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

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

Steffany Lopez Segura
steffany@utp.edu.co

Fecha de Entrega: Febrero 24, 2016
Profesor: Ing.Msc(c) Ramiro Andres Barrios Valencia

1 Objectivos

- Fortalecer y poner en práctica la teoria de circuitos combinacionales.
- Fortalecer el desarrllo de sistemas digitales, utilizando el lenguaje *VHDL* y el entorno de desarrollo Xilinx ISE.
- Identificar la arquitectura, esquematicos 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 acoplarnos con los otros módulos a desarrollar.

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 opciones 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 mas apropiada y eficiente en la implementación,

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

pues al ser sistemas mecánicos estos no se degradan tan rápido como lo harían las pantallas táctiles. Una de las grandes desventajas que poseen los pulsadores son los rebotes, los elementos mecánicos del pulsador son elásticos, esto hace que cuando pulsemos o despulsemos, haya un instante en el que se realicen varios contactos seguidos, además de una barrera o GAP que se produce por el aire allí*(típicamente es despreciable por su baja fuerza discipativa)* que existen entre los elementos elásticos, esto puede inducir múltiples pulsaciones, flancos y falsas señales instantáneamente, ocasionando un mal funcionamiento del dispositivo. Existen muchas formas de lidiar con este problema, tanto soluciones por hardware físico o por software como lo realizaremos en este reporte, mediante la implementación de un circuito lógico provisto por la empresa Digikey[3]
En definitiva, se trata de un mecanismo simple (los hay muy sofisticados), constituido por un par de contactos eléctricos que se unen o separan por medios mecánicos. En electricidad, los falsos contactos que se producen el ser utilizados normalmente, en algunos casos produce una chispa debido a la corriente que atraviesa los contactos, provocando que quemen en parte y ennegreciendo los contactos eléctricos, lo que a la larga acaba deteriorando dichos contactos. 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.
Esto que en electricidad se considera normal, en electrónica es un verdadero nido de problemas, debido a dichos falsos contactos. Por su propia naturaleza, al cambiar de posición un interruptor, los contactos chocan entre sí y esto significa una serie de falsos contactos que se reproducen de un modo sin control, por lo que se generan los temidos rebotes (debounce en inglés), estos rebotes, se producen incluso cuando unimos dos cables desnudos, simulando un interruptor o pulsador.
Los problemas

3 Arquitectura del 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 :

3. http://www.digikey.com/
4. Digilentinc Pagina 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

Connectores

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

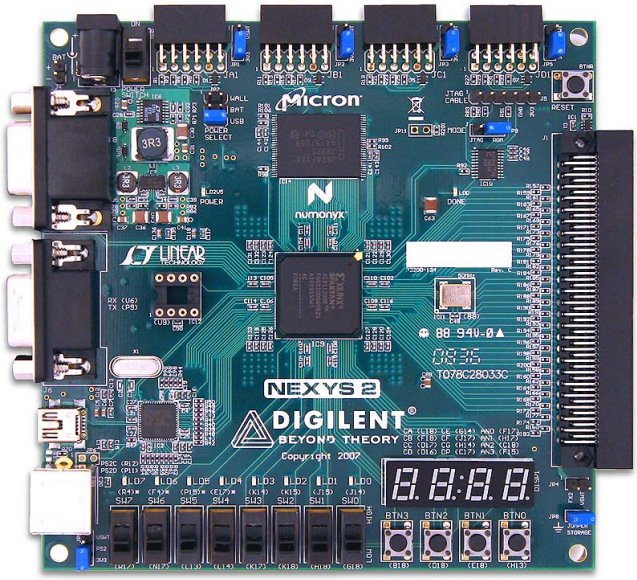

Figure 3.1: Modelo de la Nexxys , Versión 2.

Algunas de las que mas nos llamaron la atencion 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 electroestatica

- Incluye 8 leds, cuatro display siete segmentos, cuatro pulsadores, 8 switches four pushbuttons, eight slide switches

entre otras...

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 momentaneas e instántaneas, 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 3.2

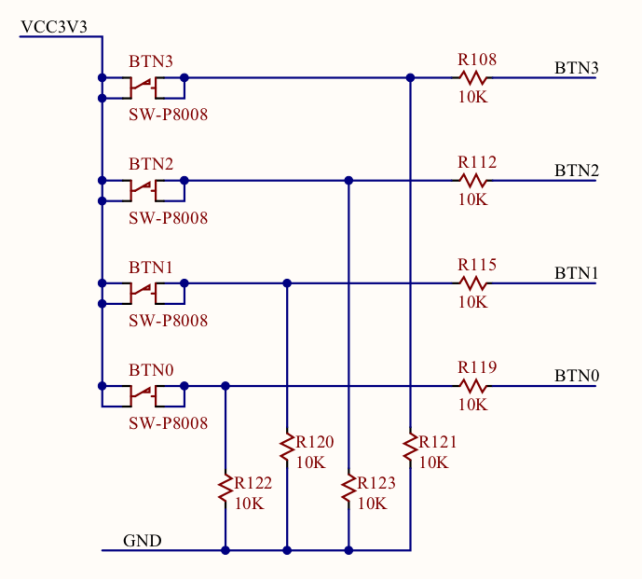

Figure 3.2: Esquema provisto en el esquematico, 4 pulsadores.

La figura 3.3 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 3.3, 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* 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.

4 Explicación Código VHDL

This is some VHDL code: we need to describe it very carefully and in a detail form.

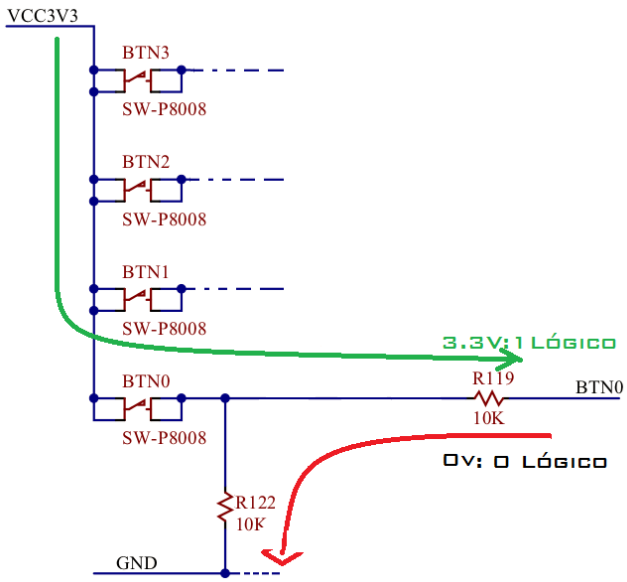

Figure 3.3: Circulación de la corriente en el circuito, resistencia de 10k protección contra cortocircuito

```
process
begin
  CLK <= '1'; wait for 10 NS;
  CLK <= '0'; wait for 10 NS;
end process;
```

Codigo de Ejemplo de Anti-Rebote.

```
--------------------------------------------------------------------------------
--
--   FileName:         debounce.vhd
--   Dependencies:     none
--   Design Software:  Quartus II 32-bit Version 11.1 Build 173 SJ Full Version
--
--   HDL CODE IS PROVIDED "AS IS."  DIGI-KEY EXPRESSLY DISCLAIMS ANY
--   WARRANTY OF ANY KIND, WHETHER EXPRESS OR IMPLIED, INCLUDING BUT NOT
--   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
--   PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL DIGI-KEY
--   BE LIABLE FOR ANY INCIDENTAL, SPECIAL, INDIRECT OR CONSEQUENTIAL
--   DAMAGES, LOST PROFITS OR LOST DATA, HARM TO YOUR EQUIPMENT, COST OF
--   PROCUREMENT OF SUBSTITUTE GOODS, TECHNOLOGY OR SERVICES, ANY CLAIMS
--   BY THIRD PARTIES (INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF),
--   ANY CLAIMS FOR INDEMNITY OR CONTRIBUTION, OR OTHER SIMILAR COSTS.
--
```

```
--   Version History
--   Version 1.0 3/26/2012 Scott Larson
--     Initial Public Release
--
--------------------------------------------------------------------------------

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;

ENTITY debounce IS
  GENERIC(
    counter_size  :  INTEGER := 19); --counter size (19 bits gives 10.5ms with 50MHz clock)
  PORT(
    clk     : IN  STD_LOGIC;  --input clock
    button  : IN  STD_LOGIC;  --input signal to be debounced
    result  : OUT STD_LOGIC); --debounced signal
END debounce;

ARCHITECTURE logic OF debounce IS
  SIGNAL flipflops   : STD_LOGIC_VECTOR(1 DOWNTO 0); --input flip flops
  SIGNAL counter_set : STD_LOGIC;                    --sync reset to zero
  SIGNAL counter_out : STD_LOGIC_VECTOR(counter_size DOWNTO 0) := (OTHERS => '0'); --counte
BEGIN

  counter_set <= flipflops(0) xor flipflops(1);   --determine when to start/reset counter

  PROCESS(clk)
  BEGIN
    IF(clk'EVENT and clk = '1') THEN
      flipflops(0) <= button;
      flipflops(1) <= flipflops(0);
      If(counter_set = '1') THEN                  --reset counter because input is changing
        counter_out <= (OTHERS => '0');
      ELSIF(counter_out(counter_size) = '0') THEN --stable input time is not yet met
        counter_out <= counter_out + 1;
      ELSE                                        --stable input time is met
        result <= flipflops(1);
      END IF;
    END IF;
  END PROCESS;
END logic;
```

5 Prácticas Experimentales

Poner a qui la practica realizada en laboratorio.