Módulo 1: Antirrebote o Detector de Pulso.

Desarrollo de unidad de control para la detección en la pulsación de botones.

Lady Johanna Trejos Hernández

Andrés Felipe Betancurt Rivera

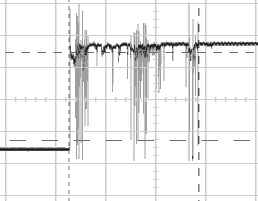
Docente Ramiro Andrés Barrios Valencia

Universidad Tecnológica de Pereira

Facultad de Ingenierías FEECS

Asignatura Laboratorio de Electrónica Digital

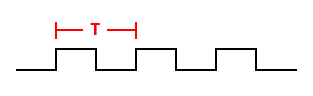
22 de febrero de 2017

Cuando un botón es pulsado, tarda cierto tiempo en alcanzar un estado estable, es decir, que durante muy poco tiempo (del orden de los milisegundos) pueden cambiar entre el estado pulsado y no pulsado hasta que la señal se estabiliza. A pesar de ser muy poco tiempo, es el suficiente para que un sistema digital lo detecte como varias pulsaciones. La siguiente imagen muestra una señal de rebote típica:

Es por esta razón que al implementar botones en un circuito, se debe desarrollar también una unidad de control para la detección en la pulsación de estos y diferenciarlos de una señal de rebote.

Conociendo que el dispositivo está sincronizado con una señal de reloj, todos los procesos deben estar regidos por ella. Es así como para reconocer si una señal con valor '1' significa que el botón se ha pulsado o simplemente está dentro del periodo de rebote, se debe establecer un tiempo mínimo durante el cual el botón debe mantenerse presionado para que la señal se pueda diferenciar en cada caso. En este módulo se utilizó un valor mínimo de 200 ms.

Para mantener la señal activa durante 200 ms se debe tener en cuenta que la tarjeta de desarrollo utilizada para este módulo es la *Nexys 2* que cuenta con un reloj de frecuencia f de 50 MHz y por ende, su periodo T será 20 ns. Sabiendo que un periodo durará 20 ns, se deben tener 10 millones de ciclos antes de volver a evaluar la señal entregada por el botón, para cumplir los 200 ms.



El módulo VHDL desarrollado recibe dos señales de entrada, el pulsador y la señal de reloj, y una salida representando ‘0’ si no hay pulsación y ‘1’ si se presenta pulsación.

Se utiliza un vector como contador, el cual puede almacenar hasta 24 bits, permitiéndonos contar 10 millones de ciclos.

Para eliminar los rebotes, una vez se pasa de tener la señal de salida de 0 a 1, no se tiene en cuenta dicha señal hasta después de 200 ms.

El código desarrollado para el detector de pulso es el siguiente:

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity Detectordepulso is

Port ( Button : in STD\_LOGIC; *--botón*

Clk : in STD\_LOGIC; *--reloj*

S : out STD\_LOGIC); *--respuesta*

end Detectordepulso;

architecture Behavioral of Detectordepulso is

signal cont: std\_logic\_vector (23 downto 0) := (others => '0'); *-- variable contador de ciclos, alcanza 24 bits*

begin

process (Clk)

begin

if rising\_edge(Clk)then *--Si hay un flanco de subida en la señal clk*

S <= '0'; *--Inicializar la señal de salida en 0*

if (Button = '1' or (Button = '0' and cont > 0)) then *--Si el botón fue pulsado o ya se había*

*detectado una pulsación*

if (cont < 10000000) then *--Y si contador<10 millones, no han pasado 200ms*

cont <= cont + 1; *--Aumentar el contador de ciclos*

else *--Si ya pasaron 10 millones de ciclos*

if(Button = '1') then *--Si el botón sigue pulsado después de 10 millones*

S <= '1'; *--Asignar a la señal de salida el valor de 1*

end if;

cont <= (others => '0'); *--Reiniciar el contador en 0*

end if;

end if;

end if;

end process;

end Behavioral;

Para finalizar, se simuló el proceso con el siguiente Test Bench donde se simulan los rebotes del pulsador intercalando entre 0 y 1 la señal Button durante unos milisegundos y después dejándola en 1 por el resto de la pulsación. La señal de salida solo estará en 1 una vez trascurridos 200 ms desde que el pulsador fue activado.

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY Detectordepulso\_TB IS

END Detectordepulso\_TB;

ARCHITECTURE behavior OF Detectordepulso\_TB IS

-- Component Declaration for the Unit Under Test (UUT)

COMPONENT Detectordepulso

PORT(

Button : IN std\_logic;

Clk : IN std\_logic;

S : OUT std\_logic

);

END COMPONENT;

--Inputs

signal Button : std\_logic := '0';

signal Clk : std\_logic := '0';

--Outputs

signal S : std\_logic;

-- Clock period definitions

constant Clk\_period : time := 20 ns;

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: Detectordepulso PORT MAP (

Button => Button,

Clk => Clk,

S => S

);

-- Clock process definitions

Clk\_process :process

begin

Clk <= '0';

wait for Clk\_period/2;

Clk <= '1';

wait for Clk\_period/2;

end process;

-- Stimulus process

stim\_proc: process

begin

-- hold reset state for 100 ns.

wait for 100 ns;

Button <= '0';

wait for 190 ns;

Button <= '1';

wait for 20 ms;

Button <= '0';

wait for 20 ms;

Button <= '1';

wait for 20 ms;

Button <= '0';

wait for 20 ms;

Button <= '1';

wait for 130 ms;

Button <= '0';

wait for 200 ms;

Button <= '1';

wait for 20 ms;

Button <= '0';

wait for 20 ms;

Button <= '1';

wait for 20 ms;

Button <= '0';

wait for 20 ms;

Button <= '1';

wait for 200 ms;

Button <= '0';

wait for 10 ns;

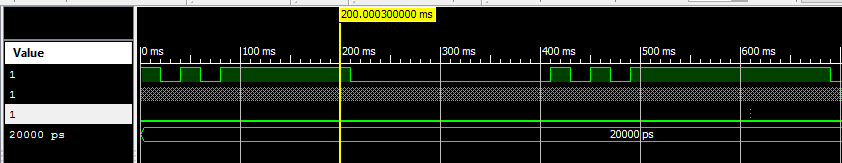
wait for Clk\_period\*10;

-- insert stimulus here

wait;

end process;

END;

Produciendo la siguiente simulación:

**WEBGRAFÍA**

*Circuitos digitales secuenciales I: Resumen del contenido z Estructura de un sistema digital z Latch R/S (The R/S Latch) z Estados ilegales y condiciones. - ppt descargar*. (2017). *Slideplayer.es*. Recuperado 22 de febrero de 2017, de <http://slideplayer.es/slide/1698429/>

*Curso VHDL – Sumador/Restador con operadores aritméticos*. (2017). *Mexchip*. Recuperado 22 de febrero de 2017, de <http://www.mexchip.com/2011/12/curso-vhdl-sumadorrestador-con-operadores-aritmeticos/>

Hernández Stengele, F. (2008). *Diseño y construcción de prototipo neumático de prótesis de pierna humana* (1st ed., pp. 151-155). Cholula, Puebla, México. Recuperado de <http://catarina.udlap.mx/u_dl_a/tales/documentos/lep/hernandez_s_f/capitulo9.pdf>

Noriega, S. (2015). *Introducción al diseño lógico con VHDL* (1st ed.). Recuperado de <https://catedra.ing.unlp.edu.ar/electrotecnia/islyd/Tema%2012b%20Logica%20Programable%20VHDL%202015.pdf>

*Pulsador antirrebote en VHDL (debouncing) – Digilogic*. (2017). *Digilogic.es*. Recuperado 22 de febrero de 2017, de <http://www.digilogic.es/pulsador-antirrebote-en-vhdl-debouncing/>

Sisterna, C. (2017). *Correcto uso de Reset en FPGAs y su Codificación en VHDL* (1st ed.). Recuperado de <http://www.c7t-hdl.com/Docs/C7T_NT10_Reset_VHDL_FPGA.pdf>