Universidad de Buenos Aires
Facultad De Ingeniería
Año 2023 – 2.^{do} Cuatrimestre

(86.41) Sistemas digitales

Trabajo práctico N.º 2

Docentes a cargo:
Alvarez, Nicolás
Alpago, Octavio

Integrante	Padrón	Correo electrónico
Lützelschwab, Nahila	100686	nlützelschwab@fi.uba.ar

1. Introducción

El presente trabajo práctico tiene como objetivo implementar el algoritmo CORDIC en modo rotación y vectorización aplicando el lenguaje de descripción de hardware VHDL en dos arquitecturas; enrollada y desenrollada.

2. Desarrollo

Se implementó el algoritmo CORDIC en modo rotación y vectorización para la rotación de vectores en el plano xy. Se emplea para realizar eficientemente operaciones trigonométricas, como rotaciones y cálculos de magnitudes de vectores. Este enfoque se basa en iteraciones y operaciones de desplazamiento bit a bit.

En el modo de rotación, se rota un vector en un ángulo especificado. El proceso inicia con la inicialización del acumulador angular con el ángulo de rotación deseado. En cada iteración, la decisión de rotación se lleva a cabo de tal manera de disminuir el ángulo residual en el acumulador angular utilizando su signo.

En cambio, el modo vectorización se rota un vector hacia el eje de coordenadas x, guardando los ángulos requeridos para lograrlo. Se busca minimizar la componente y del vector residual, la dirección de rotación se decide por el signo de la componente y residual.

$$x_{i+1} = x_i - y_i.d_i.2^{-i}$$
$$y_{i+1} = y_i + x_i.d_i.2^{-i}$$
$$z_{i+1} = z_i - d_i.tg^{-1}(2^{-i})$$

Figura 1

Donde para el modo rotación $d_i = -1$ si $z_i < 0$, en otro caso +1. Dando como resultado las siguientes ecuaciones para el modo rotación:

$$x_n = A_n(x_0.cos(z_0) - y_0.sen(z_0))$$
$$y_n = A_n(y_0.cos(z_0) + x_0.sen(z_0))$$
$$z_n = 0$$

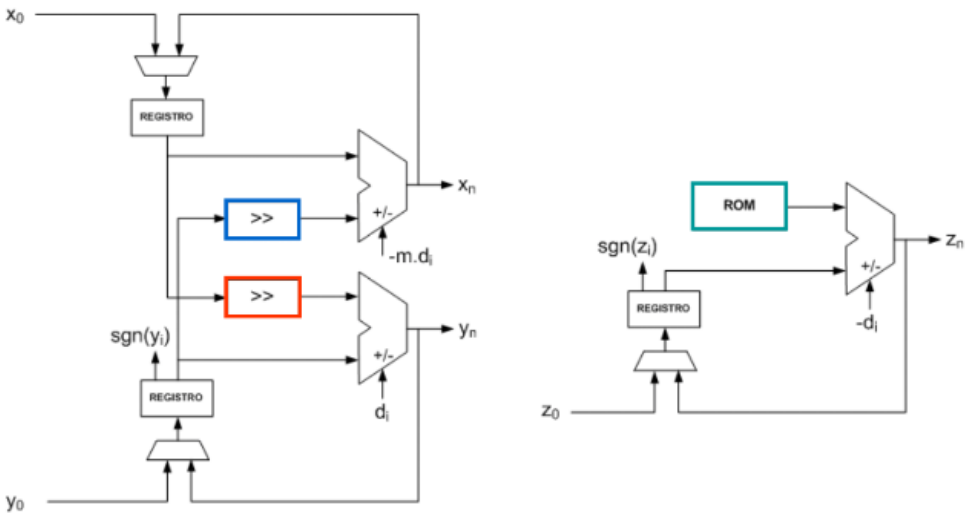
Figura 2

En cambio, para el modo vectorización, $d_i = +1$ si $y_i < 0$, en otro caso -1. Dando como resultado las siguientes ecuaciones para el modo vectorización:

$$x_n = A_n(x_0.cos(z_0) - y_0.sen(z_0))$$
$$y_n = A_n(y_0.cos(z_0) + x_0.sen(z_0))$$
$$z_n = 0$$

Figura 3

A continuación se muestran los diagramas de las arquitecturas enrollada y desenrollada con pipelining que se a implementaron en este trabajo:



$$x_{i+1} = x_i - y_i.d_i.2^{-i}$$
$$y_{i+1} = y_i + x_i.d_i.2^{-i}$$
$$z_{i+1} = z_i - d_i\,tg^{-1}(2^{-i})$$

Figura 4: Arquitectura enrollada

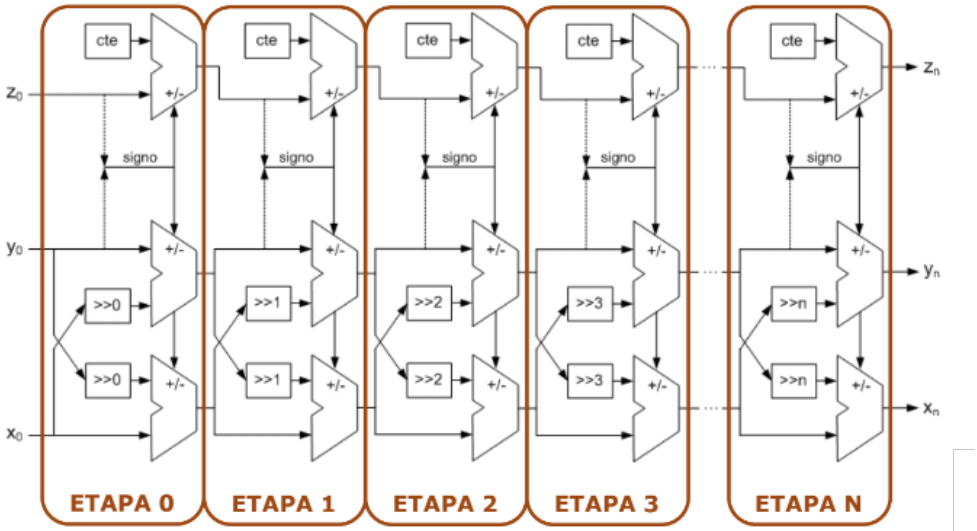


Figura 5: Arquitectura desenrollada con pipelining

El bloque pre-cordic es una lógica combinacional que se encarga de negar las coordenadas x e y originales para poder rotar aquellos casos en los que el ángulo de rotación de entrada sea mayor a 90º o menor a –90º.

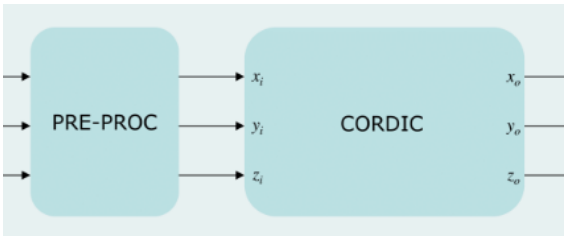


Figura 6

2.1. Implementación en VHDL

La implementación en VHDL es la siguiente:

2.2. Suma/resta

```
-- Nahila Lutzelschwab - TP3 - padron: 100686 - addition/subtracion

-- Declaration of common library and use
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity add_subtract is
    generic(N : natural := 17);
    port(
        add1_sub0 : in std_logic;

        x     : in std_logic_vector(N-1 downto 0);
        y     : in std_logic_vector(N-1 downto 0);
        z           : out std_logic_vector(N-1 downto 0)
    );
end add_subtract;

architecture behavioral of add_subtract is

begin
    z <= std_logic_vector(signed(x) + signed(y)) when add1_sub0 = '1' else
         std_logic_vector(signed(x) - signed(y));

end behavioral;
```

2.3. ROM - LUT

```
-- Nahila Lutzelschwab - TP3 - padron: 100686 - beta-LUT

-- Declaration of common library and use
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
use IEEE.math_real.all;

entity beta_lut is
    generic (
        DATA_W : natural := 16;
            ADD_W  : natural := 17
    );
    port (
        address : in std_logic_vector(ADD_W-1 downto 0);
        data_out : out std_logic_vector(DATA_W-1 downto 0)
    );
end entity beta_lut;

architecture behavioral of beta_lut is

    type lut_type is array (natural range <>) of std_logic_vector(DATA_W-1 downto 0);

    function arctan_function(constant i : natural) return std_logic_vector is
    begin
        return std_logic_vector(to_unsigned(integer(round( (arctan(real(2)**real(-1*i)) / arctan(real(1))) * real(2**(DA
     end;

    signal lut : lut_type(0 to 2**DIREC-1);

begin

    lut_table : for i in 0 to DATA_W-1 generate
        lut(i) <= arctan_function(i);
    end generate lut_table;

    -- In case the index adress exceeds the limit of the LUT the output is NULL
    data_out <= (others => '0') when unsigned(address) > to_unsigned(DATA_W-1,ADD_W) else
```

```
            lut(to_integer(unsigned(address)));

end behavioral;
```

2.4. Registros

```
-- Nahila Lutzelschwab - TP3 - padron: 100686 - Register for pipeline

-- Declaration of common library and use
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity registers is
    generic(
        N : natural := 17
    );
    port (
        clk     : in std_logic;
        rst     : in std_logic;
        input   : in std_logic_vector(N-1 downto 0);
        output  : out std_logic_vector(N-1 downto 0)
    );
end registers;

architecture behavioral of registers is
    signal reg : std_logic_vector(N-1 downto 0);
begin
    process (clk, rst)
    begin
        if rst = '1' then
            reg <= (others => '0');

        elsif clk = '1' and clk'event then
            reg <= input;
        end if;
    end process;

    output <= reg;
end behavioral;
```

2.5. Pre-cordic

```
-- Nahila Lutzelschwab - TP3 - padron: 100686 - pre-cordic

-- Declaration of common library and use
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity pre_cordic is
    generic(
        N : natural := 17
    );
    port(
        rot0_vec1  : in std_logic;

        x_i        : in std_logic_vector(N-1 downto 0);
        y_i        : in std_logic_vector(N-1 downto 0);
        z_i        : in std_logic_vector(N-1 downto 0);

        x_o        : out std_logic_vector(N-1 downto 0);
        y_o        : out std_logic_vector(N-1 downto 0);
        z_o        : out std_logic_vector(N-1 downto 0)
    );

end pre_cordic;
```

```
25
26  architecture behavioral of pre_cordic is
27      signal   x_aux : std_logic_vector(N-1 downto 0);
28      signal   y_aux : std_logic_vector(N-1 downto 0);
29      signal   z_aux : std_logic_vector(N-1 downto 0);
30      signal MSB_inverted : std_logic;
31      signal z_MSB_inverted : std_logic_vector(N-1 downto 0);
32
33
34  begin
35
36      x_aux <= x_i;
37      y_aux <= y_i;
38      z_aux <= z_i;
39
40      process (rot0_vec1, x_aux, y_aux, z_aux)
41      begin
42          if rot0_vec1 = '0' then
43              if z_aux(N-2) = '1' then
44                  x_o <= std_logic_vector(signed(not x_aux) + 1);
45                  y_o <= std_logic_vector(signed(not y_aux) + 1);
46                  MSB_inverted <= not z_aux(N-1);
47                  z_MSB_inverted <= MSB_inverted & z_aux(N-2 downto 0);
48                  z_o <= z_MSB_inverted;
49              else
50                  x_o <= x_aux;
51                  y_o <= y_aux;
52                  z_o <= z_aux;
53              end if;
54          else
55              if x_aux(N-1) = '1' then
56                  x_o <= std_logic_vector(signed(not x_aux) + 1);
57                  y_o <= std_logic_vector(signed(not y_aux) + 1);
58                  MSB_inverted <= not z_aux(N-1);
59                  z_MSB_inverted <= MSB_inverted & z_aux(N-2 downto 0);
60                  z_o <= z_MSB_inverted;
61              else
62                  x_o <= x_aux;
63                  y_o <= y_aux;
64                  z_o <= z_aux;
65              end if;
66          end if;
67      end process;
68
69
70  end behavioral;
```

2.6. Entidad cordic

```
1   -- Nahila Lutzelschwab - TP3 - padron: 100686 - cordic-entity (same for rolled and unrolled)
2
3   -- Declaration of common library and use
4   library IEEE;
5   use IEEE.std_logic_1164.all;
6   use IEEE.numeric_std.all;
7   use IEEE.math_real.all;
8
9   entity cordic_entity is
10      generic(
11          N : natural := 17;
12          -- fp_amount : natural := natural(ceil(log2(real(N-1))));
13          fp_amount : natural := 4
14      );
15      port(
16          iter_num : in unsigned(fp_amount-1 downto 0);
17          rot0_vec1: in std_logic;
18              atan_2mi : in std_logic_vector(N-3 downto 0); -- arctan(2^-i)
19
```

```
        -- Inputs
        x_i : in std_logic_vector(N-1 downto 0);
        y_i : in std_logic_vector(N-1 downto 0);
        z_i : in std_logic_vector(N-1 downto 0);

       -- Outputs
       -- x_i+1, y_i+1, z_i+1
        x_ip1 : out std_logic_vector(N-1 downto 0);
        y_ip1 : out std_logic_vector(N-1 downto 0);
        z_ip1 : out std_logic_vector(N-1 downto 0)
    );
end cordic_entity;

architecture behavioral of cordic_entity is

     signal beta_i: std_logic_vector(N-1 downto 0);
     signal x_aux : std_logic_vector(N-1 downto 0);
     signal y_aux: std_logic_vector(N-1 downto 0);
     signal z_aux : std_logic_vector(N-1 downto 0);

    -- Shifted input
    signal x_shifted : std_logic_vector(N-1 downto 0);
    signal y_shifted : std_logic_vector(N-1 downto 0);

    -- Outputs
    signal x_o: std_logic_vector(N-1 downto 0);
    signal y_o : std_logic_vector(N-1 downto 0);
    signal z_o : std_logic_vector(N-1 downto 0);

    -- Selection line
    signal di: std_logic;
    signal not_di: std_logic;


begin
    x_aux <= x_i;
    y_aux <= y_i;
    z_aux <= z_i;

    x_shifted <= std_logic_vector(shift_right(signed(x_aux), to_integer(iter_num)));
    y_shifted <= std_logic_vector(shift_right(signed(y_aux), to_integer(iter_num)));

    di <= z_i(N-1) when rot0_vec1 = '0' else not(y_i(N-1));

    not_di <= not(di);

    adder_subtractor_x: entity work.add_subtract(behavioral)
    generic map(N => N)
    port map(
            add1_sub0 => di,
        x => x_aux,
        y => y_shifted,
        z => x_o
    );

    adder_subtractor_y: entity work.add_subtract(behavioral)
    generic map(N => N)
    port map(
            add1_sub0 => not_di,
        x => y_aux,
        y => x_shifted,
        z => y_o
    );

    beta_i <= "00" & atan_2mi;

    adder_subtractor_z: entity work.add_subtract(behavioral)
    generic map(N => N)
    port map(
```

```
            add1_sub0 => di,
        x => z_aux,
        y => beta_i,
        z => z_o
    );

    x_ip1 <= x_o;
    y_ip1 <= y_o;
    z_ip1 <= z_o;

end behavioral;
```

2.7. Cordic enrollado

```
-- Nahila Lutzelschwab - TP3 - padron: 100686 - cordic-rolled

-- Declaration of common library and use
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
use IEEE.math_real.all;

entity cordic_rolled is
    generic(
        N : natural := 17
    );
    port(
        rst: in std_logic;
    clk: in std_logic;
    req: in std_logic;
    ack: out std_logic;
    rot0_vec1: in std_logic;

    x_0 : in std_logic_vector(N-1 downto 0);
    y_0 : in std_logic_vector(N-1 downto 0);
    z_0 : in std_logic_vector(N-1 downto 0);

    x_ip1_o : out std_logic_vector(N-1 downto 0);
    y_ip1_o : out std_logic_vector(N-1 downto 0);
    z_ip1_o : out std_logic_vector(N-1 downto 0)
    );
end cordic_rolled;

architecture behavioral of cordic_rolled is

    constant counter_amount : natural := natural(ceil(log2(real(N))));

    signal counter : unsigned(counter_amount-1 downto 0);

    signal x_i : std_logic_vector(N-1 downto 0);
    signal y_i : std_logic_vector(N-1 downto 0);
    signal z_i : std_logic_vector(N-1 downto 0);

    signal x_ip1 : std_logic_vector(N-1 downto 0);
    signal y_ip1 : std_logic_vector(N-1 downto 0);
    signal z_ip1 : std_logic_vector(N-1 downto 0);

    signal beta_i : std_logic_vector(N-3 downto 0);
    signal ack_aux : std_logic;

begin

    LUT:entity work.beta_lut(behavioral)
        generic map(
            DATA_W => N-2,
            ADD_W  => counter_amount
        )
        port map(
```

```
            address => std_logic_vector(counter),
            data_out => beta_i
        );


    CORDIC_ENTITY: entity work.cordic_entity(behavioral)
    generic map(
        N => N,
        fp_amount => counter_amount
    )
    port map(
        iter_num => counter,
        rot0_vec1 => rot0_vec1,
            atan_2mi => beta_i,

        x_i => x_i,
        y_i => y_i,
        z_i => z_i,

        x_ip1 => x_ip1,
        y_ip1 => y_ip1,
        z_ip1 => z_ip1
    );

    process(clk,rst)
    begin
        if rst = '1' then
            counter <= (others => '0');
        elsif clk'event and clk = '1' then
            if req = '1' then
                counter <= (others => '0');
            elsif counter /= (N-1) then
                counter <= counter + 1;
            end if;
        end if;
    end process;

    ack_aux <= '1' when  counter = (N-1) else '0';

    ack <= ack_aux;

    process(clk,rst)
    begin
        if rst = '1' then
            x_i <= (others => '0');
            y_i <= (others => '0');
            z_i <= (others => '0');
        elsif clk'event and clk = '1' then
            if req = '1' then
                x_i <= x_0;
                y_i <= y_0;
                z_i <= z_0;
            else
                x_i <= x_ip1;
                y_i <= y_ip1;
                z_i <= z_ip1;
            end if;
        end if;
    end process;

    x_ip1_o <= x_ip1 when ack_aux = '1' else (others => '0');
    y_ip1_o <= y_ip1 when ack_aux = '1' else (others => '0');
    z_ip1_o <= z_ip1 when ack_aux = '1' else (others => '0');

end behavioral;
```

2.8. Cordic desenrollado con pipelining

```
-- Nahila Lutzelschwab - TP3 - padron: 100686 - cordic-unrolled

-- Declaration of common library and use
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
use IEEE.math_real.all;

entity cordic_unrolled is
    generic(
        N : natural := 17
    );
    port(
            rst: in std_logic;
        clk: in std_logic;
        req: in std_logic;
        ack: out std_logic;
        rot0_vec1: in std_logic;

        x_i : in std_logic_vector(N-1 downto 0);
        y_i : in std_logic_vector(N-1 downto 0);
        z_i : in std_logic_vector(N-1 downto 0);

        x_ip1_o : out std_logic_vector(N-1 downto 0);
        y_ip1_o : out std_logic_vector(N-1 downto 0);
        z_ip1_o : out std_logic_vector(N-1 downto 0)
    );
end cordic_unrolled;

architecture behavioral of cordic_unrolled is

    type matrix_type is array (natural range <>) of std_logic_vector(N-1 downto 0);

    constant counter_amount : natural := natural(ceil(log2(real(N))));

    signal count : unsigned(counter_amount-1 downto 0) := (others => '0');
    signal x_vect : matrix_type(N downto 0);
    signal y_vect : matrix_type(N downto 0);
    signal z_vect : matrix_type(N downto 0);

    signal x_reg : matrix_type(N-1 downto 0);
    signal y_reg : matrix_type(N-1 downto 0);
    signal z_reg : matrix_type(N-1 downto 0);

    signal ack_aux : std_logic;
    signal beta_i : std_logic_vector(N-1 downto 0);


begin

    ROM: entity work.beta_lut(behavioral)
        generic map(
            DATA_W => N-2,
            ADD_W  => counter_amount
        )
        port map(
            address => std_logic_vector(count),
            data_out => beta_i
        );

    CORDIC_UNROLLED: for i in 0 to N-1 generate

        CORDIC_ENTITY: entity work.cordic_entity(behavioral)
        generic map(
            N => N,
            fp_amount => counter_amount
        )
```

```
        port map(
            iter_num => count,
            rot0_vec1 => rot0_vec1,
            atan_2mi => beta_i,

            x_i => x_vect(i),
            y_i => y_vect(i),
            z_i => z_vect(i),

            x_ip1 => x_reg(i),
            y_ip1 => y_reg(i),
            z_ip1 => z_reg(i)
        );

    end generate CORDIC_UNROLLED;

    REGISTER_X: for i in 0 to N-1 generate
        REG_X: entity work.registers(behavioral)
        generic map(
            N => N
        )
        port map(
            clk    => clk,
            rst    => rst,
        input  => x_reg(i),
        output => x_vect(i+1)
        );
    end generate REGISTER_X;

    REGISTER_Y: for i in 0 to N-1 generate
        REG_Y: entity work.registers(behavioral)
        generic map(
            N => N
        )
        port map(
            clk    => clk,
            rst    => rst,
        input  => y_reg(i),
        output => y_vect(i+1)
        );
    end generate REGISTER_Y;

    REGISTER_Z: for i in 0 to N-1 generate
        REG_Z: entity work.registers(behavioral)
        generic map(
            N => N
        )
        port map(
            clk    => clk,
            rst    => rst,
        input  => z_reg(i),
        output => z_vect(i+1)
        );
    end generate REGISTER_Z;

    x_vect(0) <= x_i;
    y_vect(0) <= y_i;
    z_vect(0) <= z_i;
    x_ip1_o <= x_reg(N-1);
    y_ip1_o <= y_reg(N-1);
    z_ip1_o <= z_reg(N-1);

    process(clk,rst)
    begin
        if rst = '1' then
            count <= (others => '0');
        elsif clk'event and clk = '1' then
            if req = '1' then
                count <= (others => '0');
```

```
            elsif count /= N-1 then
                count <= count + 1;
            end if;
        end if;
    end process;

    ack_aux <= '1' when  count = N-1 else
               '0';

    ack <= ack_aux;


end behavioral;
```

2.9. Cordic - enrollado/desenrollado

```
-- Nahila Lutzelschwab - TP2 - padron: 100686 - cordic

-- Declaration of common library and use
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
use IEEE.math_real.all;

entity cordic is
    generic(
        N : natural := 18
    );
    port(
        rst             : in std_logic;
        clk             : in std_logic;
        req             : in std_logic;
        ack             : out std_logic;
        rot0_vec1     : in std_logic;

        x_i             : in std_logic_vector(N-1 downto 0);
        y_i             : in std_logic_vector(N-1 downto 0);
        z_i             : in std_logic_vector(N-1 downto 0);

        x_o             : out std_logic_vector(N-1 downto 0);
        y_o             : out std_logic_vector(N-1 downto 0);
        z_o             : out std_logic_vector(N-1 downto 0)

    );
end cordic;


architecture behavioral_rolled of cordic is

    -- Pre-CORDIC inputs
    signal x_i_precordic : std_logic_vector(N-1 downto 0);
    signal y_i_precordic : std_logic_vector(N-1 downto 0);
    signal z_i_precordic : std_logic_vector(N-1 downto 0);

    -- Pre-CORDIC outputs
    signal x_o_precordic : std_logic_vector(N-1 downto 0);
    signal y_o_precordic : std_logic_vector(N-1 downto 0);
    signal z_o_precordic : std_logic_vector(N-1 downto 0);

    -- Unrolloed CORDIC inputs and outputs
    signal x_i_cordic : std_logic_vector(N-1 downto 0);
    signal y_i_cordic : std_logic_vector(N-1 downto 0);
    signal z_i_cordic : std_logic_vector(N-1 downto 0);

    signal ack_aux : std_logic;


```

```
52  begin
53
54      x_i_precordic <= x_i;
55      y_i_precordic <= y_i;
56      z_i_precordic <= z_i;
57
58      ack <= ack_aux;
59
60      PRE_CORDIC: entity work.pre_cordic(behavioral)
61      generic map(N => N)
62      port map(
63          rot0_vec1 => rot0_vec1,
64
65          x_i => x_i_precordic,
66          y_i => y_i_precordic,
67          z_i => z_i_precordic,
68
69          x_o => x_o_precordic,
70          y_o => y_o_precordic,
71          z_o => z_o_precordic
72      );
73
74      CORDIC_ROLLED: entity work.cordic_rolled(behavioral)
75      generic map(N => N)
76      port map(
77          rst       => rst,
78          clk       => clk,
79          req       => req,
80          ack       => ack_aux,
81          rot0_vec1 => rot0_vec1,
82
83          x_0 => x_o_precordic,
84          y_0 => y_o_precordic,
85          z_0 => z_o_precordic,
86
87          x_ip1_o => x_o,
88          y_ip1_o => y_o,
89          z_ip1_o => z_o
90      );
91
92  end behavioral_rolled;
93
94
95  architecture behavioral_unrolled of cordic is
96
97      -- Entradas y salidas de Pre - Cordic
98      signal x_i_precordic : std_logic_vector(N-1 downto 0);
99      signal y_i_precordic : std_logic_vector(N-1 downto 0);
100     signal z_i_precordic : std_logic_vector(N-1 downto 0);
101
102     signal x_o_precordic : std_logic_vector(N-1 downto 0);
103     signal y_o_precordic : std_logic_vector(N-1 downto 0);
104     signal z_o_precordic : std_logic_vector(N-1 downto 0);
105
106     -- Entradas y salidas de Cordic_iterativo
107     signal x_i_cordic : std_logic_vector(N-1 downto 0);
108     signal y_i_cordic : std_logic_vector(N-1 downto 0);
109     signal z_i_cordic : std_logic_vector(N-1 downto 0);
110
111     signal ack_aux : std_logic;
112
113
114
115 begin
116
117     x_i_precordic <= x_i;
118     y_i_precordic <= y_i;
119     z_i_precordic <= z_i;
120     ack <= ack_aux;
```

```
121
122
123     PRE_CORDIC: entity work.pre_cordic(behavioral)
124     generic map(N => N)
125     port map(
126         rot0_vec1 => rot0_vec1,
127
128         x_i       => x_i_precordic,
129         y_i       => y_i_precordic,
130         z_i       => z_i_precordic,
131
132         x_o       => x_o_precordic,
133         y_o       => y_o_precordic,
134         z_o       => z_o_precordic
135     );
136
137     CORDIC_UNROLLED: entity work.cordic_unrolled(behavioral)
138     generic map(
139         N => N
140     )
141     port map(
142             rst => rst,
143         clk => clk,
144         req => req,
145         ack => ack_aux,
146         rot0_vec1 => rot0_vec1,
147
148         x_i => x_o_precordic,
149         y_i => y_o_precordic,
150         z_i => z_o_precordic,
151
152         x_ip1_o => x_o,
153         y_ip1_o => y_o,
154         z_ip1_o => z_o
155     );
156
157 end behavioral_unrolled;
```

2.10. Cordic - testbench

```
1   -- Nahila Lutzelschwab - TP3 - padron: 100686 - cordic testbench (rolled and unrolled)
2
3   -- Declaration of common library and use
4   library IEEE;
5   use IEEE.std_logic_1164.all;
6   use IEEE.numeric_std.all;
7   use IEEE.math_real.all;
8   use std.textio.all;
9
10  entity cordic_tb is
11  end entity cordic_tb;
12
13  architecture cordic_tb_arch of cordic_tb is
14
15      constant WORD_SIZE  : natural := 17;
16      constant FILE_PATH  : string := "datos.txt";
17
18      signal rst : std_logic := '1';  -- reset
19      signal clk : std_logic := '0';  -- clock
20      signal req : std_logic := '0';  --request
21      signal ack : std_logic;         --acknowledge
22      signal rot0_vec1 : std_logic := '0';  -- indicator for rotation or vectorization
23
24      signal x_file : std_logic_vector(WORD_SIZE-1 downto 0):= (others => '0');
25      signal y_file : std_logic_vector(WORD_SIZE-1 downto 0):= (others => '0');
26      signal z_file : std_logic_vector(WORD_SIZE-1 downto 0):= (others => '0');
27
28      signal x_out : std_logic_vector(WORD_SIZE-1 downto 0);
```

```
    signal y_out : std_logic_vector(WORD_SIZE-1 downto 0);
    signal z_out : std_logic_vector(WORD_SIZE-1 downto 0);

    signal cycles   : integer := 0;

    file data: text open read_mode is FILE_PATH;


begin

    clk <= not clk after 10 us;
    -- rst <= '0', '1' after 1 ns, '0' after 20 ns;
    rst <= '0' after 2 us;

    Test_Sequence: process

        variable l   : line;
        variable ch  : character := ' ';
        variable aux : integer;
        -- variable z_file: integer;
        -- variable ANG_RAD: real;

    begin

        while not(endfile(data)) loop
            wait until rising_edge(clk);
            cycles <= cycles + 1;
            -- A line is read from the test values file
            readline(data, l);
            -- An integer is extracted from the line
            read(l, aux);
            -- The value of the X coordinate is loaded
            x_file <= std_logic_vector(to_signed(aux, WORD_SIZE));
            -- The space character is read
            read(l, ch);
            -- Another integer is read from the line
            read(l, aux);
            -- The value of the Y coordinate is loaded
            y_file <= std_logic_vector(to_signed(aux, WORD_SIZE));
            -- Another space character is read
            read(l, ch);
            -- Another integer is read
            read(l, aux);
            -- The value of the angle to rotate is loaded (in degrees)
            --z_file_ := aux;

            -- Operating with the angle to rotate
            -- ANG_RAD := (real(z_file)*MATH_PI)/real(180); -- radians


            -- The corresponding value of the angle to rotate is loaded
            z_file <= std_logic_vector(to_signed(aux,WORD_SIZE));

            -- req <= '1';
            -- wait until rising_edge(clk);
            -- req <= '0';
            -- wait until ack = '1';
            -- wait until rising_edge(clk);
        end loop;

        file_close(data); -- The file is closed

        -- The simulation is aborted (end of file)
        assert false report
             "Fin de la simulacion" severity failure;

    end process Test_Sequence;

    -- DUT: entity work.cordic(behavioral_rolled)
```

```
    DUT: entity work.cordic(behavioral_unrolled)
    generic map(
        N => WORD_SIZE
    )
    port map(
        rst => rst,
        clk => clk,
        req => req,
            ack => ack,
        rot0_vec1 => rot0_vec1,

        x_i => x_file,
        y_i => y_file,
        z_i => z_file,

        x_o => x_out,
        y_o => y_out,
        z_o => z_out

    );

end architecture cordic_tb_arch;
```

3. Simulación

A continuación se muestran los resultados de simulación donde se utilizaron como valores de prueba $x_{in} = 0{,}25$, $y_{in} = 0,4088592529$ y $z_{in}(grados) = 35$.

x_in	0,25	8192
y_in	0,681	22315,008
z_in	35	25486

x_out	-10028,722
y_out	37838,978
z_out	-0,002
z_out(rad)	-0,114591559

Figura 7

x_in	0,25	8192
y_in	0,681	22315,008
z_in	35	25486

x_out	39145,415
y_out	-1,325
z_out	-54723,102
z_out(rad)	-955,0983068

Figura 8

3.1. Arquitectura enrollada

3.1.1. Modo rotación

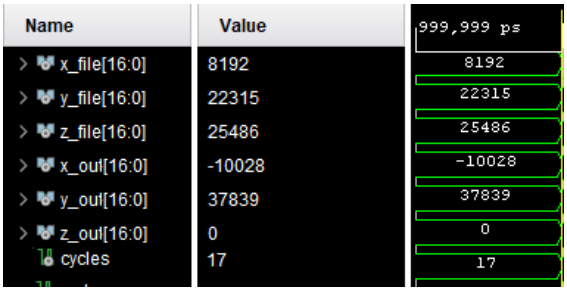


Figura 9

3.1.2. Modo vectorización

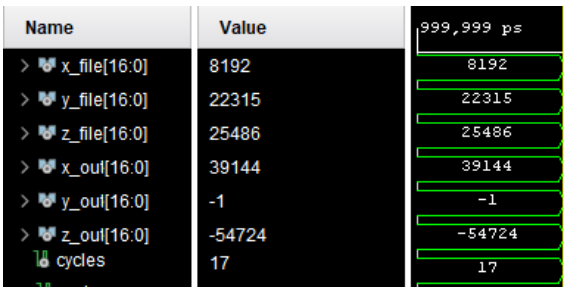


Figura 10

3.2. Arquitectura desenrollada

3.2.1. Modo rotación

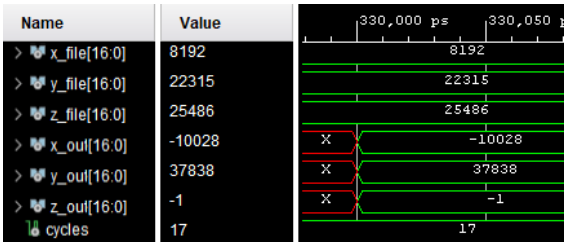


Figura 11

3.2.2. Modo vectorización

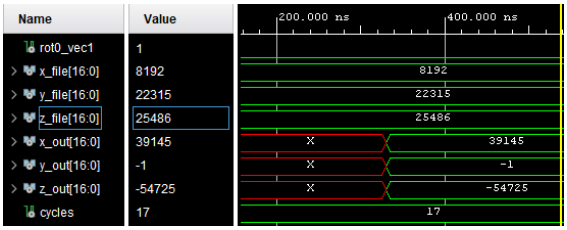


Figura 12

3.3. Sintesis

Se realizó la síntesis sobre el dispositivo xc7a15tftg256-1 mediante el software Vivado.

3.4. Arquitectura enrollada

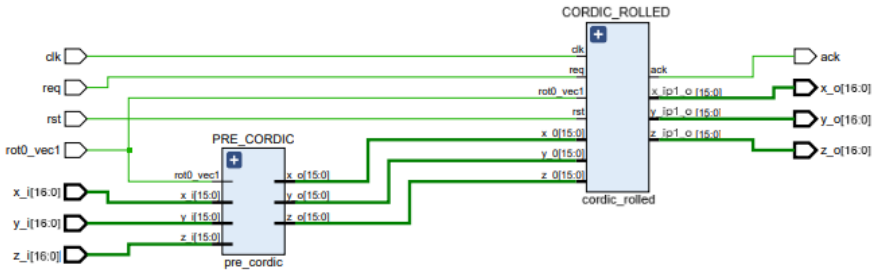


Figura 13

Resource	Utilization	Available	Utilization %
LUT	251	10400	2.41
FF	52	20800	0.25
IO	101	170	59.41

Figura 14

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	inf	Worst Hold Slack (WHS):	inf	Worst Pulse Width Slack (WPWS):	NA
Total Negative Slack (TNS):	0,000 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	NA
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	NA
Total Number of Endpoints:	157	Total Number of Endpoints:	157	Total Number of Endpoints:	NA

Figura 15

3.5. Arquitectura desenrollada

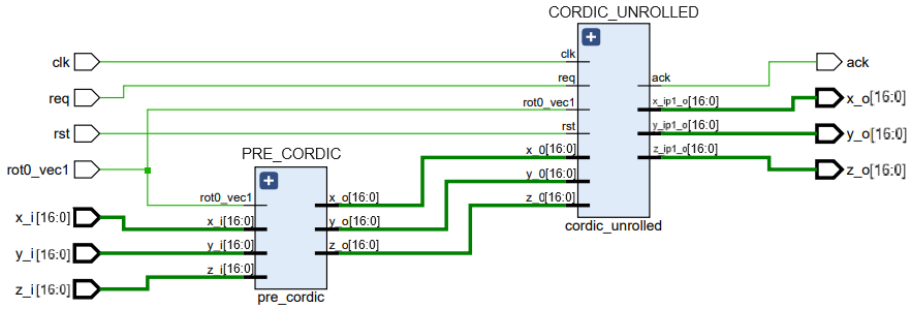


Figura 16

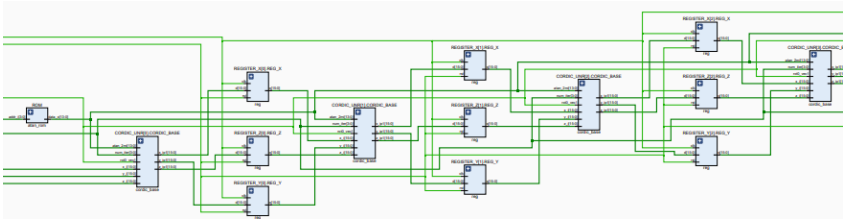


Figura 17

Resource	Utilization	Available	Utilization %
LUT	1660	10400	15.96
FF	730	20800	3.51
IO	101	170	59.41

Figura 18

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	inf	Worst Hold Slack (WHS):	inf	Worst Pulse Width Slack (WPWS):	NA
Total Negative Slack (TNS):	0,000 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	NA
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	NA
Total Number of Endpoints:	1519	Total Number of Endpoints:	1519	Total Number of Endpoints:	NA

Figura 19

4. Conclusión

Se puede concluir que el presente trabajo práctico permitió desarrollar el algoritmo CORDIC de manera correcta, dando los resultados de las simulaciones coincidente con los resultados dados por los calculos realizados en Excel.