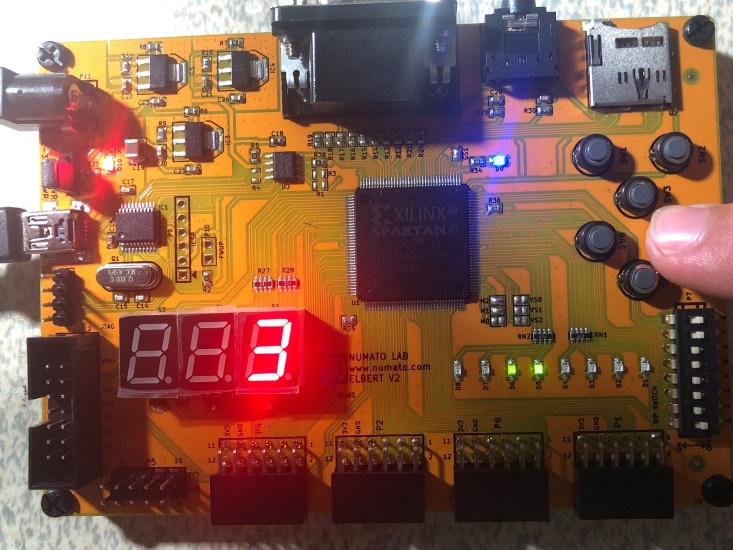
Guardamos el valor en la memoria oprimiendo el botón de RST:



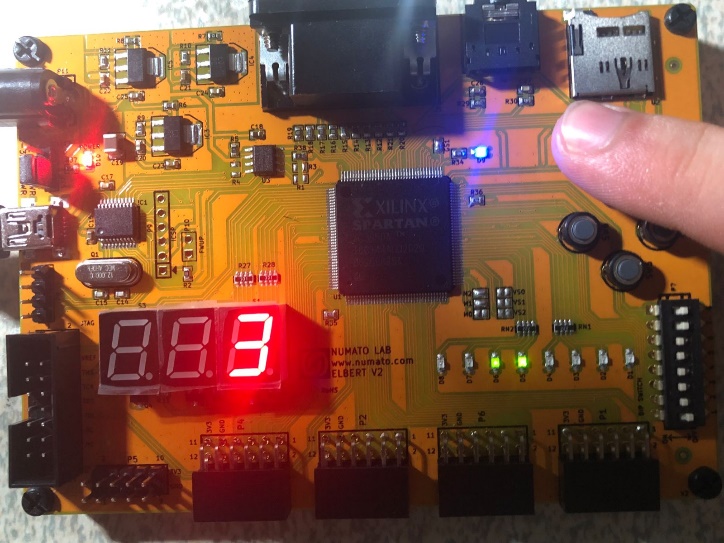
Cambiamos la posición de la memoria:



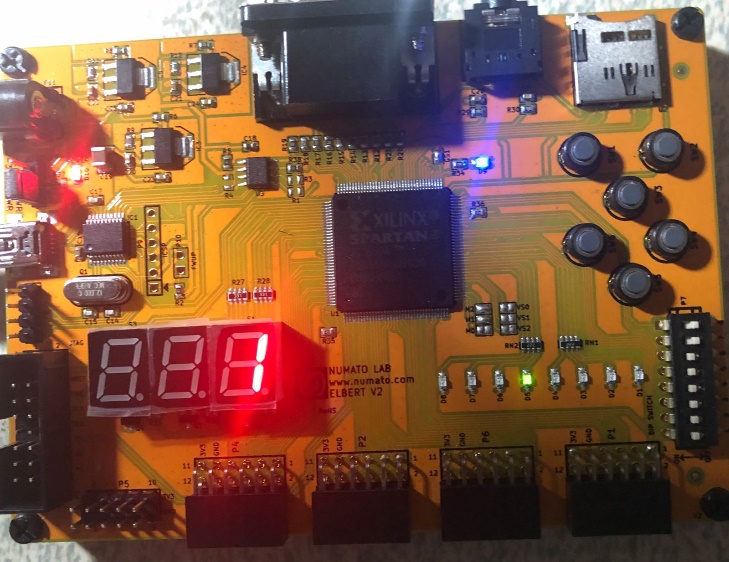
Oprimimos el btnR para operar bit a bit con los valores de D:

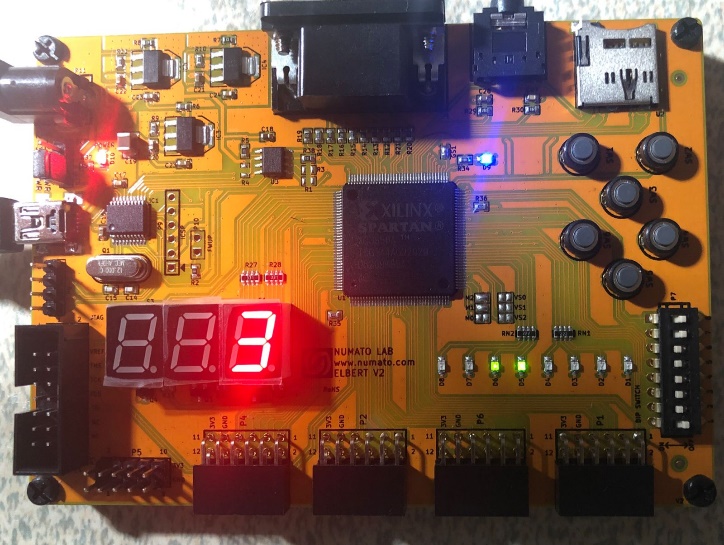


Oprimimos el botón de RST para guardar el valor en la memoria:



Comprobamos el valor escrito anteriormente en la memoria:





1. **CONCLUSIÓN**

Una memoria RAM es una aplicación directa de una máquina de estado, que implementa distintos tipos de algoritmos utilizados anteriormente incluyendo conceptos como component, código concurrente y secuencia.

Para utilizar botones de manera precisa, es necesario tomar en cuenta las fallas mecánicas del rebote que estos producen al presionarlos, para esto se utiliza un algoritmo conocido como anti-rebote, que consiste en utilizar el clock interno de la tarjeta para crear un pulso rectangular que corte las interferencias de un posible rebote del interruptor al ser presionado.

Hay muchas formas de implementar el mismo algoritmo, una forma de simplificar los procesos lógicos complejos consiste en el uso de máquinas de estado.

1. **REFERENCIAS**

D. L. Perry, VHDL. New York: McGraw-Hill, 1991