ROBOTICA – R6055 - 2010	
Profesor: M.aS. Ing. Hernan Giannetta	**JTP. : Ing. Damian Granzella**
T.P. N°: 2	**Análisis Dinámico de un Robot e implementación en FPGA**
GRUPO N°:	**Integrantes:** **Alonso, Gustavo.** **Montalti, Pablo.**
Responsable : **(e-mail del responsable)**	**Alonso Gustavo; guagea@hotmail.com** **Montalti Pablo; pmontalti@hotmail.com**
Fecha de Entrega: 09/07/10	**Fecha de Aprobación:**

UNIVERSIDAD TECNOLOGICA NACIONAL

Alonso Gustavo gusgea@hotmail.com
Montalti Pablo pmontalti@hotmail.com

1) Introducción a la dinámica del robot

El modelo dinamico de un robot tiene por objetivo conocer la relacion entre el movimiento del robot y las fuerzas implicadas en el mismo.

- Esta relacion se obtiene mediante el denominado modelo dinamico, que relaciona matematicamente:
- La localizacion del robot definida por sus variables articulares o por las coordenadas de localizacion de su extremo, y sus derivadas: velocidad y aceleracion.
- Las fuerzas y pares aplicados en las articulaciones (o en el extremo del robot).
- Los parametros dimensionales del robot, como longitud, masas e inercias de sus elementos.

La obtencion de este modelo para mecanismos de uno o dos grados de libertad no es excesivamente compleja, pero a medida que el numero de grados de libertad aumenta, el planteamiento y obtencion del modelo dinamico se complica enormemente.

Por este motivo no siempre es posible obtener un modelo dinamico expresado de una forma cerrada, esto es, mediante una serie de ecuaciones, normalmente de tipo diferencial de 2o orden, cuya integracion permita conocer que movimiento surge al aplicar unas fuerzas o que fuerzas hay que aplicar para obtener un movimiento determinado.

El modelo dinamico debe ser resuelto entonces de manera iterativa mediante la utilizacion de un procedimiento numerico.

El problema de la obtencion del modelo dinamico de un robot es, por lo tanto, uno de los aspectos mas complejos de la robotica, lo que ha llevado a ser obviado en numerosas ocasiones. Sin embargo, esimprescindible para conseguir los siguientes fines:

1. Simulacion del movimiento del robot.
2. Diseno y evaluacion de la estructura mecanica del robot.
3. Dimensionamiento de los actuadores.
4. Diseno y evaluacion del control dinamico del robot.

2) Modelado dinámico del robot

2.1) Modelado por equilibrio de fuerzas

La obtencion del modelo dinamico de un mecanismo, y en particular de un robot, se basa fundamentalmente en el planteamiento del equilibrio de fuerzas establecido en la segunda ley de Newton, o su equivalente para movimientos de rotacion, la denominada ley de Euler:

Alonso Gustavo gusgea@hotmail.com
Montalti Pablo pmontalti@hotmail.com

$$\sum \boldsymbol{F} = m\,\dot{\boldsymbol{v}}$$

$$\sum \boldsymbol{T} = \boldsymbol{I}\,\dot{\omega} + \omega \times (\boldsymbol{I}\,\omega)$$

Se tiene asi que el planteamiento del equilibrio de fuerzas y pares que intervienen sobre el robot se obtienen los denominados modelos dinamicos directos e inversos.

- **Modelos dinamico directo**: expresa la evolucion temporal de las coordenadas articulares del robot en funcion de las fuerzas y pares que intervienen.
- **Modelo dinamico inverso**: expresa las fuerzas y pares que intervienen en funcion de la evolucion de de las coordenadas articulares y sus derivadas.

Se debe tener en cuenta que intervienen las fuerzas de :

1. Inercia.
2. Gravedad.
3. Coriolis. (debida al movimiento relativo existente entre los diversos elementos)
4. Centripeta (que depende de la configuracion instantanea del manipulador)

2.2) Modelado por consideraciones energéticas

Como planteamiento alternativo para la obtencion del modelo se puede usar la formacion lagrangiana, basada en consideraciones energeticas.
Este planteamiento es mas sistematico que el anterior, y por lo tanto facilita enormemente la formacion de un modelo tan complejo como el de un robot.

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} = \tau$$

$$\mathcal{L} = k - u$$

$q_1 = Coordenadas\ generalizadas\ (en\ este\ caso\ las\ articulaciones).$
$\tau = Vector\ de\ fuerzas\ y\ pares\ aplicados\ en\ las\ q_i.$
$\mathcal{L} = Función\ Lagrangiana.$
$k = Energía\ cinética.$
$u = Energía\ potencial.$

3) Métodos para el modelado dinámico del robot

3.1) Metodo Lagrange - Euler.

Uicker en 1965 [UICKER-65] [UICKER-64] utilizo la representacion de D-H basada en las matrices de transformacion homogenea para formular el modelo dinamico de un robot mediante la ecuacion de Lagrange.

- Se trata de un procedimiento ineficiente desde el punto de vista computacional.
- Puede comprobarse que el algoritmo es de un orden de complejidad computacional O(n4) es decir, el numero de operaciones a realizar crece con la potencia 4 del numero de grados de libertad.
- 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 continuacion al algoritmo a seguir para obtener el modelo dinamico del robot por el procedimiento de Lagrange-Euler (L-E).

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

L-E 2 : Obtener las matrices de transformacion ${}^0\mathbf{A}_i$ para cada elemento i.

L-E 3 : Obtener las matrices $\mathbf{U}_{ij}$ definidas por:

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

L-E 4 : Obtener las matrices $\mathbf{U}_{ijk}$ definidas por:

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

L-E 5 : Obtener las matrices de pseudoinercias $\mathbf{J}_i$ para cada elemento, que vienen definidas por:

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

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

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

Alonso Gustavo gusgea@hotmail.com
Montalti Pablo pmontalti@hotmail.com

con i,j = 1,2, ...,n
n : numero de grados de libertad

L-E 7 :Obtener los terminos h_{ikm} 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

L-E 8 :Obtener la matriz columna de fuerzas de Coriolis y centripeta 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 estan 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

L-E 10 : La ecuacion dinamica del sistema sera:

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

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

3.2) Metodo Newton - Euler.

La formulacion de Newton-Euler utiliza las ecuaciones de equilibrio de fuerzas y
torques (pares).
Un adecuado desarrollo de estas ecuaciones conduce a una formulacion recursiva en la que se obtienen la :

1. posicion,
2. velocidad y ,
3. aceleracion

del eslabon i referidos a la base del robot a partir de los correspondientes valores de (posicion, Vel y Asc) del eslabon i-1 y del movimiento relativo de la articulacion i.
El algoritmo se basa en operaciones vectoriales (con productos escalares y vectoriales entre magnitudes vectoriales, y productos de matrices con vectores) siendo mas eficiente en comparacion con las operaciones matriciales asociadas a la formulacion Lagrangiana.
De hecho, el orden de complejidad computacional de la formulacion recursiva de Newton-Euler es O(n) lo que indica que depende directamente del numero de grados de libertad.
Se presenta a continuacion al algoritmo a seguir para obtener el modelo dinamico del robot por el procedimiento de Newton-Euler (N-E).

N-E 1. Asignar a cada eslabon un sistema de referencia de acuerdo con las normas de D-H.

N-E 2. Obtener las matrices de rotacion ${}^{i-1}\mathbf{R}_i$ y sus inversas ${}^{i}\mathbf{R}_{i-1} = ({}^{i-1}\mathbf{R}_i)^{-1} = ({}^{i-1}\mathbf{R}_i)^{T}$

$$ {}^{i-1}\mathbf{R}_i = \begin{bmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i S\theta_i \\ S\theta_i & C\alpha_i C\theta_i & -S\alpha_i C\theta_i \\ 0 & S\alpha_i & C\alpha_i \end{bmatrix} $$

N-E 3. Establecer las condiciones iniciales para el sistema de base S0..

$${}^{0}\omega_0: \text{ velocidad angular} = [0,0,0]^T$$

$${}^{0}\dot{\omega}_0: \text{ aceleracion angular} = [0,0,0]^T$$

$${}^{0}\mathbf{v}_0: \text{ velocidad lineal} = [0,0,0]^T$$

$${}^{0}\dot{\mathbf{v}}_0: \text{ aceleracion lineal} = [g_x, g_y, g_z]^T$$

Para el extremo del robot se conocera la fuerza y el par ejercidos externamente ${}^{n+1}\mathbf{f}_{n+1}$ y ${}^{n+1}\mathbf{n}_{n+1}$

${}^{1}\mathbf{z}_0 = [0,0,1]^T$
${}^{i}\mathbf{p}_i$ = Coordenadas del origen del sistema {Si } respecto a {Si-1}= [ai , di Si , di Ci]
${}^{i}\mathbf{s}i$ = Coordenadas del centro de masas del eslabon i respecto del sistema { Si }
${}^{i}\mathbf{I}i$ = Matriz de inercia del eslabon i respecto del su centro de masas expresado en { Si }

N-E 4. Obtener la velocidad angular del sistema { Si }

$$ {}^{i}\omega_i = \begin{cases} {}^{i}\mathbf{R}_{i-1}\left({}^{i-1}\omega_{i-1} + \mathbf{z}_0\,\dot{q}_i\right) & \text{si el eslabón i es de rotación} \\ {}^{i}\mathbf{R}_{i-1}\,{}^{i-1}\omega_{i-1} & \text{si el eslabón i es de traslación} \end{cases} $$

N-E 5. Obtener la aceleracion angular del sistema { Si }.

$$ {}^{i}\dot{\omega}_i = \begin{cases} {}^{i}\mathbf{R}_{i-1}\left({}^{i-1}\dot{\omega}_{i-1} + \mathbf{z}_0\,\ddot{q}_i\right) + {}^{i-1}\omega_{i-1} \times \mathbf{z}_0\,\dot{q}_i & \text{si el eslabón i es de rotación} \\ {}^{i}\mathbf{R}_{i-1}\,{}^{i-1}\dot{\omega}_{i-1} & \text{si el eslabón i es de traslación} \end{cases} $$

N-E 6. Obtener la aceleracion lineal del sistema i.

Alonso Gustavo gusgea@hotmail.com
Montalti Pablo pmontalti@hotmail.com

$$ {}^{i}\dot{\mathbf{v}}_i = \begin{cases} {}^{i}\dot{\boldsymbol{\omega}}_i \times {}^{i}\mathbf{p}_i + {}^{i}\boldsymbol{\omega}_i \times \left({}^{i}\boldsymbol{\omega}_i \times {}^{i}\mathbf{p}_i \right) + {}^{i}\mathbf{R}_{i-1}\, {}^{i-1}\dot{\mathbf{v}}_{i-1} & \text{si el eslabón i es de rotación} \\ {}^{i}\mathbf{R}_{i-1}\left(\mathbf{z}_0 \ddot{q}_i + {}^{i-1}\dot{\mathbf{v}}_{i-1} \right) + {}^{i}\dot{\boldsymbol{\omega}}_i \times {}^{i}\mathbf{p}_i + 2\,{}^{i}\boldsymbol{\omega}_i \times {}^{i}\mathbf{R}_{i-1}\mathbf{z}_0 \dot{q}_i + & \\ \quad + {}^{i}\boldsymbol{\omega}_i \times \left({}^{i}\boldsymbol{\omega}_i \times {}^{i}\mathbf{p}_i \right) & \text{si el eslabón i es de traslación} \end{cases} $$

N-E 7. Obtener la aceleracion lineal del centro de gravedad del eslabon i:
Para i = n ... 1 realizar los pasos 8 al 10.

$$ {}^{i}\mathbf{a}_i = {}^{i}\dot{\boldsymbol{\omega}}_i \times {}^{i}\mathbf{S}_i + {}^{i}\boldsymbol{\omega}_i \times \left({}^{i}\boldsymbol{\omega}_i \times {}^{i}\mathbf{s}_i \right) + {}^{i}\dot{\mathbf{v}}_i $$

N-E 8. Obtener la fuerza ejercida sobre el eslabon i:

$$ {}^{i}\mathbf{f}_i = {}^{i}\mathbf{R}_{i+1}\, {}^{i+1}\mathbf{f}_{i+1} + m_i\, {}^{i}\mathbf{a}_i $$

N-E 9. Obtener el par ejercido sobre el eslabon i:

$$ {}^{i}\mathbf{n}_i = {}^{i}\mathbf{R}_{i+1}\left[{}^{i+1}\mathbf{n}_i + \left({}^{i+1}\mathbf{R}_i\, {}^{i}\mathbf{p}_i \right) \times {}^{i+1}\mathbf{f}_{i+1} \right] + \left({}^{i}\mathbf{p}_i + {}^{i}\mathbf{s}_i \right) \times m_i\, {}^{i}\mathbf{a}_i + {}^{i}\mathbf{I}_i\, {}^{i}\dot{\boldsymbol{\omega}}_i + {}^{i}\boldsymbol{\omega}_i \times \left({}^{i}\mathbf{I}_i\, {}^{i}\boldsymbol{\omega}_i \right) $$

N-E 10. Obtener la fuerza o torque (par) aplicado a la articulacion i.
Donde ζ es el torque (par) o fuerza efectiva (torque motor menos pares de rozamiento o perturbacion).

$$ \tau_i = \begin{cases} {}^{i}\mathbf{n}_i^{T}\, {}^{i}\mathbf{R}_{i-1}\mathbf{z}_0 & \text{si el eslabón i es de rotación} \\ {}^{i}\mathbf{f}_i^{T}\, {}^{i}\mathbf{R}_{i-1}\mathbf{z}_0 & \text{si el eslabón i es de traslación} \end{cases} $$

3.3) Metodo de variables de estado:

Las variables de estado naturales del sistema seran las posiciones y velocidades de cada una de las articulaciones, siendo por lo tanto el vector de estado:

$$ [\mathbf{q}, \dot{\mathbf{q}}]^{T}. $$

Por lo tanto la expresion dinámica resulta:

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

$$ \ddot{\mathbf{q}} = \mathbf{D}^{-1}[\tau - \mathbf{N}] $$

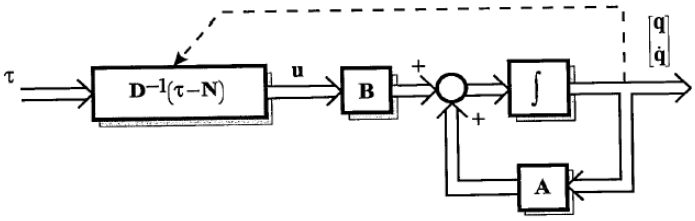
con **N** = **H** + **C** y haciendo uso del vector de estado:

$$ \frac{d}{dt}\begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{q}} \\ -\mathbf{D}^{-1}\mathbf{N} \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{D}^{-1} \end{bmatrix} \tau $$

donde **D**-1 es funcion de q y N es funcion de q y la derivada primera de q.

La expresion anterior puede ponerse tambien de la siguiente forma:

Alonso Gustavo gusgea@hotmail.com
Montalti Pablo pmontalti@hotmail.com

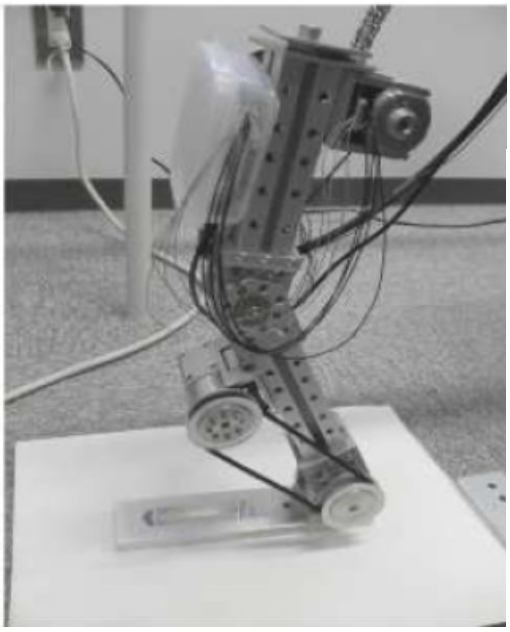


$$\mathbf{A} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} \text{ matriz (2n x 2n)} \quad \text{y} \quad \mathbf{B} = \begin{bmatrix} 0 \\ I \end{bmatrix} \text{ matriz (2n x n)}$$

$$\frac{d}{dt}\begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} \cdot \mathbf{u}$$

$$\mathbf{u} = \mathbf{D}^{-1}(\tau - \mathbf{N})$$

4) análisis dinámico de la estructura mecánica del "One Legged Robot":



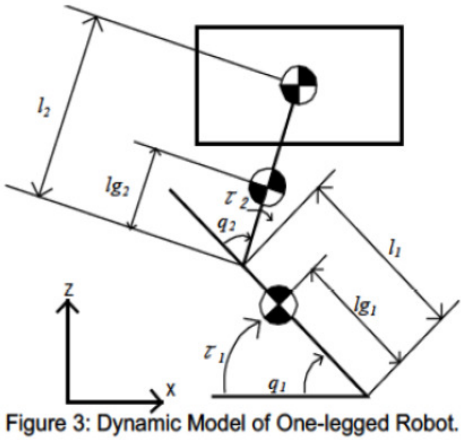


Figure 3: Dynamic Model of One-legged Robot.

En el archivo anexo "Lagrange-Euler.doc" se puede encontrar el desarrollo del método correspondiente cuya conclusión ha sido la siguiente:

Ecuación dinámica del sistema:

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = D.\begin{bmatrix} ** \\ q_1 \\ ** \\ q_2 \end{bmatrix} + H + C$$

$$C = \begin{bmatrix} -m_1.g.C1.Lg_1 - m_2.(g.C12.Lg_2 + g.C1.L1) \\ m_2.g.C12.Lg_2 \end{bmatrix}$$

Alonso Gustavo gusgea@hotmail.com
Montalti Pablo pmontalti@hotmail.com

$$H = \begin{bmatrix} L_{g2}.m_2.L1.S2.(\overset{*}{q}_1.\overset{*}{q}_2 + \overset{*}{q}_2.\overset{*}{q}_1 + \overset{*}{q}_2\overset{*}{q}_2) \\ -L_{g2}.m_2.L1.S2.\overset{*}{q}_1.\overset{*}{q}_1 \end{bmatrix}$$

$$D.\begin{bmatrix} \overset{**}{q}_1 \\ \overset{**}{q}_2 \end{bmatrix} = \begin{bmatrix} m_1.L_{g1}{}^2 + m_2(L_{g2}{}^2 + 2.L_{g2}.L1.C2 + L1^2) & L_{g2}{}^2.m_2 + L_{g2}.m_2.L1.C2 \\ L_{g2}{}^2.m_2 + L_{g2}.m_2.L1.C2 & m_2.L_{g2}{}^2 \end{bmatrix}.\begin{bmatrix} \overset{**}{q}_1 \\ \overset{**}{q}_2 \end{bmatrix}$$

$$\begin{aligned} &Tabla\ de\ valores: \\ &m_1 = 0.465\,Kg \\ &m_2 = 0.450\,Kg \\ &L1 = 0.18\,m \\ &L_{g1} = 0.09\,m \\ &L2 = 0.13\,m \\ &L_{g2} = 0.065\,m \\ &g = 9.8\,m/seg^2 \end{aligned}$$

$$C = \begin{bmatrix} -1.20393 * C1 - 0.28665 * C12 \\ 0.28665 * C12 \end{bmatrix}$$

$$H = \begin{bmatrix} 5.265x10^{-3} * S2.(\overset{*}{q}_1.\overset{*}{q}_2 + \overset{*}{q}_2.\overset{*}{q}_1 + \overset{*}{q}_2\overset{*}{q}_2) \\ 5.265x10^{-3} * S2.\overset{*}{q}_1.\overset{*}{q}_1 \end{bmatrix}$$

$$D.\begin{bmatrix} \overset{**}{q}_1 \\ \overset{**}{q}_2 \end{bmatrix} = \begin{bmatrix} 20.24775x10^{-3} + 10.53x10^{-3} * C2 & 1.90125x10^{-3} + 5.265x10^{-3} * C2 \\ 1.90125x10^{-3} + 5.265x10^{-3} * C2 & 1.90125x10^{-3} \end{bmatrix}.\begin{bmatrix} \overset{**}{q}_1 \\ \overset{**}{q}_2 \end{bmatrix}$$

Algunas simulaciones del movimiento y torques necesarios pueden encontrarse en el archivo mencionado.

5) Conclusión del análisis dinámico del "One Legged Robot":

Tratándose de un sistema de 2 GDL, vemos la complejidad de las expresiones matriciales resultantes.

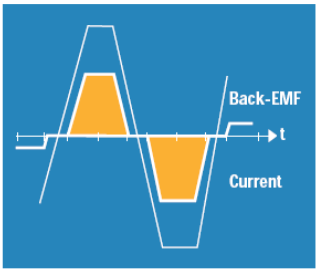
Se ha propuesto un ejemplo de un movimiento de "flexión", proponiendo una trayectoria de los ángulos *q1* y *q2* tipo rampa, y por medio del modelado dinámico inverso se obtuvo la distribución de los torques necesarios. Es de llamar la atención, que los torques tienen magnitudes no muy grandes, según los valores propuestos de la estructura mecánica. Es esperable que los torques sean opuestos en sentido, dado que el movimiento de la pierna así lo exige, pero es una incógnita el hecho que los torques cambian de signo, es decir inician en un sentido y luego cambian luego de un determinado tiempo.

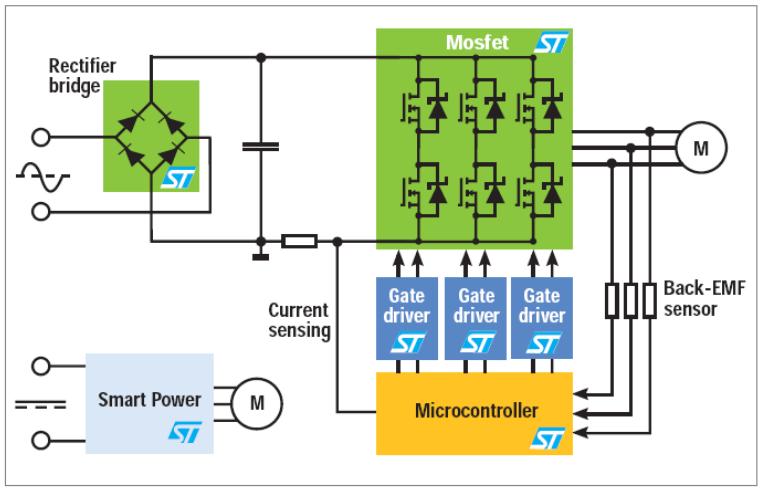
No se probó realizar un ejemplo de dinámica directa, pues vemos que las ecuaciones diferenciales que resultan son de una complejidad considerable.

Alonso Gustavo gusgea@hotmail.com
Montalti Pablo pmontalti@hotmail.com

6) CONTROL DE MOTOR BRUSHLESS MEDIANTE FPGA:

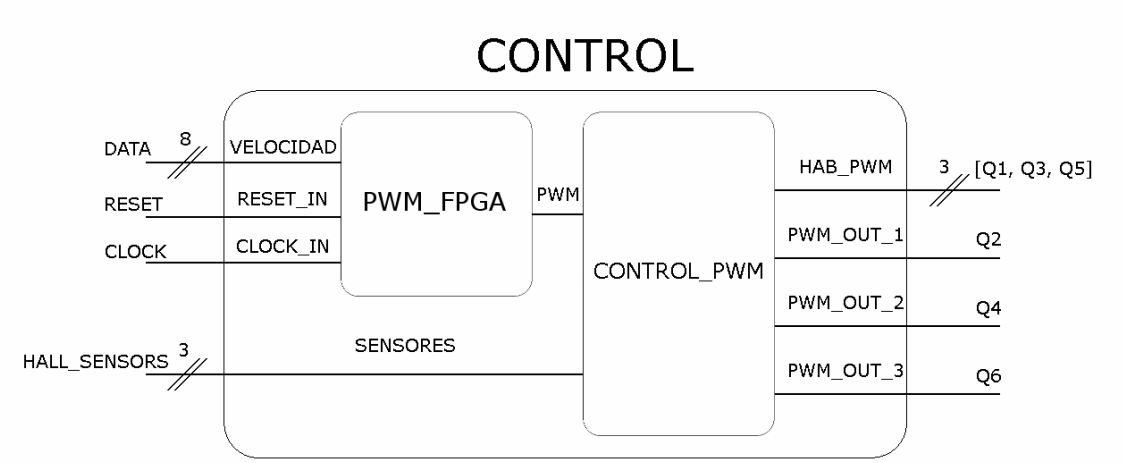
Partimos del esquema básico en el cuál se basa este tipo de control:





High-frequency three-phase brushless DC motor drive

Nosotros implementaremos este control mediante el siguiente diagrama en bloques:



PWM_FPGA

El bloque interno PWM_FPGA es el mismo utilizado en el ejemplo hecho por Atmel (subido al campus virtual como *pwm_fpga.vhd*), el cuál es el encargado de general la señal pwm de acuerdo a la velocidad indicada. El código del mismo es el siguiente:

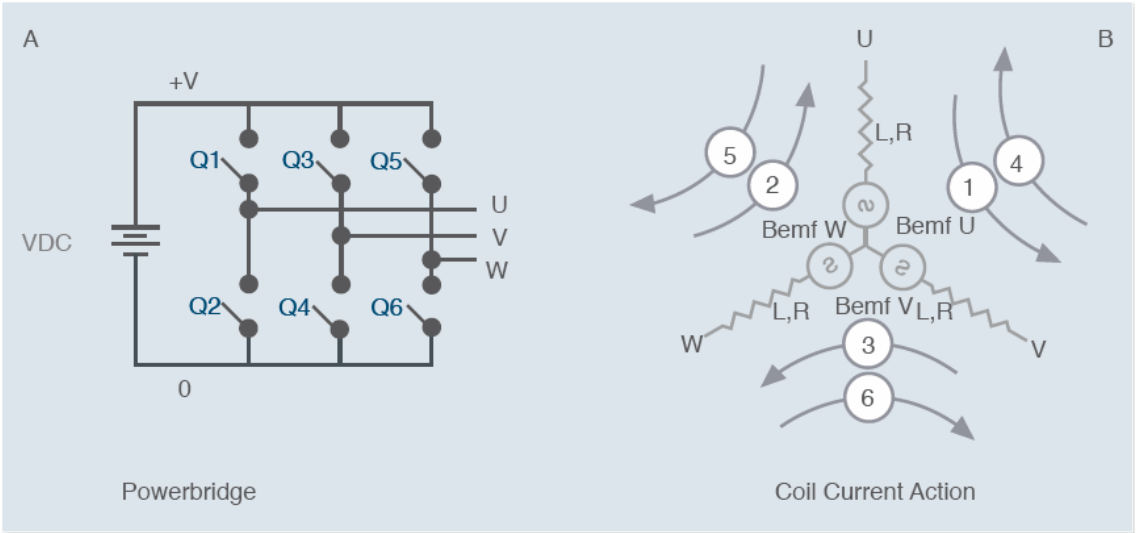
```
library IEEE;
USE   ieee.std_logic_1164.all;
USE     ieee.std_logic_arith.all ;
USE   work.user_pkg.all;


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;
```

CONTROL_PWM

EL bloque de CONTROL_PWM es el encargado de manejar el puente H de acuerdo a lo que cada sensor de efecto Hall sensa y la señal pwm generada por el bloque anterior. Esto lo hace siguiendo la siguiente tabla de verdad y circuito:



Nº de etapa	Sensor 1	Sensor 2	Sensor 3	Llaves activas
1	1	0	1	Q1 y Q4
2	0	0	1	Q1 y Q6
3	0	1	1	Q3 y Q6
4	0	1	0	Q3 y Q2
5	1	1	0	Q5 y Q2
6	1	0	0	Q5 y Q4

NOTA: El sentido de giro es siempre en el de las agujas del reloj. Para el trabajo no se tuvo en cuenta el sentido antihorario.

El código de este bloque es el siguiente:

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

ENTITY Control_PWM IS
PORT    (     pwm_in               : in STD_LOGIC;
                hall_sens       : in STD_LOGIC_VECTOR( 2 downto 0);
                HAB_PWM_OUT   : out STD_LOGIC_VECTOR (2 downto 0);
                Q2, Q4, Q6   : out STD_LOGIC
          );
END Control_PWM;

ARCHITECTURE arch_control OF Control_PWM IS
  BEGIN
    PROCESS (pwm_in, hall_sens)
    BEGIN

    CASE hall_sens IS
      WHEN "101" =>
        HAB_PWM_OUT <= "100";
        Q2 <= '0';
        Q4 <= pwm_in;
        Q6 <= '0';
      WHEN "001" =>
        HAB_PWM_OUT <= "100";
        Q2 <= '0';
        Q4 <= '0';
        Q6 <= pwm_in;
      WHEN "011" =>
        HAB_PWM_OUT <= "001";
        Q2 <= '0';
        Q4 <= '0';
        Q6 <= pwm_in;
      WHEN "010" =>
        HAB_PWM_OUT <="001";
        Q2 <= pwm_in;
        Q4 <= '0';
        Q6 <= '0';
      WHEN "110" =>
        HAB_PWM_OUT <="010";
        Q2 <= pwm_in;
        Q4 <= '0';
```

```
        Q6 <= '0';
      WHEN "100" =>
        HAB_PWM_OUT <="010";
        Q2 <= '0';
        Q4 <= pwm_in;
        Q6 <= '0';
      WHEN OTHERS =>
        HAB_PWM_OUT <= "000";
        Q2 <= '0';
        Q4 <= '0';
        Q6 <= '0';
    END CASE;
  END PROCESS;
END arch_control;
```

CONTROL

Estos dos bloques se unieron para formar un único bloque denominado CONTROL el cual tiene como único propósito simplificar el manejo de los dos bloques anteriores. La descripción es la siguiente:

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

ENTITY Control IS
  PORT  ( Velocidad         : in STD_LOGIC_VECTOR ( 7 downto 0 );
          Clock_IN          : in STD_LOGIC;
          Reset_IN          : in STD_LOGIC;
          Sensores          : in STD_LOGIC_VECTOR (2 downto 0);
          HAB_PWM           : out STD_LOGIC_VECTOR (2 downto 0);
          PWM_OUT_1, PWM_OUT_2, PWM_OUT_3 : OUT STD_LOGIC
        );
END Control;

ARCHITECTURE ControlArch OF Control 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;

  COMPONENT Control_PWM
  PORT      (      pwm_in               : in STD_LOGIC;
          hall_sens       : in STD_LOGIC_VECTOR( 2 downto 0);
          HAB_PWM_OUT   : out STD_LOGIC_VECTOR (2 downto 0);
          Q2, Q4, Q6  : out STD_LOGIC
       );
  END COMPONENT;

  SIGNAL pwm_SIG : STD_LOGIC;

  BEGIN
```

```
--************************************
-- Describo interconexion de los bloques
--************************************
  BloquePwm : pwm_fpga
    PORT MAP (
      clock => Clock_IN,
      reset => Reset_IN,
      data_value => Velocidad,
      pwm => pwm_SIG
      );

  BloqueControl : Control_PWM
    PORT MAP (
      pwm_in => pwm_SIG,
      hall_sens => Sensores,
      HAB_PWM_OUT => HAB_PWM,
      Q2 => PWM_OUT_1,
      Q4 => PWM_OUT_2,
      Q6 => PWM_OUT_3
      );

END ControlArch;
```

7) TEST BENCH

Para poder graficar las señales de salida, modificamos el archivo *pwm_pretb.vhd* incluido en el ejemplo subido al Campus y agregamos las señales nuevas introducidas por nuestros módulos. En el mismo generamos dos procesos diferentes:

1- El primero de ellos genera diferentes velocidades cada 50µs. 3 diferentes velocidades en total, que se repiten 6 veces para poder graficar los 6 pasos del motor
2- El segundo proceso esta destinado a simular los sensores de efecto hall para recorer los 6 pasos del motor

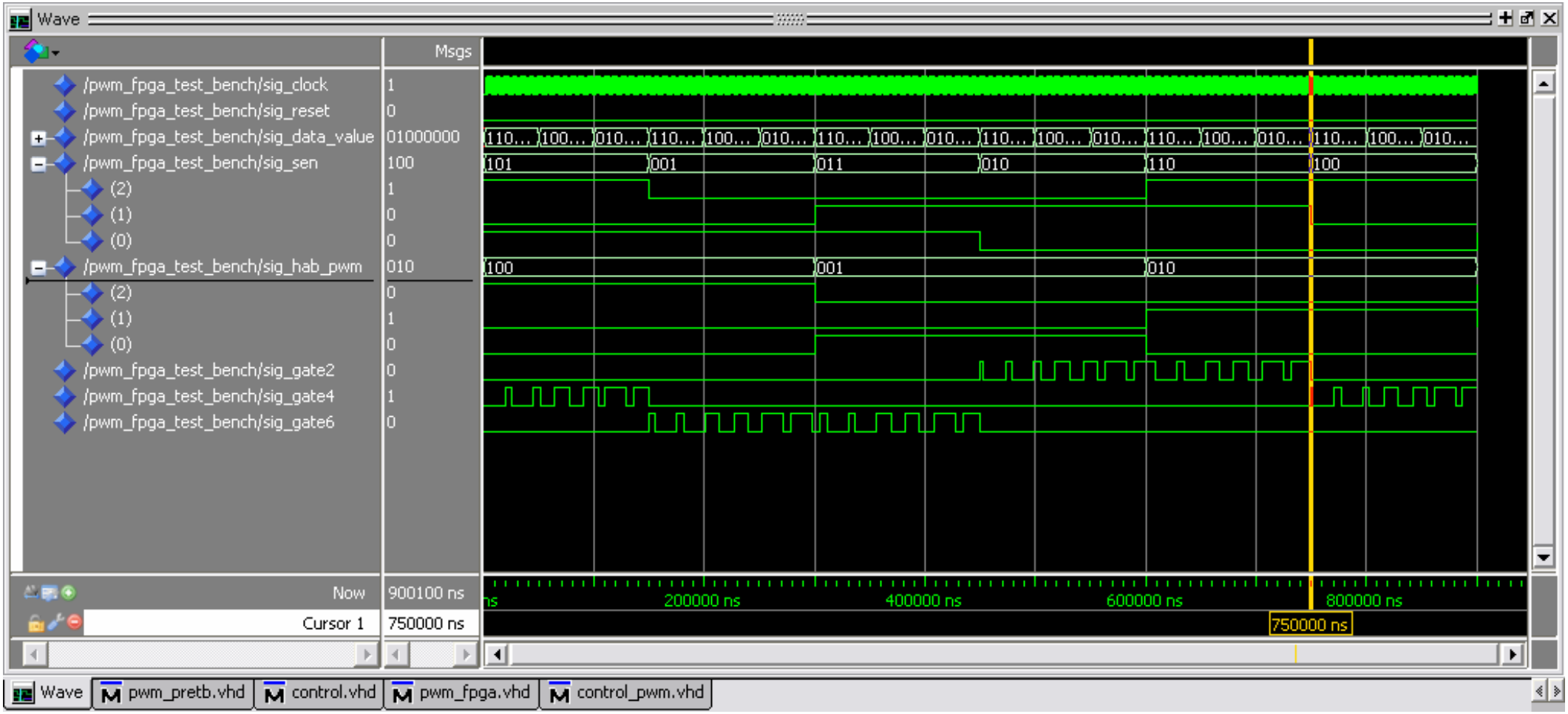
La descripción de este test bench es la siguiente:

```
--*******************************************************
--    Testbench File for design pwm_fpga produced by
--          Atmel System Designer on
--          May 25, 2001 1:54:17 pm
--*******************************************************

LIBRARY ieee;

use ieee.std_logic_1164.all;


ENTITY pwm_fpga_test_bench IS

END pwm_fpga_test_bench;

ARCHITECTURE arch_test_bench OF pwm_fpga_test_bench IS

COMPONENT Control
PORT  ( Velocidad         : in STD_LOGIC_VECTOR ( 7 downto 0 );
        Clock_IN          : in STD_LOGIC;
        Reset_IN          : in STD_LOGIC;
        Sensores          : in STD_LOGIC_VECTOR (2 downto 0);
        HAB_PWM           : out STD_LOGIC_VECTOR (2 downto 0);
        PWM_OUT_1, PWM_OUT_2, PWM_OUT_3 : OUT STD_LOGIC
      );
END COMPONENT;

-- Internal signal declaration

SIGNAL sig_clock       : std_logic;
SIGNAL sig_reset       : std_logic;
SIGNAL sig_data_value   : std_logic_vector(7 downto 0);
SIGNAL sig_sen        : std_logic_vector (2 downto 0);
SIGNAL sig_hab_pwm    : std_logic_vector (2 downto 0);
SIGNAL sig_gate2      : std_logic;
SIGNAL sig_gate4      : std_logic;
SIGNAL sig_gate6      : std_logic;

-- SIGNAL sig_pwm : std_logic;
shared variable ENDSIM: boolean:=false;
  constant clk_period:TIME:=100 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_pwm_fpga : Control
PORT MAP(
            Clock_IN    => sig_clock,
            Reset_IN    => sig_reset,
            Velocidad   => sig_data_value,
            Sensores => sig_sen,
    HAB_PWM => sig_hab_pwm,
            PWM_OUT_1 => sig_gate2,
            PWM_OUT_2 => sig_gate4,
            PWM_OUT_3 => sig_gate6
      );

stimulus_process: PROCESS

BEGIN

      sig_reset <= '1';
      wait for 100 ns;
      sig_reset <= '0';

      sig_data_value <= "11000000";
      wait for 50 us;
      sig_data_value <= "10000000";
      wait for 50 us;
      sig_data_value <= "01000000";
      wait for 50 us;
  sig_data_value <= "11000000";
      wait for 50 us;
      sig_data_value <= "10000000";
      wait for 50 us;
      sig_data_value <= "01000000";
      wait for 50 us;
      sig_data_value <= "11000000";
      wait for 50 us;
      sig_data_value <= "10000000";
      wait for 50 us;
      sig_data_value <= "01000000";
      wait for 50 us;
      sig_data_value <= "11000000";
      wait for 50 us;
      sig_data_value <= "10000000";
      wait for 50 us;
      sig_data_value <= "01000000";
      wait for 50 us;
```

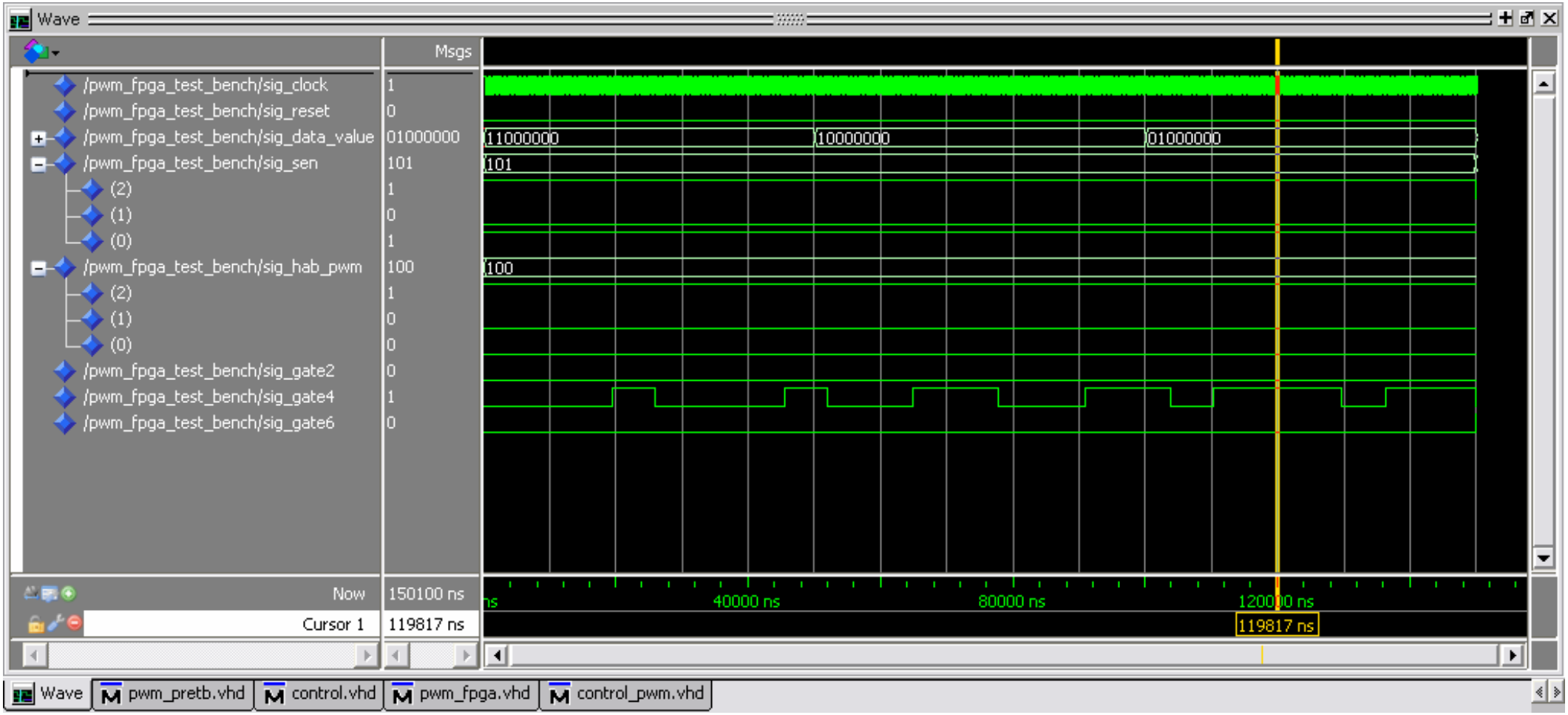
```
  sig_data_value <= "11000000";
      wait for 50 us;
      sig_data_value <= "10000000";
      wait for 50 us;
      sig_data_value <= "01000000";
      wait for 50 us;
      sig_data_value <= "11000000";
      wait for 50 us;
      sig_data_value <= "10000000";
      wait for 50 us;
      sig_data_value <= "01000000";
      wait for 50 us;
wait;


 END PROCESS stimulus_process;

 control_process : PROCESS

 BEGIN
   if (sig_reset = '1') then
     sig_sen   <= "000";
     wait for 100ns;
   else
      sig_sen   <= "101";
      wait for 150us;
      sig_sen   <= "001";
      wait for 150us;
      sig_sen   <= "011";
      wait for 150us;
      sig_sen   <= "010";
      wait for 150us;
      sig_sen   <= "110";
      wait for 150us;
      sig_sen   <= "100";
      wait for 150us;
  end if;

END PROCESS;

END arch_test_bench;
```

8) SIMULACIÓN:



Alonso Gustavo gusgea@hotmail.com
Montalti Pablo pmontalti@hotmail.com

Para observar más en detalle las diferentes velocidades ampliamos el primer paso del motor:



9) Conclusiones de la simulación:

Podemos observar en los gráficos de simulación la variación del ancho de pulso a medida que cambiamos la velocidad a la cuál queremos que gire el motor. Todo esto se suma al sensado de las diferentes posiciones del motor mediante los sensores de efecto Hall, pasando de esta forma por todos los pasos del motor a la vez que se van alternando los Mosfet.

Esto comprueba lo visto en la teoría de otras materias así como el facil manejo que permite el lenguaje VHDL en cuanto a dispositivos lógicos. Si bien el ejemplo usado no es de gran complejidad, este lenguaje descriptivo permite una infinidad de implementaciones en el campo ingenieril.

Alonso Gustavo gusgea@hotmail.com
Montalti Pablo pmontalti@hotmail.com