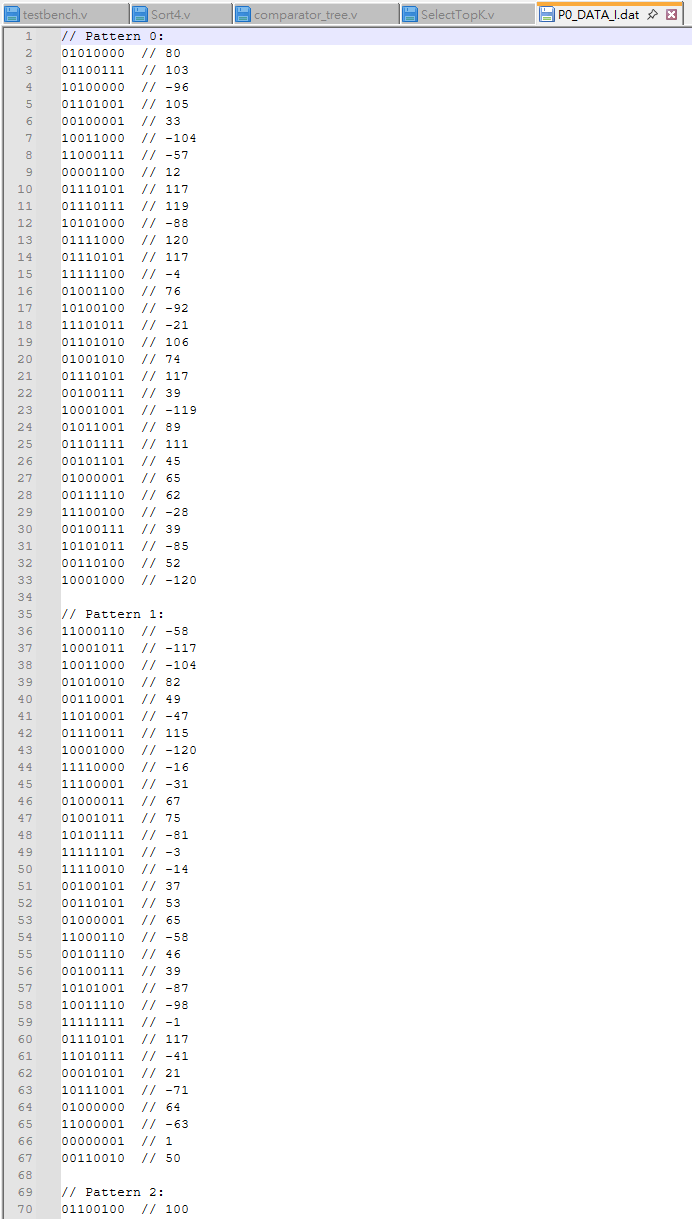
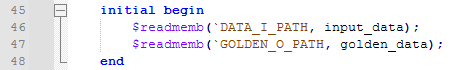
DSP in VLSI

HW1

電子所ICS組, R13943015, 張根齊

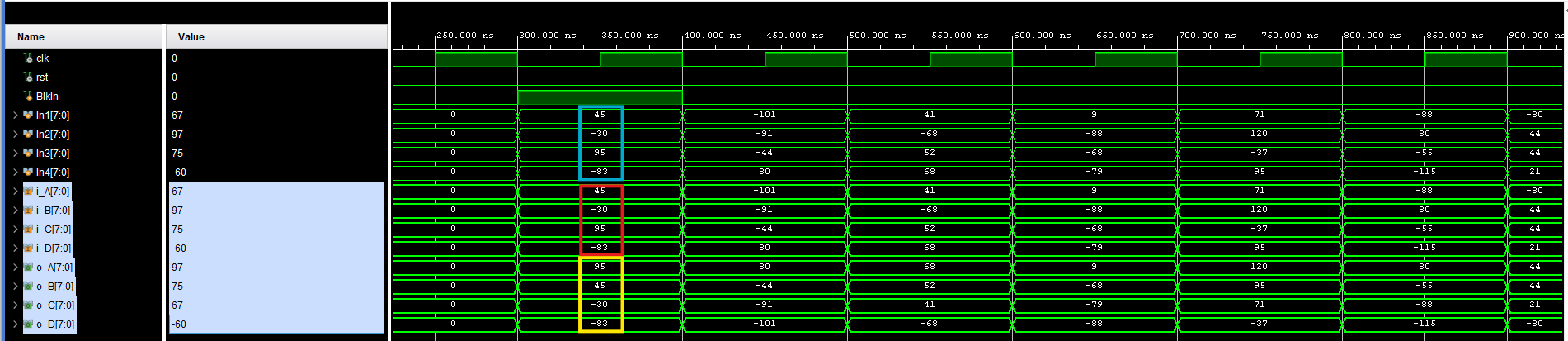
1. First, generate several random sequences with 32 elements [𝑥1 𝑥2 … 𝑥32] in a block between -128 and 127 as your test inputs and prepare as the testbench. (10%)

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.

1. 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.



1. 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%)

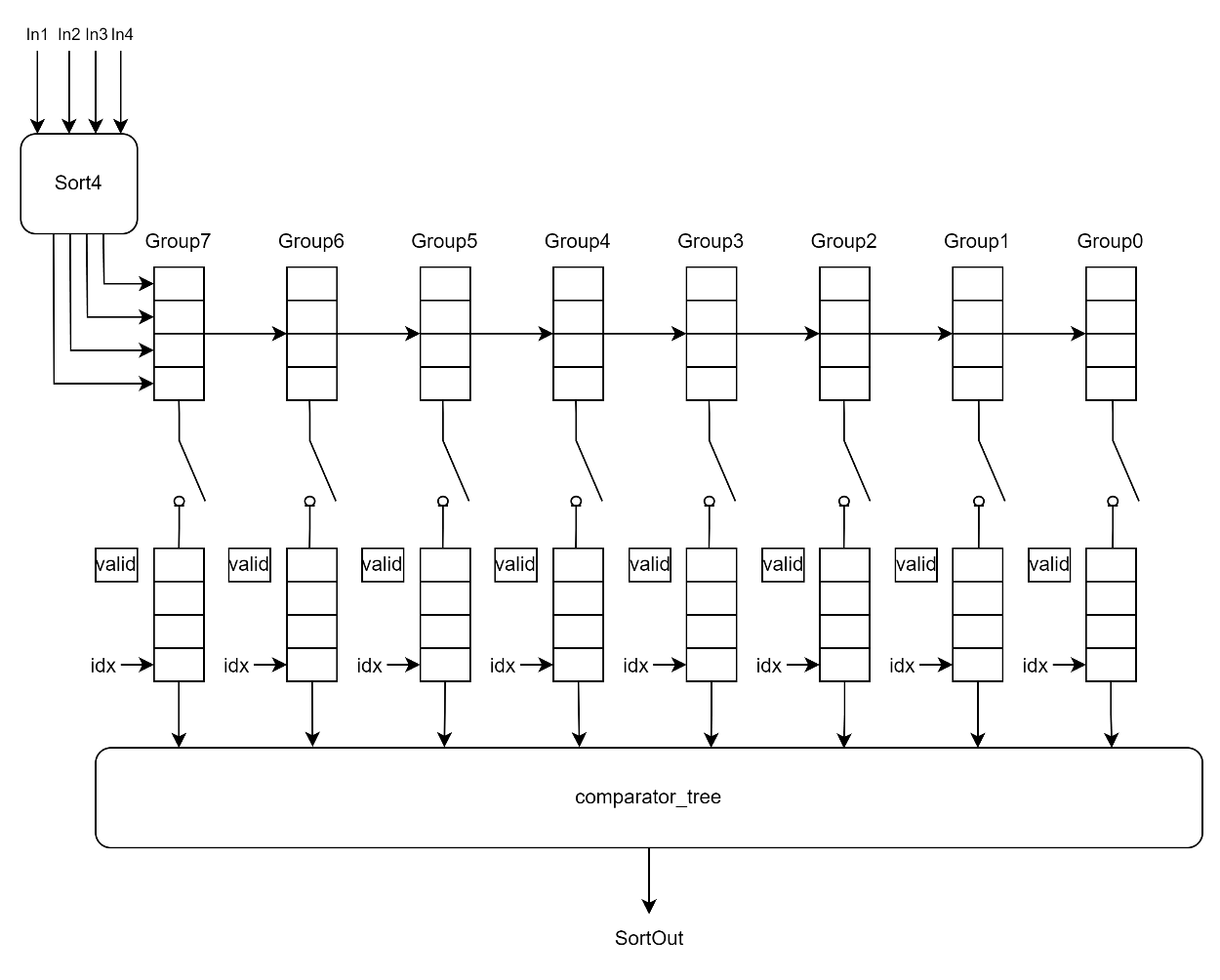
3.2. Show the timing diagram of your behavior simulation results (10%)

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

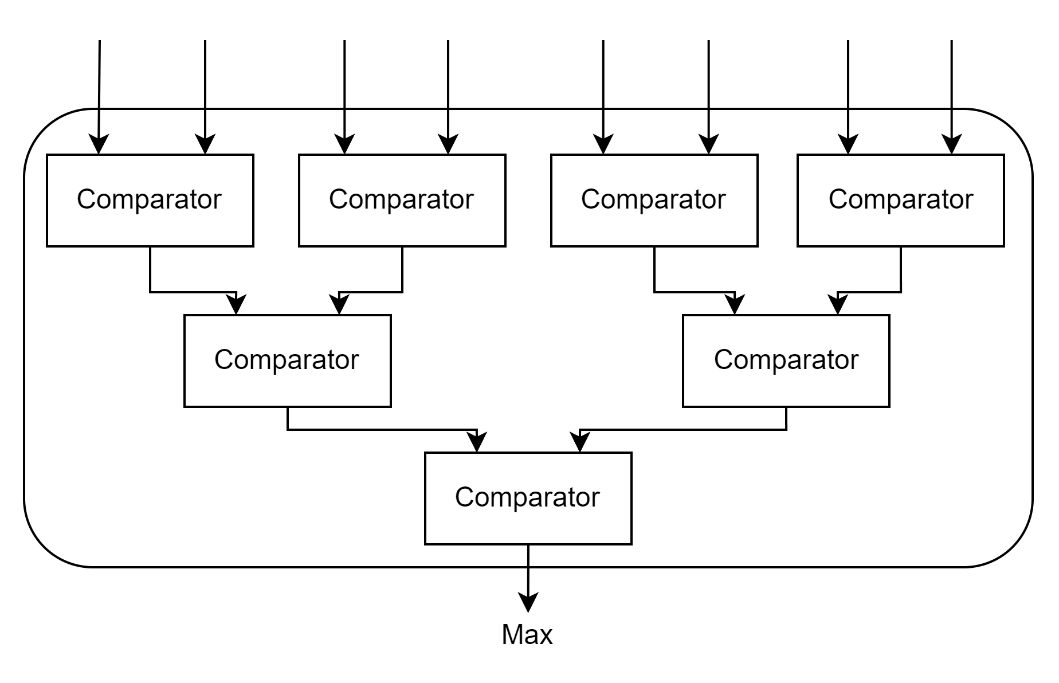
**3.1:**

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.

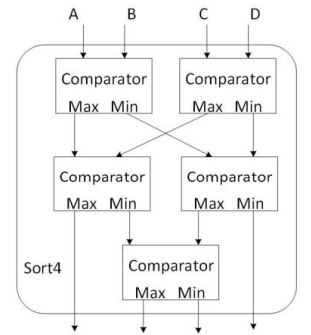
SelectTopK:



comparator\_tree:



Sort4:

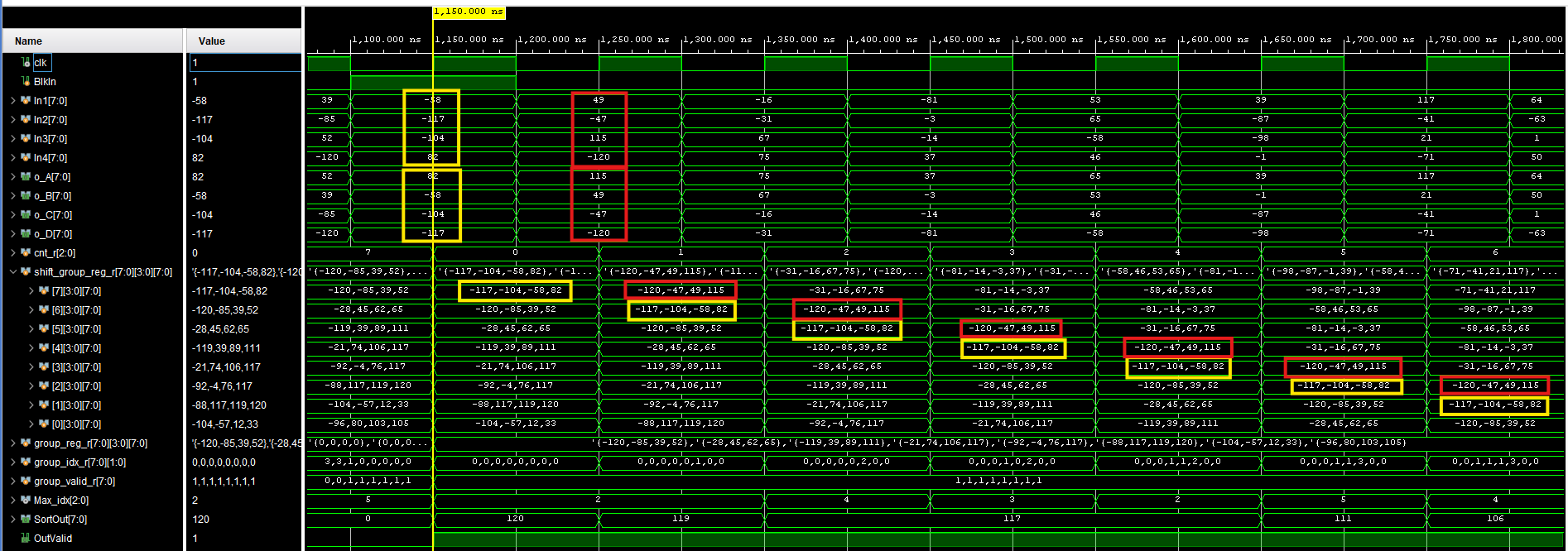


number of adders/subtractors/comparators

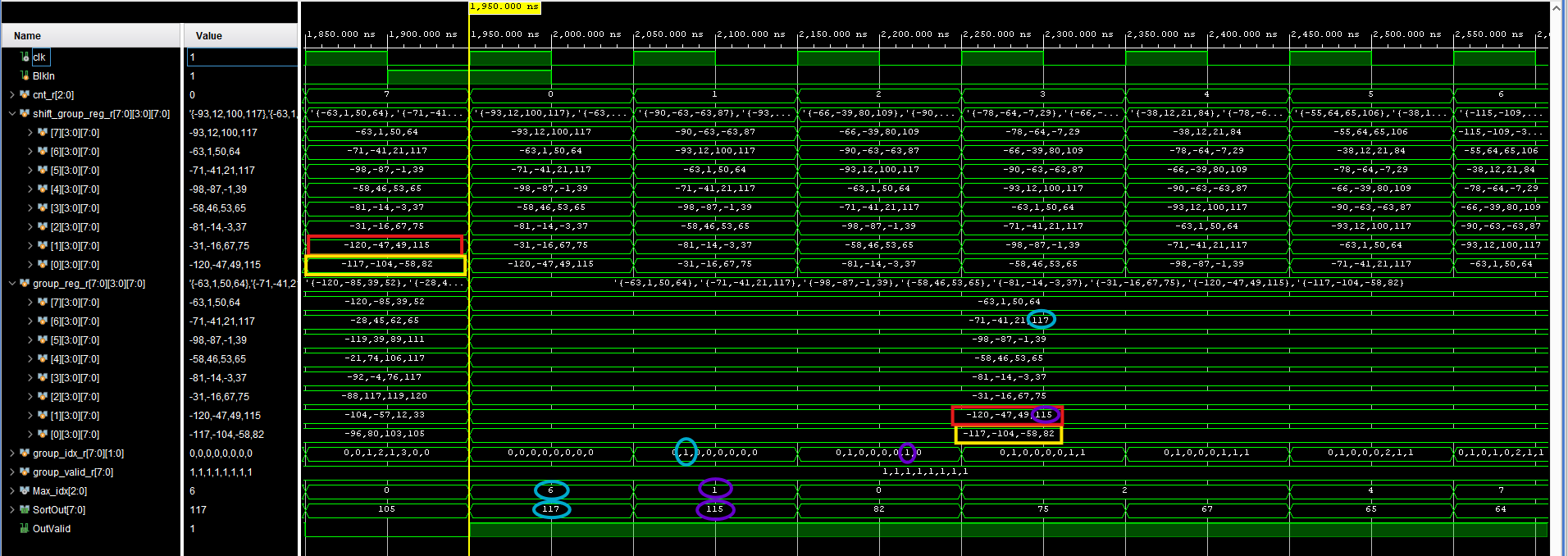
|  |  |  |  |
| --- | --- | --- | --- |
| module | Sort4 | comparator\_tree | Total |
| #comparator | 5 | 7 | 12 |

**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.



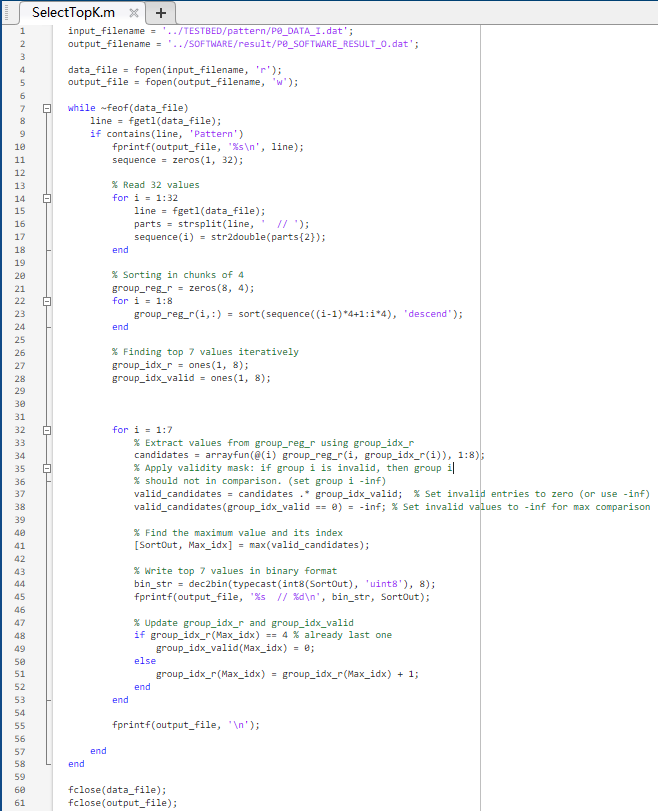
After collecting Group 0 to Group 7, the shift registers *shift\_group\_reg\_r* are transferred to another set of group registers *group\_reg\_r*. Finally, the comparator tree determines the maximum value among the eight groups and outputs it over seven cycles to obtain the top 7 values. After outputting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



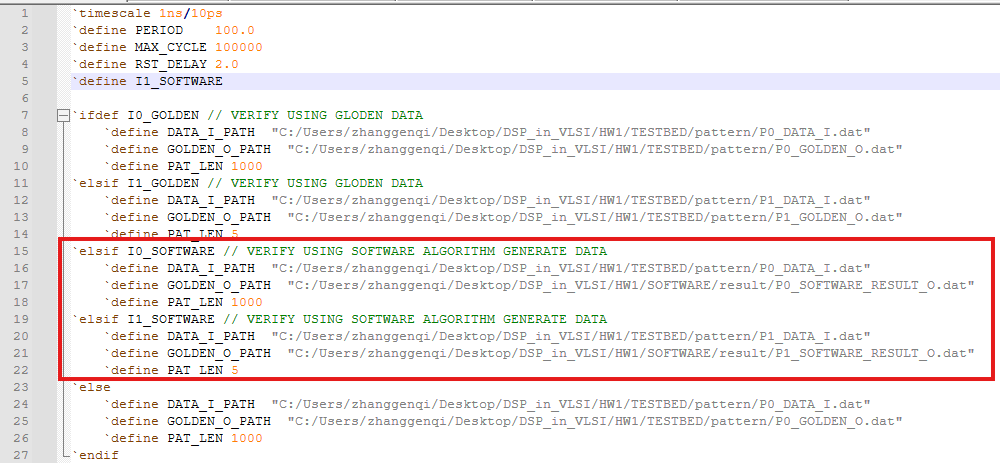
**3.3:**

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

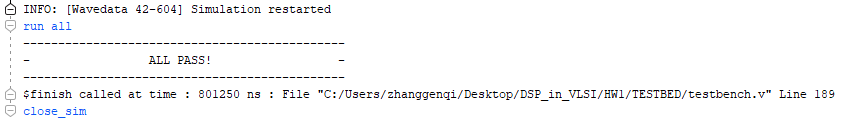
Matlab software code:

****

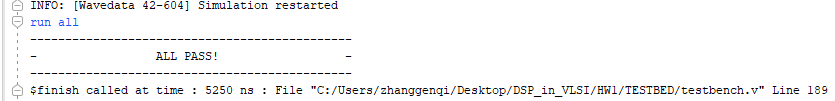
Testbench test pattern selection:

****

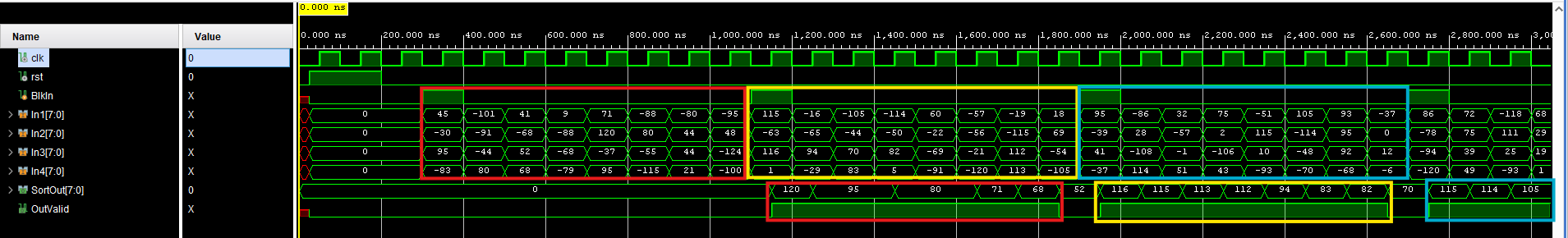
I0\_SOFTWARE result:



I1\_SOFTWARE result:

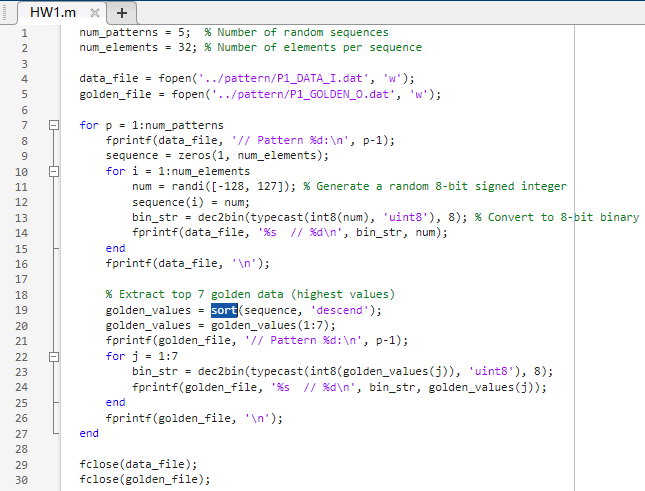
****

1. 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%)

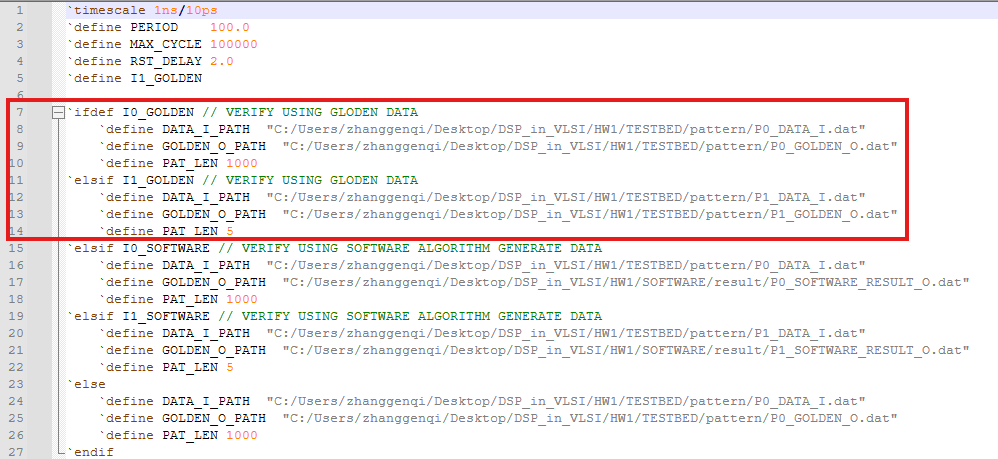
****

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

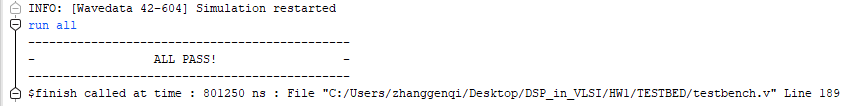
Matlab generate test pattern code:

****

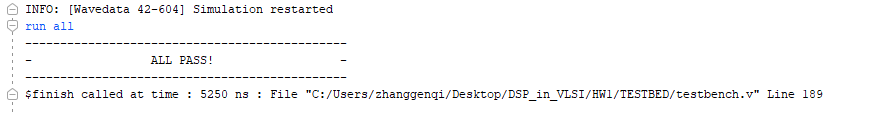
Testbench test pattern selection:

****

I0\_GOLDEN result:

****

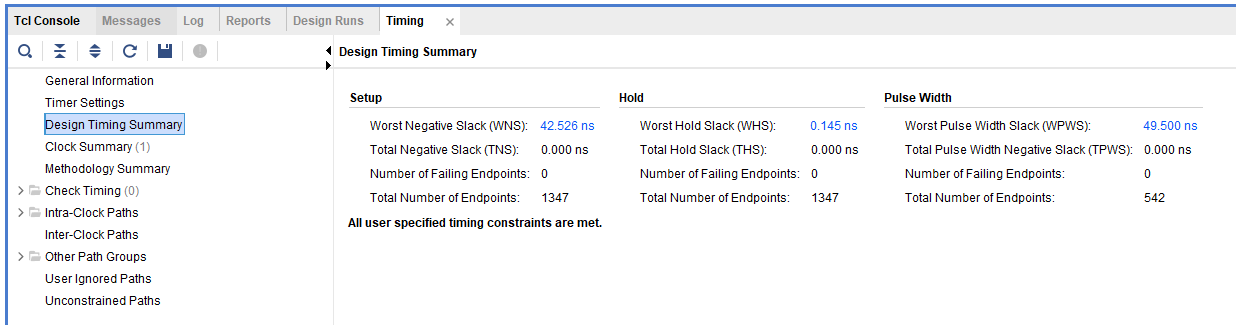
I1\_GOLDEN result:

****

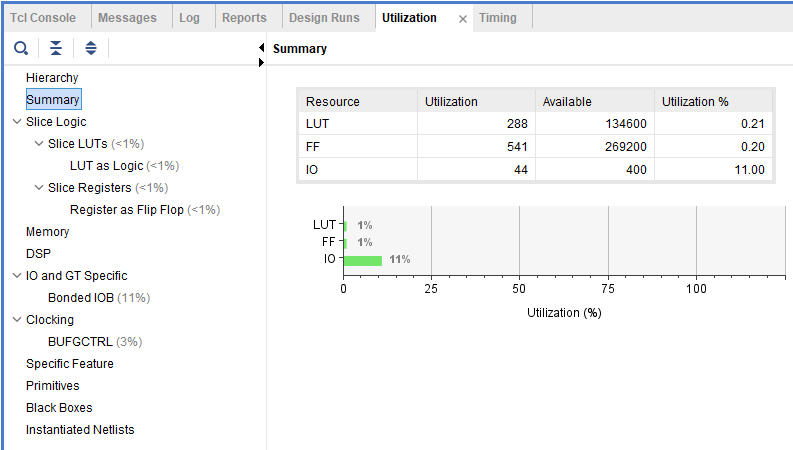
1. Synthesis your design in Q4. Show the number of adders/subtractors/comparators in your design. Sum them up together to see if it matches with your block diagram. (10%) (Because you may use counters to control your circuits, we will not ask that the two values should be exactly the same. However, there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

Clk period: 100ns

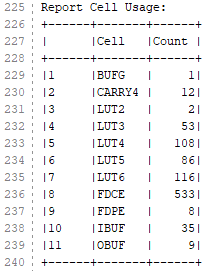
Synthesis timing result:



Synthesis utilization result:

****

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

****