****

****

Jianqi Zhang<jianqiz@usc.edu>, 1052509893

2020/04/02

Github URL: <https://github.com/JianqiZhang/EE599_Jianqi-Zhang_1052509893>

****

A Barrel Shifter is a logic circuit for shifting a word by a varying amount. It has a control

input (select bits) that specifies the number of bit positions that it shifts. A Barrel Shifter is

implemented with a sequence of shift multiplexers, each shifting a word by 2k bit positions

for different values of k. The diagram below (Figure 1) shows a pipeline clock-wise barrel

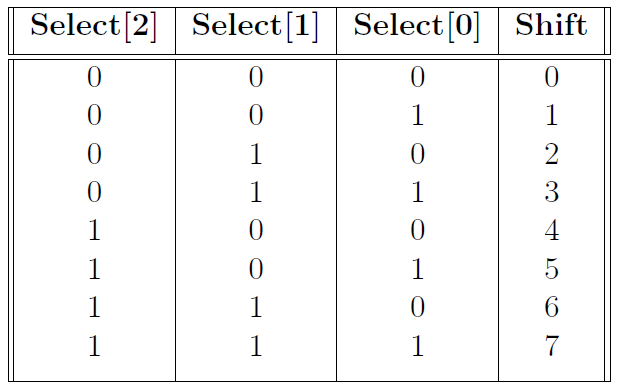
shifter for 8 inputs.



Figure 1: An Example Pipeline Barrel Shifter

Notice that we can shift the input elements by 0 - 7 elements using an implementation like Figure 1. Table 1 shows all the shifting possibilities for the above example. Similarly, a scalable Barrel Shifter with N inputs and maximum shift of N-1 can be implemented.

Table 1: All the shifting possibilities for the given example



* 

Consider only 8-bit arithmetic. **You must implement a scalable design.**

1. Implement a barrel shifter design in Verilog which takes inputs with 8 bits and shift them by (value of is passed to select bits and ).

2. For a 16 elements design with a maximum shift of 15, write a testbench and verify the waveforms.

3. Elaborate the design and include all the schematics screenshots of the modules in the report.

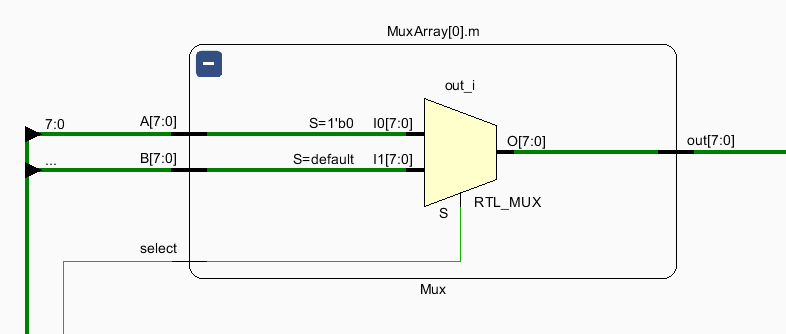
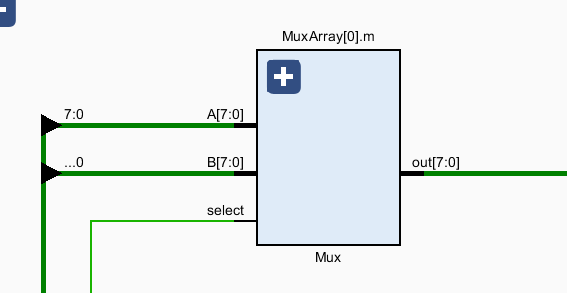
4. Synthesis the design and include the schematics screenshots in the report.

5. Generate Resource and timing estimations and include them in the report.

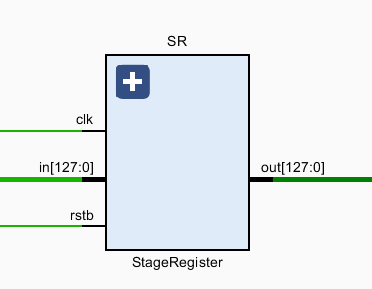
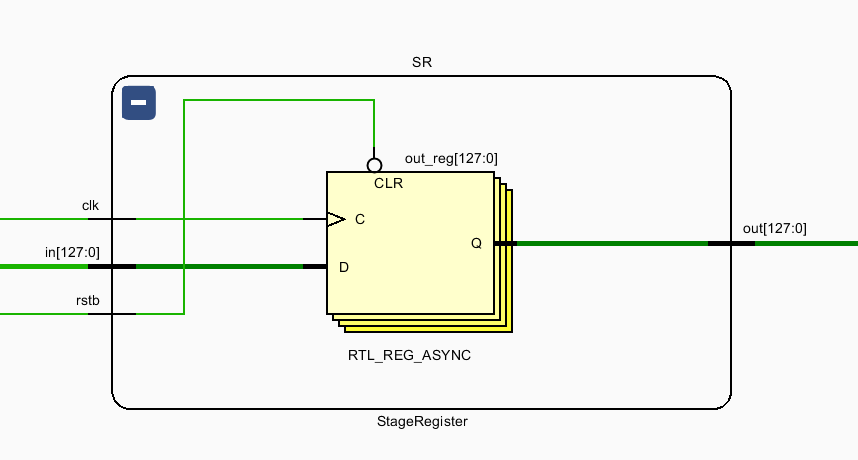
6. Redo parts 3, 4, 5 for 64 elements (with maximum shift of 63).

**Schematics’ screenshots** are shown below:

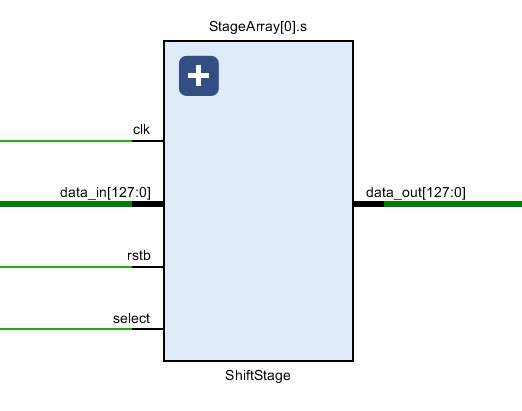
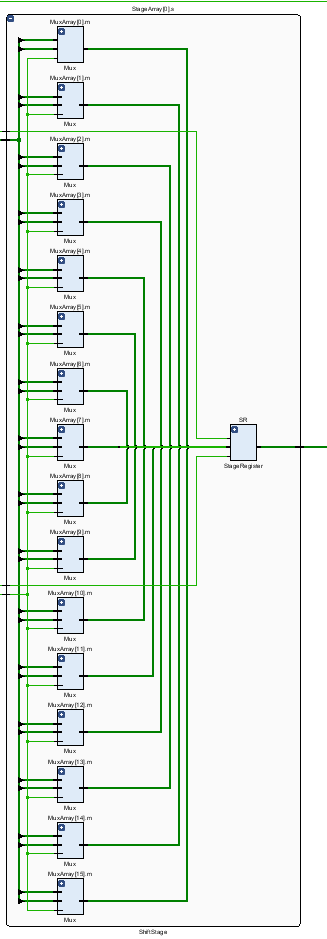
1. **2 Inputs Mux** is designed to select one 8 bits-input to output by the control signal.



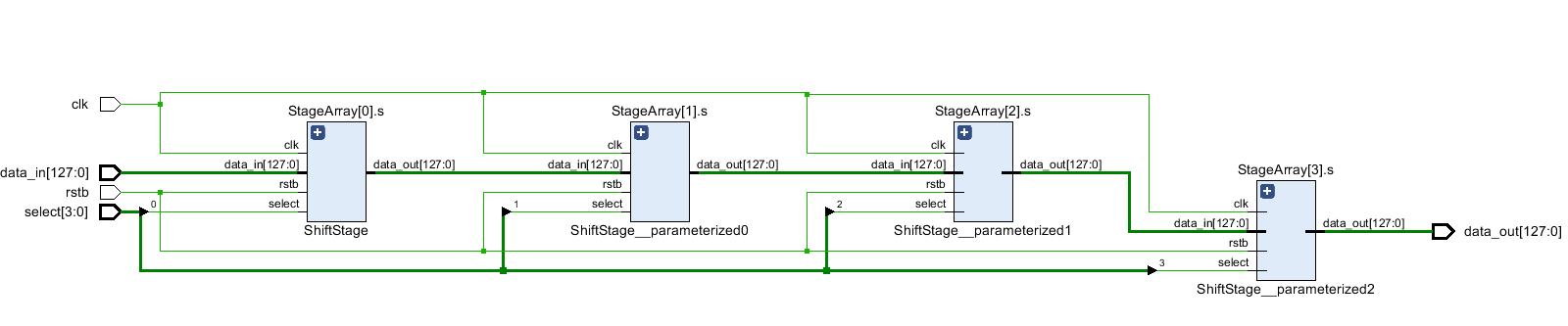
1. **Stage Register** is acted as D Filp-Flop. In every stage, all 16 elements will go through the stage register to next stage so that build a pipeline architecture.

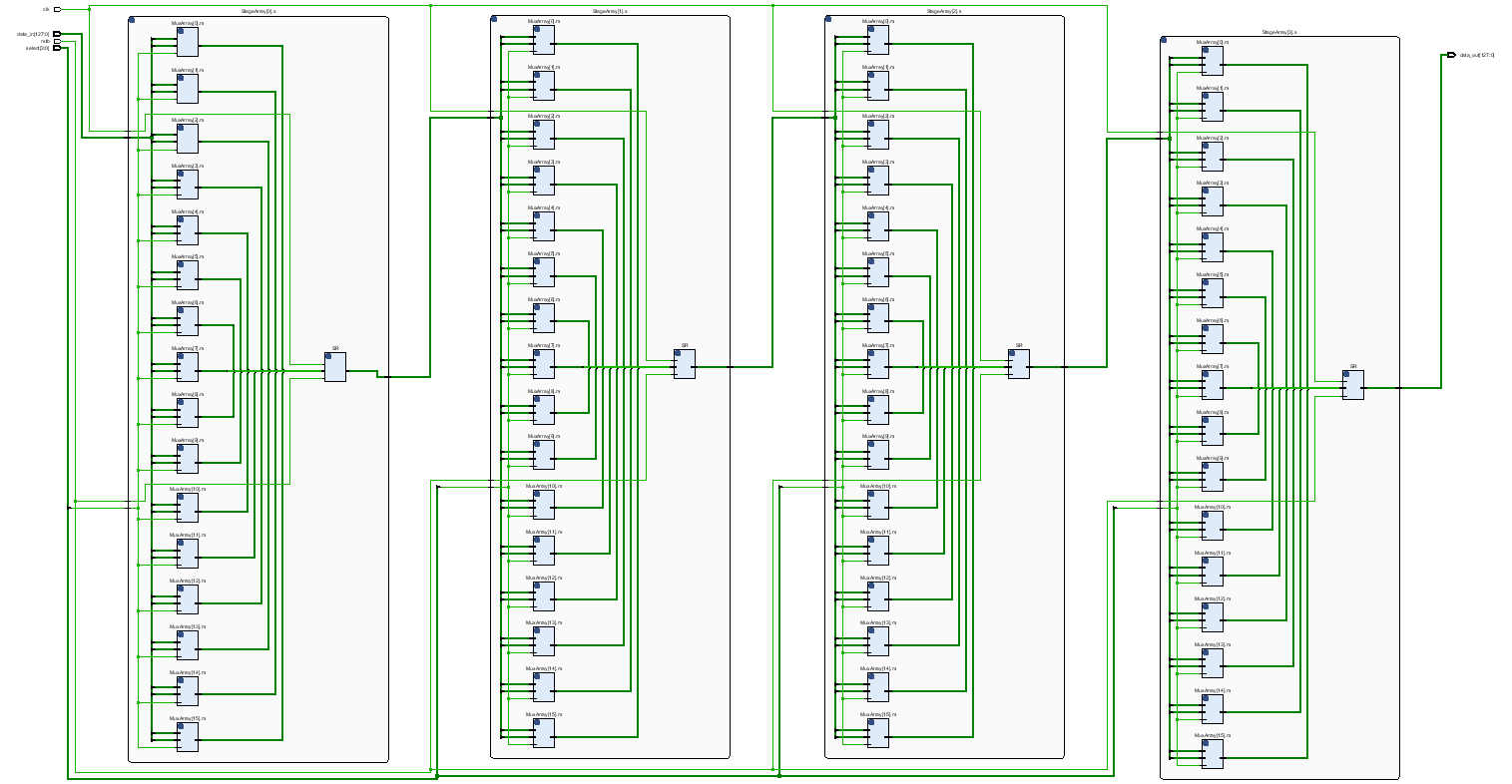
 

1. **StageArray** contains 16 Muxes and 1 stage register.

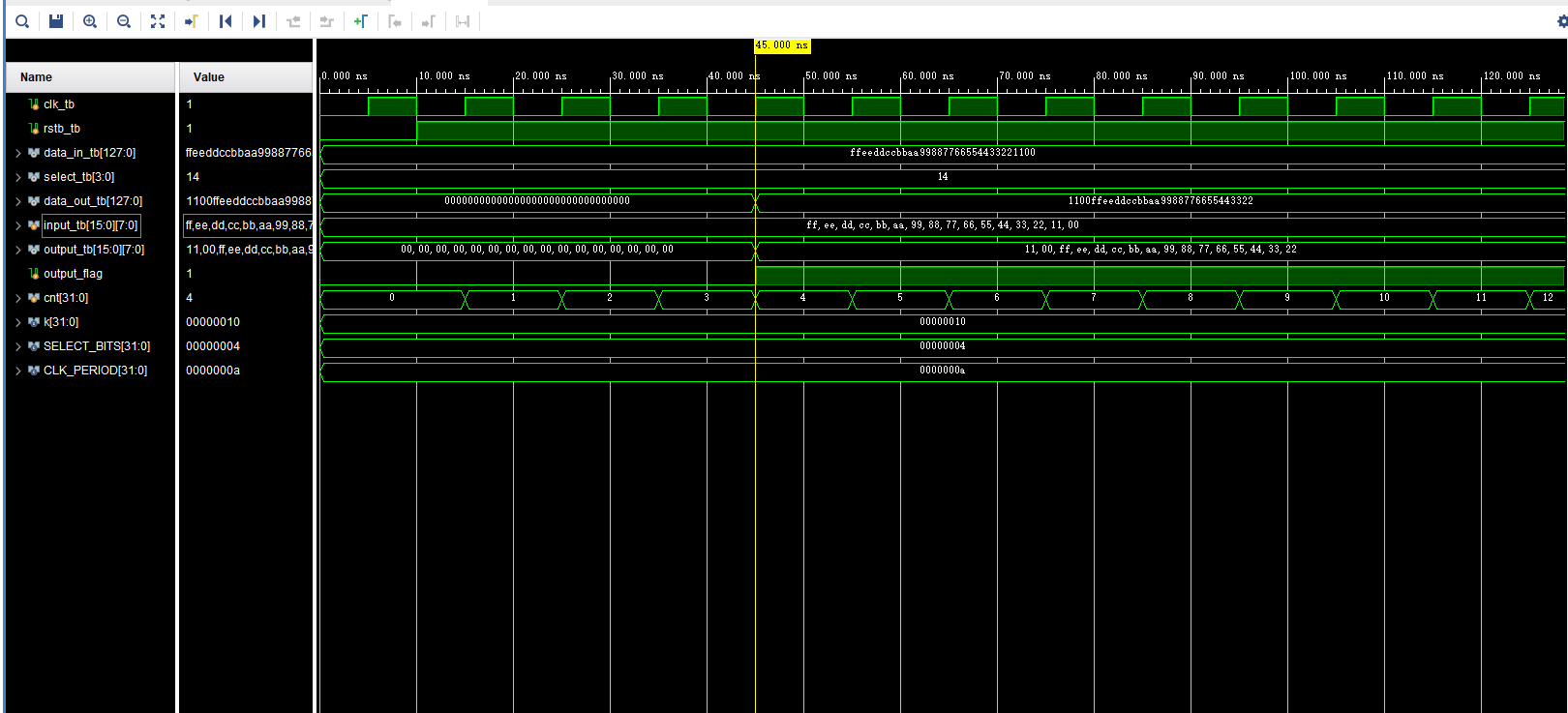
1. **BarrelShifter** is the top module of the design. For 16 elements barrel shifter, we need 4 stages connected end to end.





**The total** **schematic is shown in the Appendix.**

For a 16 elements test bench, we give 16 8-bit numbers as input, and after 2\*16 = 32 clocks, the module will output sorted results. The waveform is shown as followed:



We can see input\_tb includes 16 8-bit numbers, after 4 clocks, output\_tb gives the number array which shift by 14.

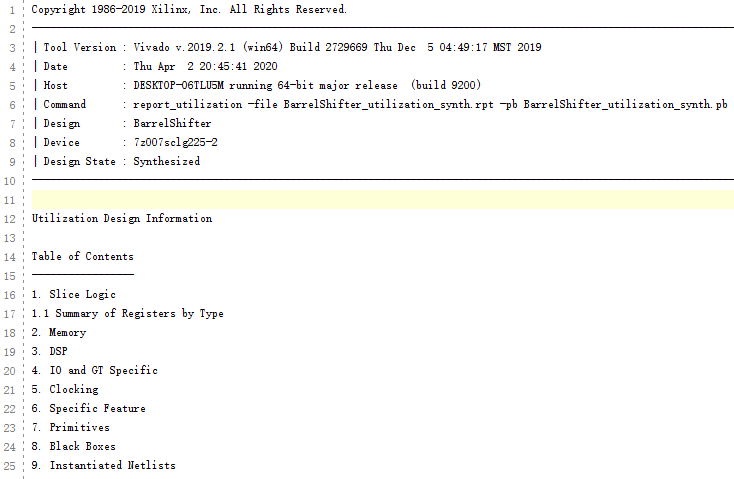
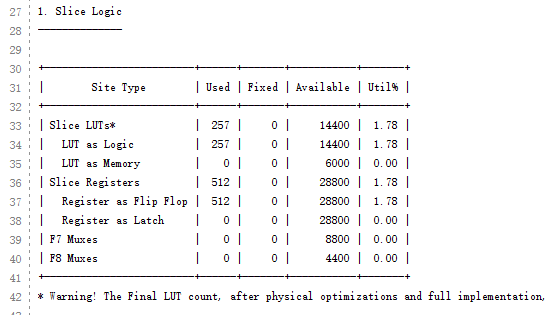
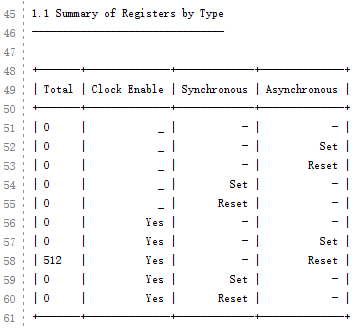
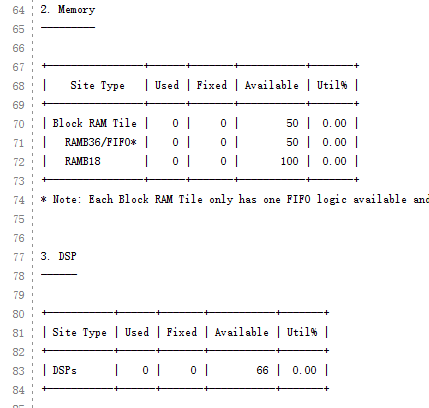
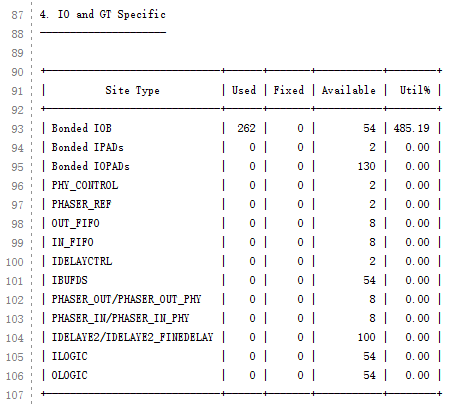
Then, we change the number of shifting.

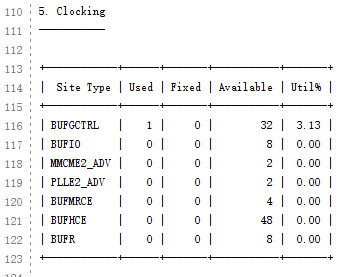
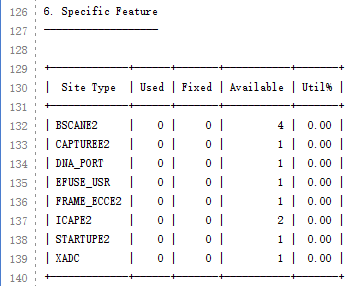
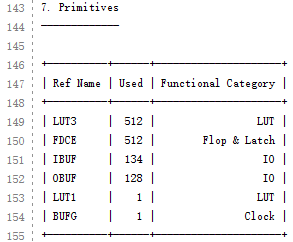


We can see input\_tb includes 16 8-bit numbers, after 4 clocks, output\_tb gives the number array which shift by 5.

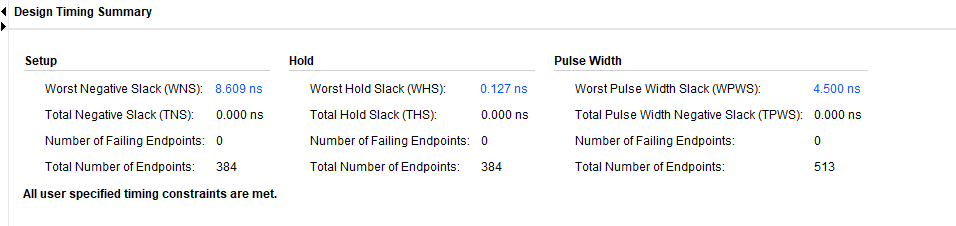
**The full synthesis schematic please see in the Appendix.**

**Resource Estimations:**

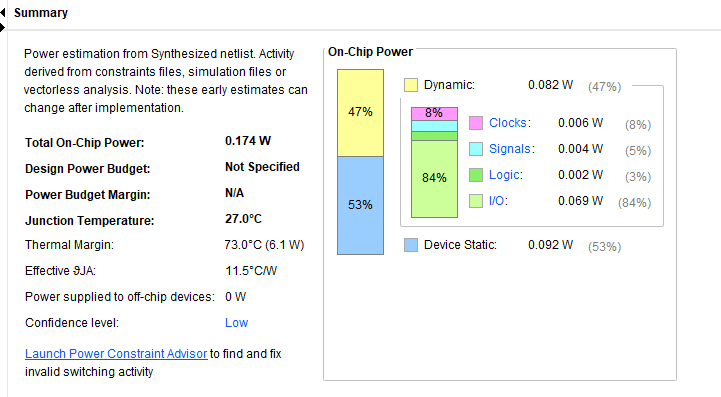
**** ****  ****

**** **** 

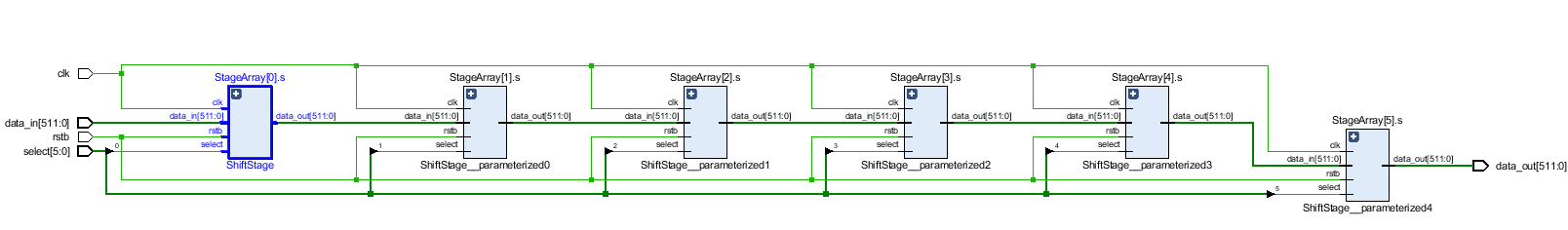
**Timing Estimation:**

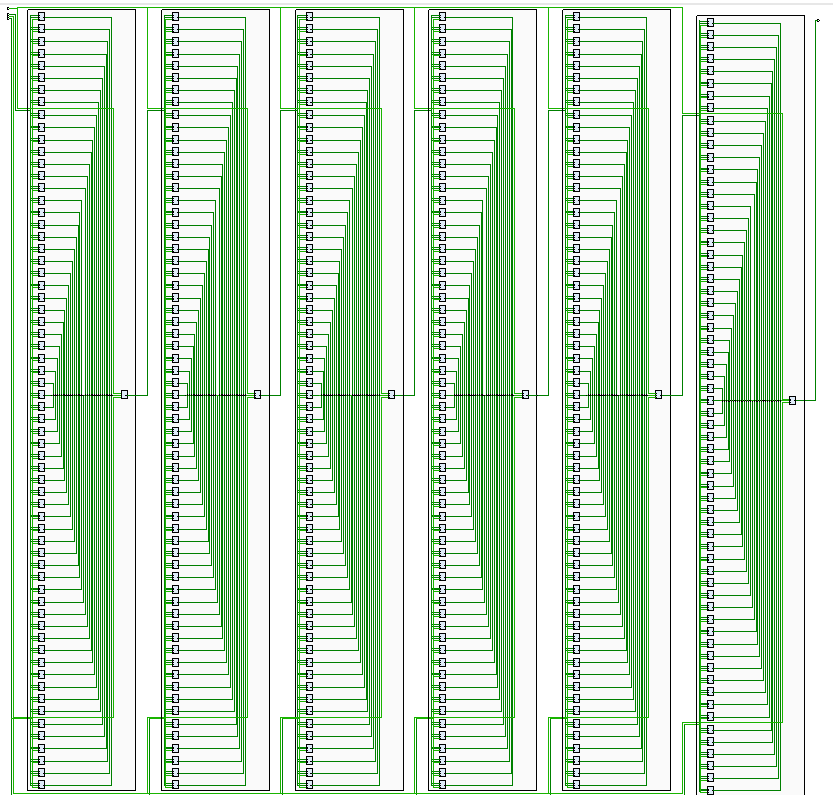
****

**Power Estimation:**

****

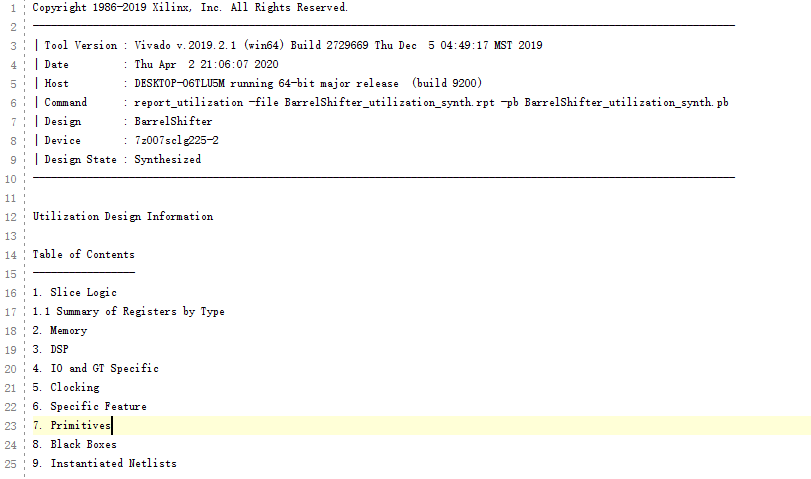
**Now redo for 64 inputs.**

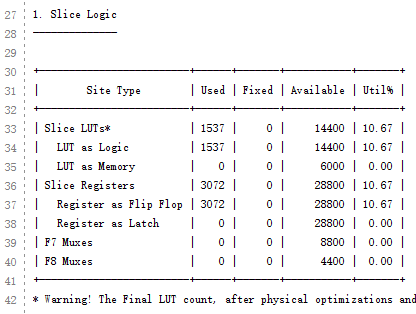
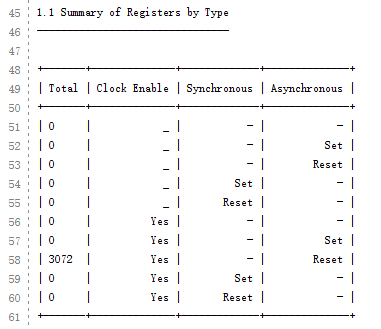
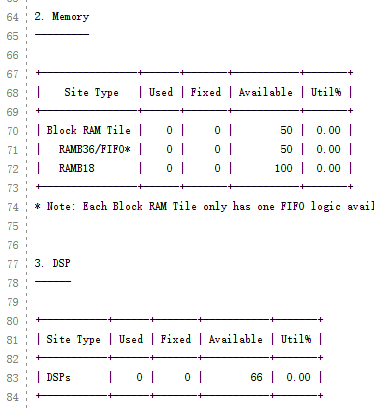
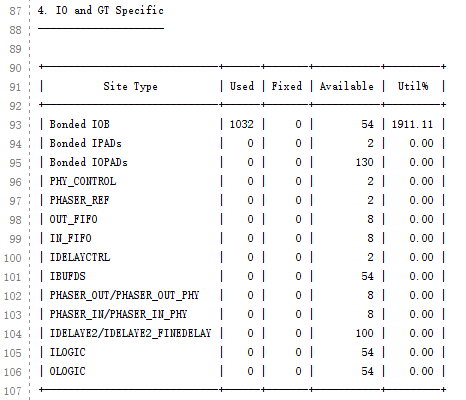
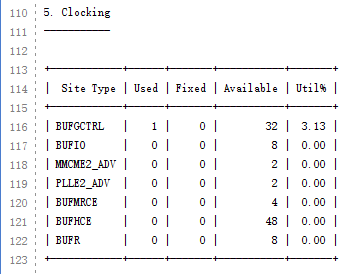
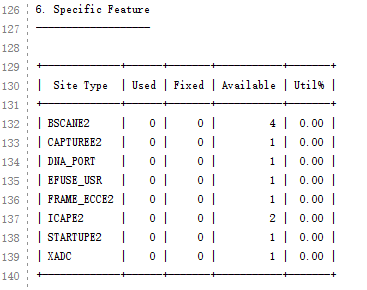
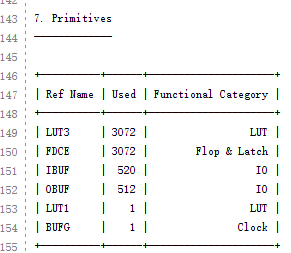
****

****

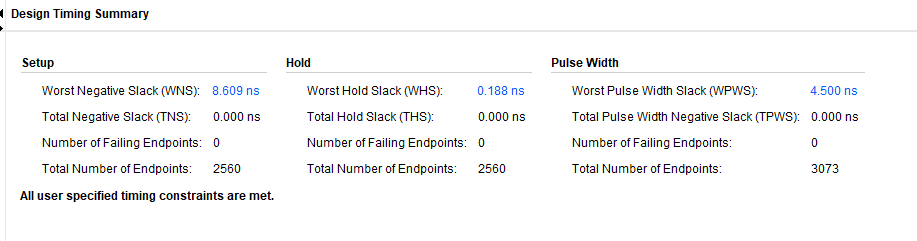
**The full synthesis schematic please see in the Appendix.**

**Resource Estimation:**

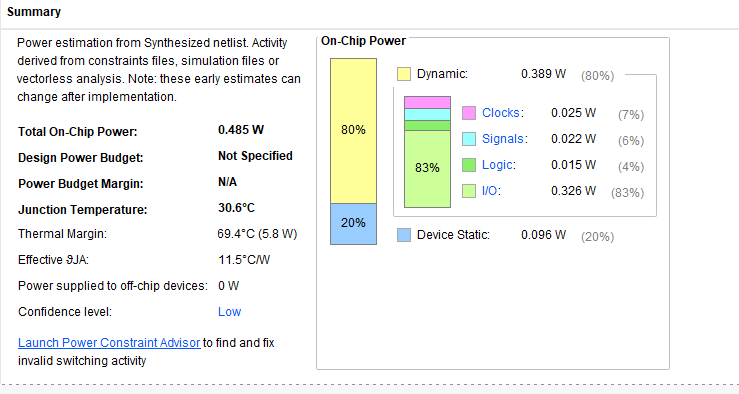
****

**** ****     

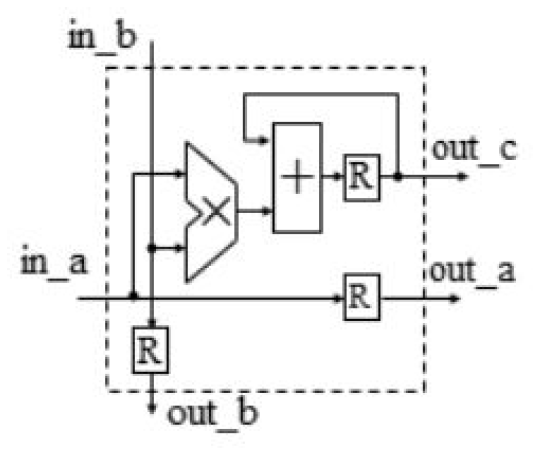
**Timing Estimation:**

****

**Power Estimation:**



****

****

Internals of the PE



Example 3 3 Systolic Array

The Systolic Architecture consists of an array of processing elements, where data flows between neighboring elements, synchronously, from different directions. Processing element takes data from Top, Left, and output the results to Right, Bottom.

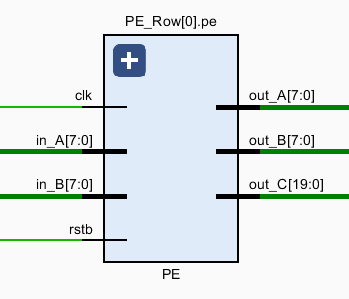
One of the main applications of Systolic Architecture is matrix multiplication. As the following figure depicts, in a, in b are inputs to the processing element and out a, out bare output to the processing element. out c is to get the output result of each processing element.

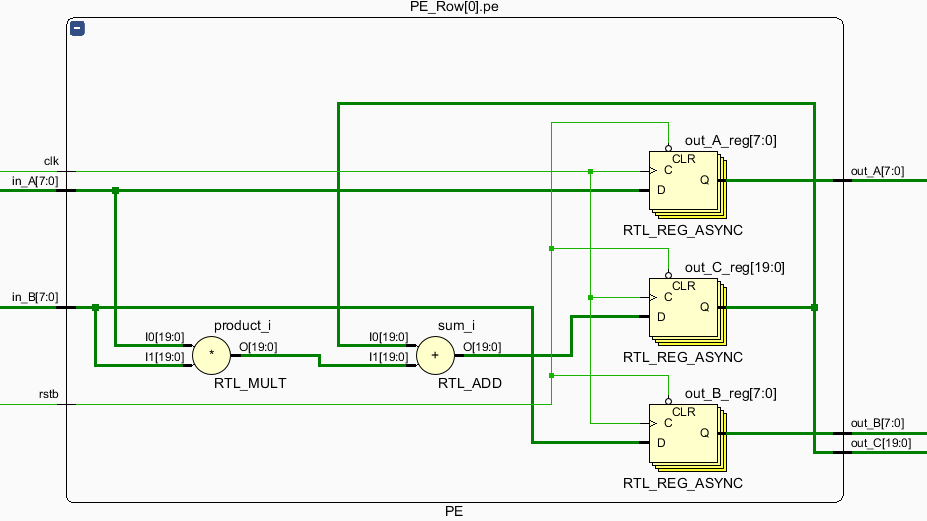
Processing elements are arranged in the form of an array. In the following example, we analyze, multiplication of 3 3 matrices, which can be easily extended. Let say the two matrices are A and B. Figure above depicts how matrix A and B are fed into PE array.

****

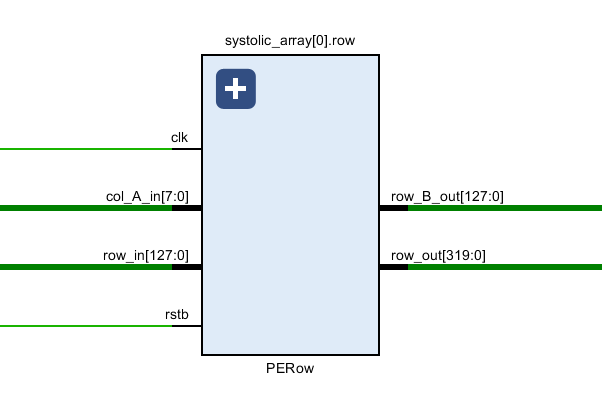
**Schematics’ screenshots** are shown below:

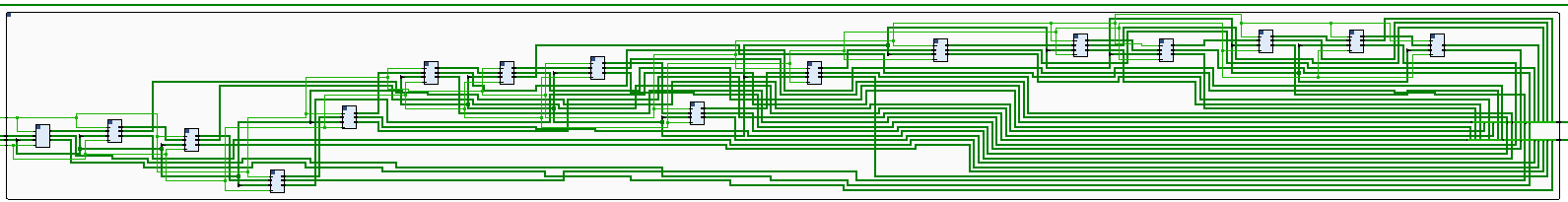
**PE** is designed to calculate the product of two 8 bits-input A and B, and accumulate all the products produced by own. Also, each PE will also transfer the two inputs to the next every clock. So there are a multiplier, an adder and three output register.



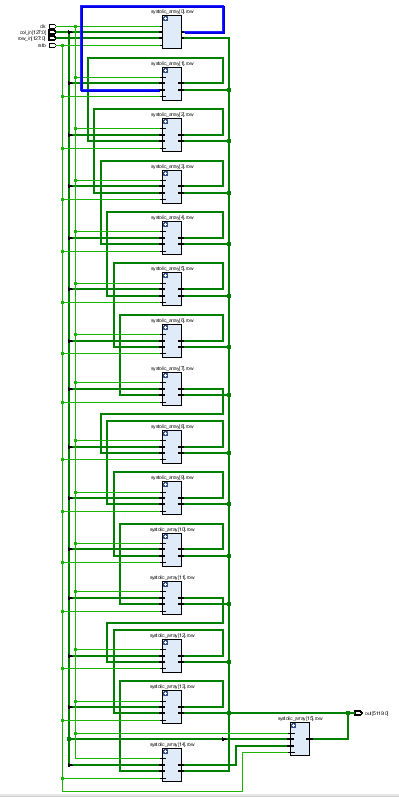
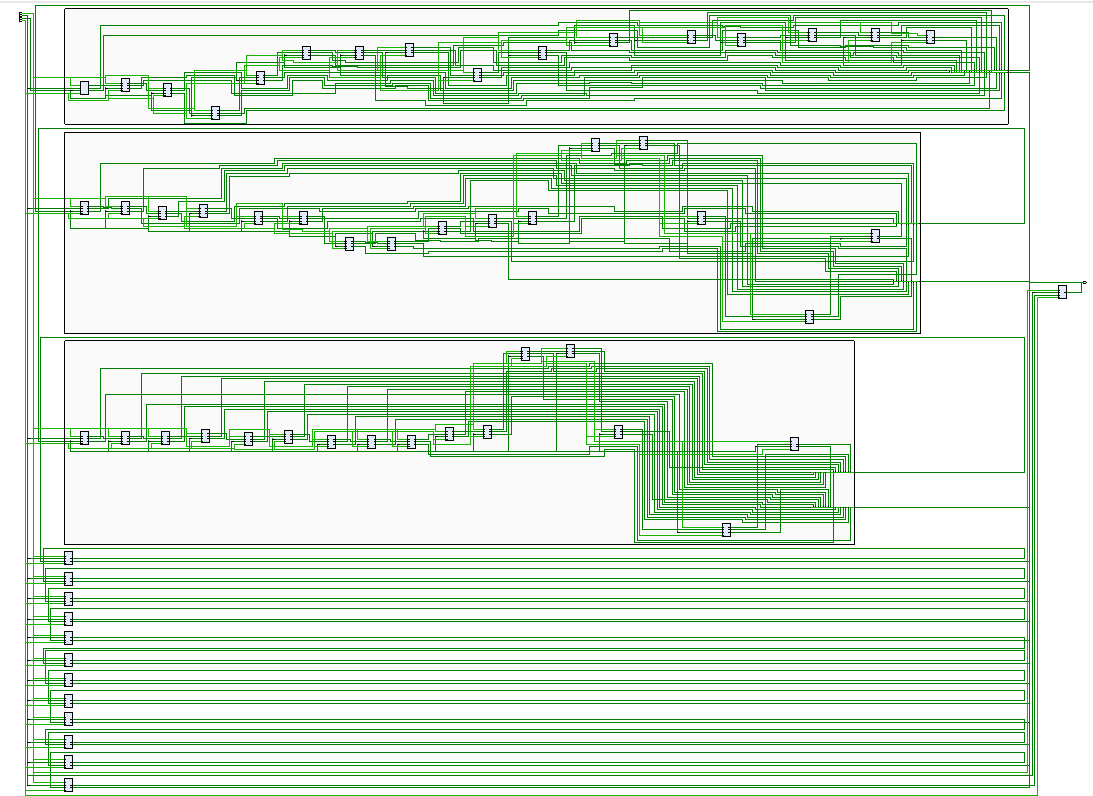


**PERow** is designed to build a row of systolic array. For a systolic array of n n size, PERow contains n PEs.

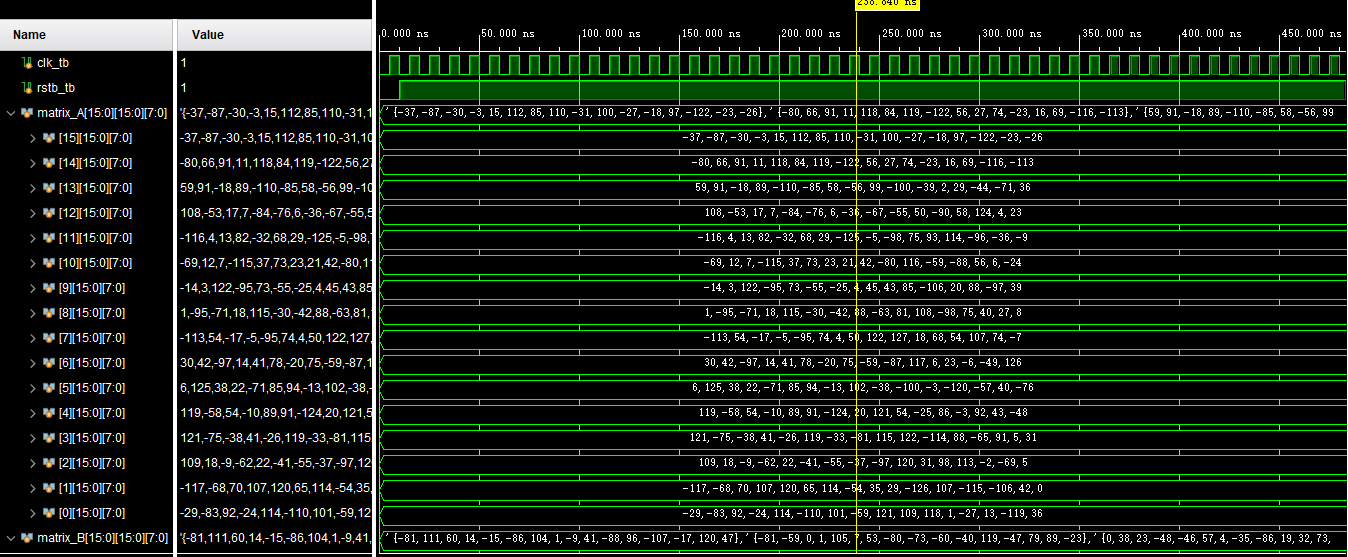


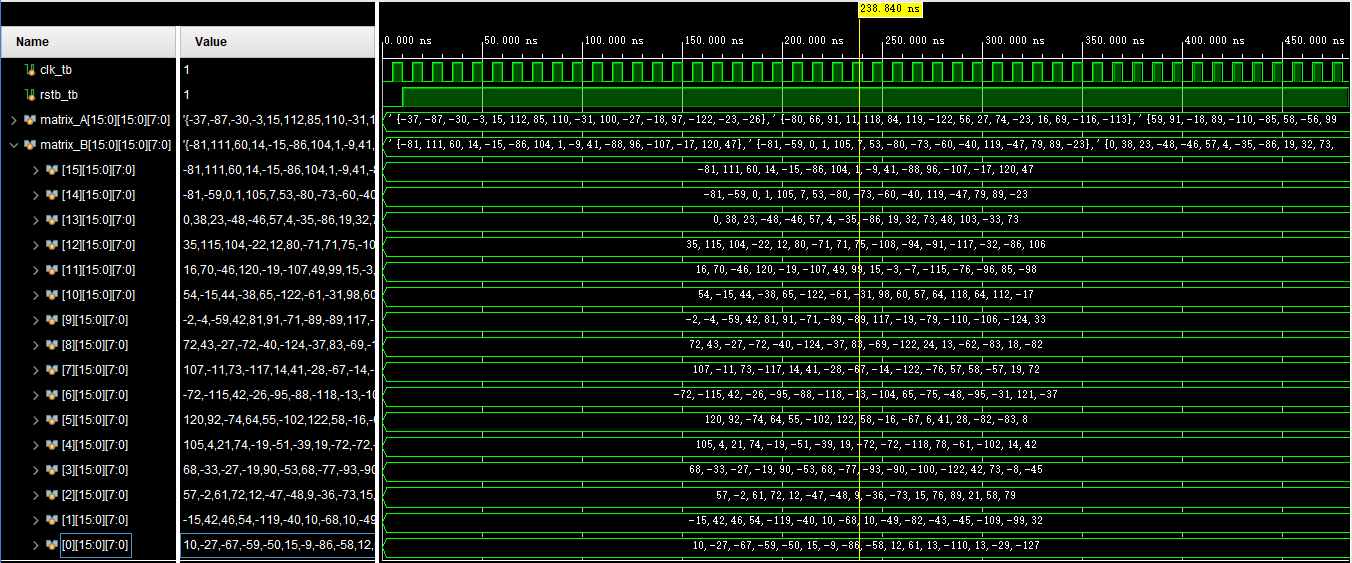


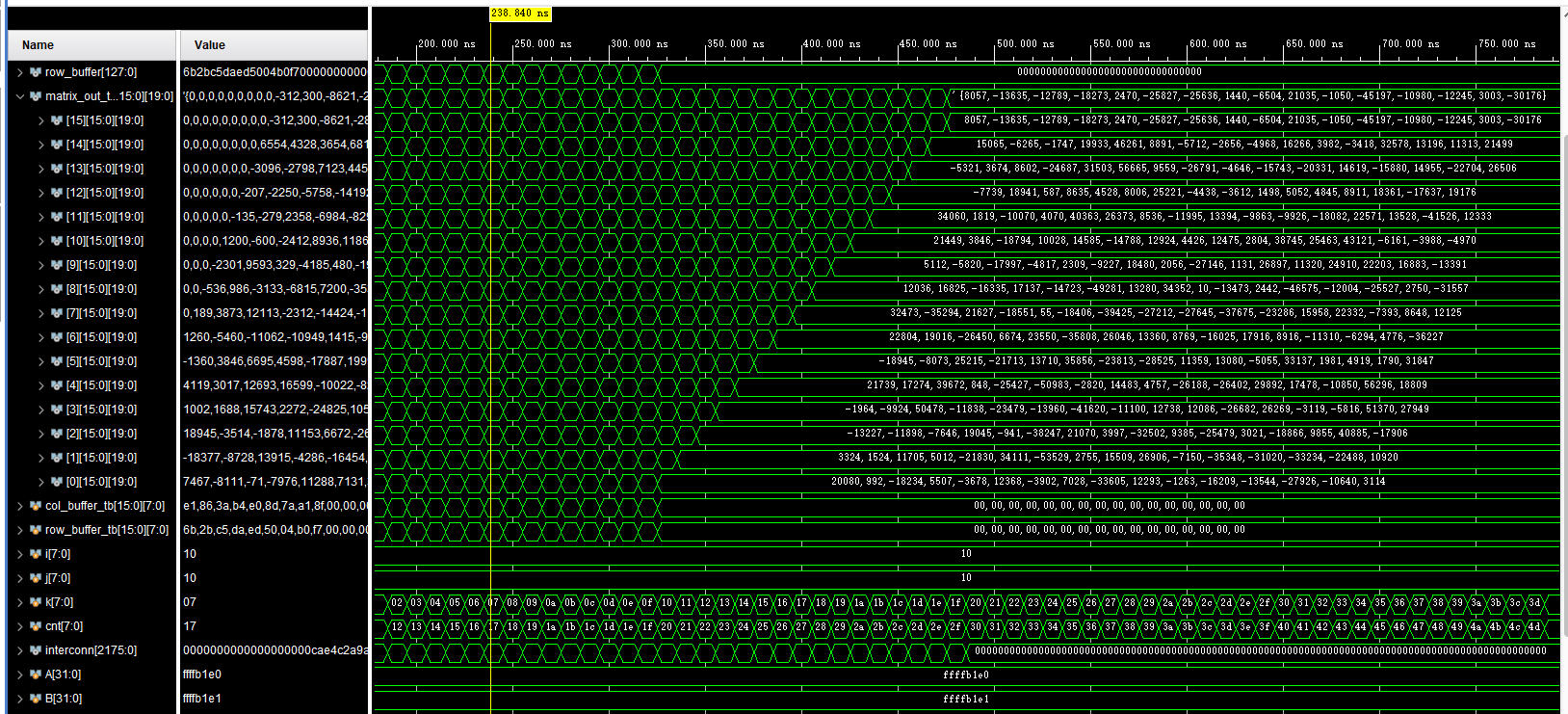
The Top Module, which is **SystolicArray** consist of n PERows.

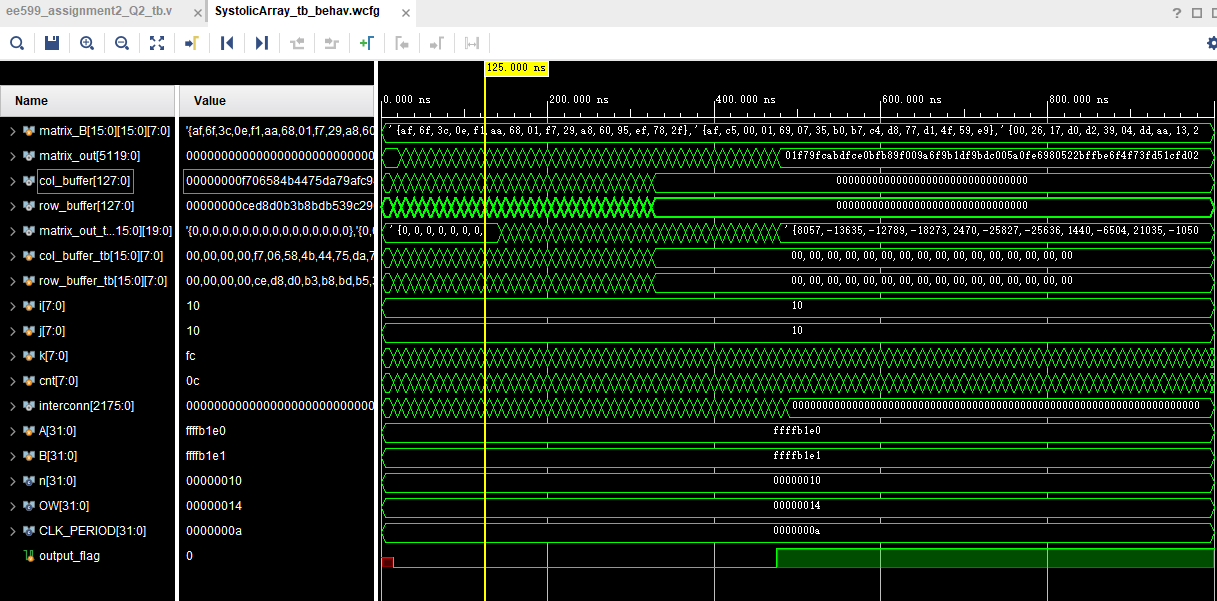
 

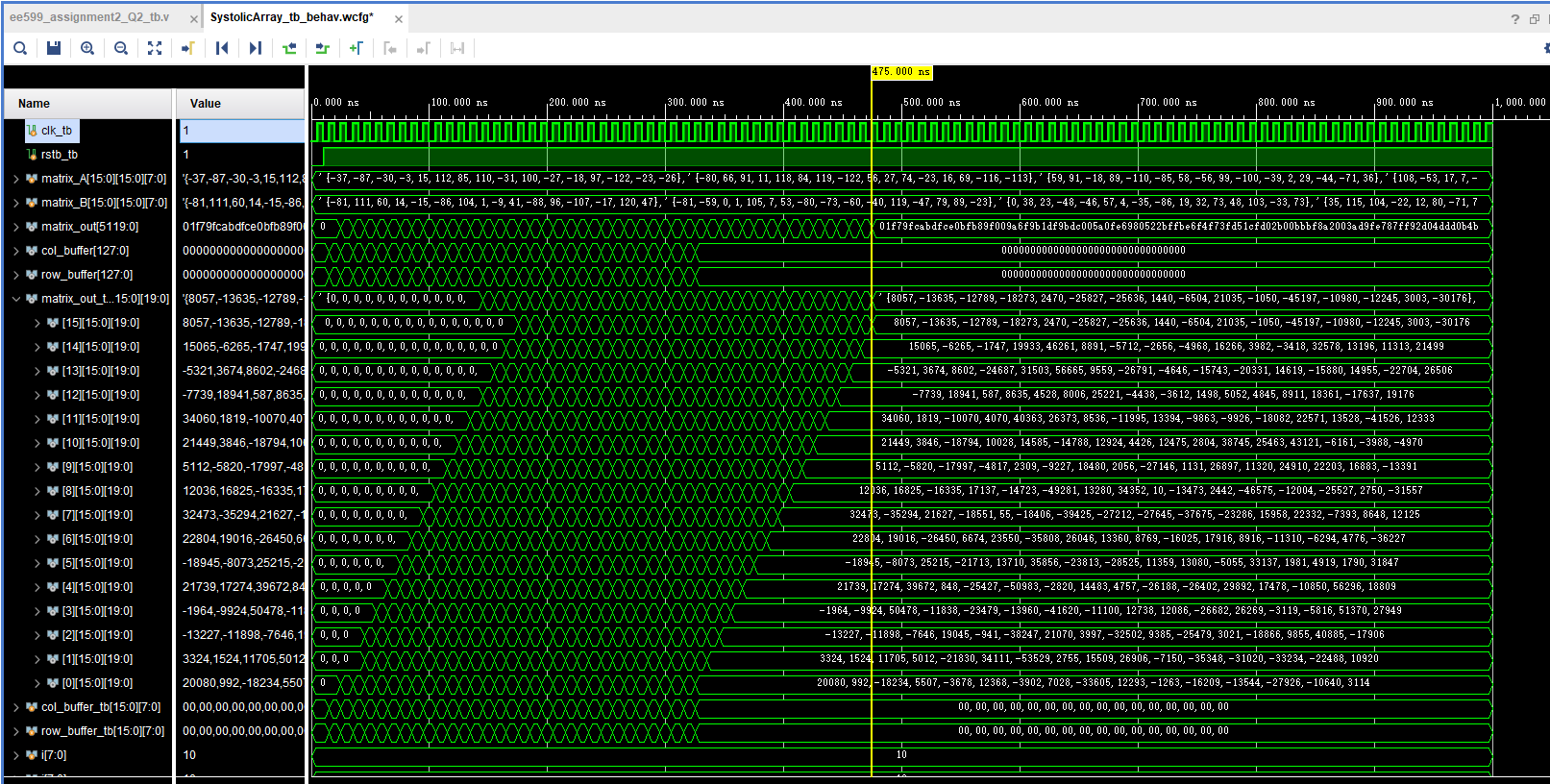
For a size systolic array test bench, we give two size matrices as input, and after = 47 clocks, the module will output results. The waveform is shown as followed:



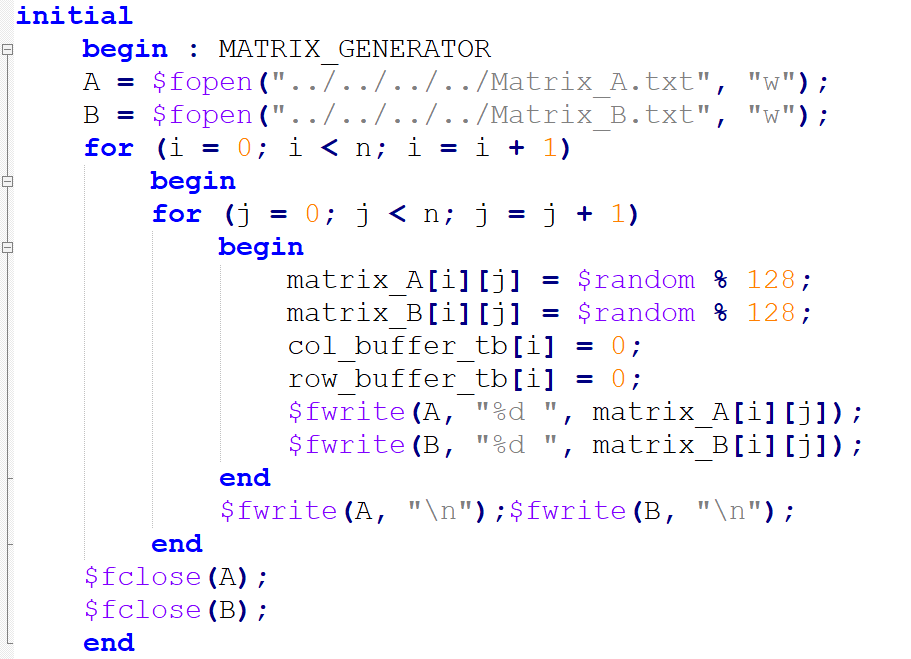


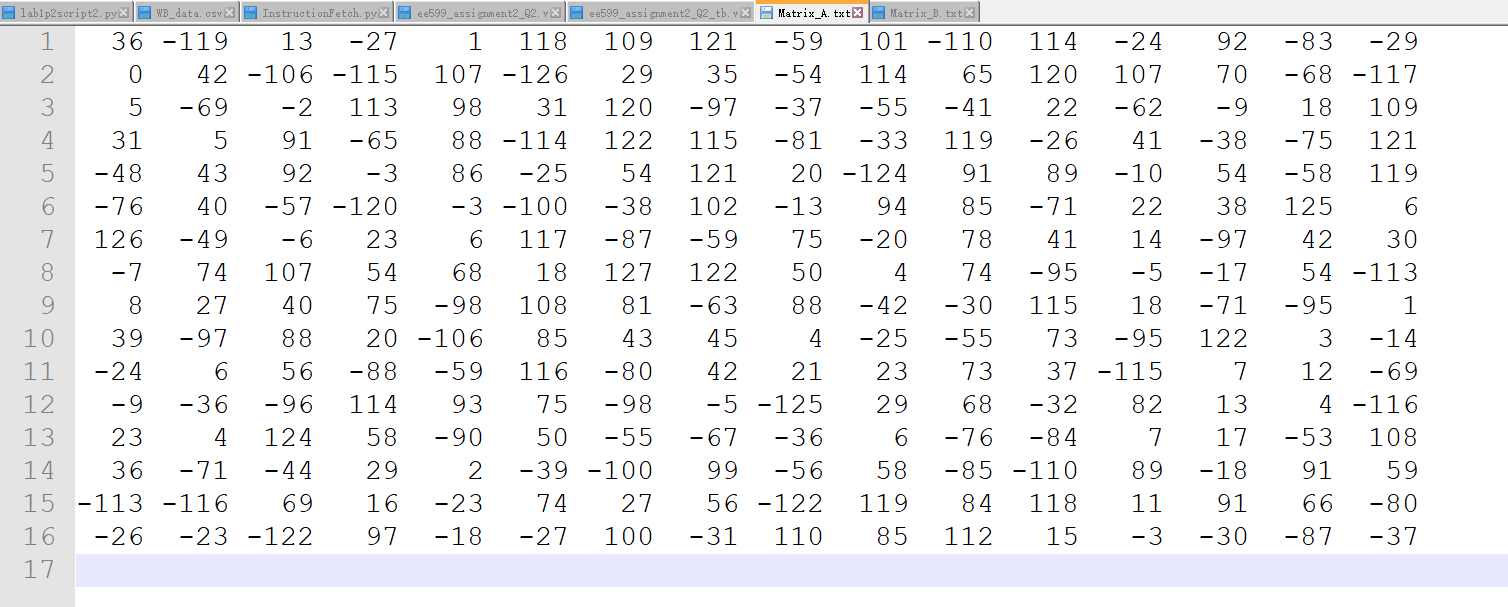




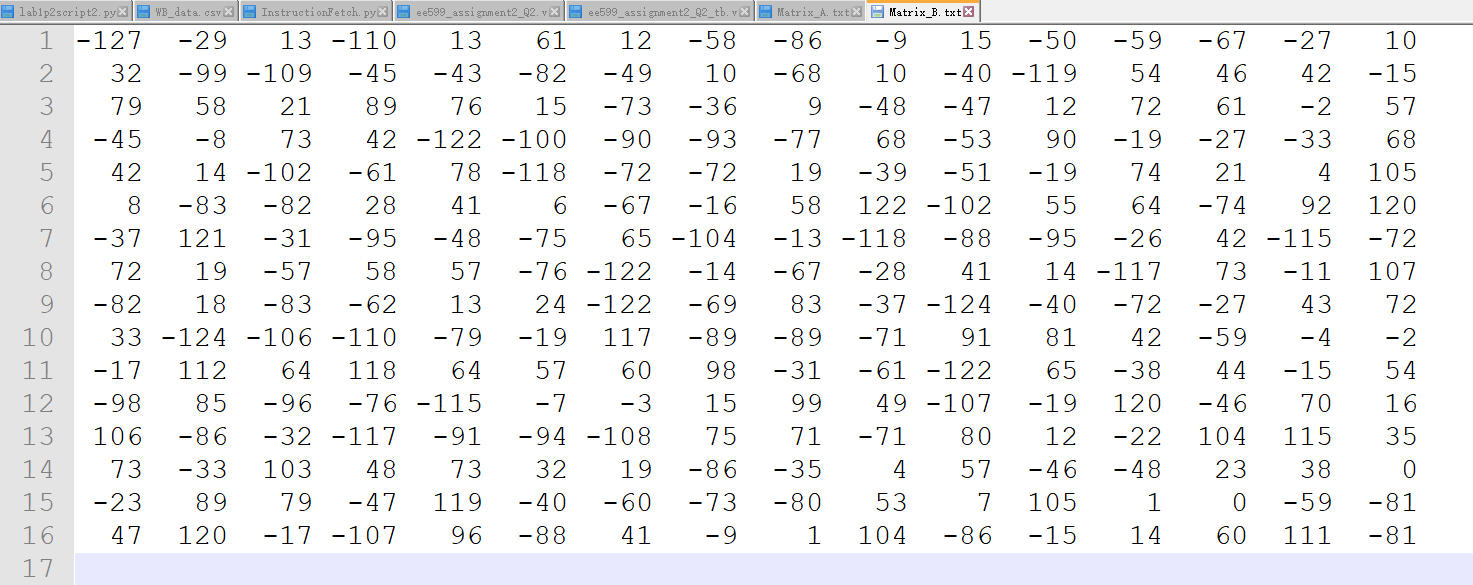


Here, we generate the two matrices by random. Using Verilog system function, we can easily generate the data and store them into two files Matrix\_A.txt and Matrix\_B.txt.





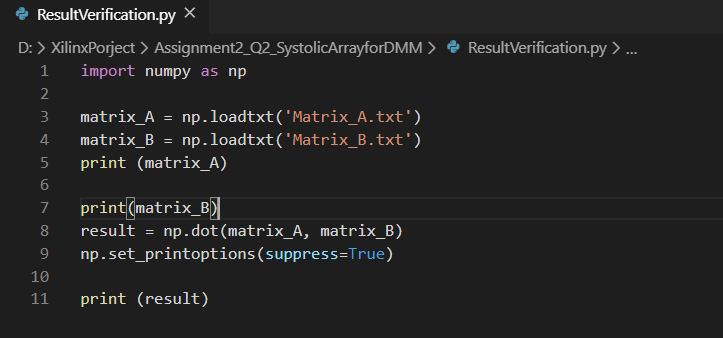
Matrix\_A.txt



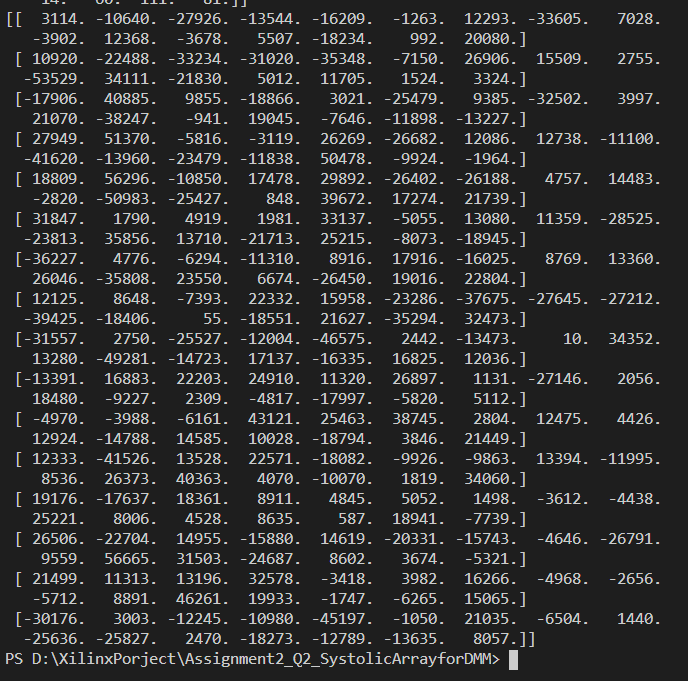
Matrix\_B.txt

We can get the result from waveform, and we can also read from the output file Matrix\_Result.txt.

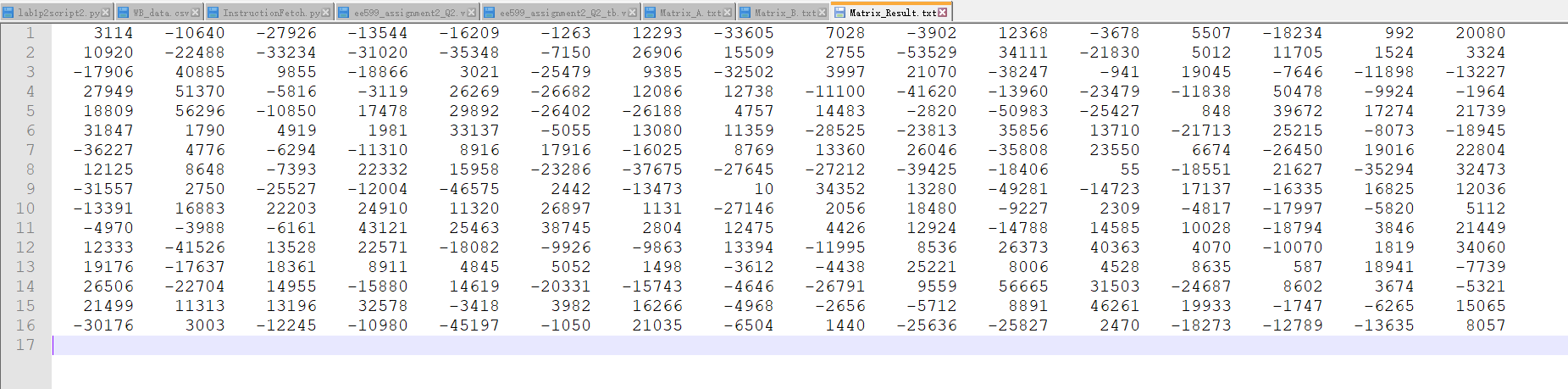
Then we write a Python script to verify the result.



And use python we can get the result as below.



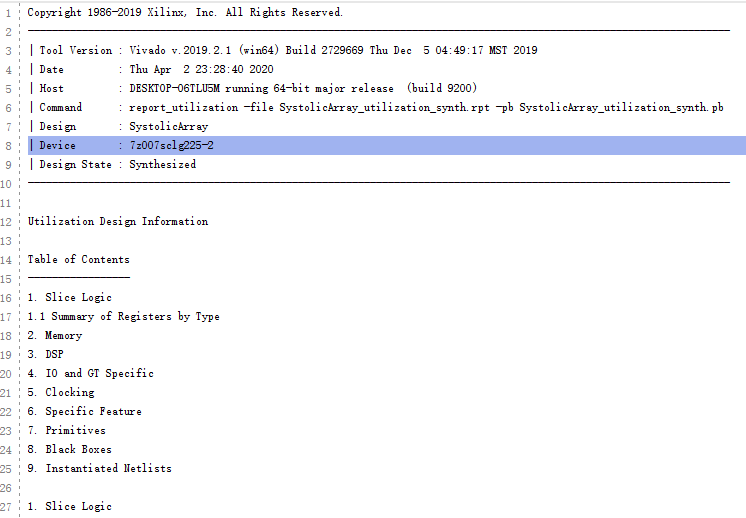
And compare the output file or waveform from the simulation, we can verify the correctness of systolic array.

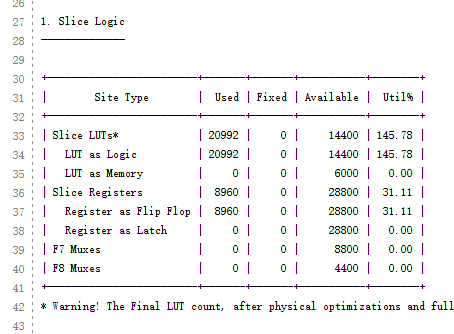
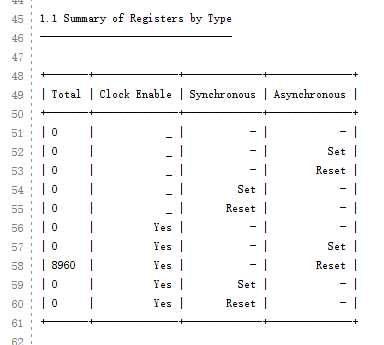
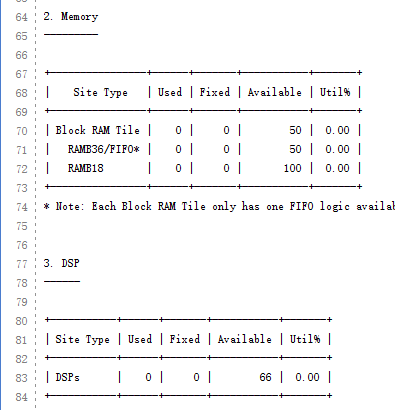
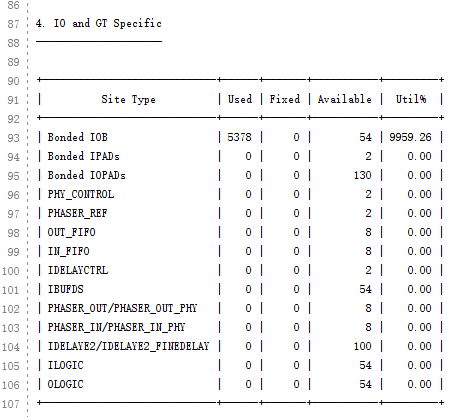
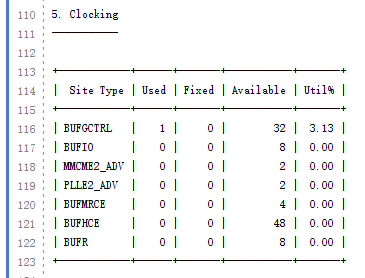
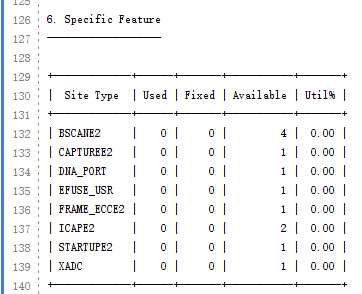
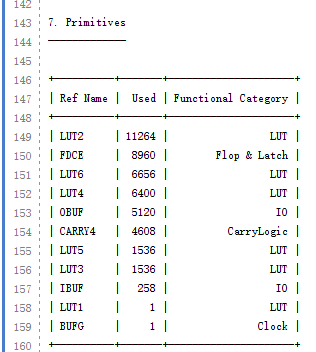


Matrix\_Result.txt

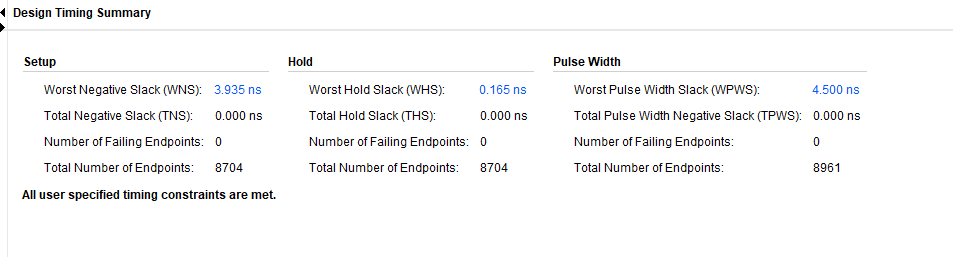
**The synthesis schematic please see in the Appendix.**

**Resource Estimation for 16x16 size:**

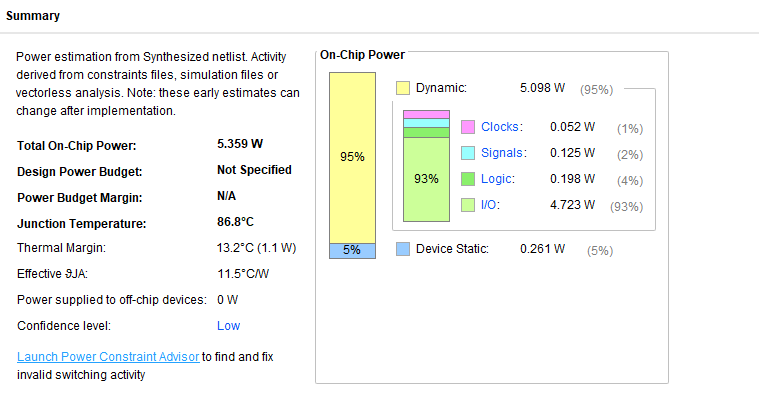


**Timing Estimation:**

****

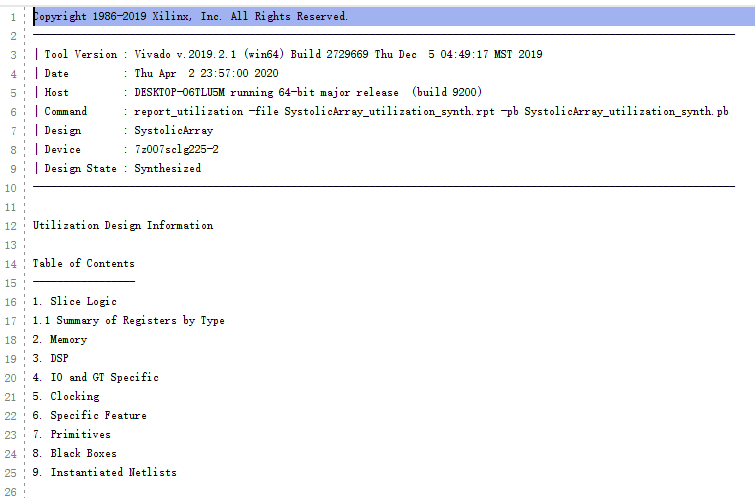
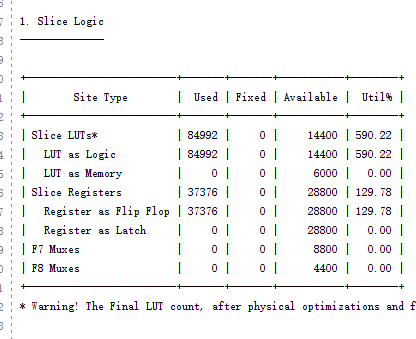
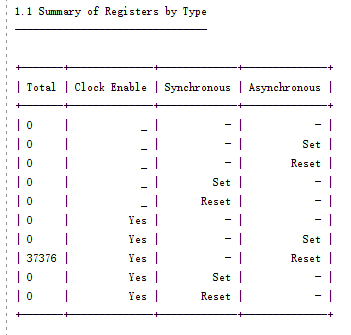
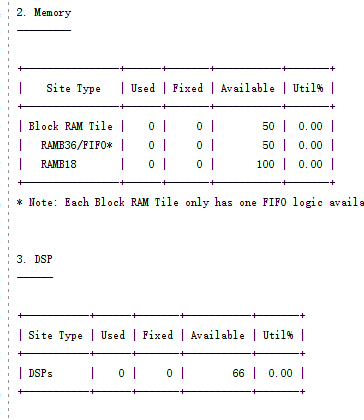
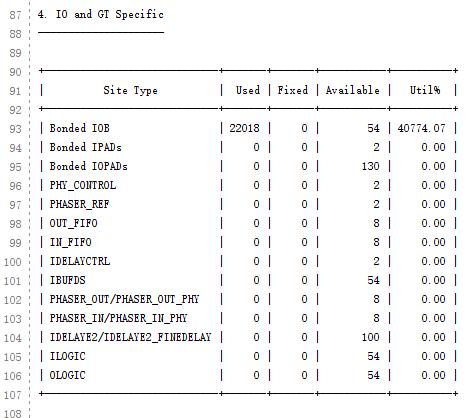
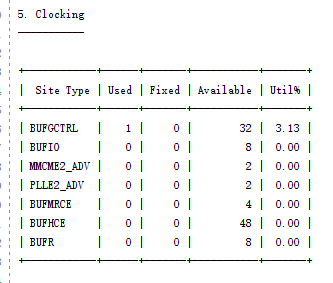
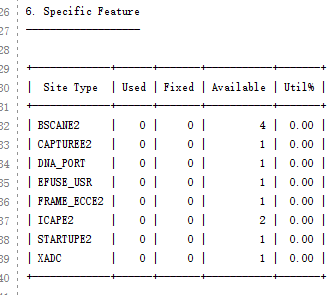
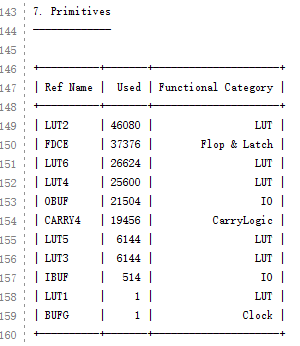
**Power Estimation:**



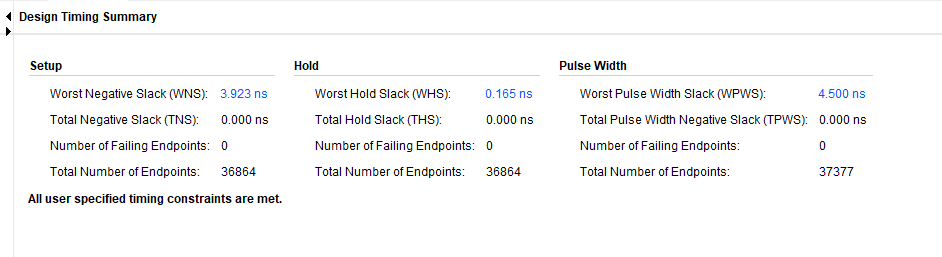
**Now redo for 32x32 size.**

**The full synthesis schematic please see in the Appendix.**

**Resource Estimation:**

**** ****      

**Timing Estimation:**

****

**Power Estimation:**

