**Objetivo**

El alumno deberá realizar las modificaciones pertinentes para poder girar el motor las vueltas necesarias que representen los dos últimos dígitos de su número de cuenta; se deben combinar los giros horario, antihorario y detenido.

**Introducción**

Para la realización de esta práctica igual que en las practicas anteriores requerimos de diferentes conocimientos adquiridos tanto en materias pasadas como en la materia de Diseño Digital VLSI. Algunos de los conocimientos requeridos para la realización correcta de esta practican son: conocimiento fundamental sobre cómo funcionan las FPGA, su arquitectura y cómo programarlas utilizando el lenguaje VHDL. En este caso en particular se hará uso de la herramienta de desarrollo FPGA Intel Quartus, para escribir, compilar y cargar el código en la tarjeta FPGA.

También será de gran importancia tener conocimiento sobre algunos de los principios de control de motores, en esta práctica en específico se trabajará con un motor a pasos por lo que será importante conocer cómo controlarlos mediante señales de control apropiadas. Para esta práctica usaremos un driver del motor a pasos para evitar algún daño en la tarjeta. En la realización del código que nos ayudara a controlar el motor de la tarjeta deberemos conocer cómo conectar y comunicarte con el motor desde la FPGA, lo que implica comprender los protocolos de comunicación y los pines de entrada/salida (E/S) necesarios.

Se necesitará también calcular la cantidad de vueltas necesarias que representen los dos últimos dígitos del número de cuenta y cómo traducir ese valor en instrucciones de control para el motor. Además de comprender cómo generar secuencias de pasos y controlar la velocidad del motor para lograr giros horario, antihorario y detenido según sea necesario.

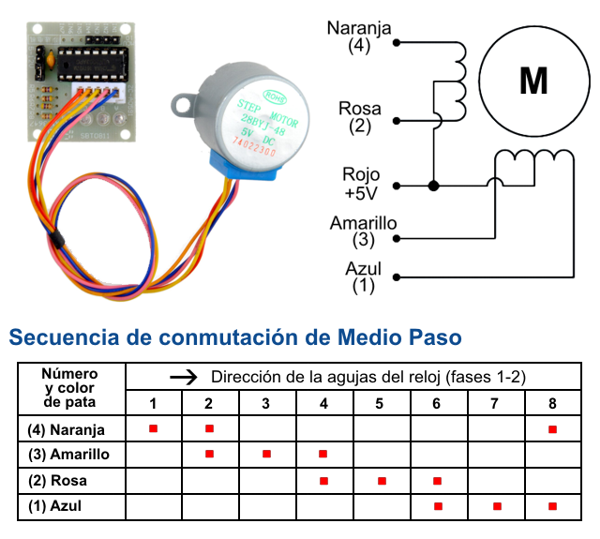
**Desarrollo**

El alumno deberá realizar las modificaciones pertinentes para poder girar el motor las vueltas necesarias que representen los dígitos de su número de cuenta se deben combinar los giros horario, antihorario y detenido.

Primero debemos contar el material a utilizar, ya que algunos motores tienen características diferentes. Para el desarrollo de la práctica se utilizó un motor a pasos 28byj-48 y un driver ULN2003, donde cabe resaltar la importancia del driver con el cual podemos suministrar energía al motor de forma más estable y evitar daño en la FPGA, asimismo el driver nos ayudara a proporcionar las entradas necesarias al motor para que este realice los movimientos correspondientes.



El driver para controlar el motor a pasos y el motor a pasos funciona de la siguiente manera:



Para controlar un motor paso a paso con el controlador ULN2003, primero debemos conectar el motor al controlador. El motor tiene cuatro bobinas, que están identificadas por las letras A, B, C y D. Las bobinas A y B forman una fase, las bobinas C y D forman otra fase, y las bobinas A y D forman una tercera fase.

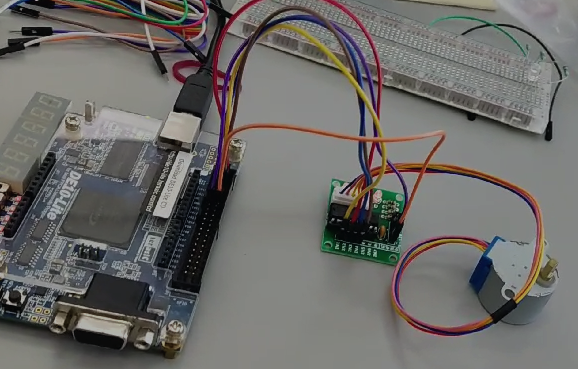
Cuando el motor está conectado al controlador, podemos enviar pulsos eléctricos a las bobinas. Para hacer que el motor gire en un sentido, enviaremos pulsos eléctricos a las bobinas en la secuencia correcta.

En el siguiente esquema podemos ver el controlador, donde se conecta el motor a pasos y las entradas están marcadas como IN1, IN2..., para alimentar el controlador se hará desde la entrada de la derecha, donde un pin recibe GND (tierra) y el otro pin recibe de 5 a 12 V, por lo que usaremos los pines de suministro de energía de la tarjeta, respectivamente GND y 5V.



Controlador ULN2003

Por lo que el esquema de conexión final quedaría de la siguiente manera:



El código que usaremos para controlar el motor a pasos es el siguiente: Donde utilizaremos una señal std\_logic para controlar de manera automática el giro del motor, hemos declarado esta señal para que genere un contador interno que ira aumentando cada ciclo de trabajo del reloj donde realice una vuelta para que de manera automática se detenga. El indicador será que se prendera un led en la tarjeta para que este pueda indicar que ha acabado el proceso. Este se puede modificar en la siguiente linea: if (vueltas >= 79) then donde el numero de vueltas se compara con el estado actual y este se para de forma automatica y prende el indicador led.

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity MotPasos is

Port (reloj: in STD\_LOGIC;

UD: in STD\_LOGIC;

rst: in STD\_LOGIC;

FH: in STD\_LOGIC\_VECTOR(1 downto 0);

led: out STD\_LOGIC\_VECTOR(3 downto 0);

MOT: out STD\_LOGIC\_VECTOR(3 downto 0);

stop\_motor: out STD\_LOGIC -- Señal para detener el motor cuando se alcanzan 79 vueltas

);

end MotPasos;

architecture Behavioral of MotPasos is

signal div: STD\_LOGIC\_VECTOR (17 downto 0);

signal clks: STD\_LOGIC;

type estado is (sm0, sm1, sm2, sm3, sm4, sm5, sm6, sm7, sm8, sm9, sm10);

signal pres\_s, next\_s: estado;

signal motor: STD\_LOGIC\_VECTOR (3 downto 0);

signal vueltas: INTEGER := 0; -- Contador de vueltas

begin

process(reloj, rst)

begin

if (rst = '0') then

div <= (others => '0');

elsif (reloj'event and reloj = '1') then

div <= div + 1;

end if;

end process;

clks <= div(17);

process(clks, rst)

begin

if (rst = '0') then

pres\_s <= sm0;

elsif (clks'event and clks = '1') then

pres\_s <= next\_s;

end if;

end process;

process(pres\_s, FH, UD)

begin

case (pres\_s) is

when sm0 => --estado 0

next\_s <= sm1;

-- Incrementar el contador de vueltas cuando vuelva a sm0

if (pres\_s = sm0) then

vueltas <= vueltas + 1;

end if;

-- Detener el motor cuando se alcanzan 79 vueltas

if (vueltas >= 79) then

stop\_motor <= '1';

else

stop\_motor <= '0';

end if;

when sm1 => --estado 1

if (FH = "00") then --motor bipolar

if (UD = '1') then

next\_s <= sm3;

else

next\_s <= sm7;

end if;

elsif (FH = "01") then

if (UD = '1') then

next\_s <= sm2;

else

next\_s <= sm8;

end if;

elsif (FH = "10") then

if (UD = '1') then

next\_s <= sm2;

else

next\_s <= sm8;

end if;

elsif (FH = "11") then

if (UD = '1') then

next\_s <= sm9;

else

next\_s <= sm4;

end if;

else

next\_s <= sm1;

end if;

when sm2 => --estado 2

if (FH="00")then --motor bipolar

if (UD='1')then

next\_s<=sm1;

else

next\_s<=sm7;

end if;

elsif(FH="01")then

if(UD='1')then

next\_s<=sm4;

else

next\_s<=sm8;

end if;

elsif(FH="10")then

if(UD='1')then

next\_s<=sm3;

else

next\_s<=sm1;

end if;

elsif(FH="11")then

if(UD='1')then

next\_s<=sm9;

else

next\_s<=sm4;

end if;

else

next\_s<=sm2;

end if;

when sm3 => --estado 3

if (FH="00")then --motor bipolar

if (UD='1')then

next\_s<=sm5;

else

next\_s<=sm1;

end if;

elsif(FH="01")then

if(UD='1')then

next\_s<=sm2;

else

next\_s<=sm8;

end if;

elsif(FH="10")then

if(UD='1')then

next\_s<=sm4;

else

next\_s<=sm2;

end if;

elsif(FH="11")then

if(UD='1')then

next\_s<=sm9;

else

next\_s<=sm4;

end if;

else

next\_s<=sm3;

end if;

when sm4 => --estado 4

if (FH="00")then --motor bipolar

if (UD='1')then

next\_s<=sm1;

else

next\_s<=sm7;

end if;

elsif(FH="01")then

if(UD='1')then

next\_s<=sm6;

else

next\_s<=sm2;

end if;

elsif(FH="10")then

if(UD='1')then

next\_s<=sm5;

else

next\_s<=sm3;

end if;

elsif(FH="11")then

if(UD='1')then

next\_s<=sm9;

else

next\_s<=sm10;

end if;

else

next\_s<=sm4;

end if;

when sm5 => --estado 5

if (FH="00")then --motor bipolar

if (UD='1')then

next\_s<=sm7;

else

next\_s<=sm3;

end if;

elsif(FH="01")then

if(UD='1')then

next\_s<=sm2;

else

next\_s<=sm8;

end if;

elsif(FH="10")then

if(UD='1')then

next\_s<=sm6;

else

next\_s<=sm4;

end if;

elsif(FH="11")then

if(UD='1')then

next\_s<=sm9;

else

next\_s<=sm4;

end if;

else

next\_s<=sm3;

end if;

when sm6 => --estado 6

if (FH="00")then --motor bipolar

if (UD='1')then

next\_s<=sm1;

else

next\_s<=sm7;

end if;

elsif(FH="01")then

if(UD='1')then

next\_s<=sm8;

else

next\_s<=sm4;

end if;

elsif(FH="10")then

if(UD='1')then

next\_s<=sm7;

else

next\_s<=sm5;

end if;

elsif(FH="11")then

if(UD='1')then

next\_s<=sm9;

else

next\_s<=sm4;

end if;

else

next\_s<=sm7;

end if;

when sm7 => --estado 7

if (FH="00")then --motor bipolar

if (UD='1')then

next\_s<=sm1;

else

next\_s<=sm5;

end if;

elsif(FH="01")then

if(UD='1')then

next\_s<=sm2;

else

next\_s<=sm8;

end if;

elsif(FH="10")then

if(UD='1')then

next\_s<=sm8;

else

next\_s<=sm6;

end if;

elsif(FH="11")then

if(UD='1')then

next\_s<=sm9;

else

next\_s<=sm4;

end if;

else

next\_s<=sm7;

end if;

when sm8 => --estado 8

if (FH="00")then --motor bipolar

if (UD='1')then

next\_s<=sm1;

else

next\_s<=sm7;

end if;

elsif(FH="01")then

if(UD='1')then

next\_s<=sm2;

else

next\_s<=sm6;

end if;

elsif(FH="10")then

if(UD='1')then

next\_s<=sm1;

else

next\_s<=sm7;

end if;

elsif(FH="11")then

if(UD='1')then

next\_s<=sm10;

else

next\_s<=sm8;

end if;

else

next\_s<=sm1;

end if;

when sm9 => --estado 9

if (FH="00")then --motor bipolar

if (UD='1')then

next\_s<=sm1;

else

next\_s<=sm7;

end if;

elsif(FH="01")then

if(UD='1')then

next\_s<=sm2;

else

next\_s<=sm8;

end if;

elsif(FH="10")then

if(UD='1')then

next\_s<=sm1;

else

next\_s<=sm8;

end if;

elsif(FH="11")then

if(UD='1')then

next\_s<=sm8;

else

next\_s<=sm9;

end if;

else

next\_s<=sm1;

end if;

when sm10 => --estado 10

if (FH="00")then --motor bipolar

if (UD='1')then

next\_s<=sm1;

else

next\_s<=sm7;

end if;

elsif(FH="01")then

if(UD='1')then

next\_s<=sm2;

else

next\_s<=sm8;

end if;

elsif(FH="10")then

if(UD='1')then

next\_s<=sm1;

else

next\_s<=sm8;

end if;

elsif(FH="11")then

if(UD='1')then

next\_s<=sm4;

else

next\_s<=sm8;

end if;

else

next\_s<=sm10;

end if;

when others=>

next\_s<=sm0;

end case;

end process;

-- Código de salidas de estados al motor

process(pres\_s)

begin

case (pres\_s) is

when sm0 => motor <= "0000";

when sm1 => motor <= "1000";

when sm2 => motor <= "1100";

when sm3 => motor <= "0100";

when sm4 => motor <= "0110";

when sm5 => motor <= "0010";

when sm6 => motor <= "0011";

when sm7 => motor <= "0001";

when sm8 => motor <= "1010";

when sm9 => motor <= "0101";

when sm10 => motor <= "0000";

when others => motor <= "0000";

end case;

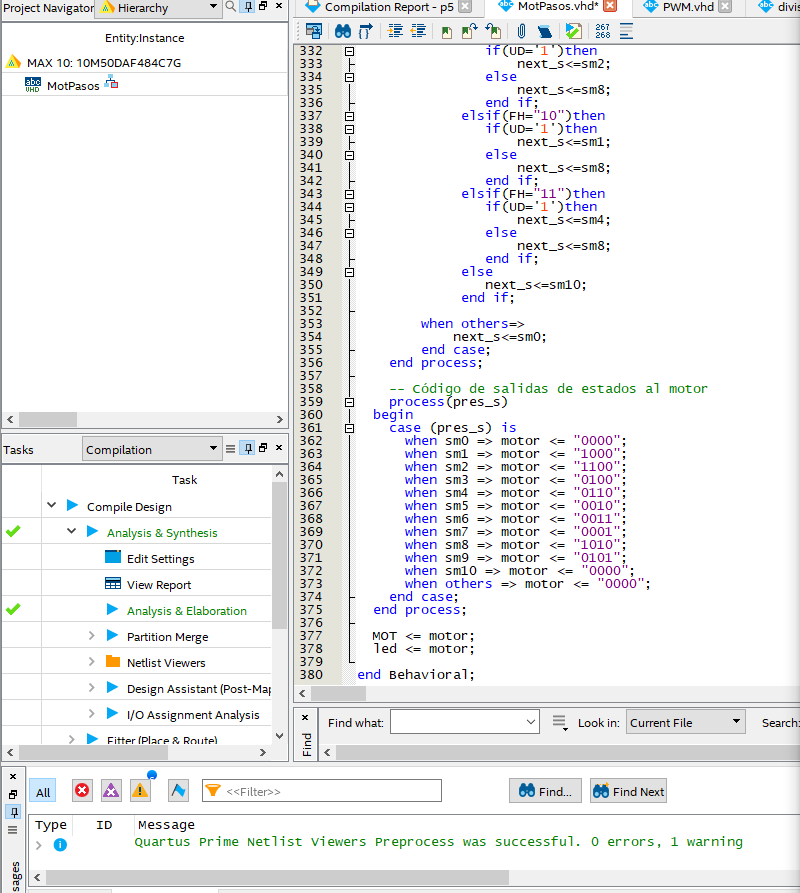
end process;

MOT <= motor;

led <= motor;

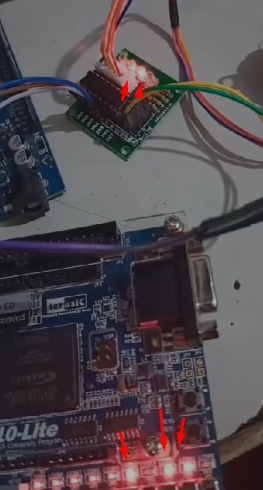
end Behavioral;

Prueba de compilación y ejecución:

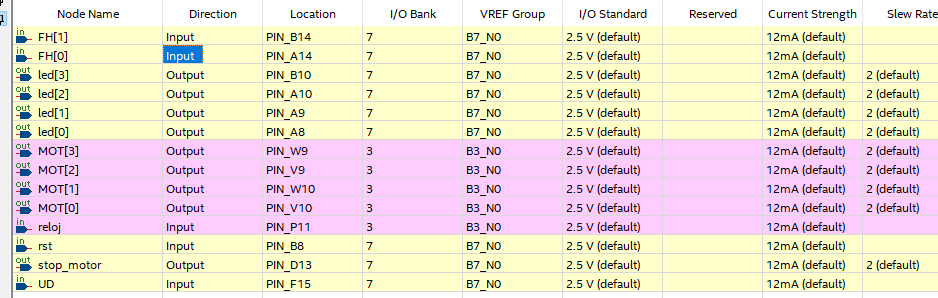
* 

La configuración de pines es la siguiente:

* Los estados FH (1) y (0) ayudan a controlar los estados del motor
* Las salidas Led (3,2,1,0) nos mostrara el estado y los pasos de motor que se encontraran trabajando, como hemos asignado un reloj de 50Mhz no se alcanza a percibir de manera adecuada, pero al usarlo con uno de 10 MHz se logra percibir como en la siguiente imagen:

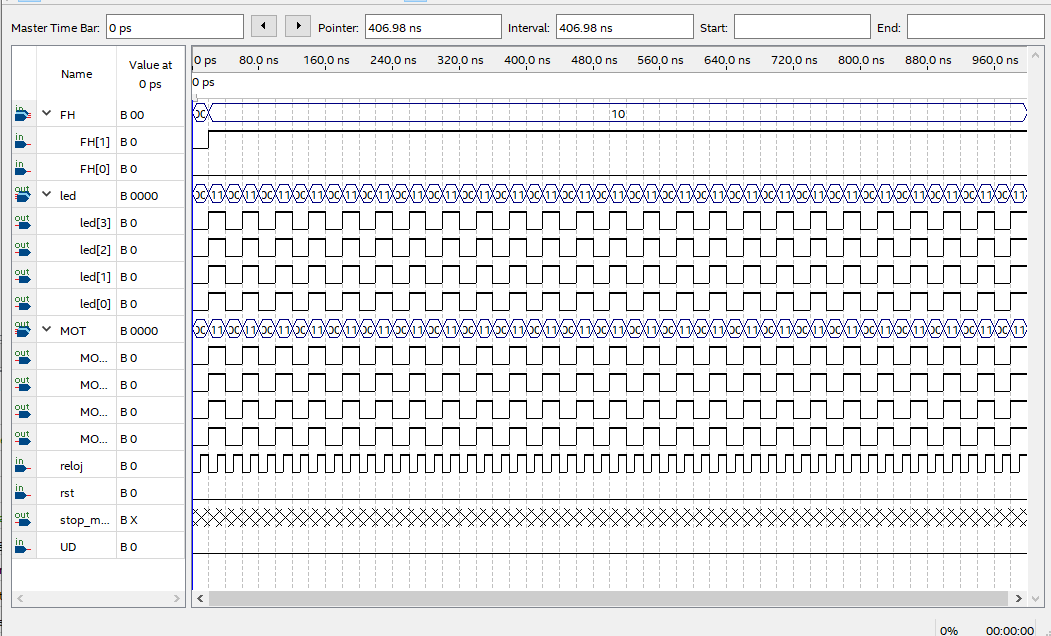


* Las salidas de MOT son las encargadas de mandar las señales de mandar los impulsos al motor de pasos
* EL RST es un boton que asignamos para poner en estado resset el programa
* STOP\_MOTOR es una salida que nos indicara que el ciclo ha terminado
* UD: Es la encargada de controlar el sentido horario del motor

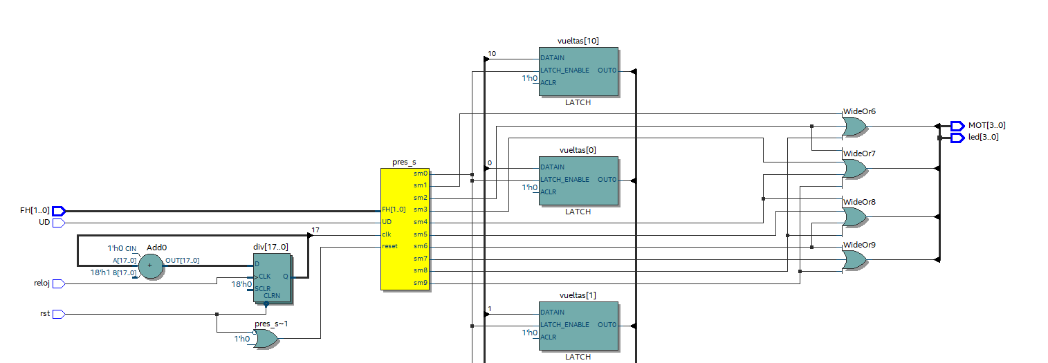


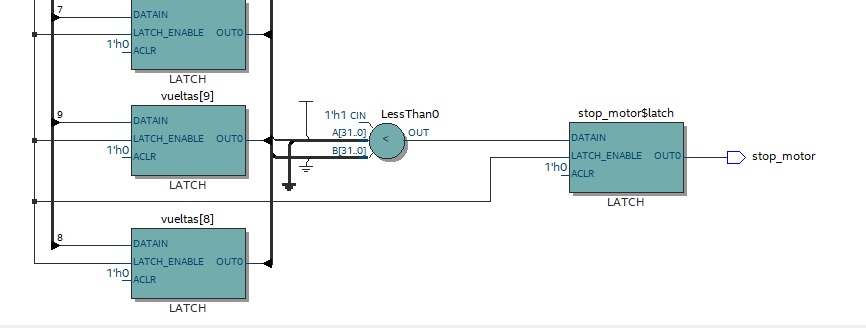
Realizando la simulación:

Obtenemos los siguiente, donde podemos ver que la respuesta va conforme al ciclo de reloj, en este caso no se aplicó ni una respuesta de resset por lo que sale tachado toda la línea.



El RTL del circuito es el siguiente donde se muestra de manera lógica el diagrama de conexión donde se implementó el stop para el motor a pasos, se cortan los estados de las vueltas porque son demasiados.





Los videos correspondientes a las vueltas de cada integrante del equipo son los siguientes. Asimismo, colocamos los 2 últimos dígitos de cada integrante para que se sepa el número de vueltas de cada uno:

**Bibliografía**

* Rosero, D. (2022, 23 de enero). Implementación de un motor paso a paso en FPGA. YouTube. [Video].
* Ecarletti. (s. f.). *Arduino: Motor paso a paso 28BYJ-48 y Módulo ULN2003 | Robots didácticos*. <https://robots-argentina.com.ar/didactica/arduino-motor-paso-a-paso-28byj-48-y-modulo-uln2003/>
* Terasic. (2017). DE10-Lite User Manual. <https://www.terasic.com.tw/cgi-bin/page/archive_download.pl?Language=China&No=1021&FID=a13a2782811152b477e60203d34b1baa>