ECE M.Eng project report

Systolic implementation based on SystemVerilog on AWS

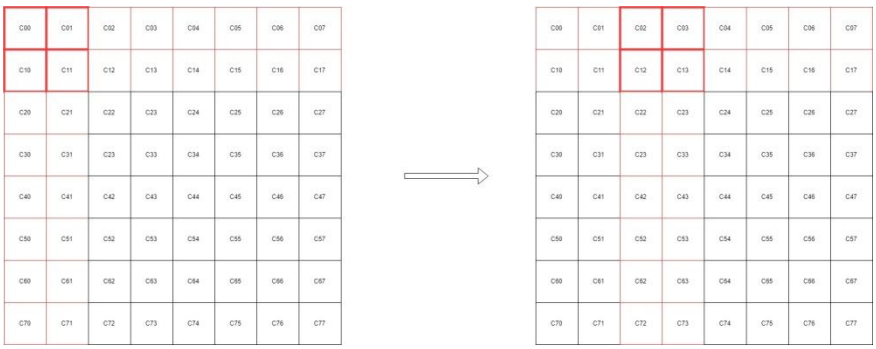
In this design we use the enviornment is quartus ii 18.0 and modelsim result to show the result and implementation of the systolic array.

In this report we want use the tailing algorithm to implement the matrix multiplication.

```
parameter data_size = 8
parameter block_size = 2
parameter matrix_size = 8
/* Loop over all the tiles, stride by tile size */
for ( int i = 0; i < matrix_size; i += block_size )
    for ( int j = 0; j < matrix_size; i += block_size )
        for ( int k = 0; k < matrix_size; k += block_size)
            /* Regular multiply inside the tiles */
                for ( int y = i; y < i+block_size; y++ )
                    for ( int x = j; x < j+block_size; x++ )
                        for ( int z = k; z < k+block_size; z++ )
                            C(y,x) += A(y,z)*B(z,x);
```

For each block calculation used in the systolic array algorithm is that we need to calculate each subblock result and push back to memory. Each of the systolic array architecutre will execute the red word function which seems like a small matrix-matrix multiplication. So the tiling algorithm used for the systolic array should be like:

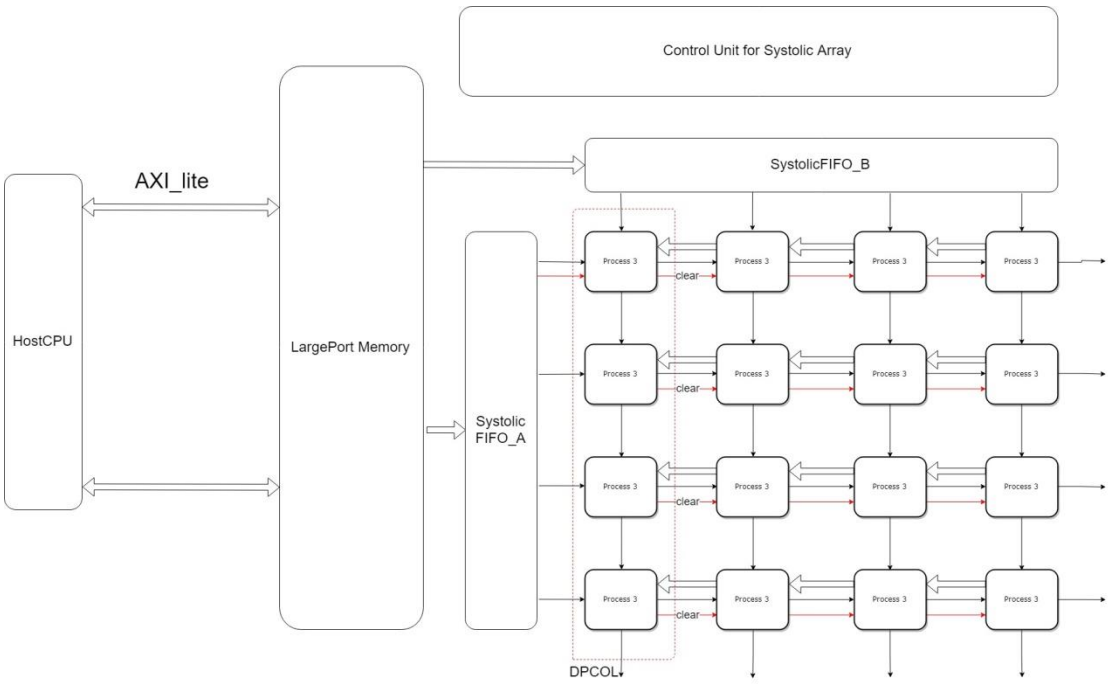
```
parameter data_size = 8
parameter block_size = 2
parameter matrix_size = 8
/* Loop over all the tiles, stride by tile size */
for ( int i = 0; i < matrix_size; i += block_size )
    for ( int j = 0; j < matrix_size; i += block_size )
        systolic_array_cal[i,i+block_size],[j,j+block_size]
```



So for the tiling algorithm introduced before, we can assume that the systolic array size = 2 and matrix_size = 8, so the first iteration of the systolic array calculation is that we need to implement the calculation result of C00,C01,C10,C11 is should calculate 8 times in each processelement. And the second iteration is that we need to implement the calculation result of C02,C03,C12,C13 is

should calculate 8 times in each processelement.

For the algorithm, it will reduce the matrix-matrix mutiplication time complexity from O(data_size^3) to O(data_size^3/block_size^2)



We have several models which have different use:

1, systolictop (top-level)

2, systolicDP (generate model)

3, systolicDPcol (generate model)

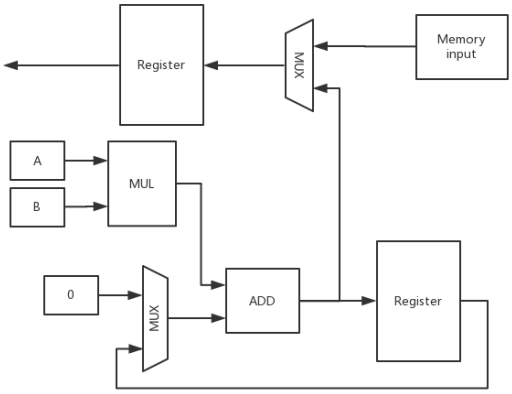
4, processelement (calculate the result based on the PE)

5, systolicCU (control unit to give the memory and PE signal to clear the value)

6, systolicFIFO (FIFO to generate systolic stream)

Introduction of processelement

The basic design of the systolic elements is based on the following:



Which should have the following input and output:

Control signals:

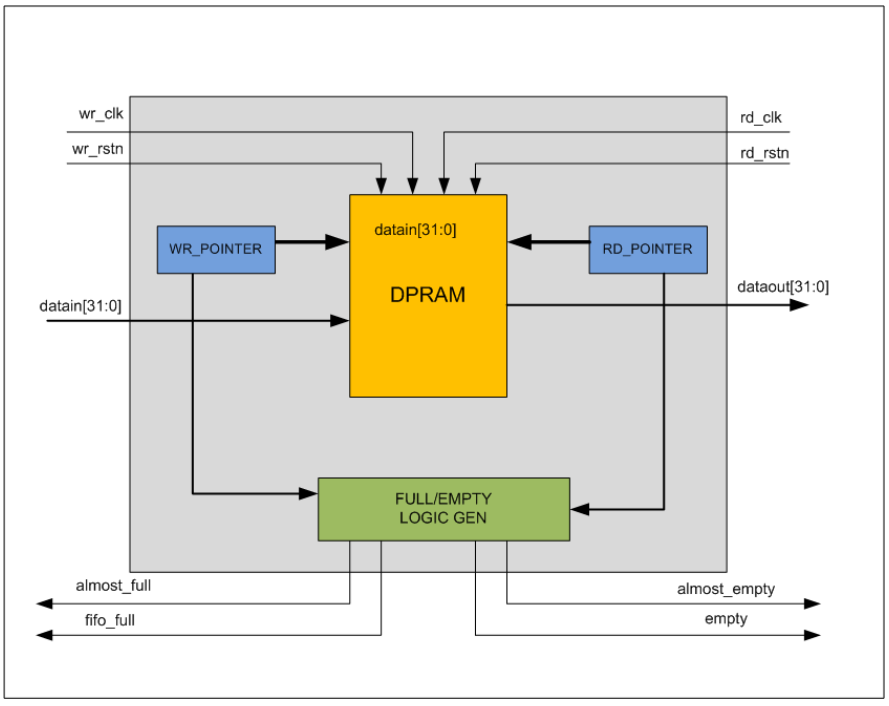
```
input    wire cal_ele_cho,
input    wire mem_ele_cho,
input    wire mem_change,
```

Data stream:

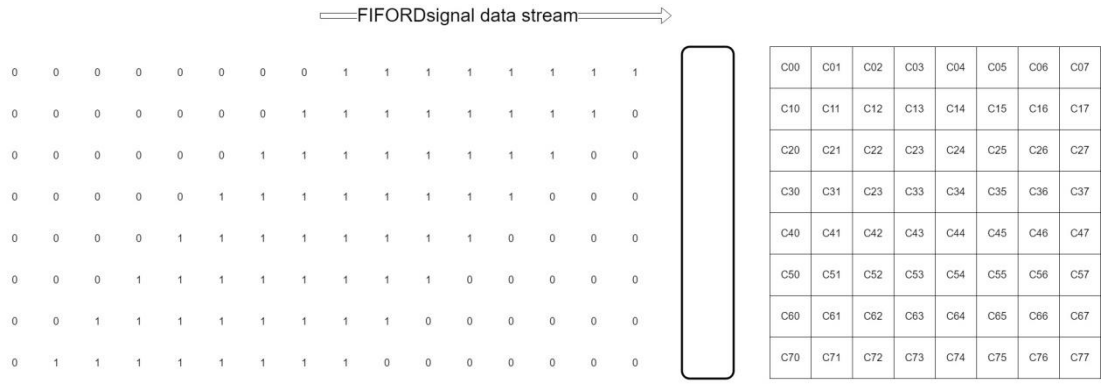
```
input    wire [2*data_size:0] MEM_in
input    wire [data_size:0] MUL_A_in,
input    wire [data_size:0] MUL_B_in,
output reg [data_size:0] MUL_A_out,
output reg [data_size:0] MUL_B_out,
output reg [data_size:0] MEM_out,
output reg [data_size:0] MUL_C_out,
```

The data will be pushed forward and memory data should be pushed back. There will be a register for the data A and data B temp storage

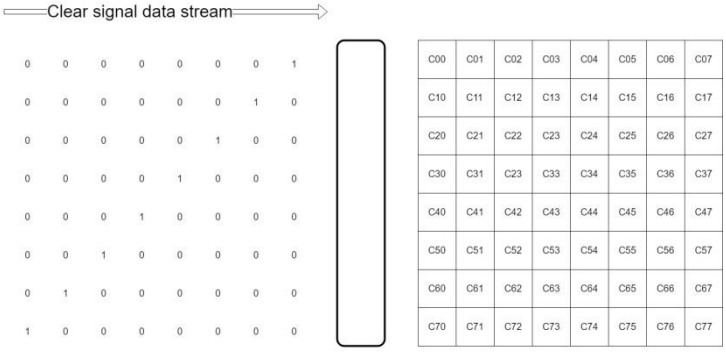
Introduction of systolicFIFO (FIFO to generate systolic stream)



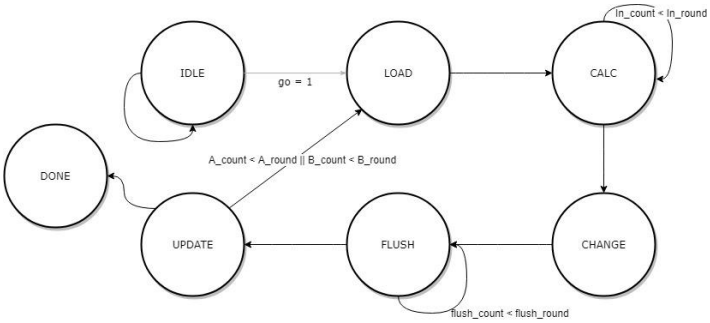
We use the An asynchronous FIFO refers to a FIFO design where data values are written to a FIFO buffer from one clock domain and the data values are read from the same FIFO buffer from the same clock domain, where the two clock domains are synchronous to each other. For synchronous FIFO design (a FIFO where writes to, and reads from the FIFO buffer are conducted in the same clock domain), we use count register counts the number of writes to, and reads from the FIFO buffer to increment (on FIFO write but no read), decrement (on FIFO read but no write) or hold (no writes and reads, or simultaneous write and read operation) the current fill value of the FIFO buffer. The FIFO is full when the FIFO counter reaches a predetermined full value and the FIFO is empty when the FIFO counter is zero.



So the write signal for the systolic array is atomic and read signal for the systolic array is seems like the picture showed. The Read signal should seems like a data stream to make the data transferred into systolic datapath is also streams-like.



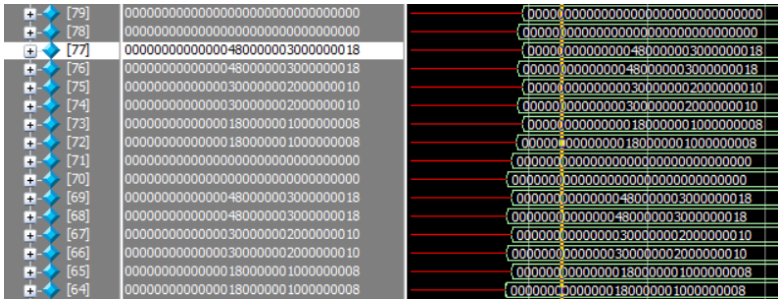
The clear signal for the systolic array is seems like the picture showed. The clear signal should seems like a data stream to make the data transferred into systolic datapath is also streams-like. And to make the systolic array start calculating.



Matrix_A	01	02	03	04	00	01	02	03	
Matrix_B	01	02	03	04	00	01	02	03	

the test case with memory result

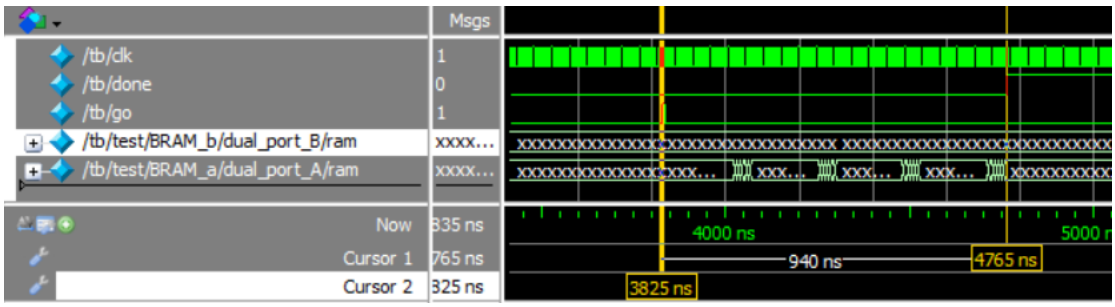
// 1 1 1 1 1 1 1 1
// 2 2 2 2 2 2 2 2
// 3 3 3 3 3 3 3 3
// 0 0 0 0 0 0 0 0
// 1 1 1 1 1 1 1 1
// 2 2 2 2 2 2 2 2
// 3 3 3 3 3 3 3 3
// 0 0 0 0 0 0 0 0
// the matrix of B is
// 1 2 3 0 1 2 3 0
// 1 2 3 0 1 2 3 0
// 1 2 3 0 1 2 3 0
// 1 2 3 0 1 2 3 0
// 1 2 3 0 1 2 3 0
// 1 2 3 0 1 2 3 0
// 1 2 3 0 1 2 3 0
// 1 2 3 0 1 2 3 0
// so the result of matrix A * matrix B is
// 8 16 24 0 8 16 24 0
// 16 32 48 0 16 32 48 0
// 24 48 72 0 24 48 72 0
// 0 0 0 0 0 0 0 0
// 8 16 24 0 8 16 24 0
// 16 32 48 0 16 32 48 0
// 24 48 72 0 24 48 72 0
// 0 0 0 0 0 0 0 0



Time analysis:

So we need up to 94 cycles to deal with 8*8 matrix in 4*4 systolic array which usually in CPU it

will cost 8*8*8 = 512 cycles.



Still on going!