UNIVERSITÀ DI PISA

CORDIC: Cartesian to Polar Coordinate Transformation

ANNO ACCADEMICO 2024/2025

Indice

1 Introduction **3**

1.1 Specification 3

1.2 Circuit Applications 3

2 Architecture **4**

2.1 Data Representation 4

3 VHDL code **5**

3.1 Cordic 5

4 Verification and testing **10**

4.1 Testbench 10

5 Synthesis and Implementation **11**

5.1 Vivado Design flow 11

5.2 RTL 11

5.3 RTL Elaboration 11

5.4 Synthesis and Implementation 11

6 Vivado results **12**

6.1 Critical Path 12

6.2 Utilization Report 12

6.3 Power Report 12

7 Final considerations **13**

1 Introduction

1.1 Specification

1.2 Circuit Applications

2 Architecture

2.1 Data Representation

3 VHDL code

3.1 Cordic

```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
ENTITY CORDIC IS

    GENERIC (
        N : POSITIVE := 20;
        ITERATIONS : POSITIVE := 16;
        ITER_BITS : POSITIVE := 4
    );
    PORT (
        clk : IN STD_LOGIC;
        rst : IN STD_LOGIC;
        x : IN STD_LOGIC_VECTOR(N - 1 DOWNTO 0);
        y : IN STD_LOGIC_VECTOR(N - 1 DOWNTO 0);
        start : IN STD_LOGIC;
        rho : OUT STD_LOGIC_VECTOR(N - 1 DOWNTO 0);
        theta : OUT STD_LOGIC_VECTOR(N - 1 DOWNTO 0);
        valid : OUT STD_LOGIC
    );

END ENTITY;

ARCHITECTURE behavioral OF CORDIC IS

    -- CONSTANT k : SIGNED(N - 1 DOWNTO 0) := to_signed(1304065748, N); -- 1/(Gain factor) multiplied by 2^N-1
    CONSTANT k : SIGNED(N - 1 DOWNTO 0) := to_signed(INTEGER(0.6072529351031394 * (2 ** (N - 2))), N); -- todo documentare meglio il N-2 (vivado si lamenta) ((probabilmente per la dimensione massima degli integer))
    CONSTANT HALF_PI : SIGNED(N - 1 DOWNTO 0) := to_signed(INTEGER(1.570796327 * (2 ** (N - 3))), N); -- todo documentare meglio il N-3

    -- internal registers
    SIGNAL x_t : SIGNED(N - 1 DOWNTO 0);
    SIGNAL y_t : SIGNED(N - 1 DOWNTO 0);
    SIGNAL z_t : SIGNED(N - 1 DOWNTO 0);

    SIGNAL x_out : STD_LOGIC_VECTOR(N - 1 DOWNTO 0);
    SIGNAL z_out : STD_LOGIC_VECTOR(N - 1 DOWNTO 0);

    SIGNAL address : STD_LOGIC_VECTOR(ITER_BITS - 1 DOWNTO 0);
    SIGNAL atan_out : STD_LOGIC_VECTOR(N - 1 DOWNTO 0);

    SIGNAL sign : STD_LOGIC;

    -- atan table
    COMPONENT ATAN_LUT IS
        PORT (
            address : IN STD_LOGIC_VECTOR(ITER_BITS - 1 DOWNTO 0);
            lut_out : OUT STD_LOGIC_VECTOR(N - 1 DOWNTO 0)
        );
    END COMPONENT;

    -- state type and registers
    TYPE state_t IS (WAITING, FIX_STEP, COMPUTING, FINISHED);
    SIGNAL current_state : state_t;

    SIGNAL counter : UNSIGNED(ITER_BITS - 1 DOWNTO 0);

BEGIN
    -----------------------------------------------
    -- ISTANTIATE ALL COMPONENTS
    -----------------------------------------------

    -- output assignment
    rho <= x_out;
    theta <= z_out;

    -- sign bit
    sign <= '0' WHEN y_t > 0 ELSE
        '1';

    -- atan table
    -- todo probably needs fix (next clock)
    atan_lut_inst : ATAN_LUT
    PORT MAP(
        address => address,
        lut_out => atan_out
    );

    -- atan table address
    address <= STD_LOGIC_VECTOR(counter);

    -- todo decidere se tenere o togliere gli assegnamenti
    ↪  stupidi
    -- todo forse z dovrebbe avere solo 2 bit interi e il resto
    ↪  frazionari
    -- aggiustare meglio spiegazione e codice per i 29 bit di
    ↪  atan

    -- control part
    controllo : PROCESS (clk, rst)
    BEGIN
        IF (rising_edge(clk)) THEN

            IF rst = '1' THEN
                current_state <= WAITING;
            ELSE

                CASE current_state IS

                    WHEN WAITING =>
                        IF start = '1' THEN
                            current_state <= FIX_STEP;
                        ELSE
                            current_state <= WAITING;
                        END IF;

                    WHEN FIX_STEP =>
                        current_state <= COMPUTING;

                    WHEN COMPUTING =>
                        IF counter = ITERATIONS - 1 THEN
                            current_state <= FINISHED;
                        ELSE
                            current_state <= COMPUTING;
                        END IF;

                    WHEN FINISHED =>
                        current_state <= WAITING;
                    WHEN OTHERS =>
                        current_state <= WAITING;

                END CASE;
            END IF;
        END IF;
    END PROCESS;

    -- operation part
    operativa : PROCESS (clk, rst)
    BEGIN
        IF (rising_edge(clk)) THEN

            IF rst = '1' THEN
                valid <= '0';
                -- Non utili
                x_out <= (OTHERS => '0');
                z_out <= (OTHERS => '0');
                counter <= (OTHERS => '0');
                x_t <= (OTHERS => '0');
                y_t <= (OTHERS => '0');
                z_t <= (OTHERS => '0');
            ELSE

                -- Default assignment
                -- todo vedere se tenere o togliere
                x_t <= (OTHERS => '-');
                y_t <= (OTHERS => '-');
                z_t <= (OTHERS => '-');
                x_out <= (OTHERS => '-');
                z_out <= (OTHERS => '-');
                counter <= (OTHERS => '-');

                CASE current_state IS
                    WHEN WAITING =>

                        x_t <= signed(x);
                        y_t <= signed(y);
                        z_t <= to_signed(0, N);
                        valid <= '1';
                        x_out <= x_out;
                        z_out <= z_out;

                    WHEN FIX_STEP =>
                        IF sign = '0' THEN
                            x_t <= y_t;
                            y_t <= - x_t;
                            z_t <= z_t + HALF_PI;
                        ELSE
                            x_t <= - y_t;
                            y_t <= x_t;
                            z_t <= z_t - HALF_PI;
                        END IF;

                        valid <= '0';
                        counter <= (OTHERS => '0');

                    WHEN COMPUTING =>
                        IF sign = '1' THEN
                            -- x_t <= x_t - y_t/(2 **
                             ↪  to_integer(counter));
                            x_t <= x_t - shift_right(y_t,
                             ↪  to_integer(counter));
                            -- y_t <= y_t + x_t/(2 **
                             ↪  to_integer(counter));
                            y_t <= y_t + shift_right(x_t, to_integer(counter));
                            z_t <= z_t - signed(atan_out);
                        ELSE
                            -- x_t <= x_t + y_t/(2 ** to_integer(counter));
                            x_t <= x_t + shift_right(y_t, to_integer(counter));
                            -- y_t <= y_t - x_t/(2 ** to_integer(counter));
                            y_t <= y_t - shift_right(x_t, to_integer(counter));
                            z_t <= z_t + signed(atan_out);
                        END IF;

                        counter <= counter + 1;
                        valid <= '0';

                    WHEN FINISHED =>
                        -- x_out <= STD_LOGIC_VECTOR(resize(shift_right(x_t * k , N-2),N));
                        x_out <= STD_LOGIC_VECTOR(resize(x_t * k/(2 ** (N - 2)), N));
                        -- x_out <= STD_LOGIC_VECTOR(x_t);
                        z_out <= STD_LOGIC_VECTOR(z_t);
                        valid <= '1';
                END CASE;
            END IF;
          END IF;

        END PROCESS;
END ARCHITECTURE;
```

Codice 3.1: Esempio di SSB

4 Verification and testing

4.1 Testbench

5 Synthesis and Implementation

5.1 Vivado Design flow

5.2 RTL

5.3 RTL Elaboration

5.4 Synthesis and Implementation

6 Vivado results

6.1 Critical Path

6.2 Utilization Report

6.3 Power Report

7 Final considerations