Desarrollo de un Turnero Digital Usando FPGA's Laboratorio de Electrónica Digital Módulo: 1

Héctor F. Jiménez S.
hfjimenez@utp.edu.co
PGP KEY ID: 0xB05AD7B8

Sebastian Zapata
sebastzapata93@gmail.com
PGP KEY ID: 0xfffff

Jorge Mario Gil
jorgemario.gil@utp.edu.co
PGP KEY ID: 0xfffff

Fecha de Entrega: Agosto, 2016
Profesor: Ing.Msc(c) Ramiro Andres Barrios Valencia

1 Objetivos

- Fortalecer y poner en práctica la teoría de circuitos combinacionales.
- Fortalecer el desarrollo de sistemas digitales, utilizando el lenguaje *VHDL* y el entorno de desarrollo Xilinx ISE.
- Identificar la arquitectura, esquemáticos y hardware integrado de la placa de desarrollo *NEXYS2*.
- Implementar en lenguaje VHDL un módulo que permita la detección correcta de los pulsos entregados por los botones presentes en la tarjeta de desarrollo[1]
- Identificar posibles problemas a la hora de acoplar los otros módulos a desarrollar.

1. Datasheet Oficial de la placa Nexys 2 Manual[2]

2 Marco Teórico

Utilizar switches y pulsadores mecánicos en un sistema electrónico es una práctica muy común, pues sirven de interface entre el usuario y el sistema embebido para seleccionar y programar funciones que este posee; en la práctica actual muchos de estos sistemas mecánicos están siendo reemplazados por displays táctiles pero hay situaciones industriales principalmente en las cuales un pulsador mecánico sería una solución más apropiada y eficiente en la implementación, pues al ser un sistemas mecánicos estos no se degradan tan rápido como lo harían las pantallas táctiles en un entorno industrial. Una de las grandes desventajas que poseen los pulsadores son los rebotes o saltos como se puede observar en la figura 1.

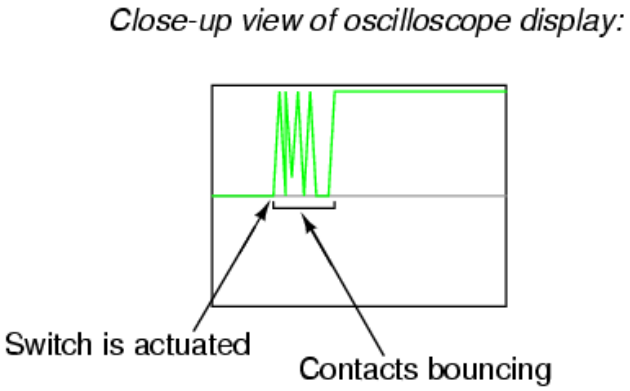

Figure 1: Rebotes al presionar un pulsador mecánico, señales de osciloscopio.

Los elementos mecánicos del pulsador están constituidos por un par de contactos eléctricos que se unen o separan por medios mecánicos, en la figura 2 se muestra un pulsador común, figuras3 4 son vista detalladas. Los falsos contactos que se producen cuando los accionamos, crean falsas señales instantáneas, ocasionando un mal funcionamiento del dispositivo, no olvidando mencionar que existe una barrera o GAP que se produce por el aire entre los elementos mecanicos*(aunque típicamente esto es despreciable por su baja fuerza disipativa)* induciendo múltiples pulsaciones o flancos como se muestra en la figura 1.

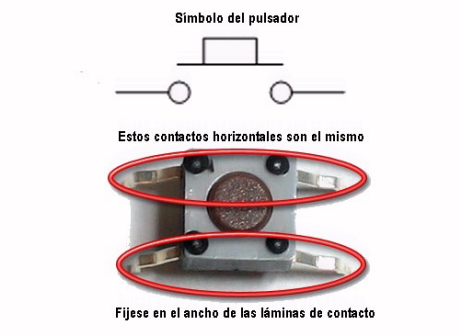

Figure 2: Pulsador Común, Composición.

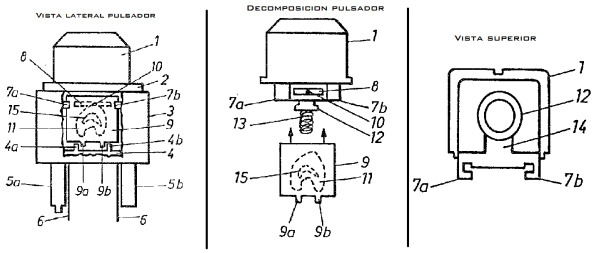

Figure 3: Decomposicion de un Pulsador común

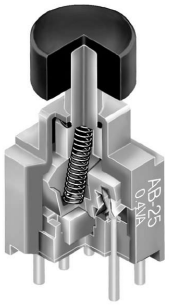

Figure 4: Vista en 3d de un pulsador y sus elementos

En algunos casos esta acción puede producir una chispa debido a la corriente que atraviesa los contactos, disminuyendo el tiempo útil de los contactos eléctricos. La chispa se produce siempre al separar los contactos (desconectar), en ocasiones parece que también salta al conectarlos, eso es debido a los rebotes mecánicos que se producen al cambiar de estado. Existen muchas formas de lidiar con este problema, tanto soluciones por hardware físico como el circuito

simple de la figura 5 o por software como lo realizaremos para el desarrollo de este modulo, mediante la implementación de un circuito lógico provisto por la empresa Digikey.[3]

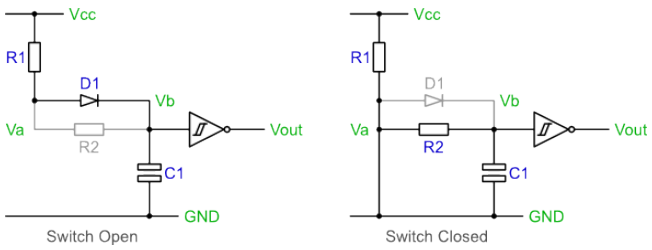

Figure 5: Solución utilizando un simple arreglo electrónico. Pulsadores con resistencias de Pull Ups.

3 Arquitectura de la Placa *NEXYS 2*

La tarjeta de entrenamiento *NEXYS 2* es una plataforma de desarrollo fabricada por la empresa [4] aunque se encuentra descontinuada para la venta y recomiendan utilizar esta misma en su version **4**, utiliza una fpga Spartan 3E optimizada para desarrollo de aplicaciones donde se requiere implementar diseños de lógica compleja, e ideal para el procesamiento de señales y desarrollo de sistemas embebidos como se menciona en General Spartan Versions[6], Spartan-3E FPGA Family: Introduction and Ordering Information[7].
Algunos elementos destacables de la placa son :

Connectores

- Puerto USB2.0
- Puerto VGA
- PS/2
- Puerto RS232

3. `http://www.digikey.com/`
4. Digilentinc Página Oficial Nexys2 Spartan3E[5]
6. `http://www.xilinx.com/support/documentation-navigation/silicon-devices/mature-products/spartan-3e.html`
7. `http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf`

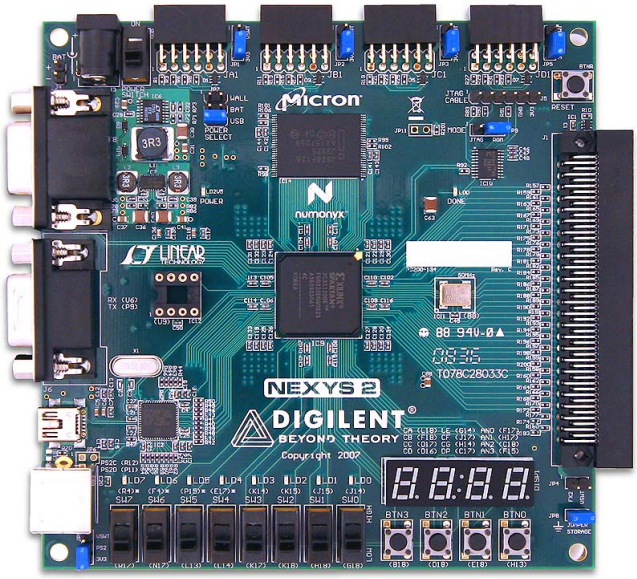

Figure 6: Modelo de la Nexxys , Versión 2.

Algunas de las que más nos llamaron la atención son:

Caracteristicas

- Xilinx Spartan-3E FPGA 500K
- Memoria PSDRAM de 16 MB fast Micron®
- Memoria de 16 MB Intel® StrataFlash® Flash R
- Trabaja con la version Free de ISE®/WebPACK
- Oscillador de 50 MHz
- Fuentes reguladas de 3.3V@3A/100mA(principal),3.3V@150mA/60mA,2.5V/1.2V@1.4A/50mA
- Todas las entradas tiene protección contra cortocircuitos y descargas electroestática
- Incluye 8 leds, cuatro display siete segmentos, cuatro pulsadores, 8 switches.. . .

Para plantear una solución al problema mencionado en 2, nosotros hemos descargado el esquemático para comprender circuitalmente cuál es la configuración de los pulsadores, aunque generalmente para desarrollos importantes en electrónica, se compran pulsadores de alta calidad, y resistencia mecánica a presiones momentáneas e instantáneas, pero dado que esto es una placa de entrenamiento,

estudiantil asumimos que los pulsadores que esta posee son netamente mecánicos que cuenta con el diseño de la figura 7

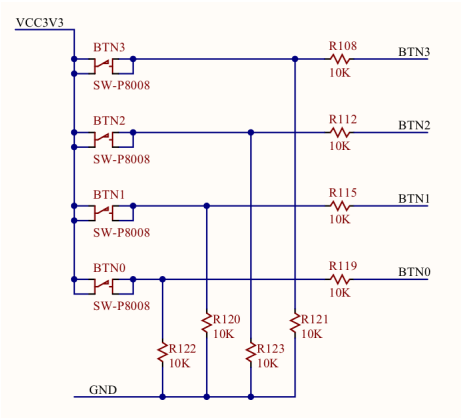

Figure 7: Esquema provisto en el esquemático, 4 pulsadores.

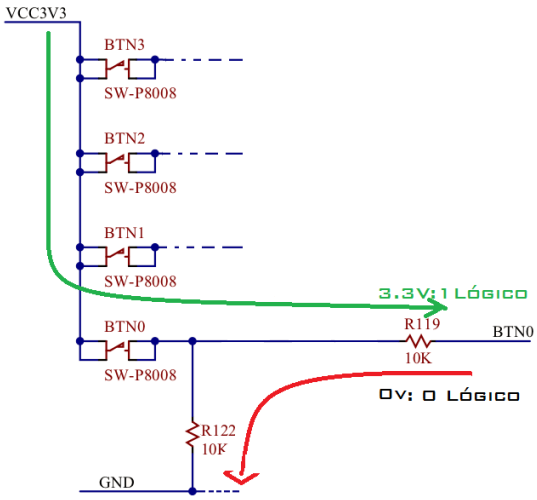

Figure 8: Circulación de la corriente en el circuito, resistencia de 10kΩ protección contra cortocircuito

La figura 8 también muestra una configuración de un solo pulsador vemos la resistencia de pull down. Con colores hemos indicado el accionamiento del pulsador, tome como ejemplo 8, el color rojo indica cuando el pulsador **BTN0** no ha sido presionado, esto indica que los pulsadores son de tipo **N.O** ó *Normally Open in Low Mode* presentando un **0 lógico** a la salida, al presionar el pulsador, el contacto interno que este posee cierra el circuito conectándolo con la fuente

de 3.3v como lo muestra el color verde de la figura 8.

4 Antirebote por Software

Resolver el problema de los rebotes de un pulsador puede ser una tarea fácil o compleja, la idea básica es *muestrear* la señales de entrada de los pulsadores en un intervalo regular de tiempo para notar si hay cambios en la entrada, si no existieron cambios se debe mantener el estado actual del pulsador, si hay pulsos anormales debemos filtrar la señal. Hay dos aproximaciones como se menciona en el sitio web Labbookpages.co.uk[8] para un configuración de pull up, nosotros utilizaremos esto mismo para una configuración de pull-down sin decidir si será modulo ya que debemos realizar prácticas experimentales con la tarjeta:

Contador Este contador nos dirá cuanto tiempo ha pasado desde que la señal ha estado en alto, si la señal ha estado en alto durante un tiempo definido por nosotros *(50 mS)*, entonces es considerado como un pulso estable, y debería de setear la salida a 1. Esta aproximación es parecida vista en la implementación de DigiKey[9] pero más completa.

Registro 8bits Esta aproximación es similar a la mencionada anteriormente, pero utiliza un registro de desplazamiento de 8bits en vez de un contador, lo que hace es ir desplazando los bits hasta alcanzar su valor máximo.

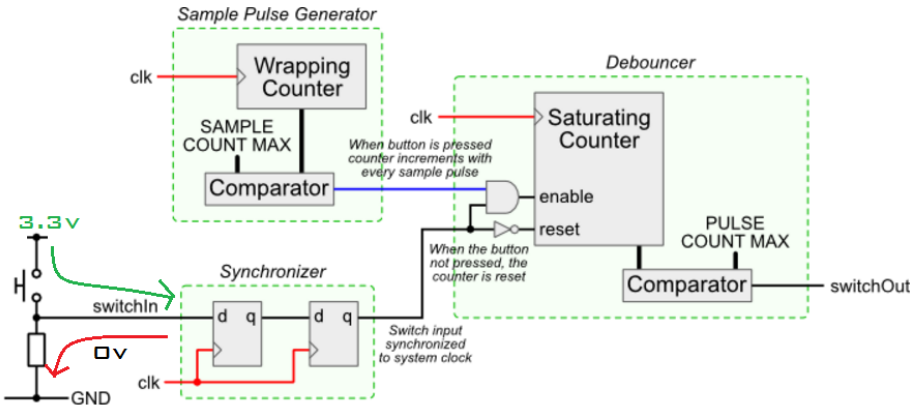

Figure 9: Desarrollo a implementar. Tiempo para presionar el pulsador mínimo de 33mS

La figura 9 está compuesta por tres partes principales donde cada una tiene encargada una función como se muestra a continuación

8. http://www.labbookpages.co.uk/electronics/debounce.html
9. https://eewiki.net/pages/viewpage.action?pageId=4980758

Sample Pulse Generator: Es un contador que incrementa el reloj en cada cambio de flanco Cuando el alcanza la máxima cantidad de pulsos muestreados un pulso es generado y la cuenta se reinicia, el pulso muestreado es usado para muestrear la entrada del pulsador.

Synchroniser: Se encarga de sincronizar la señal de entrada en la figura 9 **SwitchIn** con la señal de reloj, así que si no se presiona el pulsador el contador estará deshabilitado, ya que su reset es activo bajo.

Debouncer: Este contador se incrementa en cada pulso muestreado cuando el pulsador es presionado. Cuando el pulsador no es presionado el contador está en reset. Si el contador llega a su valor máximo, la salida del comparador se pone en alto, de lo contrario está en bajo. **AGREGAR ECUACIONES DE TIEMPOS!**

5 Explicación Código VHDL

Descripción del Pseudo código para el módulo anti rebote:

```
1 Setup a counter variable, initialise to zero.
2 Setup a regular sampling event, perhaps using a timer. Use a period of about 1ms.
3 On a sample event:
4   if switch signal is high then
5     Reset the counter varaible to zero
6     Set internal switch state to released
7   else
8     Increment the counter variable to a maximum of 10
9   end if
10  if counter=10 then
11    Set internal switch state to pressed
12  end if
```

Posible implementacion de un Anti-Rebote.

La implementación acontinuacion presenta el codigo implementado para el primer modulo. Se puede observar que la entidad es la sección que describe las entradas y salidas que utilizaremos, nuestras entradas seran el boton y la señal de reloj *CLK, btn_in*, y como salida la señal *Q*.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity counters_1 is
    port(CLK,btn_in: in std_logic;
         Q : out std_logic);
end counters_1;
--11100100111000011100000
architecture archi of counters_1 is
signal counter: std_logic_vector (23 downto 0) := (others => '0');

begin
    process (CLK, btn_in)
    begin
        IF (CLK'event and CLK='1') THEN
                                Q <= '0';
                                IF counter > 0 AND btn_in = '0' THEN
                                                        counter <= counter + 1;
                                    END IF;
                        IF (btn_in='1') THEN
                                        IF (counter >= 10000000) THEN
                                                    Q <= '1';
                                                            counter <= (others =>'0');
                                    ELSE
                                                counter <= counter + 1;
                                    END IF;
                                END IF;
                          END IF;
END PROCESS;

END archi        ;
```

6 Prácticas Experimentales

No se pudieron realizar pruebas con la placa nexys2, no se programa la fpga.

7 Anexos

los demás archivos son adicionales que resuelven la misma implementación, se encuentran en el repositorio `github.comheticor915DigitalElectronicsLabtreemasterReportsModule1`
Nota: Las referencias utilizadas se encuentran en los pies de página. Si requiere de manera detallada estas contacte con *miembros del equipo.*