UNIVERSIDAD TECNOLÓGICA NACIONAL – FACULTAD REGIONAL BUENOS AIRES
INGENIERÍA ELECTRÓNICA

Asignatura: Robótica *Código:* 95-0482

Curso: R-6055 *Año:* 2010

Trabajo Práctico n° 2:

Análisis Dinámico de un robot e implementación en FPGA

Alumnos:

- BORNEO, Pablo N.
- DI VRUNO, Federico
- SERATTI, Federico

Profesor: Mas. Ing. GIANNETA, Hernan

JTP: Ing. GRANZELLA, Damian

ÍNDICE

INTRODUCCIÓN TEÓRICA

Dinámica del robot

El modelo dinámico del robot se ocupa de la relación entre sus movimientos y las fuerzas implicadas en estos. El modelo dinámico relaciona las siguientes propiedades del robot:

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, es decir mediante una serie de ecuaciones, normalmente de tipo diferencial de 2do orden, cuya integración permite conocer que movimiento surge al aplicar determinadas 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.

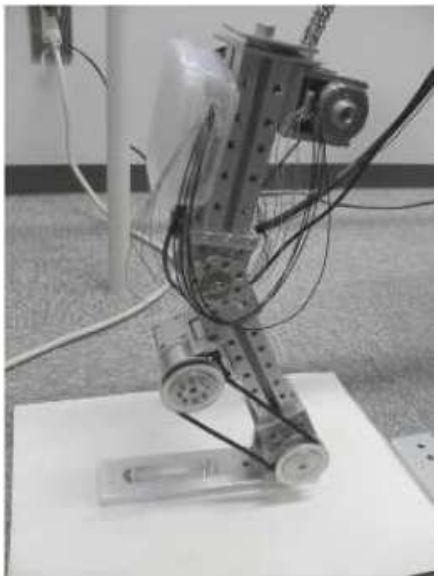
En el presente trabajo se hará el análisis utilizando las formulaciones lagrangianas por ser una herramienta muy eficaz a medida que aumentan los grados de libertad. Es importante señalar que existen otras formulaciones también validas como las newtonianas y variantes entre estas dos que se han ido adaptando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

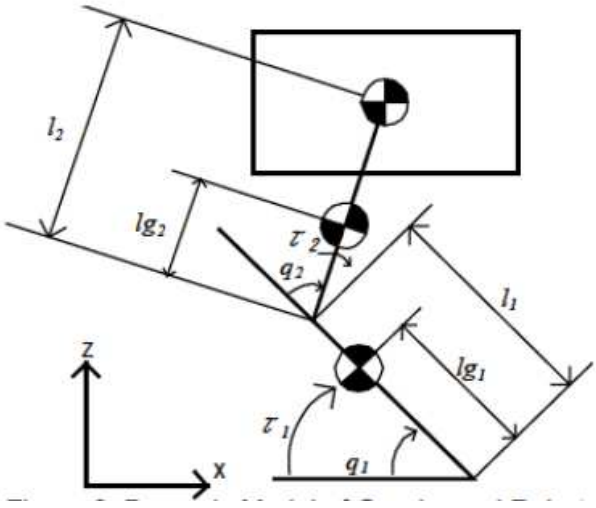
Para terminar se presentarán los resultados de la simulación, los cuales permitirán analizar el comportamiento del robot y los sistemas mecánicos ante las trayectorias propuestas.

DESARROLLO DE LA PRÁCTICA

Se realizara el análisis dinámico de la estructura mecánica del “One Legged Robot”, y luego implementar el control de una maquina PWM en lenguaje VHDL .



El esquema representativo del robot, que será utilizado para realizar el estudio es el siguiente:



Para el estudio se considera solamente el movimiento en el plano XZ. A continuación se detalla la tabla con los valores de cada elemento del robot que serán luego utilizados.

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. Elementos del "One Legged Robot"

Matrices para el cálculo de los torques

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.

Estas tres matrices están determinadas de la siguiente manera:

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

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

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

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

Referencias

S1 sin(q1)

S12 sin(q1+q2)

C1 cos(q1)

C12 cos(q1+q2)

g 9,81m/s2

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 ($I=\frac{1}{3}.m.L^2$):

$I_1=\frac{1}{3}.m_1.l_1^2 = \frac{1}{3}.0{,}465Kg.(0{,}18m)^2 = 0{,}005022Kg.m^2$.

$I_2 = \frac{1}{3}.m_2.l_2^2 = \frac{1}{3}.0{,}450Kg.(0{,}13m)^2 = 0{,}002535Kg.m^2$.

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

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

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

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

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

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

Finalmente obtenemos la matriz de torques t1 y t2 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}).

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

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

Fuente01: usr_pkg.vhd

```
1   LIBRARY IEEE;
2   USE IEEE.STD_LOGIC_1164.ALL;
3   
4   PACKAGE user_pkg IS
5       function INC(X: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;
6       function DEC(X: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;
7   END user_pkg ;
8   
9   PACKAGE BODY user_pkg IS
10  
11  function INC(X: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR is
12      variable XV: STD_LOGIC_VECTOR(X'LENGTH - 1 downto 0);
13    begin
14    XV := X;
15    for I in 0 to XV'HIGH LOOP
16        if XV(I) = '0' then
17           XV(I) := '1';
18           exit;
19        else XV(I) := '0';
20        end if;
21    end loop;
22  return XV;
23  end INC;
24  
25  function DEC(X: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR is
26      variable XV: STD_LOGIC_VECTOR(X'LENGTH - 1 downto 0);
27      begin
28    XV := X;
29  
30    for I in 0 to XV'HIGH LOOP
31        if XV(I) = '1' then
32           XV(I) := '0';
33           exit;
34        else XV(I) := '1';
35        end if;
36    end loop;
37  return XV;
38  end DEC;
39  
40  END user_pkg;
41  
42  
```

Fuente02: pwm_fpga.vhd

```
1   library IEEE;
2   USE ieee.std_logic_1164.all;
3   USE ieee.std_logic_arith.all ;
4   USE work.user_pkg.all;
5
6   ENTITY pwm_fpga IS
7   PORT (  clock,reset      :in  STD_LOGIC;
8           Data_value       :in  std_logic_vector(7 downto 0);
9           pwm              :out STD_LOGIC
10       );
11  END pwm_fpga;
12
13  ARCHITECTURE arch_pwm OF pwm_fpga IS
14  SIGNAL reg_out      : std_logic_vector(7 downto 0);
15  SIGNAL cnt_out_int    : std_logic_vector(7 downto 0);
16  SIGNAL pwm_int, rco_int    : STD_LOGIC ;
17
18  BEGIN
19   -- 8 BIT DATA REGISTER TO STORE THE MARKING VALUES .
20   -- THE MARKING VALUES WILL DETERMINE THE DUTY CYCLE OF PWM OUTPUT
21
22  PROCESS(clock,reg_out,reset)
23      BEGIN
24        IF (reset ='1') THEN
25        reg_out <="00000000";
26        ELSIF (rising_edge(clock)) THEN
27          reg_out <= data_value;
28        END IF;
29      END PROCESS;
30
31   -- 8 BIT UPDN COUNTER. COUNTS UP OR DOWN BASED ON THE PWM_INT SIGNAL AND GENERATES
32   -- TERMINAL COUNT WHENEVER COUNTER REACHES THE MAXIMUM VALUE OR WHEN IT TRANSISTS
33   -- THROUGH ZERO. THE TERMINAL COUNT WILL BE USED AS INTERRUPT TO AVR FOR GENERATING
34   -- THE LOAD SIGNAL.
35   -- INC and DEC are the two functions which are used for up and down counting. They are defined in sepearate user_
36   PROCESS (clock,cnt_out_int,rco_int,reg_out)
37    BEGIN
38     IF (rco_int = '1') THEN
39         cnt_out_int <= reg_out;
40     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;
```

Fuente03: control_motor.vhd

Este es el archivo VHDL que instancia 3 módulos PWM definidos en pwm_fpga.vhd y según las entradas HALL y el sentido de giro activa el correspondiente pwm.

```
1   library IEEE;
2   USE ieee.std_logic_1164.all;
3   USE ieee.std_logic_arith.all;
4
5   ENTITY control_motor IS
6   PORT (  clockSys,resetSys,sentido :in  STD_LOGIC;
7           DataSys                   :in  std_logic_vector(7 downto 0);
8           HALLs                     :in  std_logic_vector(2 downto 0);
9           pwm1,pwm2,pwm3            :out STD_LOGIC;
10          salidasQ                  :out std_logic_vector(2 downto 0)
11       );
12  END control_motor;
13
14
15
16  ARCHITECTURE uno OF control_motor IS
17
18    COMPONENT pwm_fpga
19      PORT (clock: IN STD_LOGIC;
20            reset: IN STD_LOGIC;
21            data_value: IN std_logic_vector(7 downto 0);
22            pwm: OUT STD_LOGIC);
23    END COMPONENT;
24
25  SIGNAL reset1,reset2,reset3 : STD_LOGIC;
26  BEGIN
27     -- Instanciado de los 3 modulos pwm
28     U1:pwm_fpga PORT MAP(clock => clockSys, reset => reset1, data_value => DataSys, pwm => pwm1);
29     U2:pwm_fpga PORT MAP(clock => clockSys, reset => reset2, data_value => DataSys, pwm => pwm2);
30     U3:pwm_fpga PORT MAP(clock => clockSys, reset => reset3, data_value => DataSys, pwm => pwm3);
31
32
33  PROCESS(resetSys)
34    BEGIN
35        IF (resetSys ='1') THEN
36          reset1 <= '1';
37          reset2 <= '1';
38          reset3 <= '1';
39        END IF;
40    END PROCESS;
41
42
```

```
44  PROCESS(HALLs,sentido)
45    BEGIN
46       IF(sentido = '0') THEN --sentido horario
47
48          CASE HALLs IS
49              WHEN "001" =>
50                salidasQ <= "001"; reset3 <= '0'; reset2 <= '1'; reset1 <= '1';
51              WHEN "000" =>
52                salidasQ <= "001"; reset3 <= '1'; reset2 <= '0'; reset1 <= '1';
53              WHEN "100" =>
54                salidasQ <= "100"; reset3 <= '1'; reset2 <= '0'; reset1 <= '1';
55              WHEN "110" =>
56                salidasQ <= "100"; reset3 <= '1'; reset2 <= '1'; reset1 <= '0';
57              WHEN "111" =>
58                salidasQ <= "010"; reset3 <= '1'; reset2 <= '1'; reset1 <= '0';
59              WHEN "011" =>
60                salidasQ <= "010"; reset3 <= '0'; reset2 <= '1'; reset1 <= '1';
61              WHEN OTHERS => NULL;
62          END CASE;
63       END IF;
64
65       IF(sentido = '1') THEN --sentido antihorario
66
67         CASE HALLs IS
68               WHEN "011" =>
69                  salidasQ <= "100"; reset3 <= '1'; reset2 <= '0'; reset1 <= '1';
70               WHEN "111" =>
71                  salidasQ <= "001"; reset3 <= '1'; reset2 <= '0'; reset1 <= '1';
72               WHEN "110" =>
73                   salidasQ <= "001"; reset3 <= '0'; reset2 <= '1'; reset1 <= '1';
74               WHEN "100" =>
75                   salidasQ <= "010"; reset3 <= '0'; reset2 <= '1'; reset1 <= '1';
76               WHEN "000" =>
77                   salidasQ <= "010"; reset3 <= '1'; reset2 <= '1'; reset1 <= '0';
78               WHEN "001" =>
79                   salidasQ <= "100"; reset3 <= '1'; reset2 <= '1'; reset1 <= '0';
80               WHEN OTHERS => NULL;
81           END CASE;
82       END IF;
83    END PROCESS;
84
85   END ARCHITECTURE uno;
```

Fuente04: test_control_motor.vhd

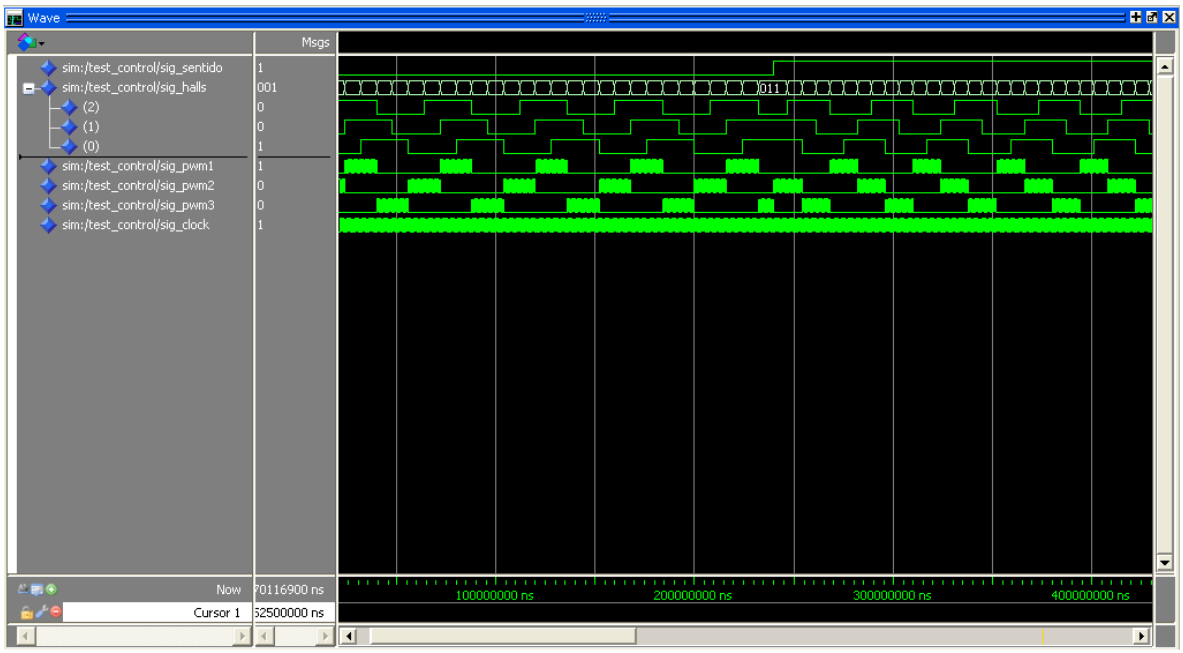
```
1   LIBRARY ieee;
2   use ieee.std_logic_1164.all;
3   ENTITY test_control IS
4   END test_control;
5
6   ARCHITECTURE Uno OF test_control IS
7
8   COMPONENT control_motor
9   PORT (
10    clockSys:        in std_logic;
11    resetSys:        in std_logic;
12    sentido:         in std_logic;
13    DataSys:         in std_logic_vector(7 downto 0);
14    HALLs:           in std_logic_vector(2 downto 0);
15    pwm1,pwm2,pwm3: out std_logic;
16    salidasQ:        out std_logic_vector(2 downto 0) );
17  END COMPONENT;
18
19  -- Internal signal declaration
20  SIGNAL sig_clock : std_logic;
21  SIGNAL sig_reset : std_logic;
22  SIGNAL sig_sentido : std_logic;
23  SIGNAL sig_data_value : std_logic_vector(7 downto 0);
24  SIGNAL sig_HALLs : std_logic_vector(2 downto 0);
25  SIGNAL sig_pwm1, sig_pwm2, sig_pwm3 : std_logic;
26  SIGNAL sig_salidasQ : std_logic_vector(2 downto 0);
27  shared variable ENDSIM: boolean:=false;
28  constant clk_period:TIME:=200 ns;
29
30
31  BEGIN
32  clk_gen: process
33      BEGIN
34      If ENDSIM = false THEN
35        sig_clock <= '1';
36        wait for clk_period/2;
37        sig_clock <= '0';
38        wait for clk_period/2;
39      else
40        wait;
41      end if;
42    end process;
```

```
43  -- Instantiating top level design Component pwm_fpga
44
45  inst_control_motor : control_motor
46  PORT MAP(
47      clockSys => sig_clock,
48      resetSys => sig_reset,
49      sentido => sig_sentido,
50      DataSys => sig_data_value,
51      HALLs => sig_HALLs,
52      pwm1 => sig_pwm1,
53      pwm2 => sig_pwm2,
54      pwm3 => sig_pwm3,
55      salidasQ => sig_salidasQ
56    );
57
58  stimulus_process: PROCESS
59  -- la idea de este test es simular el giro del motor a una velocidad real por ejemplo 1200 RPM
60  -- para esto calculo que el tiempo entre las secuencia del los Halls es aprox. 8ms
61  -- 1200/60= 20 vueltas x segundo  1 vuelta = 0,05 segundos como son 6 estados 0,05/6 = 8,333 ms
62  BEGIN
63    sig_sentido <= '0'; --sentido horario
64    sig_reset <= '1';
65    wait for 500 ns;
66    sig_reset <= '0';
67    sig_data_value <= "10000000";  --duty 25%
68    wait for 500 ns;
69    for i in 1 to 5 loop   --simulo 5 vueltas del motor(sentido horario)aprox 1200RPM
70      sig_HALLs <= "001";
71      wait for 8 ms;
72      sig_HALLs <= "000";
73      wait for 8 ms;
74      sig_HALLs <= "100";
75      wait for 8 ms;
76      sig_HALLs <= "110";
77      wait for 8 ms;
78      sig_HALLs <= "111";
79      wait for 8 ms;
80      sig_HALLs <= "011";
81      wait for 8 ms;
82   end loop;
83   wait for 1000 ns;
84   sig_sentido <= '1'; --sentido antihorario
```

```
85    for i in 1 to 5 loop   --simulo 5 vueltas del motor(sentido antihorario)
86        sig_HALLs <= "011";
87        wait for 7 ms;
88        sig_HALLs <= "111";
89        wait for 7 ms;
90        sig_HALLs <= "110";
91        wait for 7 ms;
92        sig_HALLs <= "100";
93        wait for 7 ms;
94        sig_HALLs <= "000";
95        wait for 7 ms;
96        sig_HALLs <= "001";
97        wait for 7 ms;
98     end loop;
99   wait;
100   END PROCESS stimulus_process;
101 END uno;
```

RESULTADOS DE LA SIMULACIÓN

Resultados de simular con el test bench “test_control_motor”. Se observan las señales de los sensores hall y los pwm.



CONCLUSIONES

Este trabajo nos permitió conocer como se realiza un análisis dinámico de una estructura robótica, además observamos que por el gran aumento de la complejidad de este análisis, deben utilizarse herramientas computacionales.

La segunda parte del trabajo fue muy interesante ya que nos introdujo al mundo del VHDL con la utilización de código de ejemplo para luego realizar algo más grande instanciando estos componentes. Seria interesante poder bajar este código en una FPGA y medir las señales.