

# **Procesamiento Digital de Señales**

## **Trabajo Práctico**

**Jairo Mena**

**Presentado a:  
Gonzalo Lavigna**

**Universidad de Buenos Aires  
MSE 5Co2020**

Repository: <https://github.com/jamenaso/PDS.git>

## 1. Ejercicio 1 “Repaso VHDL”

### a. Parte 1 ALU Simple:

ALU Simple con entradas y salidas de 16 bits.

$\text{SEL} = "00" \Rightarrow C = A + B$   
 $\text{SEL} = "01" \Rightarrow C = A - B$   
 $\text{SEL} = "10" \Rightarrow C = A \mid B$  (bitwise OR)  
 $\text{SEL} = "11" \Rightarrow C = A \& B$  (bitwise AND)

### Código RTL

```

1. library ieee;
2. use ieee.std_logic_1164.all;
3. use ieee.numeric_std.all;
4.
5. entity simple_alu is
6.   generic (N : integer);
7.   port (
8.     sel_i : in std_logic_vector(1 downto 0);
9.     a_i : in std_logic_vector(N-1 downto 0);
10.    b_i : in std_logic_vector(N-1 downto 0);
11.    c_o : out std_logic_vector(N-1 downto 0)
12.  );
13. end simple_alu;
14.
15. architecture Behavioral of simple_alu is
16.
17. begin
18.
19.   --Se implementa un multiplexor donde la operación de salida en (c_o)
20.   --depende de la señal de selección (sel_i)
21.   out_mux :
22.   c_o <=  std_logic_vector(unsigned(a_i) + unsigned(b_i)) when sel_i = "00" else
23.             std_logic_vector(unsigned(a_i) - unsigned(b_i)) when sel_i = "01" else
24.             a_i or b_i when sel_i = "10" else
25.             a_i and b_i when sel_i = "11";
26.
27. end Behavioral;

```

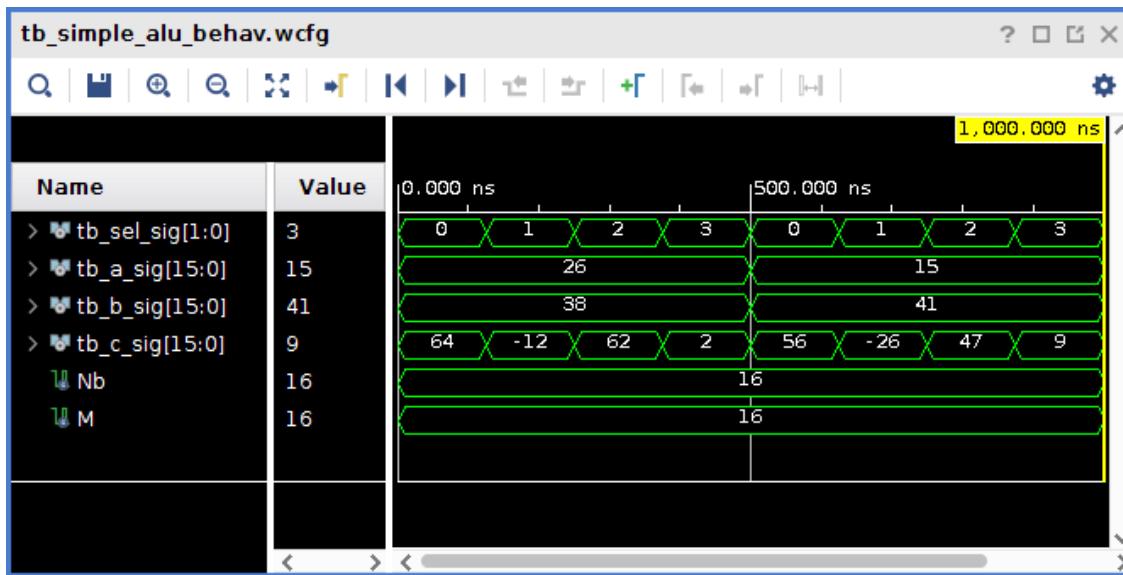
### Código de verificación

```

1. library ieee;
2. use ieee.std_logic_1164.all;
3. use ieee.numeric_std.all;
4.
5. entity tb_simple_alu is
6.   generic (Nb : integer := 16);
7. end tb_simple_alu;
8.
9. architecture Behavioral of tb_simple_alu is
10.
11.   component simple_alu is
12.     generic (N : integer);
13.     port (
14.       sel_i : in std_logic_vector(1 downto 0);
15.       a_i : in std_logic_vector(N-1 downto 0);
16.       b_i : in std_logic_vector(N-1 downto 0);
17.       c_o : out std_logic_vector(N-1 downto 0)
18.     );
19.   end component simple_alu;
20.
21.   constant M : integer := Nb;
22.   signal tb_sel_sig : std_logic_vector(1 downto 0);
23.   signal tb_a_sig : std_logic_vector(Nb-1 downto 0);
24.   signal tb_b_sig : std_logic_vector(Nb-1 downto 0);
25.   signal tb_c_sig : std_logic_vector(Nb-1 downto 0);
26. begin
27.   inst_simple_alu : simple_alu
28.   generic map(
29.     N => 16
30.   )
31.   port map(
32.     sel_i    => tb_sel_sig,
33.     a_i      => tb_a_sig,
34.     b_i      => tb_b_sig,
35.     c_o      => tb_c_sig
36.   );
37.
38. --Se implementa un proceso que introduce dos valores de entrada (26 y 38) y realiza
39. --las cuatro operaciones cambiando la señal de selección sobre la instancia de la ALU,
40. --realiza el mismo proceso con otro dos valores de entrada (15 y 38)
41. process
42. begin
43.   tb_a_sig  <= std_logic_vector(to_unsigned(26,M));
44.   tb_b_sig  <= std_logic_vector(to_unsigned(38,M));
45.   tb_sel_sig  <= "00";
46.   wait for 125 ns;
47.   tb_sel_sig  <= "01";
48.   wait for 125 ns;
49.   tb_sel_sig  <= "10";
50.   wait for 125 ns;
51.   tb_sel_sig  <= "11";
52.   wait for 125 ns;
53.   tb_a_sig  <= std_logic_vector(to_unsigned(15,M));
54.   tb_b_sig  <= std_logic_vector(to_unsigned(41,M));
55.   tb_sel_sig  <= "00";
56.   wait for 125 ns;
57.   tb_sel_sig  <= "01";
58.   wait for 125 ns;
59.   tb_sel_sig  <= "10";
60.   wait for 125 ns;
61.   tb_sel_sig  <= "11";
62.   wait for 125 ns;
63.   wait;
64. end process;
65. end Behavioral;

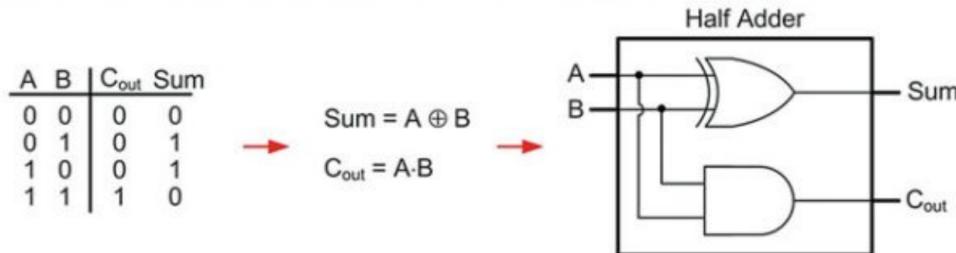
```

## Simulación



### b. Parte 2 Half Adder:

Half Adder de dos bits de entrada, con bit de suma de salida y bit de carry.



### Código RTL

```

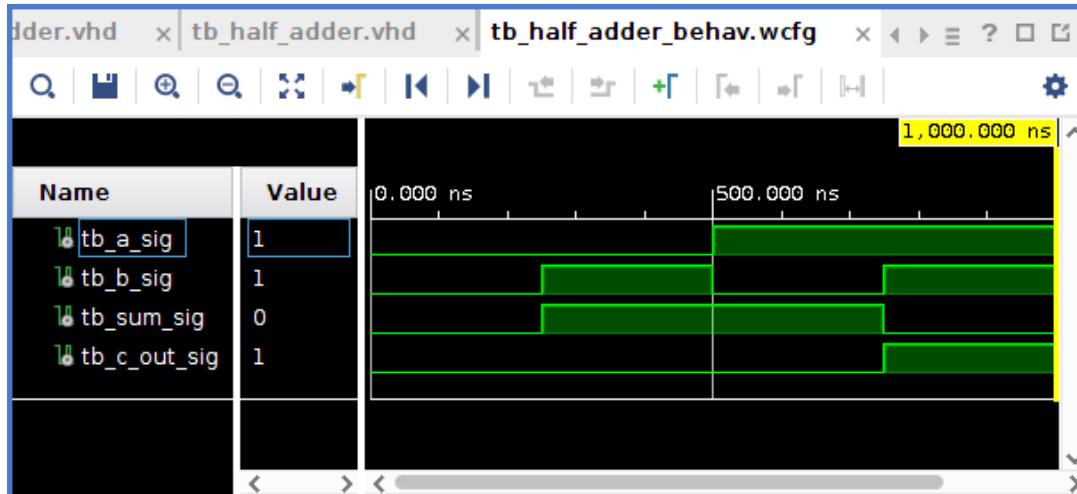
1. library ieee;
2. use ieee.std_logic_1164.all;
3. use ieee.numeric_std.all;
4.
5. entity half_adder is
6.   Port ( a_i : in std_logic;
7.         b_i : in std_logic;
8.         sum_o : out std_logic;
9.         c_out_o : out std_logic);
10. end half_adder;
11.
12. architecture Behavioral of half_adder is
13.
14. begin
15.
16.   sum_o  <=  a_i xor b_i;
17.   c_out_o <=  a_i and b_i;
18.
19. end Behavioral;

```

### Código de verificación

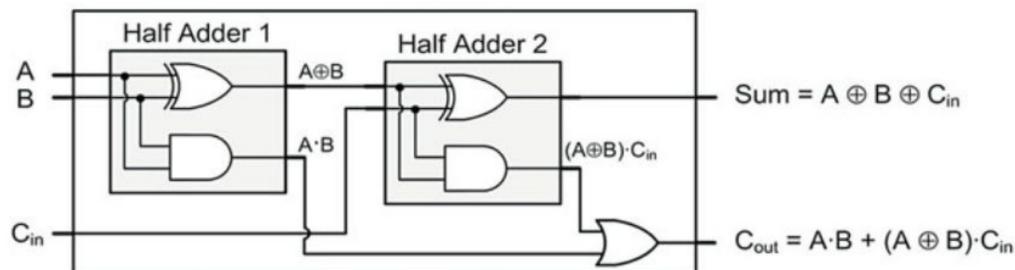
```
1.  library ieee;
2.  use ieee.std_logic_1164.all;
3.  use ieee.numeric_std.all;
4.
5.  entity tb_half_adder is
6.  end tb_half_adder;
7.
8.  architecture Behavioral of tb_half_adder is
9.
10. component half_adder is
11.    Port ( a_i : in std_logic;
12.           b_i : in std_logic;
13.           sum_o : out std_logic;
14.           c_out_o : out std_logic);
15.  end component half_adder;
16.
17.      signal tb_a_sig : std_logic;
18.      signal tb_b_sig : std_logic;
19.      signal tb_sum_sig : std_logic;
20.      signal tb_c_out_sig : std_logic;
21.
22. begin
23.
24.     inst_half_adder : half_adder
25.     port map
26.     (
27.         a_i      =>      tb_a_sig,
28.         b_i      =>      tb_b_sig,
29.         sum_o   =>      tb_sum_sig,
30.         c_out_o =>      tb_c_out_sig
31.     );
32.
33.     --Proceso que simula todos lo posible estados de las entrada a y b
34. process
35. begin
36.     tb_a_sig <= '0';
37.     tb_b_sig <= '0';
38.     wait for 250 ns;
39.     tb_a_sig <= '0';
40.     tb_b_sig <= '1';
41.     wait for 250 ns;
42.     tb_a_sig <= '1';
43.     tb_b_sig <= '0';
44.     wait for 250 ns;
45.     tb_a_sig <= '1';
46.     tb_b_sig <= '1';
47.     wait for 250 ns;
48.     wait;
49. end process;
50.
51. end Behavioral;
```

## Simulación



### c. Parte 3 Full Adder:

Full Adder compuesto por dos Half Adder



C <sub>in</sub>	A	B	C <sub>out</sub>	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

## Código RTL

```

1.  library ieee;
2.  use ieee.std_logic_1164.all;
3.  use ieee.numeric_std.all;
4.
5.  entity full_adder is
6.    Port ( a_i : in std_logic;
7.           b_i : in std_logic;
8.           c_in_i : in std_logic;
9.           sum_o : out std_logic;
10.          c_out_o : out std_logic);
11. end full_adder;
12.
13. architecture Behavioral of full_adder is
14.
15.   component half_adder is
16.     Port ( a_i : in std_logic;
17.             b_i : in std_logic;
18.             sum_o : out std_logic;
19.             c_out_o : out std_logic);
20.   end component half_adder;
21.
22.   signal a_xor_b_sig : std_logic;
23.   signal a_and_b_sig : std_logic;
24.   signal a_xor_b_and_c_sig : std_logic;
25.
26. begin
27.
28.   half_adder_1 : half_adder
29.   port map(
30.     a_i      => a_i,
31.     b_i      => b_i,
32.     sum_o   => a_xor_b_sig,
33.     c_out_o => a_and_b_sig
34.   );
35.
36.   half_adder_2 : half_adder
37.   port map(
38.     a_i      => a_xor_b_sig,
39.     b_i      => c_in_i,
40.     sum_o   => sum_o,
41.     c_out_o => a_xor_b_and_c_sig
42.   );
43.   c_out_o <= a_and_b_sig or a_xor_b_and_c_sig;
44. end Behavioral;

```

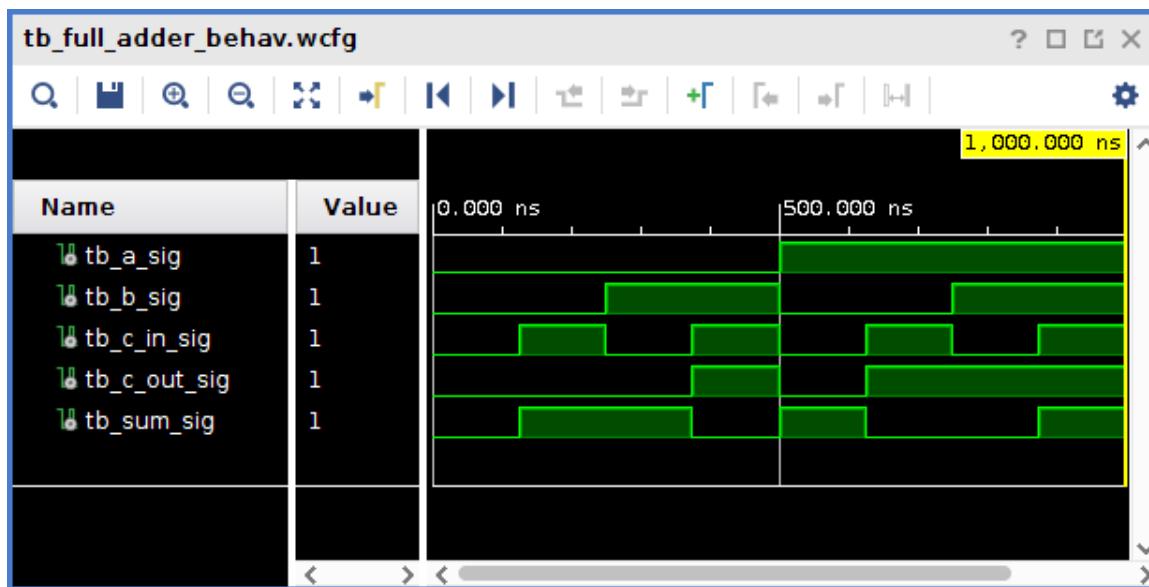
## Código de verificación

```

1. library ieee;
2. use ieee.std_logic_1164.all;
3. use ieee.numeric_std.all;
4.
5. entity tb_full_adder is
6. end tb_full_adder;
7.
8. architecture Behavioral of tb_full_adder is
9.
10. component full_adder is
11.     Port ( a_i : in std_logic;
12.             b_i : in std_logic;
13.             c_in_i : in std_logic;
14.             sum_o : out std_logic;
15.             c_out_o : out std_logic);
16. end component full_adder;
17.
18. signal tb_a_sig : std_logic;
19. signal tb_b_sig : std_logic;
20. signal tb_c_in_sig : std_logic;
21. signal tb_sum_sig : std_logic;
22. signal tb_c_out_sig : std_logic;
23.
24. begin
25.     inst_full_adder : full_adder
26.     port map(
27.         a_i      =>      tb_a_sig,
28.         b_i      =>      tb_b_sig,
29.         c_in_i  =>      tb_c_in_sig,
30.         sum_o   =>      tb_sum_sig,
31.         c_out_o =>      tb_c_out_sig
32.     );
33.
34.     process
35.     begin
36.         tb_a_sig <= '0';
37.         tb_b_sig <= '0';
38.         tb_c_in_sig <= '0';
39.         wait for 125 ns;
40.         tb_a_sig <= '0';
41.         tb_b_sig <= '0';
42.         tb_c_in_sig <= '1';
43.         wait for 125 ns;
44.         tb_a_sig <= '0';
45.         tb_b_sig <= '1';
46.         tb_c_in_sig <= '0';
47.         wait for 125 ns;
48.         tb_a_sig <= '0';
49.         tb_b_sig <= '1';
50.         tb_c_in_sig <= '1';
51.         wait for 125 ns;
52.         tb_a_sig <= '1';
53.         tb_b_sig <= '0';
54.         tb_c_in_sig <= '0';
55.         wait for 125 ns;
56.         tb_a_sig <= '1';
57.         tb_b_sig <= '0';
58.         tb_c_in_sig <= '1';
59.         wait for 125 ns;
60.         tb_a_sig <= '1';
61.         tb_b_sig <= '1';
62.         tb_c_in_sig <= '0';
63.         wait for 125 ns;
64.         tb_a_sig <= '1';
65.         tb_b_sig <= '1';
66.         tb_c_in_sig <= '1';
67.         wait for 125 ns;
68.         wait;
69.     end process;
70. end Behavioral;

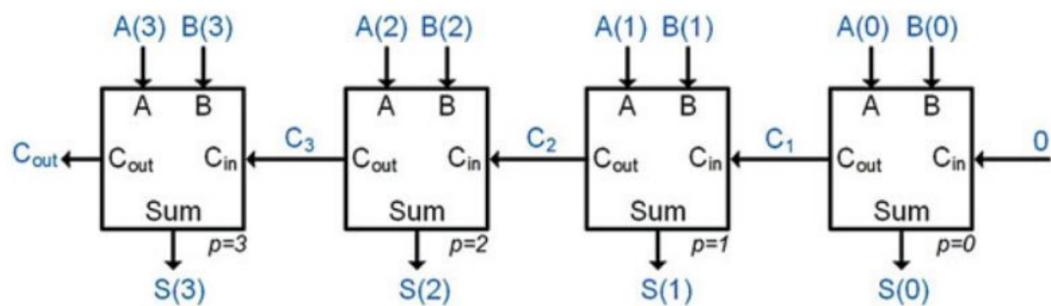
```

## Simulación



### d. Parte 4 Ripple Carry Adder 4 bits:

Se utilizan cuatro Full Adders para generar un Adder de 4 bits.



## Código RTL

```

1.  library ieee;
2.  use ieee.std_logic_1164.all;
3.  use ieee.numeric_std.all;
4.
5.  entity ripple_carry_adder is
6.    Port ( a_i : in std_logic_vector (3 downto 0);
7.           b_i : in std_logic_vector (3 downto 0);
8.           s_o : out std_logic_vector (3 downto 0);
9.           c_o : out std_logic);
10. end ripple_carry_adder;
11.
12. architecture Behavioral of ripple_carry_adder is
13.
14.   component full_adder is
15.     Port ( a_i : in std_logic;
16.             b_i : in std_logic;
17.             c_in_i : in std_logic;
18.             sum_o : out std_logic;
19.             c_out_o : out std_logic);
20.   end component full_adder;
21.
22.   signal c_1_sig : std_logic;
23.   signal c_2_sig : std_logic;
24.   signal c_3_sig : std_logic;
25.
26. begin
27.
28.   full_adder_0 : full_adder
29.     port map(
30.       a_i      =>      a_i(0),
31.       b_i      =>      b_i(0),
32.       c_in_i  =>      '0',
33.       sum_o   =>      s_o(0),
34.       c_out_o =>      c_1_sig
35.     );
36.
37.   full_adder_1 : full_adder
38.     port map(
39.       a_i      =>      a_i(1),
40.       b_i      =>      b_i(1),
41.       c_in_i  =>      c_1_sig,
42.       sum_o   =>      s_o(1),
43.       c_out_o =>      c_2_sig
44.     );
45.
46.   full_adder_2 : full_adder
47.     port map(
48.       a_i      =>      a_i(2),
49.       b_i      =>      b_i(2),
50.       c_in_i  =>      c_2_sig,
51.       sum_o   =>      s_o(2),
52.       c_out_o =>      c_3_sig
53.     );
54.
55.   full_adder_3 : full_adder
56.     port map(
57.       a_i      =>      a_i(3),
58.       b_i      =>      b_i(3),
59.       c_in_i  =>      c_3_sig,
60.       sum_o   =>      s_o(3),
61.       c_out_o =>      c_o
62.     );
63.
64. end Behavioral;

```

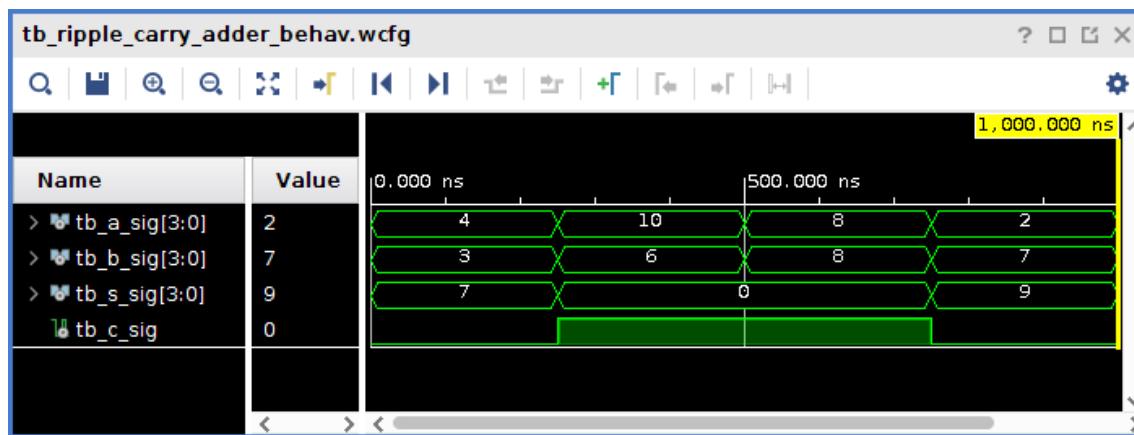
### Código de verificación

```

1. library ieee;
2. use ieee.std_logic_1164.all;
3. use ieee.numeric_std.all;
4.
5. entity tb_ripple_carry_adder is
6. end tb_ripple_carry_adder;
7.
8. architecture Behavioral of tb_ripple_carry_adder is
9.
10. component ripple_carry_adder is
11.     Port ( a_i : in std_logic_vector (3 downto 0);
12.             b_i : in std_logic_vector (3 downto 0);
13.             s_o : out std_logic_vector (3 downto 0);
14.             c_o : out std_logic);
15. end component ripple_carry_adder;
16.
17. signal tb_a_sig : std_logic_vector(3 downto 0);
18. signal tb_b_sig : std_logic_vector(3 downto 0);
19. signal tb_s_sig : std_logic_vector(3 downto 0);
20. signal tb_c_sig : std_logic;
21.
22. begin
23.
24.     inst_ripple_carry_adder : ripple_carry_adder
25.     port map
26.     (
27.         a_i      =>      tb_a_sig,
28.         b_i      =>      tb_b_sig,
29.         s_o      =>      tb_s_sig,
30.         c_o      =>      tb_c_sig
31.     );
32.
33. process
34. begin
35.     tb_a_sig  <= std_logic_vector(to_unsigned(4,4));
36.     tb_b_sig  <= std_logic_vector(to_unsigned(3,4));
37.     wait for 250 ns;
38.
39.     tb_a_sig  <= std_logic_vector(to_unsigned(10,4));
40.     tb_b_sig  <= std_logic_vector(to_unsigned(6,4));
41.     wait for 250 ns;
42.
43.     tb_a_sig  <= std_logic_vector(to_unsigned(8,4));
44.     tb_b_sig  <= std_logic_vector(to_unsigned(8,4));
45.     wait for 250 ns;
46.
47.     tb_a_sig  <= std_logic_vector(to_unsigned(2,4));
48.     tb_b_sig  <= std_logic_vector(to_unsigned(7,4));
49.     wait for 250 ns;
50.
51.     wait;
52. end process;
53.
54. end Behavioral;

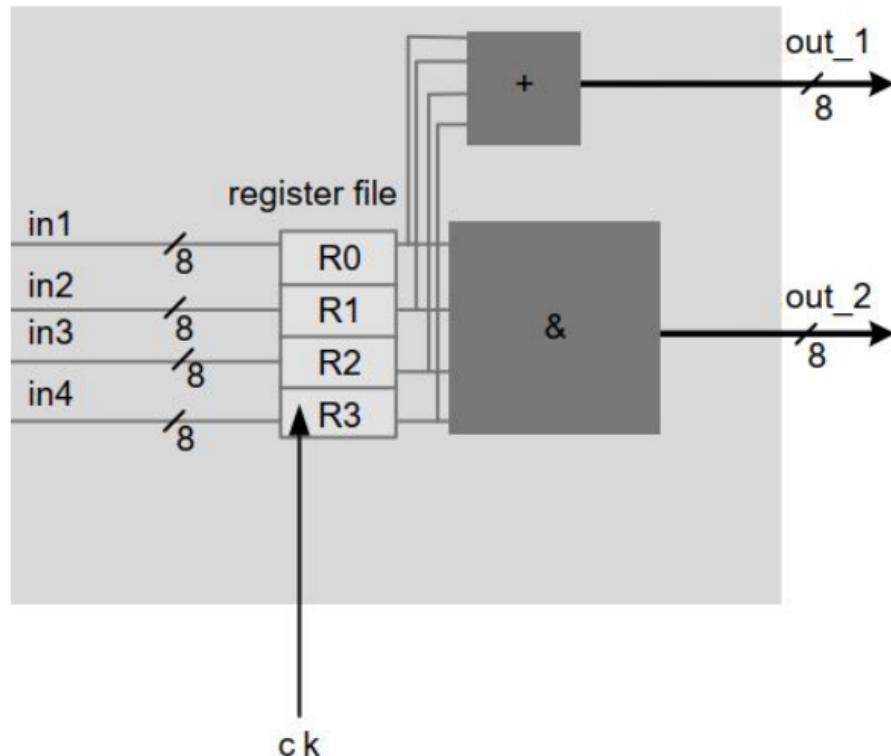
```

## Simulación



### e. Parte 5 Sistemas sincrónicos:

Se desarrolla el sistema sincrónico de la figura utilizando registros sincronizados con la señal de reloj en las entradas.



## Código RTL

```

1.  library ieee;
2.  use ieee.std_logic_1164.all;
3.  use ieee.numeric_std.all;
4.
5.      entity sinc_system is
6.          Port (
7.              clk_i : in std_logic;
8.              rst_i : in std_logic;
9.              in1_i : in std_logic_vector (7 downto 0);
10.             in2_i : in std_logic_vector (7 downto 0);
11.             in3_i : in std_logic_vector (7 downto 0);
12.             in4_i : in std_logic_vector (7 downto 0);
13.             out1_o : out std_logic_vector (7 downto 0);
14.             out2_o : out std_logic_vector (7 downto 0)
15.         );
16.     end sinc_system;
17.
18. architecture Behavioral of sinc_system is
19.
20.     signal out1_sig : std_logic_vector(7 downto 0);
21.     signal out2_sig : std_logic_vector(7 downto 0);
22.
23. begin
24.
25.     reg_file : process (rst_i, clk_i)
26.     begin
27.         if (rst_i = '0') then
28.             out1_sig <= (others => '0');
29.             out2_sig <= (others => '0');
30.         elsif (rising_edge(clk_i)) then
31.
32.             out1_sig <= std_logic_vector(unsigned(in1_i)
33.                                         + unsigned(in2_i)
34.                                         + unsigned(in3_i)
35.                                         + unsigned(in4_i));
36.
37.             out2_sig <= in1_i and in2_i and in3_i and in4_i;
38.
39.         end if;
40.     end process reg_file;
41.
42.     out1_o <= out1_sig;
43.     out2_o <= out2_sig;
44.
45. end Behavioral;

```

## Código de verificación

```

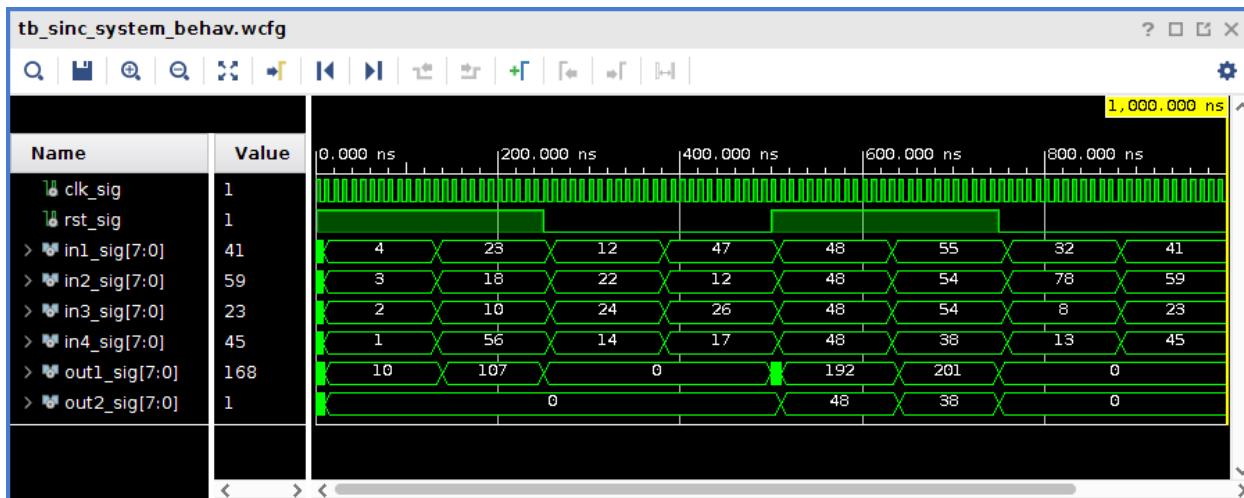
1. library ieee;
2. use ieee.std_logic_1164.all;
3. use ieee.numeric_std.all;
4.
5. entity tb_sinc_system is
6. end tb_sinc_system;
7.
8. architecture Behavioral of tb_sinc_system is
9.
10. component sinc_system is
11. Port (
12.     clk_i : in std_logic;
13.     rst_i : in std_logic;
14.     in1_i : in std_logic_vector (7 downto 0);
15.     in2_i : in std_logic_vector (7 downto 0);
16.     in3_i : in std_logic_vector (7 downto 0);
17.     in4_i : in std_logic_vector (7 downto 0);
18.     out1_o : out std_logic_vector (7 downto 0);
19.     out2_o : out std_logic_vector (7 downto 0)
20. );
21. end component sinc_system;
22.
23. signal clk_sig : std_logic;
24. signal rst_sig : std_logic;
25. signal in1_sig : std_logic_vector (7 downto 0);
26. signal in2_sig : std_logic_vector (7 downto 0);
27. signal in3_sig : std_logic_vector (7 downto 0);
28. signal in4_sig : std_logic_vector (7 downto 0);
29. signal out1_sig : std_logic_vector (7 downto 0);
30. signal out2_sig : std_logic_vector (7 downto 0);
31.
32. begin
33.
34.     inst_sinc_system : sinc_system
35.     port map(
36.         clk_i      => clk_sig,
37.         rst_i      => rst_sig,
38.         in1_i      => in1_sig,
39.         in2_i      => in2_sig,
40.         in3_i      => in3_sig,
41.         in4_i      => in4_sig,
42.         out1_o     => out1_sig,
43.         out2_o     => out2_sig
44.     );
45.
46. -----
47. -- Clocks y Reset
48. -----
49. clk_gen : process
50. begin
51.     clk_sig <= '1';
52.     wait for 5 ns;
53.     clk_sig <= '0';
54.     wait for 5 ns;
55. end process clk_gen;
56.
57. rst_gen : process
58. begin
59.     rst_sig <= '1';
60.     wait for 250 ns;
61.     rst_sig <= '0';
62.     wait for 250 ns;
63. end process rst_gen;
64.
```

```

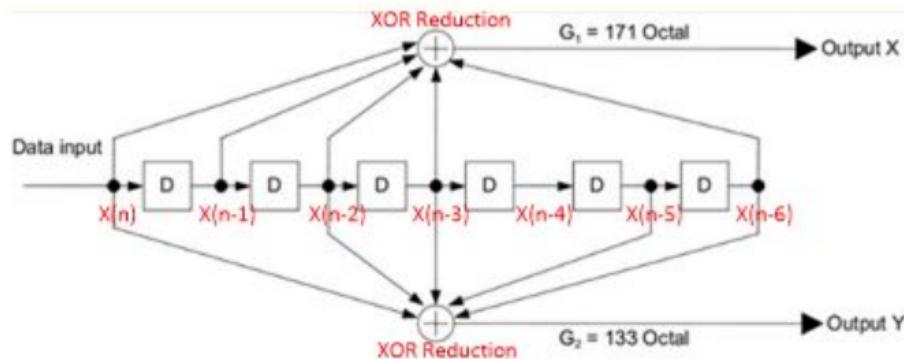
65. -----
66. -- Proceso de estímulos a las señales
67. -----
68. process
69. begin
70.     in1_sig    <= std_logic_vector(to_unsigned(32,8));
71.     in2_sig    <= std_logic_vector(to_unsigned(78,8));
72.     in3_sig    <= std_logic_vector(to_unsigned(8,8));
73.     in4_sig    <= std_logic_vector(to_unsigned(13,8));
74.     wait for 2 ns;
75.     in1_sig    <= std_logic_vector(to_unsigned(41,8));
76.     in2_sig    <= std_logic_vector(to_unsigned(59,8));
77.     in3_sig    <= std_logic_vector(to_unsigned(23,8));
78.     in4_sig    <= std_logic_vector(to_unsigned(45,8));
79.     wait for 2 ns;
80.     in1_sig    <= std_logic_vector(to_unsigned(12,8));
81.     in2_sig    <= std_logic_vector(to_unsigned(22,8));
82.     in3_sig    <= std_logic_vector(to_unsigned(24,8));
83.     in4_sig    <= std_logic_vector(to_unsigned(14,8));
84.     wait for 2 ns;
85.     in1_sig    <= std_logic_vector(to_unsigned(47,8));
86.     in2_sig    <= std_logic_vector(to_unsigned(12,8));
87.     in3_sig    <= std_logic_vector(to_unsigned(26,8));
88.     in4_sig    <= std_logic_vector(to_unsigned(17,8));
89.     wait for 2 ns;
90.     in1_sig    <= std_logic_vector(to_unsigned(4,8));
91.     in2_sig    <= std_logic_vector(to_unsigned(3,8));
92.     in3_sig    <= std_logic_vector(to_unsigned(2,8));
93.     in4_sig    <= std_logic_vector(to_unsigned(1,8));
94.     wait for 125 ns;
95.     in1_sig    <= std_logic_vector(to_unsigned(23,8));
96.     in2_sig    <= std_logic_vector(to_unsigned(18,8));
97.     in3_sig    <= std_logic_vector(to_unsigned(10,8));
98.     in4_sig    <= std_logic_vector(to_unsigned(56,8));
99.     wait for 125 ns;
100.    in1_sig   <= std_logic_vector(to_unsigned(12,8));
101.    in2_sig   <= std_logic_vector(to_unsigned(22,8));
102.    in3_sig   <= std_logic_vector(to_unsigned(24,8));
103.    in4_sig   <= std_logic_vector(to_unsigned(14,8));
104.    wait for 125 ns;
105.    in1_sig   <= std_logic_vector(to_unsigned(47,8));
106.    in2_sig   <= std_logic_vector(to_unsigned(12,8));
107.    in3_sig   <= std_logic_vector(to_unsigned(26,8));
108.    in4_sig   <= std_logic_vector(to_unsigned(17,8));
109.    wait for 125 ns;
110.    in1_sig   <= std_logic_vector(to_unsigned(48,8));
111.    in2_sig   <= std_logic_vector(to_unsigned(48,8));
112.    in3_sig   <= std_logic_vector(to_unsigned(48,8));
113.    in4_sig   <= std_logic_vector(to_unsigned(48,8));
114.    wait for 125 ns;
115.    in1_sig   <= std_logic_vector(to_unsigned(55,8));
116.    in2_sig   <= std_logic_vector(to_unsigned(54,8));
117.    in3_sig   <= std_logic_vector(to_unsigned(54,8));
118.    in4_sig   <= std_logic_vector(to_unsigned(38,8));
119.    wait for 125 ns;
120.    in1_sig   <= std_logic_vector(to_unsigned(32,8));
121.    in2_sig   <= std_logic_vector(to_unsigned(78,8));
122.    in3_sig   <= std_logic_vector(to_unsigned(8,8));
123.    in4_sig   <= std_logic_vector(to_unsigned(13,8));
124.    wait for 125 ns;
125.    in1_sig   <= std_logic_vector(to_unsigned(41,8));
126.    in2_sig   <= std_logic_vector(to_unsigned(59,8));
127.    in3_sig   <= std_logic_vector(to_unsigned(23,8));
128.    in4_sig   <= std_logic_vector(to_unsigned(45,8));
129.    wait for 125 ns;
130. end process;
131. end Behavioral;

```

## Simulación



## 2. Ejercicio 2 “Encoder Convolucional”



Se implementa un encoder convolucional con los siguientes parámetros:

- P1 = 171 OCTAL (Ver el gráfico de la figura). 0b0111\_1001 en binario.
- P2 = 133 OCTAL (Ver el gráfico de la figura). 0b0101\_1011 en binario.
- K = 7, orden del shift register.
- R = 1/2. Input = 1 bits Output = 2 bits. O por cada bit de entrada salen 2 bits a la salida.
- Entrada AXI Stream ancho de word 16 bits.
- Salida AXI Stream ancho de word 32 bits.
- Reset asincrónico activo bajo.

## Código RTL

```

1. library ieee;
2. use ieee.std_logic_1164.all;
3. use ieee.numeric_std.all;
4.
5. entity conv_encoder is
6.   generic (
7.     POL_1           : integer := 8#171#;
8.     POL_2           : integer := 8#133#;
9.     N               : integer := 16;
10.    M               : integer := 32
11. );
12.  Port (
13.    clk_i           : in std_logic;
14.    rst_i           : in std_logic;
15.    s_axis_tdata_i  : in std_logic_vector (N-1 downto 0);
16.    s_axis_tvalid_i : in std_logic;
17.    s_axis_tready_o : out std_logic;
18.    m_axis_tdata_o  : out std_logic_vector (M-1 downto 0);
19.    m_axis_tvalid_o : out std_logic;
20.    m_axis_tready_i : in std_logic);
21. end conv_encoder;
22.
23. architecture Behavioral of conv_encoder is
24.
25.   --Tamaño del shift register.
26.   constant K : integer := 7;
27.
28.   --Polinomios Generadores.
29.   constant POL1_SLV : std_logic_vector(K-1 downto 0) := std_logic_vector(to_unsigned(POL_1,K));
30.   constant POL2_SLV : std_logic_vector(K-1 downto 0) := std_logic_vector(to_unsigned(POL_2,K));
31.
32.   --Ciclos del proceso del encoder
33.   constant CYCLES_PROCESS : integer := N + 1;
34.
35.   --Definición de los estados de la maquina de estados del proceso del encoder.
36.   type state_type is
37.   (
38.     ST_WAIT_DATA,
39.     ST_PROCESS_DATA,
40.     ST_TRANSMIT_DATA
41. );
42.
43.   --Generamos el arreglo de registro
44.   type reg_axi_type is record
45.     state      : state_type;
46.     --Registro de corrimiento de entrada de 16 bits
47.     data_in_reg : std_logic_vector(N-1 downto 0);
48.     --Registro shift register para la convolución
49.     shift_reg : std_logic_vector(K-1 downto 0);
50.     --contador del proceso
51.     process_count : integer range 0 to CYCLES_PROCESS;
52.     --Señal de Ready del puerto de salida del AXI Stream Slave
53.     s_axis_tready : std_logic;
54.     --Señales para el manejo del puerto AXIS4-Stream
55.     m_axis_tdata : std_logic_vector(m_axis_tdata_o'RANGE);
56.     m_axis_tvalid : std_logic;
57.   end record;
58.
59.   --Valor por defecto de los registros.
60.   constant reg_ctr_default : reg_axi_type := (
61.     state      => ST_WAIT_DATA,
62.     data_in_reg => (others => '0'),
63.     shift_reg  => (others => '0'),
64.     process_count => 0,
65.     s_axis_tready => '0',
66.     m_axis_tdata  => (others => '0'),
67.     m_axis_tvalid  => '0'
68. );
69.
70.   --Registros internos de variables de control
71.   signal axi_crt : reg_axi_type := reg_ctr_default;
72.
73. begin
74.
75.   --Maquina de estado Moore, que maneja los cambios de los estados del encoder de convolución
76.   --Proceso que actualiza el estado actual de la maquina de estados y
77.   --los registros de cada flanco de subida del la señal clk
78.   --Realiza la lectura de la entrada y almacena la información en el shift register

```

```

79.      --Actualiza el contador en el estado de proceso
80.      --Realiza el manejo de la señal ready del AXI Stream slave.
81.      --Realiza el manejo de la señal valid del AXI Stream Master.
82.      --Proceso que actualiza las señales que estan conectadas a las salidas de la maquina de estados
83.      fsm_process : process (clk_i, rst_i)
84.          variable result : std_logic_vector(1 downto 0) := (others => '0');
85.          variable tdata  : std_logic_vector(m_axis_tdata_o'RANGE) := (others => '0');
86.      begin
87.          if(rst_i = '0') then
88.              axi_crt <= reg_ctr_default;
89.              result := (others => '0');
90.              tdata := (others => '0');
91.          elsif rising_edge(clk_i) then
92.              case(axi_crt.state) is
93.                  when ST_WAIT_DATA =>
94.                      axi_crt.s_axis_tready <= '1';
95.                      axi_crt.process_count <= 0;
96.                      axi_crt.m_axis_tvalid <= '0';
97.
98.                  if (s_axis_tvalid_i = '1' and s_axis_tready_o = '1') then
99.                      axi_crt.state <= ST_PROCESS_DATA;
100.                     axi_crt.s_axis_tready <= '0';
101.                     axi_crt.data_in_reg <= s_axis_tdata_i;
102.                     tdata := (others => '0');
103.                 end if;
104.             when ST_PROCESS_DATA =>
105.                 axi_crt.s_axis_tready <= '0';
106.
107.                 if(axi_crt.process_count >= 0 and axi_crt.process_count <= 15) then
108.                     axi_crt.shift_reg <= axi_crt.data_in_reg(axi_crt.process_count)
109.                     & axi_crt.shift_reg(axi_crt.shift_reg'LEFT downto 1);
110.                 end if;
111.
112.                 result(0) := xor(axi_crt.shift_reg and POL1_SLV);
113.                 result(1) := xor(axi_crt.shift_reg and POL2_SLV);
114.
115.                 tdata := result & tdata(tdata'LEFT downto 2);
116.
117.                 if (axi_crt.process_count >= CYCLES_PROCESS - 1) then
118.                     axi_crt.process_count <= 0;
119.                     axi_crt.m_axis_tvalid <= '1';
120.                     axi_crt.state <= ST_TRANSMIT_DATA;
121.
122.                 else
123.                     axi_crt.process_count <= axi_crt.process_count + 1;
124.                 end if;
125.             when ST_TRANSMIT_DATA =>
126.                 axi_crt.s_axis_tready <= '1';
127.                 axi_crt.m_axis_tvalid <= '0';
128.                 axi_crt.process_count <= 0;
129.
130.                 if (m_axis_tready_i = '1' and m_axis_tvalid_o = '1') then
131.                     axi_crt.state <= ST_WAIT_DATA;
132.                 end if;
133.             when others => null;
134.         end case;
135.     end if;
136.     axi_crt.m_axis_tdata <= tdata;
137.
138. end process;
139.
140. m_axis_tdata_o  <= axi_crt.m_axis_tdata;
141. m_axis_tvalid_o <= axi_crt.m_axis_tvalid;
142. s_axis_tready_o <= axi_crt.s_axis_tready;
143.
144. end Behavioral;

```

## Código de verificación

```

1. library ieee;
2. use ieee.std_logic_1164.all;
3. use ieee.numeric_std.all;
4.
5. entity tb_conv_encoder is
6. end tb_conv_encoder;
7.
8. architecture Behavioral of tb_conv_encoder is
9.
10. constant CLK_PERIOD : time := 10 ns;
11. signal clk_i : std_logic := '1';
12. signal rst_i : std_logic := '0';
13.
14. -- Testbench DUT generics
15. constant POL_1 : integer := 8#171#;
16. constant POL_2 : integer := 8#133#;
17.
18. -- Testbench DUT ports
19. signal s_axis_tdata_w : std_logic_vector(15 downto 0) := (others => '0');
20. signal s_axis_tvalid_w : std_logic := '0';
21. signal s_axis_tready_w : std_logic;
22. signal m_axis_tdata_w : std_logic_vector(31 downto 0);
23. signal m_axis_tvalid_w : std_logic;
24. signal m_axis_tready_w : std_logic;
25.
26. component conv_encoder is
27. generic (
28.     POL_1           : integer := 8#171#;
29.     POL_2           : integer := 8#133#;
30.     N               : integer := 16;
31.     M               : integer := 32
32. );
33. Port (
34.     clk_i           : in std_logic;
35.     rst_i           : in std_logic;
36.     s_axis_tdata_i : in std_logic_vector (N-1 downto 0);
37.     s_axis_tvalid_i : in std_logic;
38.     s_axis_tready_o : out std_logic;
39.     m_axis_tdata_o : out std_logic_vector (M-1 downto 0);
40.     m_axis_tvalid_o : out std_logic;
41.     m_axis_tready_i : in std_logic);
42. end component conv_encoder;
43.
44. constant S_AXIS_VALID_PERIOD : time := 250 ns;
45. constant S_AXIS_DUTY_CYCLE : real := 0.1;
46.
47. --Procedimiento para generar retardos de M ciclos de reloj.
48. procedure generate_delay (
49.     signal    clk_i   : in std_logic;
50.     constant  M_CYCLE : in integer
51. ) is
52. begin
53.     wait_ncycle : for i in 0 to M_CYCLE-1 loop
54.         wait until rising_edge(clk_i);
55.     end loop;
56. end procedure generate_delay;
57.
58. begin
59.

```

```

60.      clk_i <= not clk_i after CLK_PERIOD/2;
61.      rst_i <= '1' after 20 ns;
62.
63.      generate_stimulus : process is
64.          variable i : integer;
65.          variable tdata : integer := 32769;
66.      begin
67.          --Valores de reset del bus de las señales
68.          m_axis_tready_w <= '1';
69.          s_axis_tdata_w <= (others => '0');
70.          s_axis_tvalid_w <= '0';
71.          wait until rst_i = '1';
72.          generate_delay(clk_i,10);
73.
74.          loop_generate_data : for i in 0 to 4 loop
75.
76.              s_axis_tvalid_w <= '1';
77.              s_axis_tdata_w <= std_logic_vector(to_unsigned(tdata,
78.                                              s_axis_tdata_w'LENGTH));
79.              generate_delay(clk_i,1);
80.
81.              s_axis_tdata_w <= (others => '0');
82.              s_axis_tvalid_w <= '0';
83.
84.              generate_delay(clk_i,40);
85.              tdata := tdata + 1;
86.          end loop;
87.
88.          wait;
89.      end process; -- generate_stimulus
90.
91.
92.      int_conv_encoder : conv_encoder
93.          generic map (
94.              POL_1 => POL_1,
95.              POL_2 => POL_2
96.          )
97.          port map (
98.              clk_i          => clk_i,
99.              rst_i          => rst_i,
100.              s_axis_tdata_i => s_axis_tdata_w,
101.              s_axis_tvalid_i => s_axis_tvalid_w,
102.              s_axis_tready_o => s_axis_tready_w,
103.              m_axis_tdata_o => m_axis_tdata_w,
104.              m_axis_tvalid_o => m_axis_tvalid_w,
105.              m_axis_tready_i => m_axis_tready_w
106.          );
107.
108.      end Behavioral;

```

Código de generación de vectores de prueba con Python.

```
import numpy as np
import sk_dsp_comm.fec_conv as fec

def bitfield(n):
    return [int(digit) for digit in '{0:16b}'.format(n)]

cc1 = fec.fec_conv(['1111001','1011011'])

state = '00000000'
result = np.array([],dtype=np.uint8)
int_array = np.array([0x8001,0x8002,0x8003,0x8004],dtype=np.uint64)
print("Para las Entradas:")

for item in int_array:
    print('0x' + hex(item))
for item in int_array:
    bit_array = bitfield(item)
    bit_array = bit_array[::-1]
    y,state = cc1.conv_encoder(bit_array,state)
    y = np.packbits(y[::-1].astype(np.int64),bitorder = 'big')
    result = np.concatenate([result,y])

result = result.reshape(result.size//4,4).astype(dtype = np.int64)

print("\nSe obtiene las Salidas:")
for item in result:
    print('0x' + hex(int.from_bytes(list(item),byteorder = 'big'))[2:].zfill(8))
```

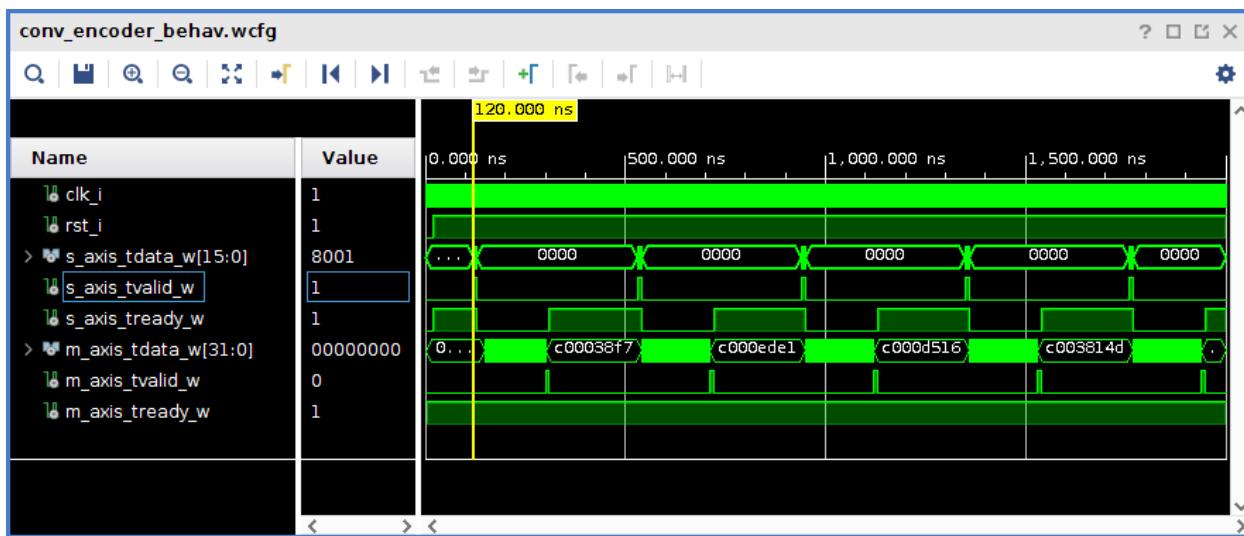
Para las Entradas:

0x0x8001  
0x0x8002  
0x0x8003  
0x0x8004

Se obtiene las Salidas:

0xc00038f7  
0xc000edel  
0xc000d516  
0xc003814d

## Simulación



## Conclusión

Se puede apreciar que los resultados de la simulación del encoder y los valores de los vectores de prueba generados en Python, coinciden perfectamente teniendo como valores de entrada en hexadecimal de 16 bits de: 0x8001, 0x8002, 0x8003 y 0x8004. Y los valores de salida de 32 bits de: 0xC00038F7, 0xC000EDE1, 0xC000D516 y 0xC003814D.

### 3. Ejercicio 3 “AXI Stream y Ecuaciones de Diferencias”

#### Análisis del sistema lineal

Se tiene la siguiente ecuación de diferencia:

$$y(n) = x(n) - x(n - 1) + x(n - 2) + x(n - 3) + 0.5y(n - 1) + 0.25y(n - 2)$$

Se realiza la transformada Z a ambos lados de la igualdad.

$$Y(z) = X(z) - z^{-1} \cdot X(z) + z^{-2} \cdot X(z) + z^{-3} \cdot X(z) + 0.5z^{-1} \cdot Y(z) + 0.25z^{-2} \cdot Y(z)$$

$$Y(z) \cdot (1 - 0.5z^{-1} - 0.25z^{-2}) = X(z) \cdot (1 - z^{-1} + z^{-2} + z^{-3})$$

$$\frac{Y(z)}{X(z)} = H(z) = \frac{1 - z^{-1} + z^{-2} + z^{-3}}{1 - 0.5z^{-1} - 0.25z^{-2}} = \frac{N(z)}{D(z)}$$

Se calcula las singularidades de los polinomios N(z) y D(z).

$$b = [1, -1, 1, 1]$$

$$a = [1, -0.5, -0.25]$$

```
import numpy as np
#Se describe los polinomios.
b = [1, -1, 1, 1]
a = [1, -0.5, -0.25]

#Polos y ceros
p = np.roots(a)
z = np.roots(b)

print("El sistema tiene un número de polos igual a: {}".format(p.size))
print("Primer polo está ubicado en z = {}".format(p[0]))
print("Segundo polo está ubicado en z = {}".format(p[1]))

print("El sistema tiene un número de zeros igual a: {}".format(z.size))
print("Primer zero está ubicado en z = {}".format(z[0]))
print("Segundo polo está ubicado en z = {}".format(z[1]))
print("Tercer esta ubicado en z = {}".format(z[2]))
```

El sistema tiene un número de polos igual a: 2

Primer polo está ubicado en z = 0.8090169943749475

Segundo polo está ubicado en z = -0.3090169943749474

El sistema tiene un número de zeros igual a: 3

Primer zero está ubicado en z = (0.771844506346038+1.1151425080399364j)

Segundo polo está ubicado en z = (0.771844506346038-1.1151425080399364j)

Tercer esta ubicado en z = (-0.5436890126920764+0j)

El sistema se puede denominar como IIR debido a que posee dos polos y la respuesta del sistema al impulso es infinita.

A continuación se grafica el diagrama de polos y ceros.

```
#Declaracion de importaciones y de la funcion para dibujar el zplane
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import patches
from matplotlib.figure import Figure
from matplotlib import rcParams

#No tenemos zplane declara da con lo cual directamente
#ubicamos esta implementacion en stackoverflow.
def zplane(b,a,filename=None):
    """Plot the complex z-plane given a transfer function.
    """
    # get a figure/plot
    ax = plt.subplot(111)

    # create the unit circle
    uc = patches.Circle((0,0), radius=1, fill=False,
                         color='black', ls='dashed')
    ax.add_patch(uc)

    # The coefficients are less than 1, normalize the coefficients
    if np.max(b) > 1:
        kn = np.max(b)
        b = b/float(kn)
    else:
        kn = 1

    if np.max(a) > 1:
        kd = np.max(a)
        a = a/float(kd)
    else:
        kd = 1

    # Get the poles and zeros
    p = np.roots(a)
    z = np.roots(b)
    k = kn/float(kd)

    # Plot the zeros and set marker properties
    t1 = plt.plot(z.real, z.imag, 'go', ms=10)
    plt.setp(t1, markersize=10.0, markeredgewidth=1.0,
              markeredgecolor='k', markerfacecolor='g')

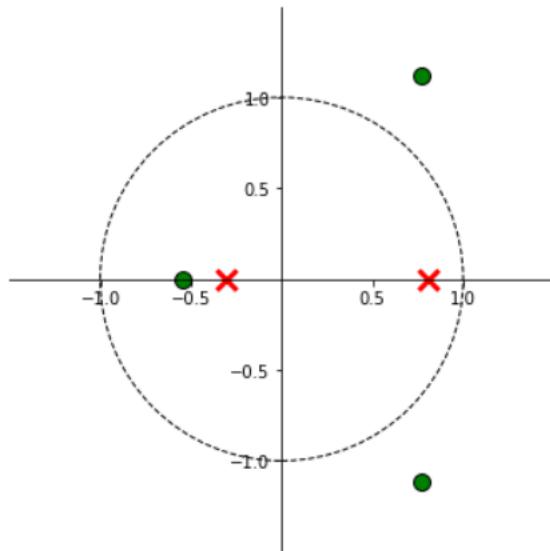
    # Plot the poles and set marker properties
    t2 = plt.plot(p.real, p.imag, 'rx', ms=10)
    plt.setp(t2, markersize=12.0, markeredgewidth=3.0,
              markeredgecolor='r', markerfacecolor='r')

    ax.spines['left'].set_position('center')
    ax.spines['bottom'].set_position('center')
    ax.spines['right'].set_visible(False)
    ax.spines['top'].set_visible(False)

    # set the ticks
    r = 1.5; plt.axis('scaled'); plt.axis([-r, r, -r, r])
    ticks = [-1, -.5, .5, 1]; plt.xticks(ticks); plt.yticks(ticks)

    if filename is None:
        plt.show()
    else:
        plt.savefig(filename)
    return z, p, k
```

```
#Teniendo como parametro a b y a que son los polinomios
fig = plt.figure(figsize=(10,6))
z,p,k=zplane(b,a)
```



Se puede apreciar que el sistema tiene todos los polos sobre el eje real dentro del círculo unitario. De lo anterior se puede concluir que el sistema es estable.

**A continuación se grafica la respuesta en frecuencia del sistema en magnitud y en fase.**

```
from scipy import signal
#Sacamos la respuesta en frecuencia

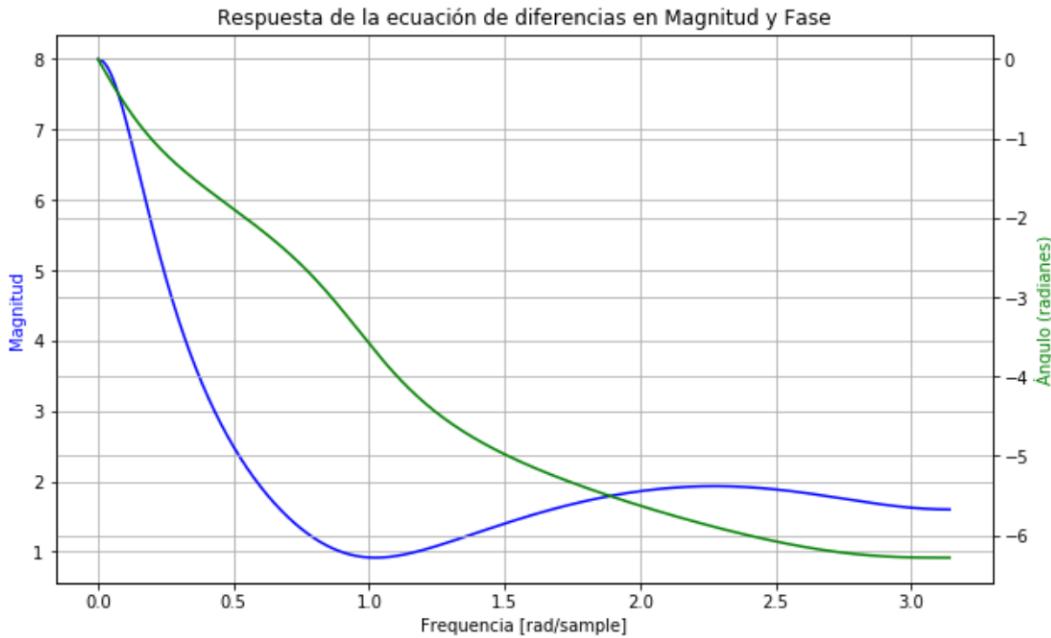
#Calculo de la respuesta en frecuencia
w , h = signal.freqz(b,a)
#Como esta respuesta en teoría es continua, podemos obtener infinitos puntos.
print("Cantidad de puntos calculados:{}".format(np.size(w)))

#Calculo de la respuesta en frecuencia
w , h = signal.freqz(b,a,worN=8192)
#Como esta respuesta en teoría es continua, podemos obtener infinitos puntos.
print("Cantidad de puntos calculados:{}".format(np.size(w)))

#Dibujamos la respuesta en Frecuencia y la Fase en el mismo grafico
fig,ax1 = plt.subplots(figsize=(10,6))
ax1.set_title('Respuesta de la ecuación de diferencias en Magnitud y Fase')
ax1.plot(w, (abs(h)), 'b')
ax1.set_ylabel('Magnitud', color='b')
ax1.set_xlabel('Frecuencia [rad/sample]')

ax2 = ax1.twinx()
angles = np.unwrap(np.angle(h))
ax2.plot(w, angles, 'g')
ax2.set_ylabel('Ángulo (radianes)', color='g')
ax1.grid()
ax2.grid()
ax1.axis('tight')
```

Cantidad de puntos calculados:512  
Cantidad de puntos calculados:8192



## Código RTL

```

1. library ieee ;
2. use ieee.std_logic_1164.all ;
3. use ieee.numeric_std.all ;
4.
5. entity diff_ecuation is
6.   generic (
7.     --Bits de la palabra de entrada
8.     N : integer := 16;
9.     --Bits de la palabra de salida
10.    M : integer := 18);
11.  Port (
12.    clk_i : in STD_LOGIC;
13.    rst_i : in STD_LOGIC;
14.    s_axis_tdata_i : in STD_LOGIC_VECTOR (N-1 downto 0);
15.    s_axis_tvalid_i : in STD_LOGIC;
16.    s_axis_tready_o : out STD_LOGIC;
17.    m_axis_tdata_o : out STD_LOGIC_VECTOR (M-1 downto 0);
18.    m_axis_tvalid_o : out STD_LOGIC;
19.    m_axis_tready_i : in STD_LOGIC);
20. end diff_ecuation;
21.
22. architecture Behavioral of diff_ecuation is
23.
24.   constant FACT_X : integer := 3;
25.   constant FACT_Y : integer := 2;
26.
27.   --Vector con los datos que se van a ir almacenando
28.   type vector_type_x is array (integer range <>) of std_logic_vector(s_axis_tdata_i'RANGE);
29.   signal x_vector_reg : vector_type_x(0 to FACT_X);
30.
31.   type vector_type_y is array (integer range <>) of std_logic_vector(16-1 downto 0);
32.   signal y_vector_reg : vector_type_y(0 to FACT_Y);
33.
34.   component ff_module is
35.     generic (
36.       N : integer := N
37.     );
38.     Port ( clk_i      : in STD_LOGIC;

```

```

39.          rst_i      : in  STD_LOGIC;
40.          enable_t_i : in  STD_LOGIC;
41.          data_i     : in  STD_LOGIC_VECTOR (N-1 downto 0);
42.          data_o     : out STD_LOGIC_VECTOR (N-1 downto 0));
43.      end component ff_module;
44.
45.
46.      --Definición de los estados de la maquina de estados del proceso del encoder.
47.      type state_type is
48.      (
49.          ST_WAIT_DATA,
50.          ST_PROCESS_DATA,
51.          ST_TRANSMIT_DATA
52.      );
53.
54.      --Generamos el arreglo de registro
55.      type reg_axi_type is record
56.          state           : state_type;
57.          --Señal de Ready del puerto de salida del AXI Stream Slave
58.          s_axis_tready   : std_logic;
59.          --Señales para el manejo del puerto AXIS4-Stream
60.          m_axis_tdata    : std_logic_vector(m_axis_tdata_o'RANGE);
61.          m_axis_tvalid   : std_logic;
62.          enable_process  : std_logic;
63.      end record;
64.
65.      --Valor por defecto de los registros.
66.      constant reg_ctr_default : reg_axi_type := (
67.          state        => ST_WAIT_DATA,
68.          s_axis_tready => '0',
69.          m_axis_tdata  => (others => '0'),
70.          m_axis_tvalid => '0',
71.          enable_process => '0'
72.      );
73.
74.      --Registros internos de variables de control
75.      signal axi_crt : reg_axi_type := reg_ctr_default;
76.
77.      constant SIZE_Q4_7 : integer := 4+7;
78.      constant SIZE_32 : integer := 32;
79.
80.      signal result_x : signed(SIZE_Q4_7-1 downto 0);
81.      signal result_y : signed(SIZE_32-1 downto 0);
82.      signal result_t : signed(SIZE_Q4_7-1 downto 0);
83.
84.      --Se escribe lo factores de multiplicacion en Q2.7
85.      constant MUL_FACTOR_REAL_0_5 : real := (0.5)*real((2**7));
86.      constant MUL_FACTOR_0_5       : integer := integer(MUL_FACTOR_REAL_0_5);
87.
88.      constant MUL_FACTOR_REAL_0_25 : real := (0.25)*real((2**7));
89.      constant MUL_FACTOR_0_25     : integer := integer(MUL_FACTOR_REAL_0_25);
90.
91.      -----
92.      --Comienzo del comportamiento del la entidad diff_ecuation
93.      -----
94.      begin
95.
96.          x_vector_reg(0) <= s_axis_tdata_i;
97.          gen_array_ff_x : for i in 0 to FACT_X - 1 generate
98.              reg_inst : ff_module
99.                  generic map (
100.                      N => N
101.                  )
102.                  port map (
103.                      clk_i      => clk_i,
104.                      rst_i      => rst_i,
105.                      enable_t_i => axi_crt.enable_process,
106.                      data_i     => x_vector_reg(i),
107.                      data_o     => x_vector_reg(i + 1)
108.                  );
109.          end generate;
110.
111.          y_vector_reg(0) <= std_logic_vector(resize(result_t,16));
112.          gen_array_ff_y : for i in 0 to FACT_Y - 1 generate
113.              reg_inst : ff_module
114.                  generic map (

```

```

115.          N => 16
116.      )
117.      port map (
118.          clk_i      => clk_i,
119.          rst_i      => rst_i,
120.          enable_t_i  => axi_crt.enable_process,
121.          data_i      => y_vector_reg(i),
122.          data_o      => y_vector_reg(i + 1)
123.      );
124. end generate;
125.
126. --Suma de los factores de x => x(n) - x(n-1) + x(n-2) + x(n+3)
127. result_x <= to_signed(to_integer(signed(x_vector_reg(0))))
128.           - to_integer(signed(x_vector_reg(1)))
129.           + to_integer(signed(x_vector_reg(2)))
130.           + to_integer(signed(x_vector_reg(3))),SIZE_Q4_7);
131.
132. --Suma de los factores de y => 0.5 * y(n-1) + 0.25 * y(n-2)
133. result_y <= to_signed(MUL_FACTOR_0_5 * to_integer(signed(y_vector_reg(1))), SIZE_32)
134.           + to_signed(MUL_FACTOR_0_25 * to_integer(signed(y_vector_reg(2))), SIZE_32);
135.
136. --Suma de los factores de x & y,
137. --acomodando el formato de destino que es Q4.7 en un vector de 16 bits
138. result_t <= result_x + result_y(17 downto 7);
139.
140. -----
141. --Manejo de los AXI4 Stream (Slave y Master)
142. -----
143.
144. fsm_process : process (clk_i, rst_i)
145. begin
146.     if(rst_i = '0') then
147.         axi_crt <= reg_ctr_default;
148.     elsif rising_edge(clk_i) then
149.         case(axi_crt.state) is
150.             when ST_WAIT_DATA =>
151.                 axi_crt.s_axis_tready <= '1';
152.                 axi_crt.m_axis_tvalid <= '0';
153.                 axi_crt.enable_process <= '0';
154.                 if (s_axis_tvalid_i = '1' and s_axis_tready_o = '1') then
155.                     axi_crt.state <= ST_PROCESS_DATA;
156.                     axi_crt.s_axis_tready <= '0';
157.                     axi_crt.enable_process <= '1';
158.                 end if;
159.             when ST_PROCESS_DATA =>
160.                 axi_crt.s_axis_tready <= '0';
161.                 axi_crt.m_axis_tvalid <= '1';
162.                 axi_crt.enable_process <= '0';
163.                 axi_crt.m_axis_tdata <= std_logic_vector(resize(result_t,16));
164.                 axi_crt.state <= ST_TRANSMIT_DATA;
165.             when ST_TRANSMIT_DATA =>
166.                 axi_crt.s_axis_tready <= '1';
167.                 axi_crt.m_axis_tvalid <= '0';
168.                 axi_crt.enable_process <= '0';
169.                 if (m_axis_tready_i = '1' and m_axis_tvalid_o = '1') then
170.                     axi_crt.state <= ST_WAIT_DATA;
171.                 end if;
172.             when others => null;
173.         end case;
174.     end if;
175. end process;
176.
177. m_axis_tdata_o  <= axi_crt.m_axis_tdata;
178. m_axis_tvalid_o <= axi_crt.m_axis_tvalid;
179. s_axis_tready_o <= axi_crt.s_axis_tready;
180.
181. end Behavioral;

```

### Código de verificación

```

1. library ieee ;
2. use ieee.std_logic_1164.all ;
3. use ieee.numeric_std.all ;
4. use std.textio.all;
5. use ieee.std_logic_textio.all;
6.
7. entity tb_diff_ecuation is
8. end tb_diff_ecuation;
9.
10. architecture Behavioral of tb_diff_ecuation is
11.
12.     --Bits de la palabra de entrada
13.     constant N : integer := 8;
14.     --Bits de la palabra de salida
15.     constant M : integer := 16;
16.
17.     constant CLK_PERIOD : time := 10 ns;
18.     signal clk_sig : std_logic := '1';
19.     signal rst_sig : std_logic := '0';
20.
21.     signal s_axis_tdata_tb      : std_logic_vector(N-1 downto 0);
22.     signal s_axis_tvalid_tb    : std_logic;
23.     signal s_axis_tready_tb   : std_logic;
24.     signal m_axis_tdata_tb      : std_logic_vector(M-1 downto 0);
25.     signal m_axis_tvalid_tb    : std_logic;
26.     signal m_axis_tready_tb   : std_logic;
27.
28.     --Entidad del módulo ecuación diferencial
29.     component diff_ecuation is
30.         generic (
31.             --Bits de la palabra de entrada
32.             N : integer := 8;
33.             --Bits de la palabra de salida
34.             M : integer := 16);
35.         Port (
36.             clk_i          : in STD_LOGIC;
37.             rst_i          : in STD_LOGIC;
38.             s_axis_tdata_i : in STD_LOGIC_VECTOR (N-1 downto 0);
39.             s_axis_tvalid_i : in STD_LOGIC;
40.             s_axis_tready_o : out STD_LOGIC;
41.             m_axis_tdata_o : out STD_LOGIC_VECTOR (M-1 downto 0);
42.             m_axis_tvalid_o : out STD_LOGIC;
43.             m_axis_tready_i : in STD_LOGIC);
44.         end component diff_ecuation;
45.
46.         constant in_file   : string  := "data_in.txt";
47.         constant out_file  : string  := "data_out.txt";
48.
49.         file r_fptr,w_fptr : text;
50.
51.         --Items para guardar
52.         constant ITEMS_TO_SAVE : integer := 200;
53.
54.         constant C_CLK_PERIOD : real := 10.0e-9; -- NS
55.     begin
56.
57.         -----
58.         -- Clocks and Reset
59.         -----
60.
61.         clk_sig <= not clk_sig after CLK_PERIOD/2;
62.         rst_sig <= '1'after 100 ns;
63.
64.         -----
65.         --Estimulos a las señales
66.         -----

```

```

67.      -- Read file process
68.      p_read_file : process is
69.          variable fstatus    : file_open_status;
70.          variable file_line   : line;
71.          variable slv_v       : integer;
72.      begin
73.          --Por defecto deshabilitamos la transacción
74.          --Abrimos el archivo
75.          file_open(fstatus, r_fp, in_file, read_mode);
76.          --Asignación por defecto.
77.          s_axis_tdata_tb     <= (others => '0');
78.          s_axis_tvalid_tb    <= '0';
79.          wait until (rst_sig = '1');
80.          --Hacemos un espera en ciclos de reloj
81.          delay_1 : for i in 0 to 10-1 loop
82.              wait until (clk_sig'event and clk_sig = '1');
83.          end loop;
84.          --Recorremos el archivo
85.          loop_file : while not endfile(r_fp) loop
86.              readline(r_fp,file_line);
87.              read(file_line,slv_v);
88.              report "Valor leido: " & integer'image(slv_v);
89.              --Generaremos la señal AXI con el pulso en '1' cuando recibamos el tready.
90.              if (s_axis_tready_tb = '1') then
91.                  s_axis_tdata_tb <= std_logic_vector(to_signed(slv_v, N));
92.                  s_axis_tvalid_tb <= '1';
93.              end if;
94.              --Esperamos 10 ciclos de relog y se obtiene el periodo de la frecuencia de muestreo
95.              wait until (clk_sig'event and clk_sig = '1');
96.              s_axis_tvalid_tb <= '0';
97.              delay_2 : for i in 0 to 9-1 loop
98.                  wait until (clk_sig'event and clk_sig = '1');
99.              end loop;
100.             end loop ; -- loop_file
101.             s_axis_tdata_tb     <= (others => '0');
102.             s_axis_tvalid_tb    <= '0';
103.             report "Fin lectura del archivo";
104.             file_close(r_fp);
105.             wait;
106.         end process; -- p_read_file
107.
108.         p_write_file : process is
109.             variable fstatus    : file_open_status;
110.             variable file_line   : line;
111.             variable v_int       : integer;
112.             variable v_std_lv    : std_logic_vector((m_axis_tdata_tb'LENGTH - 1) downto 0);
113.         begin
114.             -- Para este caso vamos a aceptar todos los datos que salgan.
115.             m_axis_tready_tb <= '1';
116.             --Abrimos el archivo
117.             file_open(fstatus, w_fp, out_file, write_mode);
118.             wait until (rst_sig = '1');
119.
120.             --Vamos a escribir solo 200.
121.             write_file : for i in 0 to ITEMS_TO_SAVE loop
122.                 wait until (m_axis_tvalid_tb = '1');
123.                 v_int      := to_integer(signed(m_axis_tdata_tb));
124.                 v_std_lv   := m_axis_tdata_tb;
125.                 write(file_line,v_int);
126.                 write(file_line,v_std_lv,right,40);
127.                 writeline(w_fp, file_line);
128.                 report "Valor ESCRITO: " & integer'image(v_int);
129.                 report "Indice: " & integer'image(i);
130.             end loop;
131.             report "Fin escritura del archivo";
132.             file_close(w_fp);
133.             wait;
134.         end process; -- p_write_file
135.
136. 
```

```

137. --Instancia que se pone a prueba en el testbench
138. -----
139.
140.     inst_diff_ecuation : diff_ecuation
141.         generic map (
142.             N => 8,
143.             M => 16
144.         )
145.         port map (
146.             clk_i          => clk_sig,
147.             rst_i          => rst_sig,
148.             s_axis_tdata_i => s_axis_tdata_tb,
149.             s_axis_tvalid_i => s_axis_tvalid_tb,
150.             s_axis_tready_o => s_axis_tready_tb,
151.             m_axis_tdata_o => m_axis_tdata_tb,
152.             m_axis_tvalid_o => m_axis_tvalid_tb,
153.             m_axis_tready_i => m_axis_tready_tb
154.         );
155.
156. end Behavioral;

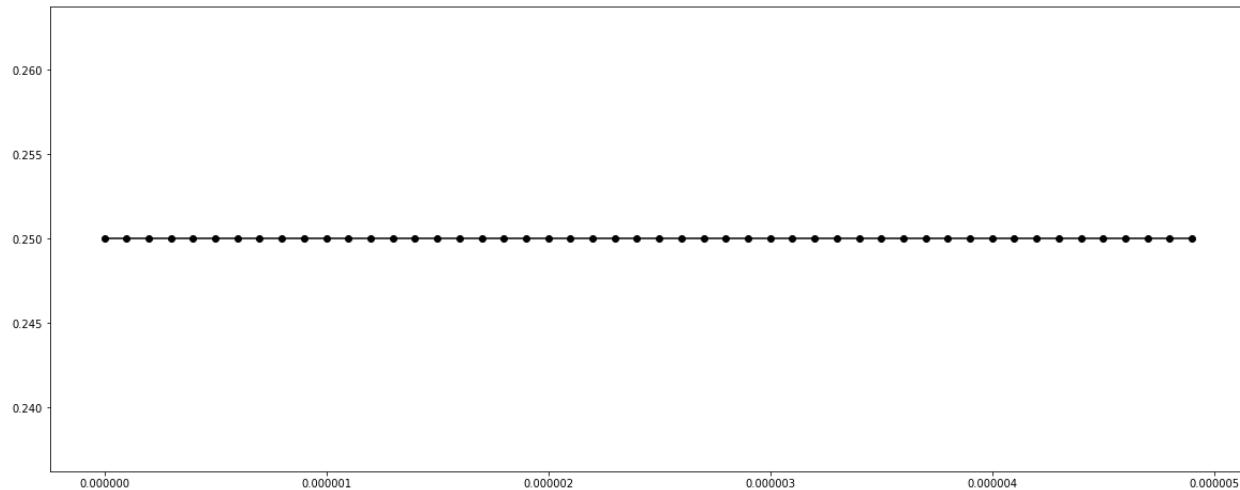
```

## Simulación

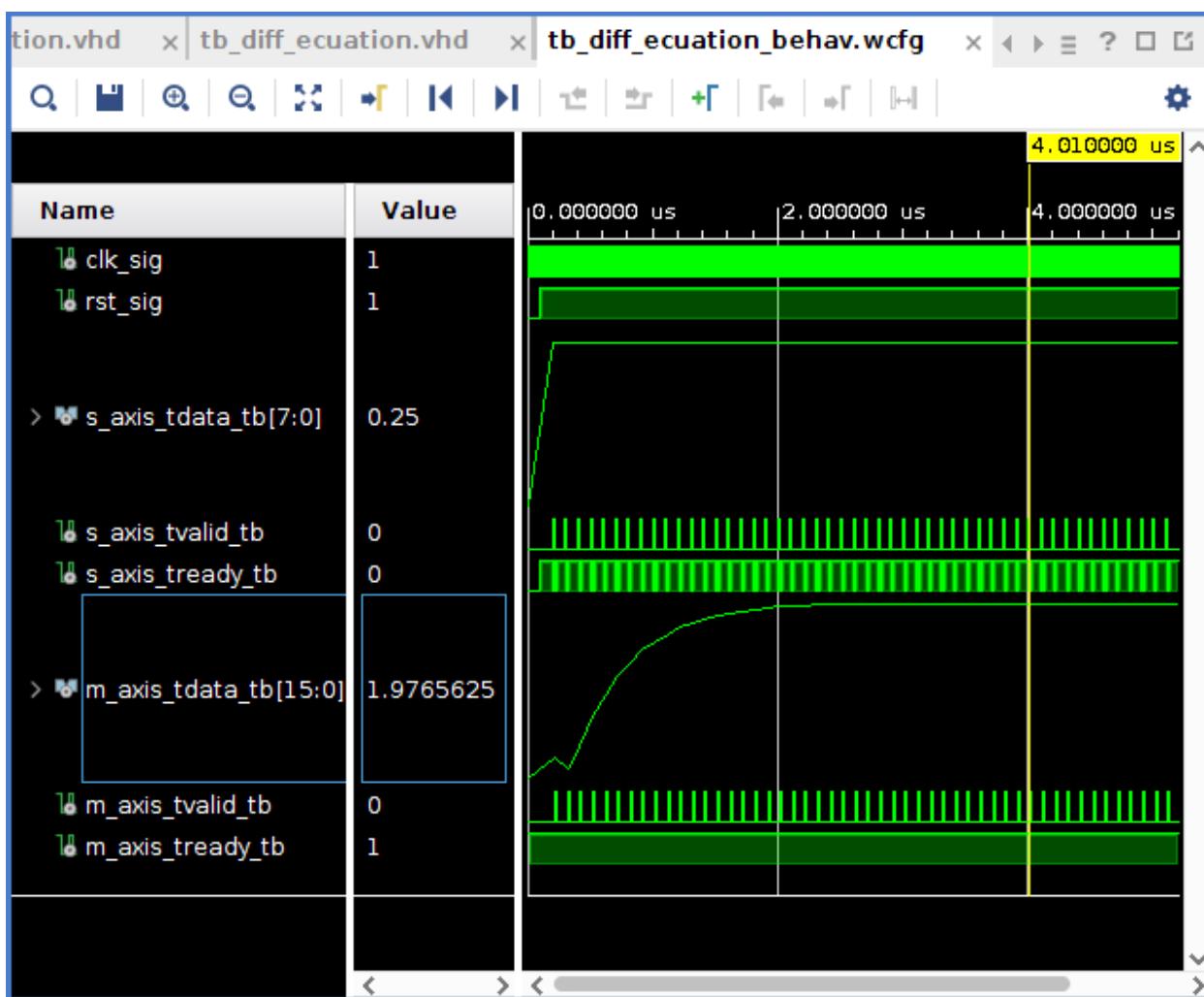
**Entrada al sistema señal de DC con amplitud de 0.25.**

Se obtiene el vector de los valores desde Python para la excitación del sistema con 50 muestras en total. Se estableció una señal de DC con una amplitud de 0,25.

La siguiente figura expone el vector de la señal generada en Python.

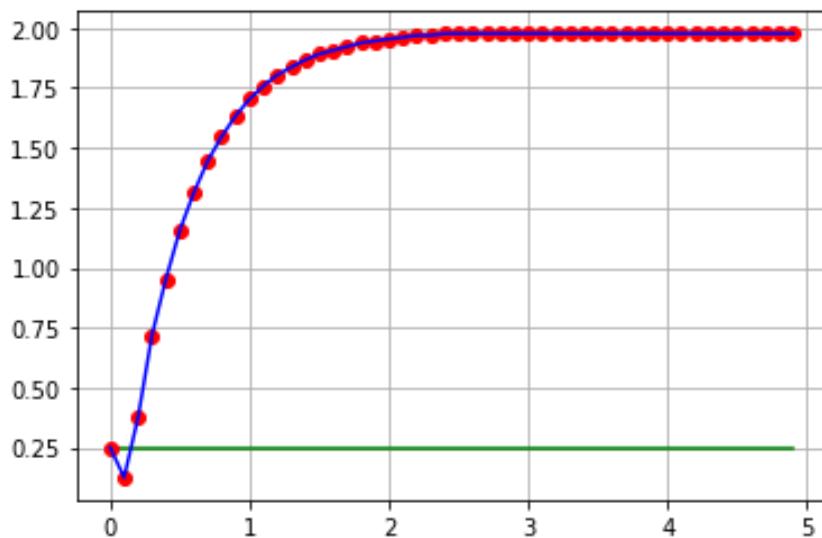


Después de realizar la simulación del sistema se obtiene la siguiente respuesta.



Se recogió la respuesta de la señal en el archivo de salida y se cargó en el script de Python para el análisis.

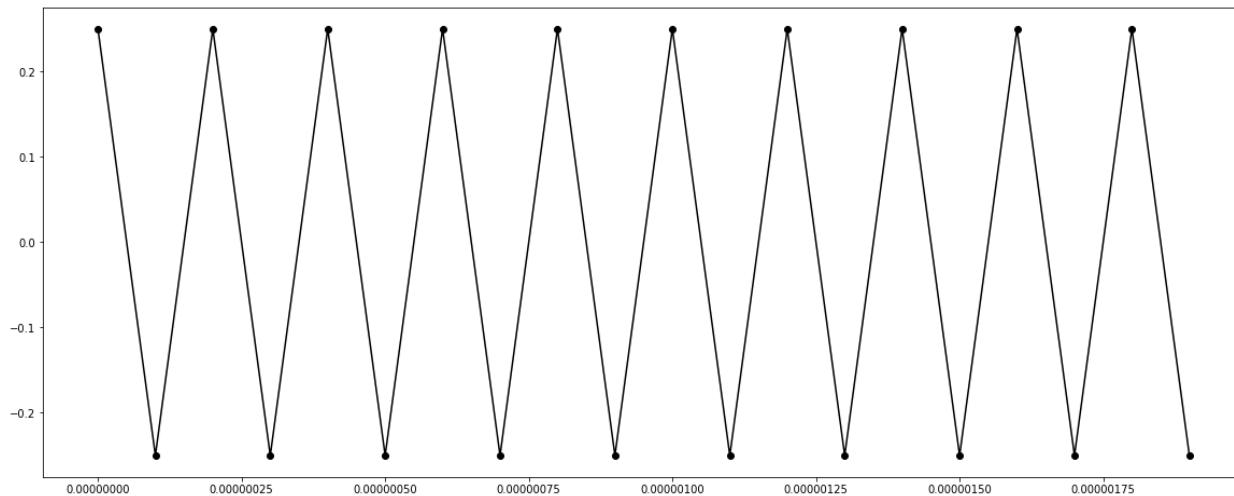
Se puede apreciar que la señal de salida tiene una amplitud cercana a dos. Eso quiere decir que la señal de entrada en DC es multiplicada por 8, como se aprecia en la figura de respuesta del sistema en magnitud y fase.



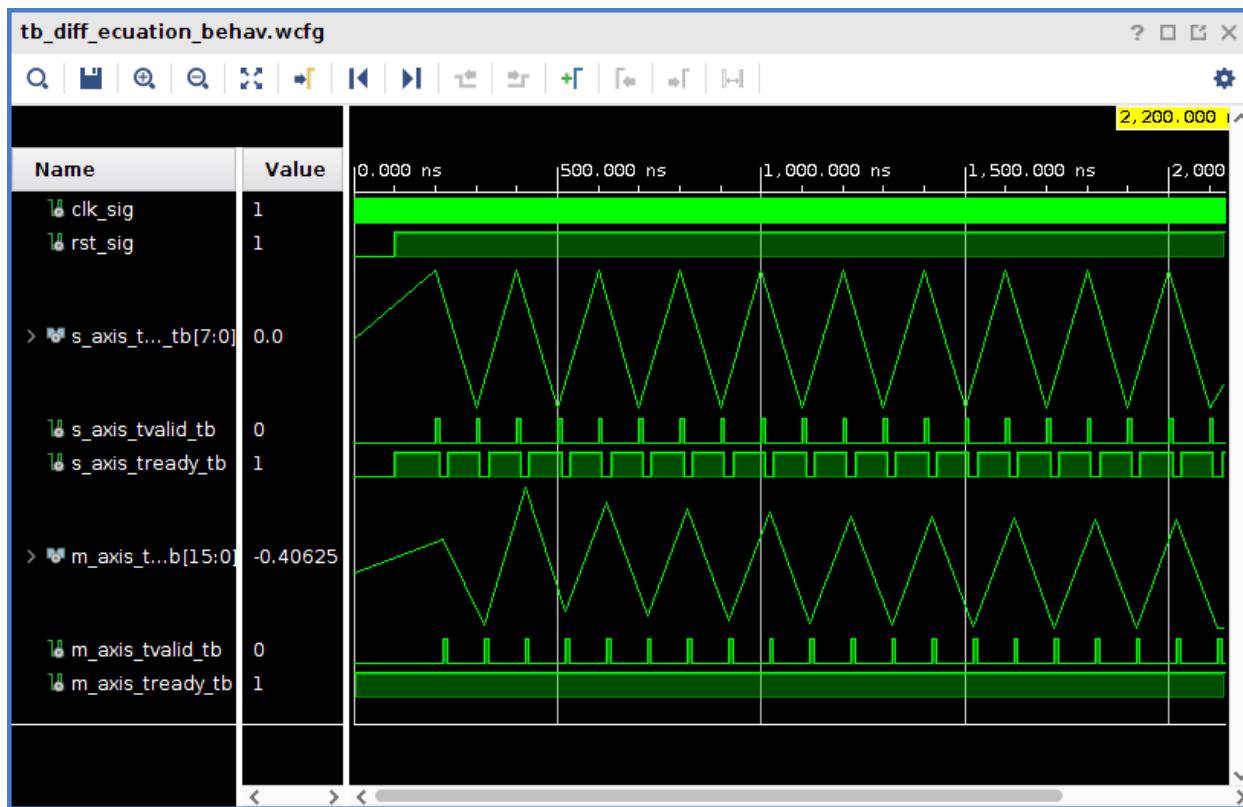
### Entrada al sistema señal sinusoidal con Frecuencia Nyquist.

Se obtiene el vector de los valores desde Python para la excitación del sistema con 20 muestras en total. Se estableció como frecuencia de muestreo de 10 MHz con un valor de periodo de 100 ns, por lo tanto, la frecuencia de Nyquist del sistema es igual a 5 MHz con un periodo de 200 nS.

La siguiente figura expone el vector de la señal generada en Python.

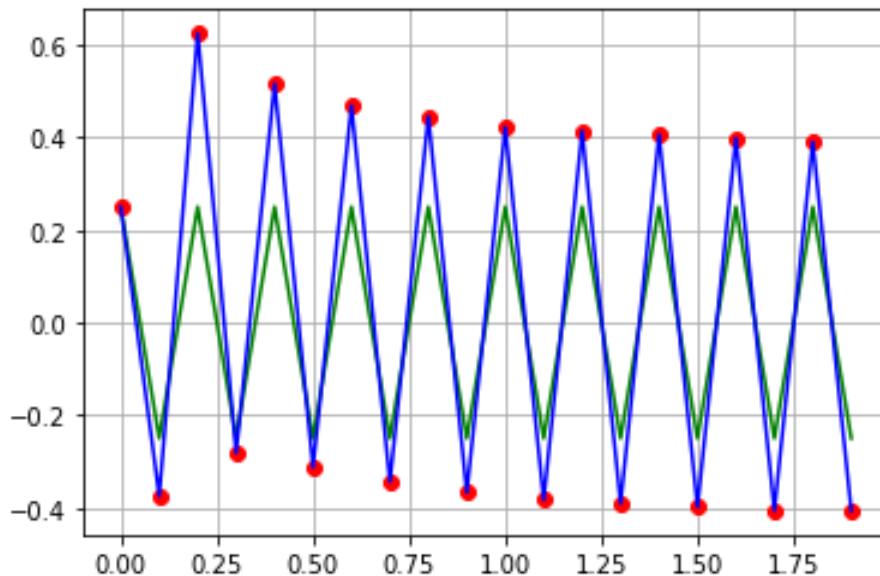


Después de realizar la simulación del sistema con la frecuencia de Nyquist como señal de entrada se obtiene la siguiente respuesta.



Se recogió la respuesta de la señal en el archivo de salida y se cargó en el script de Python para el análisis.

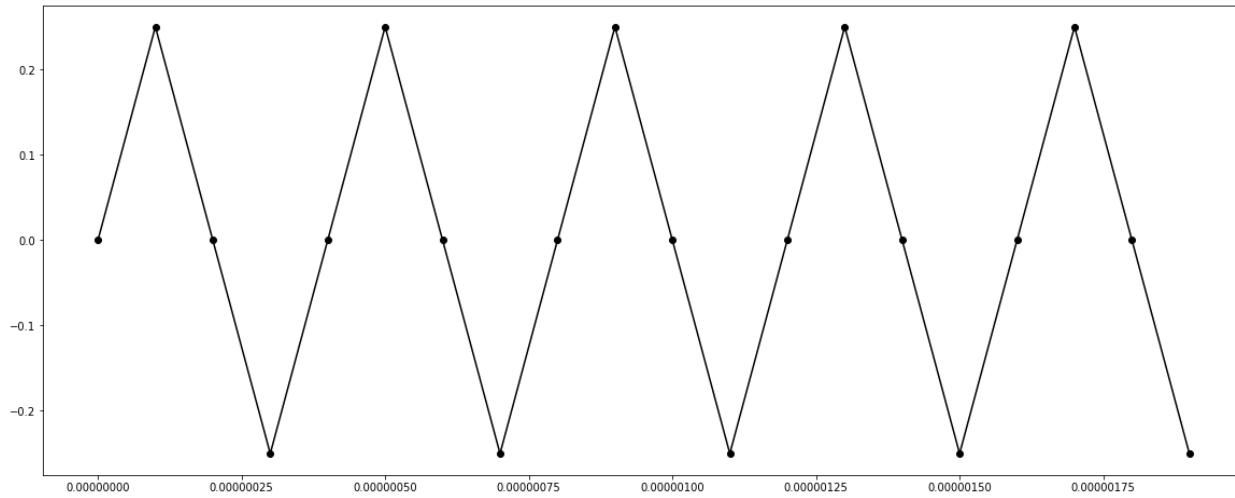
El tiempo está dado en (uSeg), la frecuencia de Nyquist es de 5 MHz con un periodo de 200 ns y amplitud de entrada de 0,25.



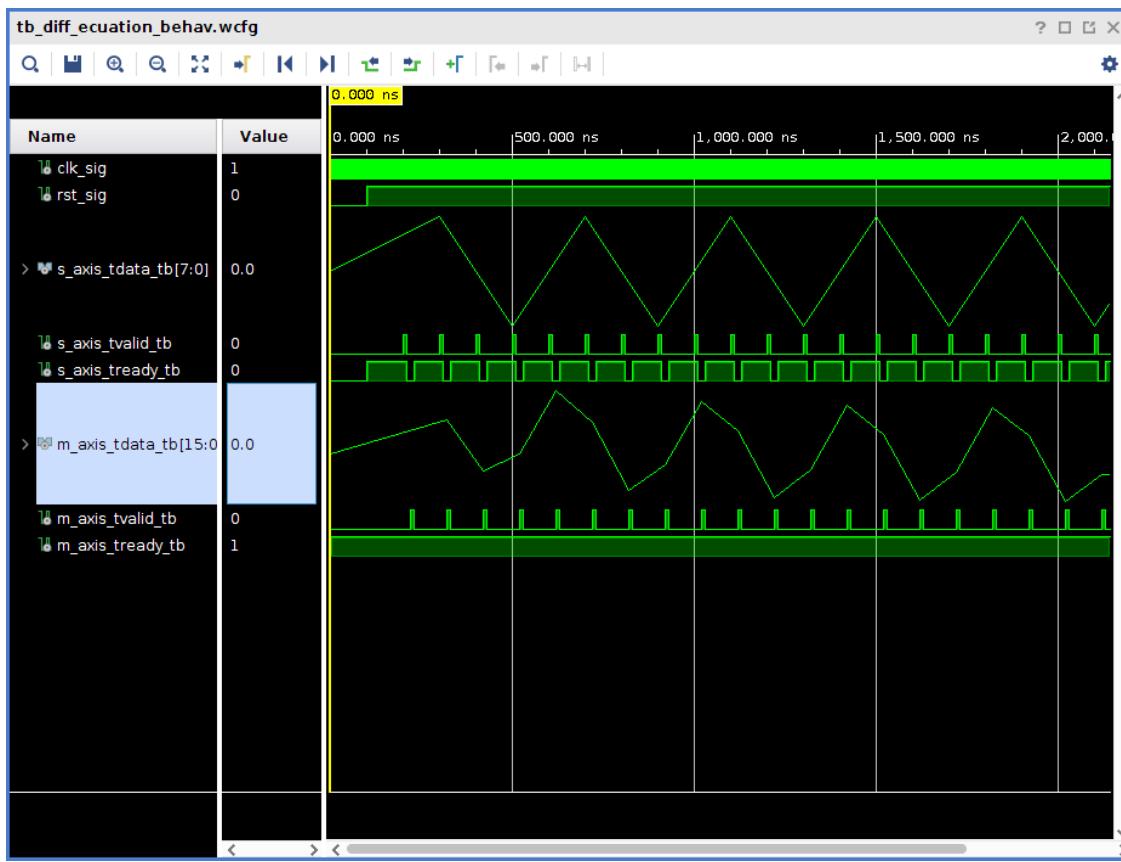
**Entrada al sistema señal sinusoidal con Frecuencia Nyquist / 2.**

Se aplica una señal de entrada con una frecuencia de Nyquist / 2 igual a 2,5 MHz con un periodo de 400 ns y amplitud de 0,25.

La siguiente figura expone el vector de la señal generada en Python.

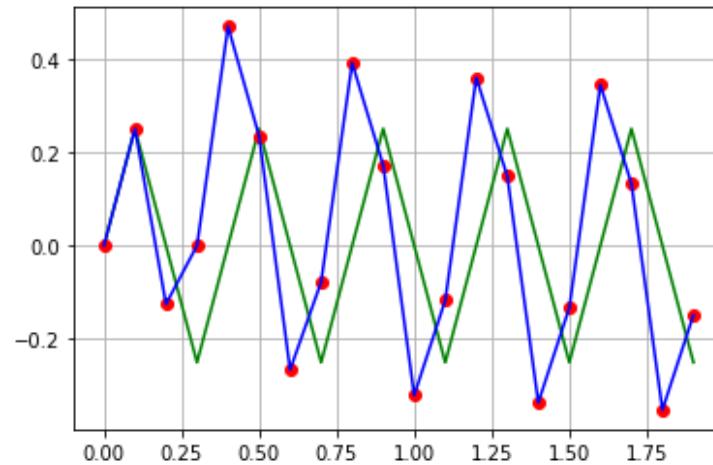


Después de realizar la simulación del sistema con la frecuencia de Nyquist / 2 como señal de entrada se obtiene la siguiente respuesta.



Se recogió la respuesta de la señal en el archivo de salida y se cargó en el script de Python para el análisis.

El tiempo está dado en (uSeg), la frecuencia de Nyquist / 2 es de 2,5 MHz con un periodo de 400 ns.

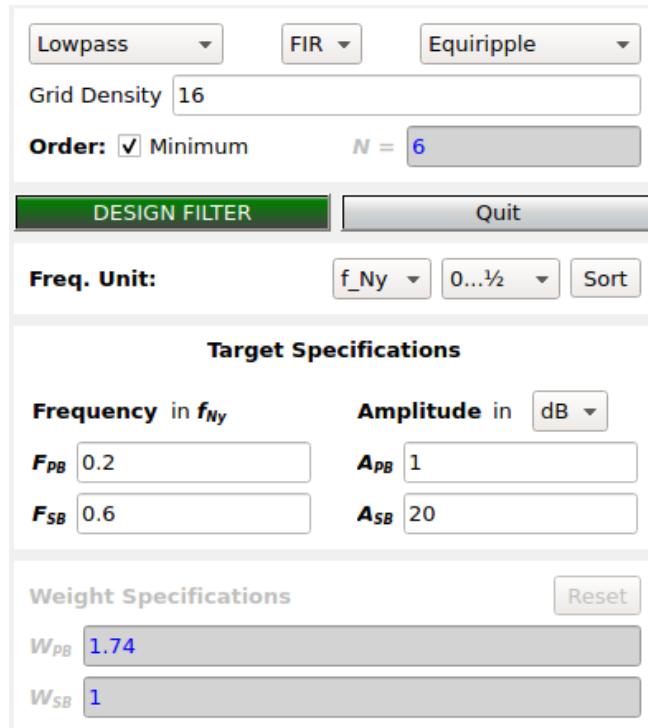


## 4. Ejercicio 4 “FIR Simétrico AXI S”

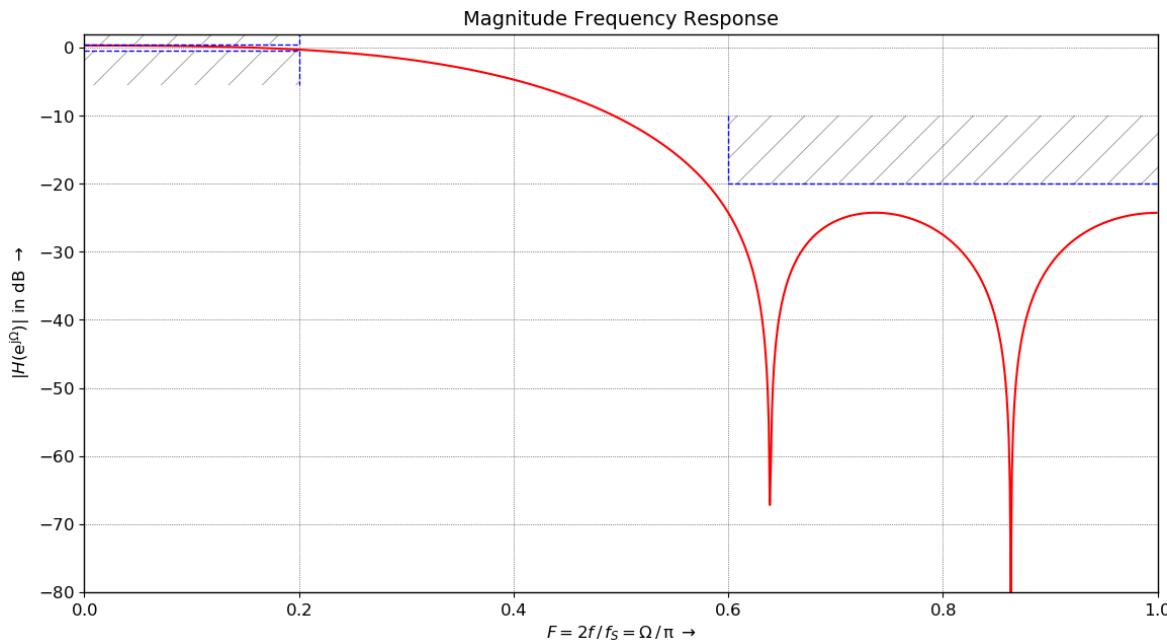
Para el análisis de sistema lineales:

Se diseñó en pyfda (@Python) un filtro FIR de fase lineal siguiendo la siguiente plantilla:

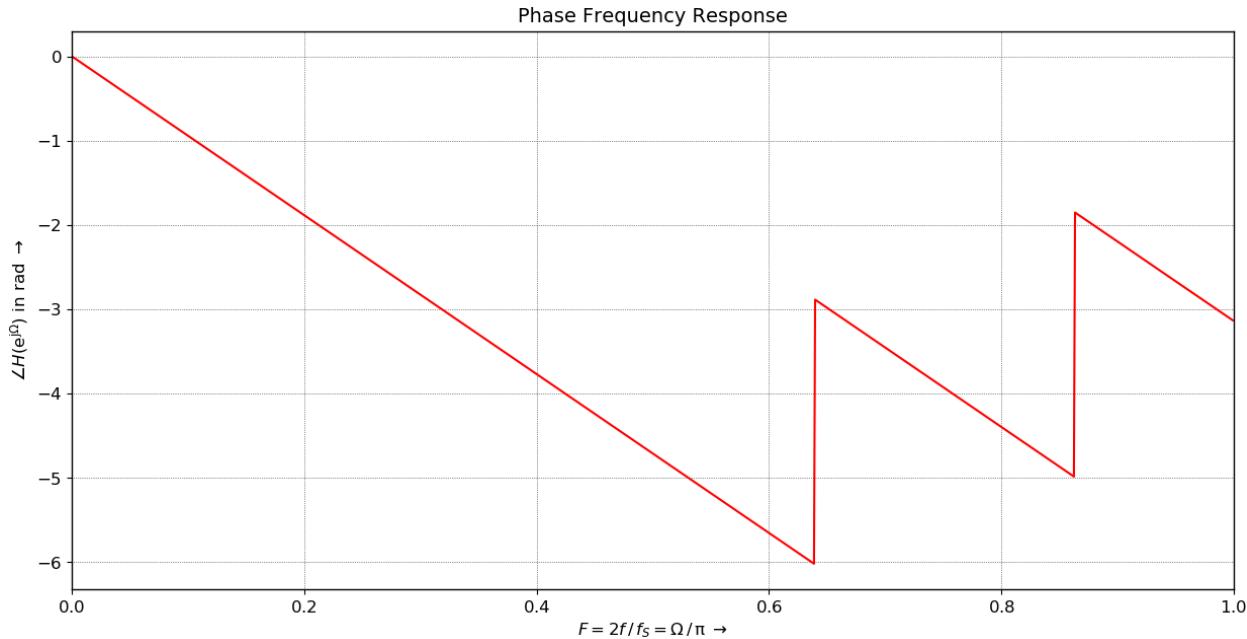
- Tipo de Filtro: Pasabajos
- Método de diseño : Equiripple (Orden Mínimo)
- Frecuencia de banda de paso:  $0.2 f_{Ny} \Rightarrow 0.1 \text{ fs}$
- Frecuencia de banda de rechazo:  $0.6 f_{Ny} \Rightarrow 0.3 \text{ fs}$
- Ripple en banda de paso: 1 db
- Atenuación en banda de rechazo: 20 db



- Como se puede apreciar el filtro quedó establecido de orden 6 para cumplir con los requerimientos de diseño.
- Se puede apreciar la respuesta en frecuencia de la magnitud.



- Se puede apreciar la respuesta en frecuencia de la fase.



- Se realiza la transformada Z del polinomio y se obtiene  $H(z)$ .

Se tiene la siguiente singularidad del polinomio N(z) y D(z):

$$b = [-0.04896, 0.06276, 0.2925, 0.4226, 0.2925, 0.06276, -0.04896]$$

Se obtiene la siguiente ecuación de diferencia:

$$y(n) = -0.04896x(n) + 0.06276x(n-1) + 0.2925x(n-2) + 0.4226x(n-3) + 0.2925x(n-4) + 0.06276x(n-5) - 0.04896x(n-6)$$

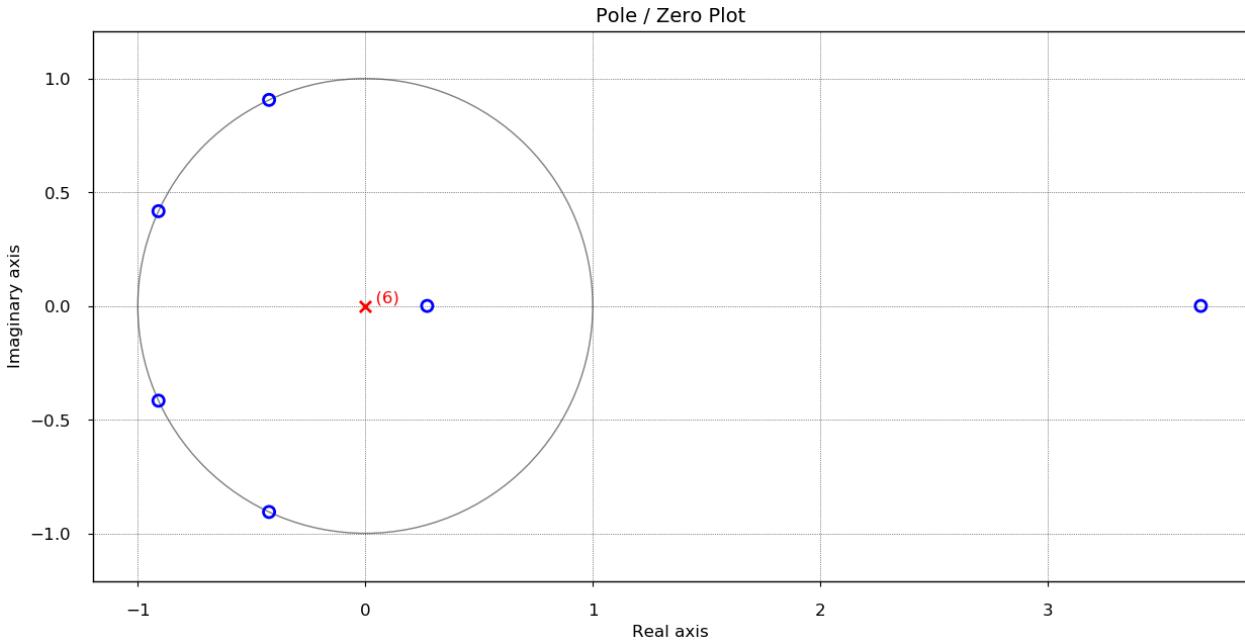
Se realiza la transformada Z a ambos lados de la igualdad.

$$Y(z) = -0.04896X(z) + 0.06276z^{-1} \cdot X(z) + 0.2925z^{-2} \cdot X(z) + 0.4226z^{-3} \cdot X(z) + 0.2925z^{-4} \cdot X(z) + 0.06276z^{-5} \cdot X(z) - 0.04896z^{-6} \cdot X(z)$$

$$Y(z) = X(z)(-0.04896 + 0.06276z^{-1} + 0.2925z^{-2} + 0.4226z^{-3} + 0.2925z^{-4} + 0.06276z^{-5} - 0.04896z^{-6})$$

$$\frac{Y(z)}{X(z)} = H(z) = \frac{-0.04896 + 0.06276z^{-1} + 0.2925z^{-2} + 0.4226z^{-3} + 0.2925z^{-4} + 0.06276z^{-5} - 0.04896z^{-6}}{1} = \frac{N(z)}{D(z)}$$

- Diagrama de polos y ceros del filtro.



Con el diagrama de polos y ceros se puede concluir dos situaciones:

El sistema es **FIR** ya que no posee polos, o se encuentran en el punto cero del sistema y la respuesta al pulso tiende a ser finita.

También que el sistema es estable debido a que no existen polos fuera del círculo unitario.

## Código RTL

```

1.  library ieee;
2.  use ieee.std_logic_1164.all ;
3.  use ieee.numeric_std.all ;
4.
5.  entity symmetrical_fir is
6.    generic (
7.      --Bits de la palabra de entrada
8.      N : integer := 16;
9.      --Bits de la palabra de salida
10.     M : integer := 23);
11.   Port (
12.     clk_i : in STD_LOGIC;
13.     rst_i : in STD_LOGIC;
14.     s_axis_tdata_i : in STD_LOGIC_VECTOR (N-1 downto 0);
15.     s_axis_tvalid_i : in STD_LOGIC;
16.     s_axis_tready_o : out STD_LOGIC;
17.     m_axis_tdata_o : out STD_LOGIC_VECTOR (M-1 downto 0);
18.     m_axis_tvalid_o : out STD_LOGIC;
19.     m_axis_tready_i : in STD_LOGIC);
20. end symmetrical_fir;
21.
22. architecture Behavioral of symmetrical_fir is
23.
24.   constant n_taps: integer := 6;
25.
26.   --Vector con los datos que se van a ir almacenando
27.   type vector_type is array (integer range <>) of std_logic_vector(s_axis_tdata_i'RANGE);
28.   signal vector_reg : vector_type(0 to n_taps);
29.
30.   component ff_module is
31.     generic (
32.       N : integer := N
33.     );
34.     Port ( clk_i      : in STD_LOGIC;
35.            rst_i      : in STD_LOGIC;
36.            enable_t_i : in STD_LOGIC;
37.            data_i     : in STD_LOGIC_VECTOR (N-1 downto 0);
38.            data_o     : out STD_LOGIC_VECTOR (N-1 downto 0));
39.   end component ff_module;
40.
41.   component mul_module is
42.     generic (
43.       N : integer := 16;
44.       M : integer := 18);
45.     Port ( coeff_in : in STD_LOGIC_VECTOR (N - 1 downto 0);
46.            data_in : in STD_LOGIC_VECTOR (N - 1 downto 0);
47.            data_out : out STD_LOGIC_VECTOR (M - 1 downto 0));
48.   end component mul_module;
49.
50.   --Definición de los estados de la maquina de estados del proceso del encoder.
51.   type state_type is
52.   (
53.     ST_WAIT_DATA,
54.     ST_PROCESS_DATA,
55.     ST_TRANSMIT_DATA
56.   );
57.
58.   --Generamos el arreglo de registro
59.   type reg_axi_type is record
60.     state          : state_type;
61.     --Señal de Ready del puerto de salida del AXI Stream Slave
62.     s_axis_tready  : std_logic;

```

```

63.      --Señales para el manejo del puerto AXIS4-Stream
64.      m_axis_tdata      : std_logic_vector(m_axis_tdata_o'RANGE);
65.      m_axis_tvalid     : std_logic;
66.      enable_process   : std_logic;
67.  end record;
68.
69.      --Valor por defecto de los registros.
70.      constant reg_ctrl_default : reg_axi_type := (
71.          state        => ST_WAIT_DATA,
72.          s_axis_tready => '0',
73.          m_axis_tdata  => (others => '0'),
74.          m_axis_tvalid => '0',
75.          enable_process => '0'
76.      );
77.
78.      --Registros internos de variables de control
79.      signal axi_crt : reg_axi_type := reg_ctrl_default;
80.
81.      type coeff_type is array(0 to n_taps) of real;
82.      constant coeff : coeff_type := (
83.          -0.04895593695235194,
84.          0.06275537458572507,
85.          0.29246401053451393,
86.          0.4226075033374576,
87.          0.29246401053451393,
88.          0.06275537458572507,
89.          -0.04895593695235194);
90.
91.      --constante para los decimales de los factores en este caso es Q1.15
92.      constant NDEC_FACTOR : integer := 15;
93.
94.      --Se realiza una conversión de los coeficientes de los factores de real
95.      --a un entero con formato Q1.15
96.      type coeff_type_Q15 is array(0 to n_taps) of integer;
97.      constant coeff_Q15 : coeff_type_Q15 := (
98.          integer((coeff(0))*real((2**NDEC_FACTOR))),
99.          integer((coeff(1))*real((2**NDEC_FACTOR))),
100.         integer((coeff(2))*real((2**NDEC_FACTOR))),
101.         integer((coeff(3))*real((2**NDEC_FACTOR))),
102.         integer((coeff(4))*real((2**NDEC_FACTOR))),
103.         integer((coeff(5))*real((2**NDEC_FACTOR))),
104.         integer((coeff(6))*real((2**NDEC_FACTOR)))
105.     );
106.
107.     type factor_type_Q17 is array(0 to n_taps) of std_logic_vector(18 - 1 downto 0);
108.     signal factor_Q17 : factor_type_Q17;
109.
110.     signal result_t : signed(M-1 downto 0);
111.
112.     -----
113.     --Comienzo del comportamiento de la entidad diff_ecuation
114.     -----
115. begin
116.
117.     --Se genera el arreglo de 5 registros de entrada de la señal
118.     vector_reg(0) <= s_axis_tdata_i;
119.     gen_array_ff : for i in 0 to n_taps - 1 generate
120.         inst_ff_module : ff_module
121.             generic map (
122.                 N => N
123.             )
124.             port map (
125.                 clk_i      => clk_i,
126.                 rst_i      => rst_i,
127.                 enable_t_i => axi_crt.enable_process,
128.                 data_i     => vector_reg(i),
129.                 data_o     => vector_reg(i + 1)
130.             );
131.     end generate;
132.
```

```

133.      --Se genera el arreglo de los multiplicadores y se incluye el truncado a Q1.17
134.      gen_array_mul : for i in 0 to n_taps generate
135.          inst_mul_module : mul_module
136.              port map (
137.                  coeff_in  => std_logic_vector(to_signed(coeff_Q15(i),N)),
138.                  data_in   => vector_reg(i),
139.                  data_out  => factor_Q17(i)
140.              );
141.      end generate;
142.
143.      --Se realiza la suma de los resultados de la multiplicación de las
144.      --muestras con determinado factor
145.      --El resultado es un vector de 23 bits con formato Q6.17
146.      sum_process : process(all) is
147.          variable temp : signed(M - 1 downto 0);
148.      begin
149.          temp := (others => '0');
150.          loop_vector : for i in 0 to n_taps loop
151.              temp := temp + signed(factor_Q17(i));
152.          end loop;
153.          result_t <= temp;
154.      end process;
155.
156.      -----
157.      --Manejo de los AXI4 Stream (Slave y Master)
158.      -----
159.
160.      fsm_process : process (clk_i, rst_i)
161.      begin
162.          if(rst_i = '0') then
163.              axi_crt <= reg_ctr_default;
164.          elsif rising_edge(clk_i) then
165.              case(axi_crt.state) is
166.                  when ST_WAIT_DATA =>
167.                      axi_crt.s_axis_tready <= '1';
168.                      axi_crt.m_axis_tvalid <= '0';
169.                      axi_crt.enable_process <= '0';
170.                      if (s_axis_tvalid_i = '1' and s_axis_tready_o = '1') then
171.                          axi_crt.state <= ST_PROCESS_DATA;
172.                          axi_crt.s_axis_tready <= '0';
173.                          axi_crt.enable_process <= '1';
174.                      end if;
175.                  when ST_PROCESS_DATA =>
176.                      axi_crt.s_axis_tready <= '0';
177.                      axi_crt.m_axis_tvalid <= '1';
178.                      axi_crt.enable_process <= '0';
179.                      axi_crt.m_axis_tdata <= std_logic_vector(resize(result_t,M));
180.                      axi_crt.state <= ST_TRANSMIT_DATA;
181.                  when ST_TRANSMIT_DATA =>
182.                      axi_crt.s_axis_tready <= '1';
183.                      axi_crt.m_axis_tvalid <= '0';
184.                      axi_crt.enable_process <= '0';
185.                      if (m_axis_tready_i = '1' and m_axis_tvalid_o = '1') then
186.                          axi_crt.state <= ST_WAIT_DATA;
187.                      end if;
188.                  when others => null;
189.              end case;
190.          end if;
191.      end process;
192.
193.      m_axis_tdata_o  <= axi_crt.m_axis_tdata;
194.      m_axis_tvalid_o <= axi_crt.m_axis_tvalid;
195.      s_axis_tready_o <= axi_crt.s_axis_tready;
196.
197.  end Behavioral;

```

```

1.      library ieee ;
2.      use ieee.std_logic_1164.all ;
3.      use ieee.numeric_std.all ;
4.
5.      entity ff_module is
6.          generic (
7.              N : integer := 8
8.          );
9.          Port ( clk_i      : in STD_LOGIC;
10.                 rst_i      : in STD_LOGIC;
11.                 enable_t_i : in STD_LOGIC;
12.                 data_i     : in STD_LOGIC_VECTOR (N-1 downto 0);
13.                 data_o     : out STD_LOGIC_VECTOR (N-1 downto 0));
14.      end ff_module;
15.
16.      architecture Behavioral of ff_module is
17.          signal reg_signal : std_logic_vector(data_o'RANGE) ;
18.      begin
19.
20.          reg_process : process (clk_i, rst_i)
21.          begin
22.              if (rst_i = '0') then
23.                  reg_signal <= (others => '0');
24.              elsif (rising_edge(clk_i)) then
25.                  if (enable_t_i = '1') then
26.                      reg_signal <= data_i;
27.                  end if;
28.              end if;
29.          end process reg_process;
30.
31.          data_o <= reg_signal;
32.
33.      end Behavioral;

```

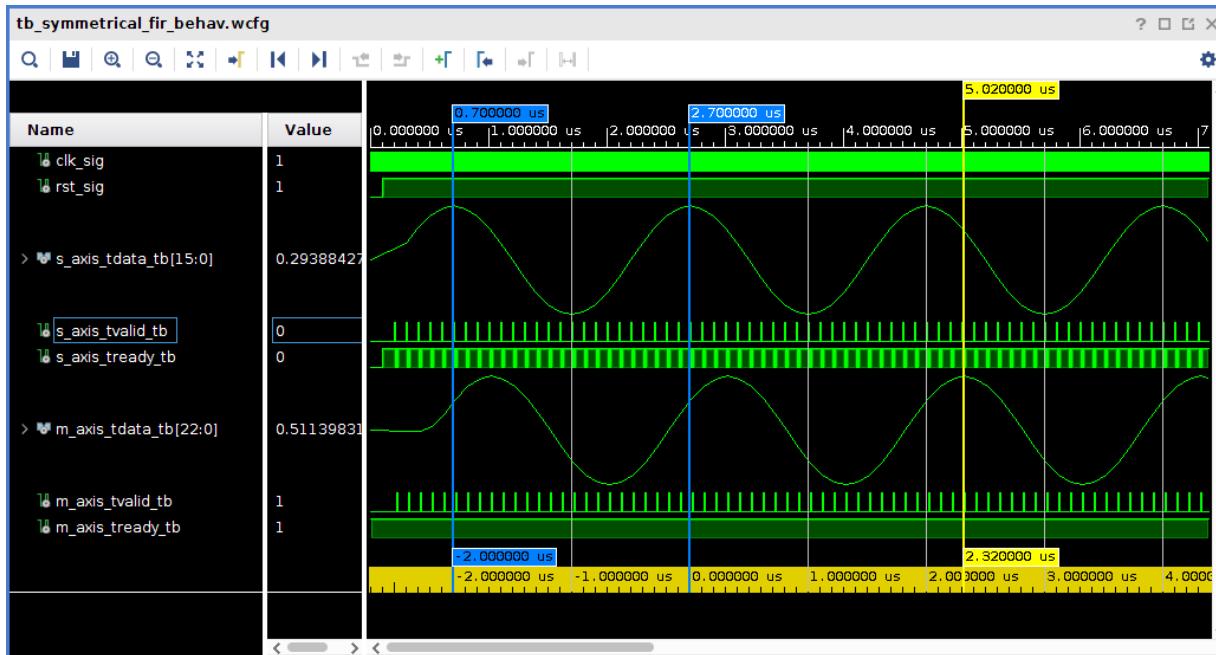
```

1.      library ieee;
2.      use ieee.std_logic_1164.all ;
3.      use ieee.numeric_std.all ;
4.
5.      entity mul_module is
6.          generic (
7.              N : integer := 17;
8.              M : integer := 18);
9.          Port ( coeff_in : in STD_LOGIC_VECTOR (N - 1 downto 0);
10.                   data_in : in STD_LOGIC_VECTOR (N - 1 downto 0);
11.                   data_out : out STD_LOGIC_VECTOR (M-1 downto 0));
12.      end mul_module;
13.
14.      architecture Behavioral of mul_module is
15.
16.          --Señal que contiene el resultado de la multiplicación de la entrada y su factor.
17.          --EL resultado tiene una palabra de 34bits con formato Q4.30
18.          signal mul : std_logic_vector(2*N - 1 downto 0);
19.
20.      begin
21.
22.          --Se realiza la multiplicación de la entrada con el factor los formatos de los dos
23.          --números son Q1.15, 16bits
24.          mul <= std_logic_vector(to_signed(to_integer(signed(coeff_in)) *
25.          to_integer(signed(data_in)), 2*N));
26.          --Se realiza el truncado del resultado quedando un número con formato Q1.17, 18 bits
27.          data_out <= mul(30 downto 13);
28.
29.      end Behavioral;

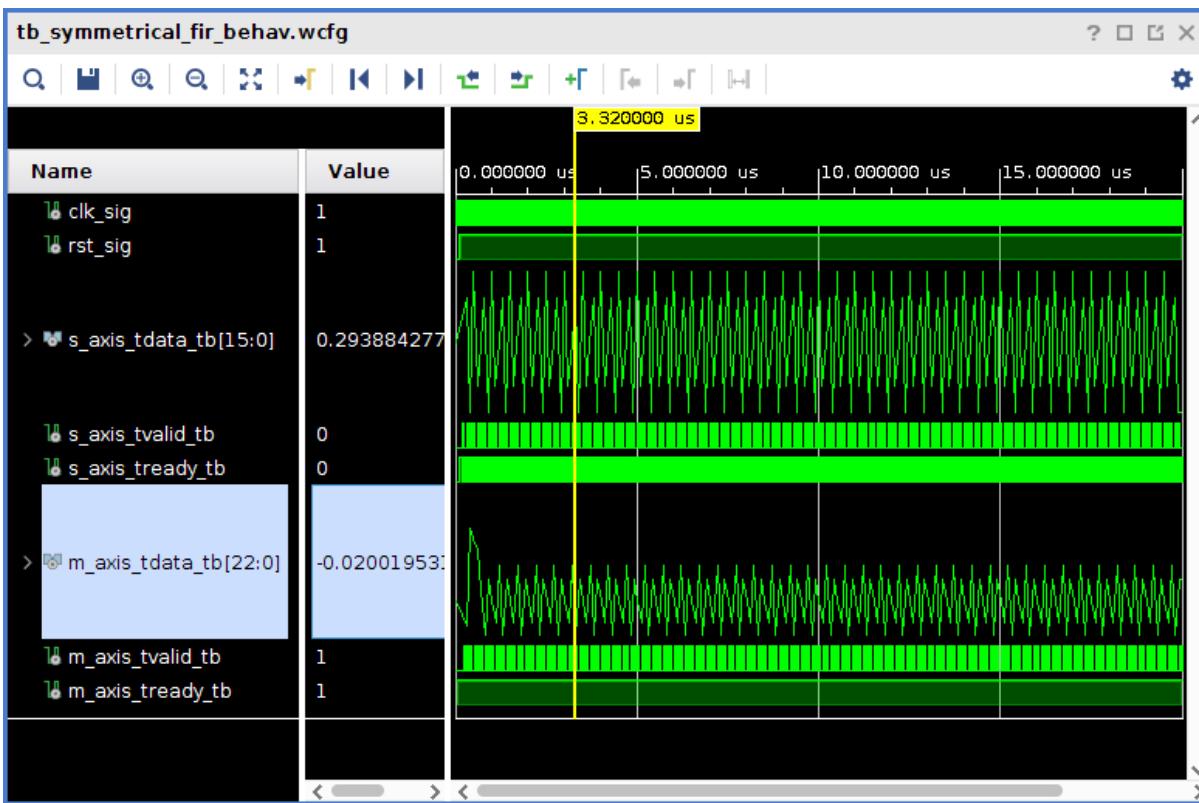
```

## Simulación

Como se aprecia en la siguiente figura se excitó el sistema con una señal con frecuencia igual a 0.1 de Nyquist y con una amplitud de 0.5. Como la señal de muestreo se estableció en 10 Mhz entonces la frecuencia de Nyquist es igual a 5 Mhz por el 0.1 da como resultado una señal sinusoidal de 500 Khz con periodo de 2 us.



En la siguiente figura se excitó el sistema con una señal igual a 0.8 de la frecuencia de Nyquist y a una amplitud de 0.5. Como la señal de muestreo fs se estableció en 10 Mhz entonces la frecuencia de Nyquist es igual a 5 Mhz por el 0.8 da como resultado una señal sinusoidal de 4 Mhz con periodo de 250 ns.



Se puede apreciar que con la frecuencia de 0.1 Nyquist la señal pasa totalmente a través del filtro. Cuando se ingresa una frecuencia de 0.8 Nyquist la amplitud se atenúa considerablemente. Los resultados cumplen con lo esperado y con las especificaciones del filtro basándose en la respuesta en frecuencia de la magnitud.

**Se realizan los puntos 5,6,7 y 8 teniendo en cuenta la simetría del filtro.**

**Código RTL**

```

1.      library ieee;
2.      use ieee.std_logic_1164.all ;
3.      use ieee.numeric_std.all ;
4.
5.      entity symmetrical_fir is
6.          generic (
7.              --Bits de la palabra de entrada
8.              N : integer := 16;
9.              --Bits de la palabra de salida
10.             M : integer := 21);
11.            Port (
12.                 clk_i : in STD_LOGIC;
13.                 rst_i : in STD_LOGIC;
14.                 s_axis_tdata_i : in STD_LOGIC_VECTOR (N-1 downto 0);
15.                 s_axis_tvalid_i : in STD_LOGIC;
16.                 s_axis_tready_o : out STD_LOGIC;
17.                 m_axis_tdata_o : out STD_LOGIC_VECTOR (M-1 downto 0);
18.                 m_axis_tvalid_o : out STD_LOGIC;
19.                 m_axis_tready_i : in STD_LOGIC);
20.         end symmetrical_fir;
21.
22.         architecture Behavioral of symmetrical_fir is
23.
24.             constant n_taps: integer := 6;
25.             constant n_taps_op: integer := 4;
26.
27.             --Vector con los datos que se van a ir almacenando
28.             type vector_type is array (integer range <>) of std_logic_vector(s_axis_tdata_i'RANGE);
29.             signal vector_reg : vector_type(0 to n_taps);
30.
31.             component ff_module is
32.                 generic (
33.                     N : integer := N
34.                 );
35.                 Port ( clk_i          : in STD_LOGIC;
36.                         rst_i          : in STD_LOGIC;
37.                         enable_t_i     : in STD_LOGIC;
38.                         data_i          : in STD_LOGIC_VECTOR (N-1 downto 0);
39.                         data_o          : out STD_LOGIC_VECTOR (N-1 downto 0));
40.         end component ff_module;
41.
42.             component mul_module is
43.                 generic (
44.                     N : integer := 17;
45.                     M : integer := 18);
46.                 Port ( coeff_in : in STD_LOGIC_VECTOR (N - 1 downto 0);
47.                         data_in : in STD_LOGIC_VECTOR (N - 1 downto 0);
48.                         data_out : out STD_LOGIC_VECTOR (M - 1 downto 0));
49.         end component mul_module;
50.
51.             --Definición de los estados de la maquina de estados del proceso del encoder.
52.             type state_type is
53.             (
54.                 ST_WAIT_DATA,
55.                 ST_PROCESS_DATA,
56.                 ST_TRANSMIT_DATA
57.             );
58.
59.             --Generamos el arreglo de registro
60.             type reg_axi_type is record
61.                 state           : state_type;
62.                 --Señal de Ready del puerto de salida del AXI Stream Slave
63.                 s_axis_tready   : std_logic;
64.                 --Señales para el manejo del puerto AXIS4-Stream
65.                 m_axis_tdata    : std_logic_vector(m_axis_tdata_o'RANGE);
66.                 m_axis_tvalid   : std_logic;
67.                 enable_process : std_logic;

```

```

68.      end record;
69.
70.      --Valor por defecto de los registros.
71.      constant reg_ctrl_default : reg_axi_type := (
72.          state          => ST_WAIT_DATA,
73.          s_axis_tready  => '0',
74.          m_axis_tdata   => (others => '0'),
75.          m_axis_tvalid  => '0',
76.          enable_process => '0'
77.      );
78.
79.      --Registros internos de variables de control
80.      signal axi_crt : reg_axi_type := reg_ctrl_default;
81.
82.      type coeff_type is array(0 to n_taps_op - 1) of real;
83.      constant coeff : coeff_type := (
84.          -0.04895593695235194,
85.          0.06275537458572507,
86.          0.29246401053451393,
87.          0.4226075033374576
88.      );
89.
90.      --constante para los decimales de los factores en este caso es Q1.15
91.      constant NDEC_FACTOR : integer := 15;
92.
93.      --Se realiza una conversión de los coeficientes de los factores de real
94.      --a un entero con formato Q1.15
95.      type coeff_type_Q15 is array(0 to n_taps_op - 1) of integer;
96.      constant coeff_Q15 : coeff_type_Q15 := (
97.          integer((coeff(0))*real((2**NDEC_FACTOR))), 
98.          integer((coeff(1))*real((2**NDEC_FACTOR))), 
99.          integer((coeff(2))*real((2**NDEC_FACTOR))), 
100.         integer((coeff(3))*real((2**NDEC_FACTOR)))
101.     );
102.
103.     type factor_type_Q2_15 is array(0 to n_taps_op - 1) of std_logic_vector(17 - 1 downto 0);
104.     signal factor_Q2_15 : factor_type_Q2_15;
105.
106.     type factor_type_Q17 is array(0 to n_taps_op - 1) of std_logic_vector(18 - 1 downto 0);
107.     signal factor_Q17 : factor_type_Q17;
108.
109.     signal result_t : signed(M-1 downto 0);
110.
111. -----
112. --Comienzo del comportamiento del la entidad diff_ecuation
113. -----
114. begin
115.
116.     --Se genera el arreglo de 5 registros de entrada de la señal
117.     vector_reg(0) <= s_axis_tdata_i;
118.     gen_array ff : for i in 0 to n_taps - 1 generate
119.         inst_ff_module : ff_module
120.             generic map (
121.                 N => N
122.             )
123.             port map (
124.                 clk_i       => clk_i,
125.                 rst_i       => rst_i,
126.                 enable_t_i  => axi_crt.enable_process,
127.                 data_i      => vector_reg(i),
128.                 data_o      => vector_reg(i + 1)
129.             );
130.     end generate;
131.
132.     factor_Q2_15(0) <= std_logic_vector(to_signed(to_integer(signed(vector_reg(0))) +
133.                                             to_integer(signed(vector_reg(n_taps))),17));
134.     factor_Q2_15(1) <= std_logic_vector(to_signed(to_integer(signed(vector_reg(0 + 1))) +
135.                                             to_integer(signed(vector_reg(n_taps - 1))),17));
136.     factor_Q2_15(2) <= std_logic_vector(to_signed(to_integer(signed(vector_reg(0 + 2))) +
137.                                             to_integer(signed(vector_reg(n_taps - 2))),17));

```

```

135.      factor_Q2_15(3) <= std_logic_vector(resize(signed(vector_reg(3)),17));
136.
137.      --Se genera el arreglo de los multiplicadores y se incluye el truncado a Q1.17
138.      gen_array_mul : for i in 0 to n_taps_op - 1 generate
139.          inst_mul_module : mul_module
140.          port map (
141.              coeff_in => std_logic_vector(to_signed(coeff_Q15(i),17)),
142.              data_in => factor_Q2_15(i),
143.              data_out => factor_Q17(i)
144.          );
145.      end generate;
146.
147.      --Se realiza la suma de los resultados de la multiplicación de las muestras con
148.      --determinado factor
149.      --El resultado es un vector de 23bits con formato Q4.17
150.      sum_process : process(all) is
151.          variable temp : signed(M - 1 downto 0);
152.      begin
153.          temp := (others => '0');
154.          loop_vector : for i in 0 to n_taps_op - 1 loop
155.              temp := temp + signed(factor_Q17(i));
156.          end loop;
157.          result_t <= temp;
158.      end process;
159.
160.      --Manejo de los AXI4 Stream (Slave y Master)
161.
162.
163.      fsm_process : process (clk_i, rst_i)
164.      begin
165.          if(rst_i = '0') then
166.              axi_crt <= reg_ctr_default;
167.          elsif rising_edge(clk_i) then
168.              case(axi_crt.state) is
169.                  when ST_WAIT_DATA =>
170.                      axi_crt.s_axis_tready <= '1';
171.                      axi_crt.m_axis_tvalid <= '0';
172.                      axi_crt.enable_process <= '0';
173.                      if (s_axis_tvalid_i = '1' and s_axis_tready_o = '1') then
174.                          axi_crt.state <= ST_PROCESS_DATA;
175.                          axi_crt.s_axis_tready <= '0';
176.                          axi_crt.enable_process <= '1';
177.                      end if;
178.                  when ST_PROCESS_DATA =>
179.                      axi_crt.s_axis_tready <= '0';
180.                      axi_crt.m_axis_tvalid <= '1';
181.                      axi_crt.enable_process <= '0';
182.                      axi_crt.m_axis_tdata <= std_logic_vector(resize(result_t,M));
183.                      axi_crt.state <= ST_TRANSMIT_DATA;
184.                  when ST_TRANSMIT_DATA =>
185.                      axi_crt.s_axis_tready <= '1';
186.                      axi_crt.m_axis_tvalid <= '0';
187.                      axi_crt.enable_process <= '0';
188.                      if (m_axis_tready_i = '1' and m_axis_tvalid_o = '1') then
189.                          axi_crt.state <= ST_WAIT_DATA;
190.                      end if;
191.                  when others => null;
192.              end case;
193.          end if;
194.      end process;
195.
196.      m_axis_tdata_o <= axi_crt.m_axis_tdata;
197.      m_axis_tvalid_o <= axi_crt.m_axis_tvalid;
198.      s_axis_tready_o <= axi_crt.s_axis_tready;
199.
200.  end Behavioral;

```

```

1. library ieee;
2. use ieee.std_logic_1164.all ;
3. use ieee.numeric_std.all ;
4.
5. entity mul_module is
6.   generic (
7.     N : integer := 17;
8.     M : integer := 18);
9.   Port ( coeff_in : in STD_LOGIC_VECTOR (N - 1 downto 0);
10.          data_in : in STD_LOGIC_VECTOR (N - 1 downto 0);
11.          data_out : out STD_LOGIC_VECTOR (M-1 downto 0));
12. end mul_module;
13.
14. architecture Behavioral of mul_module is
15.
16.   --Señal que contiene el resultado de la multiplicación de la entrada y su factor.
17.   --El resultado tiene una palabra de 34bits con formato Q4.30
18.   signal mul : std_logic_vector(2*N - 1 downto 0);
19.
20. begin
21.
22.   --Se realiza la multiplicación de la entrada con el factor
23.   --los formatos de los dos números son Q1.15, 16bits
24.   mul <= std_logic_vector(to_signed(to_integer(signed(coeff_in))
25.                           * to_integer(signed(data_in)), 2*N));
26.   --Se realiza el truncado del resultado quedando un número con formato Q1.17, 18bits
27.   data_out <= mul(30 downto 13);
28.
29. end Behavioral;

```

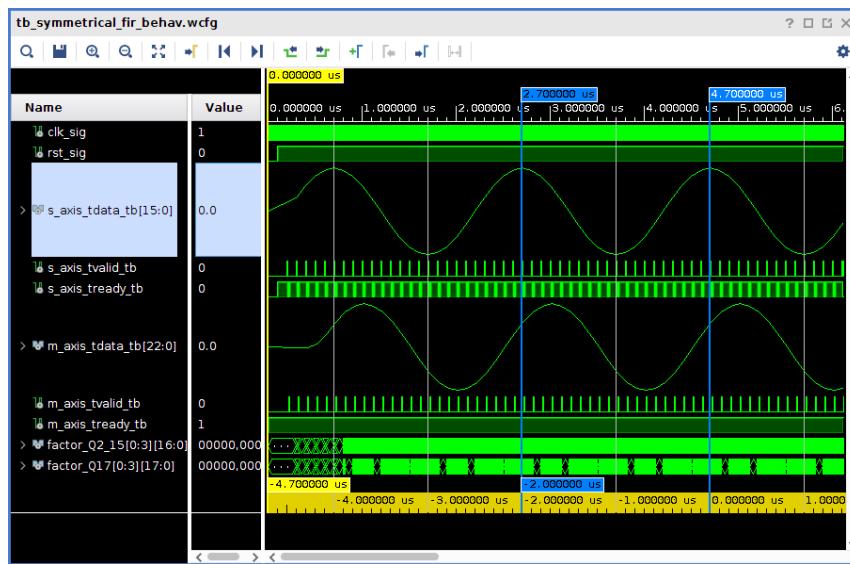
```

1. library ieee ;
2. use ieee.std_logic_1164.all ;
3. use ieee.numeric_std.all ;
4.
5. entity ff_module is
6.   generic (
7.     N : integer := 8
8.   );
9.   Port ( clk_i      : in STD_LOGIC;
10.          rst_i      : in STD_LOGIC;
11.          enable_t_i : in STD_LOGIC;
12.          data_i     : in STD_LOGIC_VECTOR (N-1 downto 0);
13.          data_o     : out STD_LOGIC_VECTOR (N-1 downto 0));
14. end ff_module;
15.
16. architecture Behavioral of ff_module is
17.   signal reg_signal : std_logic_vector(data_o'RANGE);
18. begin
19.
20.   reg_process : process (clk_i, rst_i)
21.   begin
22.     if (rst_i = '0') then
23.       reg_signal <= (others => '0');
24.     elsif (rising_edge(clk_i)) then
25.       if (enable_t_i = '1') then
26.         reg_signal <= data_i;
27.       end if;
28.     end if;
29.   end process reg_process;
30.
31.   data_o <= reg_signal;
32.
33. end Behavioral;

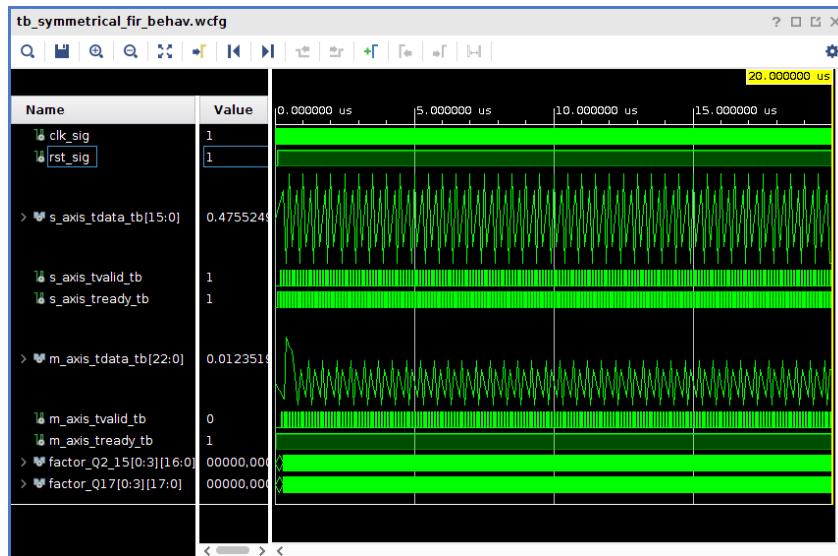
```

## Simulación

Se aprecia en la siguiente figura se excitó el sistema con una señal con frecuencia igual a 0.1 de Nyquist y con una amplitud de 0.5. Se cambió la implementación del filtro teniendo en cuenta la simetría y se realizaron 4 multiplicaciones, al contrario, de las 6 que se hicieron en el punto anterior.



Se excitó el sistema con una señal de frecuencia de 0.8 de la frecuencia de Nyquist y se obtuvo la siguiente respuesta.



Se puede apreciar que los resultados son exactamente los mismos de la anterior implementación.

**Se realizan los puntos 5,6,7 y 8 realizando redondeo en vez de truncado.**

### Código RTL

```

1. library ieee;
2. use ieee.std_logic_1164.all ;
3. use ieee.numeric_std.all ;
4.
5. entity symmetrical_fir is
6.   generic (
7.     --Bits de la palabra de entrada
8.     N : integer := 16;
9.     --Bits de la palabra de salida
10.    M : integer := 23);
11.  Port (
12.    clk_i : in STD_LOGIC;
13.    rst_i : in STD_LOGIC;
14.    s_axis_tdata_i : in STD_LOGIC_VECTOR (N-1 downto 0);
15.    s_axis_tvalid_i : in STD_LOGIC;
16.    s_axis_tready_o : out STD_LOGIC;
17.    m_axis_tdata_o : out STD_LOGIC_VECTOR (M-1 downto 0);
18.    m_axis_tvalid_o : out STD_LOGIC;
19.    m_axis_tready_i : in STD_LOGIC);
20. end symmetrical_fir;
21.
22. architecture Behavioral of symmetrical_fir is
23.
24.   constant n_taps: integer := 6;
25.   --Vector con los datos que se van a ir almacenando
26.   type vector_type is array (integer range <>) of std_logic_vector(s_axis_tdata_i'RANGE);
27.   signal vector_reg : vector_type(0 to n_taps);
28.
29.   component ff_module is
30.     generic (
31.       N : integer := N
32.     );
33.     Port ( clk_i      : in STD_LOGIC;
34.            rst_i      : in STD_LOGIC;
35.            enable_t_i : in STD_LOGIC;
36.            data_i     : in STD_LOGIC_VECTOR (N-1 downto 0);
37.            data_o     : out STD_LOGIC_VECTOR (N-1 downto 0));
38.   end component ff_module;
39.
40.   component mul_module is
41.     generic (
42.       N : integer := 16;
43.       M : integer := 18);
44.     Port ( coeff_in : in STD_LOGIC_VECTOR (N - 1 downto 0);
45.            data_in : in STD_LOGIC_VECTOR (N - 1 downto 0);
46.            data_out : out STD_LOGIC_VECTOR (M - 1 downto 0));
47.   end component mul_module;
48.
49.   --Definición de los estados de la máquina de estados del proceso del encoder.
50.   type state_type is
51.   (
52.     ST_WAIT_DATA,
53.     ST_PROCESS_DATA,
54.     ST_TRANSMIT_DATA
55.   );

```

```

56.
57.    --Generamos el arreglo de registro
58.    type reg_axi_type is record
59.        state           : state_type;
60.        --Señal de Ready del puerto de salida del AXI Stream Slave
61.        s_axis_tready   : std_logic;
62.        --Señales para el manejo del puerto AXIS4-Stream
63.        m_axis_tdata     : std_logic_vector(m_axis_tdata_o'RANGE);
64.        m_axis_tvalid    : std_logic;
65.        enable_process   : std_logic;
66.    end record;
67.
68.    --Valor por defecto de los registros.
69.    constant reg_ctr_default : reg_axi_type := (
70.        state          => ST_WAIT_DATA,
71.        s_axis_tready  => '0',
72.        m_axis_tdata   => (others => '0'),
73.        m_axis_tvalid  => '0',
74.        enable_process => '0'
75.    );
76.
77.    --Registros internos de variables de control
78.    signal axi_crt : reg_axi_type := reg_ctr_default;
79.
80.    type coeff_type is array(0 to n_taps) of real;
81.    constant coeff : coeff_type := (
82.        -0.04895593695235194,
83.        0.06275537458572507,
84.        0.29246401053451393,
85.        0.4226075033374576,
86.        0.29246401053451393,
87.        0.06275537458572507,
88.        -0.04895593695235194);
89.
90.    --constante para los decimales de los factores en este caso es Q1.15
91.    constant NDEC_FACTOR : integer := 15;
92.
93.    --Se realiza una conversión de los coeficientes de los factores de real
94.    --a un entero con formato Q1.15
95.    type coeff_type_Q15 is array(0 to n_taps) of integer;
96.    constant coeff_Q15 : coeff_type_Q15 := (
97.        integer((coeff(0))*real((2**NDEC_FACTOR))),
98.        integer((coeff(1))*real((2**NDEC_FACTOR))),
99.        integer((coeff(2))*real((2**NDEC_FACTOR))),
100.       integer((coeff(3))*real((2**NDEC_FACTOR))),
101.       integer((coeff(4))*real((2**NDEC_FACTOR))),
102.       integer((coeff(5))*real((2**NDEC_FACTOR))),
103.       integer((coeff(6))*real((2**NDEC_FACTOR)))
104.    );
105.
106.   type factor_type_Q16 is array(0 to n_taps) of std_logic_vector(18 - 1 downto 0);
107.   signal factor_Q16 : factor_type_Q16;
108.
109.   signal result_t : signed(M-1 downto 0);
110.
111. -----
112. --Comienzo del comportamiento del la entidad diff_ecuation
113. -----
114. begin
115.
116.    --Se genera el arreglo de 5 registros de entrada de la señal
117.    vector_reg(0) <= s_axis_tdata_i;
118.    gen_array_ff : for i in 0 to n_taps - 1 generate
119.        inst_ff_module : ff_module
120.            generic map (
121.                N => N
122.            )
123.            port map (
124.                clk_i      => clk_i,
125.                rst_i      => rst_i,

```

```

126.           enable_t_i  => axi_crt.enable_process,
127.           data_i      => vector_reg(i),
128.           data_o      => vector_reg(i + 1)
129.       );
130.   end generate;
131.
132.   --Se genera el arreglo de los multiplicadores y se incluye el truncado a Q1.17
133.   gen_array_mul : for i in 0 to n_taps generate
134.     inst_mul_module : mul_module
135.       port map (
136.         coeff_in  => std_logic_vector(to_signed(coeff_Q15(i),N)),
137.         data_in   => vector_reg(i),
138.         data_out  => factor_Q16(i)
139.       );
140.   end generate;
141.   --Se realiza la suma de los resultados de la multiplicación
142.   --de las muestras con determinado factor
143.   --El resultado es un vector de 23bits con formato Q7.16
144.   sum_process : process(all) is
145.     variable temp : signed(M - 1 downto 0);
146.   begin
147.     temp := (others => '0');
148.     loop_vector : for i in 0 to n_taps loop
149.       temp := temp + signed(factor_Q16(i));
150.     end loop;
151.     result_t <= temp;
152.   end process;
153.
154. -----
155. --Manejo de los AXI4 Stream (Slave y Master)
156. -----
157. fsm_process : process (clk_i, rst_i)
158. begin
159.   if(rst_i = '0') then
160.     axi_crt <= reg_ctrl_default;
161.   elsif rising_edge(clk_i) then
162.     case(axi_crt.state) is
163.       when ST_WAIT_DATA =>
164.         axi_crt.s_axis_tready <= '1';
165.         axi_crt.m_axis_tvalid <= '0';
166.         axi_crt.enable_process <= '0';
167.         if (s_axis_tvalid_i = '1' and s_axis_tready_o = '1') then
168.           axi_crt.state <= ST_PROCESS_DATA;
169.           axi_crt.s_axis_tready <= '0';
170.           axi_crt.enable_process <= '1';
171.         end if;
172.       when ST_PROCESS_DATA =>
173.         axi_crt.s_axis_tready <= '0';
174.         axi_crt.m_axis_tvalid <= '1';
175.         axi_crt.enable_process <= '0';
176.         axi_crt.m_axis_tdata <= std_logic_vector(resize(result_t,M));
177.         axi_crt.state <= ST_TRANSMIT_DATA;
178.       when ST_TRANSMIT_DATA =>
179.         axi_crt.s_axis_tready <= '1';
180.         axi_crt.m_axis_tvalid <= '0';
181.         axi_crt.enable_process <= '0';
182.         if (m_axis_tready_i = '1' and m_axis_tvalid_o = '1') then
183.           axi_crt.state <= ST_WAIT_DATA;
184.         end if;
185.       when others => null;
186.     end case;
187.   end if;
188. end process;
189.
190.   m_axis_tdata_o  <= axi_crt.m_axis_tdata;
191.   m_axis_tvalid_o <= axi_crt.m_axis_tvalid;
192.   s_axis_tready_o <= axi_crt.s_axis_tready;
193. end Behavioral;

```

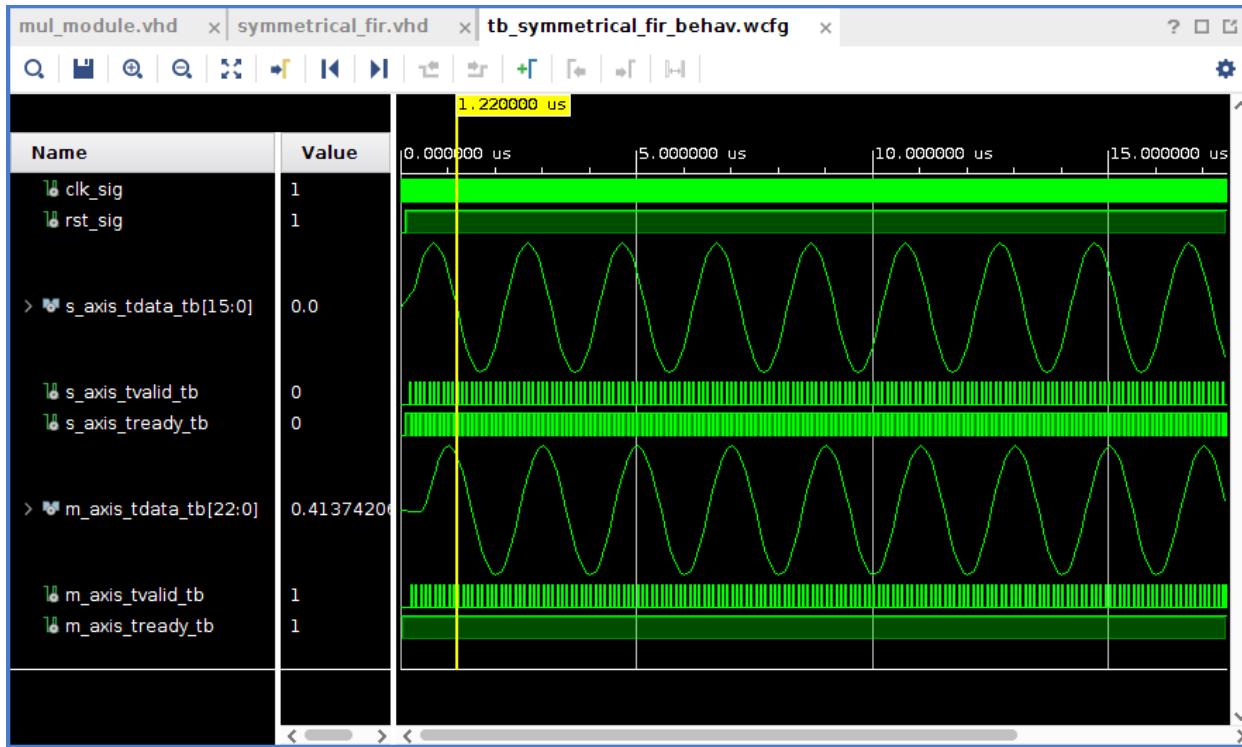
```

1. library ieee;
2. use ieee.std_logic_1164.all ;
3. use ieee.numeric_std.all ;
4.
5. entity mul_module is
6. generic (
7.   N : integer := 16;
8.   M : integer := 18);
9. Port ( coeff_in : in STD_LOGIC_VECTOR (N - 1 downto 0);
10.        data_in : in STD_LOGIC_VECTOR (N - 1 downto 0);
11.        data_out : out STD_LOGIC_VECTOR (M-1 downto 0));
12. end mul_module;
13.
14. architecture Behavioral of mul_module is
15.
16.   component round is
17. generic (
18.   N           : integer      := 32;          --BITS DE ENTRADA
19.   M           : integer      := 18;          --BITS DE SALIDA
20.   ENA_ROUND   : std_logic := '1'           --Saturacion o Overflow
21. );
22. port (
23.   data_i : in std_logic_vector(N-1 downto 0);
24.   data_o : out std_logic_vector(M-1 downto 0)
25. );
26. end component round;
27.
28. --Señal que contiene el resultado de la multiplicación de la entrada y su factor.
29. --EL resultado tiene una palabra de 32 bits con formato Q2.30
30. signal mul : std_logic_vector(2*N - 1 downto 0);
31.
32. begin
33.
34.   --Se realiza la multiplicación de la entrada con el factor
35.   --los formatos de los dos números son Q1.15, 16bits
36.   mul <= std_logic_vector(to_signed(to_integer(signed(coeff_in))
37.                           * to_integer(signed(data_in)), 2*N));
38.   --Se realiza el truncado del resultado quedando un número
39.   --con formato Q2.16, 18bits
40.   saturation_inst : round
41.   generic map (
42.     N    => 32,
43.     M    => 18,
44.     ENA_ROUND => '1'
45.   )
46.   port map (
47.     data_i => mul,
48.     data_o => data_out
49.   );
50. end Behavioral;

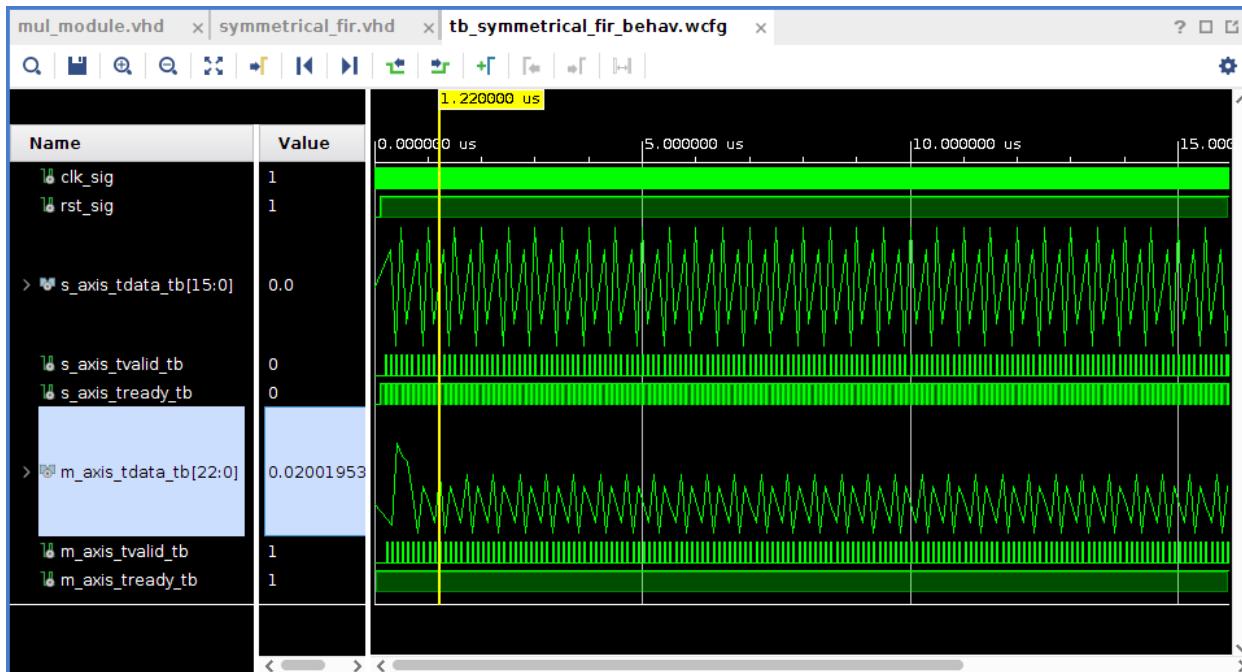
```

## Simulación

Se aprecia en la siguiente figura se excitó el sistema con una señal con frecuencia igual a 0.1 de Nyquist y con una amplitud de 0.5. Se cambió la implementación del filtro realizando redondeo en vez de truncado en las multiplicaciones.



Se excitó el sistema con una señal de frecuencia de 0.8 de la frecuencia de Nyquist y se obtuvo la siguiente respuesta.



Se puede apreciar que los resultados son parecidos exceptuando un poco de precisión decimal. Los dos primeros puntos tienen una salida de formato Q7.16 y cuando se realizó redondeo se obtuvo un formato de Q6.17.

#### Código de verificación para los tres puntos del ejercicio 4

```

1. library ieee ;
2. use ieee.std_logic_1164.all ;
3. use ieee.numeric_std.all ;
4. use std.textio.all;
5. use ieee.std_logic_textio.all;
6.
7. entity tb_symmetrical_fir is
8. end tb_symmetrical_fir;
9.
10. architecture Behavioral of tb_symmetrical_fir is
11.
12.   --Bits de la palabra de entrada
13.   constant N : integer := 16;
14.   --Bits de la palabra de salida
15.   constant M : integer := 23;
16.
17.   constant CLK_PERIOD : time := 10 ns;
18.   signal clk_sig : std_logic := '1';
19.   signal rst_sig : std_logic := '0';
20.
21.   signal s_axis_tdata_tb      : std_logic_vector(N-1 downto 0);
22.   signal s_axis_tvalid_tb    : std_logic;
23.   signal s_axis_tready_tb   : std_logic;
24.   signal m_axis_tdata_tb      : std_logic_vector(M-1 downto 0);
25.   signal m_axis_tvalid_tb    : std_logic;
26.   signal m_axis_tready_tb   : std_logic;
27.
28.   --Entidad del módulo ecuación diferencial
29.   component symmetrical_fir is
30.     generic (
31.       --Bits de la palabra de entrada
32.       N : integer := 16;
33.       --Bits de la palabra de salida
34.       M : integer := 16);
35.     Port (
36.       clk_i          : in STD_LOGIC;
37.       rst_i          : in STD_LOGIC;
38.       s_axis_tdata_i : in STD_LOGIC_VECTOR (N-1 downto 0);
39.       s_axis_tvalid_i : in STD_LOGIC;
40.       s_axis_tready_o : out STD_LOGIC;
41.       m_axis_tdata_o : out STD_LOGIC_VECTOR (M-1 downto 0);
42.       m_axis_tvalid_o : out STD_LOGIC;
43.       m_axis_tready_i : in STD_LOGIC);
44.   end component symmetrical_fir;
45.
46.   constant in_file    : string  := "data_in.txt";
47.   constant out_file   : string  := "data_out.txt";
48.
49.   file r_fptr,w_fptr : text;
50.
51.   --Items para guardar
52.   constant ITEMS_TO_SAVE : integer := 200;
53.
54.   constant C_CLK_PERIOD : real := 10.0e-9; -- NS
55. begin

```

```

56.
57. -----
58. -- Clocks and Reset
59. -----
60.
61. clk_sig <= not clk_sig after CLK_PERIOD/2;
62. rst_sig <= '1'after 100 ns;
63.
64. -----
65. --Estimulos a las señales
66. -----
67. -- Read file process
68. p_read_file : process is
69.     variable fstatus      : file_open_status;
70.     variable file_line   : line;
71.     variable slv_v       : integer;
72. begin
73.     --Por defecto deshabilitamos la transacción
74.     --Abrimos el archivo
75.     file_open(fstatus, r_fptr,in_file, read_mode);
76.     --Asignación por defecto.
77.     s_axis_tdata_tb     <= (others => '0');
78.     s_axis_tvalid_tb    <= '0';
79.     wait until (rst_sig = '1');
80.     --Hacemos un espera en ciclos de reloj
81.     delay_1 : for i in 0 to 10-1 loop
82.         wait until (clk_sig'event and clk_sig = '1');
83.     end loop;
84.     --Recorremos el archivo
85.     loop_file : while not endfile(r_fptr) loop
86.         readline(r_fptr,file_line);
87.         read(file_line,slv_v);
88.         report "Valor leido: " & integer'image(slv_v);
89.         --Generaremos la señal AXI con el pulso en '1' cuando recibamos el tready.
90.         if (s_axis_tready_tb = '1') then
91.             s_axis_tdata_tb <= std_logic_vector(to_signed(slv_v, N));
92.             s_axis_tvalid_tb <= '1';
93.         end if;
94.         --Esperamos 10 ciclos de relog y se obtiene el periodo de la frecuencia de muestreo
95.         wait until (clk_sig'event and clk_sig = '1');
96.         s_axis_tvalid_tb <= '0';
97.         delay_2 : for i in 0 to 9-1 loop
98.             wait until (clk_sig'event and clk_sig = '1');
99.         end loop;
100.        end loop ; -- loop_file
101.        s_axis_tdata_tb     <= (others => '0');
102.        s_axis_tvalid_tb    <= '0';
103.        report "Fin lectura del archivo";
104.        file_close(r_fptr);
105.        wait;
106.    end process; -- p_read_file
107.
108.    p_write_file : process is
109.        variable fstatus      : file_open_status;
110.        variable file_line   : line;
111.        variable v_int       : integer;
112.        variable v_std_lv   : std_logic_vector((m_axis_tdata_tb'LENGTH - 1) downto 0);
113. begin
114.     -- Para este caso vamos a aceptar todos los datos que salgan.
115.     m_axis_tready_tb <= '1';
116.     --Abrimos el archivo
117.     file_open(fstatus, w_fptr,out_file,write_mode);
118.     wait until (rst_sig = '1');
119.
120.     --Vamos a escribir solo 200.
121.     write_file : for i in 0 to ITEMS_TO_SAVE loop
122.         wait until (m_axis_tvalid_tb = '1');
123.         v_int    := to_integer(signed(m_axis_tdata_tb));
124.         v_std_lv := m_axis_tdata_tb;
125.         write(file_line,v_int);

```

```

126.           write(file_line,v_std_lv,right,40);
127.           writeline(w_fptr, file_line);
128.           report "Valor ESCRITO: " & integer'image(v_int);
129.           report "Indice: " & integer'image(i);
130.       end loop;
131.       report "Fin escritura del archivo";
132.       file_close(w_fptr);
133.       wait;
134.   end process; -- p_write_file
135.
136. -----
137. --Instancia que se pone a prueba en el testbench
138. -----
139.
140.     inst_symmetrical_fir : symmetrical_fir
141.         generic map (
142.             N => 16,
143.             M => 23
144.         )
145.         port map (
146.             clk_i          => clk_sig,
147.             rst_i          => rst_sig,
148.             s_axis_tdata_i => s_axis_tdata_tb,
149.             s_axis_tvalid_i => s_axis_tvalid_tb,
150.             s_axis_tready_o => s_axis_tready_tb,
151.             m_axis_tdata_o  => m_axis_tdata_tb,
152.             m_axis_tvalid_o => m_axis_tvalid_tb,
153.             m_axis_tready_i => m_axis_tready_tb
154.         );
155.
156. end Behavioral;

```