DSP in VLSI

HW1

電子所 ICS 組, R13943015, 張根齊

1. First, generate several random sequences with 32 elements [$x1$ $x2$ … $x32$] in a block between -128 and 127 as your test inputs and prepare as the testbench. (10%)

I use MATLAB to generate N random sequences, where N is user-defined. I write these random sequences to the *PX_DATA_I.dat* file as shown below. Then, I use *readmemb* to read the random sequences into an array in the testbench.

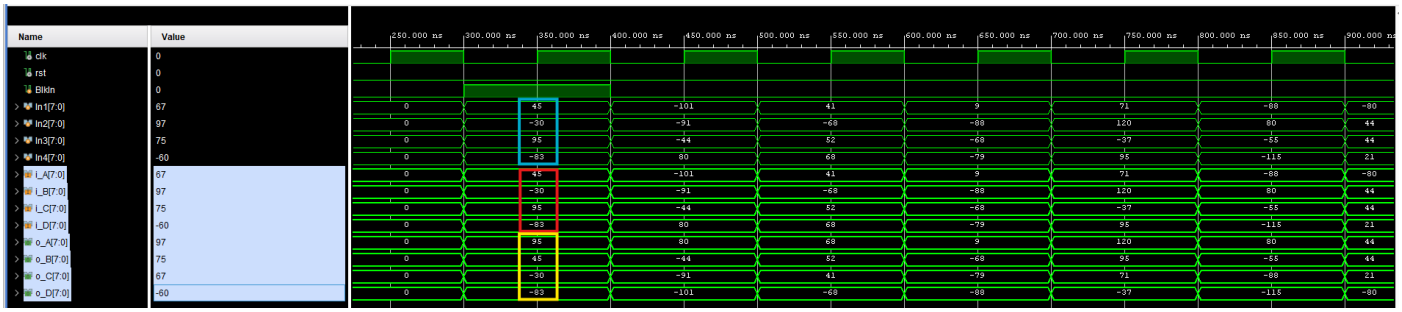
testbench.v | Sort4.v | comparator_tree.v | SelectTopK.v | P0_DATA_I.dat

```
// Pattern 0:
01010000  // 80
01100111  // 103
10100000  // -96
01101001  // 105
00100001  // 33
10011000  // -104
11000111  // -57
00001100  // 12
01110101  // 117
01110111  // 119
10101000  // -88
01111000  // 120
01110101  // 117
11111100  // -4
01001100  // 76
10100100  // -92
11101011  // -21
01101010  // 106
01001010  // 74
01110101  // 117
00100111  // 39
10001001  // -119
01011001  // 89
01101111  // 111
00101101  // 45
01000001  // 65
00111110  // 62
11100100  // -28
00100111  // 39
10101011  // -85
00110100  // 52
10001000  // -120

// Pattern 1:
11000110  // -58
10001011  // -117
10011000  // -104
01010010  // 82
00110001  // 49
11010001  // -47
01110011  // 115
10001000  // -120
11110000  // -16
11100001  // -31
01000011  // 67
01001011  // 75
10101111  // -81
11111101  // -3
11110010  // -14
00100101  // 37
00110101  // 53
01000001  // 65
11000110  // -58
00101110  // 46
00100111  // 39
10101001  // -87
10011110  // -98
11111111  // -1
01110101  // 117
11010111  // -41
00010101  // 21
10111001  // -71
01000000  // 64
11000001  // -63
00000001  // 1
00110010  // 50

// Pattern 2:
01100100  // 100
```

```
45    initial begin
46        $readmemb(`DATA_I_PATH, input_data);
47        $readmemb(`GOLDEN_O_PATH, golden_data);
48    end
```

2. Please use Verilog to implement Sort4. Feed the 4 inputs [$x1$ $x2$ $x3$ $x4$] simultaneously in one clock cycle. Observe the outputs to realize sorting among four elements in one group. Use [$x1$ $x2$ $x3$ $x4$] as the input and check for the correctness of the function “Sort4”, which can generate sorted output from maximum to minimum as shown in the following Fig. 4. Indicate this function output in the timing diagram of your behavior simulation results. (10%)

In1 to In4 are the inputs of *SelectTopK* (blue block). They are directly fed into *Sort4* as *i_A* to *i_D* (red block). *Sort4* generates a sorted output in descending order, labeled as *o_A* to *o_D* (yellow block). The function operates correctly, as shown below.



3. Construct the module (SelectTopK) of merge sort to select top 7 among 32 input elements ([$x1$ $x2$ $x3$ … $x32$]) under your control. One example is given as in the following Fig. 5. Use a comparator tree to feed the sorted results from each group and then activate your comparator tree to select top 7 values.

3.1. Draw the block diagram of your own implementation and calculate the number of adders/subtractors/comparators in your block diagram. (20%)

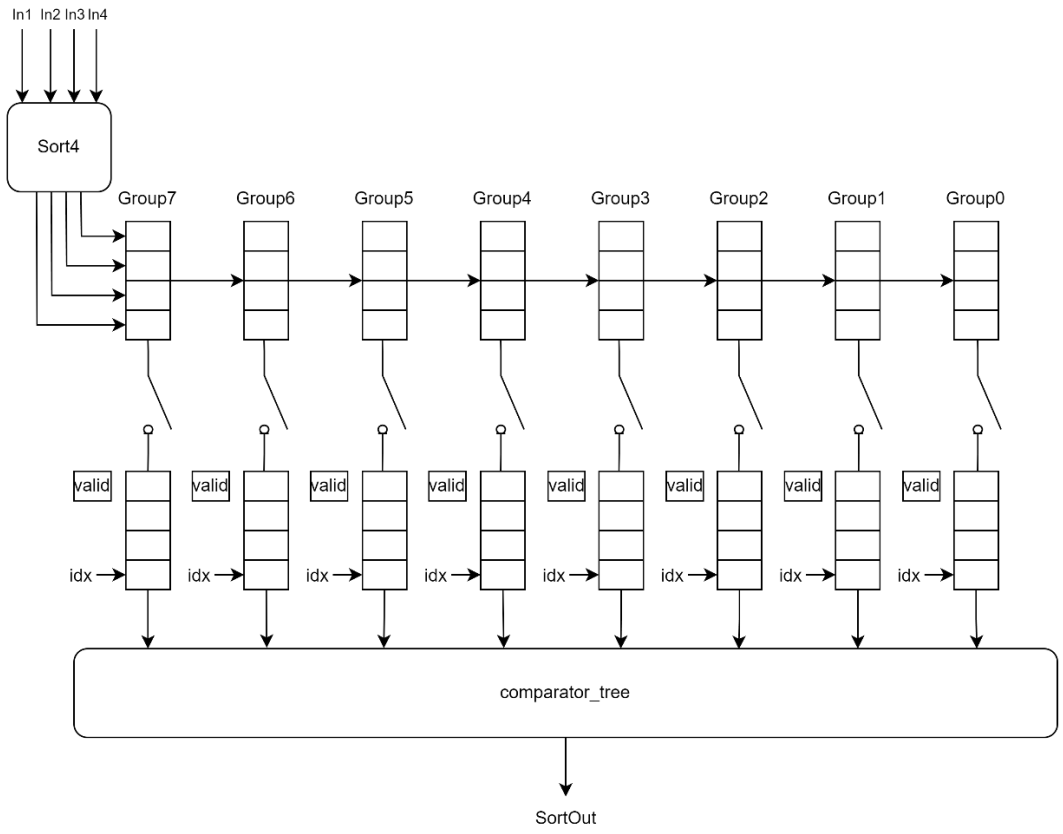
3.2. Show the timing diagram of your behavior simulation results (10%)

3.3. Show the hardware output is correct by comparing it to your Matlab results. (10%)

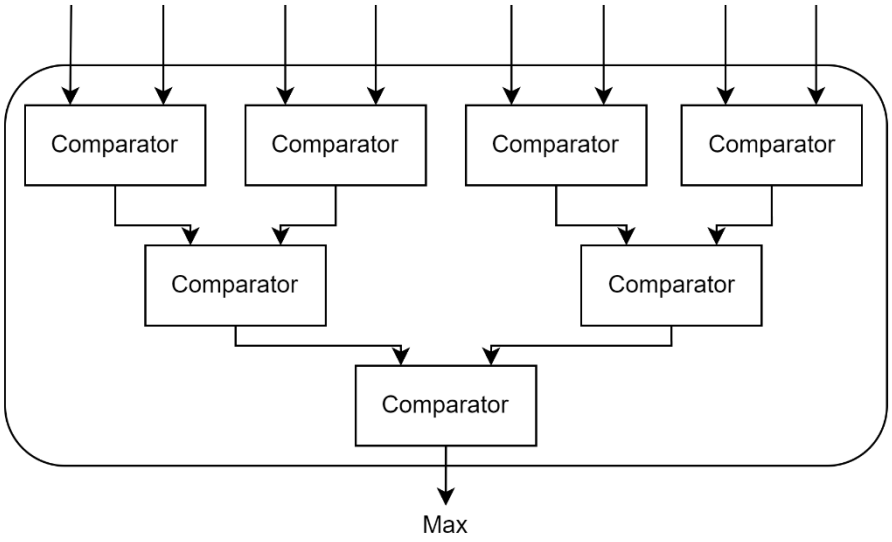
3.1:

Inputs *In1* to *In4* are fed directly into *Sort4*, which produces four sorted outputs. These outputs are then passed to shift registers. After collecting *Group 0* to *Group 7*, the shift registers are transferred to another set of group registers. Finally, a comparator tree determines the maximum value among the eight groups and outputs it over seven cycles to obtain the top 7 values.

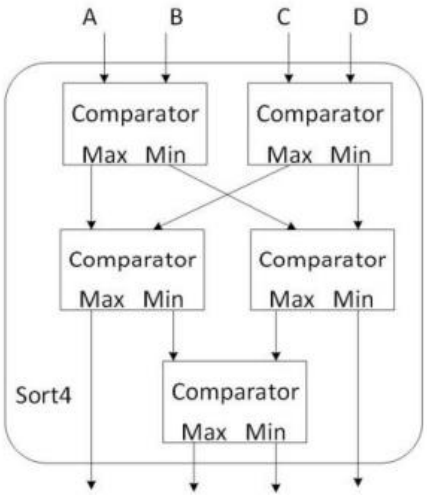
SelectTopK:



comparator_tree:



Sort4:

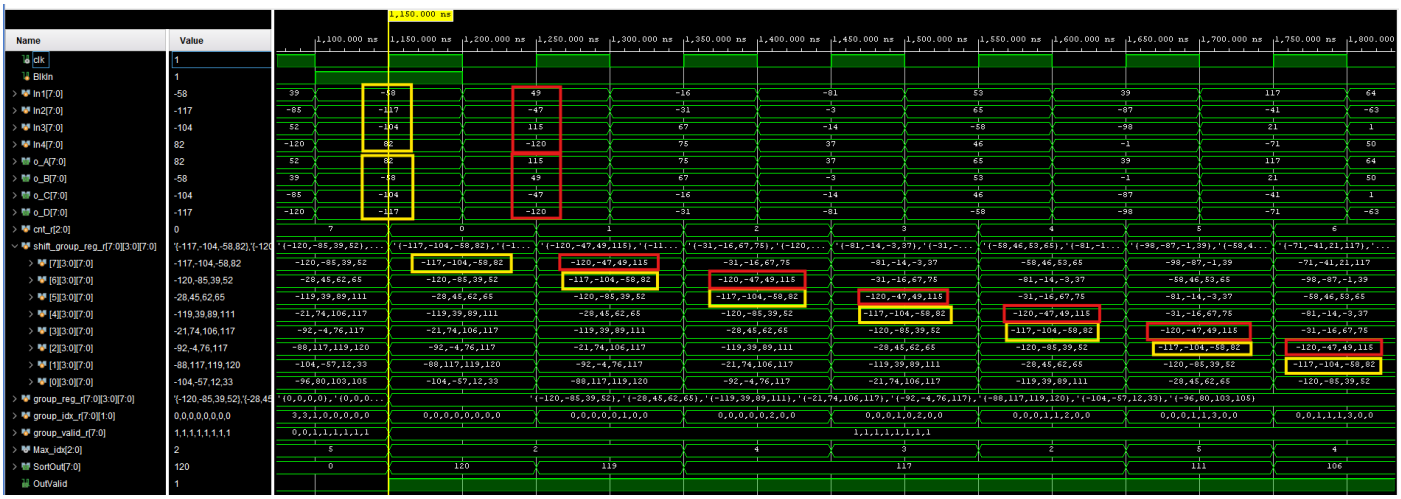


number of adders/subtractors/comparators

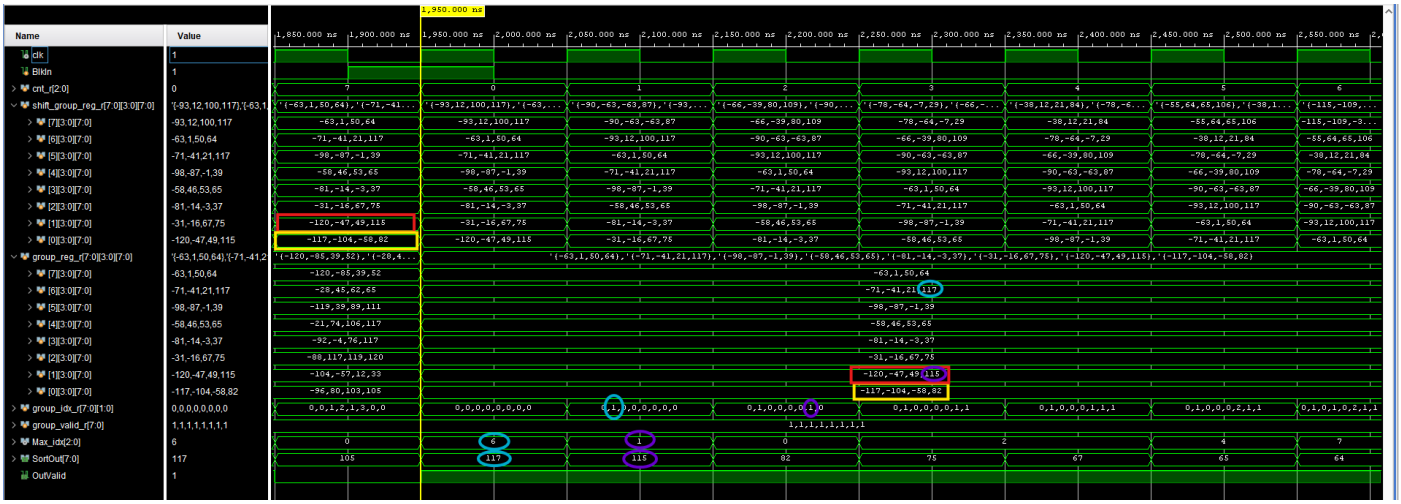
module	Sort4	comparator_tree	Total
#comparator	5	7	12

3.2:

Inputs *In1* to *In4* are fed directly into *Sort4*, which produces four sorted outputs *o_A* to *o_D*. These outputs are then passed to shift registers *shift_group_reg_r*. Waveform is shown as below.



After collecting *Group 0* to *Group 7*, the shift registers *shift_group_reg_r* are transferred to another set of group registers *group_reg_r*. Finally, the comparator tree determines the maximum value among the eight groups and outputs it over seven cycles to obtain the top 7 values. After outputting each element, *group_reg_r* updates its pointer *group_idx_r*, as shown below.



3.3:

I use Matlab to implement the algorithm in software. And compare the software result with hardware result.

Matlab software code:

SelectTopK.m

```
input_filename = '../TESTBED/pattern/P0_DATA_I.dat';
output_filename = '../SOFTWARE/result/P0_SOFTWARE_RESULT_O.dat';

data_file = fopen(input_filename, 'r');
output_file = fopen(output_filename, 'w');

while ~feof(data_file)
    line = fgetl(data_file);
    if contains(line, 'Pattern')
        fprintf(output_file, '%s\n', line);
        sequence = zeros(1, 32);

        % Read 32 values
        for i = 1:32
            line = fgetl(data_file);
            parts = strsplit(line, '  // ');
            sequence(i) = str2double(parts{2});
        end

        % Sorting in chunks of 4
        group_reg_r = zeros(8, 4);
        for i = 1:8
            group_reg_r(i,:) = sort(sequence((i-1)*4+1:i*4), 'descend');
        end

        % Finding top 7 values iteratively
        group_idx_r = ones(1, 8);
        group_idx_valid = ones(1, 8);



        for i = 1:7
            % Extract values from group_reg_r using group_idx_r
            candidates = arrayfun(@(i) group_reg_r(i, group_idx_r(i)), 1:8);
            % Apply validity mask: if group i is invalid, then group i
            % should not in comparison. (set group i -inf)
            valid_candidates = candidates .* group_idx_valid;  % Set invalid entries to zero (or use -inf)
            valid_candidates(group_idx_valid == 0) = -inf; % Set invalid values to -inf for max comparison

            % Find the maximum value and its index
            [SortOut, Max_idx] = max(valid_candidates);

            % Write top 7 values in binary format
            bin_str = dec2bin(typecast(int8(SortOut), 'uint8'), 8);
            fprintf(output_file, '%s  // %d\n', bin_str, SortOut);

            % Update group_idx_r and group_idx_valid
            if group_idx_r(Max_idx) == 4 % already last one
                group_idx_valid(Max_idx) = 0;
            else
                group_idx_r(Max_idx) = group_idx_r(Max_idx) + 1;
            end
        end

        fprintf(output_file, '\n');

    end
end

fclose(data_file);
fclose(output_file);
```

Testbench test pattern selection:

```
`timescale 1ns/10ps
`define PERIOD    100.0
`define MAX_CYCLE 100000
`define RST_DELAY 2.0
`define I1_SOFTWARE

`ifdef I0_GOLDEN // VERIFY USING GLODEN DATA
    `define DATA_I_PATH  "C:/Users/zhanggenqi/Desktop/DSP_in_VLSI/HW1/TESTBED/pattern/P0_DATA_I.dat"
    `define GOLDEN_O_PATH  "C:/Users/zhanggenqi/Desktop/DSP_in_VLSI/HW1/TESTBED/pattern/P0_GOLDEN_O.dat"
    `define PAT_LEN 1000
`elsif I1_GOLDEN // VERIFY USING GLODEN DATA
    `define DATA_I_PATH  "C:/Users/zhanggenqi/Desktop/DSP_in_VLSI/HW1/TESTBED/pattern/P1_DATA_I.dat"
    `define GOLDEN_O_PATH  "C:/Users/zhanggenqi/Desktop/DSP_in_VLSI/HW1/TESTBED/pattern/P1_GOLDEN_O.dat"
    `define PAT_LEN 5
`elsif I0_SOFTWARE // VERIFY USING SOFTWARE ALGORITHM GENERATE DATA
    `define DATA_I_PATH  "C:/Users/zhanggenqi/Desktop/DSP_in_VLSI/HW1/TESTBED/pattern/P0_DATA_I.dat"
    `define GOLDEN_O_PATH  "C:/Users/zhanggenqi/Desktop/DSP_in_VLSI/HW1/SOFTWARE/result/P0_SOFTWARE_RESULT_O.dat"
    `define PAT_LEN 1000
`elsif I1_SOFTWARE // VERIFY USING SOFTWARE ALGORITHM GENERATE DATA
    `define DATA_I_PATH  "C:/Users/zhanggenqi/Desktop/DSP_in_VLSI/HW1/TESTBED/pattern/P1_DATA_I.dat"
    `define GOLDEN_O_PATH  "C:/Users/zhanggenqi/Desktop/DSP_in_VLSI/HW1/SOFTWARE/result/P1_SOFTWARE_RESULT_O.dat"
    `define PAT_LEN 5
`else
    `define DATA_I_PATH  "C:/Users/zhanggenqi/Desktop/DSP_in_VLSI/HW1/TESTBED/pattern/P0_DATA_I.dat"
    `define GOLDEN_O_PATH  "C:/Users/zhanggenqi/Desktop/DSP_in_VLSI/HW1/TESTBED/pattern/P0_GOLDEN_O.dat"
    `define PAT_LEN 1000
`endif
```

I0_SOFTWARE result:

```
INFO: [Wavedata 42-604] Simulation restarted
run all
----------------------------------------------
-               ALL PASS!                    -
----------------------------------------------
$finish called at time : 801250 ns : File "C:/Users/zhanggenqi/Desktop/DSP_in_VLSI/HW1/TESTBED/testbench.v" Line 189
close_sim
```

I1_SOFTWARE result:

```
INFO: [Wavedata 42-604] Simulation restarted
run all
------------------------------------------------
-                ALL PASS!                     -
------------------------------------------------
$finish called at time : 5250 ns : File "C:/Users/zhanggenqi/Desktop/DSP_in_VLSI/HW1/TESTBED/testbench.v" Line 189
```

4. Show that your design can generate required top-7 output in descending order every 8 clock cycles as indicated in Fig. 3 in the timing diagram with proper input signal “BlkIn” and output indicator “OutValid”. If it is too long, please cut it down into several segments to make the numerical expressions in the timing diagram clear. If the numerical expressions are hard to distinguish, correct evaluation may not be 5 given. (30%)

5. Please use Matlab command “sort” to verify your results of your randomly generated sequence and compare to the Verilog simulation results (10%).

Matlab generate test pattern code:

HW1.m

```
num_patterns = 5;  % Number of random sequences
num_elements = 32; % Number of elements per sequence

data_file = fopen('../pattern/P1_DATA_I.dat', 'w');
golden_file = fopen('../pattern/P1_GOLDEN_O.dat', 'w');

for p = 1:num_patterns
    fprintf(data_file, '// Pattern %d:\n', p-1);
    sequence = zeros(1, num_elements);
    for i = 1:num_elements
        num = randi([-128, 127]); % Generate a random 8-bit signed integer
        sequence(i) = num;
        bin_str = dec2bin(typecast(int8(num), 'uint8'), 8); % Convert to 8-bit binary
        fprintf(data_file, '%s  // %d\n', bin_str, num);
    end
    fprintf(data_file, '\n');

    % Extract top 7 golden data (highest values)
    golden_values = sort(sequence, 'descend');
    golden_values = golden_values(1:7);
    fprintf(golden_file, '// Pattern %d:\n', p-1);
    for j = 1:7
        bin_str = dec2bin(typecast(int8(golden_values(j)), 'uint8'), 8);
        fprintf(golden_file, '%s  // %d\n', bin_str, golden_values(j));
    end
    fprintf(golden_file, '\n');
end

fclose(data_file);
fclose(golden_file);
```

Testbench test pattern selection:

```
`timescale 1ns/10ps
`define PERIOD    100.0
`define MAX_CYCLE 100000
`define RST_DELAY 2.0
`define I1_GOLDEN

`ifdef I0_GOLDEN // VERIFY USING GLODEN DATA
    `define DATA_I_PATH  "C:/Users/zhanggenqi/Desktop/DSP_in_VLSI/HW1/TESTBED/pattern/P0_DATA_I.dat"
    `define GOLDEN_O_PATH  "C:/Users/zhanggenqi/Desktop/DSP_in_VLSI/HW1/TESTBED/pattern/P0_GOLDEN_O.dat"
    `define PAT_LEN 1000
`elsif I1_GOLDEN // VERIFY USING GLODEN DATA
    `define DATA_I_PATH  "C:/Users/zhanggenqi/Desktop/DSP_in_VLSI/HW1/TESTBED/pattern/P1_DATA_I.dat"
    `define GOLDEN_O_PATH  "C:/Users/zhanggenqi/Desktop/DSP_in_VLSI/HW1/TESTBED/pattern/P1_GOLDEN_O.dat"
    `define PAT_LEN 5
`elsif I0_SOFTWARE // VERIFY USING SOFTWARE ALGORITHM GENERATE DATA
    `define DATA_I_PATH  "C:/Users/zhanggenqi/Desktop/DSP_in_VLSI/HW1/TESTBED/pattern/P0_DATA_I.dat"
    `define GOLDEN_O_PATH  "C:/Users/zhanggenqi/Desktop/DSP_in_VLSI/HW1/SOFTWARE/result/P0_SOFTWARE_RESULT_O.dat"
    `define PAT_LEN 1000
`elsif I1_SOFTWARE // VERIFY USING SOFTWARE ALGORITHM GENERATE DATA
    `define DATA_I_PATH  "C:/Users/zhanggenqi/Desktop/DSP_in_VLSI/HW1/TESTBED/pattern/P1_DATA_I.dat"
    `define GOLDEN_O_PATH  "C:/Users/zhanggenqi/Desktop/DSP_in_VLSI/HW1/SOFTWARE/result/P1_SOFTWARE_RESULT_O.dat"
    `define PAT_LEN 5
`else
    `define DATA_I_PATH  "C:/Users/zhanggenqi/Desktop/DSP_in_VLSI/HW1/TESTBED/pattern/P0_DATA_I.dat"
    `define GOLDEN_O_PATH  "C:/Users/zhanggenqi/Desktop/DSP_in_VLSI/HW1/TESTBED/pattern/P0_GOLDEN_O.dat"
    `define PAT_LEN 1000
`endif
```

I0_GOLDEN result:

```
INFO: [Wavedata 42-604] Simulation restarted
run all
------------------------------------------------
-                ALL PASS!                     -
------------------------------------------------
$finish called at time : 801250 ns : File "C:/Users/zhanggenqi/Desktop/DSP_in_VLSI/HW1/TESTBED/testbench.v" Line 189
```

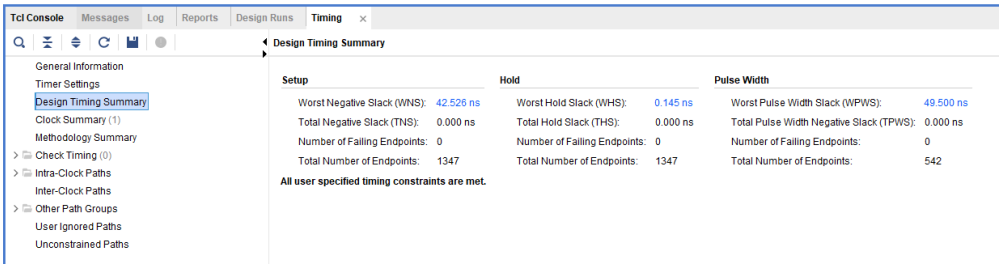
I1_GOLDEN result:

```
INFO: [Wavedata 42-604] Simulation restarted
run all
------------------------------------------------
-                ALL PASS!                     -
------------------------------------------------
$finish called at time : 5250 ns : File "C:/Users/zhanggenqi/Desktop/DSP_in_VLSI/HW1/TESTBED/testbench.v" Line 189
```

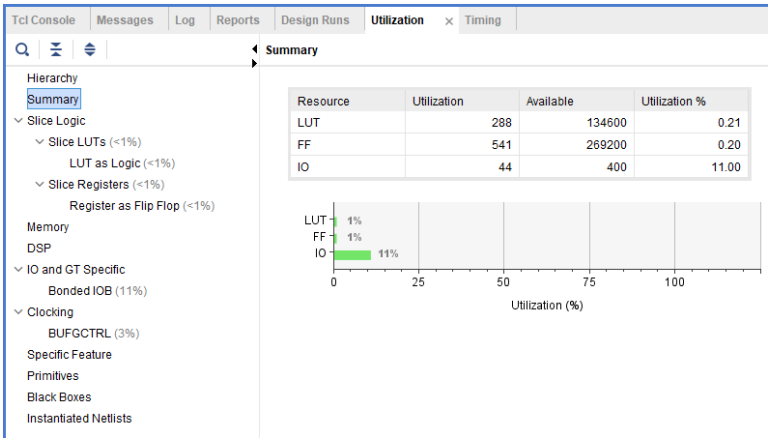
6. Synthesis your design in Q4. Show the number of adders/subtractors/comparators in your design. Sum them up together to see if it matches with your block diagram. (10%) (Because you may use counters to control your circuits, we will not ask that the two values should be exactly the same. However, there should not be a large difference.)

Clk period: 100ns

Synthesis timing result:



Synthesis utilization result:



Number of adders/subtractors/comparators: 12 (matched)

```
Report Cell Usage:
+------+-------+------+
|      |Cell   |Count |
+------+-------+------+
|1     |BUFG   |     1|
|2     |CARRY4 |    12|
|3     |LUT2   |     2|
|4     |LUT3   |    53|
|5     |LUT4   |   108|
|6     |LUT5   |    86|
|7     |LUT6   |   116|
|8     |FDCE   |   533|
|9     |FDPE   |     8|
|10    |IBUF   |    35|
|11    |OBUF   |     9|
+------+-------+------+
```