

UTN – FRBA



Robótica

TPN2

Análisis Dinámico de un Robot e implementación en FPGA

Profesor: Hernán Giannetta

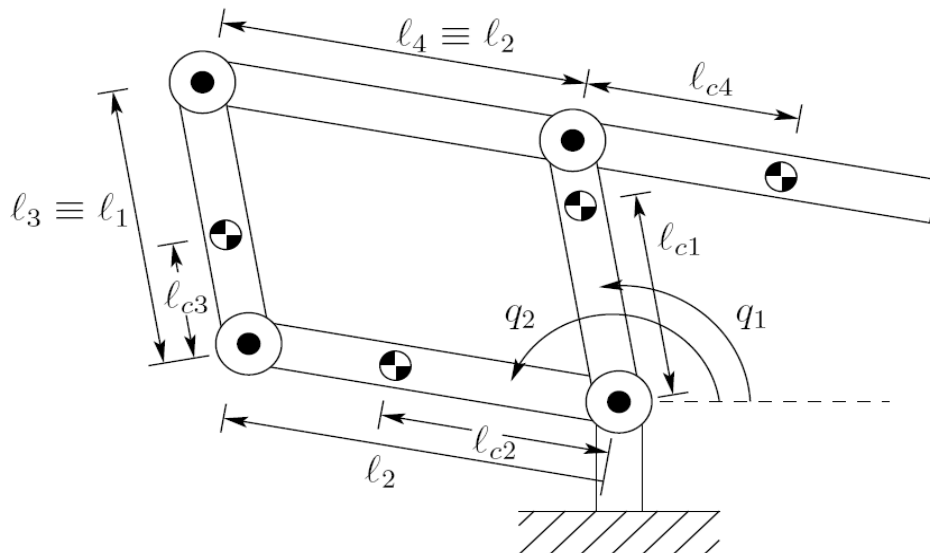
Alumnos:

Martín Cabrera	111492-0
Christian Gutierrez	96985-5

1C/2009

Analisis dinámico del Arreglo de acoplanimeto de 5 barras (five bar linkage)

Se busca demostrar que si los parametros constructivos del dispositivo cumplen con una simple relacion, entonces las ecuaciones del manipulador son independientes, por lo que los valores de q_1 y q_2 peden controlarse independientemente una de la otra.



El dispositivo consta solamente de cuatro barras, pero es una convencion en la teoria de mecanismos el contar la tierra como un acoplamiento mas, dano asi sentido al nombre del arreglo.

Se asume que en el arreglo las longitudes l_1 y l_3 son iguales, lo mismo que las longitudes l_2 y l_4 , de esta manera el camino cerrado constituye un paralelogramo, lo cual simplifica notoriamente los calculos.

Si bien las longitudes de los brazos debe ser la misma no ocurre lo mismo con los centros de masa, los cuales pueden ser distintos como en el ejemplo de la figura.

Este tipo de arreglo permite el montaje de los motores en la base, evitando que la masa del mismo pase a formar parte del dispositivo y reduciendo su capacidad de carga.



Se trata de un dispositivo de 2 grados de libertad, formando una cadena cinemática cerrada. No se pueden usar las mismas matrices jacobianas que se usan con los dispositivos estudiados de Cadena cinemática abierta.

Para el análisis Dinámico se utiliza el enfoque Energético de Lagrange-Euler.

Como primer paso se escriben las ecuaciones de los centros de masa de los diversos miembros, en coordenadas generalizadas.

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

Luego se calcularan las velocidades de los diversos centros de masa en función de \dot{q}_1 y \dot{q}_2 .

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

Se define la velocidad por el Jacobiano J_{vci} como las cuatro matrices que aparecen en las ecuaciones anteriores. A continuación, es evidente que la velocidad angular de los cuatro enlaces está dada simplemente por:

$$\omega_1 = \omega_3 = \dot{q}_1 k, \omega_2 = \omega_4 = \dot{q}_2 k$$

La matriz de inercia esta dada por:

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

Sustituyendo en esta ecuación, la obtenida en la pagina anterior, las cuales son operadas con identidades trigonométricas, obtenemos

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

Si diseñamos para la condición

$$m_3 \ell_2 \ell_{c3} = m_4 \ell_1 \ell_{c4}$$

resulta que d_{12} y d_{21} son cero, es decir, la matriz de inercia es diagonal y constante.

Como consecuencia las ecuaciones dinámicas no contendrán ni las fuerzas de Coriolis ni las fuerzas centrífugas.

Por lo tanto la energía potencial estará dada por:

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

Reescribiendo:

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

Sabiendo que Φ_1 depende solo de q_1 ; y que Φ_2 depende solo de q_2 , que es el aspecto mas complejo de este manipulador, podemos reescribir el set de ecuaciones como:

$$d_{11}\ddot{q}_1 + \phi_1(q_1) = \tau_1, \quad d_{22}\ddot{q}_2 + \phi_2(q_2) = \tau_2$$

Simulación de Comportamiento del arreglo 5 bar Linkage

Para Simular el comportamiento del dispositivo definimos los parametros de masa y longitur del arreglo, cumpliendo con las relaciones encontradas, con ellos calcularemos los parametros d_{11} y d_{22} de la matriz de inercia. Luego propondremos una trayectoria y calcularemos sus derivadas primera y segunda. Con ellas calcularemos los parametros $\dot{\theta}_1$ y $\dot{\theta}_2$ para cada instante. Una vez obtenidos podemos calcular el torque instantaneo en el brazo.

Para calcular los parámetros del motor deberemos multiplicar la velocidad angular por la relacion de transformación y dividir el torque por la misma.

Calculamos todo lo descrito usando una hoja de calculo que se transcribe a continuación y luego se muestran los gráficos obtenidos.

Definición de límites

Ángulo	min	max
q1	20	120
q2	100	200

Definición de parámetros

L1	2,00	Lc1	1,50	m1	7,00
L2	0,50	Lc2	0,10	m2	0,40
L3	2,00	Lc3	0,90	m3	3,00
L4	0,50	Lc4	0,11	m4	6,00
g	-9,80	d11	52,18	d22	7,23

Verificación de condiciones de diseño	
m3 l2 lc3=m4 l1 lc4	cumple

Factores de Inercia	
d11	d22
52,18	7,23

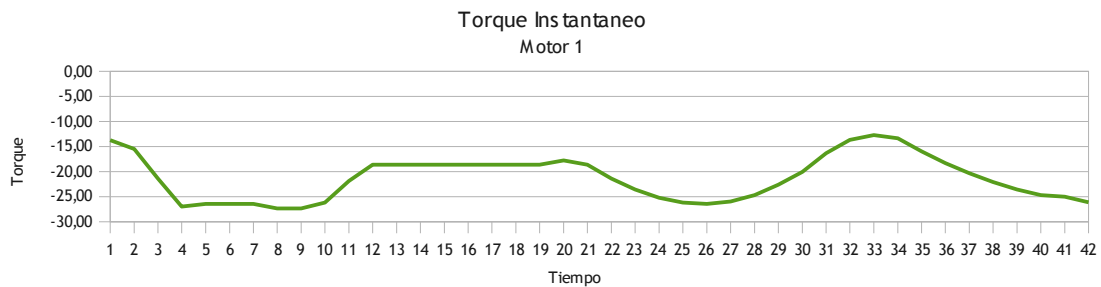
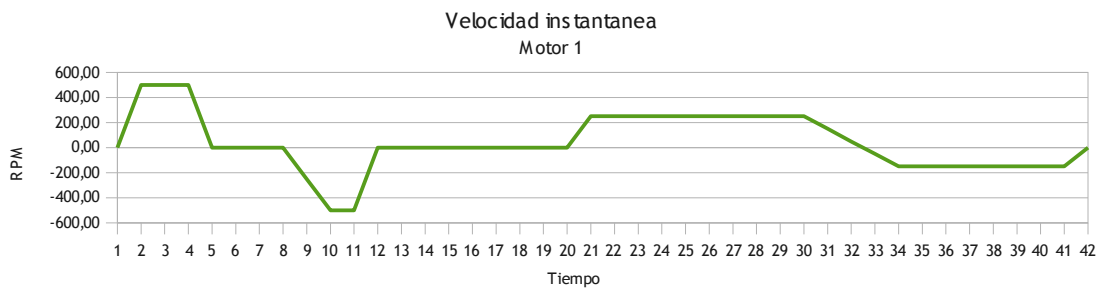
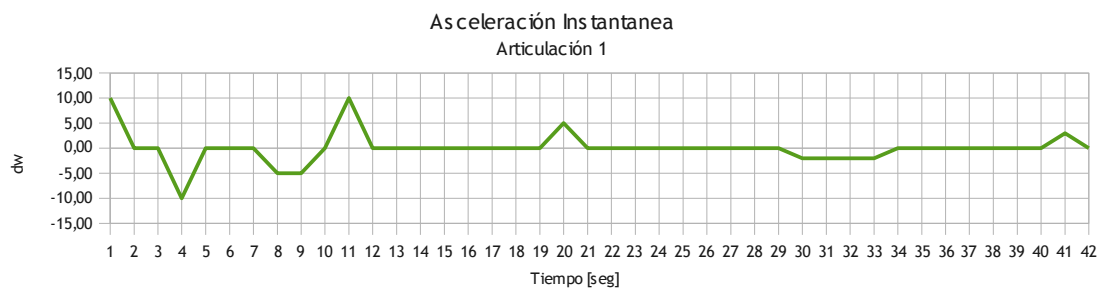
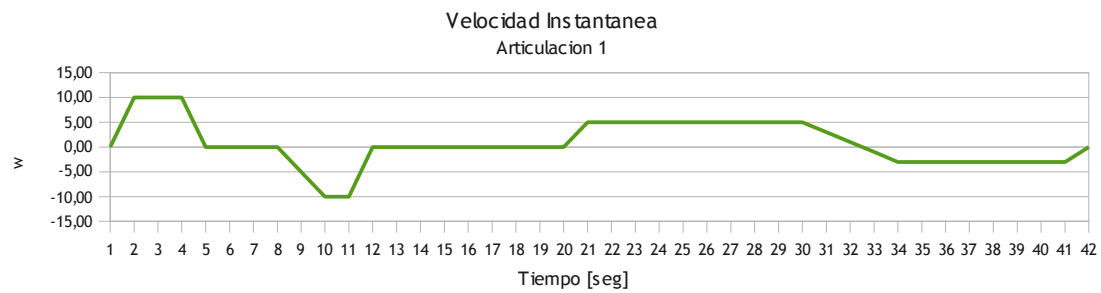
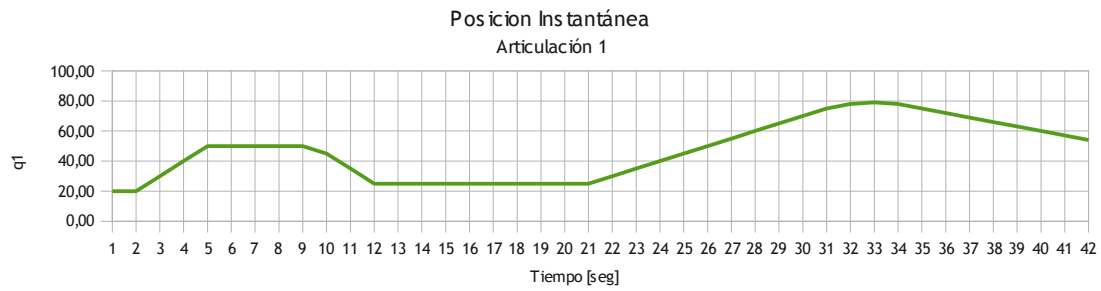
Relación de Engranajes	
Factor K	300

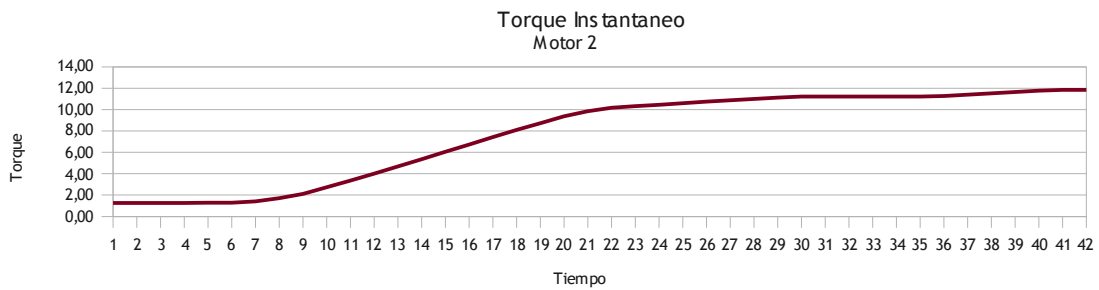
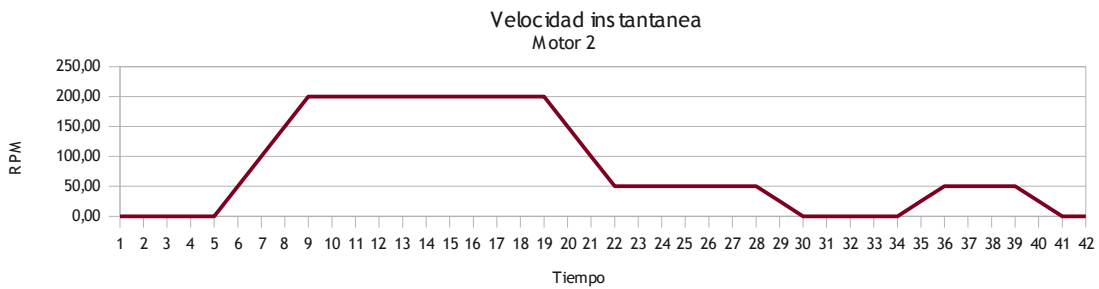
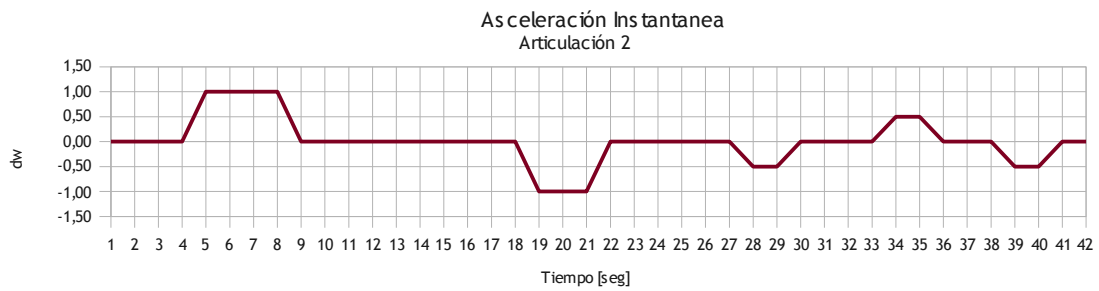
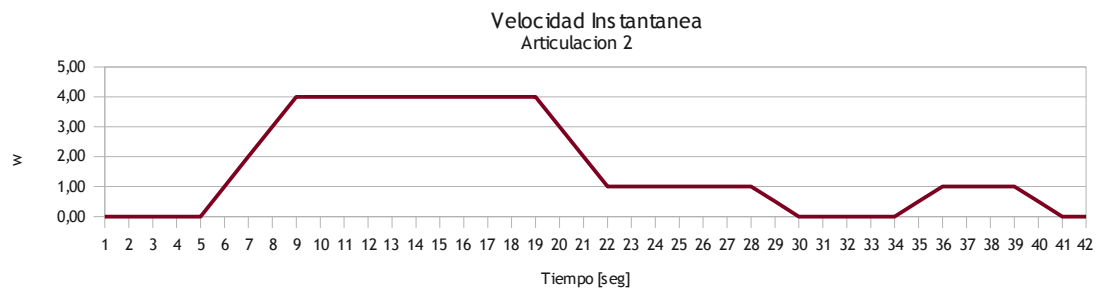
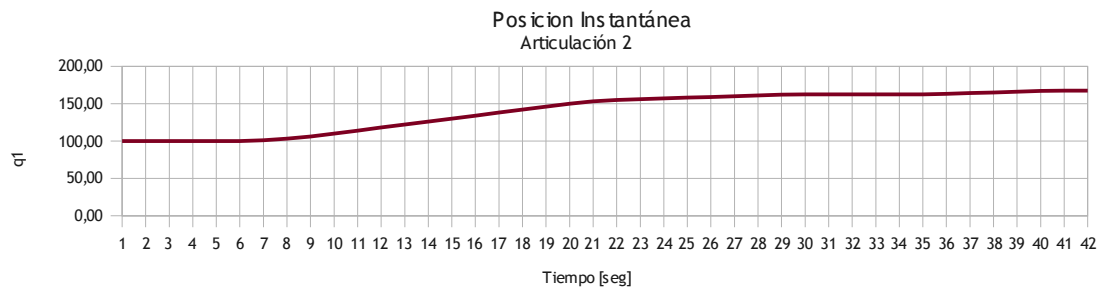
Tiempo [s]	Posición		Derivada 1°		Derivada 2°	
	q1 [°]	q2 [°]	q1 [°/s]	q2 [°/s]	q1 [°/s²]	q2 [°/s²]
1	20,00	100,00	0,00	0,00	10,00	0,00
2	20,00	100,00	10,00	0,00	0,00	0,00
3	30,00	100,00	10,00	0,00	0,00	0,00
4	40,00	100,00	10,00	0,00	-10,00	0,00
5	50,00	100,00	0,00	0,00	0,00	1,00
6	50,00	100,00	0,00	1,00	0,00	1,00
7	50,00	101,00	0,00	2,00	0,00	1,00
8	50,00	103,00	0,00	3,00	-5,00	1,00
9	50,00	106,00	-5,00	4,00	-5,00	0,00
10	45,00	110,00	-10,00	4,00	0,00	0,00
11	35,00	114,00	-10,00	4,00	10,00	0,00
12	25,00	118,00	0,00	4,00	0,00	0,00
13	25,00	122,00	0,00	4,00	0,00	0,00
14	25,00	126,00	0,00	4,00	0,00	0,00
15	25,00	130,00	0,00	4,00	0,00	0,00
16	25,00	134,00	0,00	4,00	0,00	0,00
17	25,00	138,00	0,00	4,00	0,00	0,00
18	25,00	142,00	0,00	4,00	0,00	0,00
19	25,00	146,00	0,00	4,00	0,00	-1,00
20	25,00	150,00	0,00	3,00	5,00	-1,00
21	25,00	153,00	5,00	2,00	0,00	-1,00
22	30,00	155,00	5,00	1,00	0,00	0,00
23	35,00	156,00	5,00	1,00	0,00	0,00
24	40,00	157,00	5,00	1,00	0,00	0,00
25	45,00	158,00	5,00	1,00	0,00	0,00
26	50,00	159,00	5,00	1,00	0,00	0,00
27	55,00	160,00	5,00	1,00	0,00	0,00
28	60,00	161,00	5,00	1,00	0,00	-0,50
29	65,00	162,00	5,00	0,50	0,00	-0,50
30	70,00	162,50	5,00	0,00	-2,00	0,00
31	75,00	162,50	3,00	0,00	-2,00	0,00
32	78,00	162,50	1,00	0,00	-2,00	0,00
33	79,00	162,50	-1,00	0,00	-2,00	0,00
34	78,00	162,50	-3,00	0,00	0,00	0,50
35	75,00	162,50	-3,00	0,50	0,00	0,50
36	72,00	163,00	-3,00	1,00	0,00	0,00
37	69,00	164,00	-3,00	1,00	0,00	0,00
38	66,00	165,00	-3,00	1,00	0,00	0,00
39	63,00	166,00	-3,00	1,00	0,00	-0,50
40	60,00	167,00	-3,00	0,50	0,00	-0,50
41	57,00	167,50	-3,00	0,00	3,00	0,00
42	54,00	167,50	0,00	0,00	0,00	0,00

Ø1	Ø2
-232,07	3,77
-232,07	3,77
-213,87	3,77
-189,18	3,77
-158,74	3,77
-158,74	3,77
-158,74	4,14
-158,74	4,88
-158,74	5,98
-174,63	7,42
-202,30	8,83
-223,82	10,19
-223,82	11,50
-223,82	12,76
-223,82	13,95
-223,82	15,08
-223,82	16,13
-223,82	17,11
-223,82	18,00
-223,82	18,80
-223,82	19,34
-213,87	19,67
-202,30	19,83
-189,18	19,98
-174,63	20,13
-158,74	20,27
-141,65	20,40
-123,48	20,52
-104,37	20,64
-84,47	20,70
-63,92	20,70
-51,35	20,70
-47,12	20,70
-51,35	20,70
-63,92	20,70
-76,31	20,76
-88,50	20,87
-100,45	20,97
-112,12	21,06
-123,48	21,15
-134,50	21,19
-145,16	21,19

1 [N.m]	2 [N.m]
-4119,53	376,94
-4641,33	376,94
-6416,21	376,94
-8089,09	376,94
-7937,14	384,17
-7937,14	384,17
-7937,14	425,56
-8198,04	510,18
-8198,04	634,23
-7858,22	816,67
-6558,62	1006,51
-5595,54	1202,52
-5595,54	1403,36
-5595,54	1607,64
-5595,54	1813,89
-5595,54	2020,58
-5595,54	2226,14
-5595,54	2428,96
-5595,54	2620,17
-5334,64	2812,59
-5595,54	2951,96
-6416,21	3049,35
-7080,42	3093,53
-7567,29	3137,08
-7858,22	3179,97
-7937,14	3222,17
-7790,77	3263,67
-7408,8	3300,81
-6784,04	3340,81
-6016,93	3364,13
-4898,21	3364,13
-4109,34	3364,13
-3827,01	3364,13
-4004,98	3367,74
-4793,85	3367,74
-5494,67	3383,64
-6106,68	3422,04
-6629,55	3459,61
-7063,4	3492,71
-7408,8	3528,54
-7510,19	3549,74
-7838,61	3549,74

Parámetros instantáneos del motor			
W1 RPM	W2 RPM	t1 [N.m]	t2 [N.m]
0,00	0,00	-13,73	1,26
500,00	0,00	-15,47	1,26
500,00	0,00	-21,39	1,26
500,00	0,00	-26,96	1,26
0,00	0,00	-26,46	1,28
0,00	50,00	-26,46	1,28
0,00	100,00	-26,46	1,42
0,00	150,00	-27,33	1,70
-250,00	200,00	-27,33	2,11
-500,00	200,00	-26,19	2,72
-500,00	200,00	-21,86	3,36
0,00	200,00	-18,65	4,01
0,00	200,00	-18,65	4,68
0,00	200,00	-18,65	5,36
0,00	200,00	-18,65	6,05
0,00	200,00	-18,65	6,74
0,00	200,00	-18,65	7,42
0,00	200,00	-18,65	8,10
0,00	200,00	-18,65	8,73
0,00	150,00	-17,78	9,38
250,00	100,00	-18,65	9,84
250,00	50,00	-21,39	10,16
250,00	50,00	-23,60	10,31
250,00	50,00	-25,22	10,46
250,00	50,00	-26,19	10,60
250,00	50,00	-26,46	10,74
250,00	50,00	-25,97	10,88
250,00	50,00	-24,70	11,00
250,00	25,00	-22,61	11,14
250,00	0,00	-20,06	11,21
150,00	0,00	-16,33	11,21
50,00	0,00	-13,70	11,21
-50,00	0,00	-12,76	11,21
-150,00	0,00	-13,35	11,23
-150,00	25,00	-15,98	11,23
-150,00	50,00	-18,32	11,28
-150,00	50,00	-20,36	11,41
-150,00	50,00	-22,10	11,53
-150,00	50,00	-23,54	11,64
-150,00	25,00	-24,70	11,76
-150,00	0,00	-25,03	11,83
0,00	0,00	-26,13	11,83

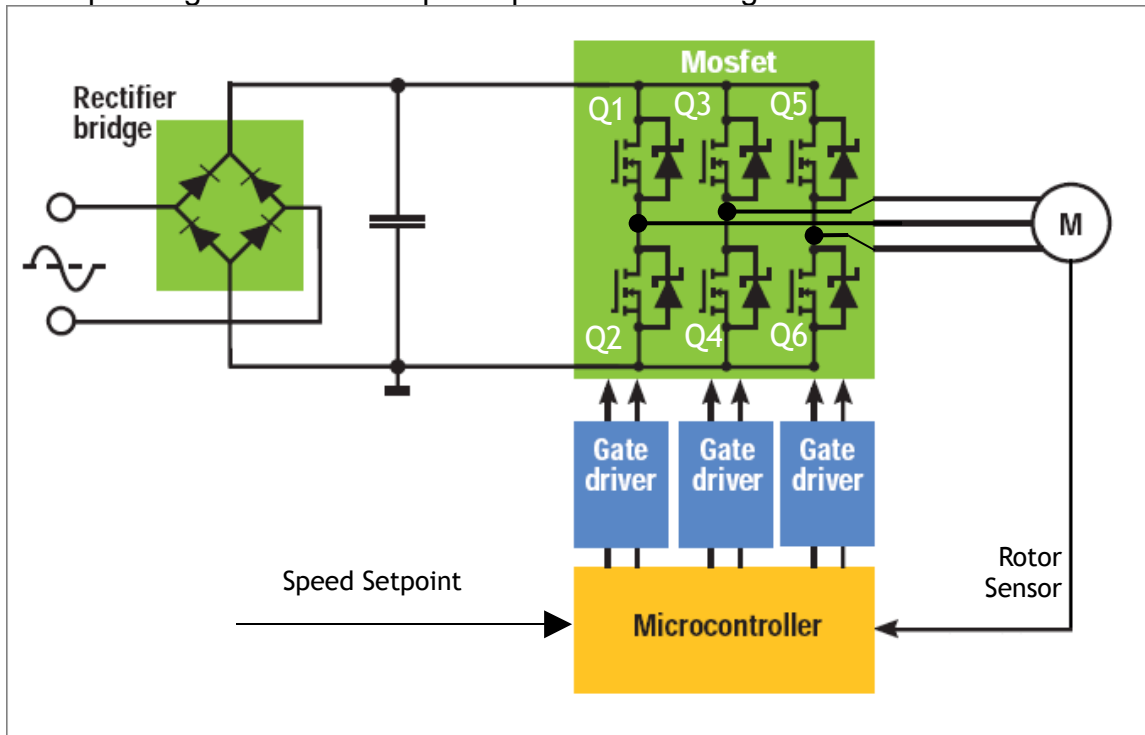
Parámetros de la articulación 1

Parámetros de la articulación 2

Implementación en código VHDL:

Utilizando el software *ModelSim PE Student Edition 6.5b* de *Mentor Graphics*, implementamos el código VHDL del controlador de una etapa de potencia para manejar un motor brushless utilizando salidas PWM.

El esquema general de la etapa de potencia es el siguiente:



Para que el motor gire en un sentido determinado, deben conmutarse los transistores del puente de manera de generar un campo rotacional en ese sentido, a medida que los sensores de efecto Hall indican el cambio de la posición angular del rotor.

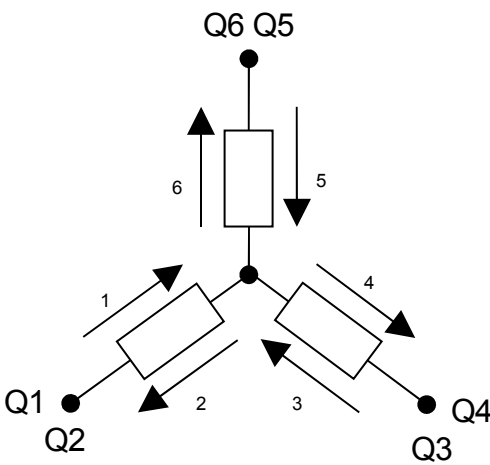
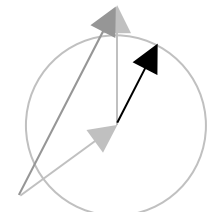
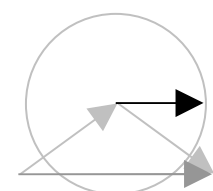
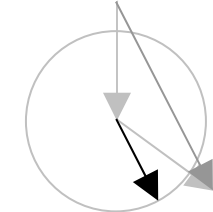
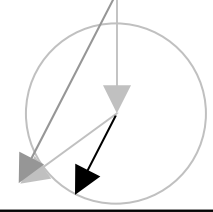
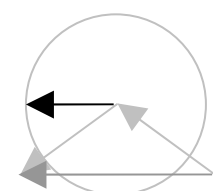
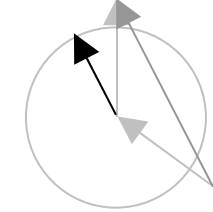
Para que circule corriente por el motor, es necesario activar al menos 2 transistores, uno de la parte superior y otro de la parte inferior del puente, quedando los mismos conectados en serie. Por este motivo es posible mantener encendidos sin modular los transistores superiores todo el tiempo que corresponda a la secuencia de activación. La variación deseada en la tensión de salida, se logra modulando en ancho de pulso solamente los transistores de la parte inferior. Para ello utilizamos internamente una consigna fija con el valor máximo en los PWM 1, 3 y 5; mientras aplicamos la consigna externa al resto (PWM 2, 4 y 6).

Utilizamos la siguiente secuencia de 6 pasos:

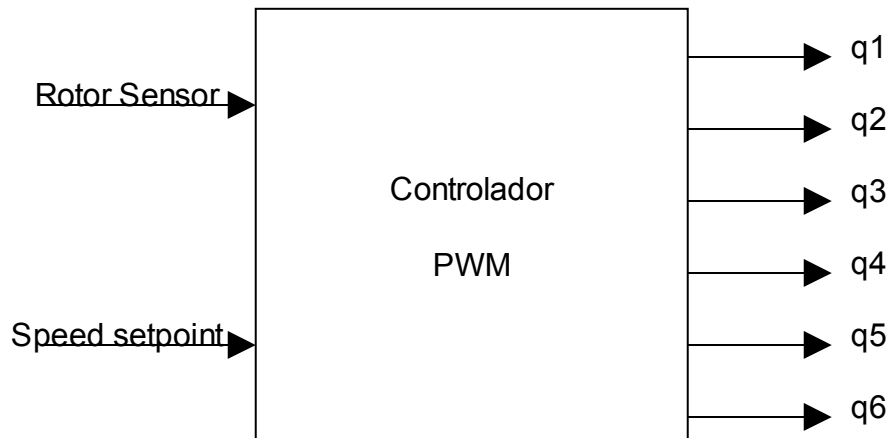
Paso	Q Activos (+) (-)	Transistores superiores (+)			Transistores inferiores (-)		
		Q1	Q3	Q5	Q2	Q4	Q6
5	3 - 6	OFF	ON	OFF	OFF	OFF	ON
0	1 - 6	ON	OFF	OFF	OFF	OFF	ON
1	1 - 4	ON	OFF	OFF	OFF	ON	OFF
2	5 - 4	OFF	OFF	ON	OFF	ON	OFF
3	5 - 2	OFF	OFF	ON	ON	OFF	OFF
4	3 - 2	OFF	ON	OFF	ON	OFF	OFF
5	3 - 6	OFF	ON	OFF	OFF	OFF	ON
0	1 - 6	ON	OFF	OFF	OFF	OFF	ON

Invirtiendo esta secuencia, se consigue invertir el sentido de giro del motor, ya que el campo magnético resultante giraría en sentido inverso al de una secuencia directa.

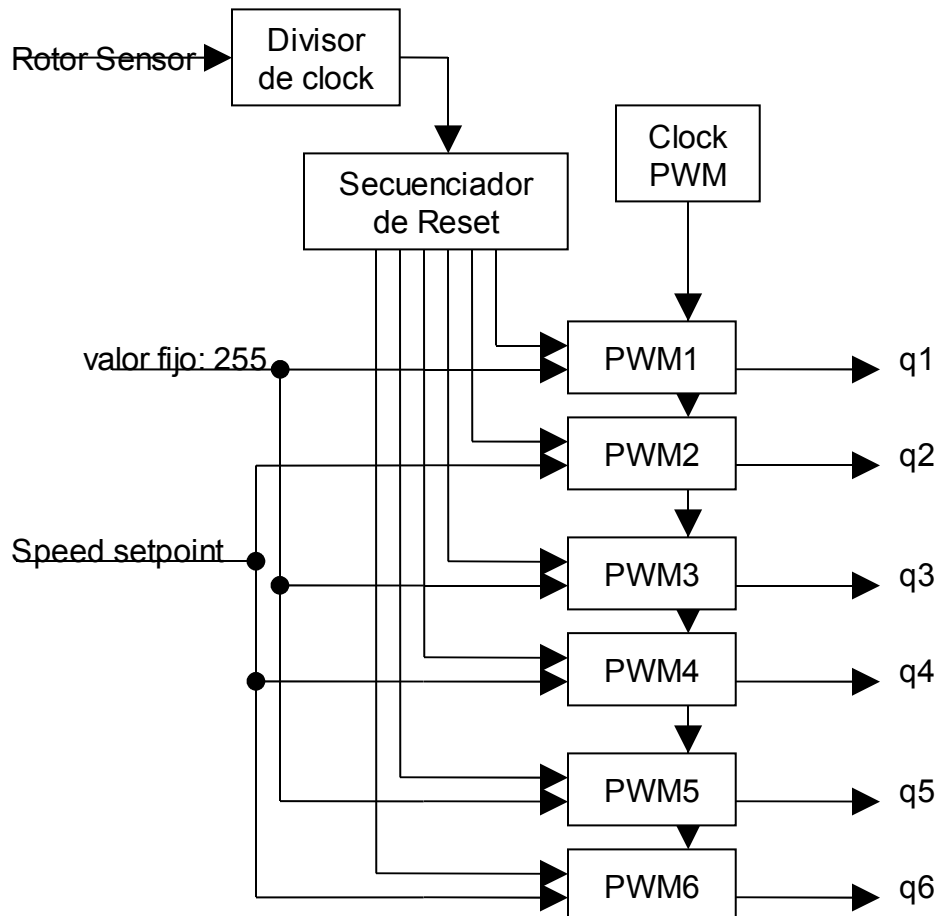
En la página siguiente se encuentra una tabla donde representamos la construcción del campo magnético en rotación.

Disposición de las bobinas del motor	Paso	Q Activos	Campo resultante
	0	1 – 6	
	1	1 – 4	
	2	5 – 4	
	3	5 – 2	
	4	3 – 2	
	5	3 – 6	

Del esquema general, desarrollamos la parte del Microcontrolador en VHDL:



El diagrama en bloques interno del controlador es el siguiente:



El código fue desarrollado en forma modular, a continuación se describe cada uno de los módulos:

pwm_fpga.vhd

○Descripción:

Es el bloque generador de señal PWM. Utiliza las funciones `INC()` y `DEC()`, definidas en el archivo `user_pkg_inc_dec.vhd`.

○Entradas:

- `clock`: señal de reloj de base para la salida PWM. El período de la salida es 256 veces el de la señal de clock.
- `reset`: fuerza la salida a estado inactivo y pone el contador interno en cero.
- `Data_value`: 8 bits de datos con el valor de la consigna (0 – 255).

○Salidas:

- `pwm`: salida cuyo valor medio será directamente proporcional al valor de entrada.

divisor_clock.vhd

○Descripción:

Este bloque auxiliar genera un clock a la salida cuya frecuencia está dividida N veces respecto de la de entrada.

○Entradas:

- `clk_in`: señal de reloj de entrada.

○Salidas:

- `clk_out`: señal de reloj de salida.

driver_pwm.vhd

○Descripción:

Es el bloque principal del controlador PWM. Utiliza 1 bloque divisor de clock para procesar la señal de entrada proveniente de los sensores Hall del motor. Genera la secuencia de resets en función del sentido de giro. Utiliza 6 bloques PWM, uno para cada salida. Estas salidas controlarían la conducción de los transistores del puente trifásico.

○Entradas:

- `clk_pwm`: señal de reloj necesaria para el funcionamiento de los bloques PWM.
- `clk_motor`: señal de realimentación de la posición del rotor, proveniente de los sensores de efecto Hall.
- `reset`: señal de reset general del módulo. Desactiva todas las salidas y fuerza la secuencia inicial a cero.
- `sentido`: selecciona el sentido de giro del motor. La secuencia de activación de los transistores del puente depende de esta entrada.
- `Data_value`: 8 bits de datos con el valor de la consigna de velocidad para el motor (0 – 255).

- **Salidas:**

- q_1, q_2, \dots, q_6 : señales de activación de los transistores de salida del puente trifásico, de manera que las conmutaciones provoquen un giro en el motor, en el sentido seleccionado. La tensión media es proporcional a la consigna.

driver_tb.vhd

- **Descripción:**

Este módulo no implementa ninguna funcionalidad, sino que se utiliza para testear o simular el funcionamiento de **driver_pwm.vhd**. Genera las señales de clock tanto para los PWM como para la posición del rotor, y también genera una rampa en la consigna de velocidad. Utiliza un sentido de giro fijo.

- **Entradas:** ninguna.

- **Salidas:** ninguna.

Resultados de la simulación

Simulamos el arranque del motor utilizando una rampa como consigna de velocidad, para visualizar las señales resultantes a la salida del módulo **driver_pwm.vhd**.

Nota: Para esta simulación no se tuvieron en cuenta las inercias ni la velocidad real del rotor en cada momento del arranque. Simplemente se trata de ver el comportamiento de las salidas frente a la secuencia de giro y a la variación de la consigna de velocidad. De la misma forma, los tiempos y frecuencias mostrados en los gráficos son ilustrativos y no se corresponden con los reales de un motor.

En cada gráfico, se muestran las señales de entrada y salida en función del tiempo. El detalle es el siguiente:

sig_clock_pwm: señal de reloj para los PWM.

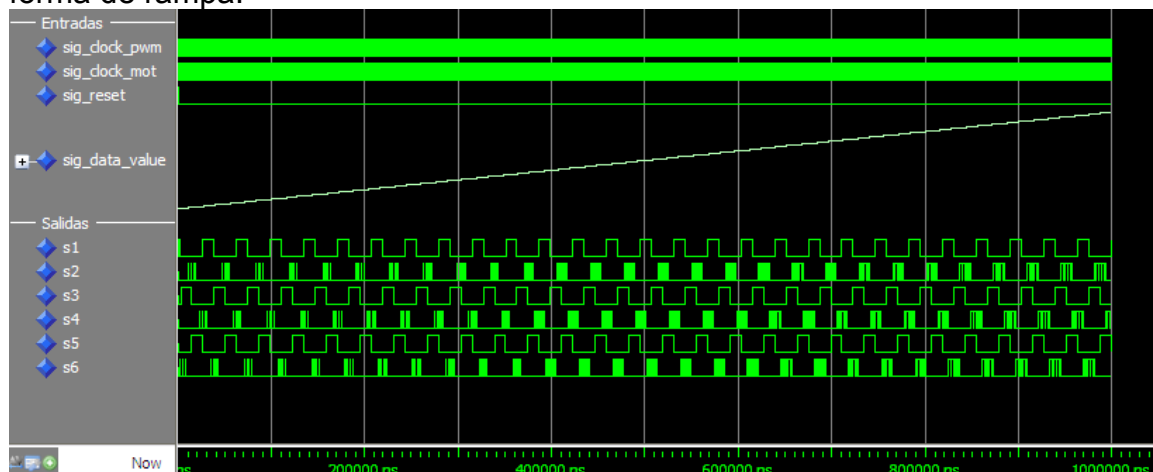
sig_clock_mot: señal de reloj que simula los sensores de efecto Hall del motor.

sig_reset: señal de reset general controlador, se activa por un corto lapso, sólo al inicio.

sig_data_value: valor de la consigna de velocidad (8 bits, sin signo) Se genera una rampa desde 0 hasta 250.

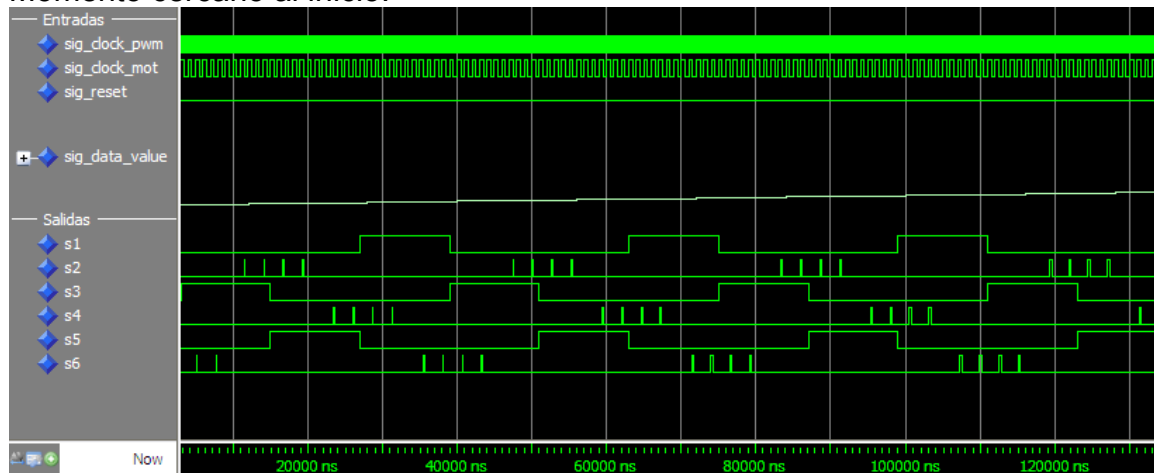
s1, s2,...,s6: conmutación de los transistores de salida del puente trifásico.

A continuación podemos ver la simulación completa, con la consigna variando en forma de rampa:

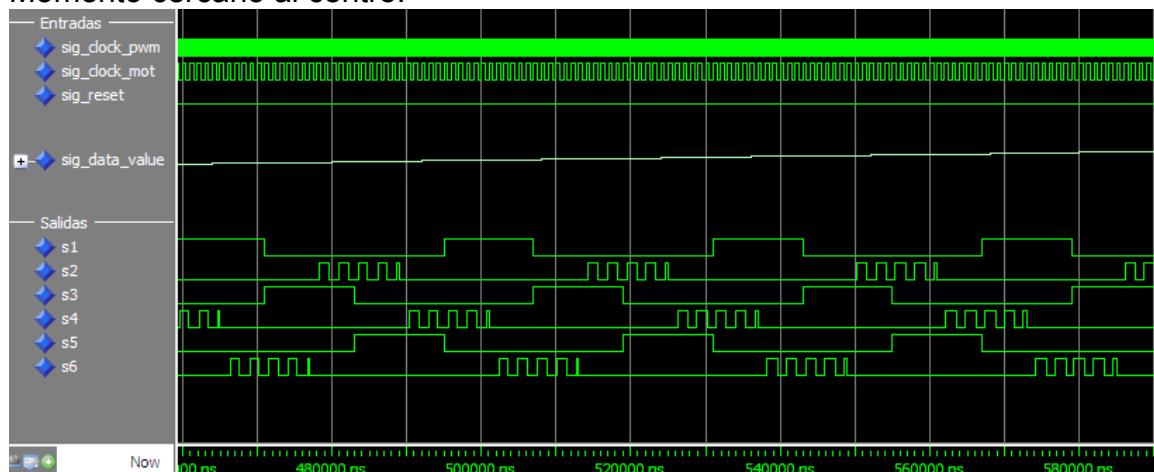


Haciendo un zoom en 3 momentos diferentes de la simulación, podemos comprobar la variación del ciclo de actividad de los PWM.

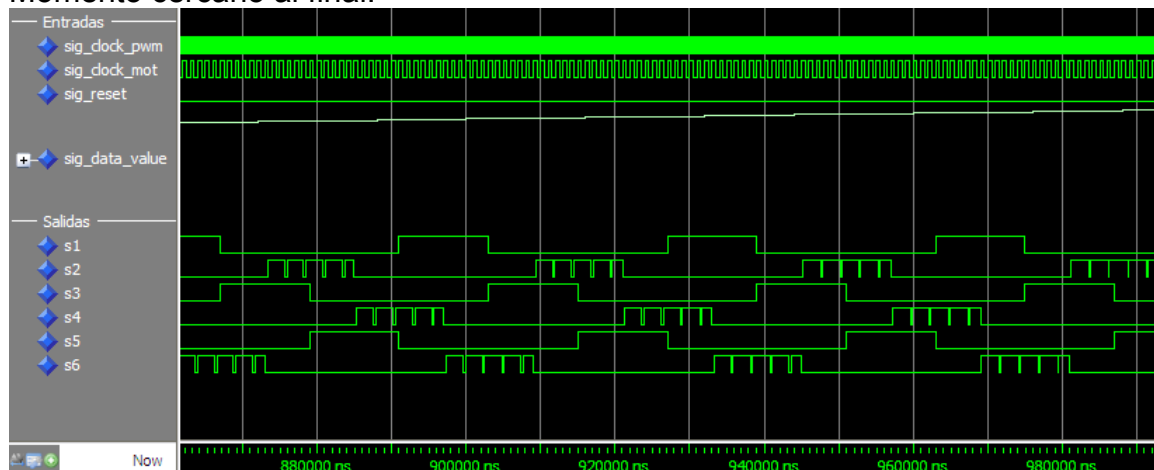
Momento cercano al inicio:



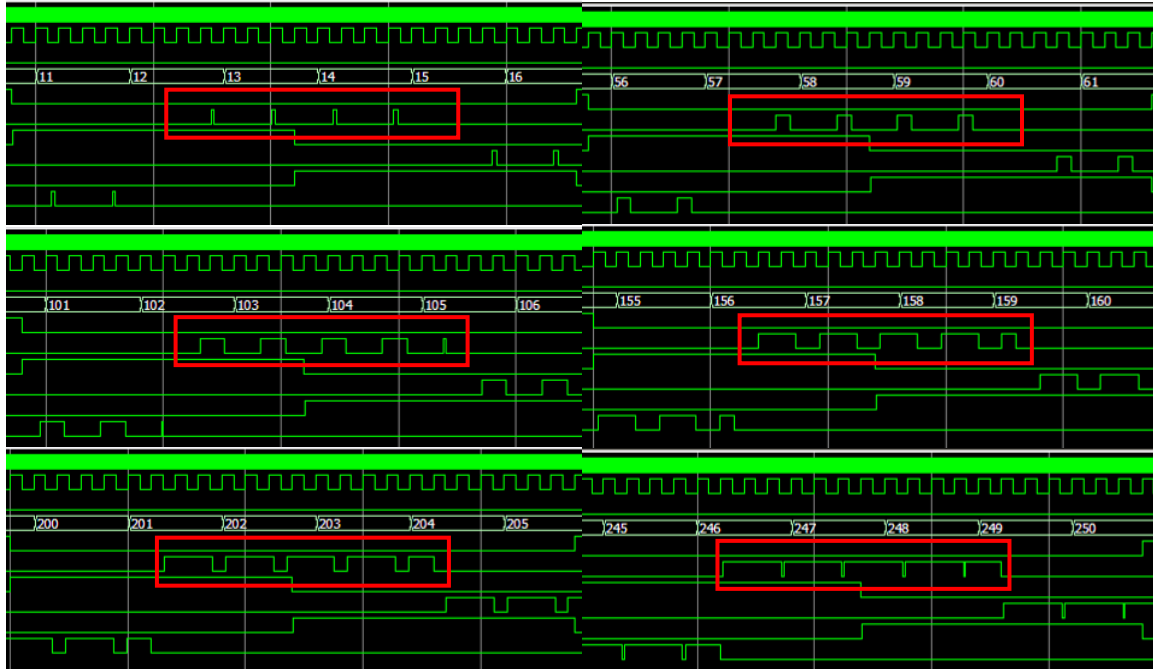
Momento cercano al centro:



Momento cercano al final:

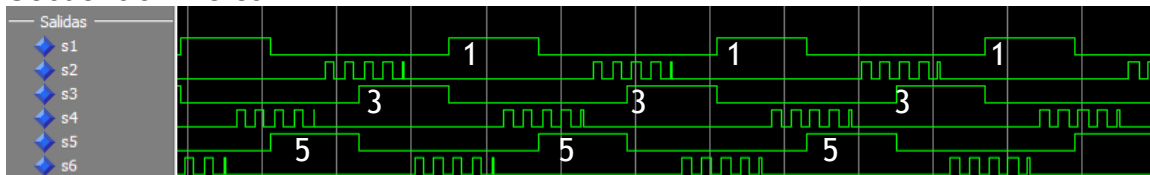


Aquí un detalle mayor de la variación del PWM, donde se pueden ver los valores de la consigna en color blanco:

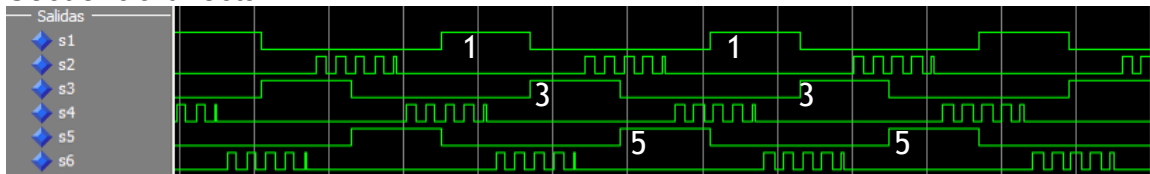


Finalmente comprobamos la correcta generación de la secuencia inversa cambiando el valor de la entrada sentido:

Secuencia inversa:



Secuencia directa:



Conclusiones finales

Se puede apreciar la particularidad de esta configuración paralelogramo, que si se respetan las condiciones impuestas, nos da la clave de la popularidad de esta configuración para los robots industriales.

Cumplidas las condiciones anteriores, se puede operar q_1 y q_2 independientemente, sin preocuparse por las interacciones entre los dos ángulos, lo que facilita el control dinámico del robot.

Además, comprobamos mediante simulación el funcionamiento un controlador para un motor brushless variando la tensión por medio de modulación de ancho de pulso.

APENDICE 1

Codigo VHDL de usado en la simulación

user_pkg_inc_dec.vhd

```
-- *****
--
-- Company           : ATMEL CORPORATION
-- File Name         : user_pkg_inc_dec.vhd
-- Title            : USER DEFINED PACKAGE FILE
-- Version           : 1.0
-- Last updated      : 05/24/01
-- Target            :
--
-- Support email     : fpga@atmel.com
-- Support Hotline    : +1(408)436-4119 (USA)
--
-- Revision          : 1.0
--
-- OVERVIEW
-- This code describes the function used in the pwm_fpga.vhd file
--
-- *****

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

PACKAGE user_pkg IS
    function INC(X: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;
    function DEC(X: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;
END user_pkg ;

PACKAGE BODY user_pkg IS

function INC(X: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR is
    variable XV: STD_LOGIC_VECTOR(X'LENGTH - 1 downto 0);
    begin
        XV := X;
        for I in 0 to XV'HIGH LOOP
            if XV(I) = '0' then
                XV(I) := '1';
                exit;
            else XV(I) := '0';
            end if;
        end loop;

    return XV;
end INC;

function DEC(X: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR is
    variable XV: STD_LOGIC_VECTOR(X'LENGTH - 1 downto 0);
    begin
        XV := X;
        for I in 0 to XV'HIGH LOOP
            if XV(I) = '1' then
                XV(I) := '0';
                exit;
            else XV(I) := '1';
            end if;
        end loop;

    return XV;
end DEC;

END user_pkg;
```

pwm_fpga.vhd

```

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);
    ELSIF reg_out="11111111" THEN
        pwm_int <= '1';
    ELSE
        pwm_int <= pwm_int;
    END IF;
END PROCESS;
pwm <= pwm_int;

END arch_pwm;
```

driver_tb.vhd

```
LIBRARY ieee;

use ieee.std_logic_1164.all;
USE      work.user_pkg.all;

ENTITY driver_tb IS

END driver_tb;

ARCHITECTURE default OF driver_tb IS

    COMPONENT driver_pwm
    port(  clk_pwm, clk_motor, reset :  in std_logic;
          Data_value :in std_logic_vector(7 downto 0);
          q1, q2, q3, q4, q5, q6:      out std_logic);
    END COMPONENT;

    -- Internal signal declaration

    SIGNAL sig_clock_pwm   : std_logic :='0' ;
    SIGNAL sig_clock_mot   : std_logic :='0' ;
    SIGNAL sig_reset       : std_logic :='0';
    SIGNAL sig_data_value  : std_logic_vector(7 downto 0) := "00000100" ;
    SIGNAL s1,s2,s3,s4,s5,s6      : std_logic;
    shared variable ENDSIM: boolean:=false;
    constant pwm_period : TIME := 10 ns; --periodo del clock PWM
    constant mot_period : TIME := 1 us;  --periodo del clock motor
```

```
BEGIN

    -- Generacion del clock de los pwm
    clk1_gen: process

        BEGIN

            If ENDSIM = FALSE THEN
                sig_clock_pwm <= '1';
                wait for pwm_period/2;
                sig_clock_pwm <= '0';
                wait for pwm_period/2;
            else
                wait;
            end if;
        end process;

    -- Generacion del clock de la pos del motor
    clk2_gen: process

        BEGIN

            If ENDSIM = FALSE THEN
                sig_clock_mot <= '1';
                wait for mot_period/2;
                sig_clock_mot <= '0';
                wait for mot_period/2;
            else
                wait;
            end if;
        end process;

-- Instantiating top level design Component pwm_fpga

inst_pwm_fpga : driver_pwm
PORT MAP(
    clk_pwm=> sig_clock_pwm,
    clk_motor => sig_clock_mot,
    reset => sig_reset,
    Data_value => sig_data_value,
    q1 => s1,
    q2 => s2,
    q3 => s3,
    q4 => s4,
    q5 => s5,
    q6 => s6
);

stimulus_process: PROCESS

BEGIN

    sig_reset <= '1';
    wait for 47 ns;
    sig_reset <= '0';

    for I in 1 to 1000 loop    --simular por 1ms!!!
        sig_data_value <= INC(sig_data_value) ;
        wait for 5 us;
    end loop;

    ENDSIM := TRUE;
    wait;

END PROCESS stimulus_process;

END default;
```

driver_pwm.vhd

```

library IEEE;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all ;
--USE work.pwm_fpga.all;

entity driver_pwm is
port( clk_pwm, clk_motor, reset : in std_logic;
      Data_value :in std_logic_vector(7 downto 0);
      q1, q2, q3, q4, q5, q6: out std_logic);
end driver_pwm;

ARCHITECTURE default OF driver_pwm IS

SIGNAL valor_h : std_logic_vector(7 downto 0) := "11111111";
SIGNAL valor_l : std_logic_vector(7 downto 0) := "00000000";
SIGNAL secuencia : integer :=0 ;
signal pwm1, pwm2, pwm3, pwm4, pwm5, pwm6 : std_logic;
signal rst1, rst2, rst3, rst4, rst5, rst6 : std_logic;
signal clk_motor_div : std_logic;

BEGIN

DIVCLK: entity WORK.divisor_clock port map (clk_motor, clk_motor_div);

valor_l <= Data_value; --solo actualizo los pwm de abajo

--Calculo de las senales de reset de los modulos PWM
--en funcion de la secuencia de giro del motor
process (secuencia, reset)
begin
    rst1<='1';
    rst2<='1';
    rst3<='1';
    rst4<='1';
    rst5<='1';
    rst6<='1';

    if reset='0' then
        case secuencia is
            when 0 => -- secuencia 0
                rst1 <= '0';
                rst6 <= '0';
            when 1 => -- secuencia 1
                rst1 <= '0';
                rst4 <= '0';
            when 2 => -- secuencia 2
                rst5 <= '0';
                rst4 <= '0';
            when 3 => -- secuencia 3
                rst5 <= '0';
                rst2 <= '0';
            when 4 => -- secuencia 4
                rst3 <= '0';
                rst2 <= '0';
            when others => -- secuencia 5
                rst3 <= '0';
                rst6 <= '0';
            end case;
        end if;
    end process;

```

```

--Calculo de la secuencia de giro del motor
process (secuencia, reset, clk_motor_div)
begin
    if reset='1' then
        secuencia <= 0;
    elsif clk_motor_div'event and clk_motor_div = '1' then
        if secuencia < 5 then
            secuencia <= secuencia +1 ;
        else
            secuencia <= 0;
        end if;
    end if;
end process;

--- Modulos PWM
PWM1_I: entity WORK.pwm_fpga port map (clk_pwm, rst1, valor_h, pwm1);
PWM2_I: entity WORK.pwm_fpga port map (clk_pwm, rst2, valor_l, pwm2);
PWM3_I: entity WORK.pwm_fpga port map (clk_pwm, rst3, valor_h, pwm3);
PWM4_I: entity WORK.pwm_fpga port map (clk_pwm, rst4, valor_l, pwm4);
PWM5_I: entity WORK.pwm_fpga port map (clk_pwm, rst5, valor_h, pwm5);
PWM6_I: entity WORK.pwm_fpga port map (clk_pwm, rst6, valor_l, pwm6);

q1 <= pwm1; --Salidas del driver (a los transistores de salida)
q2 <= pwm2;
q3 <= pwm3;
q4 <= pwm4;
q5 <= pwm5;
q6 <= pwm6;

END default;

```

[divisor_clock.vhd](#)

```

library IEEE;
use IEEE.std_logic_1164.all;

entity divisor_clock is
    generic ( cantidad_ciclos : INTEGER := 3);
    PORT (
        clk_in: in std_logic;
        clk_out: out std_logic
    );
end divisor_clock;

architecture default of divisor_clock is

    signal cuenta :      INTEGER :=0 ;
    signal salida : std_logic :='0' ;

begin

    process (clk_in, cuenta, salida)
    begin
        if (clk_in'event AND clk_in = '1') then
            cuenta <= cuenta + 1;
            if (cuenta >= cantidad_ciclos) then
                salida <= not salida;
                cuenta <= 1;
            end if;
        end if;
        clk_out <= salida;
    end process;

end default;

```