Group 44: Qin Xuan Xu (261053393), William Zhang (260975150)

Lab 2: DE1-SOC Display of Arccos Function (g44_7_segment_decoder)

The ARCCOS circuit takes an 8-bit unsigned value X and a clock signal as the inputs. The output ANGLE is a 10-bit unsigned value computed from 10*arccos(X/256) using an approximation based on the Taylor series expansion of the function. The VHDL description is shown below in Figure 1.

```
1   --
2   -- entity name: g44_ARCCOS
3   --
4   -- Version 1.0
5   -- Authors: William Zhang and Qin Xuan Xu
6   -- Date: March ??, 2023 (enter the date of the latest edit to the file)
7   library ieee; -- allows use of the std_logic_vector type
8   use ieee.std_logic_1164.all;
9   use ieee.numeric_std.all; -- needed since you are using unsigned numbers
10  entity g44_ARCCOS is
11  port ( X : in std_logic_vector(7 downto 0);
12  CLOCK : in std_logic;
13  ANGLE : out std_logic_vector(9 downto 0));
14  end g44_ARCCOS;
15
16  architecture arch of g44_ARCCOS is
17  signal X2: unsigned(15 downto 0);
18  signal P1: unsigned(31 downto 0);
19  signal S1: unsigned(8 downto 0);
20  signal P2: unsigned(24 downto 0);
21  signal S2: unsigned(10 downto 0);
22  signal P3: unsigned(18 downto 0);
23
24
25  begin
26
27     process(CLOCK)
28     begin
29        if rising_edge(CLOCK) then
30           X2 <= unsigned(X) * unsigned(X);
31           P1 <= 86 * X2;
32           S1 <= 191 + ("00" & P1(22 downto 16));
33           P2 <= S1 * X2;
34           S2 <= 1144 + ("00" & P2(24 downto 16));
35           P3 <= S2 * unsigned(X);
36           ANGLE <= std_logic_vector(900 - P3(18 downto 9));
37     end if;
38     end process;
39  end arch;
```

Figure 1: VHDL description of the ARCCOS circuit

The seven segment decoder circuit takes a 4 bit unsigned value and a ripple blanking bit as the inputs. The outputs RB_Out and segments are, respectively, a value of 1 if there is ripple blanking and a 7 bit value for the seven segment decoder. Ripple blanking occurs when the first digit of a number is 0. In that case, the decoder should not display ‘0’, but rather nothing at all. The values for the seven segment decoder are hardcoded. The architecture first checks for the values from 1 to 9, then since only 0 is left, it checks if there is a ripple blanking input. The VHDL description is shown below in figure 2.

```vhdl
--
-- entity name: g44_7_segment_decoder
--
-- Version 1.0
-- Authors: William Zhang and Qin Xuan Xu
-- Date: April 7, 2023 (enter the date of the latest edit to the file)
library ieee; -- allows use of the std_logic_vector type
use ieee.std_logic_1164.all;
use ieee.numeric_std.all; -- needed since you are using unsigned numbers
entity g44_7_segment_decoder is
port( value : in std_logic_vector(3 downto 0);
      RB_In : in std_logic;
      RB_Out : out std_logic := '0';
      segments : out std_logic_vector(6 downto 0));
end g44_7_segment_decoder;

architecture a of g44_7_segment_decoder is
begin
         segments <= "1111001" when value ="0001" else -- 1
                     "0100100" when value ="0010" else -- 2
                     "0110000" when value ="0011" else -- 3
                     "0011001" when value ="0100" else -- 4
                     "0010010" when value ="0101" else -- 5
                     "0000010" when value ="0110" else -- 6
                     "1111000" when value ="0111" else -- 7
                     "0000000" when value ="1000" else -- 8
                     "0011000" when value ="1001" else -- 9
                     "1111111" when RB_In = '1' else
                     "1000000" when value ="0000" else -- 0
                     "1111111";
   process(value, RB_In)
   begin
         if value = "0000" and RB_In = '1' then
            RB_Out <= '0';
         end if;
   end process;
end a;
```

Figure 2: VHDL description of the 7 Segment Decoder circuit

The bin2bcd module was given as part of the lab documents and the module converts a binary number into a binary coded decimal number. The Verilog description is shown below in figure 3.

```verilog
// parametric Verilog implementation of the double dabble binary to BCD converter
// for the complete project, see
// https://github.com/AmeerAbdelhadi/Binary-to-BCD-Converter

module bin2bcd
 #( parameter                W = 10)  // input width
  ( input      [W-1      :0] bin   ,  // binary
    output reg [W+(W-4)/3:0] bcd   ); // bcd {...,thousands,hundreds,tens,ones}

  integer i,j;

  always @(bin) begin
    for(i = 0; i <= W+(W-4)/3; i = i+1) bcd[i] = 0;     // initialize with zeros
    bcd[W-1:0] = bin;                                   // initialize with input vector
    for(i = 0; i <= W-4; i = i+1)                       // iterate on structure depth
      for(j = 0; j <= i/3; j = j+1)                     // iterate on structure width
        if (bcd[W-i+4*j -: 4] > 4)                      // if > 4
          bcd[W-i+4*j -: 4] = bcd[W-i+4*j -: 4] + 4'd3; // add 3
  end

endmodule
```

Figure 3: Verilog description of the binary to BCD converter

The g44_lab_2 component is a wrapper which combines the three previous components. It takes the output of the arccos function, transforms it from binary to BCD, then separates it into three 4 bit BCD numbers for three decoders to use. The VHDL description for the wrapper is shown below in figures 4 and 5.

```
--
-- entity name: g44_lab_2
--
-- Version 1.0
-- Authors: William Zhang and Qin Xuan Xu
-- Date: April 7, 2023 (enter the date of the latest edit to the file)
library ieee; -- allows use of the std_logic_vector type
use ieee.std_logic_1164.all;
use ieee.numeric_std.all; -- needed since you are using unsigned numbers
port( X : in std_logic_vector(7 downto 0);
      CLOCK: in std_logic;
      HEX0: out std_logic_vector(6 downto 0);
      HEX1: out std_logic_vector(6 downto 0);
      HEX2: out std_logic_vector(6 downto 0));
end g44_lab_2;

architecture wrapper of g44_lab_2 is
component g44_ARCCOS is
port ( X : in std_logic_vector(7 downto 0);
CLOCK : in std_logic;
ANGLE : out std_logic_vector(9 downto 0));
end component;

component g44_7_segment_decoder is
port( value : in std_logic_vector(3 downto 0);
      RB_In : in std_logic;
      RB_Out : out std_logic;
      segments : out std_logic_vector(6 downto 0));
end component;

component bin2bcd is
   generic(
   W: integer:= 10
   );
   port(
   bin: in std_logic_vector(W-1 downto 0);
   bcd: out std_logic_vector(12 downto 0)
   );

end component;
```

Figure 4: VHDL description of the wrapper (1 / 2)

```
signal tempAngle: std_logic_vector(9 downto 0);
signal tempAngle2: std_logic_vector(12 downto 0);
signal firstNumber: std_logic_vector(3 downto 0);
signal secondNumber: std_logic_vector(3 downto 0);
signal thirdNumber: std_logic_vector(3 downto 0);
signal tempHEX0 : std_logic_vector(6 downto 0);
signal tempHEX1 : std_logic_vector(6 downto 0);
signal tempHEX2 : std_logic_vector(6 downto 0);
signal rippleBlanking1 : std_logic ;
signal rippleBlanking0 : std_logic;
signal rippleBlanking : std_logic;


begin
   arccos: g44_ARCCOS port map(X, CLOCK, tempAngle);
   conversion: bin2bcd port map (tempAngle, tempAngle2);
   thirdNumber <= tempAngle2(11 downto 8);
   secondNumber <= tempAngle2(7 downto 4);
   firstNumber <= tempAngle2(3 downto 0);
   decoder2: g44_7_segment_decoder port map (thirdNumber,'1',rippleBlanking1, tempHEX2);
   decoder1: g44_7_segment_decoder port map (secondNumber, rippleBlanking1, rippleBlanking0, tempHEX1);
   decoder0: g44_7_segment_decoder port map (firstNumber, '0', rippleBlanking, tempHEX0);
   HEX0 <= tempHEX0;
   HEX1 <= tempHEX1;
   HEX2 <= tempHEX2;
end wrapper;
```

Figure 5: VHDL description of the wrapper (2 / 2)

Group 44: Qin Xuan Xu (261053393), William Zhang (260975150)

Comparison of outputs

Value	Actual (°)	Expected (°)	Value	Actual (°)	Expected (°)
0	90.0	90.0	128	60.1	60.0
1	89.8	89.8	129	59.9	59.7
2	89.6	89.6	130	59.6	59.5
3	89.4	89.3	131	59.4	59.2
4	89.2	89.1	132	59.1	59.0
5	88.9	88.9	133	58.9	58.7
6	88.7	88.7	134	58.6	58.4
7	88.5	88.4	135	58.3	58.2
8	88.3	88.2	136	58.1	57.9
9	88.0	88.0	137	57.8	57.6
10	87.8	87.8	138	57.5	57.4
11	87.6	87.5	139	57.3	57.1
12	87.4	87.3	140	57.0	56.8
13	87.1	87.1	141	56.8	56.6
14	86.9	86.9	142	56.5	56.3
15	86.7	86.6	143	56.2	56.0
16	86.5	86.4	144	56.0	55.8
17	86.3	86.2	145	55.7	55.5
18	86.0	86.0	146	55.4	55.2
19	85.8	85.7	147	55.1	55.0
20	85.6	85.5	148	54.9	54.7
21	85.4	85.3	149	54.6	54.4
22	85.1	85.1	150	54.3	54.1
23	84.9	84.8	151	54.1	53.9
24	84.7	84.6	152	53.8	53.6
25	84.5	84.4	153	53.5	53.3
26	84.2	84.2	154	53.2	53.0
27	84.0	83.9	155	53.0	52.7
28	83.8	83.7	156	52.7	52.5
29	83.6	83.5	157	52.4	52.2
30	83.3	83.3	158	52.2	51.9
31	83.1	83.0	159	51.9	51.6

32	82.9	82.8	160	51.6	51.3
33	82.7	82.6	161	51.3	51.0
34	82.4	82.4	162	51.0	50.7
35	82.2	82.1	163	50.7	50.5
36	82.0	81.9	164	50.5	50.2
37	81.8	81.7	165	50.2	49.9
38	81.5	81.5	166	49.9	49.6
39	81.3	81.2	167	49.6	49.3
40	81.1	81.0	168	49.3	49.0
41	80.9	80.8	169	49.0	48.7
42	80.6	80.6	170	48.7	48.4
43	80.4	80.3	171	48.4	48.1
44	80.2	80.1	172	48.2	47.8
45	80.0	79.9	173	47.8	47.5
46	79.7	79.6	174	47.6	47.2
47	79.5	79.4	175	47.3	46.9
48	79.3	79.2	176	47.0	46.6
49	79.0	79.0	177	46.7	46.3
50	78.8	78.7	178	46.4	45.9
51	78.6	78.5	179	46.1	45.6
52	78.3	78.3	180	45.8	45.3
53	78.1	78.1	181	45.5	45.0
54	77.9	77.8	182	45.2	44.7
55	77.7	77.6	183	44.9	44.4
56	77.4	77.4	184	44.6	44.0
57	77.2	77.1	185	44.3	43.7
58	77.0	76.9	186	44.0	43.4
59	76.8	76.7	187	43.7	43.1
60	76.5	76.4	188	43.4	42.7
61	76.3	76.2	189	43.1	42.4
62	76.1	76.0	190	42.7	42.1
63	75.8	75.8	191	42.4	41.7
64	75.6	75.5	192	42.1	41.4
65	75.4	75.3	193	41.8	41.1
66	75.1	75.1	194	41.5	40.7

67	74.9	74.8	195	41.2	40.4
68	74.7	74.6	196	40.9	40.0
69	74.4	74.4	197	40.6	39.7
70	74.2	74.1	198	40.2	39.3
71	74.0	73.9	199	39.9	39.0
72	73.8	73.7	200	39.6	38.6
73	73.5	73.4	201	39.3	38.3
74	73.3	73.2	202	39.0	37.9
75	73.1	73.0	203	38.6	37.5
76	72.8	72.7	204	38.3	37.2
77	72.6	72.5	205	38.0	36.8
78	72.3	72.3	206	37.6	36.4
79	72.1	72.0	207	37.3	36.0
80	71.9	71.8	208	37.0	35.7
81	71.7	71.6	209	36.6	35.3
82	71.4	71.3	210	36.3	34.9
83	71.2	71.1	211	35.9	34.5
84	70.9	70.8	212	35.6	34.1
85	70.7	70.6	213	35.3	33.7
86	70.5	70.4	214	34.9	33.3
87	70.2	70.1	215	34.6	32.9
88	70.0	69.9	216	34.2	32.5
89	69.7	69.7	217	33.9	32.0
90	69.5	69.4	218	33.5	31.6
91	69.3	69.2	219	33.2	31.2
92	69.0	68.9	220	32.9	30.8
93	68.8	68.7	221	32.5	30.3
94	68.6	68.5	222	32.2	29.9
95	68.3	68.2	223	31.8	29.4
96	68.1	68.0	224	31.4	29.0
97	67.8	67.7	225	31.1	28.5
98	67.6	67.5	226	30.7	28.0
99	67.3	67.2	227	30.4	27.5
100	67.1	67.0	228	30.0	27.0
101	66.9	66.8	229	29.6	26.6

102	66.6	66.5	230	29.3	26.0
103	66.4	66.3	231	28.9	25.5
104	66.1	66.0	232	28.5	25.0
105	65.9	65.8	233	28.1	24.5
106	65.6	65.5	234	27.8	23.9
107	65.4	65.3	235	27.4	23.4
108	65.2	65.0	236	27.0	22.8
109	64.9	64.8	237	26.6	22.2
110	64.7	64.6	238	26.2	21.6
111	64.4	64.3	239	25.9	21.0
112	64.2	64.1	240	25.5	20.4
113	63.9	63.8	241	25.1	19.7
114	63.7	63.6	242	24.7	19.0
115	63.4	63.3	243	24.3	18.3
116	63.2	63.1	244	23.9	17.6
117	62.9	62.8	245	23.5	16.9
118	62.7	62.6	246	23.1	16.1
119	62.4	62.3	247	22.7	15.2
120	62.2	62.0	248	22.3	14.4
121	61.9	61.8	249	21.9	13.4
122	61.7	61.5	250	21.5	12.4
123	61.4	61.3	251	21.1	11.3
124	61.2	61.0	252	20.7	10.1
125	60.9	60.8	253	20.3	8.8
126	60.6	60.5	254	19.9	7.2
127	60.4	60.3	255	19.5	5.1

The actual values for ARCCOS are very close to the expected values until the input value of 156. After that, there are larger and larger differences between the values as expected since the approximation can only be close up to a certain point after which the two curves start separating. For example, the last input value, 255, has an output value of 19.5 compared to the expected value of 5.1 whereas the first input value, 0, has an output value of 90.0 for both the expected and actual values of ARCCOS.

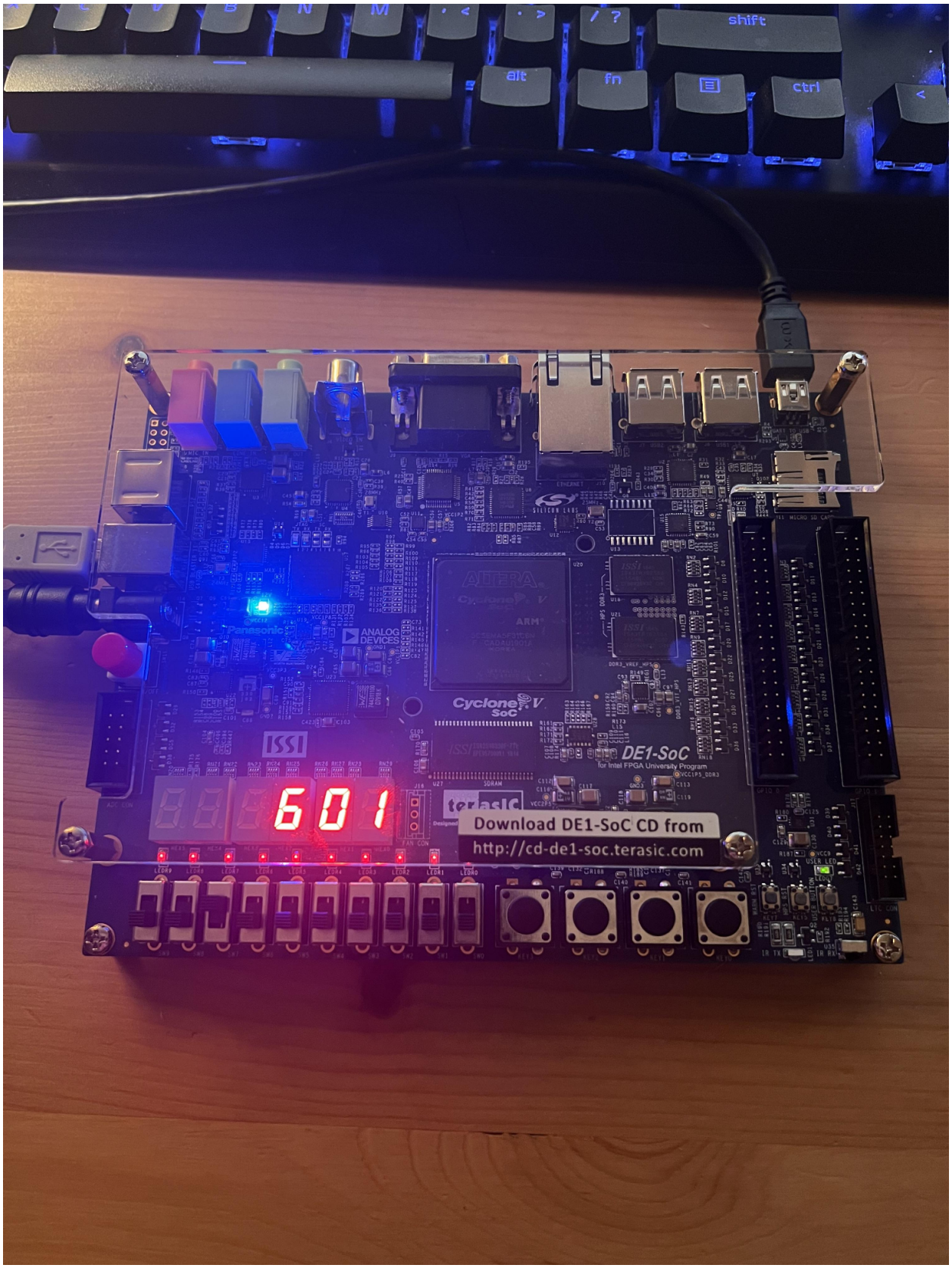

Figure 6: Value displayed for the input 128/256 (60.1 degrees)

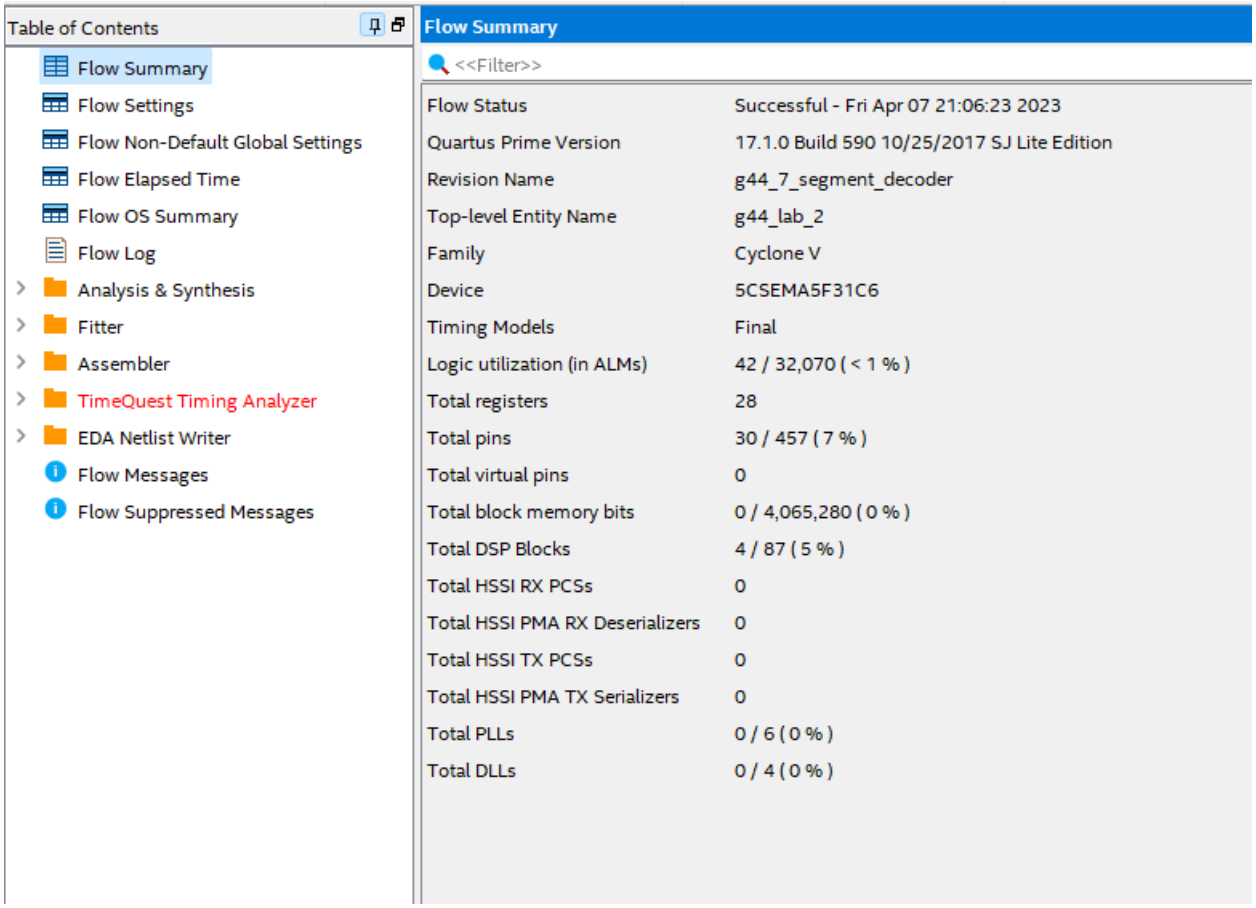

Figure 7: Flow Summary from the compilation report

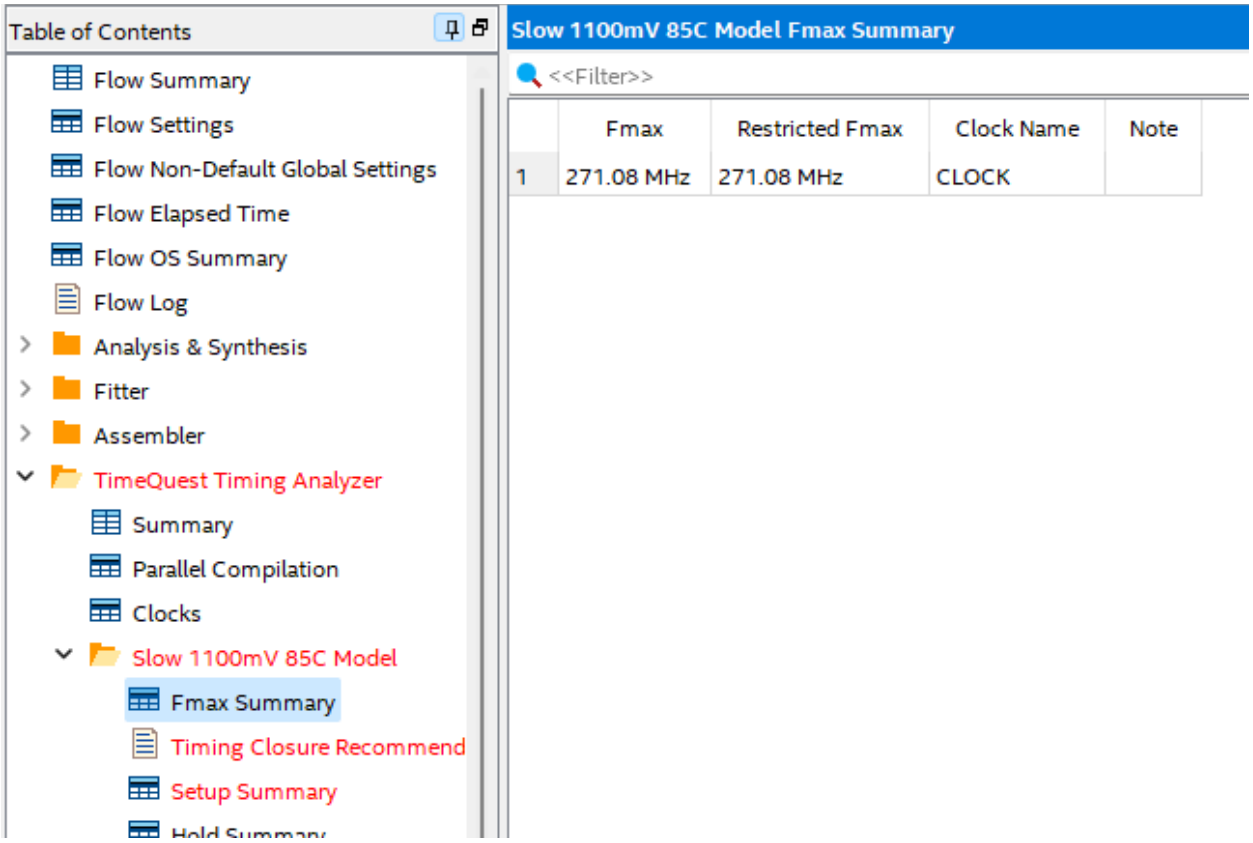

Figure 8: Fmax Summary for the Slow 1100mV 85C Model