ELEC 522, Advanced VLSI

Rice University, Fall 2022

Project 2 Report

Name: Yufei Gu

NetID: yg77

29 September 2022

1. **Description of your design approach for both matrix – matrix multiplication and matrix – vector multiplication modes**

The overview of my design is shown as Figure 1.

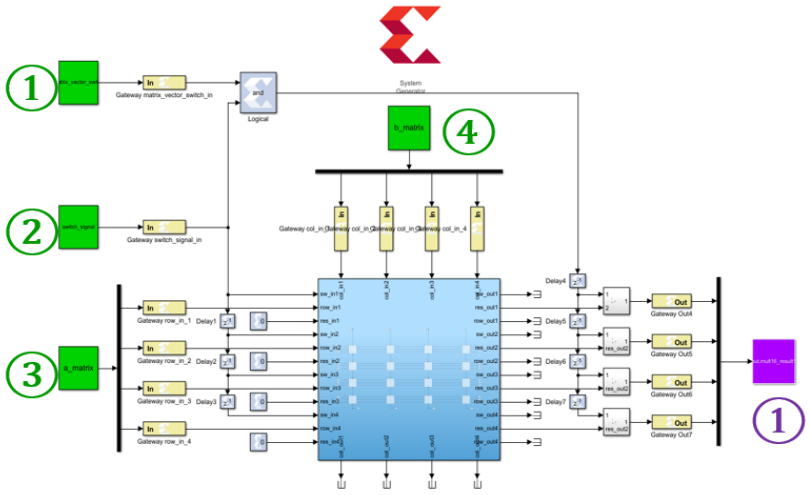


Figure 1

The green blockes are from workspace blocks.

1) The first one is to switch modes between matrix-matrix mode and matrix-vector mode, using one bit for control.
2) The second one is a signal to enable the result output of each computation unit.
3) The third one is matrix_A input, it can continuously input matrices.
4) The fourth one is matrix_B input, it can continuously input matrices as well.

The purple block is to worksapce block which is used to output result.

The blue block is a matrix calculation block, which consists of 16 identical computing units.

In one cycle, it can do two 4 by 4 matrix multiplication or one 4 by 4 matrix and one 4 by 1 vector multiplication, and all elements are 16 bits. The whole design uses 128 pins for data input, 64 for data output and 2 for control signals, 194 pins in Total.

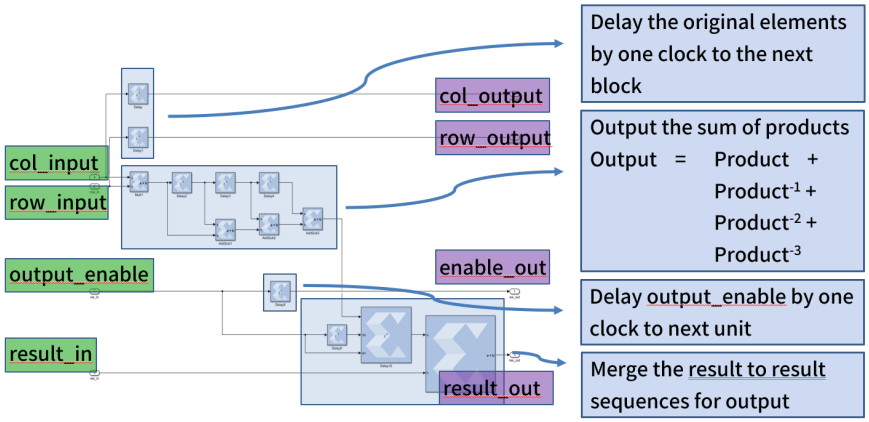


Figure 2

Each uint has 4 input ports and 4 outport ports, as shown in the Figure 2.

1) row_input will be delayed by one cycle and then output to the next right uint.
2) col_input will be delayed by one cycle and then output to the next unit below.
3) output_enable will be delayed by one cycle and then output to the next right uint because the next block on the right will get the result one clock later than this block.
4) When output_enable is valid, the unit will merge the result to results sequences for output.

At the top left of the overall block diagram, there is a matrix_vector_switch control signal to switch between matrix-matrix mode and matrix-vector mode. Set its value to 0, it is matrix-matrix mode; set its value to 1, it is matrix-vector mode. As shown in the Figure 3.

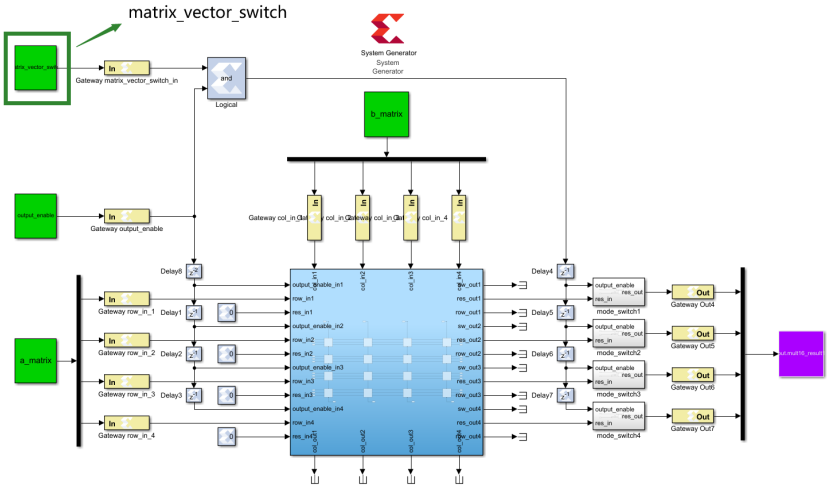


Figure 3

❖ **matrix – matrix multiplication**

At In this mode, since the matrix_vector_switch is 0, the AND gate output is always 0. Therefore, the output_enables of the four mode_switches are also always 0. As shown in the Figure 4.

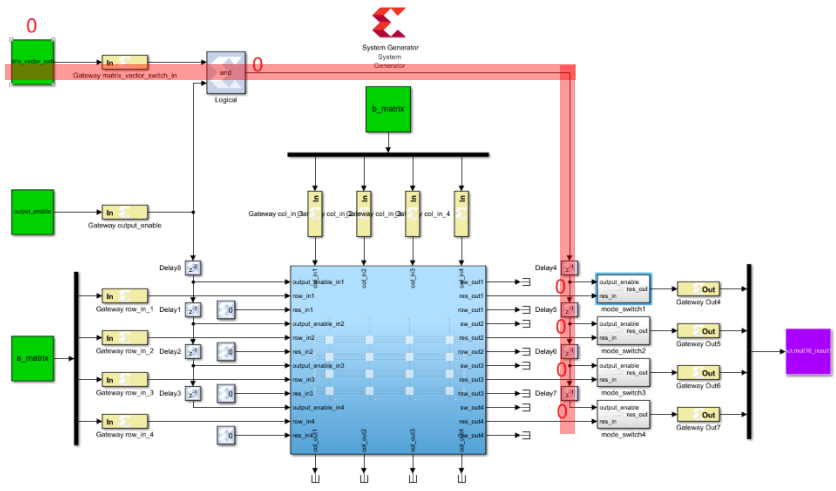


Figure 4

Inside each mode_switch, since the en port of the delay block is always 0, the output res_out is equal to result_in. As shown in the Figure 5.

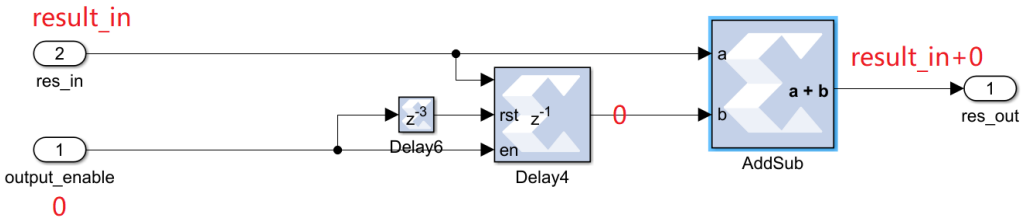


Figure 5

Therefore, in matrix-matrix mode, the mode_switch module does nothing and directly connects res_in to res_out.

❖ **matrix – vector multiplication**

At In this mode, since the matrix_vector_switch is 1, the AND gate output is the same as the output_enable. Its waveform is shown in the Figure 6.

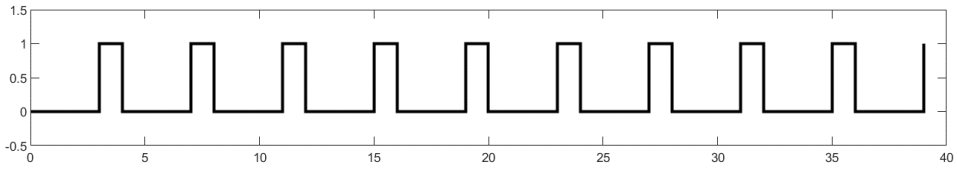


Figure 6

The waveform of each endpoint in mode_switch is shown in the Figure 7, all data entering from res_in will be kept on the bus for 3 clock cycle. Thus, the output clock frequency of matrix-vector mode is one-fourth of that of matrix-matrix mode.

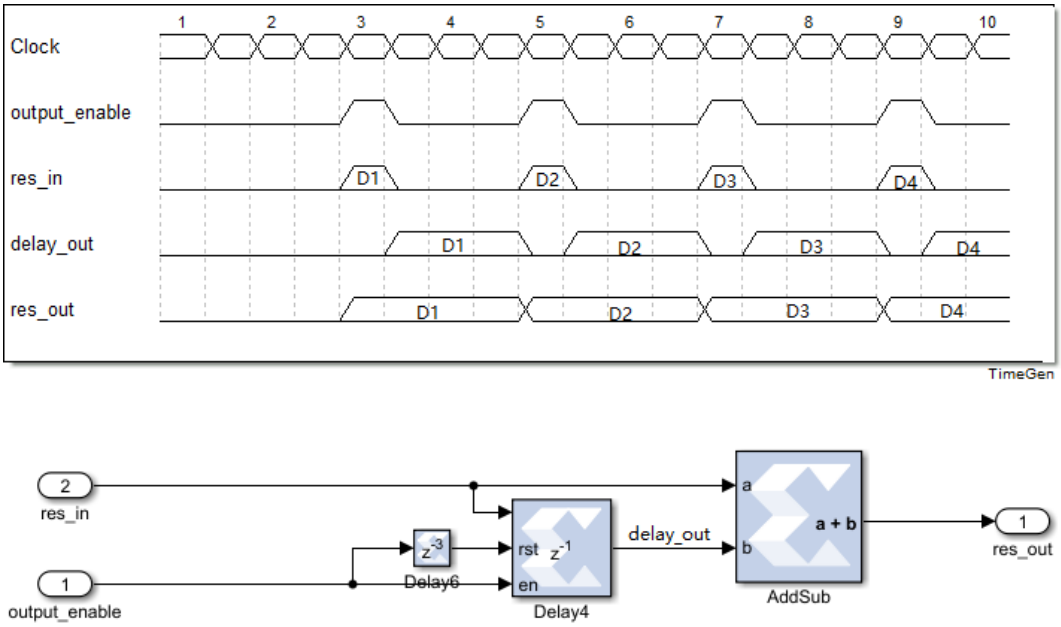


Figure 7

2. Tradeoff on input and output architecture

The data flow diagram of my design is as shown in Figure 8, it uses 8 16-bit Gateway In, 4 16-bit Gateway Out for data flow. Plus two Boolean control signals, it occupies a total of 194 pins.

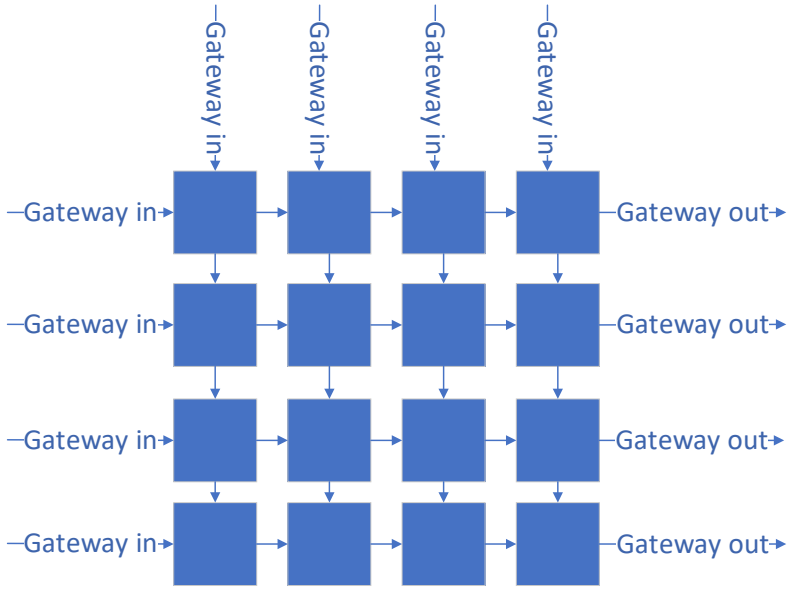


Figure 8

In my previous design as shown in Figure 9, I added a Gateway Out to each multiplication unit, so I needed 8 16-bit Gateway In, 16 16-bit Gateway Out. With two control signals, a total of 386 pins are occupied. It exceeds the number of pins available.

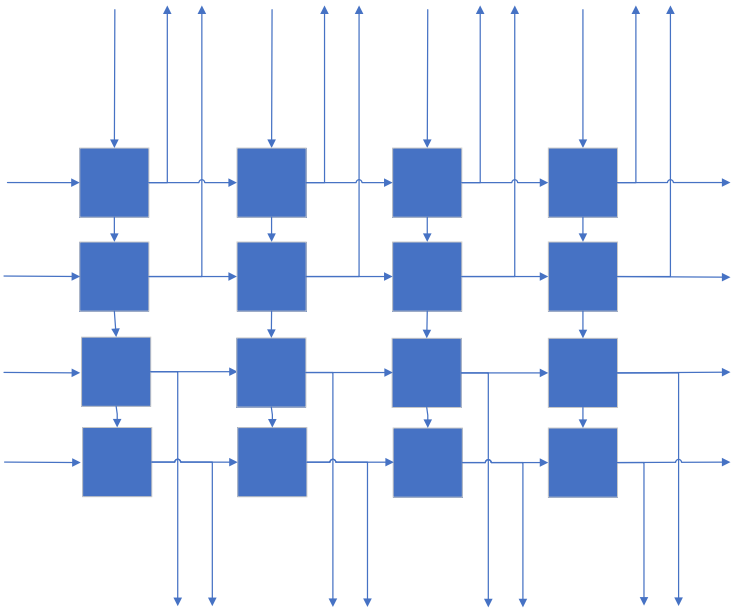


Figure 9

These two different designs are only different in pin occupancy, and are exactly the same in timing and data throughput. Therefore, in order to save the number of pins, I use a solution that occupies a small number of pins.

3. Screen shots of model composer files

Overview of Project2.slx

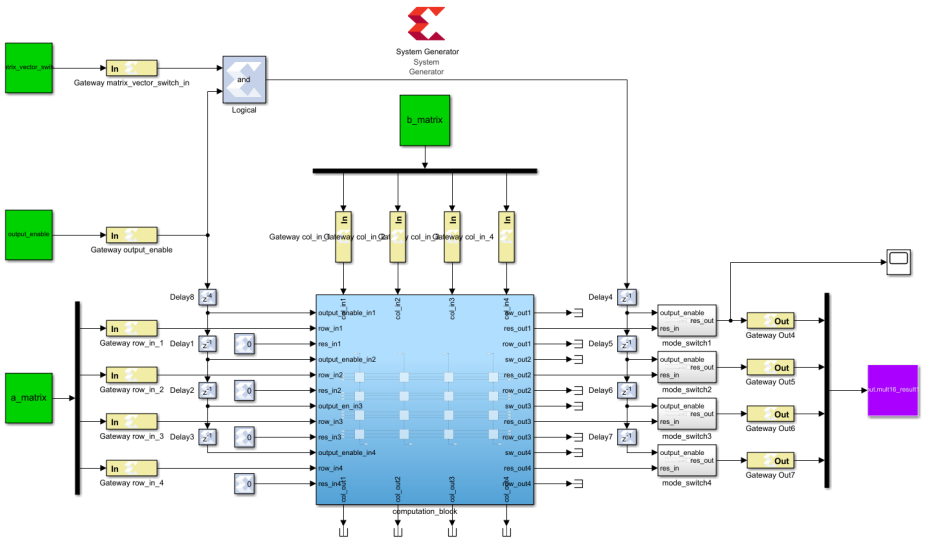


Figure 10

View of computation_block

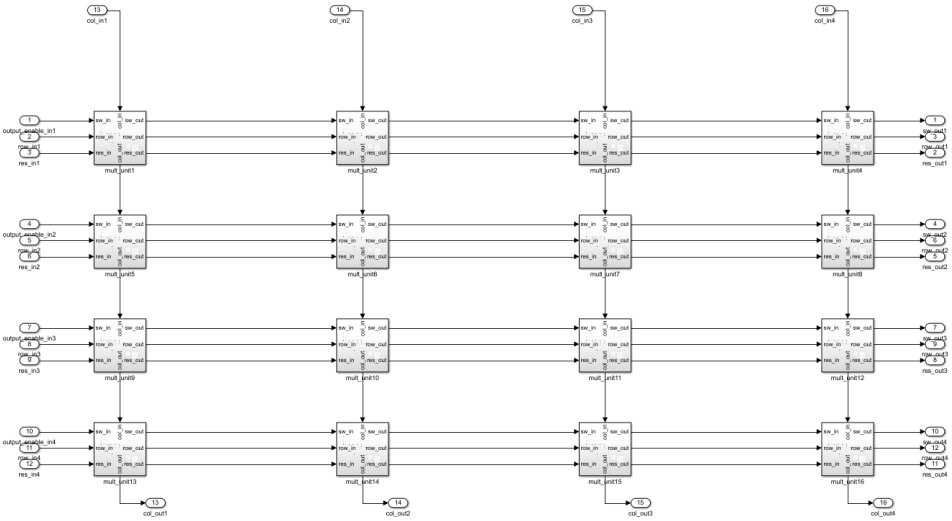


Figure 11

View of mode_switch block

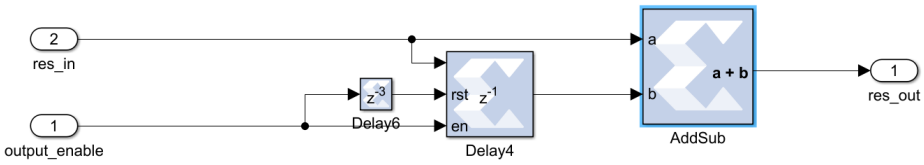


Figure 12

View of mult_unit block

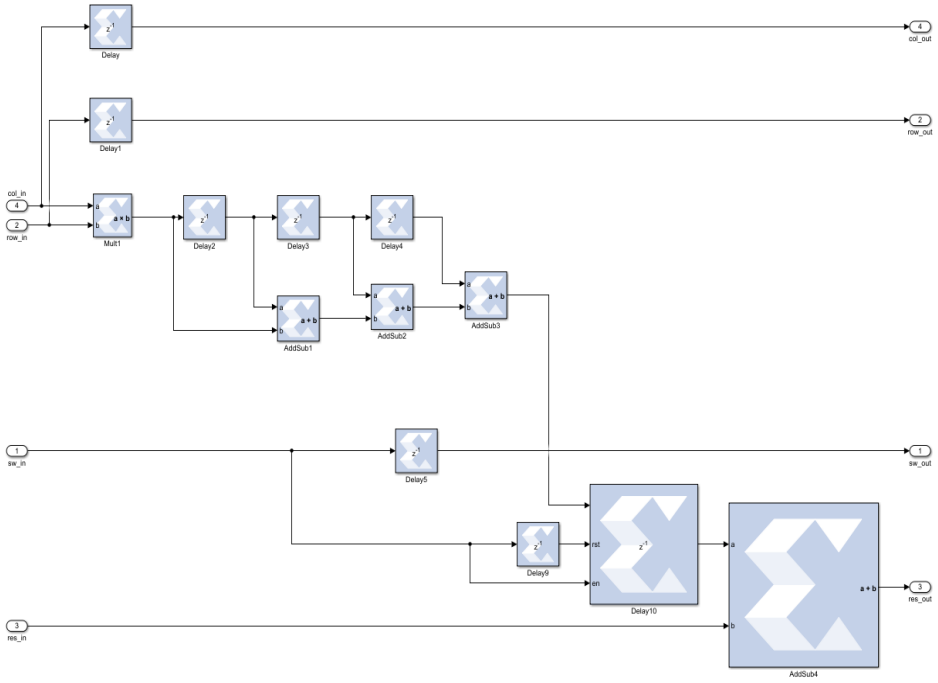


Figure 13

4. Screen shots of simulation results

❖ matrix – matrix mode test results

Command Window

Matrix A1:

1	2	3	4
2	3	4	5
3	4	5	6
4	5	6	7

Matrix B1:

1	2	2	0
1	0	0	0
1	0	0	0
1	0	0	0

Software Simulation result of A1*B1:

10	2	2	0
14	4	4	0
18	6	6	0
22	8	8	0

Correct result of A1*B1:

10	2	2	0
14	4	4	0
18	6	6	0
22	8	8	0

Software Simulation result equals Correct result ,PASSED!

Matrix A2:

1	2	3	4
2	3	4	5
3	4	5	6
4	5	6	7

Matrix B2:

2	3	4	5
3	4	5	6
4	5	6	7
5	6	7	8

Software Simulation result of A2*B2:

40	50	60	70
54	68	82	96
68	86	104	122
82	104	126	148

Correct result of A2*B2:

40	50	60	70
54	68	82	96
68	86	104	122
82	104	126	148

Software Simulation result equals Correct result ,PASSED!

Figure 14

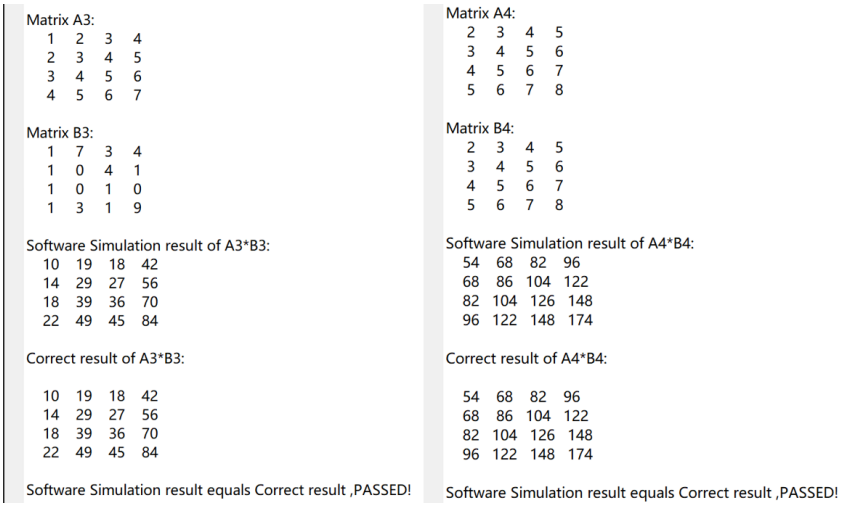


Figure 15

❖ matrix – vector mode test results

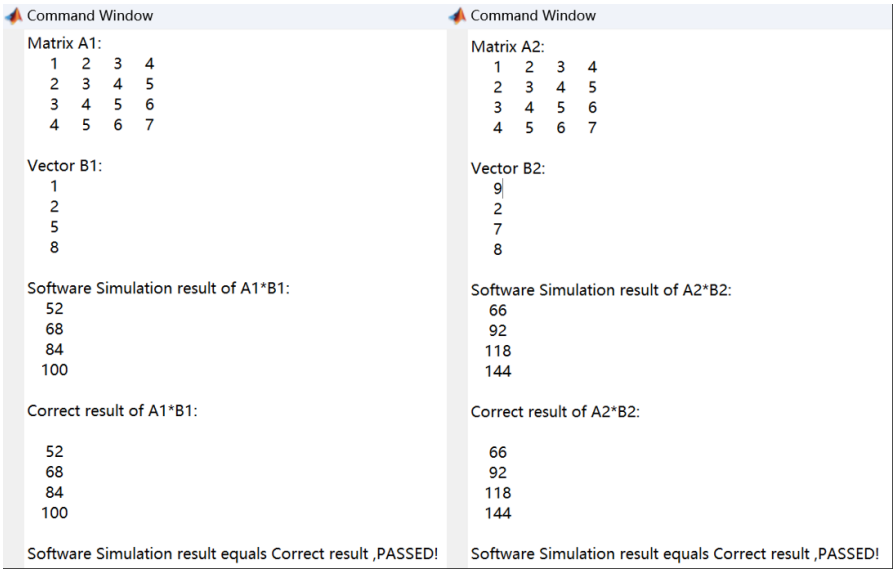


Figure 16

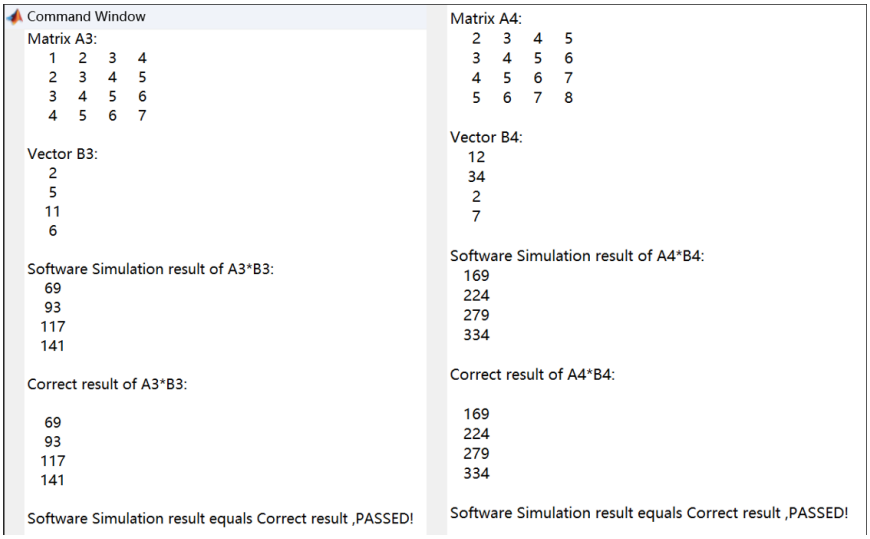


Figure 17

5. Screen shots of hardware co-simulation results

The overall block diagram is shown in the Figure 18.

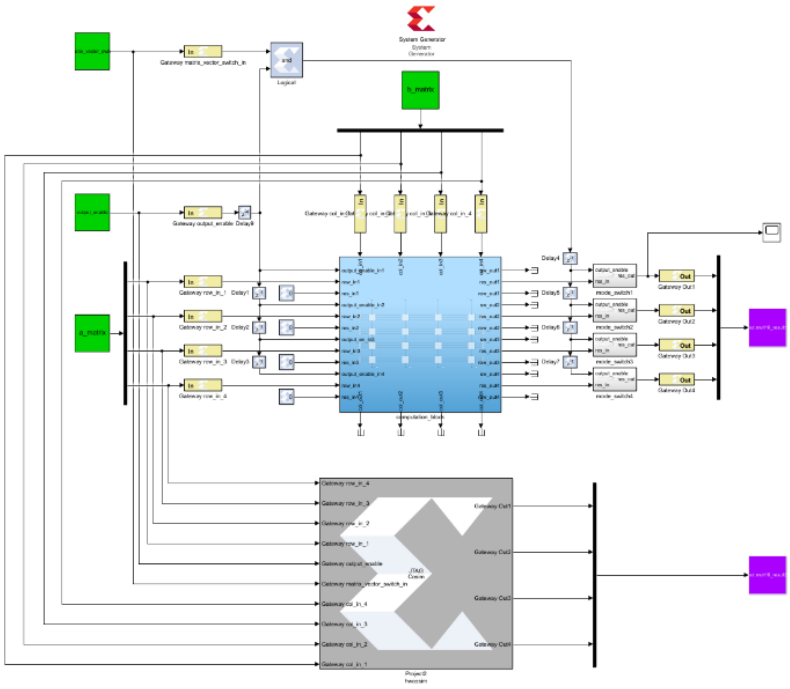
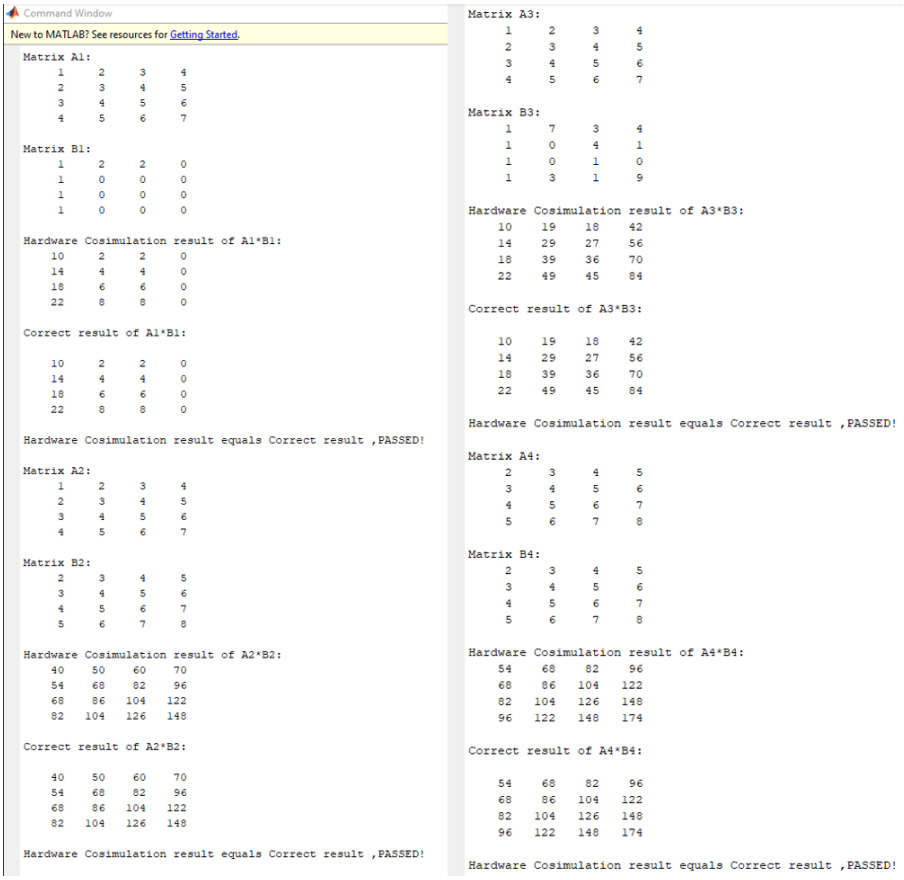


Figure 18

❖ matrix – matrix mode test results



```
Command Window
New to MATLAB? See resources for Getting Started.

Matrix A1:
     1     2     3     4
     2     3     4     5
     3     4     5     6
     4     5     6     7

Matrix B1:
     1     2     2     0
     1     0     0     0
     1     0     0     0
     1     0     0     0

Hardware Cosimulation result of A1*B1:
    10     2     2     0
    14     4     4     0
    18     6     6     0
    22     8     8     0

Correct result of A1*B1:

    10     2     2     0
    14     4     4     0
    18     6     6     0
    22     8     8     0

Hardware Cosimulation result equals Correct result ,PASSED!

Matrix A2:
     1     2     3     4
     2     3     4     5
     3     4     5     6
     4     5     6     7

Matrix B2:
     2     3     4     5
     3     4     5     6
     4     5     6     7
     5     6     7     8

Hardware Cosimulation result of A2*B2:
    40    50    60    70
    54    68    82    96
    68    86   104   122
    82   104   126   148

Correct result of A2*B2:

    40    50    60    70
    54    68    82    96
    68    86   104   122
    82   104   126   148

Hardware Cosimulation result equals Correct result ,PASSED!

Matrix A3:
     1     2     3     4
     2     3     4     5
     3     4     5     6
     4     5     6     7

Matrix B3:
     1     7     3     4
     1     0     4     1
     1     0     1     0
     1     3     1     9

Hardware Cosimulation result of A3*B3:
    10    19    18    42
    14    29    27    56
    18    39    36    70
    22    49    45    84

Correct result of A3*B3:

    10    19    18    42
    14    29    27    56
    18    39    36    70
    22    49    45    84

Hardware Cosimulation result equals Correct result ,PASSED!

Matrix A4:
     2     3     4     5
     3     4     5     6
     4     5     6     7
     5     6     7     8

Matrix B4:
     2     3     4     5
     3     4     5     6
     4     5     6     7
     5     6     7     8

Hardware Cosimulation result of A4*B4:
    54    68    82    96
    68    86   104   122
    82   104   126   148
    96   122   148   174

Correct result of A4*B4:

    54    68    82    96
    68    86   104   122
    82   104   126   148
    96   122   148   174

Hardware Cosimulation result equals Correct result ,PASSED!
```

Figure 19

❖ **matrix – vector mode test results**

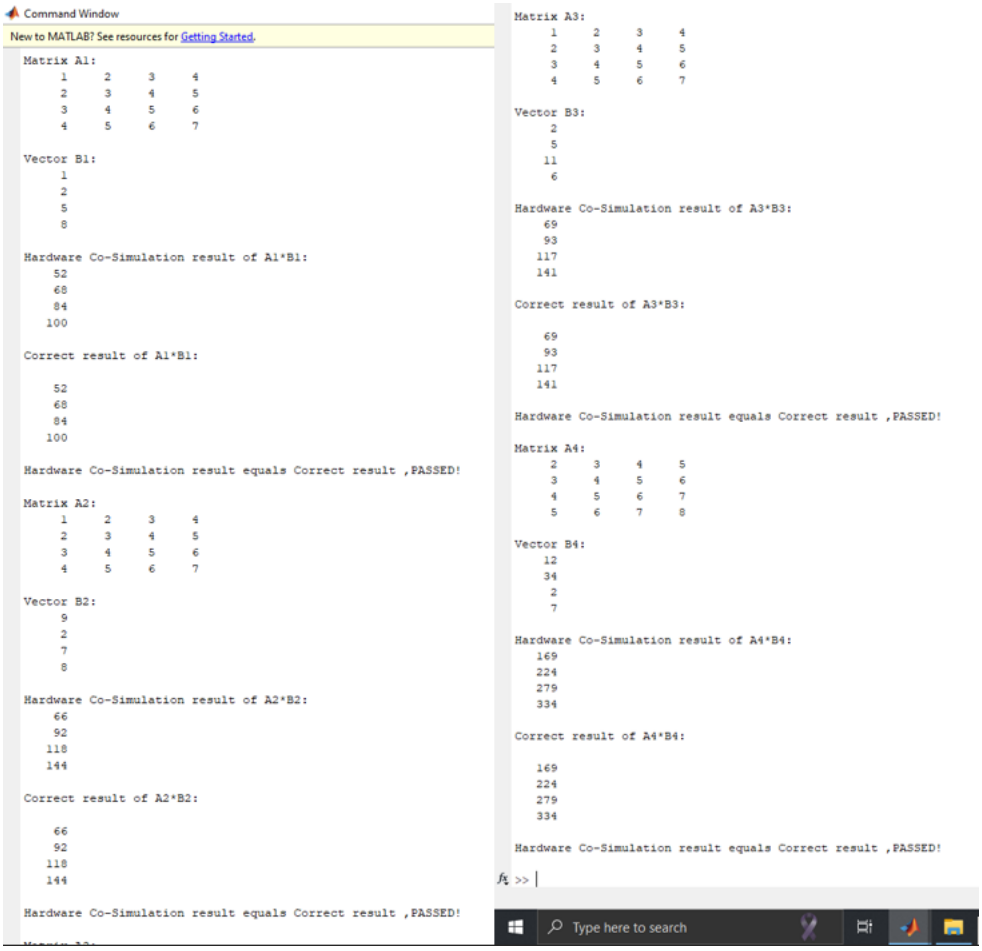


Figure 20

6. Resource utilization

The entire design uses 16 DSPs, 2413 LuTs, 3187 Registers. In the design, since multiplexers and accumulators are not used, and only adder delays are used to complete the functions of accumulation and data selection, a large number of LuTs and registers are required.

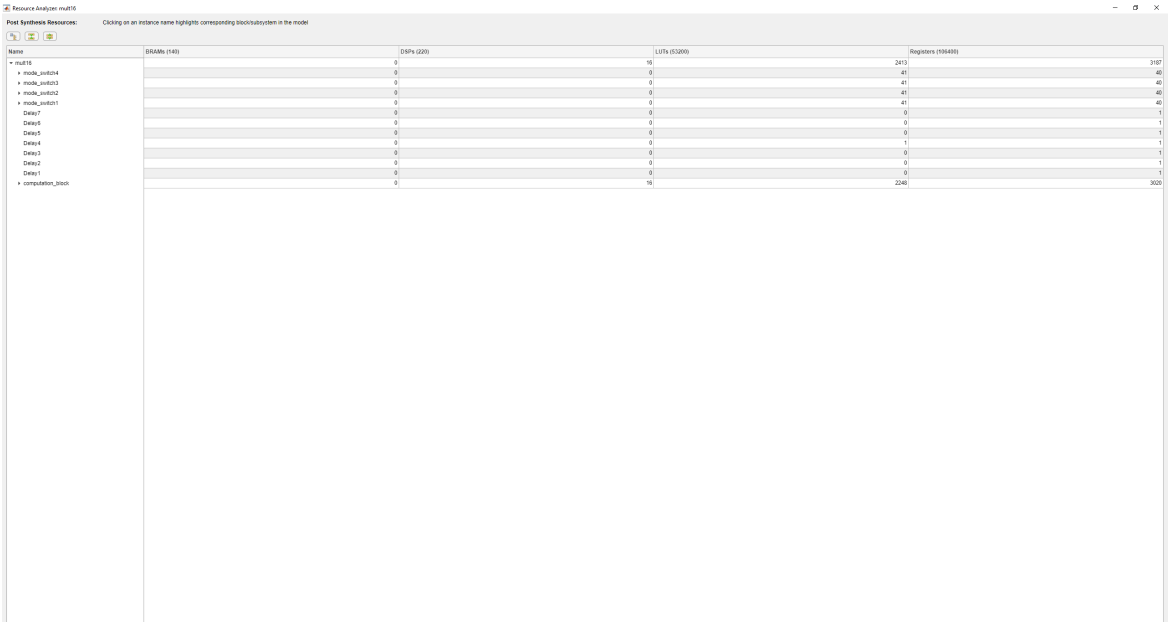


Figure 21

7. Timing information from the testing phase

In my original design, I set the lantency of all mults to 0, as shown in Figure 22.

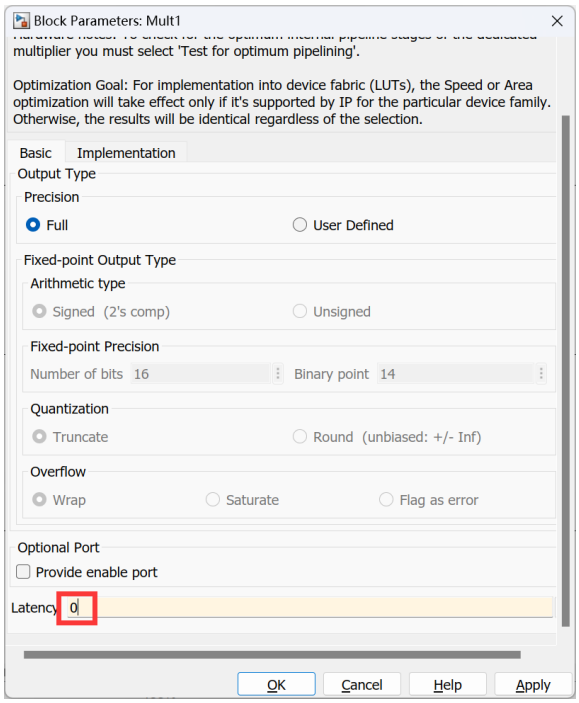


Figure 22

The data input and output timing is shown in the Figure 23. In this figure, the first row is the clock sequence, the second row is the input matrix sequence, and the third row is the output matrix sequence. It has the following properties:

1) The output of the matrix multiplication result lags its input by three clock cycles.
2) Matrix pairs can be entered continuously without waiting time.
3) The multiplication result can be output continuously without waiting time as well.
4) It can run at a clock frequency of 90.1Mhz (Period is 11ns).

Figure 23

In my final design, I set the latency of all mults to 4.

The data input and output timing is shown in the Figure 24. In this figure, the first row is the clock sequence, the second row is the input matrix sequence, and the third row is the output matrix sequence. It has the following properties:

1) The output of the matrix multiplication result lags its input by seven clock cycles.
2) Matrix pairs can be entered continuously without waiting time.
3) The multiplication result can be output continuously without waiting time as well.
4) It can run at a clock frequency of 125Mhz (Period is 8ns).

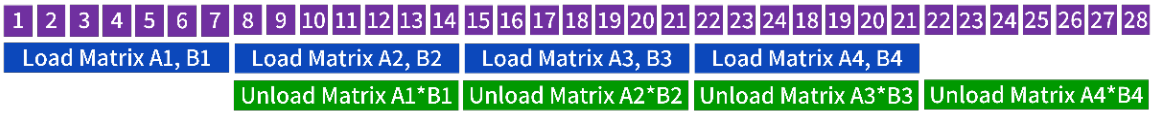


Figure 24

The final design passes timing analysis when the FPGA clock period is set to 8ns. As shown in Figure 25 and Figure 26.

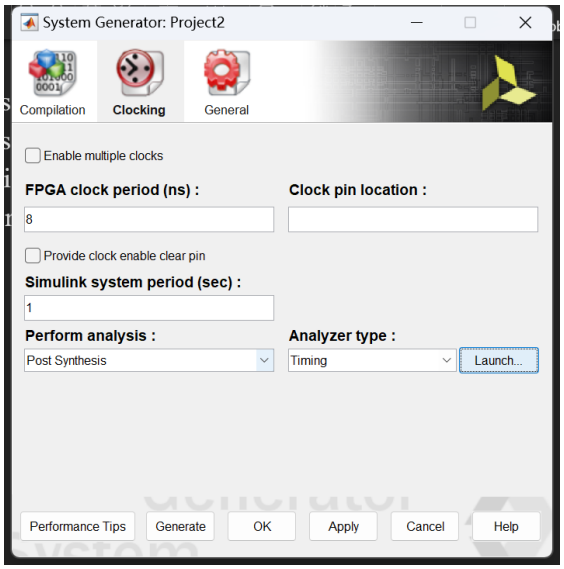


Figure 25

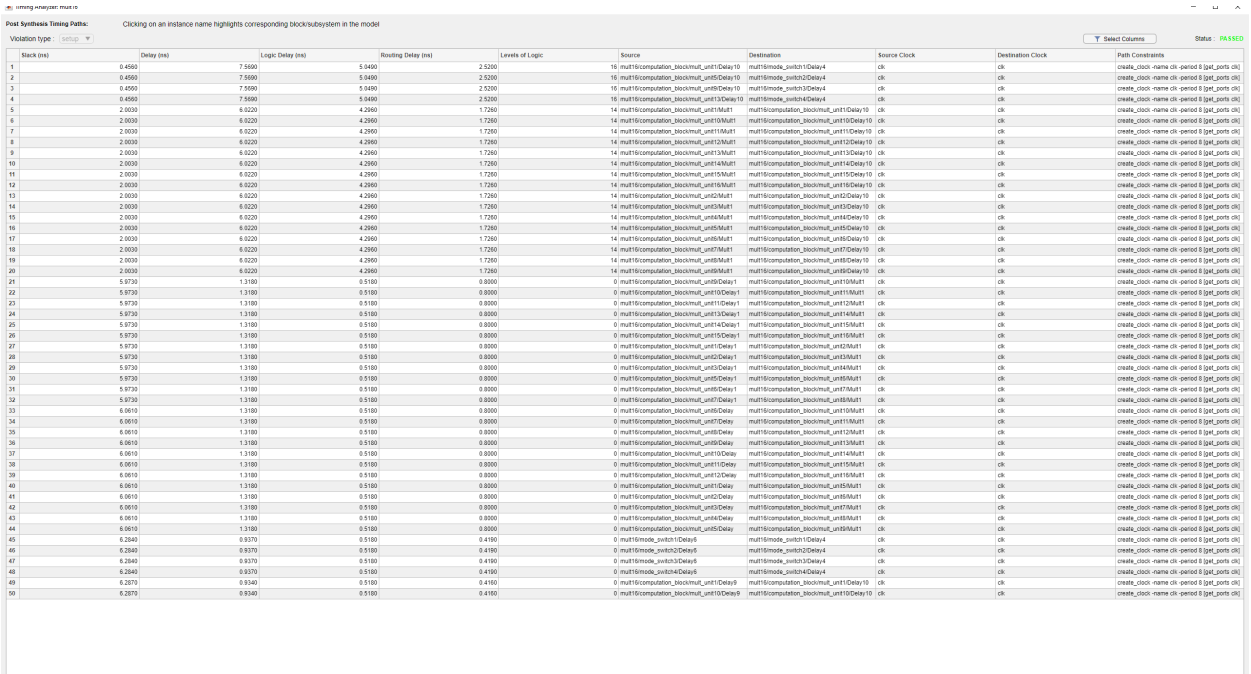


Figure 26

The final design does not pass timing analysis when the FPGA clock period is set to 7ns. As shown in Figure 27.

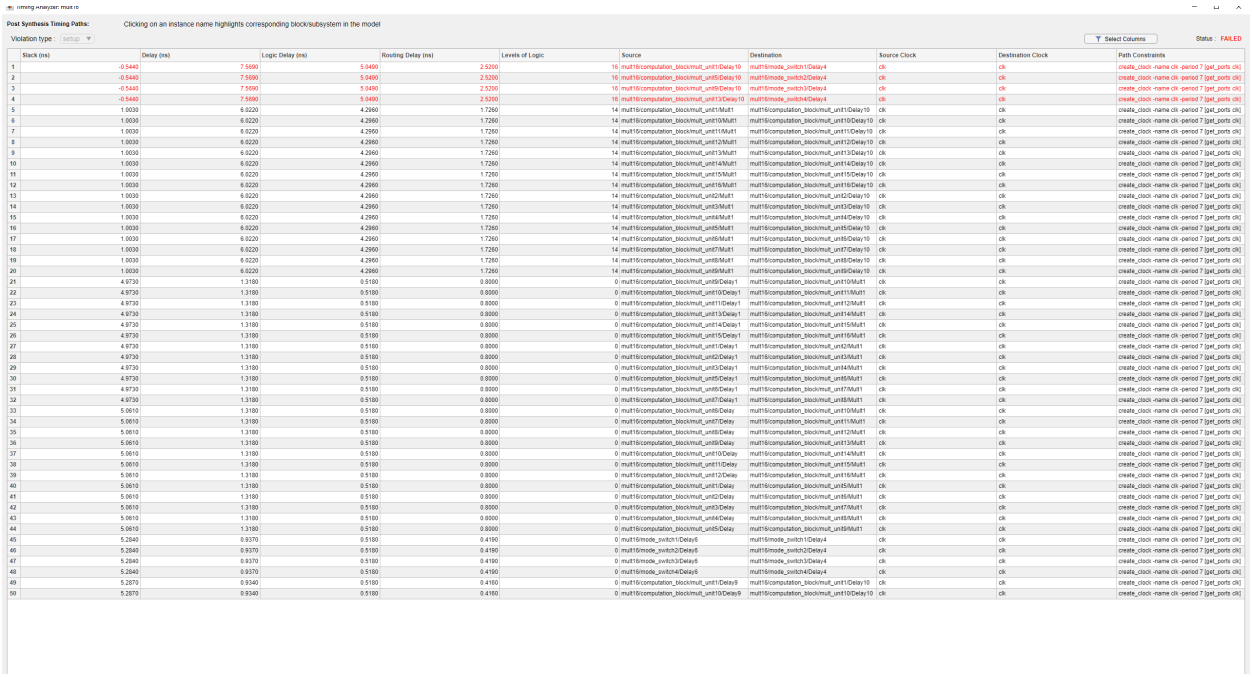


Figure 27

8. **Description of scalability of design and the ability to pipeline a series of matrix multiplications on your array and the latency and delay characteristics.**

More dimensional matrix calculations can be achieved by adding input row and column ports and output ports. Take the extension to five-by-five matrix multiplication as an example. The steps are as follows.

1) Add 1 row and 1 column of multiplication units and a mode switch block to the four-by-four matrix calculation unit, and connect them to the newly added Gateway In and Gateway Out ports, as shown in the Figure 28 and Figure 29. (to be extended to n dimensions, add n-4 columns and n-4 rows)

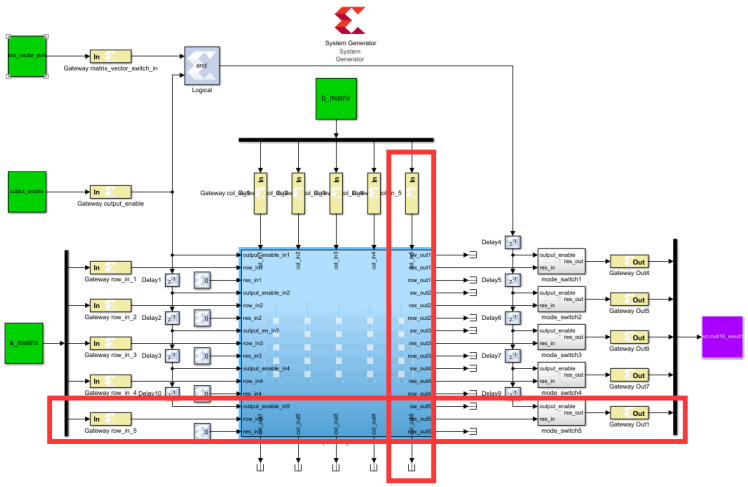


Figure 28

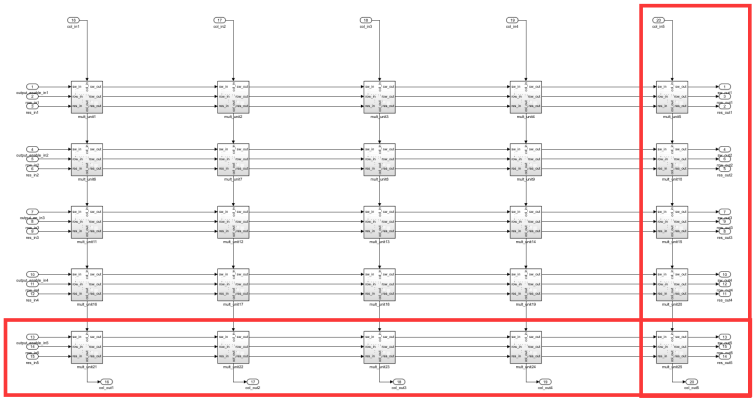


Figure 29

2) In each mult_unit block, add a delay block and an adder block, and connect as shown below (to be extended to n dimensions, add n delays and n adders).

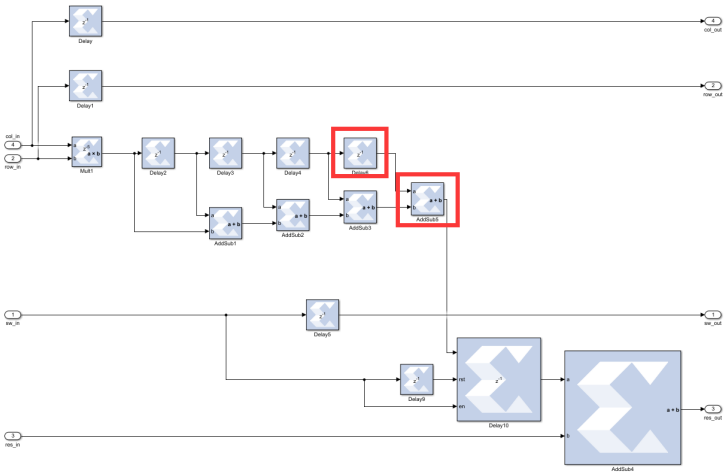


Figure 30

1) Change the latency of all 16 mults to 5 (to be extended to n dimensions, change the latency to n).

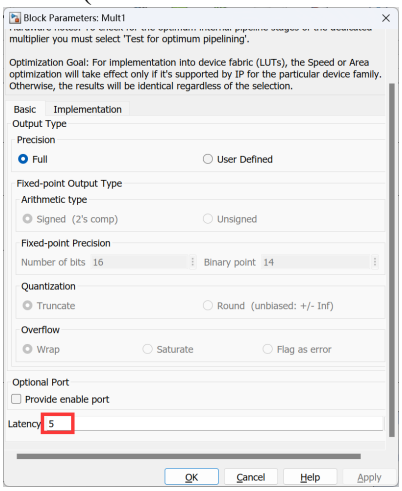


Figure 31

2) Increase the period of output_enable to 5 (n if to be extended to n dimensions). The waveform of output_enable after the change is shown in the Figure 32.

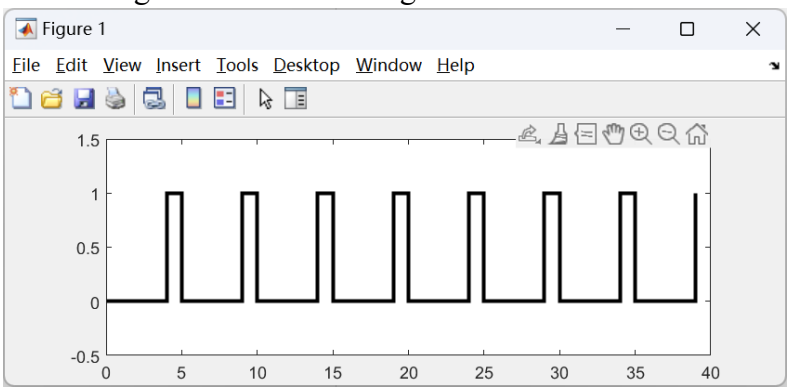


Figure 32

3) Partial test results of the five-by-five matrix multiplication calculation block.

```
Matrix A1:
     1     2     3     4     5
     2     3     4     5     6
     3     4     5     6     7
     4     5     6     7     8
     5     6     7     8     9

Matrix B1:
     0     7     8     9     5
     9     8     5     0     8
     0     9     0     9     7
     4     5     6     5     4
     5     5     6     7     2

Software Simulation result of A1*B1:
    59    95    72    91    68
    77   129    97   121    94
    95   163   122   151   120
   113   197   147   181   146
   131   231   172   211   172

Correct result of A1*B1:

    59    95    72    91    68
    77   129    97   121    94
    95   163   122   151   120
   113   197   147   181   146
   131   231   172   211   172

Software Simulation result equals Correct result ,PASSED!
```

4) Latency and delay characteristics
   a) The output of the matrix multiplication result lags its input by nine clock cycles.
   b) Each matrix input and output took 9 clock cycles.
   c) Matrix pairs can be entered continuously without waiting time.
   d) The multiplication result can be output continuously without waiting time as well.
   e) The pipeline of five-by-five matrix multiplication is shown as below.

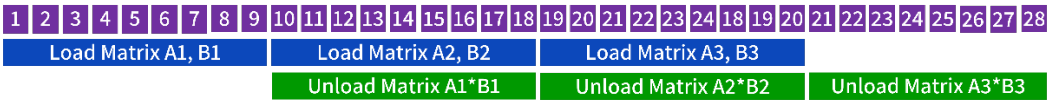


Figure 33

In a similar way, this matrix multiplier can be extended to more dimensions. If extended to an n-dimensional matrix multiplier, the input and output of the matrix takes 2*n-1 cycles, and the output of the matrix multiplication result lags its input by 2*n-1clock cycles.

9. Description of testing methodology

Here are all the files in the Project02 directory.

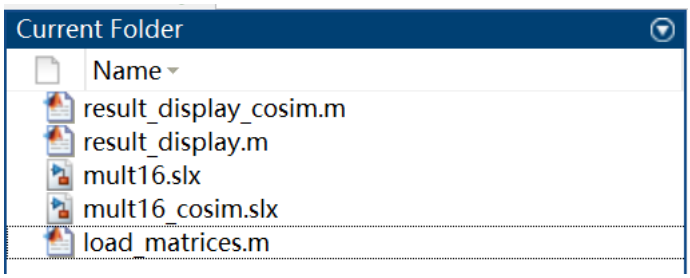


Figure 34

1) load_matrix.m is used to load test parameters, matrices and vector data.
2) result_display.m is used to display the simulation results and judge whether the simulation results are correct.
3) result_display_cosim.m is used to display the hardware co-simulation results and judge whether the simulation results are correct.
4) mult16.slx is the design file used for simulation.
5) mult16_cosim.slx is the design file used for hardware co-simulation.

❖ Software Simulation steps

1) Open Vitis Model Composer and change the working directory to the directory of Project02.

Figure 35

2) Run load_matrices.m.

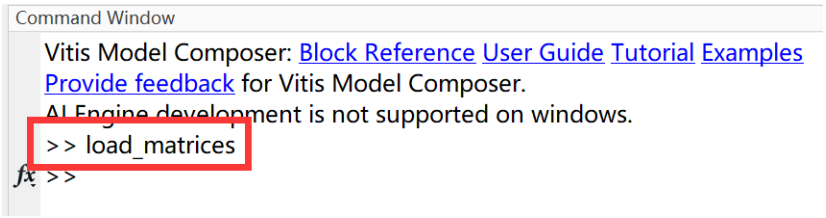


Figure 36

Matrix-matrix mode is default because mode_flag is set to 0.

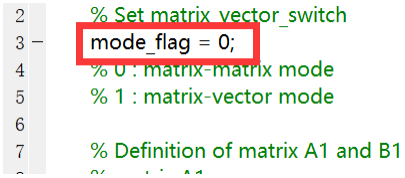


Figure 37

In this mode, matrices are multiplied two by two.

```
% Definition of matrix A1 and B1
% matrix A1 =
matrix_A(:,:,1) = [1 2 3 4; 2 3 4 5; 3 4 5 6; 4 5 6 7];
% matrix B1 =
matrix_B(:,:,1) = [1 2 2 0; 1 0 0 0; 1 0 0 0; 1 0 0 0];
vector_B(:,1) = [1 2 5 8]';

% Definition of matrix A2 and B2
% matrix A2 =
matrix_A(:,:,2) = [1 2 3 4; 2 3 4 5; 3 4 5 6; 4 5 6 7];
% matrix B2 =
matrix_B(:,:,2) = [2 3 4 5; 3 4 5 6; 4 5 6 7;5 6 7 8];
vector_B(:,2) = [9 2 7 8]';
```

A1*B1

A2*B2

Figure 38

3) Open mult16.slx, set the simulation time to 40, and click Run.

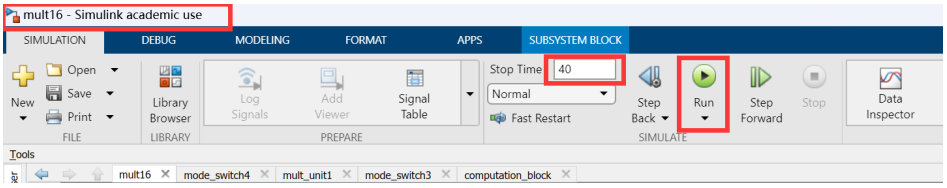


Figure 39

4) Run result_display.m, the result will be displayed in the Command Window. Here is the result.

```
Matrix A1:
     1     2     3     4
     2     3     4     5
     3     4     5     6
     4     5     6     7

Matrix B1:
     1     2     2     0
     1     0     0     0
     1     0     0     0
     1     0     0     0

Software Simulation result of A1*B1:
    10     2     2     0
    14     4     4     0
    18     6     6     0
    22     8     8     0

Correct result of A1*B1:

    10     2     2     0
    14     4     4     0
    18     6     6     0
    22     8     8     0

Software Simulation result equals Correct result ,PASSED!

Matrix A2:
     1     2     3     4
     2     3     4     5
     3     4     5     6
     4     5     6     7

Matrix B2:
     2     3     4     5
     3     4     5     6
     4     5     6     7
     5     6     7     8

Software Simulation result of A2*B2:
    40    50    60    70
    54    68    82    96
    68    86   104   122
    82   104   126   148

Correct result of A2*B2:

    40    50    60    70
    54    68    82    96
    68    86   104   122
    82   104   126   148

Software Simulation result equals Correct result ,PASSED!

Matrix A3:
     1     2     3     4
     2     3     4     5
     3     4     5     6
     4     5     6     7

Matrix B3:
     1     7     3     4
     1     0     4     1
     1     0     1     0
     1     3     1     9

Software Simulation result of A3*B3:
    10    19    18    42
    14    29    27    56
    18    39    36    70
    22    49    45    84

Correct result of A3*B3:

    10    19    18    42
    14    29    27    56
    18    39    36    70
    22    49    45    84

Software Simulation result equals Correct result ,PASSED!
```

```
Matrix A4:
     2     3     4     5
     3     4     5     6
     4     5     6     7
     5     6     7     8

Matrix B4:
     2     3     4     5
     3     4     5     6
     4     5     6     7
     5     6     7     8

Software Simulation result of A4*B4:
    54    68    82    96
    68    86   104   122
    82   104   126   148
    96   122   148   174

Correct result of A4*B4:

    54    68    82    96
    68    86   104   122
    82   104   126   148
    96   122   148   174

Software Simulation result equals Correct result ,PASSED!

>>
```

5) Modify the variable mode_flag in load_martices to 1, changing the setting to matrix-vector mode. Then rerun load_matrices.m and mult16.slx simulations. In this mode, matrix A is multiplied by vector B. As shown in Figure 40.

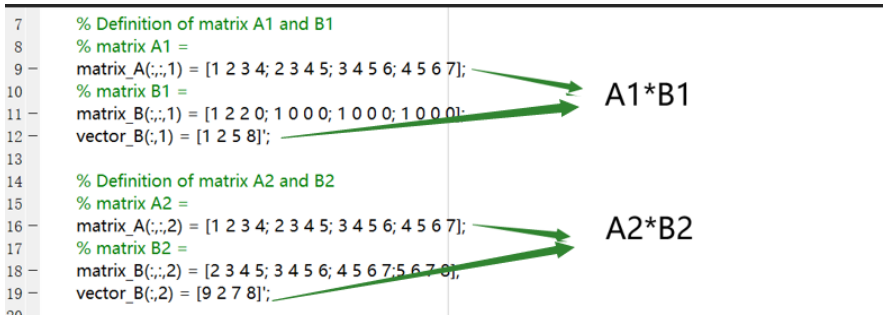


Figure 40

6) Rerun result_display.m, the result will be displayed in the Command Window. Here is the result.

```
Matrix A1:
     1     2     3     4
     2     3     4     5
     3     4     5     6
     4     5     6     7

Vector B1:
     1
     2
     5
     8

Software Simulation result of A1*B1:
    52
    68
    84
   100

Correct result of A1*B1:

    52
    68
    84
   100

Software Simulation result equals Correct result ,PASSED!

Matrix A2:
     1     2     3     4
     2     3     4     5
     3     4     5     6
     4     5     6     7

Vector B2:
     9
     2
     7
     8

Software Simulation result of A2*B2:
    66
    92
   118
   144

Correct result of A2*B2:

    66
    92
   118
   144

Software Simulation result equals Correct result ,PASSED!

Matrix A3:
     1     2     3     4
     2     3     4     5
     3     4     5     6
     4     5     6     7

Vector B3:
     2
     5
```

```
    11
     6

Software Simulation result of A3*B3:
    69
    93
   117
   141

Correct result of A3*B3:

    69
    93
   117
   141

Software Simulation result equals Correct result ,PASSED!

Matrix A4:
     2     3     4     5
     3     4     5     6
     4     5     6     7
     5     6     7     8

Vector B4:
    12
    34
     2
     7

Software Simulation result of A4*B4:
   169
   224
   279
   334

Correct result of A4*B4:

   169
   224
   279
   334

Software Simulation result equals Correct result ,PASSED!

>>
```

❖ Hardware Cosimulation steps

1) Open Vitis Model Composer and change the working directory to the directory of Project02.

Figure 41

2) Run load_matrices.m.

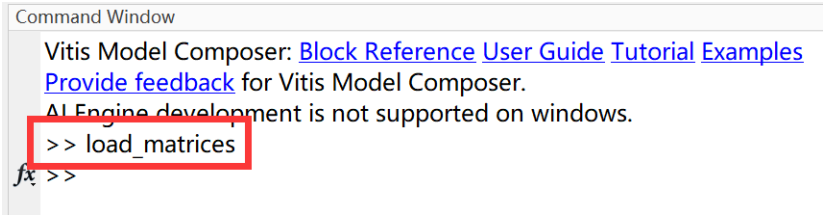
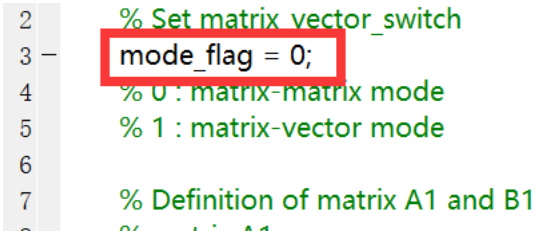


Figure 42

Matrix-matrix mode is default because mode_flag is set to 0.



In this mode, matrices are multiplied two by two.

```
% Definition of matrix A1 and B1
% matrix A1 =
matrix_A(:,:,1) = [1 2 3 4; 2 3 4 5; 3 4 5 6; 4 5 6 7];
% matrix B1 =
matrix_B(:,:,1) = [1 2 2 0; 1 0 0 0; 1 0 0 0; 1 0 0 0];
vector_B(:,1) = [1 2 5 8]';

% Definition of matrix A2 and B2
% matrix A2 =
matrix_A(:,:,2) = [1 2 3 4; 2 3 4 5; 3 4 5 6; 4 5 6 7];
% matrix B2 =
matrix_B(:,:,2) = [2 3 4 5; 3 4 5 6; 4 5 6 7;5 6 7 8];
vector_B(:,2) = [9 2 7 8]';
```

A1*B1

A2*B2

Figure 43

3) Open mult16_cosim.slx, set the simulation time to 40, and click Run.

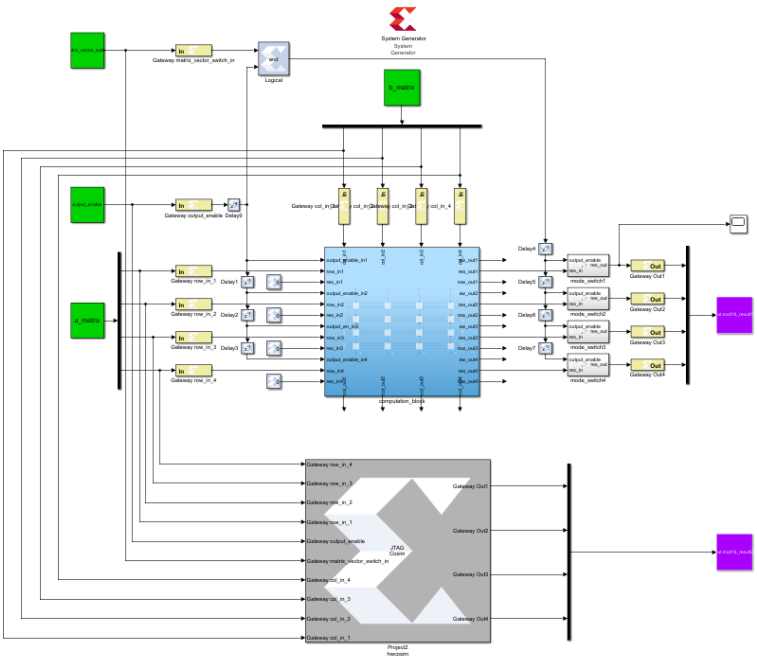


Figure 44

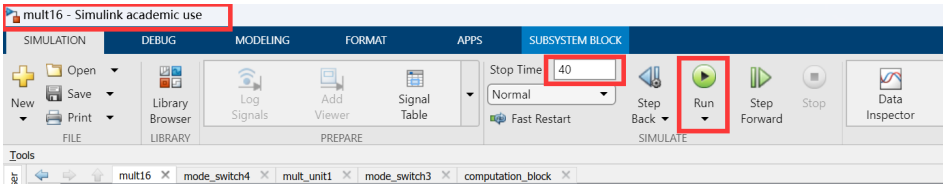


Figure 45

4) Run result_display.m, the result will be displayed in the Command Window. Here is the result.

```
Matrix A1:
     1     2     3     4
     2     3     4     5
     3     4     5     6
     4     5     6     7

Matrix B1:
     1     2     2     0
     1     0     0     0
     1     0     0     0
     1     0     0     0

Hardware Co-Simulation result of A1*B1:
    10     2     2     0
    14     4     4     0
    18     6     6     0
    22     8     8     0

Correct result of A1*B1:

    10     2     2     0
    14     4     4     0
    18     6     6     0
    22     8     8     0
```

```
Hardware Co-Simulation result equals Correct result ,PASSED!

Matrix A2:
     1     2     3     4
     2     3     4     5
     3     4     5     6
     4     5     6     7

Matrix B2:
     2     3     4     5
     3     4     5     6
     4     5     6     7
     5     6     7     8

Hardware Co-Simulation result of A2*B2:
    40    50    60    70
    54    68    82    96
    68    86   104   122
    82   104   126   148

Correct result of A2*B2:

    40    50    60    70
    54    68    82    96
    68    86   104   122
    82   104   126   148

Hardware Co-Simulation result equals Correct result ,PASSED!

Matrix A3:
     1     2     3     4
     2     3     4     5
     3     4     5     6
     4     5     6     7

Matrix B3:
     1     7     3     4
     1     0     4     1
     1     0     1     0
     1     3     1     9

Hardware Co-Simulation result of A3*B3:
    10    19    18    42
    14    29    27    56
    18    39    36    70
    22    49    45    84

Correct result of A3*B3:

    10    19    18    42
    14    29    27    56
    18    39    36    70
    22    49    45    84

Hardware Co-Simulation result equals Correct result ,PASSED!

Matrix A4:
     2     3     4     5
     3     4     5     6
     4     5     6     7
     5     6     7     8

Matrix B4:
     2     3     4     5
     3     4     5     6
     4     5     6     7
     5     6     7     8

Hardware Co-Simulation result of A4*B4:
    54    68    82    96
    68    86   104   122
    82   104   126   148
    96   122   148   174

Correct result of A4*B4:

    54    68    82    96
    68    86   104   122
    82   104   126   148
    96   122   148   174

Hardware Co-Simulation result equals Correct result ,PASSED!

>>
```

5) Modify the variable mode_flag in load_martices to 1, changing the setting to matrix-vector mode. Then rerun load_matrices.m and mult16.slx simulations. In this mode, matrix A is multiplied by vector B.
6) Rerun result_display.m, the result will be displayed in the Command Window. Here is the result.

```
Matrix A1:
     1     2     3     4
     2     3     4     5
     3     4     5     6
     4     5     6     7

Vector B1:
     1
     2
     5
     8

Hardware Co-Simulation result of A1*B1:
    52
    68
    84
   100

Correct result of A1*B1:

    52
    68
    84
   100

Hardware Co-Simulation result equals Correct result ,PASSED!

Matrix A2:
     1     2     3     4
     2     3     4     5
     3     4     5     6
     4     5     6     7

Vector B2:
     9
```

```
     2
     7
     8

Hardware Co-Simulation result of A2*B2:
    66
    92
   118
   144

Correct result of A2*B2:

    66
    92
   118
   144

Hardware Co-Simulation result equals Correct result ,PASSED!

Matrix A3:
     1     2     3     4
     2     3     4     5
     3     4     5     6
     4     5     6     7

Vector B3:
     2
     5
    11
     6

Hardware Co-Simulation result of A3*B3:
    69
    93
   117
   141

Correct result of A3*B3:

    69
    93
   117
   141

Hardware Co-Simulation result equals Correct result ,PASSED!

Matrix A4:
     2     3     4     5
     3     4     5     6
     4     5     6     7
     5     6     7     8

Vector B4:
    12
    34
     2
     7

Hardware Co-Simulation result of A4*B4:
   169
   224
   279
   334

Correct result of A4*B4:

   169
   224
   279
   334

Hardware Co-Simulation result equals Correct result ,PASSED!

>>
```