Módulo 2: Módulo de control de motores PaP o servomotores.

Lady Johanna Trejos Hernández

Andrés Felipe Betancurt Rivera

Docente Ramiro Andrés Barrios Valencia

Universidad Tecnológica de Pereira

Facultad de Ingenierías FEECC

Asignatura Laboratorio de Electrónica Digital

22 de febrero de 2017

Un *motor paso a paso*, como todo motor, es en esencia un conversor electromecánico, que transforma energía eléctrica en mecánica. Mientras que un motor convencional gira libremente al aplicarle una tensión, el motor paso a paso gira un determinado ángulo de forma incremental (transforma impulsos eléctricos en movimientos de giro controlados), lo que le permite realizar desplazamientos angulares fijos muy precisos (pueden variar desde 1.80º hasta unos 90º). (Universidad Politécnica de Valencia, 2002)

Otro tipo de motores importantes debido a su precisión, son los *servomotores* también conocidos como *servos*, son motores de corriente continua capaces de moverse en un rango generalmente de 180° y mantenerse en determinada posición. Los servos contienen tres cables: dos de alimentación (positivo y negativo) y un cable de control que indica la posición deseada mediante señales PWD (“*Pulse Width Modulation”*), formadas por pulsos positivos cuya duración es proporcional a la posición deseada del servo y que se repiten cada 20ms.

Debido a la precisión y facilidad de manejo, se decidió trabajar este módulo con motores paso a paso. Puesto que en el proyecto final se necesitará torque para mover los ejes con fuerza y velocidad, estos motores serán de gran utilidad. También al momento de detenerse, cuentan con torque de detención que hace que el motor se mantenga firmemente en su posición cuando no está girando.

Si bien es cierto que los motores paso a paso funcionan controlados por un pulso de avance, el control de un motor paso a paso no se realiza aplicando en directo este pulso eléctrico que lo hace avanzar. Estos motores tienen varios bobinados que, para producir el avance de ese paso, deben ser alimentados en una adecuada secuencia. Si se invierte el orden de esta secuencia, se logra que el motor gire en sentido opuesto. ("Motores paso a paso - Características básicas - Robots Argentina", 2017)

Teniendo en cuenta que se pueden producir corrientes basura, es necesario añadir un circuito de protección como el diodo. Los motores suelen funcionar con 9 V – 12 V y los circuitos de control con 5 V, para aumentar este voltaje se puede utilizar un *transistor* que funciona como fuente de control también.

Se desarrolló el código en VHDL para un motor paso a paso con frecuencia máxima de operación de 100 ms, por ende, se implementa un contador de flancos de subida para determinar cuándo han pasado exactamente 100 ms, simulando un divisor de frecuencia. Mientras el contador se encuentre funcionando, no se tendrán en cuenta otros cambios de paso.

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity Motor\_a\_pasos is

Port ( StepDrive : out std\_logic\_vector(3 downto 0);

Direction : in std\_logic;

StepEnable : in std\_logic;

CLK : in std\_logic);

end Motor\_a\_pasos;

*--Direction: indica la dirección a la que va a girar el motor*

*--StepEnable: switch que activa o no el movimiento del motor*

*--CLK: para llevar un conteo de pasos*

*--StepDrive: las salidas de los 4 pines.*

architecture Behavioral of Motor\_a\_pasos is

*--Variables temporales:*

signal aux : std\_logic\_vector (3 downto 0) := "0000";

signal state : std\_logic\_vector(1 downto 0) := "00";

signal StepCounter : std\_logic\_vector(31 downto 0) := (others => '0');

constant StepLockOut : std\_logic\_vector(31 downto 0) := " 00000000000001111010000100100000";

begin

*--State: los posibles estados del motor.*

*--StepCounter: contador que aumenta cada vez que encuentra un flanco de subida en la señal de reloj.*

*--StepLockOut: indica la frecuencia a la cual el motor va a dar cada paso.*

StepDrive <= aux;

process(CLK)

begin

if ((CLK'event) and (CLK='1')) then *--Esto indica que cada vez que el reloj*

StepCounter <= StepCounter + 1; *-- este en frente de subida se le*

*--aumentará en 1 a StepCounter*

if (StepCounter >= StepLockOut) then *--Se resetea el contador*

StepCounter <= (others => '0'); *--si es mayor a la frecuencia*

aux <= "1111";

if (StepEnable = '1') then *--Habilitador activado*

if (Direction = '1') then state <= state + "01"; end if;

if (Direction = '0') then state <= state - "01"; end if;

case state is *--Determina hacia dónde va a girar*

when "00" =>aux <= "1000";

when "01" =>aux <= "0100";

when "10" =>aux <= "0010";

when "11" =>aux <= "0001";

when others => end case;

end if;

end if;

end if;

end process;

end Behavioral;

Para la simulación en el Test Bench se plantearon tres casos, en los primeros 300 ms se va a tener el habilitador en 1 y la dirección será 1 (derecha), en los siguientes 400 ms se tendrá el habilitador en 1 y la dirección en 0 (izquierda), y en el resto de la simulación se tendrá el habilitador en 0, para un total de 1 segundo de simulación.

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY Motorapasos\_tb IS

END Motorapasos\_tb;

ARCHITECTURE behavior OF Motorapasos\_tb IS

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

COMPONENT Motor\_a\_pasos

PORT(

StepDrive : OUT std\_logic\_vector(3 downto 0);

Direction : IN std\_logic;

StepEnable : IN std\_logic;

CLK : IN std\_logic

);

END COMPONENT;

*--Inputs*

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

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

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

*--Outputs*

signal StepDrive : std\_logic\_vector(3 downto 0);

*-- Clock period definitions*

constant CLK\_period : time := 20 ns;

BEGIN

*-- Instantiate the Unit Under Test (UUT)*

uut: Motor\_a\_pasos PORT MAP (

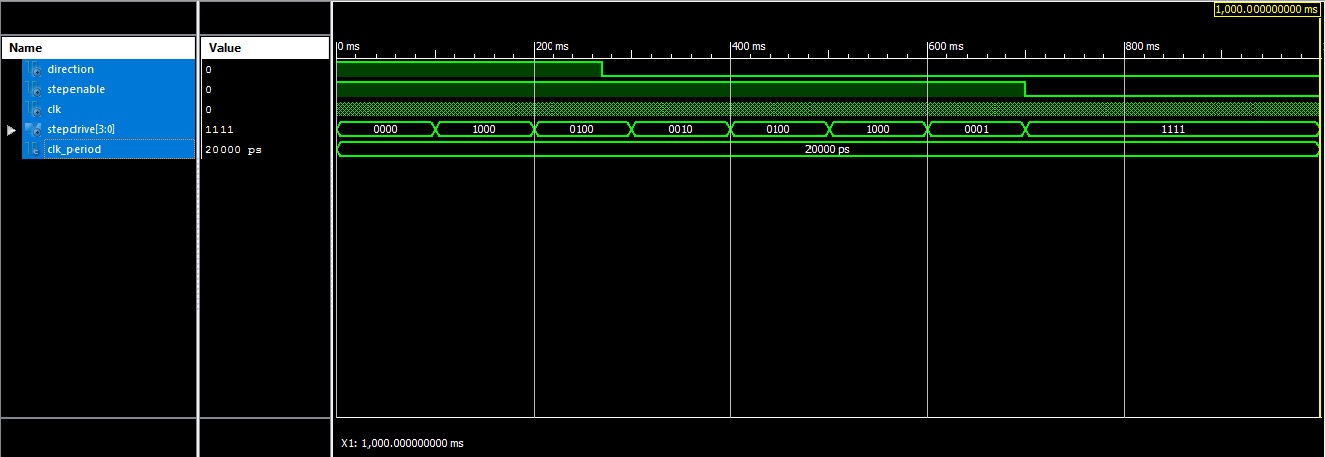
StepDrive => StepDrive,

Direction => Direction,

StepEnable => StepEnable,

CLK => CLK

);



CLK\_process :process  *-- Clock process definitions*

begin

CLK <= '0';

wait for CLK\_period/2;

CLK <= '1';

wait for CLK\_period/2;

end process;

stim\_proc: process *-- Stimulus process*

begin

StepEnable <= '1';

direction <= '1'; *--Girar a la derecha*

wait for 270 ms;

direction <= '0'; *--Girar a la izquierda*

wait for 430 ms;

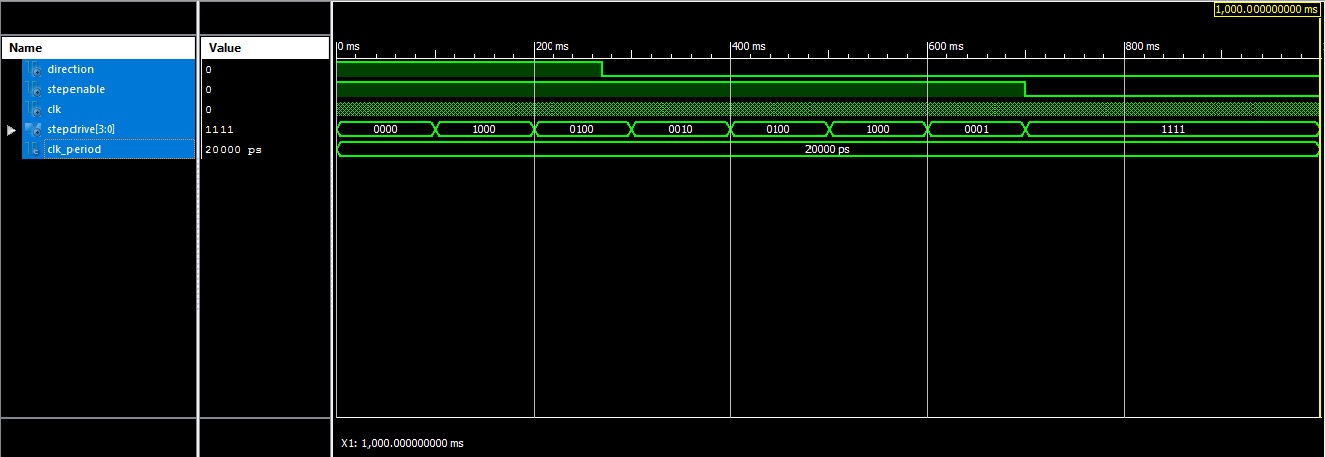
StepEnable <= '0'; *--Apagar habilitador*

--wait for CLK\_period\*10;

wait;

end process;

END;



**WEBGRAFÍA**

Aguilar Vega, V. (2015). *Controlador de Motor Paso a Paso con VHDL*. *prezi.com*. Recuperado el 7 de marzo de 2017, de <https://prezi.com/uvbzrdzbrc_-/controlador-de-motor-paso-a-paso-con-vhdl/>

*CONTROL DE MOTORES PASO A PASO MEDIANTE MICROCONTROLADORES*. (2002). *Universidad Politécnica de Valencia*. Recuperado el 7 de marzo de 2017, de <http://server-die.alc.upv.es/asignaturas/lsed/2002-03/MotoresPasoaPaso/Motorespasoapaso.pdf>

Candelas Herías, F. & Corrales Ramón, J. (2007). *Servomotores*. *Universidad de Alicante*. Recuperado el 7 de marzo de 2017, de http://www.aurova.ua.es/previo/dpi2005/docs/publicaciones/pub09-ServoMotores/servos.pdf

*Diseño y simulación de un circuito integrado para el control de un motor PAP, utilizando programación VHDL*. (2006). *Universidad Nacional de Ingeniería*. Recuperado el 7 de marzo de 2017, de <http://www.bibliotecacentral.uni.edu.pe/pdfs/TECNIA/2,2006/art_006.pdf>

López Tello, L. (2010). *Motor a pasos controlado con tarjeta spartan 3*. *Lopeztel.blogspot.com.co*. Recuperado el 7 de marzo de 2017, de <http://lopeztel.blogspot.com.co/2010/11/motor-pasos-controlado-con-tarjeta.html>

*Motores paso a paso*. *Platea.pntic.mec.es*. Recuperado el 7 de marzo de 2017, de <http://platea.pntic.mec.es/vgonzale/cyr_0204/cyr_01/robotica/sistema/motores_p-p.htm>

Posadas, S. (2015). *Motores paso a paso (II): Criterios selección motores y drivers*. *Impresión 3D Prototipado | 3D Printing Prototyping | Dima 3D*. Recuperado el 7 de marzo de 2017, de <http://www.dima3d.com/motores-paso-a-paso-en-impresion-3d-ii-criterios-de-seleccion-de-motores-y-drivers/>

*Motores paso a paso - Características básicas - Robots Argentina*. (2017). *Robots-argentina.com.ar*. Recuperado el 7 de marzo de 2017, de <http://robots-argentina.com.ar/MotorPP_basico.htm>