ECSE 325 Lab 4 Report

5th Order Polynomial Fixed-Point Sine Approximation using VHDL

Group 06

Adrian Wang	260769387
Irene Ma	260776283
Xiangyun Wang	260771591

1. Circuit function description

The circuit takes as input a 16-bit unsigned number as the angle and generates a 16-bit unsigned value for the sine of that angle. The angle input values are assumed to be non-negative and cover the range from 0 degrees to 90 degrees. Since both the angle and the sine of the angle are non-negative over this range, we can use unsigned numbers. Values for other angles can be obtained by various symmetry transformations, which we won't do in this lab. The algorithm implements the following formula: Output = y * 2^-n * (A1 - 2^(q-p)* y * 2^-n * y * 2^-n * (B1 - 2^-r * y * 2^(-n) * C1 * y)) * 2^(a-q). There are seven steps in total in achieving this algorithm. For the first step, the innermost multiplication by y has been preceded by a multiplication by 2^(-n).

2. Combinational block implementation

2.1. Flow Summary

Flow Summary

<<Filter>>

Flow Status	Successful - Sat Mar 27 16:47:14 2021
Quartus Prime Version	16.1.0 Build 196 10/24/2016 SJ Lite Edition
Revision Name	g06_Sine
Top-level Entity Name	g06_Sine
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	67 / 32,070 (< 1 %)
Total registers	13
Total pins	33 / 457 (7 %)
Total virtual pins	0
Total block memory bits	0 / 4,065,280 (0 %)
Total DSP Blocks	10 / 87 (11 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 4 (0 %)

2.2. Timing Analysis

The circuit contains many combinational blocks without process, and computation is heavily relied on multiplication and bit shift. Therefore, it can be expected that the delay is relatively high. A series of timing simulations are performed and the results are shown in the table below.

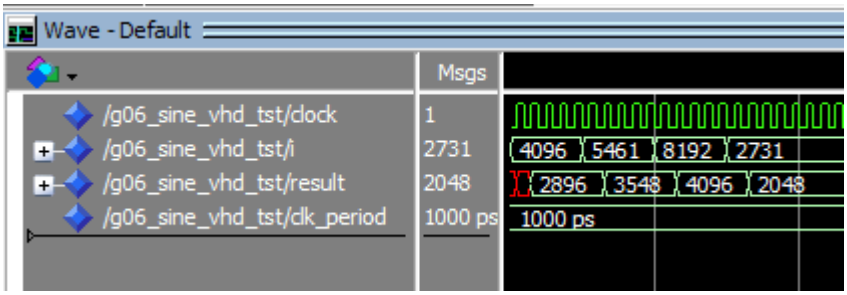
Clock Period	12	24	36	40
Fmax 85C	27.83 MHz	28.1 MHz	27.73 MHz	27.82 MHz
Slow 85C Setup	-23.933	-11.590	-0.058	4.057
Slow 0C Setup	-24.663	-12.502	-0.876	3.079
Fast 85C Setup	-7.777	4.384	16.122	20.344
Fast 0C Setup	-6.637	5.532	17.248	21.421

2.3. Functional Simulation

To test the circuit a simple testbench is written and four cases of input angles between 0 and 90 degrees are tested. As shown in the figure below, the angles of 45, 60, 90, and 30 degrees

are considered. To represent the values in binary, the input is first scaled into a range of 0-8192, with respect to the original full scale of 90 degrees. The output is scale to 0-4096, representing the sine value ranging from 0 to 1. Both scaling are uniformly distributed and the correct result comparison is shown in the table below.

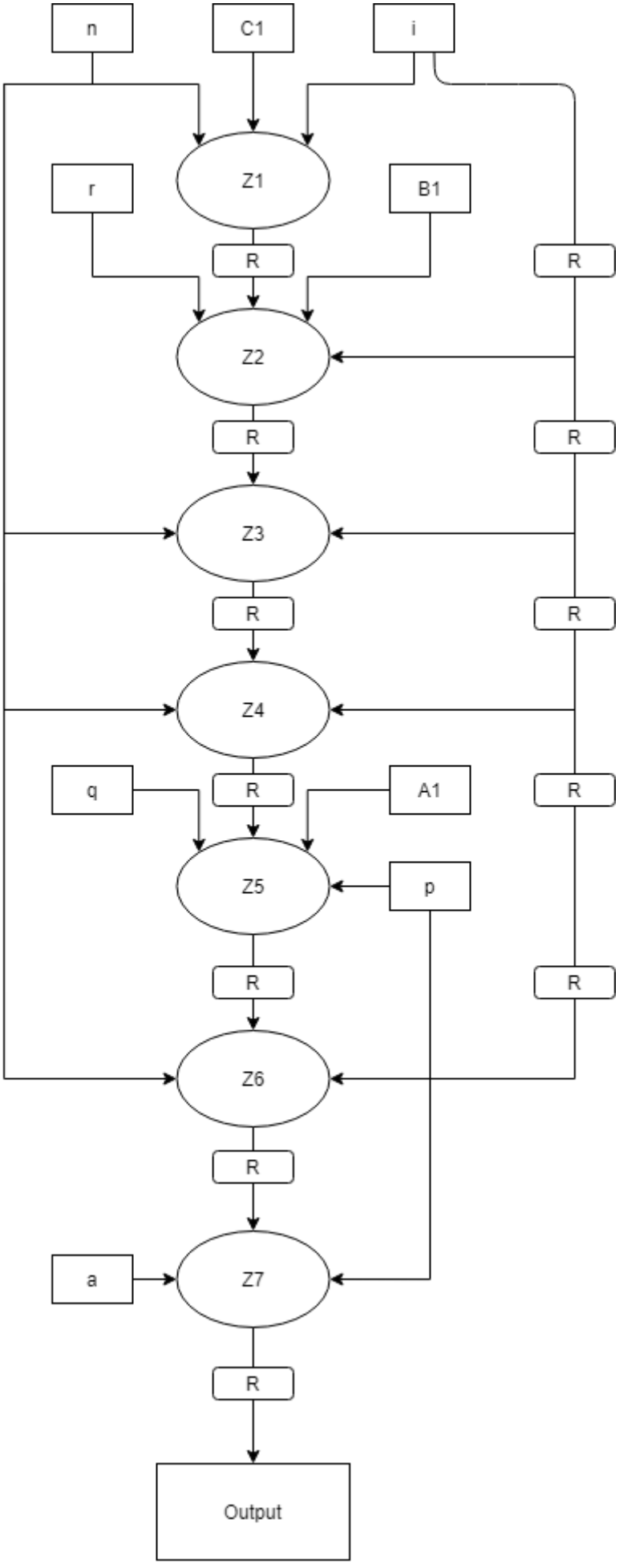
```
-- input Full Scale (FS): 8192
-- pi/4 => 4096
i<="0001000000000000";
wait for 5 * clk_period;
-- pi/3 => 5461.3333
i<="0001010101010101";
wait for 5 * clk_period;
-- pi/2 => 8192
i<="0010000000000000";
wait for 5 * clk_period;
-- pi/6 => 2730.6667
i<="0000101010101011";
wait;
```



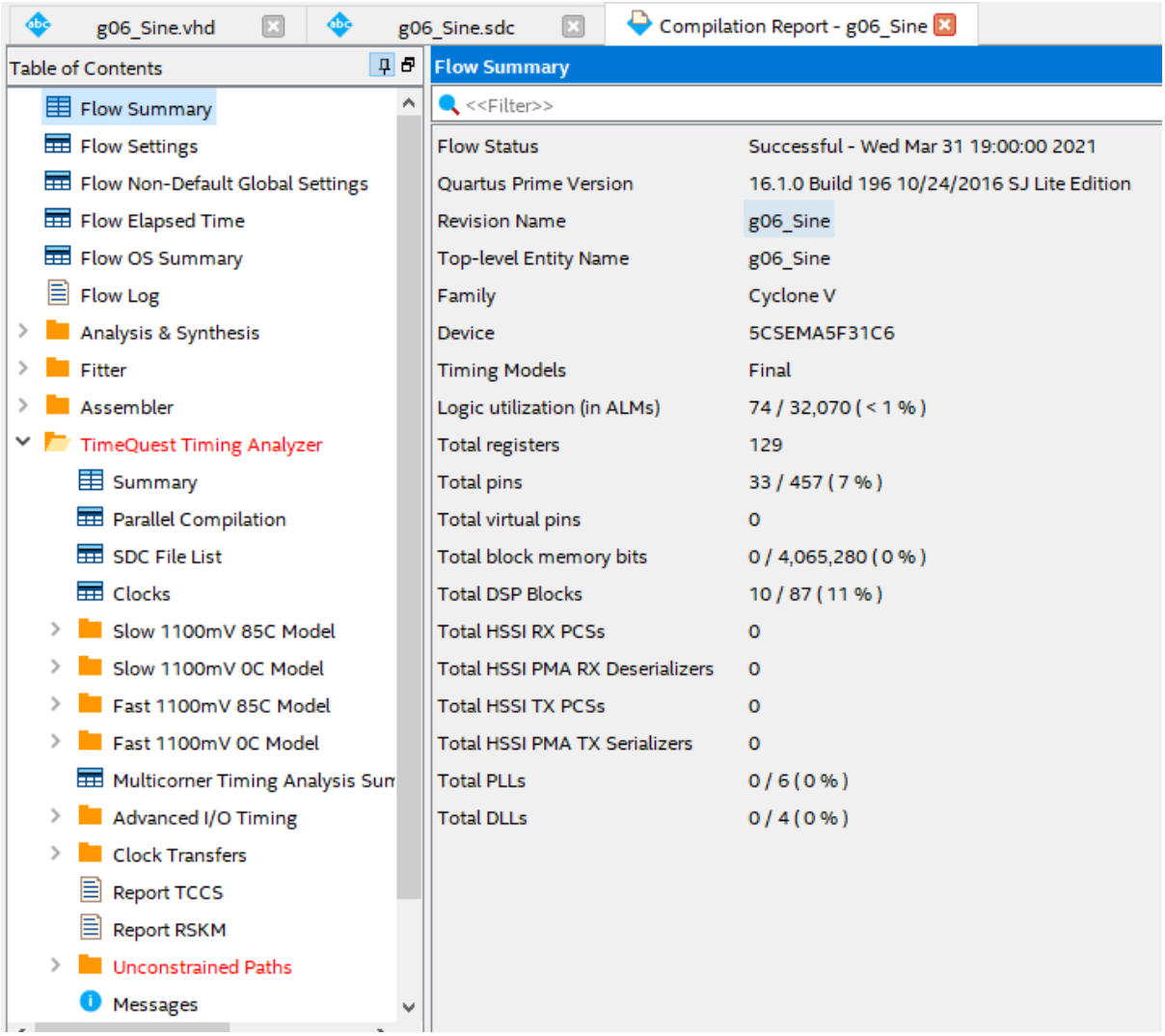
Input Angle	Mathematical Result	Wave Result	Wave Result / 4096
45	0.7071	2896	0.7070
60	0.8660	3548	0.8662
90	1	4096	1
30	0.5	2048	0.5

3. Pipelined implementation

3.1. Flow Diagram


n
C1
i
Z1
r
B1
R
R
Z2
R
R
Z3
R
R
Z4
q
R
A1
R
Z5
p
R
R
Z6
R
a
Z7
R
Output

3.2. Flow Summary

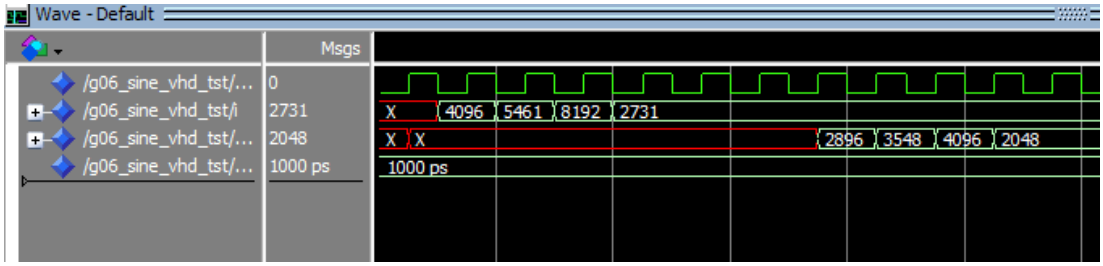


3.3. Timing Analysis

Clock Period	6	8	12	24
Fmax 85C	105.3 MHz	97.88 MHz	103.27 MHz	93.46 MHz
Slow 85C Setup	-3.497	-2.217	2.317	13.300
Slow 0C Setup	-3.769	-2.476	2.027	13.031
Fast 85C Setup	0.856	2.370	6.728	18.153
Fast 0C Setup	1.111	2.670	7.000	18.496

3.4. Functional Simulation

With a minimal modification, the testbench of unpipelined version can be directly used to test the pipelined version. The results should and are indeed the same as before, while showing that the pipeline has seven stages as shown in the flow diagram. This is why the result appears seven clock cycles after the circuit receives the input.



4. Discussion and Comparison

In the first version of the program, the VHDL is directly translated from C code and all data processing is done in one combinational block. This is an easy way to get the VHDL code written and running, however, since the nature of two languages are different, the translated code tends to be less efficient. Within the combinational block, each line is supposed to execute concurrently to form the circuit. Therefore, if there are too many dependencies in between, the parts of the circuit need to wait for the data needed which will cause significant delay. This can be seen in the timing analysis that the maximum frequency is only around 28 MHz.

To optimize the VHDL code for the first version, pipelining is a correct direction by dividing the multiplication operations so that the irrelevant parts can be computed at the same time. The core part of the second version, the process (figure below), shows the balanced-path implementation of the flow diagram. In using the registers z and i which represents y and input in the algorithm, there are *next_x* signals that store the intermediate values from each stage of calculation.

```
update : process(clock)
begin
  if (clock'event and rising_edge(clock)) then
    z_reg_1 <= next_1;
    z_reg_2 <= next_2;
    z_reg_3 <= next_3;
    z_reg_4 <= next_4;
    z_reg_5 <= next_5;
    z_reg_6 <= next_6;
    z_reg_7 <= next_7;
    i_reg_5 <= i_reg_4;
    i_reg_4 <= i_reg_3;
    i_reg_3 <= i_reg_2;
    i_reg_2 <= i_reg_1;
    i_reg_1 <= i_change;
    --i_change <= "0000000000000000"&i;
  end if;
end process;
```

From the flow summary we can see that the ALM logic usage is slightly higher than the first version (74 versus 67) but is still acceptable. The main advantage of pipelined implementation is that the delay in the path could be reduced significantly and the maximum frequency is correspondingly increased by about three times larger (103.27 MHz compared with 28.1 MHz).