UNIVERSIDAD TECNOLÓGICA NACIONAL - FRBA

ROBOTICA

TRABAJO PRACTICO DE LABORATORIO Nº 2

Análisis Dinámico de un Robot y su Implementación en FPGA

ALUMNOS: Perilli, Hernán
Granzella, Damián

CODIGO: 95-0482

CURSO: R6055

DOCENTE: Ing. Hernán Giannetta

AÑO: 2009

Análisis Dinámico de un Robot

Introducción Teórica al Manipulador tipo “Five Bar Linkage”

El esquema básico del manipulador en cuestión es el que se muestra a continuación

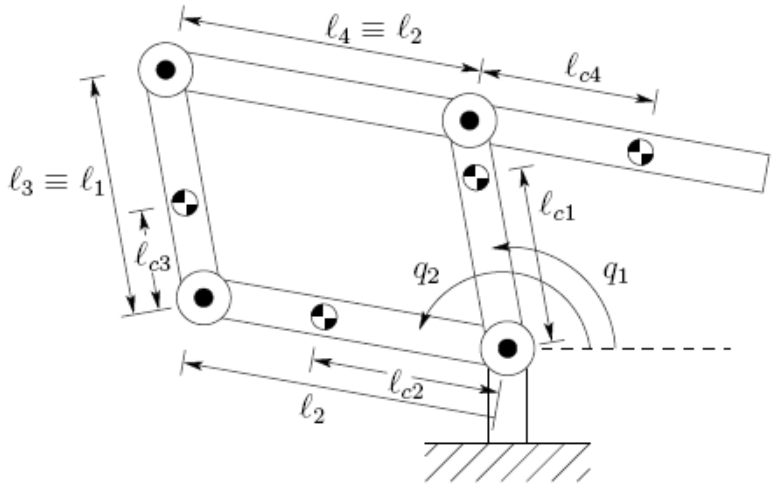


Fig. 1

Este esquema permite un simple y rápido manejo y control del manipulador, siempre y cuando se respete una simple relación entra las partes constructivas, permitiendo que las ecuaciones del mismo queden desacopladas, logrando finalmente que los movimientos rotacionales de q1 y q2 puedan ser controlados de manera independiente uno del otro.

Como se aprecia en la figura, el manipulador está formado por 4 brazos, pero en la teoría del mecanismo se contempla que existe uno más adicional constituyendo la parte de la extremidad, dándole así el nombre de la configuración por la que se lo conoce Five Bar Linkage. La configuración obtenida es la de un paralelogramo, y es que por eso también se lo conoce y se lo suele llamar “Parallelogram Arm”.

Las relaciones que se deben cumplir, y como se puede observar en la figura, son que li debe ser igual a l3, y l2 igual a l4, permitiendo así lograr el paralelogramo que es lo que finalmente lo hace al análisis y manejo más simple. A su vez se ve también que los centros de masas de los brazos no tienen que ser necesariamente iguales, pero sí mantener las relaciones de longitudes anteriormente destacadas.

Finalmente tendremos 4 links que obtendrán movimiento, a través de la actuación de los 2 grados de libertad del mismo, señalados como q1 y q2.

Este manipulador particular, a diferencia de los usualmente estudiados, forma una cadena cinemática cerrada, por lo cual no se podrán utilizar los resultados obtenidos con anterioridad como la Matriz Jacobiana.

Análisis Dinámico

En primer lugar debemos escribir las coordenadas de los centros de masa de los distintos links como una función de coordenadas generalizadas.

$$
\begin{aligned}
\begin{bmatrix} x_{c1} \\ y_{c1} \end{bmatrix} &= \begin{bmatrix} \ell_{c1}\cos q_1 \\ \ell_{c1}\sin q_1 \end{bmatrix} \\
\begin{bmatrix} x_{c2} \\ y_{c2} \end{bmatrix} &= \begin{bmatrix} \ell_{c2}\cos q_2 \\ \ell_{c2}\sin q_2 \end{bmatrix} \\
\begin{bmatrix} x_{c3} \\ y_{c3} \end{bmatrix} &= \begin{bmatrix} \ell_{c2}\cos q_1 \\ \ell_{c2}\sin q_2 \end{bmatrix} + \begin{bmatrix} \ell_{c3}\cos q_1 \\ \ell_{c3}\sin q_1 \end{bmatrix} \\
\begin{bmatrix} x_{c4} \\ y_{c4} \end{bmatrix} &= \begin{bmatrix} \ell_1\cos q_1 \\ \ell_1\sin q_1 \end{bmatrix} + \begin{bmatrix} \ell_{c4}\cos(q_2-\pi) \\ \ell_{c4}\sin(q_2-\pi) \end{bmatrix} \\
&= \begin{bmatrix} \ell_1\cos q_1 \\ \ell_1\sin q_1 \end{bmatrix} - \begin{bmatrix} \ell_{c4}\cos q_2 \\ \ell_{c4}\sin q_2 \end{bmatrix}
\end{aligned}
$$

Ec. 2

Seguidamente, podremos anotar las velocidades de los centros de masa como funciones de q1' y q2' . Por conveniencia tomaremos como cero la tercera fila de cada una de las siguientes matrices Jacobiana.

$$
\begin{aligned}
v_{c1} &= \begin{bmatrix} -\ell_{c1}\sin q_1 & 0 \\ \ell_{c1}\cos q_1 & 0 \end{bmatrix} \dot{q} \\
v_{c2} &= \begin{bmatrix} 0 & -\ell_{c2}\sin q_2 \\ 0 & \ell_{c2}\cos q_2 \end{bmatrix} \dot{q} \\
v_{c3} &= \begin{bmatrix} -\ell_{c3}\sin q_1 & -\ell_2\sin q_2 \\ \ell_{c3}\cos q_1 & \ell_2\cos q_2 \end{bmatrix} \dot{q} \\
v_{c4} &= \begin{bmatrix} -\ell_1\sin q_1 & \ell_{c4}\sin q_2 \\ \ell_1\cos q_1 & \ell_{c4}\cos q_2 \end{bmatrix} \dot{q}
\end{aligned}
$$

Ec. 2

Vamos a definir la velocidad por el Jacobiano Jvci, en la forma evidente, es decir, como las cuatro matrices que aparecen en las ecuaciones anteriores. Se deduce entonces que las velocidades angulares de los 4 eslabones esta dada por la siguiente expresión.

$$\omega_1 = \omega_3 = q_1 k, \omega_2 = \omega_4 = \dot{q}_2 k. \qquad \text{Ec. 3}$$

En base a esto, la matriz de inercia queda definida de la siguiente manera.

$$D(q) = \sum_{i=1}^{4} m_i J_{vc}^T J_{vc} + \begin{bmatrix} I_1 + I_3 & 0 \\ 0 & I_2 + I_4 \end{bmatrix} \qquad \text{Ec. 4}$$

Sustituyendo la Ec. 2 en la Ec. 4, y utilizando identidades trigonométricas estándar, se obtiene

$$
\begin{aligned}
d_{11}(q) &= m_1\ell_{c1}^2 + m_3\ell_{c3}^2 + m_4\ell_1^2 + I_1 + I_3 \\
d_{12}(q) &= d_{21}(q) = (m_3\ell_2\ell_{c3} - m_4\ell_1\ell_{c4})\cos(q_2 - q_1) \\
d_{22}(q) &= m_2\ell_{c2}^2 + m_3\ell_2^2 + m_4\ell_{c4}^2 + I_2 + I_4
\end{aligned}
$$

Ec. 5

Si en la Ec. 5 se logra que

$$m_3 \ell_2 \ell_{c3} = m_4 \ell_1 \ell_{c4} \qquad \text{Ec. 6}$$

Entonces d12 y d21 resultarán cero, por lo cual la matriz de inercia es diagonal y constante.

Como consecuencia de las ecuaciones dinámicas, no contendrá ni la fuerza de Coriolis ni las fuerzas centrífugas.

Ahora la energía potencial estará dada por

$$\begin{aligned} P &= g \sum_{i=1}^{4} y_{ci} \\ &= g \sin q_1 (m_1 \ell_{c1} + m_3 \ell_{c3} + m_4 \ell_1) \\ &+ g \sin q_2 (m_2 \ell_{c2} + m_3 \ell_2 - m_4 \ell_{c4}) \end{aligned} \qquad \text{Ec. 7}$$

Por lo cual resulta

$$\begin{aligned} \phi_1 &= g \cos q_1 (m_1 \ell_{c1} + m_3 \ell_{c3} + m_4 \ell_1) \\ \phi_2 &= g \cos q_2 (m_2 \ell_{c2} + m_3 \ell_2 - m_4 \ell_{c4}) \end{aligned} \qquad \text{Ec. 8}$$

Sabiendo que Φ1 depende solamente de q1 y no de q2, y que Φ2 depende solamente de q2 y no de q1, y si la Ec. 6 es satisfecha, entonces se puede escribir finalmente las ecuaciones como

$$\boxed{d_{11}\ddot{q}_1 + \phi_1(q_1) = \tau_1, \qquad d_{22}\ddot{q}_2 + \phi_2(q_2) = \tau_2} \qquad \text{Ec. 9}$$

El torque τ1 depende de su posición angular y de su aceleración. Lo mismo sucede para τ2

Como se ve, y si la Ec 6 se cumple, entonces se podrán ajustar los ángulos q1 y q2 de manera independiente, sin preocuparse por las interacciones entre éstas. Es esta una de las razones por lo cual la configuración del paralelogramo se ha vuelto tan popular en los robots del tipo industriales.

Ejemplo de Cálculo de Dimensionamiento de Motores para Arquitectura Tipo Five Bar Linkage

Siguiendo la estructura y nomenclaturas del manipulador en análisis, tomaremos como ejemplo los siguientes datos:

$l_1 = l_3 = 0,5m$
$l_2 = l_4 = 0,8m$
$l_{c1} = 0,4m$
$l_{c2} = 0,6m$
$l_{c3} = 0,1m$
$l_{c4} = 0,2m$
$m_1 = m_3 = 1kg$
$m_2 = 1,6kg$
$m_4 = 2,4kg$

Calcularemos en primera instancia los valores correspondientes a la matriz de inercia.

$I_1 = m_1.l_{c1}^2 = 0,16kg.m^2$
$I_2 = m_2.l_{c2}^2 = 0,576kg.m^2$
$I_3 = m_3.l_{c3}^2 = 0,01kg.m^2$
$I_4 = m_4.l_{c4}^2 = 0,096kg.m^2$

Considerando las ecuaciones anteriormente deducidas y detalladas, hallaremos el cálculo de ϕ_1, ϕ_2, d_{11} y d_{22}.

Recordando:

$$\phi_1 = g \cos q_1 \left(m_1 l_{c1} + m_3 l_{c3} + m_4 l_1 \right)$$

Ahora reemplazaremos los valores en la ecuación:

$$\phi_1 = 9,8m/s^2 \cos q_1 \left(0,4kgm + 0,1kgm + 1,2kgm \right)$$
$$\phi_1 = 16,66 \cos q_1 \left[Nm \right]$$

y

$$\phi_2 = g \cos q_2 \left(m_2 l_{c2} + m_3 l_2 - m_4 l_{c4} \right)$$

Finalmente

$$\phi_2 = 9,8m/s^2 \cos q_2 \left(0,96kgm + 0,8kgm - 0,48kgm \right)$$

$$\phi_2 = 12,54 \cos q_1 \left[Nm \right]$$

Ahora haremos del mismo modo para d_{11} y d_{22}.

$$d_{11} = m_1 l_{c1}^2 + m_3 l_{c3}^2 + m_4 l_1^2 + I_1 + I_3$$
$$d_{11} = \left[0,16 + , 0,01 + 0,6 + 0,16 + 0,0,01 \right] kgm^2$$
$$d_{11} = 0,94 kgm^2$$

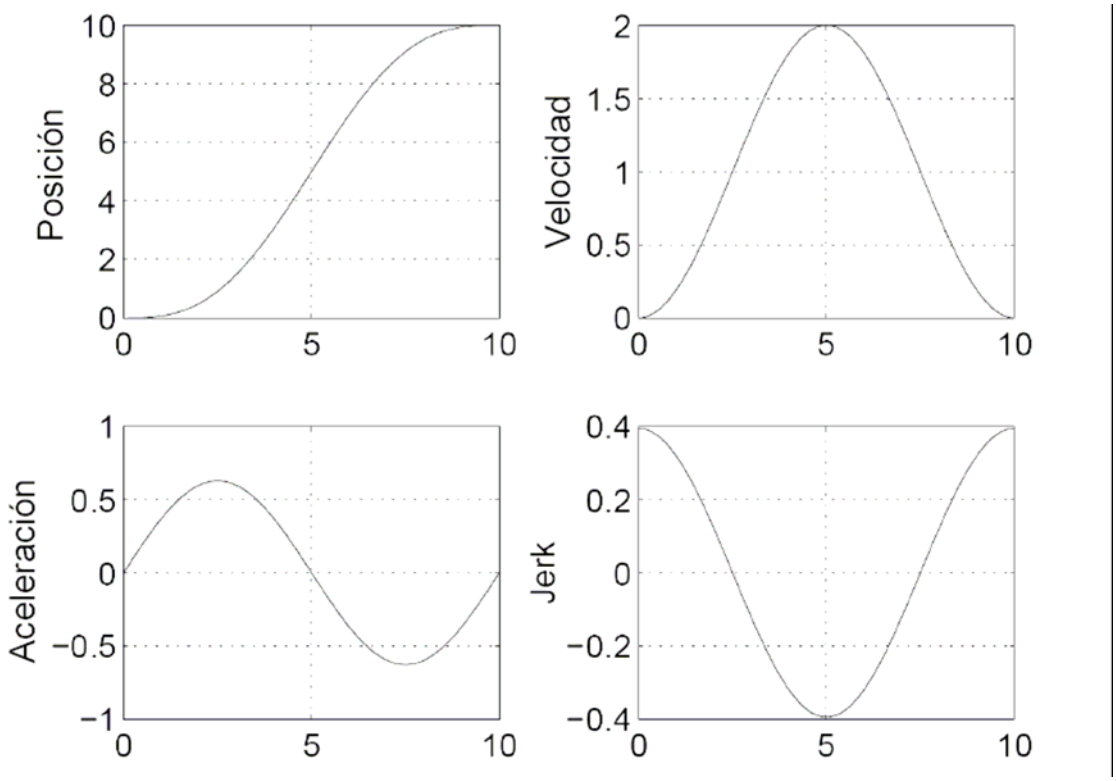
$$d_{22} = m_2 l_{c2}^2 + m_3 l_2^2 + m_4 l_{c4}^2 + I_2 + I_4$$
$$d_{22} = \left[0,576 + 0,8 + 0,48 + 0,576 + 0,096 \right] kgm^2$$
$$d_{22} = 2,528 kgm^2$$

De esta manera, las ecuaciones de torques quedan dadas de la siguiente manera

$$T_1 = d_{11} q'' + \phi_1 \qquad T_2 = d_{22} q'' + \phi_2$$

$$\boxed{T_1 = \left[0,94 q_1'' + 16,66 \cos q_1 \right] Nm} \qquad \boxed{T_2 = \left[2,528 q_2'' + 12,544 \cos q_2 \right] Nm}$$

Para este ejemplo consideraremos que las trayectorias que tomarán ambas articulaciones serán del tipo *cicloidales,* y por lo tanto las funciones de posición, velocidad, aceleración, y jerk tendrán las siguientes formas.



Ya consideradas las funcione de posición y aceleración, se deberá considerar la condición más desfavorable, por la cual se requerirá del torque máximo.

Para el caso de q_1, consideraremos que la posición final la alcanzará una vez logrado los 180º, es decir, π rad. Podemos observar en las funciones correspondientes, que el pico de aceleración máximo lo tendrá alcanzada la cuarta parte de la trayectoria total, es decir, a los 45º, o bien a los $\pi/4$ rad. Ya con esto, y volcando dichos valores, podemos calcular el toque máximo para q_1.

$$T_1 = \left[0,94 q_1 \text{"} + 16,66 \cos q_1\right] Nm$$

$$T_1 = \left[0,94.0,65 + 16,66 \cos\left(\frac{\pi}{4}\right)\right] Nm$$

$$T_1 = \left[0,611 + 11,78\right] Nm$$

$$\boxed{T_1 = 12,39 Nm}$$

Ahora si lo analizamos para el caso en que $q_1 = 0$, vemos que el primer término se anula, ya que la aceleración es cero, por lo cual sólo contamos con el segundo término, y sabiendo que $\cos 0 = 1$, podemos fácilmente saber que

$$\boxed{T_1 = 16,66 Nm}$$ (Torque a velocidad cero)

El mismo análisis se hará para q_2, pero considerando que el final de la trayectoria la realizará alcanzados los 135º, es decir $3/4\,\pi$ rad, para el mismo tipo de trayectoria.

$$T_2 = \left[2,528 q_2 \text{"} + 12,544 \cos q_2\right] Nm$$

$$T_2 = \left[2,528.0,65 + 12,544 \cos\left(\frac{3}{8}\pi\right)\right] Nm$$

$$T_2 = \left[1,643 + 12,541\right] Nm$$

$$\boxed{T_2 = 14,18 Nm}$$

Y para el caso es que $q_2 = 0$

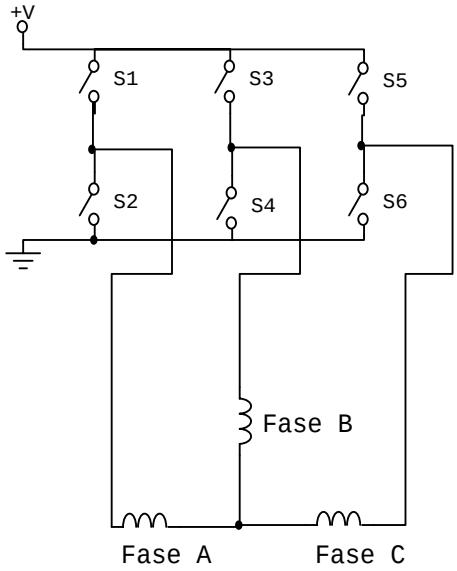
$$\boxed{T_2 = 12,54 Nm}$$ (Torque a velocidad cero)

En base a los resultados obtenidos podemos buscar en la tabla siguiente, correspondiente a los servomotores de la marca ABB, y seleccionar los mismos.

Type	Continuous torque at zero speed [5] T_{co} [Nm]	Current at continuous torque [1) 3) 5)] I_{co} [A]	Rated torque [5] T_{rat} [Nm]	Rated current [1) 3) 5)] I_{rat} [A]	Rated speed n_{rat} [r/min]	Rated frequency f_{rat} [Hz]	Mechanical rated power [5] P_{rat} [kW]	Peak torque T_{pk} [Nm]	Current at peak torque [1) 3)] I_{pk} [A]	Torque constant [1) 2) 3)] k_T [Nm/A]	B.e.m.f. between phases at rated speed [1) 2) 3)] V [V]	Moment of inertia of rotor [3] J_M [kgcm²]	Moment of inertia of rotor + brake [3] J_M with brake [kgcm²]	Weight [3) 4)] W [kg]
9C1.1.30.. ..M	1.4	1.3	1.3	1.4	3000	250	0.41	4.1	4.5	1.147	208	0.57	0.62	3.0
9C1.2.30.. ..M	2.3	1.8	2	1.7	3000	250	0.63	6.9	6.1	1.440	261	1.04	1.09	3.9
9C1.3.30.. ..M	3.2	2.7	2.8	2.5	3000	250	0.88	9.6	9.0	1.350	245	1.51	1.56	4.8
9C1.4.30.. ..M	4.2	3.3	3.5	2.9	3000	250	1.10	12.6	11.1	1.440	261	1.99	2.04	5.7
9C1.1.60.. ..M	1.4	2.1	1.2	2.0	6000	500	0.75	4.1	7.1	0.720	261	0.57	0.62	3.0
9C1.2.60.. ..M	2.3	3.6	1.6	2.7	6000	500	1.01	6.9	12.1	0.720	261	1.04	1.09	3.9
9C1.3.60.. ..M	3.2	5.2	2.3	3.9	6000	500	1.45	9.6	17.3	0.702	255	1.51	1.56	4.8
9C1.4.60.. ..M	4.2	6.5	2.5	4.1	6000	500	1.57	12.6	21.6	0.738	268	1.99	2.04	5.7
9C4.1.30.. ..M	4.3	3.0	3.9	2.8	3000	250	1.23	12.9	9.8	1.654	300	4.0	4.7	4.1
9C4.2.30.. ..M	7.5	5.0	6.1	4.3	3000	250	1.92	22.5	16.7	1.704	309	7.6	8.3	7.0
9C4.3.30.. ..M	9.4	6.0	6.9	4.6	3000	250	2.17	28.2	19.9	1.786	324	11.1	11.8	9.9
9C4.4.30.. ..M	12.0	8.2	7.5	5.4	3000	250	2.36	36.0	27.3	1.665	302	14.7	15.4	12.8
9C4.1.40.. ..M	4.3	4.0	3.7	3.6	4000	333	1.55	12.9	13.2	1.232	298	4.0	4.7	4.1
9C4.2.40.. ..M	7.5	6.9	5.4	5.2	4000	333	2.26	22.5	23.1	1.232	298	7.6	8.3	7.0
9C4.3.40.. ..M	9.4	7.8	5.8	5.1	4000	333	2.43	28.2	26.1	1.365	330	11.1	11.8	9.9
9C4.4.40.. ..M	12.0	10.0	6.3	5.5	4000	333	2.64	36.0	33.3	1.365	330	14.7	15.4	12.8
9C5.2.20.. ..M	12.3	5.9	10.3	5.2	2000	166.7	2.16	36.9	19.7	2.365	286.0	21.8	23.6	16
9C5.3.20.. ..M	18.4	9.0	14.8	7.6	2000	166.7	3.10	55.2	29.9	2.328	281.5	31.6	33.4	20
9C5.4.20.. ..M	23.5	11.6	17.1	8.9	2000	166.7	3.58	70.5	38.6	2.306	278.9	41.4	43.2	24
9C5.6.20.. ..M	30.0	12.8	22.0	9.9	2000	166.7	4.61	90.0	42.7	2.661	321.8	61.0	62.8	32
9C5.2.30.. ..M	12.3	9.0	9.0	6.9	3000	250	2.83	36.9	30.0	1.552	281.5	21.8	23.6	16
9C5.3.30.. ..M	18.4	12.1	12.4	8.6	3000	250	3.90	55.2	40.3	1.730	313.7	31.6	33.4	20
9C5.4.30.. ..M	23.5	15.1	14.0	9.4	3000	250	4.40	70.5	50.2	1.774	321.8	41.4	43.2	24
9C5.6.30.. ..M	30.0	19.2	18.0	12.1	3000	250	5.65	90.0	64.1	1.774	321.8	61.0	62.8	32

En ambos casos vemos que el servomotor **9C5.3.20** cumple con los torque máximos e iniciales para q_1 y q_2, por lo tanto estos son los que utilizaríamos en nuestro desarrollo garantizando un correcto dimensionamiento de los servomotores para nuestro manipulador.

Control PWM de motor BLDC mediante el uso de triple semi puente H



Para generar un giro completo (360º) en el eje del motor BLDC, será necesaria una secuencia de seis (6) pasos correspondiente a todas las combinaciones posibles donde dos fases simultáneamente en serie quedan conectadas al bus de corriente continua (DC) a través del accionamiento de las llaves electrónicas (S1,....,S6) según corresponda a cada caso.

La velocidad de giro se corresponderá con el período ó tiempo en que se realice cada secuencia completa. Adicionalmente, para poder controlar el torque en el eje del motor, en lugar de que en cada uno de los seis pasos de la secuencia queden las fases en conexión directa al bus de DC, deberán ser alimentadas mediante una señal pulsante modulada por ancho de pulso: "PWM".

A continuación se brindará una tabla donde se describen en cada uno de los 6 pasos de la secuencia total de giro completo, cuales son las fases y el modo en que son conectadas al bus de contínua y el correspondiente control de las llaves "S1,...,S6".

Paso de la secuencia	Fase A	Fase B	Fase C	S1	S2	S3	S4	S5	S6
1	+Vdc	GND pwm	-	**ON**	off	off	**PWM**	off	off
2	+Vdc	-	GND pwm	**ON**	off	off	off	off	**PWM**
3	-	+Vdc	GND pwm	off	off	**ON**	off	off	**PWM**
4	GND pwm	+Vdc	+Vdc	off	**PWM**	**ON**	off	off	off
5	GND pwm	-	+Vdc	off	**PWM**	off	off	**ON**	off
6	-	GND pwm	-	off	off	off	**PWM**	**ON**	off

Implementación de la lógica que controla la secuencia

En este ejemplo de control de motor BLDC, se ejecutará un cambio en la secuencia de rotación determinada anteriormente luego de la inyección de diez (10) ciclos de señal PWM en la fase correspondiente. Dicho en otras palabras: Luego de 10 períodos de la señal PWM se conmutará al siguiente estado de la secuencia de giro del motor.

La lógica de control consiste en tres procesos concurrentes adicionalmente a los ya tenidos que generan la señal simple PWM (declarada en el cuerpo principal del programa VHDL como: **Signal pwm_int: std_logic;)**.

Proceso 1:

Genera una señal "**clock_div10**" que consiste en un pulso por cada 10 ciclos de señal PWM "**pwm_int**". Esto se consigue dividiendo por 20 la señal "**rco_int**".

```
Gen_clk10: PROCESS (reset, rco_int) IS
VARIABLE cuenta: INTEGER range 0 to 19;

BEGIN
        IF (reset = '1') THEN
                cuenta:= 0;
                clock_div10 <= '1';
        ELSE
                IF (rising_edge (rco_int)) THEN
                        IF (cuenta = 20) THEN
                                cuenta:= 0;
                                clock_div10 <= '1';
                        ELSE
                                cuenta:= cuenta + 1;
                                clock_div10 <= '0';
                        END IF;
                END IF;
        END IF;

END PROCESS;
```

Proceso 2:

Genera la señal "**sel**" que consiste en una señal que adoptará valores entre 0 y 5 indicando cada uno de los 6 estados correspondientes a la secuencia de giro del motor BLDC. El cambio de valor de la señal "**sel**" se realizará al ritmo de la señal "**clock_div10**", o sea por cada 10 ciclos de la señal PWM "**pwm_int**".

```
GenVarState: PROCESS (reset, clock_div10) IS
SIGNAL cuenta: INTEGER range 0 to 5;

BEGIN
        IF (reset = '1') THEN
                cuenta:= 0;
        ELSE
                IF (rising_edge (clock_div10)) THEN
                        IF (cuenta = 5) THEN
                                cuenta:= 0;
                        ELSE
                                cuenta:= cuenta + 1;
                        END IF;
                END IF;
        END IF;

sel <= cuenta;

END PROCESS;
```

Proceso 3:

Consiste en la implementación de una FSM de seis estados indicados por la señal "**sel**". Cada uno de estos estados se corresponderá con el número de secuencia de giro del motor BLDC y en cada uno de los cuales se establecerá la conexión de la señal de control (ON, OFF, PWM) de cada uno de los seis conmutadores electrónicos (S1,...,S6).

```
Cntrl_PWM_puenteH: PROCESS (reset, sel, pwm_int) IS

BEGIN
        IF (reset = '1') THEN
                S1<= 0; S2<= 0; S3<= 0; S4<= 0; S5<= 0; S6<= 0;
        ELSE
                CASE sel IS
                        WHEN 0 =>                       --Estado 1
                                S1<= 1; S2<= 0; S3<= 0; S4<= pm_int; S5<= 0; S6<= 0;

                        WHEN 1 =>                       --Estado 2
                                S1<= 1; S2<= 0; S3<= 0; S4<= 0; S5<= 0; S6<= pwm_int;

                        WHEN 2 =>                       --Estado 3
                                S1<= 0; S2<= 0; S3<= 1; S4<= 0; S5<= 0; S6<= pwm_int;

                        WHEN 3 =>                       --Estado 4
                                S1<= 0; S2<= pwm_int; S3<= 1; S4<= 0; S5<= 0; S6<= 0;

                        WHEN 4 =>                       --Estado 5
                                S1<= 0; S2<= pwm_int; S3<= 0; S4<= 0; S5<= 1; S6<= 0;
```

```
                    WHEN 5 =>                       --Estado 6
                        S1<= 0; S2<= 0; S3<= 0; S4<= pwm_int; S5<= 1; S6<= 0;

                    WHEN OTHERS =>
                        S1<= 0; S2<= 0; S3<= 0; S4<= 0; S5<= 0; S6<= 0;
                END CASE;
        END IF;

sel <= cuenta;

END PROCESS;
```

Archivo "TPN2.VHDL" donde se describe el bloque lógico del generador de señal PWM (comportamental) y se generan las señales para la depuración de la lógica (simulación).

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

-- En la entidad se declaran los puertos de entrada/salida del dispositivo.
-- En este caso solo se declaran las señales de salida que controlarán las compuertas de
-- los conmutadores electrónicos de potencia pertenecientes a los semipuentes H.
ENTITY TPN2 IS
PORT (  S1:     out STD_LOGIC;
        S2:     out STD_LOGIC;
        S3:     out STD_LOGIC;
        S4:     out STD_LOGIC;
        S5:     out STD_LOGIC;
        S6:     out STD_LOGIC
      );
END TPN2;

-- Arquitectura principal ó cuerpo del programa principal.
ARCHITECTURE arch_pwm OF TPN2 IS
-- ---------------------------------------------------------------------------
-- Declaración de señales y variables internas al bloque.
-- ---------------------------------------------------------------------------
SIGNAL sig_clock                : std_logic;
SIGNAL sig_reset                : std_logic;
SIGNAL sig_data_value           : std_logic_vector(7 downto 0);
--SIGNAL sig_pwm                : std_logic;
shared variable ENDSIM          : boolean:=false;
constant clk_period             :TIME:=100ns;

SIGNAL reg_out                          : std_logic_vector(7 downto 0);
SIGNAL cnt_out_int                      : std_logic_vector(7 downto 0);
SIGNAL pwm_int, rco_int, clock_div10    : STD_LOGIC;
SIGNAL sel                              : integer range 0 to 5;
SIGNAL cont_sel                         : std_logic_vector(3 downto 0);

-- Inicio del bloque lógico y generación de señales de simulación.
BEGIN

-- Proceso donde se genera la señal de reloj principal del sistema de período
-- 100nS (clk_period) para la simulación de la lógica diseñada.
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;

-- Proceso correspondiente a la etapa de simulación. Se establece una señal de reset
-- durante los primeros 100ns del ciclo de simulación y luego cada 1mS se modifica el
```

```
-- valor del registro de 8 bits que contiene el valor del ancho del ciclo de actividad de la
-- señal PWM.
stimulus_process: PROCESS

BEGIN
            sig_reset <= '1';
            wait for 1000 ns;
            sig_reset <= '0';
            sig_data_value <= "11000000";
            wait for 1000 us;
            sig_data_value <= "10000000";
            wait for 1000 us;
            sig_data_value <= "01000000";
            wait for 1000 us;

    wait;
 END PROCESS stimulus_process;

-- En cada flanco de subida de la señal de reloj principal se carga en un registro auxiliar
-- de 8 bits con el valor del ciclo de actividad de la señal PWM. Se carga con un valor -
-- inicial al detectar señal de reset.
PROCESS(sig_clock,reg_out,sig_reset)

        BEGIN
            IF (sig_reset ='1') THEN
                reg_out <="00000000";
            ELSIF (rising_edge(sig_clock)) THEN
                 reg_out <= sig_data_value;
            END IF;
END PROCESS;

 -- Bloque lógico generador de la señal PWM. Se carga un contador UP/DOWM de 8 bits
-- con el valor del registro auxiliar mencionado anteriormente. Durante el ciclo de
-- actividad de la señal PWM el contador incrementa su valor con cada pulso de reloj y al
-- llegar a su valor máximo se recarga nuevamente con el valor de recarga fijo. Durante el
-- ciclo inactivo de la señal PWM el contador se decrementa con cada pulso de reloj y al
-- llegar a cero se produce nuevamente la recarga con el valor fijo y así sucesivamente.
-- Este proceso descripto corresponde a un período completo de la señal PWM.
 PROCESS (sig_clock, cnt_out_int, rco_int, reg_out)

   BEGIN
        IF (rco_int = '1') THEN
           cnt_out_int <= reg_out;
        ELSIF  rising_edge(sig_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;
```

```
-- Proceso que describe la lógica de generación de la señal RCO que consiste en un
-- pulso cada vez que el contador de ciclo PWM llega a cualquiera de sus valores
-- extremos, indicando de esta manera el momento en que la señal PWM debe conmutar
-- respecto a su valor lógico actual.
 PROCESS(cnt_out_int, rco_int, sig_clock, sig_reset)

   BEGIN
        IF (sig_reset ='1') THEN
             rco_int <='1';
        ELSIF rising_edge(sig_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;


-- Proceso que describe el biestable generador de la señal PWM al ritmo de los pulsos de
-- la señal RCO.
PROCESS (sig_clock, rco_int, sig_reset)
   BEGIN
        IF (sig_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;


-- Proceso que describe una lógica que genera una señal “clock_div10” pulsante cada 20
-- pulsos de la señal RCO.
Gen_clk10: PROCESS (sig_reset, rco_int) IS
VARIABLE cuenta: INTEGER range 0 to 20;
  BEGIN
        IF (sig_reset = '1') THEN
                cuenta:= 0;
                clock_div10 <= '1';
        ELSE
                IF rising_edge(rco_int) THEN
                        IF (cuenta = 20) THEN
                                cuenta:= 0;
                                clock_div10 <= '1';
                        ELSE
                                cuenta:= cuenta + 1;
                                clock_div10 <= '0';
                        END IF;
                END IF;
        END IF;

END PROCESS;
```

-- **Proceso donde se incrementa en valor de la variable de estado “sel” por cada pulso de la señal “clock_div10”. El valor de la varialble “sel” varia cíclicamente entre 0 y 5.**

```
GenVarState: PROCESS (sig_reset, clock_div10) IS
VARIABLE cuenta: INTEGER range 0 to 5;

 BEGIN
        IF (sig_reset = '1') THEN
                cuenta:= 0;
        ELSE
                IF rising_edge (clock_div10) THEN
                        IF (cuenta = 5) THEN
                                cuenta:= 0;
                        ELSE
                                cuenta:= cuenta + 1;
                        END IF;
                END IF;
        END IF;

sel <= cuenta;

END PROCESS;
```

-- **FSM donde en cada uno de los 6 estados diferentes se establece la fase activa.**

```
Cntrl_PWM_puenteH: PROCESS (sig_reset, sel, pwm_int) IS

BEGIN
        IF (sig_reset = '1') THEN
                S1<= '0'; S2<= '0'; S3<= '0'; S4<= '0'; S5<= '0'; S6<= '0';
        ELSE
                CASE sel IS
                        WHEN 0 =>                       --Estado 1
                                S1<= '1'; S2<= '0'; S3<= '0'; S4<= pwm_int; S5<= '0'; S6<= '0';

                        WHEN 1 =>                       --Estado 2
                                S1<= '1'; S2<= '0'; S3<= '0'; S4<= '0'; S5<= '0'; S6<= pwm_int;

                        WHEN 2 =>                       --Estado 3
                                S1<= '0'; S2<= '0'; S3<= '1'; S4<= '0'; S5<= '0'; S6<= pwm_int;

                        WHEN 3 =>                       --Estado 4
                                S1<= '0'; S2<= pwm_int; S3<= '1'; S4<= '0'; S5<= '0'; S6<= '0';

                        WHEN 4 =>                       --Estado 5
                                S1<= '0'; S2<= pwm_int; S3<= '0'; S4<= '0'; S5<= '1'; S6<= '0';

                        WHEN 5 =>                       --Estado 6
                                S1<= '0'; S2<= '0'; S3<= '0'; S4<= pwm_int; S5<= '1'; S6<= '0';

                        WHEN OTHERS =>
                                S1<= '0'; S2<= '0'; S3<= '0'; S4<= '0'; S5<= '0'; S6<= '0';
                END CASE;
        END IF;

END PROCESS;

END ARCHITECTURE arch_pwm;
```

Imágenes del resultado de la simulación del proyecto “TPN2.vhdl”

Figura 1:

Se observan los eventos durante los primeros 5 microsegundos. La señal de reset “sig_reset” permanece activa durante el primer microsegundo.

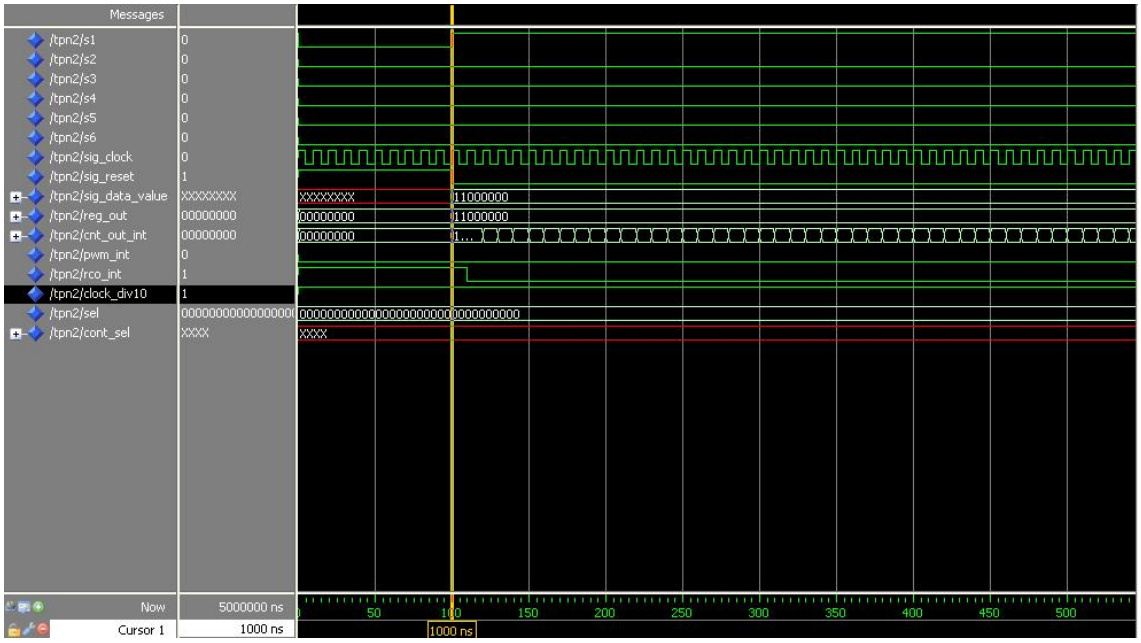


Figura 2:

Se observan los eventos sucedidos durante el primer milisegundo.

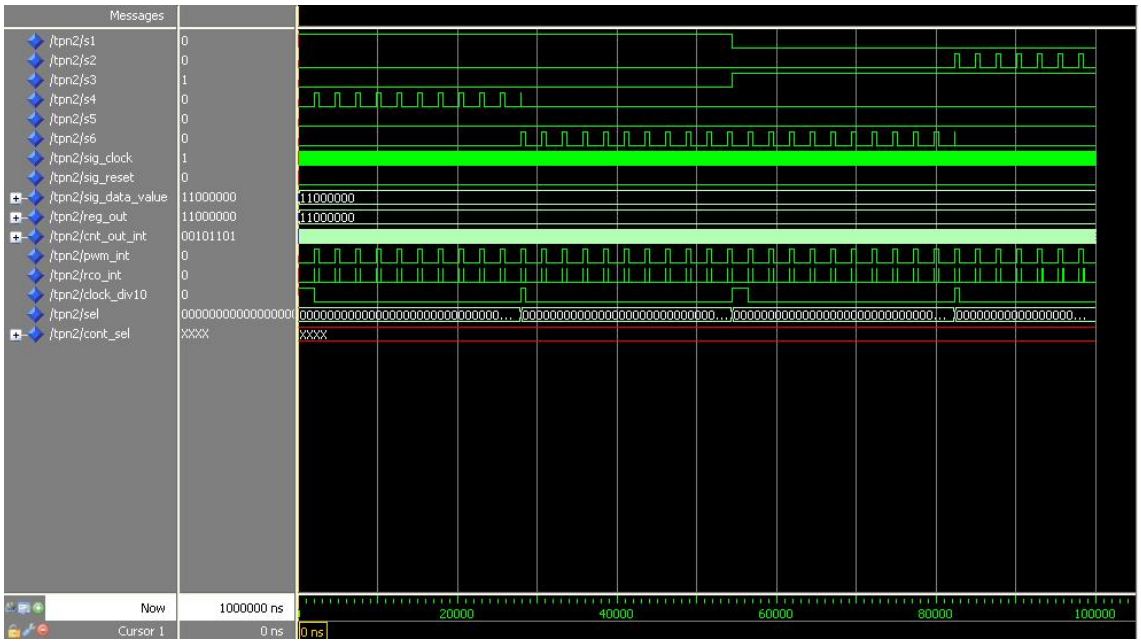


Figura 3:

Se observan los eventos sucedidos durante los primeros 5 milisegundos. Observar que se el eje del motor ha experimentado supuestamente 3 giros completos durante este ciclo de simulación. Se puede observar también, que como ha de suponerse, se han producido 18 (6 estados/giro * 3 giros) variaciones en el valor de la variable “sel”.

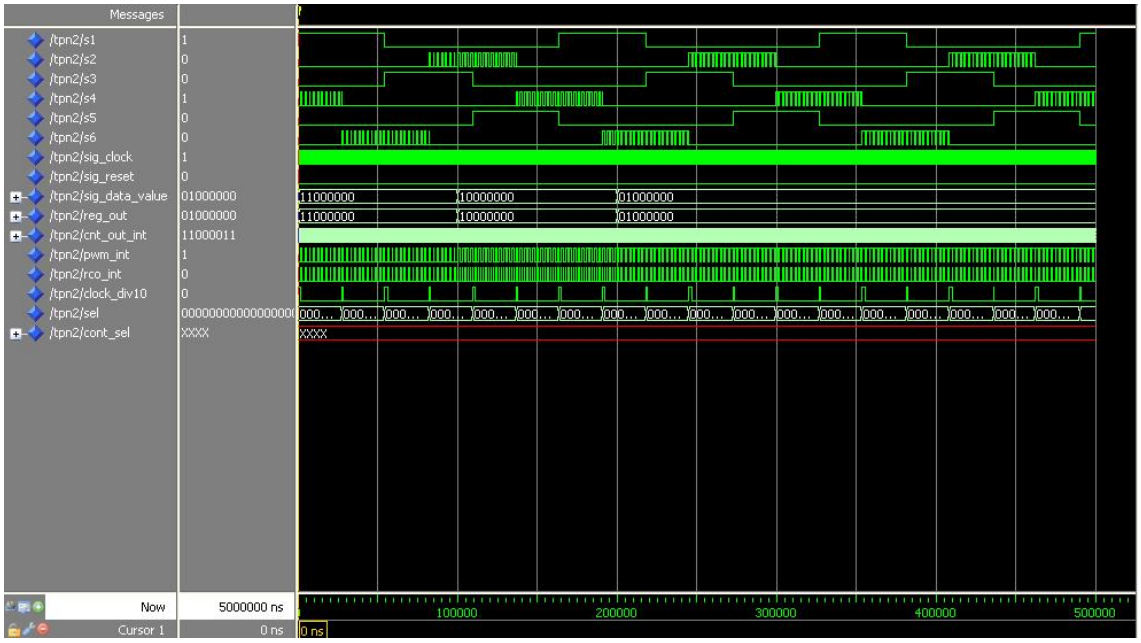


Figura 4:

Observar la variación producida en la señal “pwm_int” como efecto del valor del registro sig_data_value que varia de “11000000” a “10000000” en el instante de tiempo t = 1 mSeg.

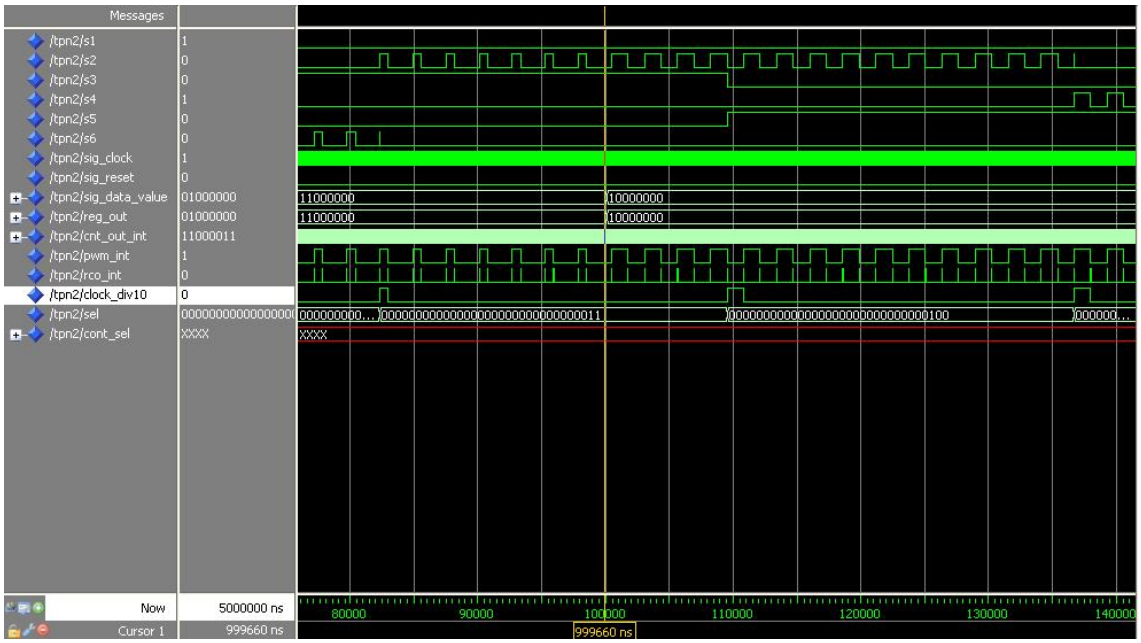


Figura 5:

Observación con mayor grado de detalle de la variación producida en la señal “pwm_int” como efecto del valor del registro sig_data_value que varia de “11000000” a “10000000” en el instante de tiempo t = 1 mSeg.

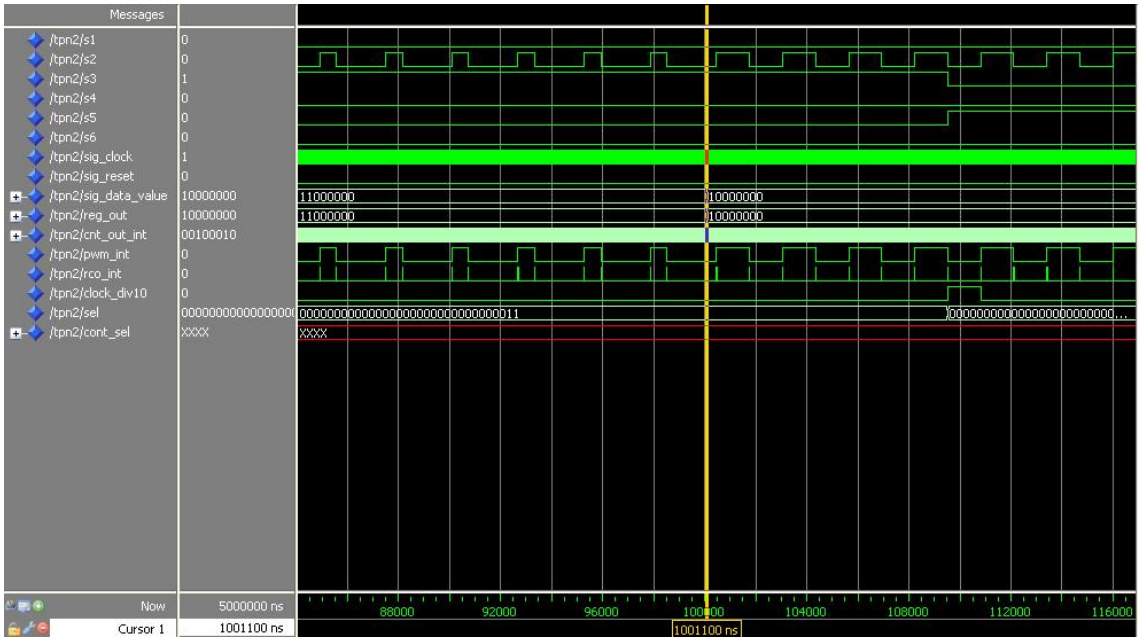


Figura 6:

Se observa en esta figura que cuando el valor del registro “cnt_out_int” alcanza en valor “00000000” se genera un pulso de la señal “rco_int” que a su vez provoca un cambio de estado de la señal “pwm_int” (‘0’ -> ‘1’). Observar que en el siguiente ciclo de reloj el registro “cnt_out_int” se carga con el valor de recarga prefijado en “reg_out”. Observar además que el valor del registro “cnt_out_int” se decrementa con cada pulso de reloj “sig_clock”.

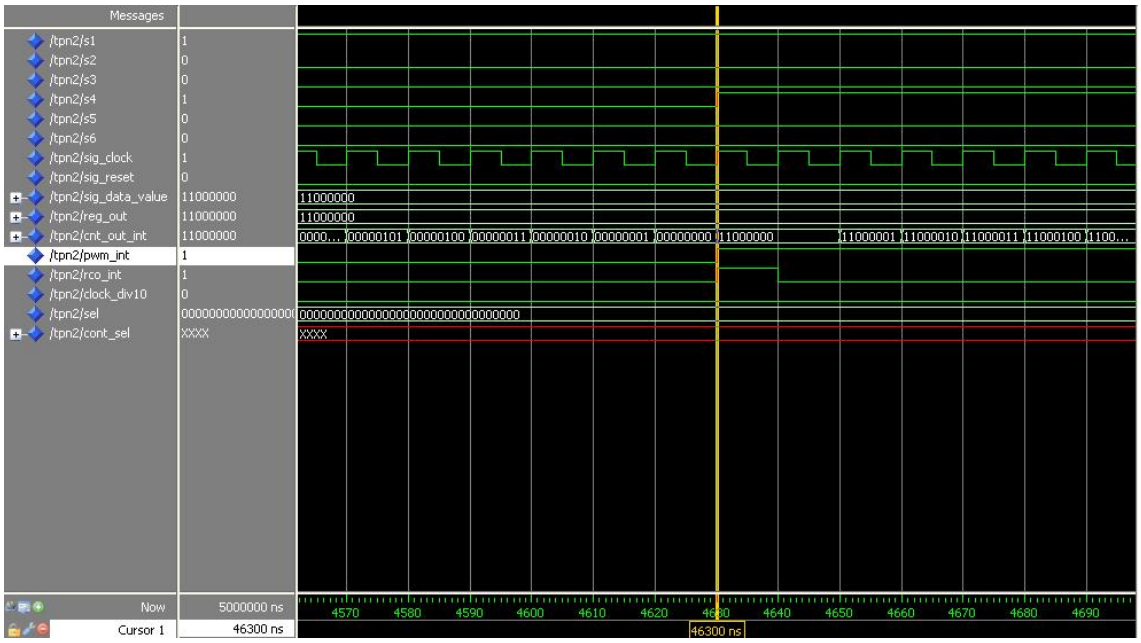
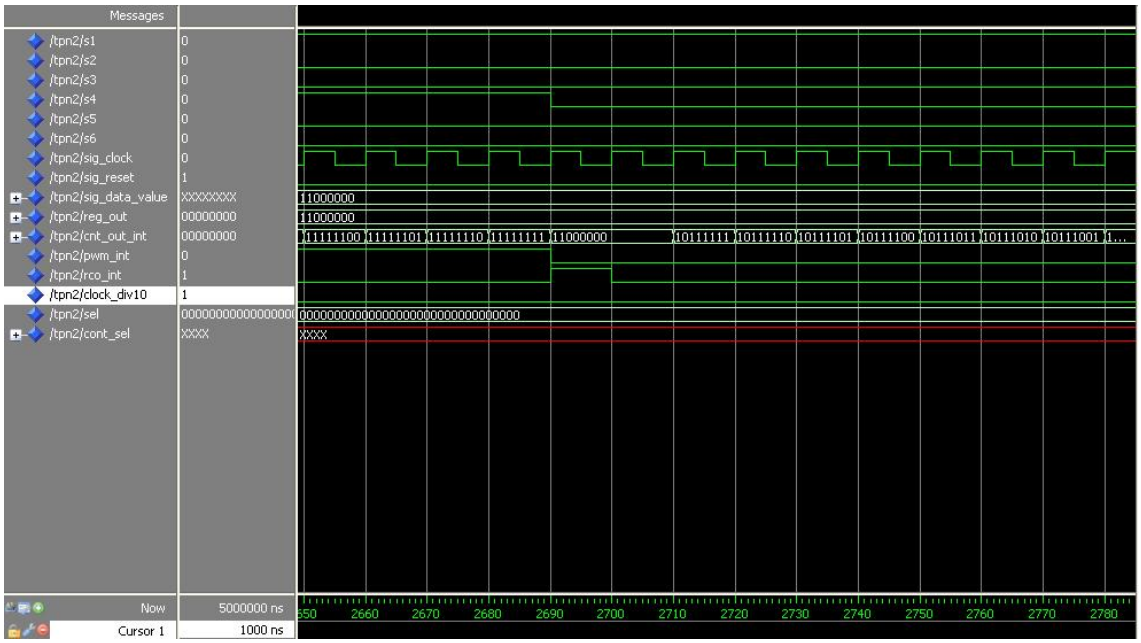


Figura 7:

Se observa en esta figura que cuando el valor del registro “cnt_out_int” alcanza en valor “11111111” se genera un pulso de la señal “rco_int” que a su vez provoca un cambio de estado de la señal “pwm_int” (‘1’ -> ‘0’). Observar que en el siguiente ciclo de reloj el registro “cnt_out_int” se carga con el valor de recarga prefijado en “reg_out”. Observar además que el valor del registro “cnt_out_int” se decrementa con cada pulso de reloj “sig_clock”.



Conclusiones Finales

En base a lo analizado durante esta práctica, podemos concluir en primera instancia, en que los análisis teóricos realizados para el cálculo y dimensionamiento de los motores para un manipulador del tipo “Five Bar Linkage” corresponden y son prácticamente realizables para el desarrollo e ingeniería de un manipulador de este tipo, dando así la tranquilidad y confiabilidad de que dichos motores contarán con el torque necesario para mover y controlar al manipulador en todo momento, es decir, en la peor condición analizada según su funcionamiento, tomando en cuenta el recorrido y tipo de trayectoria a realizar, e inclusive en el momento de posición y velocidad cero, donde deberá romper con la inercia inicial.

Por otro lado se verifica el funcionamiento del un FPGA, programado en lenguaje VHDL para el control y comando de los motores tipo BSDC, los cuales serán comandados mediante la variación de tensión sobre dichos bobinados. Dicha variación se realiza por medio de un control PWM, logrando variar el valor medio de la tensión, la cual se traduce en variación de corriente, estando ésta relacionada directamente con el torque del servomotor controlando el mismo. A través PWM y el puente tipo H se controla la tensión de cada una de las bobinas pudiendo modificar su velocidad y controlando su torque, aunque para tener un verdadero control de esto se requiere de una realimentación del estado y posición del motor, generalmente por medio de sensores del tipo de efecto Hall montados sobre el eje del rotor, aunque en esta práctica hemos simulado su variación generando una rampa y verificando satisfactoriamente su funcionamiento.