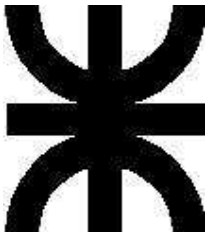


Universidad Tecnológica Nacional
Facultad Regional Buenos Aires

Materia: *Robótica*
Curso: *R-6055*
Profesor: *Ing. Hernán Gianetta*
JTP: *Ing. Damián Granzella*
Año: *2010*

Trabajo Práctico N° 2:

Análisis Dinámico de un Robot e implementación en FPGA

Alumnos: *Charec, Diego* *113768-2*
Montaño Vallejos, Hector *116021-7*

Fecha de Entrega:

Introducción teórica sobre la dinámica del robot

La dinámica es la parte de la física que describe la evolución en el tiempo de un sistema físico en relación a las causas que provocan los cambios de estado físico y/o estado de movimiento.
El objetivo de la dinámica es describir los factores capaces de producir alteraciones de un sistema físico, cuantificarlos y plantear ecuaciones de movimiento o ecuaciones de evolución para dicho sistema.
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:

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

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

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

Pasos de la cadena de una implementación sobre robótica

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.

Existen varios modelos matemáticos para el análisis dinámico, el método Lagrange-Euler, el método Newton-Euler, el método de variables de estado y el método de Kane solo nos centraremos en el método de Lagrange-Euler, y profundizar sobre su simple aplicación sobre el modelo a linealizar.

Mecánica Lagrangiana

En mecánica lagrangiana, la trayectoria de un objeto es obtenida encontrando la trayectoria que minimiza la acción, que es la integral del lagrangiano en el tiempo; siendo éste la energía cinética del objeto menos la energía potencial del mismo.

La formulación lagrangiana simplifica considerablemente muchos problemas físicos. Por ejemplo los sistemas de referencia inerciales son tratados en pie de igualdad y a diferencia de las leyes de Newton la forma de las ecuaciones del movimiento no depende del sistema de referencia elegido.

Método Lagrange - Euler.

Uicker en 1965 utilizó la representación de D-H basada en las matrices de transformación homogénea para formular el modelo dinámico de un robot mediante la ecuación de Lagrange.

Este planteamiento utiliza, por tanto, las matrices i-1**A** i que relacionan el sistema de coordenadas de referencia del elemento *i* con el del elemento *i-1*.

Se realizan en este caso operaciones de producto y suma innecesarias (recuérdese la información redundante contenida en las matrices **A** debido a la ortonormalidad de la submatriz de rotación).

Sin embargo, conduce a unas ecuaciones finales bien estructuradas donde aparecen de manera clara los diversos pares y fuerzas que intervienen en el movimiento (inercia, Coriolis, gravedad).

Se presenta a continuación al algoritmo a seguir para obtener el modelo dinámico del robot por el procedimiento de Lagrange-Euler (L-E).

Algoritmo computacional para el modelado dinámico por Lagrange-Euler.

L-E 1: Asignar a cada eslabón un sistema de referencia de acuerdo a las normas de D-H.

L-E 2: Obtener las matrices de transformación 0Ai para cada elemento i.

L-E 3: Obtener las matrices Uij definidas por:

$$\mathbf{U}_{ij} = \frac{\partial^0 \mathbf{A}_i}{\partial q_j}$$

La derivada de la matriz de D-H 0**A**i respecto de la coordenada qi puede obtenerse fácilmente de manera computacional, mediante la expresión:

$$\frac{\partial^0 \mathbf{A}_i}{\partial q_j} = \begin{cases} {}^0\mathbf{A}_{j-1}\,\mathbf{Q}_j\,{}^{j-1}\mathbf{A}_i & \text{si } j \le i \\ [0] & \text{si } j > i \end{cases}$$

$$\mathbf{Q}_i = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$ Si la articulación i es de rotación

$$\mathbf{Q}_i = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$ Si la articulación i es de trasalación

L-E 4: Obtener las matrices Uijk definidas por:

$$\mathbf{U}_{ijk} = \frac{\partial \mathbf{U}_{ij}}{\partial q_k}$$

$$\frac{\partial \mathbf{U}_{ij}}{\partial q_k} = \frac{\partial}{\partial q_k}\left(\frac{\partial\, {}^0\mathbf{A}_i}{\partial q_j}\right) = \begin{cases} {}^0\mathbf{A}_{j-1}\mathbf{Q}_j\,{}^{j-1}\mathbf{A}_{k-1}\mathbf{Q}_k\,{}^{k-1}\mathbf{A}_i & \text{si } i \geq k \geq j \\ {}^0\mathbf{A}_{k-1}\mathbf{Q}_k\,{}^{k-1}\mathbf{A}_{j-1}\mathbf{Q}_j\,{}^{j-1}\mathbf{A}_i & \text{si } i \geq j \geq k \\ [0] & \text{si } k > i \text{ o } j > i \end{cases}$$

L-E 5: Obtener las matrices de pseudoinercias **J**i para cada elemento, que vienen definidas por:

$$\mathbf{J}_i = \begin{bmatrix} \int x_i^2 dm & \int x_i y_i dm & \int x_i z_i dm & \int x_i dm \\ \int y_i x_i dm & \int y_i^2 dm & \int y_i z_i dm & \int y_i dm \\ \int z_i x_i dm & \int z_i y_i dm & \int z_i^2 dm & \int z_i dm \\ \int x_i dm & \int y_i dm & \int z_i dm & \int dm \end{bmatrix}$$

donde las integrales están extendidas al elemento i considerado, y (xi,yi,zi) son las coordenadas del diferencial de masa dm respecto al sistema de coordenadas del elemento.

L-E 6:Obtener la matriz de inercias D = [dij] cuyos elementos vienen definidos por:

$$d_{ij} = \sum_{k=(\max i,j)}^{n} \text{Traza}\left(\mathbf{U}_{kj}\mathbf{J}_k\mathbf{U}_{ki}^T\right)$$

Traza de una matriz

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

$$\operatorname{tr} A = a_{11} + a_{22} + a_{33} + a_{44}$$

con i,j = 1,2, ...,n // n : número de grados de libertad
Las matrices Ji y D son simétricas y semidefinidas positivas.

L-E 7: Obtener los términos hikm definidos por:

$$h_{ikm} = \sum_{j=\max(i,k,m)}^{n} \text{Traza}\left(\mathbf{U}_{jkm}\mathbf{J}_j\mathbf{U}_{ji}^T\right)$$

con i,k,m = 1,2, ...,n
El término hikm representa el efecto, en cuanto a fuerza o par, generado sobre el eslabón i como consecuencia del movimiento relativo entre los eslabones k y m.
Se cumple que hikm=himk y que hiii=0.

L-E 8: Obtener la matriz columna de fuerzas de Coriolis y centrípeta H = $[hi]^T$ cuyos elementos vienen definidos por:

$$h_i = \sum_{k=1}^{n}\sum_{m=1}^{n} h_{ikm}\dot{q}_k\dot{q}_m$$

L-E 9: Obtener la matriz columna de fuerzas de gravedad C = $[ci]^T$ cuyos elementos están definidos por:

$$c_i = \sum_{j=1}^{n}\left(-m_j\mathbf{g}\mathbf{U}_{ji}{}^j\mathbf{r}_j\right)$$

con i = 1,2,...,n
g: es el vector de gravedad expresado en el sistema de la base {S0} y viene expresado por (gx0 , gy0, gz0,0)
i**r**j : es el vector de coordenadas homogéneas del centro de masas del elemento j expresado en el sistema de referencia del elemento i.

L-E 10: La ecuación dinámica del sistema será:

$$\tau = \mathbf{D}\ddot{\mathbf{q}} + \mathbf{H} + \mathbf{C}$$

donde τ es el vector de fuerzas y pares motores efectivos aplicados sobre cada coordenada qi.

Desarrollo de la Práctica

Objetivos:

Realizar el análisis dinámico de la estructura mecánica del "**One Legged Robot**", y luego implementar el control de una maquina PWM en lenguaje VHDL .

Análisis Dinámico

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, velocidades de movimientos similares a las humanas....

Las ecuaciones dinámicas del 2 DOF de la figura 3 se extraen del análisis Lagrange-Euler

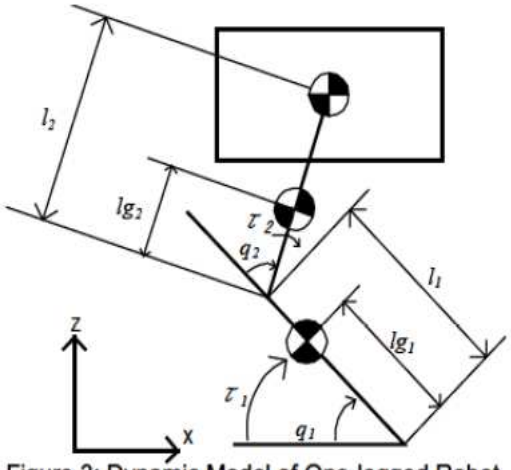


Figure 3: Dynamic Model of One-legged Robot.

Donde la ecuación final de los pares de Torque y Fuerza es la siguiente:

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

A continuación se definen las matrices cada término

$$\boldsymbol{M(q)} = \begin{bmatrix} m_1 l_{g1}^{\,2} + I_1 + m_2(l_1^{\,2} + l_{g2}^{\,2} + 2 l_1 l_{g2} c_2) + I_2 & m_2(l_{g2}^{\,2} + l_1 l_{g2} c_2) + I_2 \\ m_2(l_{g2}^{\,2} + l_1 l_{g2} c_2) + I_2 & m_2 l_{g2}^{\,2} + I_2 \end{bmatrix}$$

$$\boldsymbol{C(q,\dot{q})} = \begin{bmatrix} -2m_2 l_1 l_{g2} s_2 \dot{q}_1 \dot{q}_2 - m_2 l_1 l_{g2} s_2 \dot{q}_2^2 \\ m_2(l_{g2}^{\,2} + l_1 l_{g2} \dot{q}_1 \dot{q}_2) \end{bmatrix}$$

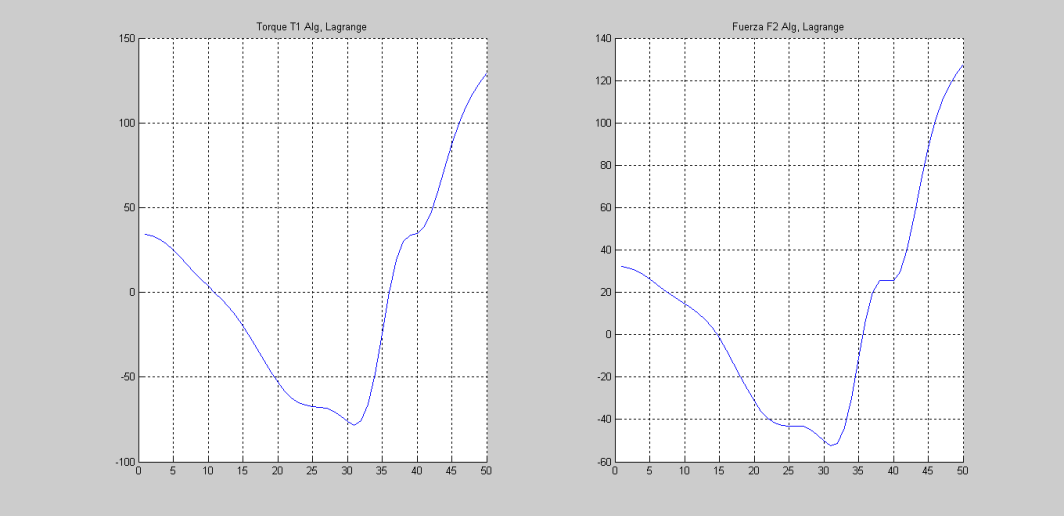
$$\boldsymbol{G(q)} = \begin{bmatrix} m_1 g l_{g1} c_1 + m_2 g(l_1 c_1 + l_{g2} c_{12}) \\ m_2 g l_{g2} c_{12} \end{bmatrix}$$

$$\boldsymbol{J(q)} = \begin{bmatrix} -l_1 s_1 - l_2 s_{12} & -l_2 s_{12} \\ l_1 c_1 + l_2 c_{12} & l_2 c_{12} \end{bmatrix}$$

Se definen las especificaciones para la resolución de las matrices dadas juntas con las de la simulación para obtener las inercias y los puntos de gravedad.

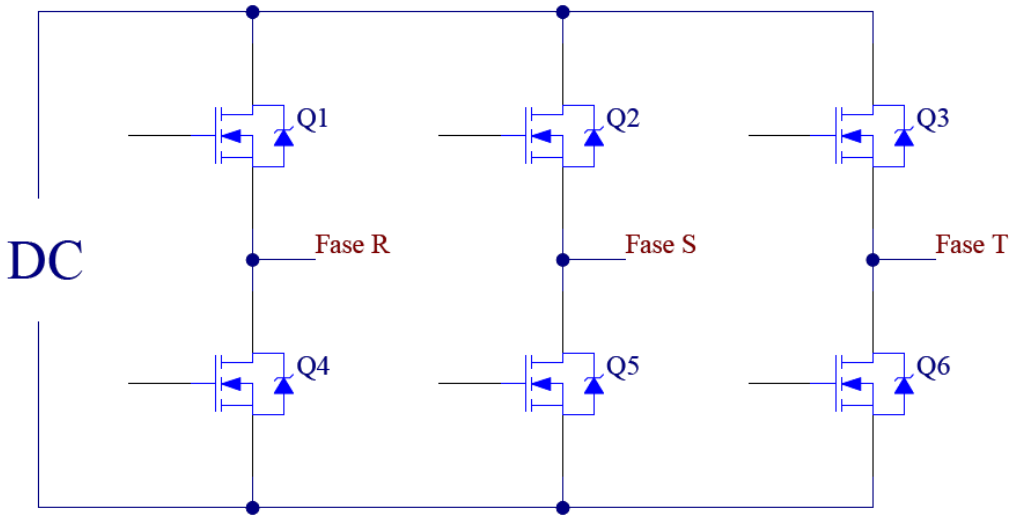
L0 = 0.18;
L1 = 0.18;
L2 = 0.13;
m0 = 0.235;
m1 = 0.465;
m2 = 0.45;
Mp = 0.185;
I1 = 2.18;
I2 = 5.103;
Lg1 = 1.785;
Lg2 = 3.456;
g = 9.8;

Dicho producto de matrices fue realizado en MATLAB debido a la gran cantidad de variables involucrada, dicho archivo se adjunta al mismo para su análisis.
Suponiendo que ambos eslabones realizan movimientos rotacionales se realiza una pequeña simulación de cómo varian el Par y la Fuerza en función del tiempo



Implementación de VHDL para el control del motor de CC

Según el esquema siguiente se implementara el código necesario para poder manejar un motor de CC mediante un proceso conocido llamado modulación por ancho de impulsos, también conocido como PWM.



En el mismo se manejaran las corrientes habilitaciones para cada fase del motor mediante la activación y desactivación de los transistores con una secuencia ordenada.

En primer caso los transistores Q1 y Q5 se activaran primero, luego Q2 y Q6, y por ultimo Q3 y Q4. Se decidió trabajar con una frecuencia de 10 KHz, que es la frecuencia estándar con la que trabajan dichos motores y para tener un periodo de 100 us para la conmutación de las fases y generar los pulsos de la modulación correspondientes.

Para tal trabajo se realizaron dos partes:
Uno que se encarga del control de la modulación, y el otro que se encarga de testear el funcionamiento del mismo, a continuación se incluyen los respectivos códigos.

Código para el control de la modulación en VHDL

```
library IEEE;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

-- Declaración de puertos de entrada y salida de la entidad diseñada --

ENTITY modulacion_pwm IS
PORT (          clock :in STD_LOGIC;
        sensor_hall_1 :in STD_LOGIC;
        sensor_hall_2 :in STD_LOGIC;
        sensor_hall_3 :in STD_LOGIC;
        pwm_1 :out STD_LOGIC;
```

```
        pwm_2 :out STD_LOGIC;
        pwm_3 :out STD_LOGIC;
        pwm_4 :out STD_LOGIC;
        pwm_5 :out STD_LOGIC;
        pwm_6 :out STD_LOGIC
);
END modulacion_pwm;


ARCHITECTURE architect_pwm OF modulacion_pwm IS

-- Declaración de señales internas --

SIGNAL active_Q1 : STD_LOGIC;
SIGNAL active_Q2 : STD_LOGIC;
SIGNAL active_Q3 : STD_LOGIC;
SIGNAL active_Q4 : STD_LOGIC;
SIGNAL active_Q5 : STD_LOGIC;
SIGNAL active_Q6 : STD_LOGIC;

BEGIN

-- Proceso que maneja la modulación pwm a la salida --

PROCESS(clock,active_Q1,active_Q2,active_Q3,active_Q4,active_Q5,active_Q6)

BEGIN

    pwm_1 <= clock and active_Q1;
    pwm_5 <= clock and active_Q5;
    pwm_2 <= clock and active_Q2;
    pwm_6 <= clock and active_Q6;
    pwm_3 <= clock and active_Q3;
    pwm_4 <= clock and active_Q4;


END PROCESS;

-- Proceso que activa los transistores según que sensor de Hall este activado --

PROCESS (sensor_hall_1,sensor_hall_2,sensor_hall_3)

BEGIN

if(sensor_hall_1 = '1') THEN
    active_Q1 <= '1';
    active_Q5 <= '1';
    active_Q2 <= '0';
    active_Q6 <= '0';
    active_Q3 <= '0';
```

```
   active_Q4 <= '0';
END IF;

if(sensor_hall_2 = '1') THEN
   active_Q1 <= '0';
   active_Q5 <= '0';
   active_Q2 <= '1';
   active_Q6 <= '1';
   active_Q3 <= '0';
   active_Q4 <= '0';
END IF;

if(sensor_hall_3 = '1') THEN
   active_Q1 <= '0';
   active_Q5 <= '0';
   active_Q2 <= '0';
   active_Q6 <= '0';
   active_Q3 <= '1';
   active_Q4 <= '1';
END IF;

END PROCESS;

END architect_pwm;
```

Código utilizado como banco de pruebas para la modulación:

```
LIBRARY ieee;
use ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

ENTITY modulacion_pwm_test_bench IS
END modulacion_pwm_test_bench;

ARCHITECTURE architect_pwm_test_bench OF modulacion_pwm_test_bench IS

-- Declaración de componentes de puertos de entrada y salida de la entidad diseñada --

COMPONENT modulacion_pwm
PORT (                  clock: in std_logic;
      sensor_hall_1 :in STD_LOGIC;
      sensor_hall_2 :in STD_LOGIC;
      sensor_hall_3 :in STD_LOGIC;
      pwm_1: out std_logic;
      pwm_2: out std_logic;
      pwm_3: out std_logic;
      pwm_4: out std_logic;
      pwm_5: out std_logic;
      pwm_6: out std_logic
);
```

```
END COMPONENT;

-- Declaración de señales internas --

SIGNAL sig_clock : std_logic;
SIGNAL sig_sensor_hall_1 :std_logic;
SIGNAL sig_sensor_hall_2 :std_logic;
SIGNAL sig_sensor_hall_3 :std_logic;
SIGNAL sig_pwm_1 :std_logic;
SIGNAL sig_pwm_2 :std_logic;
SIGNAL sig_pwm_3 :std_logic;
SIGNAL sig_pwm_4 :std_logic;
SIGNAL sig_pwm_5 :std_logic;
SIGNAL sig_pwm_6 :std_logic;
SIGNAL sig_active_Q1 :std_logic;
SIGNAL sig_active_Q2 :std_logic;
SIGNAL sig_active_Q3 :std_logic;
SIGNAL sig_active_Q4 :std_logic;
SIGNAL sig_active_Q5 :std_logic;
SIGNAL sig_active_Q6 :std_logic;
shared variable ENDSIM: boolean:=false;
constant clk_period:TIME:=100 us;


BEGIN

-- Generación de la señal de clock --

generacion_clock: 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;

-- Generación de las señales de los sensores de efecto Hall --

generacion_sensores_hall: process

BEGIN

If ENDSIM = FALSE THEN
   sig_sensor_hall_1<= '1';
```

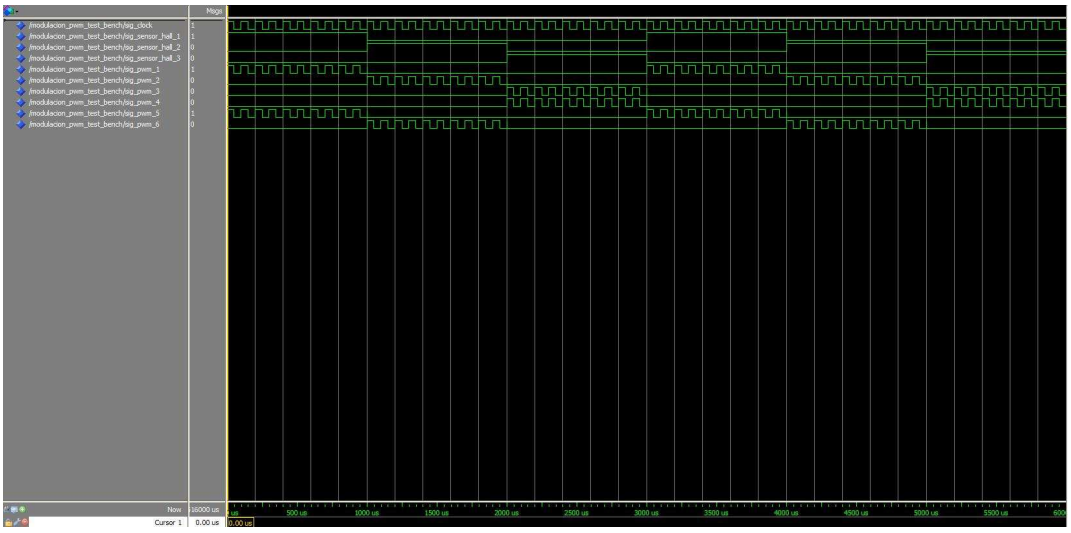
```
    sig_sensor_hall_2<= '0';
    sig_sensor_hall_3<= '0';
    wait for 1000 us;
    sig_sensor_hall_1<= '0';
    sig_sensor_hall_2<= '1';
    sig_sensor_hall_3<= '0';
    wait for 1000 us;
    sig_sensor_hall_1<= '0';
    sig_sensor_hall_2<= '0';
    sig_sensor_hall_3<= '1';
    wait for 1000 us;
else
    wait;
end if;
end process;

-- Referencia de las señales de la entidad diseñada para el banco de pruebas --

inicializacion_pwm : modulacion_pwm
PORT MAP(   clock => sig_clock,
        sensor_hall_1 => sig_sensor_hall_1,
        sensor_hall_2 => sig_sensor_hall_2,
        sensor_hall_3 => sig_sensor_hall_3,
        pwm_1 => sig_pwm_1,
        pwm_2 => sig_pwm_2,
        pwm_3 => sig_pwm_3,
        pwm_4 => sig_pwm_4,
        pwm_5 => sig_pwm_5,
        pwm_6 => sig_pwm_6
);

END architect_pwm_test_bench;
```

Resultado de la simulación en ModelSim:



Conclusión Final:

Con la primer parte del TP pudimos aprender un poco sobre el comportamiento de los One Legged Robot" y comprobar la validez del método de Lagrange-Euler aplicándolo a estos para la resolución de este TP , y volvemos a ver la potencia de una herramienta como el MATLAB, para facilitar todo el trabajo y minimizar los tiempos de desarrollo.

De la segunda parte vemos que es relativamente simple la implementación en VHDL (al menos en nuestro TP) y la programación no trae muchas complicaciones, mas allá de poder tornarse un poco larga, ya que hay definir cada señal, el comportamiento de cada entrada, de cada salida, etc. Pero la gran ventaja de poder realizar nuestro propio programa de prueba y verificar el comportamiento de cada componente programado, y como responde nuestro diseño. Dicho de otra manera, usamos software que simula hardware, para probar hardware, programado por software.