ELEC522: ADV VLSI DESIGN (Project 2)

Students: Ran.lu (rl59)/Weixu.pan (wp7)

Instructor: Prof Joseph Cavallaro

Date: 10/5/2017

Content

Introduction

Matrix multiplication of matrices is common in many algorithm areas and the method of it is very simple. In recent years, Field Programmable Gate Arrays (FPGAs) have become a platform of choice for hardware realisation of computation intensive applications [2] and have been improved considerably in speed, density, and functionality, which make them ideal for system-on-a-programmable-chip designs for a wide range of applications [3]. Traditionally, matrix multiplication operation is often performed by parallel processing systems, which distribute computations over several processors to achieve significant speedup gains [4].

Systolic arrays have been widely used in VLSI implementation when inherent parallelism exists in the algorithm. In this paper, we will introduce the design of the four-dimensional array multiplication, and how FPGAs can be used to implement systolic arrays.

Top-level design

Hardware design should be based on top-down design method. Here it should has a top-level design which clearly states the function of each module. Then enter each module to design the lower module inside with Xilinx block sets in system generator.

From the top-level design diagram, it is easy to find that the whole project contains 4 top-level modules.

1. **Module for passing Matrix A and Matrix B in from MATLAB workspace**

This module will be designed to pass the Matrix A and Matrix B to the multiplication in rows and columns. And the elements in each row and column will be passed in systolic.

2. **Module for matrix Multiplication**

This module will be designed for realizing multiplication of A and B. It receives the systolic arrays from input module. Inside it are many MAC modules to compute multiplication and accumulation.

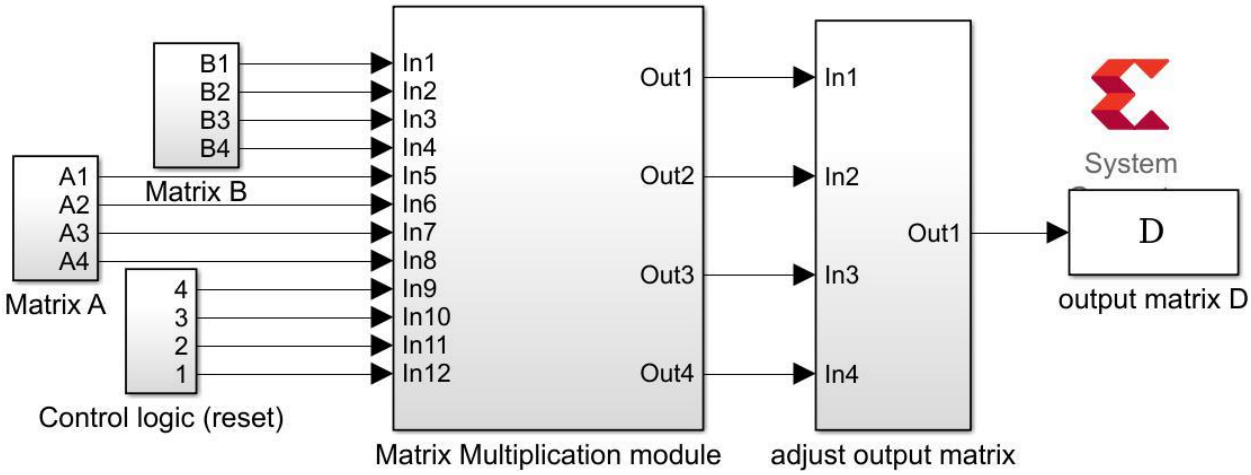
It also has shift function to shift the 16 output elements to 4 PE modules to let it output in four 1x4 vectors and this method will decrease the bits-length and the whole numbers of I/O.

3. **Module for passing Matrix C out to MATLAB workspace**

This module could combine the 4 vectors to a 4x4 matrix and output to the work space. The module inside will be designed by user in M code.

4. **Control logic**

This module is designed as a clock based module to control the whole system. There are many methods to realize it. It could be designed in algorithmic state machine (ASM) or in repetitive counter. Both of these are useful in this system design. This module will output the signal to the reset input of accumulation. It could be used to end a computation process and start a new round computation.



(Figure 1: top-level design)

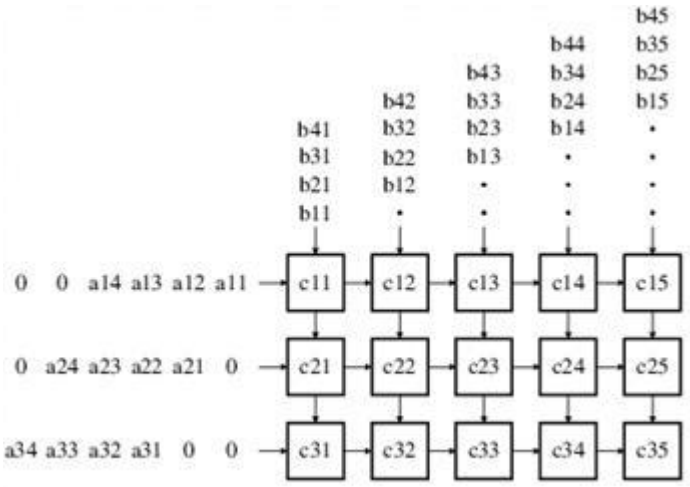
Module design

Algorithm and modules

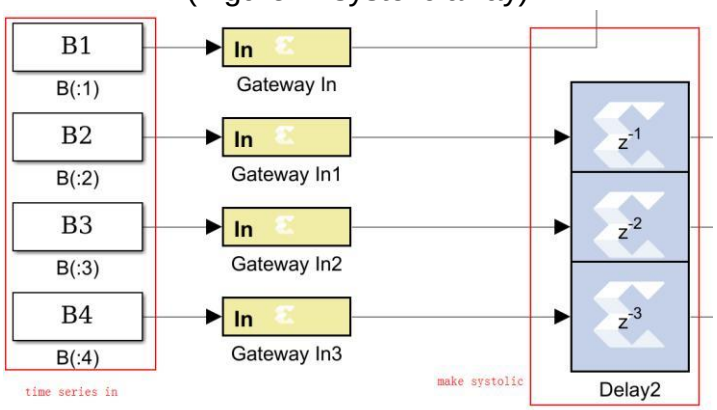
To get the product of two matrices, multiplying and accumulating the row elements of the first matrix and the column elements of the other is the simplest way. Generally speaking, the computation of a matrix multiplication, C = A×B, is represented as Eq. 1.

(1) $= \sum_{=1}$

A number base is obtained on one row in matrix A and one column in matrix B. So the traditional method is be simply designed as Figure 1[4] and the model on Simulink is shown in Figure 2.



(Figure 2: systolic array)



(Figure 3: systolic array enters in (Matrix B))

From the figure 3, B1, B2, B3 and B4 represent the 1st, 2nd, 3rd and 4th row of matrix B. from the systolic array method, there should be a one more extra delay on the next row to ensure the B1, B2, B3 and B4 to enter in systolic.

Processing element

Processing element (PE) is designed as a single computation unit for computing the numbers' multiplication and accumulation from the systolic array.

There are four inputs and three outputs in this module.

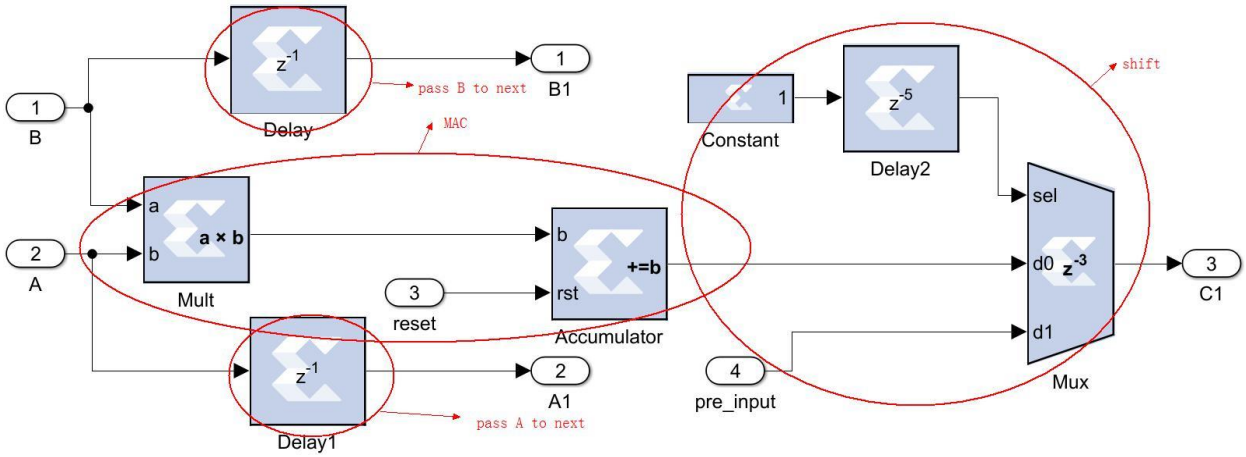
Two of the inputs (A, B) are from Matrix A and Matrix B in sequence.

The input (pre_input) will received the output from the next PE unit. This input will connect a two selective multiplexer to realise the shift function. The shift function will be introduced in the multiplication module (4x4 PE).

The input (reset) is used to control the working status of this accumulator. It will be controlled by the specified clock module to control the accumulator in period.

The output A1 and B1 are about to be passed to the next PE to compute. For example, for computing the A11*B21. A11 should be delayed 1 to make sure it enter the PE with B21 synchronously.

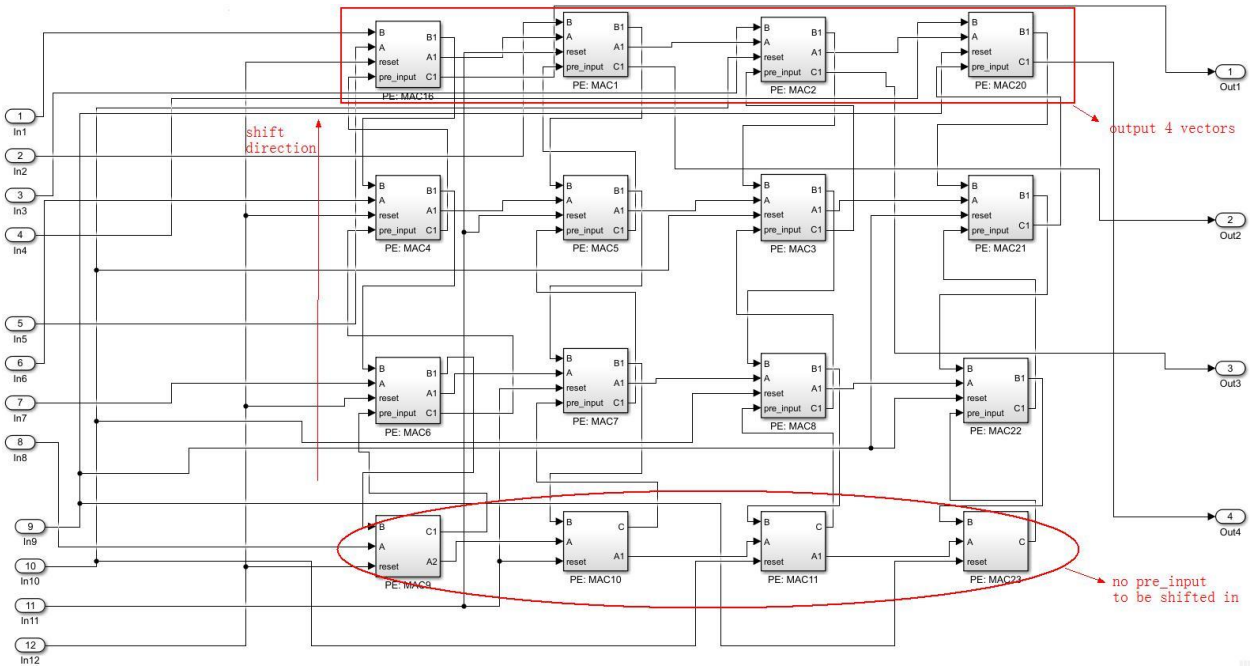
The output C1 is the result of calculation in one clock sample. C1 comes out after the accumulation.



(Figure 3: Module inside PE)

Multiplier

This module has 16 PE modules connected and let the systolic array pass in to get the output matrix. This module will spend 7 clock cycles to get the result. The module contains an input reset. Two input matrices and one output matrix. Inside module, there will be some series of shift blocks to shift the output arrays in the single PE. This step will be used to compress 16 output elements and reduce the bits-length. The number of I/O decreases. When the PE got the result, it will shift it to the one next until reach the first one. Then it will output 4 numbers in series.



(Figure 4: diagram of the module multiplier)

The shift comes from the 4th row and reach 1st row to output the number in series.

From the following chart, the number in the square shows the delay when computation finished.

4	5	6	7
5	6	7	8
6	7	8	9
7	8	9	10

Therefore, C (1,1) will obtain the final result of multiplication and accumulation of the whole line A (1,:) and B (:,1) at 4^{th} clock. So the C (1,2) and C (2,1) will get the result at 5^{th} clock.

However, the shift will take one more delay on each square. For instance, when time is at clock 4 and C (1,1) got the final result. So it will output at clock 4 but the next value will be obtained at 5 clock. So at clock 5, C (1,1) will accept and output the result shifted from C (2,1) and so on.

5	5	7	8
6	6	8	9
7	8	9	10
8	9	10	11

From clock4 to clock 7, column 1 finishes computation. Output vector at clock 7

From clock5 to clock 8, column 2 finishes computation. Output vector at clock 8

From clock6 to clock 9, column 3 finishes computation. Output vector at clock 9

From clock7 to clock 10, column 4 finishes computation. Output vector at clock 10

So the output module needs to output 4 vector synchronously. So the column 1 should have a delay 3 block, the column 1 should have a delay 2 block and the column 1 should have a delay 1 block to make sure the 4 vector are outputted at clock 10 simultaneously.

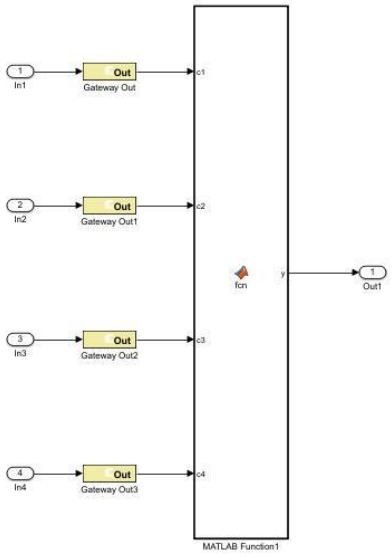
These delay is set in the mux of the MAC PE module (Figure 3).

Output result matrix to workspace

This module will accept the 4 output vector from the multiplier and combine it into a 4x4 matrix and output to the MATLAB workspace.

```
function y = fcn(c1,c2,c3,c4)
u=[c1';c2';c3';c4'];
y = u;
```

C1, C2, C3 and C4 are 4 input vectors which will be connected to the output of multiplexer. Because the shift is designed on the column of output signal. So the transfer of each C1, C2, C3 and C4 is needed then obtain and output the correct result matrix.



(Figure 5: output module)

Control logic

Here a control logic has been designed for control the system working in cycle and it could reset the accumulator in the PE module. There are many ways to realize it such as state machine.

Here the control logic module was designed by four counters and multiplexers. Each mux has one output signal. This signal is a trigger for resetting. For example, for first part (count 0-10), the counter will count the 11 numbers in total from 0 to 10. When counter is 0 to 9, the mux will output zero because input 0 is connected from 0-9 input of the mux. However, the counter will count 10 in next clock cycle but here the output from mux becomes 1.

At next clock cycle, the counter will restart at 0 and the output of mux return to 0.

So the 1 will only appear for one clock time very 11 clock cycles.

The output of the first control logic (count 0-10) will connect to the 'rst' of the counter in 2^{nd}, 3^{rd} and 4^{th} control logic. When the output signal is 1, the counter in next 3 controls will reset.

The 2^{nd} control will counter from 0 to 9;

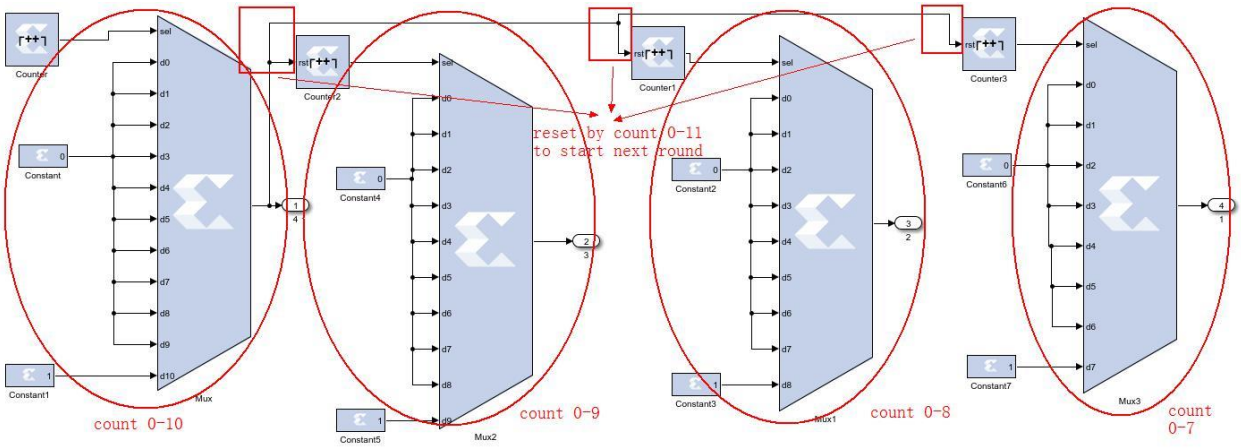
The 3^{rd} control will counter from 0 to 8;

The 4^{th} control will counter from 0 to 7;

So the output of 1^{st} control needs to be used as reset signal to 2^{nd}, 3^{rd} and 4^{th}. Without these signal, the four counter could not start from 0 simultaneously to make sure the whole system restarts normally in period for new computation.

Different count range (0 to 10 or 0 to 8) is set because there are delays at output of the computation and results are shifted. The 4 output vectors not only should be output at 4 clock but also the whole system should take fewer clock cycles. So the strategy is to reset the column once the computation of this column is finished.

Therefore, the column 1 will be reset at clock 7, the column 1 will be reset at clock 8, the column 1 will be reset at clock 9 and the column 1 will be reset at clock 10.



(Figure 6: control logic)

Test result

Set up the input matrix A and matrix B. And pass A and B into system in time series.

Editor - C:\Users\Kenny_LR\Desktop\INPUT_MATRICES.m

INPUT_MATRICES.m

```
A=[7,3,5,2;3,6,7,9;8,4,2,10;5,5,6,2];
B=[5,3,2,2;8,6,1,5;6,3,5,2;7,9,4,2];

A1=timeseries(A(1,:));
A2=timeseries(A(2,:));
A3=timeseries(A(3,:));
A4=timeseries(A(4,:));


B1=timeseries(B(:,1));
B2=timeseries(B(:,2));
B3=timeseries(B(:,3));
B4=timeseries(B(:,4));
```

(Figure 7: set up matrix A and matrix B)

Compute A*B in MATLAB workspace.

Command Window

```
>> INPUT_MATRICES
>> A*B

ans =

   103    72    50    43
   168   147    83    68
   154   144    70    60
   115    81    53    51
```

(Figure 8: MATLAB computation result)

Run the system from clock 7-10 (computation)

D — 4x4 double

	1	2	3	4
1	35	21	14	14
2	59	39	17	29
3	89	54	42	39
4	103	72	50	43

D — 4x4 double

	1	2	3	4
1	59	39	17	29
2	89	54	42	39
3	103	72	50	43
4	168	147	83	68

D — 4x4 double

	1	2	3	4
1	89	54	42	39
2	103	72	50	43
3	168	147	83	68
4	154	144	70	60

D — 4x4 double

	1	2	3	4
1	103	72	50	43
2	168	147	83	68
3	154	144	70	60
4	115	81	53	51

At clock 7: the first line of results have been computed then shift.

At clock 8: the second line of results have been computed then shift.

At clock 9: the third line of results have been computed then shift.

At clock 10: the fourth line of results have been computed then shift. (correct result)

Run the system from clock 11-14 (test reset)

D — 4x4 double

	1	2	3	4
1	168	147	83	68
2	154	144	70	60
3	115	81	53	51
4	0	0	0	0

D — 4x4 double

	1	2	3	4
1	154	144	70	60
2	115	81	53	51
3	0	0	0	0
4	0	0	0	0

D — 4x4 double

	1	2	3	4
1	115	81	53	51
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0

D — 4x4 double

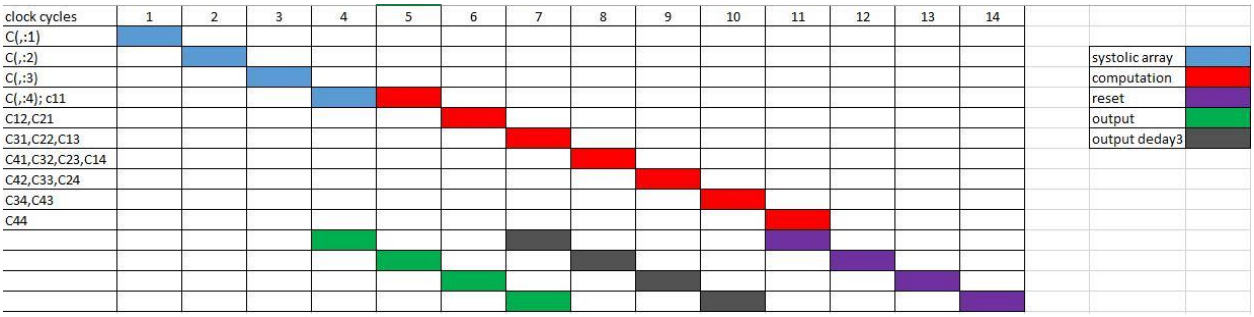
	1	2	3	4
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0

At clock 11: the first line of output was reset.

At clock 12: the second line of output was reset.

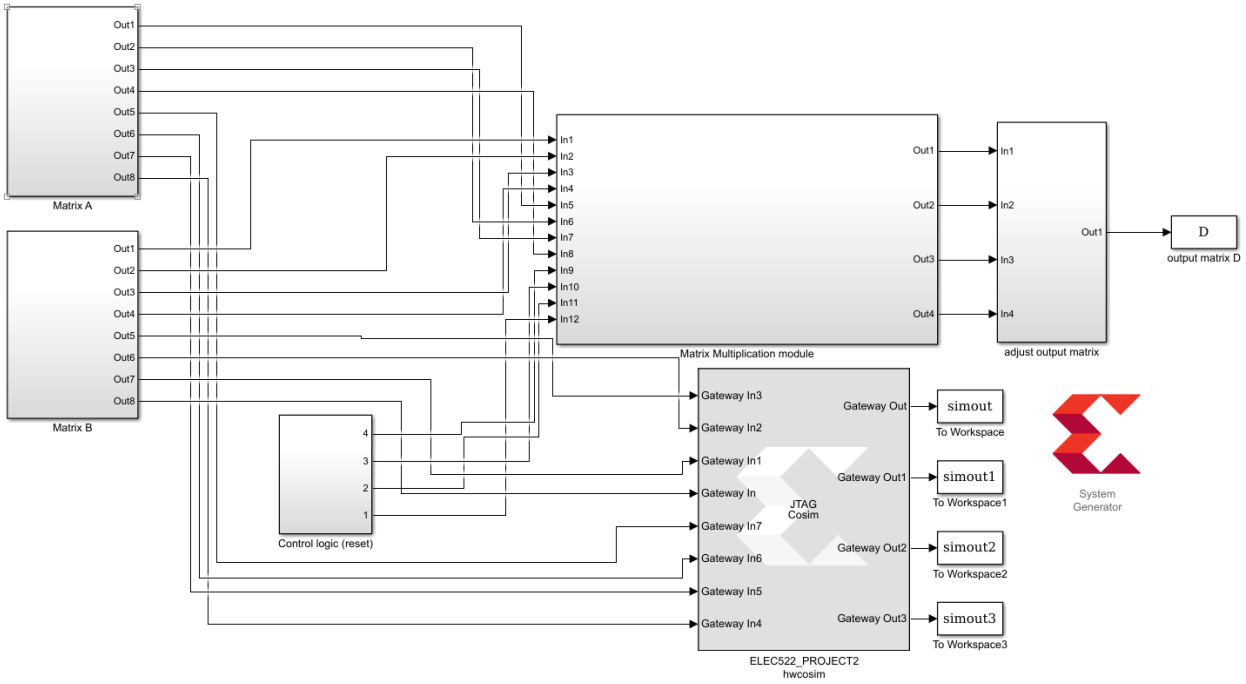
At clock 13: the third line of output was reset.

At clock 14: the fourth line of output was reset.

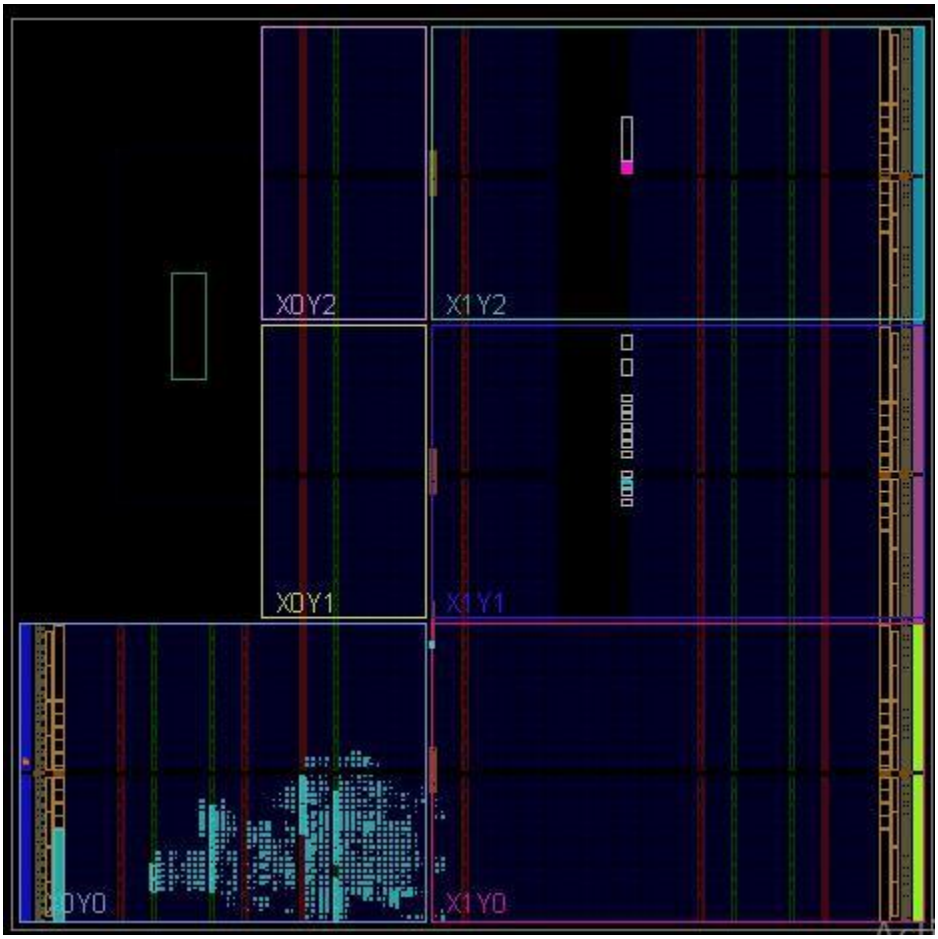


(Figure 9: system process time flow)

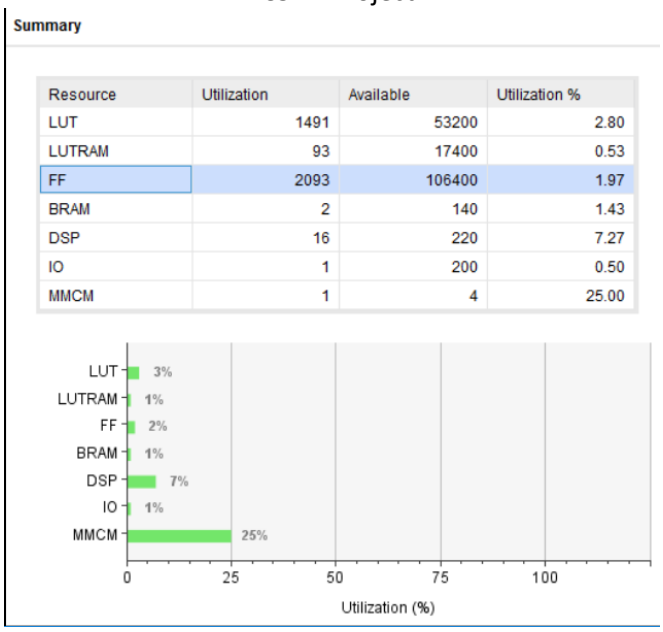
Vivado utilization result



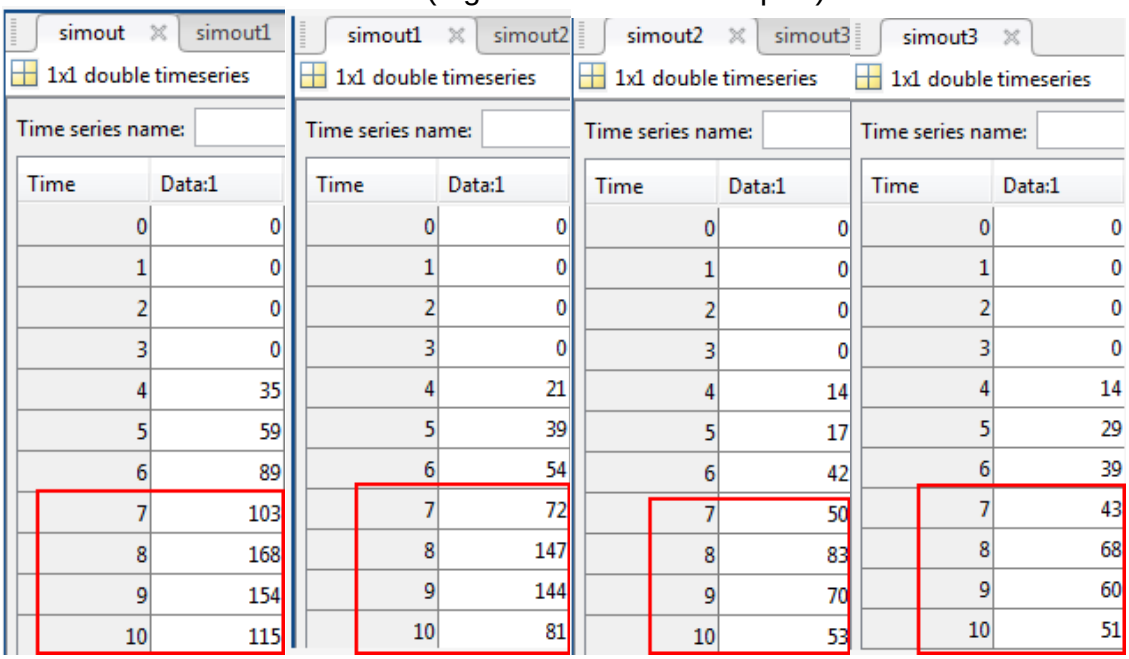
(Figure 10: JTAG)



(Figure 10: device)



(Figure 11: utilization report)



(Figure 12: Test JTAG)

Conclusion

In summary, this project aims to implement a matrix multiplication system to realize the computation of it. The design tool is Xilinx system generator and analyze the project with Xilinx Vivado.

The input matrices are in time series and in the form of systolic array.

The multiplication module is based on 16 MAC modules with shift functions. Shift function will reduce the number of I/O from 16 to 4.

The control logic has 4 outputs to reset the accumulator on each columns because every column output at different clock. The accumulators in every column will be reset once this column finished multiplication and accumulation.

The whole system will take 14 clock cycles.

Clock 1-4: input systolic arrays.

Clock 4-10: multiplication and accumulation also shift result

Clock 8-11: reset the system by each column, but delayed by 3 clock cycles. So actually reset occurs in Clock 11-14.

So the whole system will take 14 clock cycles to do one cycle computation. The next clock 15 will restart a new round to compute new input matrices.

References

. [1] Bravo, I.; Jimenez, P.; Mazo M.; Lazaro, J.L.; de las Heras, J.J.; Gardel, A. “Different proposals to matrix multiplication based on FPGAs”, In Proceedings of the IEEE International Symposium on Industrial Electronics, December 2007; pp. 1709-1714.

. [2] Syed M. Qasim, Ahmed A. Telba and Abdulhameed Y. AlMazroo, “FPGA Design and Implementation of Matrix Multiplier Architectures for Image and Signal Processing Applications”, in IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.2, February 2010.

. [3] M. Ceschia, M. Bellato, A. Paccagnella, and A. Kaminski, “Ionbeam testing of ALTERA APEX FPGAs”, in Proceedings of IEEE Radiation Effects Data Workshop, pp. 45–50, Phoenix, Ariz, USA, July 2002.

. [4] Syed M. Qasim, Shuja A. Abbasi, “Hardware Realization of Matrix Multiplication using Field Programmable Gate Array”, MASAUM Journal of Computing Vol.1 No.1 August 2009