



UNIVERSITÀ DI PISA

CORDIC: Cartesian to Polar Coordinate Transformation

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

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.ALL;
3  USE ieee.numeric_std.ALL;
4  ENTITY CORDIC IS
5
6      GENERIC (
7          N : POSITIVE := 20;
8          ITERATIONS : POSITIVE := 16;
9          ITER_BITS : POSITIVE := 4
10     );
11     PORT (
12         clk : IN STD_LOGIC;
13         rst : IN STD_LOGIC;
14         x : IN STD_LOGIC_VECTOR(N - 1 DOWNT0 0);
15         y : IN STD_LOGIC_VECTOR(N - 1 DOWNT0 0);
16         start : IN STD_LOGIC;
17         rho : OUT STD_LOGIC_VECTOR(N - 1 DOWNT0 0);
18         theta : OUT STD_LOGIC_VECTOR(N - 1 DOWNT0 0);
19         valid : OUT STD_LOGIC
20     );
21
22 END ENTITY;
23
24 ARCHITECTURE behavioral OF CORDIC IS
25
26     -- CONSTANT k : SIGNED(N - 1 DOWNT0 0) :=
27     --   ↳ to_signed(1304065748, N); -- 1/(Gain factor) multiplied by
28     --   ↳ 2N-1
29     CONSTANT k : SIGNED(N - 1 DOWNT0 0) :=
30     --   ↳ to_signed(INTEGER(0.6072529351031394 * (2 ** (N - 2))), N);
31     --   ↳ -- todo documentare meglio il N-2 (vivado si lamenta)
32     --   ↳ ((probabilmente per la dimensione massima degli integer))
33     CONSTANT HALF_PI : SIGNED(N - 1 DOWNT0 0) :=
34     --   ↳ to_signed(INTEGER(1.570796327 * (2 ** (N - 3))), N); --
35     --   ↳ todo documentare meglio il N-3
```

```

30  -- internal registers
31  SIGNAL x_t : SIGNED(N - 1 DOWNT0 0);
32  SIGNAL y_t : SIGNED(N - 1 DOWNT0 0);
33  SIGNAL z_t : SIGNED(N - 1 DOWNT0 0);
34
35  SIGNAL x_out : STD_LOGIC_VECTOR(N - 1 DOWNT0 0);
36  SIGNAL z_out : STD_LOGIC_VECTOR(N - 1 DOWNT0 0);
37
38  SIGNAL address : STD_LOGIC_VECTOR(ITER_BITS - 1 DOWNT0 0);
39  SIGNAL atan_out : STD_LOGIC_VECTOR(N - 1 DOWNT0 0);
40
41  SIGNAL sign : STD_LOGIC;
42
43  -- atan table
44  COMPONENT ATAN_LUT IS
45      PORT (
46          address : IN STD_LOGIC_VECTOR(ITER_BITS - 1 DOWNT0 0);
47          lut_out : OUT STD_LOGIC_VECTOR(N - 1 DOWNT0 0)
48      );
49  END COMPONENT;
50
51  -- state type and registers
52  TYPE state_t IS (WAITING, FIX_STEP, COMPUTING, FINISHED);
53  SIGNAL current_state : state_t;
54
55  SIGNAL counter : UNSIGNED(ITER_BITS - 1 DOWNT0 0);
56
57  BEGIN
58      -----
59      -- INSTANTIATE ALL COMPONENTS
60      -----
61
62      -- output assignment
63      rho <= x_out;
64      theta <= z_out;
65
66      -- sign bit
67      sign <= '0' WHEN y_t > 0 ELSE
68          '1';
69
70      -- atan table
71      -- todo probably needs fix (next clock)
72      atan_lut_inst : ATAN_LUT
73      PORT MAP(
74          address => address,
75          lut_out => atan_out
76      );
77
78      -- atan table address
79      address <= STD_LOGIC_VECTOR(counter);
80

```

```

81  -- todo decidere se tenere o togliere gli assegnamenti
    ↳ stupidi
82  -- todo forse z dovrebbe avere solo 2 bit interi e il resto
    ↳ frazionari
83  -- aggiustare meglio spiegazione e codice per i 29 bit di
    ↳ atan
84
85  -- control part
86  controllo : PROCESS (clk, rst)
87  BEGIN
88      IF (rising_edge(clk)) THEN
89
90          IF rst = '1' THEN
91              current_state <= WAITING;
92          ELSE
93
94              CASE current_state IS
95
96                  WHEN WAITING =>
97                      IF start = '1' THEN
98                          current_state <= FIX_STEP;
99                      ELSE
100                          current_state <= WAITING;
101                      END IF;
102
103                  WHEN FIX_STEP =>
104                      current_state <= COMPUTING;
105
106                  WHEN COMPUTING =>
107                      IF counter = ITERATIONS - 1 THEN
108                          current_state <= FINISHED;
109                      ELSE
110                          current_state <= COMPUTING;
111                      END IF;
112
113                  WHEN FINISHED =>
114                      current_state <= WAITING;
115                  WHEN OTHERS =>
116                      current_state <= WAITING;
117
118              END CASE;
119          END IF;
120      END IF;
121  END PROCESS;
122
123  -- operation part
124  operativa : PROCESS (clk, rst)
125  BEGIN
126      IF (rising_edge(clk)) THEN
127
128          IF rst = '1' THEN

```

```

129         valid <= '0';
130         -- Non utili
131         x_out <= (OTHERS => '0');
132         z_out <= (OTHERS => '0');
133         counter <= (OTHERS => '0');
134         x_t <= (OTHERS => '0');
135         y_t <= (OTHERS => '0');
136         z_t <= (OTHERS => '0');
137     ELSE
138
139         -- Default assignment
140         -- todo vedere se tenere o togliere
141         x_t <= (OTHERS => '-');
142         y_t <= (OTHERS => '-');
143         z_t <= (OTHERS => '-');
144         x_out <= (OTHERS => '-');
145         z_out <= (OTHERS => '-');
146         counter <= (OTHERS => '-');
147
148     CASE current_state IS
149     WHEN WAITING =>
150
151         x_t <= signed(x);
152         y_t <= signed(y);
153         z_t <= to_signed(0, N);
154         valid <= '1';
155         x_out <= x_out;
156         z_out <= z_out;
157
158     WHEN FIX_STEP =>
159         IF sign = '0' THEN
160             x_t <= y_t;
161             y_t <= - x_t;
162             z_t <= z_t + HALF_PI;
163         ELSE
164             x_t <= - y_t;
165             y_t <= x_t;
166             z_t <= z_t - HALF_PI;
167         END IF;
168
169         valid <= '0';
170         counter <= (OTHERS => '0');
171
172     WHEN COMPUTING =>
173         IF sign = '1' THEN
174             -- x_t <= x_t - y_t/(2 **
175             --      to_integer(counter));
176             x_t <= x_t - shift_right(y_t,
177             --      to_integer(counter));
178             -- y_t <= y_t + x_t/(2 **
179             --      to_integer(counter));

```



```

177         y_t <= y_t + shift_right(x_t,
178             ↪ to_integer(counter));
178         z_t <= z_t - signed(atan_out);
179     ELSE
180         -- x_t <= x_t + y_t/(2 **
181             ↪ to_integer(counter));
181         x_t <= x_t + shift_right(y_t,
182             ↪ to_integer(counter));
182         -- y_t <= y_t - x_t/(2 **
183             ↪ to_integer(counter));
183         y_t <= y_t - shift_right(x_t,
184             ↪ to_integer(counter));
184         z_t <= z_t + signed(atan_out);
185     END IF;
186
187     counter <= counter + 1;
188     valid <= '0';
189
190     WHEN FINISHED =>
191         -- x_out <=
192             ↪ STD_LOGIC_VECTOR(resize(shift_right(x_t
193             ↪ * k , N-2),N));
192         x_out <= STD_LOGIC_VECTOR(resize(x_t * k/(2
193             ↪ ** (N - 2)), N));
193         -- x_out <= STD_LOGIC_VECTOR(x_t);
194         z_out <= STD_LOGIC_VECTOR(z_t);
195         valid <= '1';
196     END CASE;
197     END IF;
198 END IF;
199
200 END PROCESS;
201 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