Universidad Tecnológica Nacional

Facultad Regional Buenos Aires

Cátedra: Robótica - Plan 95A

Trabajo práctico Nº2

Análisis Dinámico de un robot e implementación en FPGA

Profesor: Ing. Hernan Giannetta.

JTP: Ing. Damián Granzella.

Integrantes: Juan Pablo Perelló. Leg (132651-0)

Nicolás Pimentel. Leg (119137-8)

Fecha de entrega: 14 / 05 / 2010.

Introducción Teórica:

Dinámica del robot

La dinámica se ocupa de la relación entre las fuerzas que actúan sobre un cuerpo y el movimiento que él origina. Por lo tanto, el modelo dinámico de un robot tiene por objetivo conocer la relación entre el movimiento del robot y las fuerzas implicadas en el mismo.

Esta relación se obtiene mediante el denominado **modelo dinámico**, que relaciona matemáticamente:

1) La localización del robot definida por sus variables articulares o por las coordenadas de localización de su extremo, y sus derivadas: velocidad y aceleración.

2) Las fuerzas y pares aplicados en las articulaciones (o en el extremo del robot).

3) Los parámetros dimensionales del robot, como longitud, masas e inercias de sus elementos.

La obtención de este modelo para mecanismos de uno o dos grados de libertad no es excesivamente compleja, pero a medida que el número de grados de libertad aumenta, el planeamiento y obtención del modelo dinámico se complica enormemente. Por este motivo no siempre es posible obtener un modelo dinámico expresado de una forma cerrada, esto es, mediante una serie de ecuaciones, normalmente de tipo diferencial de 2^{do} orden, cuya integración permite conocer que movimiento surge al aplicar unas fuerzas o qué fuerzas hay que aplicar para obtener un movimiento determinado. El modelo dinámico debe ser resuelto entonces de manera iterativa mediante la utilización de un procedimiento numérico.

El problema de la obtención del modelo de un robot es, por lo tanto, uno de los aspectos más complejos de la robótica, lo que ha llevado a ser obviado en numerosas ocasiones. Sin embargo, el modelo dinámico es imprescindible para conseguir los siguientes fines:

1) Simulación del movimiento del robot.

2) Diseño y evaluación de la estructura mecánica del robot.

3) Dimensionamiento de los actuadores.

4) Diseño y evaluación del control dinámico del robot.

Este último fin es evidentemente de gran importancia, pues de la calidad del control dinámico del robot depende la precisión y velocidad de sus movimientos. La gran complejidad ya comentada existente de la obtención del modelo dinámico del robot, ha motivado que se realicen ciertas simplificaciones, de manera que así pueda ser utilizado en el diseño del controlador.

Es importante hacer notar que el modelo dinámico completo de un robot debe incluir no sólo la dinámica de sus elementos (barras o eslabones) sino también la propia de sus sistemas de transmisión, de los actuadores y sus equipos electrónicos de mando. Estos elementos incorporan al modelo dinámico nuevas inercias, rozamientos, saturaciones de los circuitos electrónicos, etc. aumentando aún más su complejidad.

Por último, es preciso señalar que si bien en la mayor parte de las aplicaciones reales de la robótica, las cargas e inercias manejadas no son suficientes como para originar deformaciones en los eslabones del robot, en determinadas ocasiones no ocurre así, siendo preciso considerar al robot como un conjunto de eslabones no rígidos. Aplicaciones de este tipo pueden encontrarse en la robótica espacial o en robots de grandes dimensiones, entre otras.

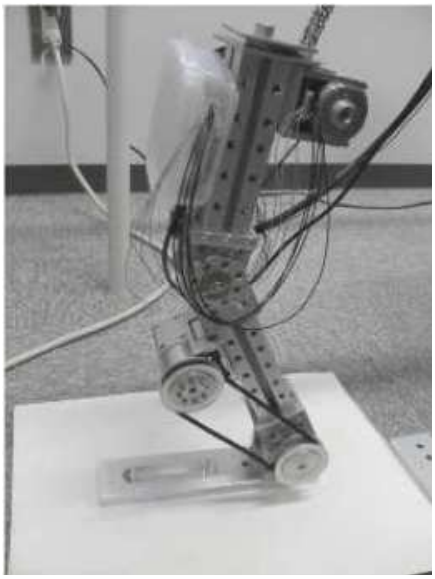
En nuestro trabajo práctico vamos a realizar el análisis dinámico de la estructura mecánica del **“One Legged Robot”.** Vamos a llevar a cabo el análisis dinámico del robot, complementando el análisis cinemático que hicimos en el trabajo practico anterior.

En el presente trabajo se hará el análisis utilizando las formulaciones lagrangianas por ser una herramienta muy eficaz a medida que aumentan los grados de libertad. Es importante señalar que existen otras formulaciones también validas como las newtonianas y variantes entre estas dos que se han ido adaptando para obtener una mejor implementación computacional.

Plantear este análisis complementa el estudio del comportamiento del robot para diseñar las etapas de control del mismo. Para la generación de las trayectorias se implementara un control mediante modulación de ancho de pulso (PWM), para el cual se utilizara un lenguaje de descripción de hardware (VHDL) y será implementado sobre un FPGA.

Para terminar se presentarán los resultados de la simulación, los cuales permitirán analizar el comportamiento del robot y los sistemas mecánicos ante las trayectorias propuestas.

Nuestro Robot (One Legged Robot)



El One Legged Robot representa el estudio necesario para implementar los Bipedrobots.

Este tipo de robots se utilizan para interactuar en el ambiente humano. De todos los robots móviles, los legged robot tienen ventajas en esquivar obstáculos, subir peldaños y velocidades de movimientos similares a las humanas.

En la figura podemos observar un robot experimental del tipo One Legged Robot.

A continuación colocamos un esquema representativo del modelo dinámico del robot

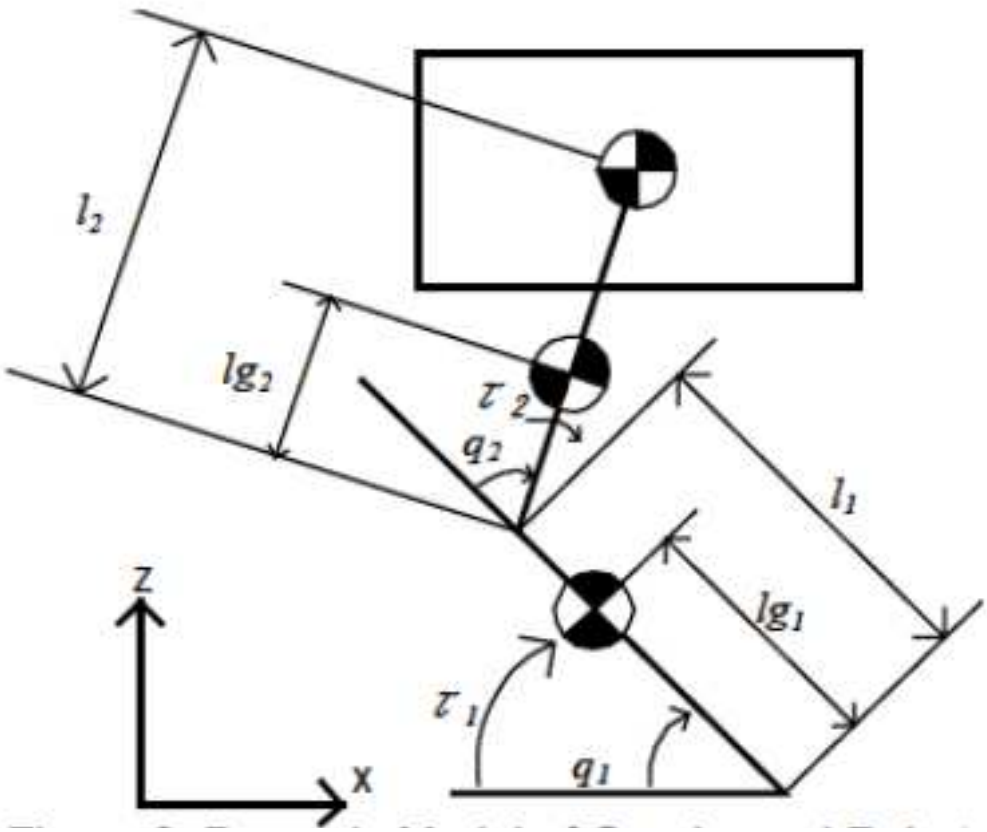


Figura 1. Modelo Dinámico del One Legged Robot.

Solo se considerara el estudio del movimiento en los planos x-z.

En la tabla siguiente se muestran los parámetros del robot y de cada link.

Mass(kg)	1.336(kg)
Width	0.18(m)
Depth	0.04(m)
Height	0.36(m)
l_0 (Body link)	0.18(m)
l_1 (2nd thigh length)	0.18(m)
l_2 (thigh length)	0.13(m)
m_0 (body mass)	0.235(kg)
m_1 (2nd thigh mass)	0.465(kg)
m_2 (thigh mass)	0.450(kg)
M_p (frame mass)	0.185(kg)
I_1	2nd thigh inertia
I_2	thigh inertia
lg_1	gravity point of 2nd thigh
lg_2	gravity point of thigh

Tabla 1. Especificaciones del On Legged Robot

La dinámica del robot de 2DOF de la figura 1 se obtuvo mediante el enfoque energético de Lagrange-Euler y se plantea de la siguiente manera:

$$M(q)\ddot{q} + C(q,\dot{q}) + G(q) = \tau$$

Donde:

τ: Vector de fuerzas y pares motores efectivos aplicados sobre cada coordenada qi.

$M(q)$: Matriz de Inercias.

$C(q,\dot{q})$: Matriz columna de fuerzas de Coriolis y Centrípeta.

$G(q)$: Matriz columna de fuerzas de gravedad.

$$M(q)=\begin{bmatrix} m_1 l_{g1}{}^2 + I_1 + m_2(l_1{}^2 + l_{g2}{}^2 + 2l_1 l_{g2} c2) + I_2 & m_2(l_{g2}{}^2 + l_1 l_{g2} c2) + I_2 \\ m_2(l_{g2}{}^2 + l_1 l_{g2} c2) + I_2 & m_2 l_{g2}{}^2 + I_2 \end{bmatrix}$$

$$C(q,\dot{q})=\begin{bmatrix} -2m_2 l_1 l_{g2} S_2 \dot{q}_1 \dot{q}_2 - m_2 l_1 l_{g2} S_2 \dot{q}_2{}^2 \\ m_2(l_{g2}{}^2 + l_1 l_{g2} \dot{q}_1 \dot{q}_2) \end{bmatrix}$$

$$G(q)=\begin{bmatrix} m_1 g l_{g1} c_1 + m_2 g(l_1 c_1 + l_{g2} c_{12}) \\ m_2 g l_{g2} c_{12} \end{bmatrix}$$

Referencias	
S_1	sin(q1)
S_{12}	sin(q1+q2
C_1	cos(q1)
C_{12}	cos(q1+q2
g	9,81m/s^2

$M_{11} = m_1 l_{g1}{}^2 + I_1 + m_2(l_1{}^2 + l_{g2}{}^2 + 2l_1 l_{g2} c2) + I_2$	$M_{12} = m_2(l_{g2}{}^2 + l_1 l_{g2} c2) + I_2$
$M_{21} = m_2(l_{g2}{}^2 + l_1 l_{g2} c2) + I_2$	$M_{22} = m_2 l_{g2}{}^2 + I_2$

$$\begin{bmatrix} \tau 1 \\ \tau 2 \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \cdot \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} C_a \\ C_b \end{bmatrix} + \begin{bmatrix} G_1 \\ G_2 \end{bmatrix}$$

Hacemos una estimación de los valores de las distancias l_{g1}, l_{g2} basándonos en la figura1 del robot utilizado:

l_{g1}=0,14m.

l_{g2}=0,07m.

Calculamos los momentos de inercia de cada link suponiendo que los brazos son de densidad uniforme y el eje de rotación pasa por el extremo ($I=⅓.m.L^2$):

$I_1=⅓.m_1.l_1{}^2 = ⅓.0,465Kg.(0,18m)^2 = 0,005022Kg.m^2$.

$I_2 = ⅓.m_2.l_2{}^2 = ⅓.0,450Kg.(0,13m)^2 = 0,002535Kg.m^2$.

$$\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \cdot \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} = \begin{bmatrix} m_1 l_{g1}{}^2 + I_1 + m_2(l_1{}^2 + l_{g2}{}^2 + 2l_1 l_{g2} c2) + I_2 & m_2(l_{g2}{}^2 + l_1 l_{g2} c2) + I_2 \\ m_2(l_{g2}{}^2 + l_1 l_{g2} c2) + I_2 & m_2 l_{g2}{}^2 + I_2 \end{bmatrix} \cdot \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix}$$

$$\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \cdot \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} = \begin{bmatrix} (0,33456 + 0,0252c_2)Kg.m^2 & (0,00474 + 0,0252c_2)Kg.m^2 \\ (0,00474 + 0,0252c_2)Kg.m^2 & 0,00474Kg.m^2 \end{bmatrix} \cdot \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix}$$

$$\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \cdot \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} = \begin{bmatrix} (0,33456 + 0,0252c_2)Kg.m^2.\ddot{q}_1 + (0,00474 + 0,0252c_2)Kg.m^2.\ddot{q}_2 \\ (0,00474 + 0,0252c_2)Kg.m^2\ddot{q}_1 + 0,00474Kg.m^2.\ddot{q}_2 \end{bmatrix}$$

$$\begin{bmatrix} C_a \\ C_b \end{bmatrix} + \begin{bmatrix} G_1 \\ G_2 \end{bmatrix} = \begin{bmatrix} -2m_2 l_1 l_{g2} S_2 \dot{q}_1 \dot{q}_2 - m_2 l_1 l_{g2} S_2 \dot{q}_2{}^2 \\ m_2(l_{g2}{}^2 + l_1 l_{g2} \dot{q}_1 \dot{q}_2) \end{bmatrix} + \begin{bmatrix} m_1 g l_{g1} c_1 + m_2 g(l_1 c_1 + l_{g2} c_{12}) \\ m_2 g l_{g2} c_{12} \end{bmatrix}$$

$$\begin{bmatrix} C_a \\ C_b \end{bmatrix} + \begin{bmatrix} G_1 \\ G_2 \end{bmatrix} = \begin{bmatrix} -2m_2 l_1 l_{g2} S_2 \dot{q}_1 \dot{q}_2 - m_2 l_1 l_{g2} S_2 \dot{q}_2{}^2 + g(m_1 l_{g1} c_1 + m_2 l_1 c_1 + m_2 l_{g2} c_{12}) \\ m_2(l_{g2}{}^2 + l_1 l_{g2} \dot{q}_1 \dot{q}_2) + m_2 g l_{g2} c_{12} \end{bmatrix}$$

$$\begin{bmatrix} C_a \\ C_b \end{bmatrix} + \begin{bmatrix} G_1 \\ G_2 \end{bmatrix} = \begin{bmatrix} (-0,01134 S_2.\dot{q}_2 \dot{q}_1 - 0,00567 \dot{q}_2{}^2)kg.m^2 + 9,81kg.\frac{m}{seg^2}(0,0651.c_1 + 0,081.c_1 + 0,0315c_{12}) \\ (0,002205 + 0,00567 \dot{q}_1 \dot{q}_2)kg.m^2 + 0,3090kg.\frac{m}{seg^2}.c_{12} \end{bmatrix}$$

Podemos observar que tenemos un problema con las unidades de las distintas matrices, queda para un próximo trabajo estudiar en detalle el porqué de este inconveniente.

Por el momento dejaremos todo expresado en función de las aceleraciones ($\ddot{q}_1$ y $\ddot{q}_2$), las velocidades ($\dot{q}_1$ y $\dot{q}_2$) y las relaciones de los ángulos (S_1, S_{12}, C_1, C_{12}).

$$\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \cdot \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} C_a \\ C_b \end{bmatrix} + \begin{bmatrix} G_1 \\ G_2 \end{bmatrix} =$$

$$= \begin{bmatrix} (-0,01134 S_2.\dot{q}_2 \dot{q}_1 - 0,00567 \dot{q}_2{}^2 + 0,33456.\ddot{q}_1 + 0,0252 c_2.\ddot{q}_1)kg.m^2 + \\ +(0,00474 + 0,0252c_2)Kg.m^2.\ddot{q}_2 + 9,81kg.\frac{m}{seg^2}(0,0651.c_1 + 0,081.c_1 + 0,0315c_{12}) + \\ (0,002205 + 0,00567 \dot{q}_1 \dot{q}_2 + 0,00474.\ddot{q}_2 + 0,00474.\ddot{q} + 0,0252 c_2.\ddot{q})kg.m^2 + \\ {}_1 0,3090kg.\frac{m}{seg^2}.c_{12} \end{bmatrix}$$

Una vez que tengamos los correspondientes valores de velocidades, aceleraciones y movimientos que deseamos que realice el robot, vamos a elegir los motores que cumplan con las características que necesitamos para nuestro robot.

Imagen de motores de la marca ABB extraída de la página oficial:

http://www.abb.com/product/seitp322/ff2fcb5f398e79e8c12572e900496fe2.aspx

Imagen extraída de la página oficial de Siemens:

http://www.automation.siemens.com/mcms/mc/en/motors/servo-and-main-spindle-motors/synchronous-motor/Pages/synchronous-motor.aspx

Codigo VHDL PWM:

```
LIBRARY IEEE;

USE IEEE.STD_LOGIC_1164.ALL;

PACKAGE user_pkg IS

   function INC(X: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;

   function DEC(X: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;

END user_pkg ;

PACKAGE BODY user_pkg IS

function INC(X: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR is

   variable XV: STD_LOGIC_VECTOR(X'LENGTH - 1 downto 0);

   begin

  XV := X;

  for I in 0 to XV'HIGH LOOP

     if XV(I) = '0' then

       XV(I) := '1';

       exit;

     else XV(I) := '0';

     end if;

  end loop;

return XV;

end INC;

function DEC(X: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR is

   variable XV: STD_LOGIC_VECTOR(X'LENGTH - 1 downto 0);

   begin

  XV := X;
```

```
 for I in 0 to XV'HIGH LOOP

    if XV(I) = '1' then

      XV(I) := '0';

      exit;

    else XV(I) := '1';

    end if;

 end loop;

return XV;

end DEC;

END user_pkg;
```

```
library IEEE;

USE ieee.std_logic_1164.all;

USE ieee.std_logic_arith.all ;

USE     work.user_pkg.all;

ENTITY pwm_fpga IS

PORT (  clock,reset                     :in  STD_LOGIC;

      Data_value                  :in  std_logic_vector(7 downto 0);

      pwm                           :out STD_LOGIC

   );

END pwm_fpga;

ARCHITECTURE arch_pwm OF pwm_fpga IS

SIGNAL reg_out                                  : std_logic_vector(7 downto 0);

SIGNAL cnt_out_int                          : std_logic_vector(7 downto 0);

SIGNAL pwm_int, rco_int                         : STD_LOGIC

BEGIN

 -- 8 BIT DATA REGISTER TO STORE THE MARKING VALUES .

 -- THE MARKING VALUES WILL DETERMINE THE DUTY CYCLE OF PWM OUTPUT


 PROCESS(clock,reg_out,reset)

                BEGIN

                        IF (reset ='1') THEN

                        reg_out <="00000000";

                        ELSIF (rising_edge(clock)) THEN

                                reg_out <= data_value;

                        END IF;

                END PROCESS;
```

```
-- 8 BIT UPDN COUNTER. COUNTS UP OR DOWN BASED ON THE PWM_INT SIGNAL AND GENERATES

-- TERMINAL COUNT WHENEVER COUNTER REACHES THE MAXIMUM VALUE OR WHEN IT TRANSISTS

-- THROUGH ZERO. THE TERMINAL COUNT WILL BE USED AS INTERRUPT TO AVR FOR GENERATING

-- THE LOAD SIGNAL.

-- INC and DEC are the two functions which are used for up and down counting. They are defined in sepearate user_pakge library

PROCESS (clock,cnt_out_int,rco_int,reg_out)

        BEGIN

         IF (rco_int = '1') THEN

            cnt_out_int <= reg_out;

         ELSIF  rising_edge(clock) THEN

        IF (rco_int = '0' and pwm_int ='1' and cnt_out_int <"11111111") THEN

              cnt_out_int <= INC(cnt_out_int);

        ELSE

           IF (rco_int ='0' and pwm_int ='0' and cnt_out_int > "00000000") THEN

                    cnt_out_int <= DEC(cnt_out_int);

                END IF;

           END IF;

        END IF;

END PROCESS;
```

```
-- Logic to generate RCO signal

 PROCESS(cnt_out_int, rco_int, clock,reset)

          BEGIN

         IF (reset ='1') THEN

               rco_int <='1';

            ELSIF rising_edge(clock) THEN

          IF ((cnt_out_int = "11111111") or (cnt_out_int ="00000000")) THEN

                 rco_int <= '1';

          ELSE

                    rco_int <='0';

              END IF;

            END IF;

END PROCESS;

-- TOGGLE FLIP FLOP TO GENERATE THE PWM OUTPUT.

          PROCESS (clock,rco_int,reset)

          BEGIN

                   IF (reset = '1') THEN

                     pwm_int <='0';

                    ELSIF rising_edge(rco_int) THEN

                     pwm_int <= NOT(pwm_int);

                    ELSE

                     pwm_int <= pwm_int;

                   END IF;

          END PROCESS;

          pwm <= pwm_int

END arch_pwm;
```

Codigo VHDL Control de Motor

```
library IEEE;

USE ieee.std_logic_1164.all;

USE ieee.std_logic_arith.all;


ENTITY MotorControl IS

PORT (  clockSys,resetSys,sentido     :in  STD_LOGIC;

      DataSys                  :in  std_logic_vector(7 downto 0);

      HALLs                :in  std_logic_vector(2 downto 0);

      pwm1,pwm2,pwm3                          :out STD_LOGIC;

      salidasQ                              :out std_logic_vector(2 downto 0)


   );

END MotorControl;


ARCHITECTURE estrucMotorContol OF MotorControl IS


 COMPONENT pwm_fpga

  PORT (clock: IN STD_LOGIC;

            reset: IN STD_LOGIC;

            data_value: IN std_logic_vector(7 downto 0);

            pwm: OUT STD_LOGIC);

 END COMPONENT;


 SIGNAL reset1,reset2,reset3 : STD_LOGIC;

 BEGIN
```

```
  --Instanciado nominal de componentes

  U1:pwm_fpga PORT MAP(clock => clockSys, reset => reset1, data_value => DataSys,
pwm => pwm1);

  U2:pwm_fpga PORT MAP(clock => clockSys, reset => reset2, data_value => DataSys,
pwm => pwm2);

  U3:pwm_fpga PORT MAP(clock => clockSys, reset => reset3, data_value => DataSys,
pwm => pwm3);


 PROCESS(resetSys)

 --las salidas pwm (conectadas en la parte baja del puente) siempre están
funcionando, las salidas Q (conectadas en la parte alta) respetan la sig. secuencia
según el estado de los HALLs

   BEGIN

    IF (resetSys ='1') THEN

      reset1 <= '1';

      reset2 <= '1';

      reset3 <= '1';

    END IF;

 END PROCESS;
```

```
PROCESS(HALLs,sentido)
  BEGIN
   IF(sentido = '0') THEN --sentido horario

    CASE HALLs IS
       WHEN "001" =>
        salidasQ <= "001"; reset3 <= '0'; reset2 <= '1'; reset1 <= '1';
       WHEN "000" =>
        salidasQ <= "001"; reset3 <= '1'; reset2 <= '0'; reset1 <= '1';
       WHEN "100" =>
        salidasQ <= "100"; reset3 <= '1'; reset2 <= '0'; reset1 <= '1';
       WHEN "110" =>
        salidasQ <= "100"; reset3 <= '1'; reset2 <= '1'; reset1 <= '0';
       WHEN "111" =>
        salidasQ <= "010"; reset3 <= '1'; reset2 <= '1'; reset1 <= '0';
       WHEN "011" =>
        salidasQ <= "010"; reset3 <= '0'; reset2 <= '1'; reset1 <= '1';
       WHEN OTHERS => NULL;
    END CASE;
  END IF;

  IF(sentido = '1') THEN --sentido antihorario

    CASE HALLs IS
        WHEN "011" =>
          salidasQ <= "100"; reset3 <= '1'; reset2 <= '0'; reset1 <= '1';
```

```
        WHEN "111" =>
          salidasQ <= "001"; reset3 <= '1'; reset2 <= '0'; reset1 <= '1';
        WHEN "110" =>
          salidasQ <= "001"; reset3 <= '0'; reset2 <= '1'; reset1 <= '1';
        WHEN "100" =>
          salidasQ <= "010"; reset3 <= '0'; reset2 <= '1'; reset1 <= '1';
        WHEN "000" =>
          salidasQ <= "010"; reset3 <= '1'; reset2 <= '1'; reset1 <= '0';
        WHEN "001" =>
          salidasQ <= "100"; reset3 <= '1'; reset2 <= '1'; reset1 <= '0';
        WHEN OTHERS => NULL;
      END CASE;
    END IF;
  END PROCESS;

 END ARCHITECTURE estrucMotorContol;
```

Codigo VHDL Test Motor Control

```
LIBRARY ieee;
use ieee.std_logic_1164.all;
ENTITY Test_MotorControl IS
END Test_MotorControl;
```

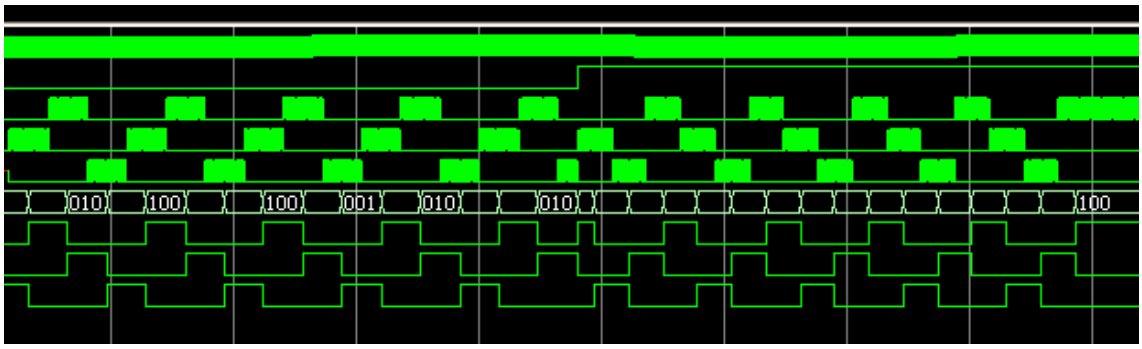
```
ARCHITECTURE archite_Test_MotorControl OF Test_MotorControl IS

COMPONENT MotorControl

PORT (

  clockSys:      in std_logic;

  resetSys:      in std_logic;

  sentido:       in std_logic;

  DataSys:       in std_logic_vector(7 downto 0);

  HALLs:         in std_logic_vector(2 downto 0);

  pwm1,pwm2,pwm3: out std_logic;

  salidasQ:      out std_logic_vector(2 downto 0)

);

END COMPONENT;

-- Internal signal declaration

SIGNAL sig_clock       : std_logic;

SIGNAL sig_reset       : std_logic;

SIGNAL sig_sentido : std_logic;

SIGNAL sig_data_value          : std_logic_vector(7 downto 0);

SIGNAL sig_HALLs : std_logic_vector(2 downto 0);

SIGNAL sig_pwm1, sig_pwm2, sig_pwm3     : std_logic;

SIGNAL sig_salidasQ : std_logic_vector(2 downto 0);

shared variable ENDSIM: boolean:=false;

constant clk_period:TIME:=200 ns;
```

```
BEGIN

clk_gen: process

   BEGIN

   If ENDSIM = FALSE THEN

     sig_clock <= '1';

     wait for clk_period/2;

     sig_clock <= '0';

     wait for clk_period/2;

   else

     wait;

   end if;

 end process;

-- Instantiating top level design Component pwm_fpga


inst_MotorControl : MotorControl

PORT MAP(

   clockSys      => sig_clock,

   resetSys      => sig_reset,

   sentido => sig_sentido,

   DataSys       => sig_data_value,

   HALLs => sig_HALLs,

   pwm1          => sig_pwm1,

   pwm2 => sig_pwm2,

   pwm3 => sig_pwm3,

   salidasQ => sig_salidasQ

 );
```

```
stimulus_process: PROCESS

-- la idea de este test es simular el giro del motor a una velocidad real por ejemplo 1200 RPM

-- para esto calculo que el tiempo entre las secuencia del los Halls es aprox. 8ms

-- 1200/60= 20 vueltas x segundo  1 vuelta = 0,05 segundos como son 6 estados 0,05/6 = 8,333 ms

BEGIN

 sig_sentido <= '0'; --sentido horario

 sig_reset <= '1';

 wait for 500 ns;

 sig_reset <= '0';

 sig_data_value <= "10000000";  --duty 25%

 wait for 500 ns;

 for i in 1 to 5 loop   --simulo 5 vueltas del motor(sentido horario)aprox 1200RPM

  sig_HALLs <= "001";

  wait for 8 ms;

  sig_HALLs <= "000";

  wait for 8 ms;

  sig_HALLs <= "100";

  wait for 8 ms;

  sig_HALLs <= "110";

  wait for 8 ms;

  sig_HALLs <= "111";

  wait for 8 ms;

  sig_HALLs <= "011";

  wait for 8 ms;
```

```
 end loop;

 wait for 1000 ns;

 sig_sentido <= '1'; --sentido antihorario

 for i in 1 to 5 loop   --simulo 5 vueltas del motor(sentido antihorario)

    sig_HALLs <= "011";

    wait for 7 ms;

    sig_HALLs <= "111";

    wait for 7 ms;

    sig_HALLs <= "110";

    wait for 7 ms;

    sig_HALLs <= "100";

    wait for 7 ms;

    sig_HALLs <= "000";

    wait for 7 ms;

    sig_HALLs <= "001";

    wait for 7 ms;

  end loop;

wait;

 END PROCESS stimulus_process;

END archite_Test_MotorControl;
```

Resultados Simulados:



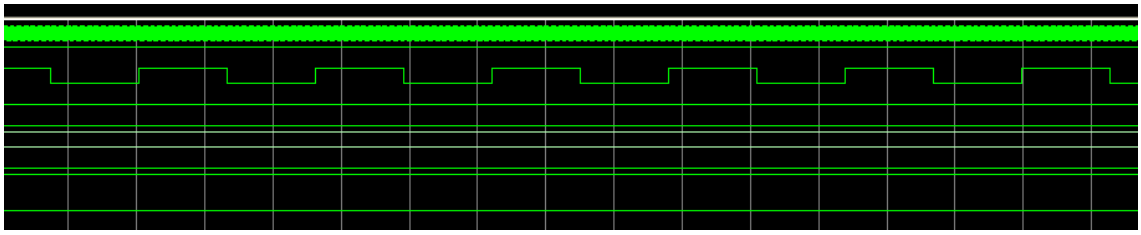
Donde la primera señal (“mancha verde”) es el clock. Es lógico que se vea de esta manera ya que es mucho más rápido que el resto de las señales.

La segunda indica el sentido “0” sentido horario, “1” sentido anti horario.

Las siguientes tres señales son los tres PWM.

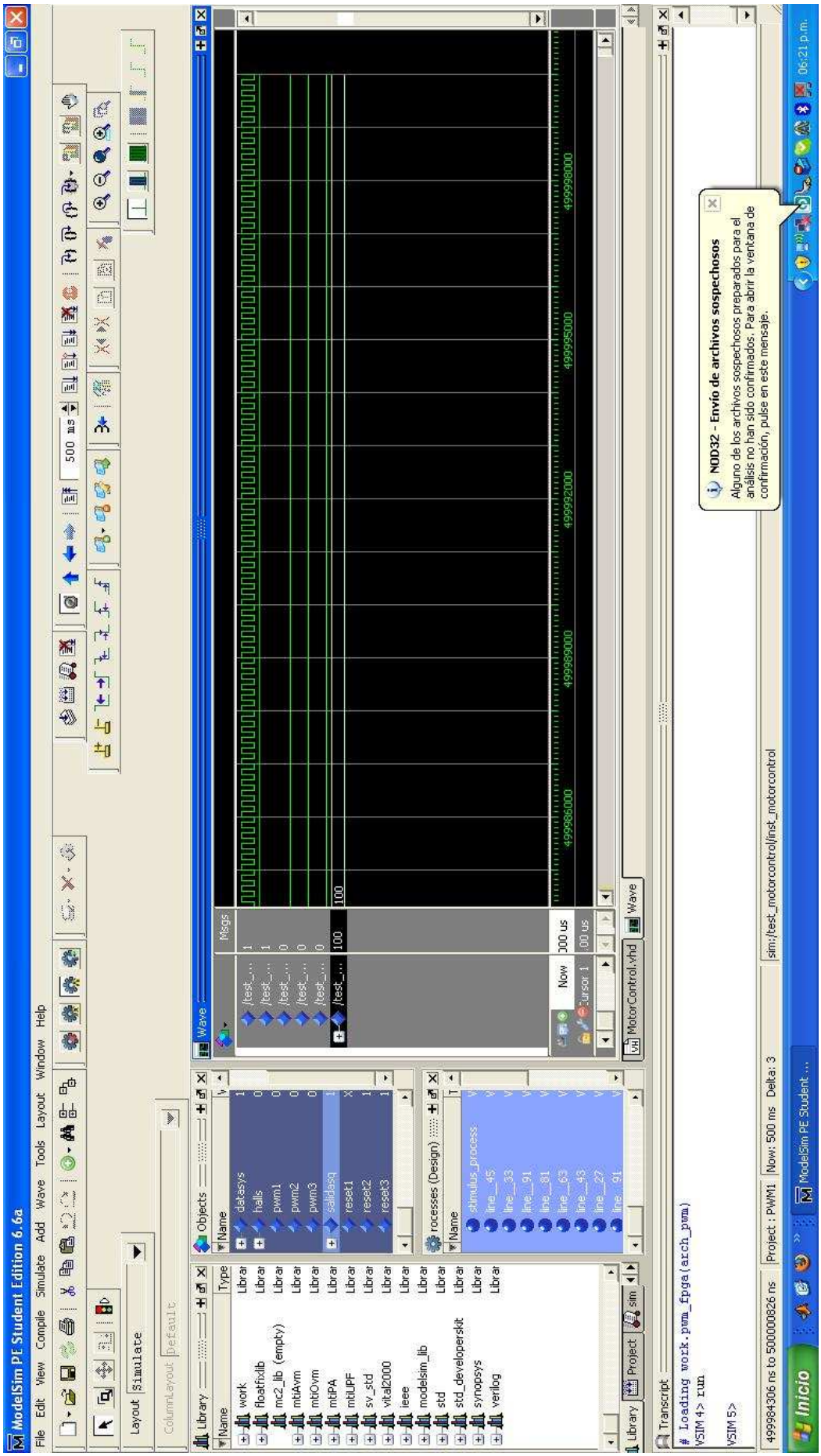
Las últimas tres son las señales de activación de los transistores de la parte alta del puente, estas respetan la secuencia indicada por los sensores de efecto Hall.

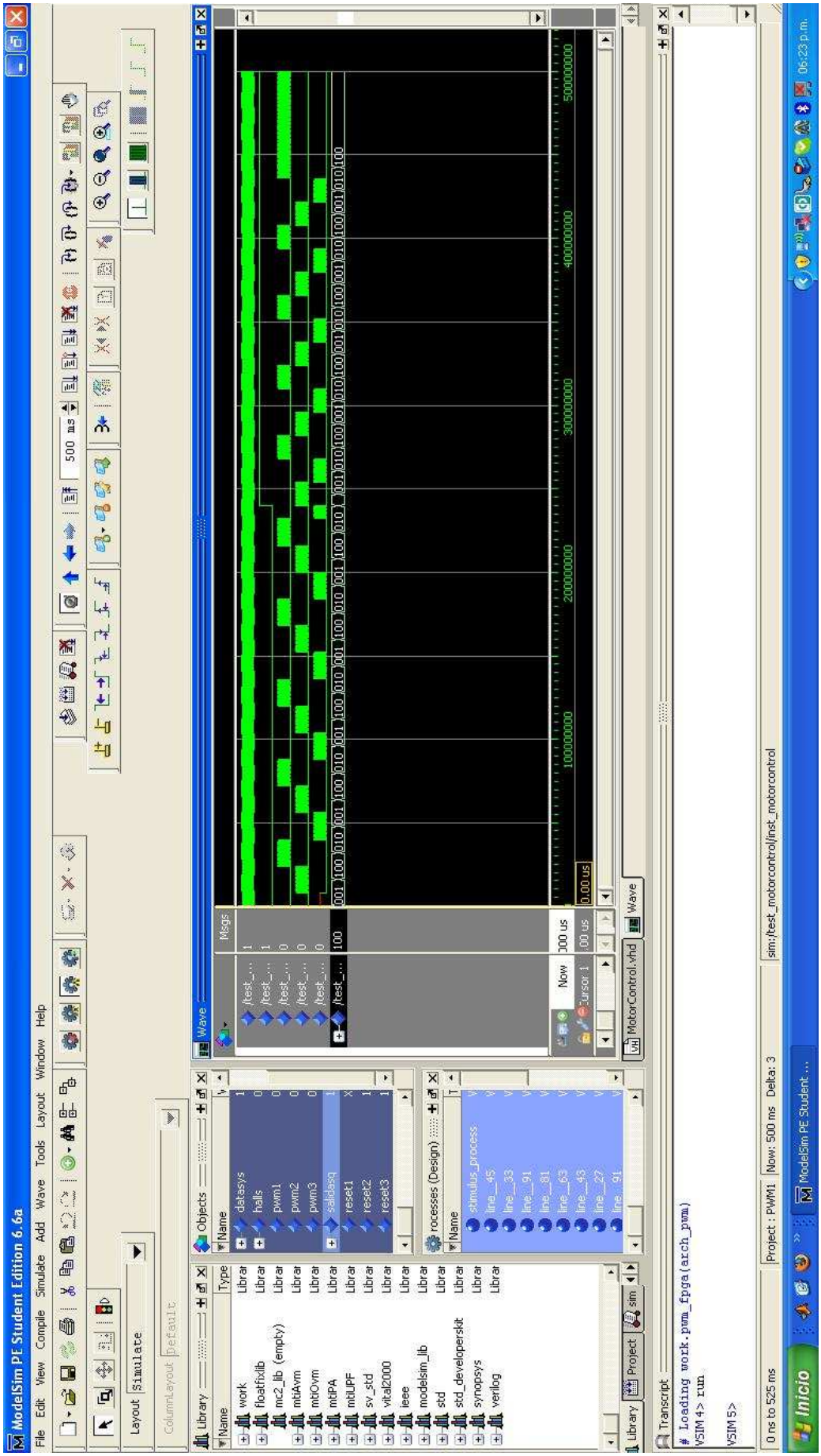
Por ultimo en esta figura se aprecia como al cambiar el sentido se invierte la secuencia de excitación de los transistores (en test se realizan cinco vueltas en un sentido y cinco en el otro).

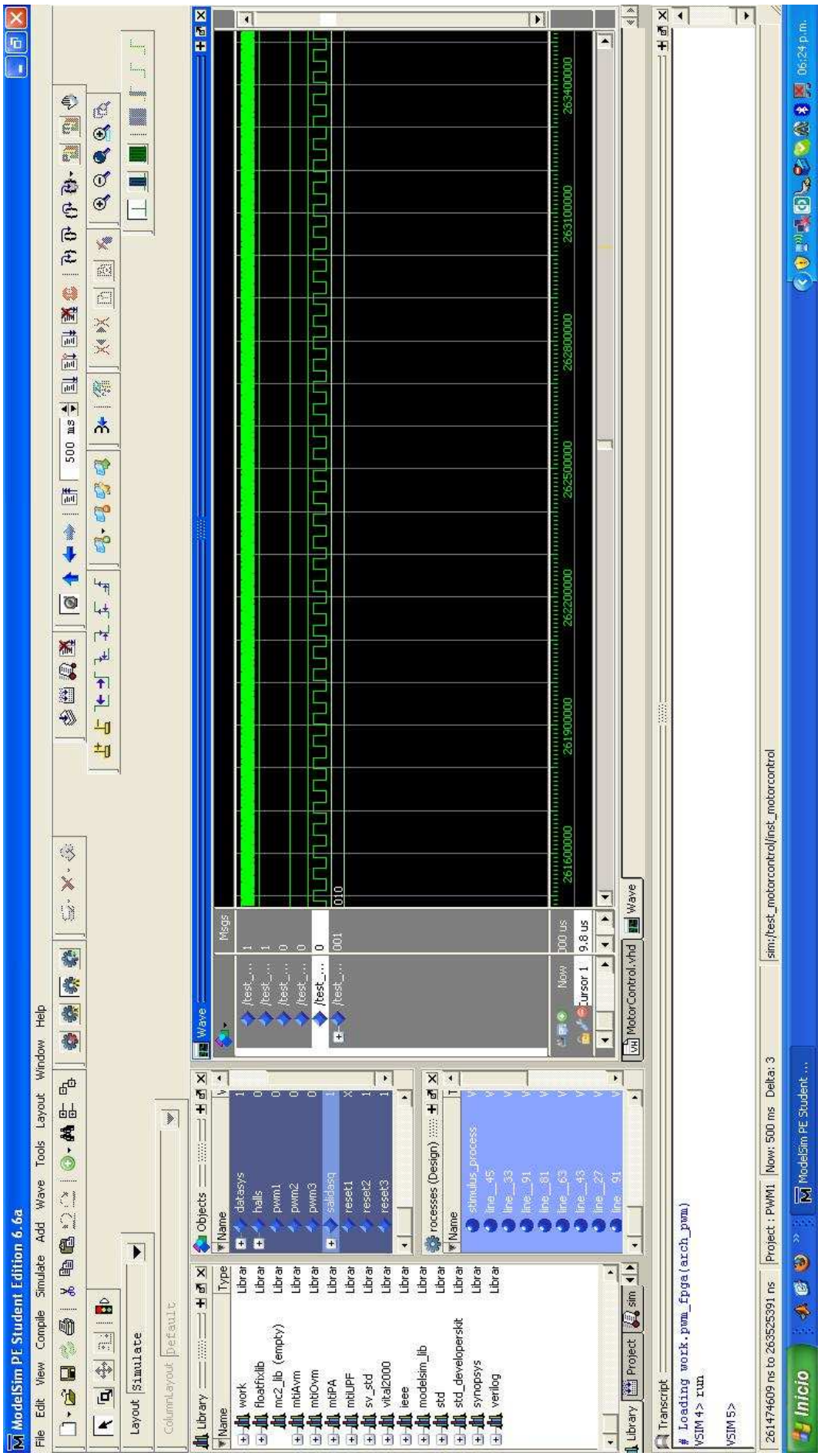


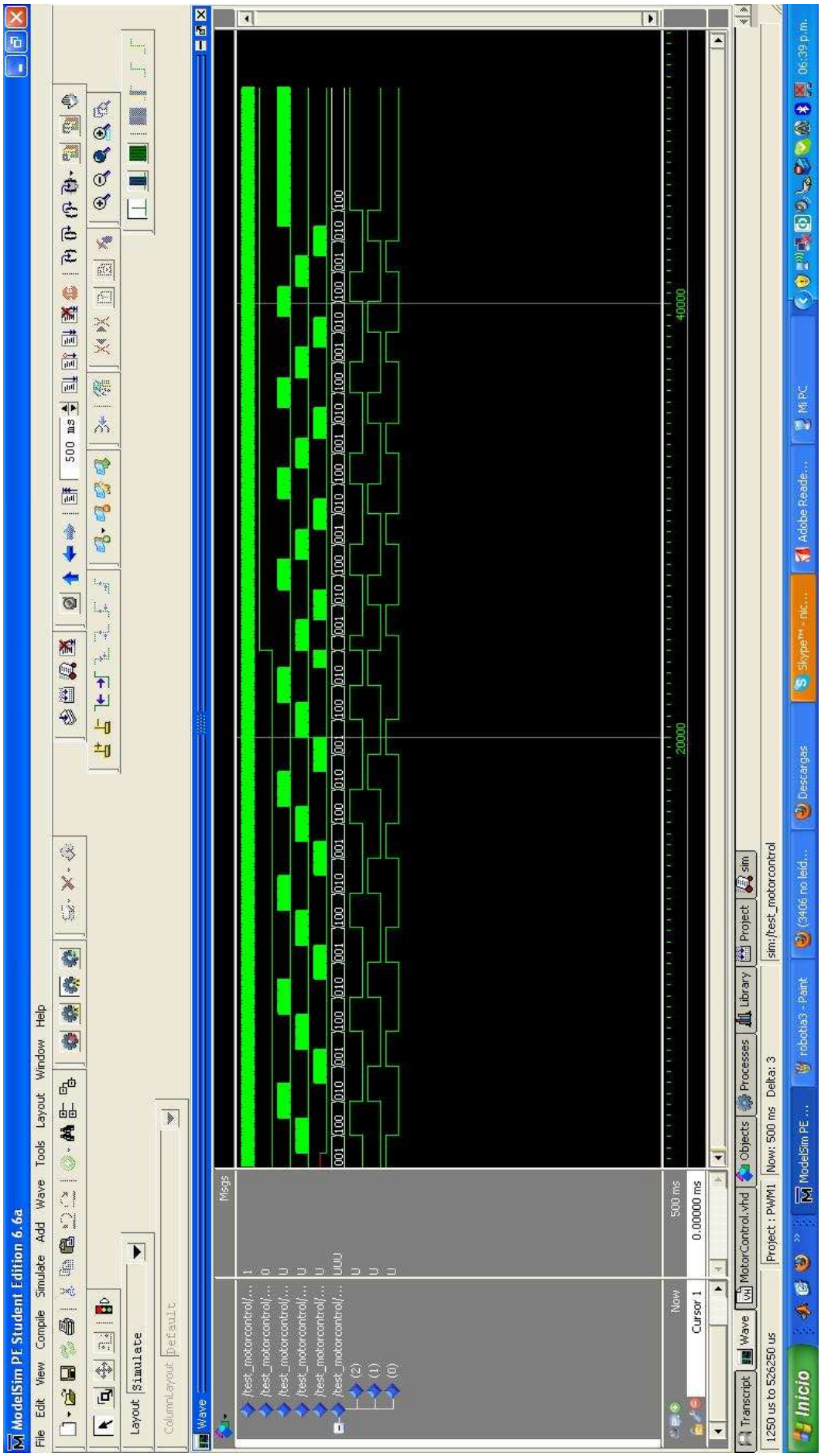
En esta imagen se aprecia una de las señales de PWM, que en este caso esta al 50%, esta señal tiene una frecuencia aprox. de 20khz esto es así para evitar zumbidos en el motor, aunque este valor es aproximado y debe ajustarse según la aplicación y el motor de que se trate.

A continuación ponemos algunas impresiones de pantalla extraídas del programa ModelSim que utilizamos para realizar la simulación del VHDL.


ModelSim PE Student Edition 6.6a
File Edit View Compile Simulate Add Wave Tools Layout Window Help
Layout Simulate
Library
Objects
Wave
Msgs
Transcript
Loading work.pwm_fpga(arch_pwm)
VSIM 4> run
VSIM 5>
MotorControl.vhd
Wave
NOD32 - Envío de archivos sospechosos
Alguno de los archivos sospechosos preparados para el análisis no han sido confirmados. Para abrir la ventana de confirmación, pulse en este mensaje.
Inicio
ModelSim PE Student ...
06:21 p.m.


ModelSim PE Student Edition 6.6a
File Edit View Compile Simulate Add Wave Tools Layout Window Help
Wave
Objects
Library
Transcript
Loading work.pwm_fpga(arch_pwm)
VSIM 4> run
VSIM 5>
Project : PWM1
Now: 500 ms Delta: 3
sim:/test_motorcontrol/inst_motorcontrol
ModelSim PE Student...
Inicio
06:23 p.m.


ModelSim PE Student Edition 6.6a
File Edit View Compile Simulate Add Wave Tools Layout Window Help
Wave
Objects
Library
Transcript
Loading work.pwm_fpga(arch_pwm)
VSIM 4> run
VSIM 5>
MotorControl.vhd
Wave
Inicio
ModelSim PE Student ...
06:24 p.m.


ModelSim PE Student Edition 6.6a
File Edit View Compile Simulate Add Wave Tools Layout Window Help
Wave
Msgs
/test_motorcontrol/...
Now
Cursor 1
0.00000 ms
20000
40000
Transcript
MotorControl.vhd
Objects
Processes
Library
Project
sim
Project : PWM1
Now: 500 ms Delta: 3
sim:/test_motorcontrol
1250 us to 526250 us
Inicio
ModelSim PE ...
robotia3 - Paint
(3406 no leid...
Descargas
Skype™ - nic...
Adobe Reade...
Mi PC
06:39 p.m.

Conclusiones:

En este trabajo pudimos notar que si bien el análisis dinámico del mecanismo es muy complejo, aumentando según los grados de libertad, existen herramientas de software tanto en programación y en simulación que nos permiten realizar una evaluación muy cercana a la realidad.

Esto tiene un gran impacto en los tiempos y los costos de desarrollo del robot. Además tuvimos contacto con los fabricantes de motores y con la programación de dispositivos de hardware reconfigurable, que sin duda representan la mejor opción para la implementación de control del robot complejo.