FAET630004: AI-Core and RISC Architecture **(Due: 4/23/21)**

Homework Assignment #2

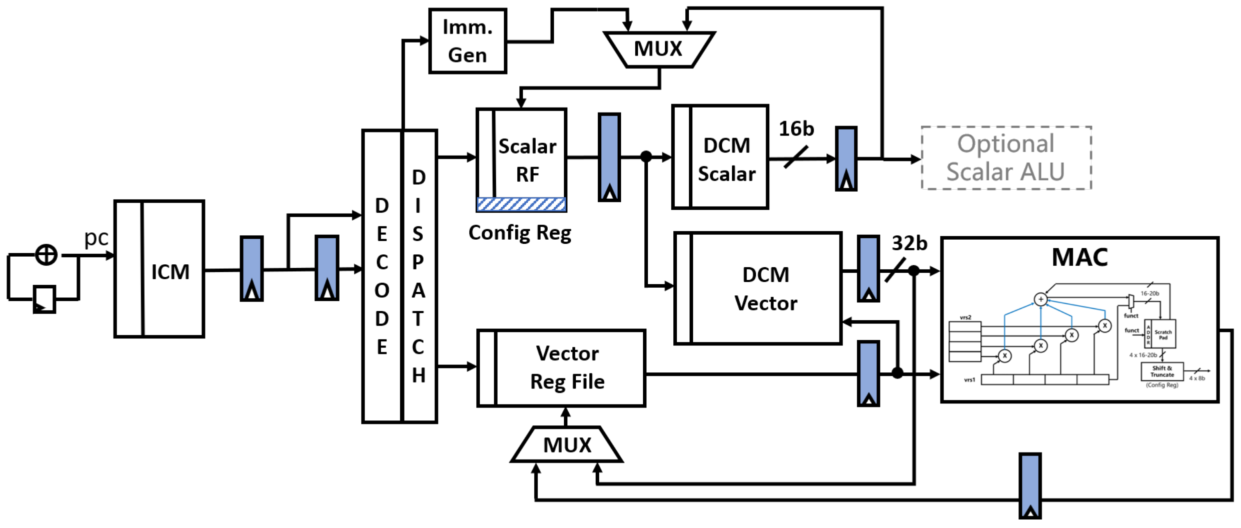
Instructor: Chixiao Chen *Name:* Chunyu Wang, *FudanID:* 20210860017

- This HW counts 15% of your final score, please treat it carefully.
- Please submit the electronic copy via mail: faet_english@126.com before the due date.
- It is encouraged to use LaTeX to edit it, the source code of the assignment is available via: https://www.overleaf.com/read/qnqfpcmqvchp
- You can also open it by Office Word, and save it as a .doc file for easy editing. Also, you can print it out, complete it and scan it by your cellphone.
- The assignment needs verilog/SV simulation. It is suggested to use Vivado from Xilinx to complete the simulation. If you do not want to install a local verilog simulator, please use an online tool: https://www.edaplayground.com/, you need register for save.
- You can answer the assignment either in Chinese or English

Problem 1: Implement a matrix multiplier on a RISC Core (8+7=15 points)

Using the following ISA and hardware architecture to compute $\mathbf{A} \cdot \mathbf{B} + \mathbf{C}$, where $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are 8×8 matrices. Each element in them are signed integers with 8b length.

Base: Scalar Instruction		Extension: Vector Instruction	
LOAD	rd , rs , imm_{5b}	Vload	vrd_{3b} , rs , imm_{5b}
Store	/ , rs1 , rs2	Vstore	imm_{5b} , rs , vrs_{3b}
MOV	rd , imm_{8b} , funct_{1b}	VMAC	vrd_{3b} , vrs1_{3b} , vrs2_{3b} , funct_{4b}



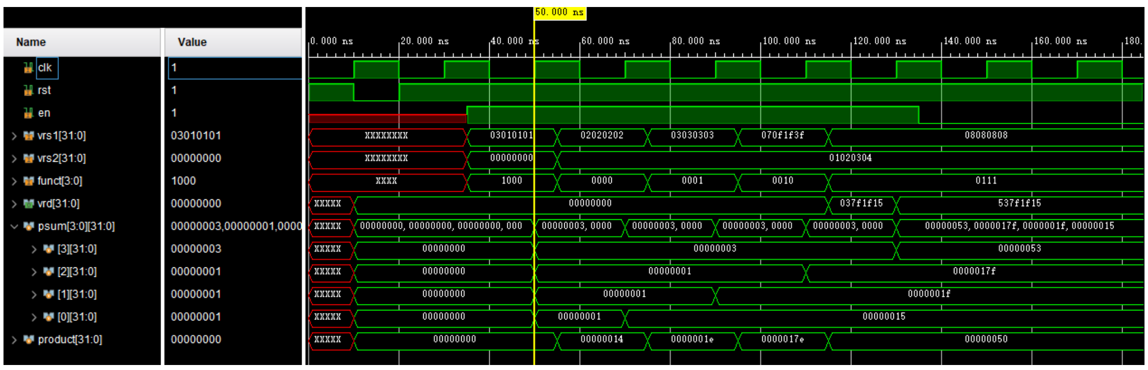
Implementatino of a VMAC module:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Author: Wang Chunyu
// Description:
// Additional Comments:
//////////////////////////////////////////////////////////////////////////////////
module VMAC(
     input         clk,rst,en,
     input    [31:0]   vrs1,
     input    [31:0]   vrs2,
     input    [3:0] funct,

     output  reg   [31:0]   vrd
    );
   reg   [31:0]   psum[3:0];
   reg   [31:0]   product;
   integer    i;

   always @(*) begin
     if (!funct[3]) begin //mac, do not set bias
          product=0;
       for(i=0;i<4;i=i+1)
          product=product+$signed(vrs1[i*8+:8])*$signed(vrs2[i*8+:8]);

       //output, shift and truncate
       if(funct[2]==1'b1) begin
          for(i=0;i<4;i=i+1) begin
             // shift 0, truncate, the lower 8 bit as output. overflow handling, remain sign bit
             if((~psum[i][31]) && (psum[i]>31'b0000000000000000000000001111111))
                vrd[i*8+:8]=8'b01111111;
             else if((psum[i][31]) && (psum[i]<31'b1111111111111111111111110000000))
                vrd[i*8+:8]=8'b10000000;
             else
                vrd[i*8+:8]=psum[i][7:0];
          end
       end
       else begin
            vrd=0;
       end
     end
     else begin
       product=0;
       vrd=0;
     end
   end
   // write psum (scratch pad)
   always@(posedge clk or negedge rst) begin
      if(!rst) begin
         for(i=0;i<4;i=i+1)
          psum[i]<=0;
      end
     else if(en) begin
       if(funct[3]==1'b1) // set bias
          for(i=0;i<4;i=i+1)
             psum[i]<=$signed(vrs1[i*8+:8]);
       else begin// 4 mac
          psum[funct[1:0]]<=psum[funct[1:0]]+product;
       end
     end
   end
endmodule
```

VMAC simulation:

$$\begin{bmatrix} 03 & 01 & 01 & 01 \end{bmatrix} + \begin{bmatrix} 01 & 02 & 03 & 04 \end{bmatrix} * \begin{bmatrix} 02 & 03 & 07 & 08 \\ 02 & 03 & 0f & 08 \\ 02 & 03 & 1f & 08 \\ 02 & 03 & 3f & 08 \end{bmatrix} = \begin{bmatrix} 53 & 7f & 1f & 15 \end{bmatrix}$$

(a)



Name	Value
clk	1
rst	1
en	1
vrs1[31:0]	03010101
vrs2[31:0]	00000000
funct[3:0]	1000
vrd[31:0]	00000000
psum[3:0][31:0]	00000003,00000001,0000
[3][31:0]	00000003
[2][31:0]	00000001
[1][31:0]	00000001
[0][31:0]	00000001
product[31:0]	00000000

50.000 ns
0.000 ns
20.000 ns
40.000 ns
60.000 ns
80.000 ns
100.000 ns
120.000 ns
140.000 ns
160.000 ns
180.
vrs1: XXXXXXXX, 03010101, 02020202, 03030303, 070f1f3f, 08080808
vrs2: XXXXXXXX, 00000000, 01020304
funct: XXXX, 1000, 0000, 0001, 0010, 0111
vrd: XXXXX, 00000000, 037f1f15, 537f1f15
psum: XXXXX, 00000000, 00000000, 00000000, 000; 00000003, 0000; 00000003, 0000; 00000003, 0000; 00000003, 0000; 00000053, 0000017f, 0000001f, 00000015
[3]: XXXXX, 00000000, 00000003, 00000053
[2]: XXXXX, 00000000, 00000001, 0000017f
[1]: XXXXX, 00000000, 00000001, 0000001f
[0]: XXXXX, 00000000, 00000001, 00000015
product: XXXXX, 00000000, 00000014, 0000001e, 0000017e, 00000050

(b)

Figure 1: Simulation (ZoomIn to get details)

The simulation data and waveform is shown in Fig.1, and the result *vrd* is correct.

(a) Write the entire assembly code for computation. (hints: 8 indexed vector register file is not sufficient for 8x8 matrix.)

The input **A**, **B** and **C** can be represented as partitioned matrix form **P**, **Q** and **R** following their storage characteristics, as shown in Fig.2(ZoomIn if you cannot see it). So we can process a 8x8 matrix multiplication by using several 4x4 VMACs.

Then we can write assembly code as follows. Every assembly code module(e.g. line 6˜26, 28˜48, ...) computes continous 4 elements(row) in output matrix Y.

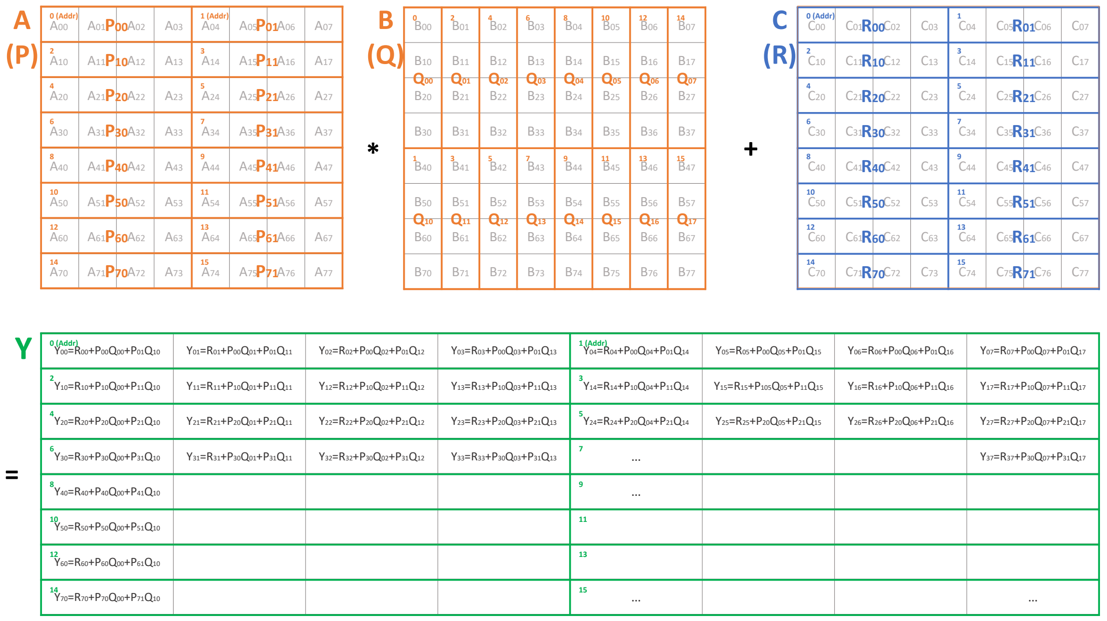


Figure 2: 8x8 matrix multiplication using a 4x4 VMAC (ZoomIn to get details)

Assembly Code:

```
MOV      r1,   $0,  [AB],     $0    // address bias (C)R
MOV      r2,   $0,  [AW],     $0    // address weights (B)Q
MOV      r3,   $0,  [AX],     $0    // address input X (A)P
MOV      r4,   $0,  [AY],     $0    // address output Y = AB+C

Vload   vr0,   r1,   $0             // load bias: R00
VMAC      /,  vr0,    /, $1000b     // mac init, set bias
Vload   vr0,   r2,   $0             // load weights: Q00
Vload   vr1,   r2,   $2             // load weights: Q01
Vload   vr2,   r2,   $4             // load weights: Q02
Vload   vr3,   r2,   $6             // load weights: Q03
Vload   vr4,   r3,   $0             // load input: P00
VMAC      /,  vr0,  vr4, $0000b
VMAC      /,  vr1,  vr4, $0001b
VMAC      /,  vr2,  vr4, $0010b
VMAC      /,  vr3,  vr4, $0011b
Vload   vr0,   r2,   $1             // load weights: Q10
Vload   vr1,   r2,   $3             // load weights: Q11
Vload   vr2,   r2,   $5             // load weights: Q12
Vload   vr3,   r2,   $7             // load weights: Q13
Vload   vr4,   r3,   $1             // load input: P01
VMAC      /,  vr0,  vr4, $0000b
VMAC      /,  vr1,  vr4, $0001b
VMAC      /,  vr2,  vr4, $0010b
VMAC    vr7,  vr3,  vr4, $0111b
Vstore   $0,   r4,  vr7             // store output:Y00,Y01,Y02,Y03

Vload   vr0,   r1,   $1             // load bias: R01
VMAC      /,  vr0,    /, $1000b     // mac init, set bias
Vload   vr0,   r2,   $8             // load weights: Q04
Vload   vr1,   r2,  $10             // load weights: Q05
Vload   vr2,   r2,  $12             // load weights: Q06
Vload   vr3,   r2,  $14             // load weights: Q07
Vload   vr4,   r3,   $0             // load input: P00
VMAC      /,  vr0,  vr4, $0000b
VMAC      /,  vr1,  vr4, $0001b
VMAC      /,  vr2,  vr4, $0010b
VMAC      /,  vr3,  vr4, $0011b
Vload   vr0,   r2,   $9             // load weights: Q14
Vload   vr1,   r2,  $11             // load weights: Q15
Vload   vr2,   r2,  $13             // load weights: Q16
Vload   vr3,   r2,  $15             // load weights: Q17
Vload   vr4,   r3,   $1             // load input: P01
VMAC      /,  vr0,  vr4, $0000b
VMAC      /,  vr1,  vr4, $0001b
VMAC      /,  vr2,  vr4, $0010b
VMAC    vr7,  vr3,  vr4, $0111b
Vstore   $1,   r4,  vr7             // store output:Y04,Y05,Y06,Y07

Vload   vr0,   r1,   $2             // load bias: R10
VMAC      /,  vr0,    /, $1000b     // mac init, set bias
Vload   vr0,   r2,   $0             // load weights: Q00
Vload   vr1,   r2,   $2             // load weights: Q01
Vload   vr2,   r2,   $4             // load weights: Q02
Vload   vr3,   r2,   $6             // load weights: Q03
Vload   vr4,   r3,   $2             // load input: P10
VMAC      /,  vr0,  vr4, $0000b
VMAC      /,  vr1,  vr4, $0001b
VMAC      /,  vr2,  vr4, $0010b
VMAC      /,  vr3,  vr4, $0011b
Vload   vr0,   r2,   $1             // load weights: Q10
Vload   vr1,   r2,   $3             // load weights: Q11
Vload   vr2,   r2,   $5           // load weights: Q12
Vload   vr3,   r2,   $7           // load weights: Q13
Vload   vr4,   r3,   $3           // load input: P11
VMAC      /,   vr0,  vr4, $0000b
VMAC      /,   vr1,  vr4, $0001b
VMAC      /,   vr2,  vr4, $0010b
VMAC    vr7,   vr3,  vr4, $0111b
Vstore   $2,   r4,   vr7          // store output:Y10,Y11,Y12,Y13

Vload   vr0,   r1,   $3           // load bias: R11
VMAC      /,   vr0,    /, $1000b  // mac init, set bias
Vload   vr0,   r2,   $8           // load weights: Q04
Vload   vr1,   r2,  $10           // load weights: Q05
Vload   vr2,   r2,  $12           // load weights: Q06
Vload   vr3,   r2,  $14           // load weights: Q07
Vload   vr4,   r3,   $2           // load input: P10
VMAC      /,   vr0,  vr4, $0000b
VMAC      /,   vr1,  vr4, $0001b
VMAC      /,   vr2,  vr4, $0010b
VMAC      /,   vr3,  vr4, $0011b
Vload   vr0,   r2,   $9           // load weights: Q14
Vload   vr1,   r2,  $11           // load weights: Q15
Vload   vr2,   r2,  $13           // load weights: Q16
Vload   vr3,   r2,  $15           // load weights: Q17
Vload   vr4,   r3,   $3           // load input: P11
VMAC      /,   vr0,  vr4, $0000b
VMAC      /,   vr1,  vr4, $0001b
VMAC      /,   vr2,  vr4, $0010b
VMAC    vr7,   vr3,  vr4, $0111b
Vstore   $3,   r4,   vr7          // store output:Y14,Y15,Y16,Y17

Vload   vr0,   r1,   $4           // load bias: R20
VMAC      /,   vr0,    /, $1000b  // mac init, set bias
Vload   vr0,   r2,   $0           // load weights: Q00
Vload   vr1,   r2,   $2           // load weights: Q01
Vload   vr2,   r2,   $4           // load weights: Q02
Vload   vr3,   r2,   $6           // load weights: Q03
Vload   vr4,   r3,   $4           // load input: P20
VMAC      /,   vr0,  vr4, $0000b
VMAC      /,   vr1,  vr4, $0001b
VMAC      /,   vr2,  vr4, $0010b
VMAC      /,   vr3,  vr4, $0011b
Vload   vr0,   r2,   $1           // load weights: Q10
Vload   vr1,   r2,   $3           // load weights: Q11
Vload   vr2,   r2,   $5           // load weights: Q12
Vload   vr3,   r2,   $7           // load weights: Q13
Vload   vr4,   r3,   $5           // load input: P21
VMAC      /,   vr0,  vr4, $0000b
VMAC      /,   vr1,  vr4, $0001b
VMAC      /,   vr2,  vr4, $0010b
VMAC    vr7,   vr3,  vr4, $0111b
Vstore   $4,   r4,   vr7          // store output:Y20,Y21,Y22,Y23

Vload   vr0,   r1,   $5           // load bias: R21
VMAC      /,   vr0,    /, $1000b  // mac init, set bias
Vload   vr0,   r2,   $8           // load weights: Q04
Vload   vr1,   r2,  $10           // load weights: Q05
Vload   vr2,   r2,  $12           // load weights: Q06
Vload   vr3,   r2,  $14           // load weights: Q07
Vload   vr4,   r3,   $4           // load input: P20
VMAC      /,   vr0,  vr4, $0000b
VMAC      /,   vr1,  vr4, $0001b
VMAC      /,   vr2,  vr4, $0010b
VMAC      /,   vr3,   vr4,  $0011b
Vload   vr0,    r2,    $9           // load weights: Q14
Vload   vr1,    r2,   $11           // load weights: Q15
Vload   vr2,    r2,   $13           // load weights: Q16
Vload   vr3,    r2,   $15           // load weights: Q17
Vload   vr4,    r3,    $5           // load input: P21
VMAC      /,   vr0,   vr4,  $0000b
VMAC      /,   vr1,   vr4,  $0001b
VMAC      /,   vr2,   vr4,  $0010b
VMAC    vr7,   vr3,   vr4,  $0111b
Vstore   $5,    r4,   vr7           // store output:Y24,Y25,Y26,Y27

Vload   vr0,    r1,    $6           // load bias: R30
VMAC      /,   vr0,     /,  $1000b  // mac init, set bias
Vload   vr0,    r2,    $0           // load weights: Q00
Vload   vr1,    r2,    $2           // load weights: Q01
Vload   vr2,    r2,    $4           // load weights: Q02
Vload   vr3,    r2,    $6           // load weights: Q03
Vload   vr4,    r3,    $6           // load input: P30
VMAC      /,   vr0,   vr4,  $0000b
VMAC      /,   vr1,   vr4,  $0001b
VMAC      /,   vr2,   vr4,  $0010b
VMAC      /,   vr3,   vr4,  $0011b
Vload   vr0,    r2,    $1           // load weights: Q10
Vload   vr1,    r2,    $3           // load weights: Q11
Vload   vr2,    r2,    $5           // load weights: Q12
Vload   vr3,    r2,    $7           // load weights: Q13
Vload   vr4,    r3,    $7           // load input: P31
VMAC      /,   vr0,   vr4,  $0000b
VMAC      /,   vr1,   vr4,  $0001b
VMAC      /,   vr2,   vr4,  $0010b
VMAC    vr7,   vr3,   vr4,  $0111b
Vstore   $6,    r4,   vr7           // store output:Y30,Y31,Y32,Y33

Vload   vr0,    r1,    $7           // load bias: R31
VMAC      /,   vr0,     /,  $1000b  // mac init, set bias
Vload   vr0,    r2,    $8           // load weights: Q04
Vload   vr1,    r2,   $10           // load weights: Q05
Vload   vr2,    r2,   $12           // load weights: Q06
Vload   vr3,    r2,   $14           // load weights: Q07
Vload   vr4,    r3,    $6           // load input: P30
VMAC      /,   vr0,   vr4,  $0000b
VMAC      /,   vr1,   vr4,  $0001b
VMAC      /,   vr2,   vr4,  $0010b
VMAC      /,   vr3,   vr4,  $0011b
Vload   vr0,    r2,    $9           // load weights: Q14
Vload   vr1,    r2,   $11           // load weights: Q15
Vload   vr2,    r2,   $13           // load weights: Q16
Vload   vr3,    r2,   $15           // load weights: Q17
Vload   vr4,    r3,    $7           // load input: P31
VMAC      /,   vr0,   vr4,  $0000b
VMAC      /,   vr1,   vr4,  $0001b
VMAC      /,   vr2,   vr4,  $0010b
VMAC    vr7,   vr3,   vr4,  $0111b
Vstore   $7,    r4,   vr7           // store output:Y34,Y35,Y36,Y37

Vload   vr0,    r1,    $8           // load bias: R40
VMAC      /,   vr0,     /,  $1000b  // mac init, set bias
Vload   vr0,    r2,    $0           // load weights: Q00
Vload   vr1,    r2,    $2           // load weights: Q01
Vload   vr2,    r2,    $4           // load weights: Q02
Vload   vr3,    r2,    $6           // load weights: Q03
Vload   vr4,    r3,    $8           // load input: P40
```

```
VMAC      /,   vr0,   vr4,  $0000b
VMAC      /,   vr1,   vr4,  $0001b
VMAC      /,   vr2,   vr4,  $0010b
VMAC      /,   vr3,   vr4,  $0011b
Vload   vr0,    r2,   $1              // load weights: Q10
Vload   vr1,    r2,   $3              // load weights: Q11
Vload   vr2,    r2,   $5              // load weights: Q12
Vload   vr3,    r2,   $7              // load weights: Q13
Vload   vr4,    r3,   $9              // load input: P41
VMAC      /,   vr0,   vr4,  $0000b
VMAC      /,   vr1,   vr4,  $0001b
VMAC      /,   vr2,   vr4,  $0010b
VMAC    vr7,   vr3,   vr4,  $0111b
Vstore   $8,    r4,   vr7             // store output:Y40,Y41,Y42,Y43

Vload   vr0,    r1,   $9              // load bias: R41
VMAC      /,   vr0,     /,  $1000b    // mac init, set bias
Vload   vr0,    r2,   $8              // load weights: Q04
Vload   vr1,    r2,  $10              // load weights: Q05
Vload   vr2,    r2,  $12              // load weights: Q06
Vload   vr3,    r2,  $14              // load weights: Q07
Vload   vr4,    r3,   $8              // load input: P40
VMAC      /,   vr0,   vr4,  $0000b
VMAC      /,   vr1,   vr4,  $0001b
VMAC      /,   vr2,   vr4,  $0010b
VMAC      /,   vr3,   vr4,  $0011b
Vload   vr0,    r2,   $9              // load weights: Q14
Vload   vr1,    r2,  $11              // load weights: Q15
Vload   vr2,    r2,  $13              // load weights: Q16
Vload   vr3,    r2,  $15              // load weights: Q17
Vload   vr4,    r3,   $9              // load input: P41
VMAC      /,   vr0,   vr4,  $0000b
VMAC      /,   vr1,   vr4,  $0001b
VMAC      /,   vr2,   vr4,  $0010b
VMAC    vr7,   vr3,   vr4,  $0111b
Vstore   $9,    r4,   vr7             // store output:Y44,Y45,Y46,Y47

Vload   vr0,    r1,  $10              // load bias: R50
VMAC      /,   vr0,     /,  $1000b    // mac init, set bias
Vload   vr0,    r2,   $0              // load weights: Q00
Vload   vr1,    r2,   $2              // load weights: Q01
Vload   vr2,    r2,   $4              // load weights: Q02
Vload   vr3,    r2,   $6              // load weights: Q03
Vload   vr4,    r3,  $10              // load input: P50
VMAC      /,   vr0,   vr4,  $0000b
VMAC      /,   vr1,   vr4,  $0001b
VMAC      /,   vr2,   vr4,  $0010b
VMAC      /,   vr3,   vr4,  $0011b
Vload   vr0,    r2,   $1              // load weights: Q10
Vload   vr1,    r2,   $3              // load weights: Q11
Vload   vr2,    r2,   $5              // load weights: Q12
Vload   vr3,    r2,   $7              // load weights: Q13
Vload   vr4,    r3,  $11              // load input: P51
VMAC      /,   vr0,   vr4,  $0000b
VMAC      /,   vr1,   vr4,  $0001b
VMAC      /,   vr2,   vr4,  $0010b
VMAC    vr7,   vr3,   vr4,  $0111b
Vstore  $10,    r4,   vr7             // store output:Y50,Y51,Y52,Y53

Vload   vr0,    r1,  $11              // load bias: R51
VMAC      /,   vr0,     /,  $1000b    // mac init, set bias
Vload   vr0,    r2,   $8              // load weights: Q04
Vload   vr1,    r2,  $10              // load weights: Q05
Vload   vr2,    r2,  $12            // load weights: Q06
Vload   vr3,    r2,  $14            // load weights: Q07
Vload   vr4,    r3,  $10            // load input: P50
VMAC      /,   vr0,  vr4, $0000b
VMAC      /,   vr1,  vr4, $0001b
VMAC      /,   vr2,  vr4, $0010b
VMAC      /,   vr3,  vr4, $0011b
Vload   vr0,    r2,   $9            // load weights: Q14
Vload   vr1,    r2,  $11            // load weights: Q15
Vload   vr2,    r2,  $13            // load weights: Q16
Vload   vr3,    r2,  $15            // load weights: Q17
Vload   vr4,    r3,  $11            // load input: P51
VMAC      /,   vr0,  vr4, $0000b
VMAC      /,   vr1,  vr4, $0001b
VMAC      /,   vr2,  vr4, $0010b
VMAC    vr7,   vr3,  vr4, $0111b
Vstore  $11,    r4,  vr7            // store output:Y54,Y55,Y56,Y57

Vload   vr0,    r1,  $12            // load bias: R60
VMAC      /,   vr0,    /, $1000b    // mac init, set bias
Vload   vr0,    r2,   $0            // load weights: Q00
Vload   vr1,    r2,   $2            // load weights: Q01
Vload   vr2,    r2,   $4            // load weights: Q02
Vload   vr3,    r2,   $6            // load weights: Q03
Vload   vr4,    r3,  $12            // load input: P60
VMAC      /,   vr0,  vr4, $0000b
VMAC      /,   vr1,  vr4, $0001b
VMAC      /,   vr2,  vr4, $0010b
VMAC      /,   vr3,  vr4, $0011b
Vload   vr0,    r2,   $1            // load weights: Q10
Vload   vr1,    r2,   $3            // load weights: Q11
Vload   vr2,    r2,   $5            // load weights: Q12
Vload   vr3,    r2,   $7            // load weights: Q13
Vload   vr4,    r3,  $13            // load input: P61
VMAC      /,   vr0,  vr4, $0000b
VMAC      /,   vr1,  vr4, $0001b
VMAC      /,   vr2,  vr4, $0010b
VMAC    vr7,   vr3,  vr4, $0111b
Vstore  $12,    r4,  vr7            // store output:Y60,Y61,Y62,Y63

Vload   vr0,    r1,  $13            // load bias: R61
VMAC      /,   vr0,    /, $1000b    // mac init, set bias
Vload   vr0,    r2,   $8            // load weights: Q04
Vload   vr1,    r2,  $10            // load weights: Q05
Vload   vr2,    r2,  $12            // load weights: Q06
Vload   vr3,    r2,  $14            // load weights: Q07
Vload   vr4,    r3,  $12            // load input: P60
VMAC      /,   vr0,  vr4, $0000b
VMAC      /,   vr1,  vr4, $0001b
VMAC      /,   vr2,  vr4, $0010b
VMAC      /,   vr3,  vr4, $0011b
Vload   vr0,    r2,   $9            // load weights: Q14
Vload   vr1,    r2,  $11            // load weights: Q15
Vload   vr2,    r2,  $13            // load weights: Q16
Vload   vr3,    r2,  $15            // load weights: Q17
Vload   vr4,    r3,  $13            // load input: P61
VMAC      /,   vr0,  vr4, $0000b
VMAC      /,   vr1,  vr4, $0001b
VMAC      /,   vr2,  vr4, $0010b
VMAC    vr7,   vr3,  vr4, $0111b
Vstore  $13,    r4,  vr7            // store output:Y64,Y65,Y66,Y67

Vload   vr0,    r1,  $14            // load bias: R70
```

```
VMAC      /,   vr0,    /,  $1000b    // mac init, set bias
Vload   vr0,   r2,   $0             // load weights: Q00
Vload   vr1,   r2,   $2             // load weights: Q01
Vload   vr2,   r2,   $4             // load weights: Q02
Vload   vr3,   r2,   $6             // load weights: Q03
Vload   vr4,   r3,  $14             // load input: P70
VMAC      /,   vr0,  vr4,  $0000b
VMAC      /,   vr1,  vr4,  $0001b
VMAC      /,   vr2,  vr4,  $0010b
VMAC      /,   vr3,  vr4,  $0011b
Vload   vr0,   r2,   $1             // load weights: Q10
Vload   vr1,   r2,   $3             // load weights: Q11
Vload   vr2,   r2,   $5             // load weights: Q12
Vload   vr3,   r2,   $7             // load weights: Q13
Vload   vr4,   r3,  $15             // load input: P71
VMAC      /,   vr0,  vr4,  $0000b
VMAC      /,   vr1,  vr4,  $0001b
VMAC      /,   vr2,  vr4,  $0010b
VMAC    vr7,   vr3,  vr4,  $0111b
Vstore  $14,   r4,  vr7             // store output:Y70,Y71,Y72,Y73

Vload   vr0,   r1,  $15             // load bias: R71
VMAC      /,   vr0,    /,  $1000b    // mac init, set bias
Vload   vr0,   r2,   $8             // load weights: Q04
Vload   vr1,   r2,  $10             // load weights: Q05
Vload   vr2,   r2,  $12             // load weights: Q06
Vload   vr3,   r2,  $14             // load weights: Q07
Vload   vr4,   r3,  $14             // load input: P70
VMAC      /,   vr0,  vr4,  $0000b
VMAC      /,   vr1,  vr4,  $0001b
VMAC      /,   vr2,  vr4,  $0010b
VMAC      /,   vr3,  vr4,  $0011b
Vload   vr0,   r2,   $9             // load weights: Q14
Vload   vr1,   r2,  $11             // load weights: Q15
Vload   vr2,   r2,  $13             // load weights: Q16
Vload   vr3,   r2,  $15             // load weights: Q17
Vload   vr4,   r3,  $15             // load input: P71
VMAC      /,   vr0,  vr4,  $0000b
VMAC      /,   vr1,  vr4,  $0001b
VMAC      /,   vr2,  vr4,  $0010b
VMAC    vr7,   vr3,  vr4,  $0111b
Vstore  $15,   r4,  vr7             // store output:Y74,Y75,Y76,Y77
```

(b) Propose a superscalar strategy (maximum 2 instruction per fetch), and calculate how many cycles needed. Compare the utilization ratio with and without the superscalar strategy.

Superscalar strategy:

The hardware architecture is shown in Fig.3, and the super scalar strategy is shown in Fig.4. IF is triggered by double edges when 2 instruction per fetch otherwise positive edge. Only when fetching a VMAC(without output) instruction at the clock's positive edge, it will fetch the next Vload instruction at the clock's negative edge, otherwise fetch the next instruction at the next positive edge.

It is worth mentioning that the WB phase is on the same clock of instruction VMAC and the very next Vload, but it will not cause conflict because the WB phase won't works in instruction VMAC(no output). While VMAC is outputing, it is a single instruction excuting.

Cycles and utilization ratio:

Originally, each assembly code vmac computing procedure(e.g. line 6˜26, 28˜48, ..., which compute continous 4 elements(row) in output matrix Y) takes 21 instructions which takes 23 cycles, with 7 VMAC cycles in it. We need 16 this procedures to output the whole 8x8 matrix Y, adding 4 instructions that MOV the starting address of $\mathbf{A}, \mathbf{B}, \mathbf{B}, \mathbf{Y}$, it uses $4 + 21 \times 16 = 340$ instructions. As all instructions can be executed sequentially,

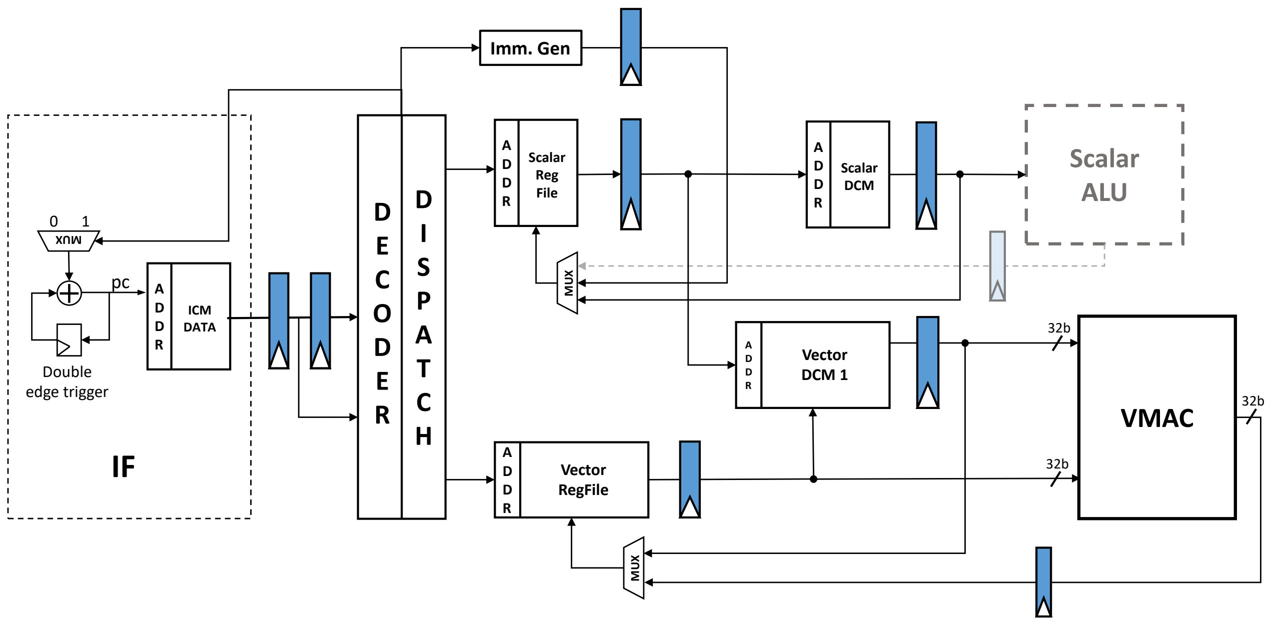
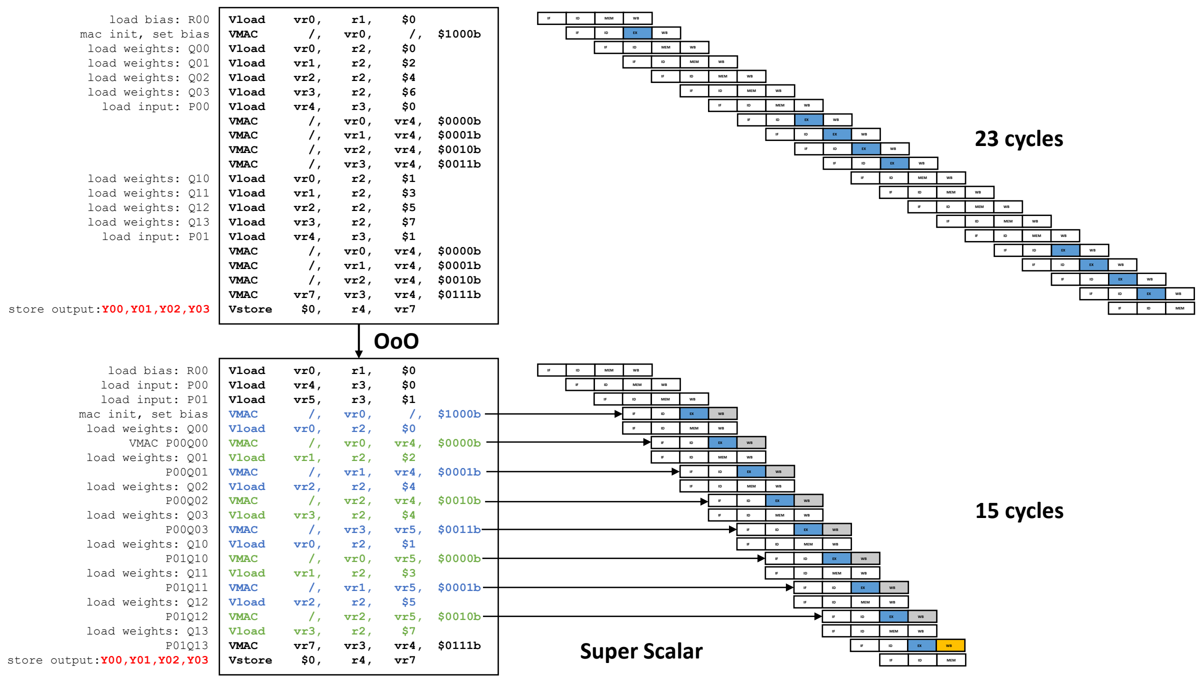


Figure 3: Hardware architecture



```
         load bias: R00  Vload    vr0,    r1,    $0
   mac init, set bias  VMAC       /,    vr0,     /,  $1000b
   load weights: Q00  Vload    vr0,    r2,    $0
   load weights: Q01  Vload    vr1,    r2,    $2
   load weights: Q02  Vload    vr2,    r2,    $4
   load weights: Q03  Vload    vr3,    r2,    $6
      load input: P00  Vload    vr4,    r3,    $0
                       VMAC       /,    vr0,   vr4,  $0000b
                       VMAC       /,    vr1,   vr4,  $0001b
                       VMAC       /,    vr2,   vr4,  $0010b
                       VMAC       /,    vr3,   vr4,  $0011b
   load weights: Q10  Vload    vr0,    r2,    $1
   load weights: Q11  Vload    vr1,    r2,    $3
   load weights: Q12  Vload    vr2,    r2,    $5
   load weights: Q13  Vload    vr3,    r2,    $7
      load input: P01  Vload    vr4,    r3,    $1
                       VMAC       /,    vr0,   vr4,  $0000b
                       VMAC       /,    vr1,   vr4,  $0001b
                       VMAC       /,    vr2,   vr4,  $0010b
                       VMAC     vr7,    vr3,   vr4,  $0111b
store output:Y00,Y01,Y02,Y03  Vstore    $0,    r4,   vr7
```

```
         load bias: R00  Vload    vr0,    r1,    $0
      load input: P00  Vload    vr4,    r3,    $0
      load input: P01  Vload    vr5,    r3,    $1
   mac init, set bias  VMAC       /,    vr0,     /,  $1000b
   load weights: Q00  Vload    vr0,    r2,    $0
          VMAC P00Q00  VMAC       /,    vr0,   vr4,  $0000b
   load weights: Q01  Vload    vr1,    r2,    $2
               P00Q01  VMAC       /,    vr1,   vr4,  $0001b
   load weights: Q02  Vload    vr2,    r2,    $4
               P00Q02  VMAC       /,    vr2,   vr4,  $0010b
   load weights: Q03  Vload    vr3,    r2,    $4
               P00Q03  VMAC       /,    vr3,   vr5,  $0011b
   load weights: Q10  Vload    vr0,    r2,    $1
               P01Q10  VMAC       /,    vr0,   vr5,  $0000b
   load weights: Q11  Vload    vr1,    r2,    $3
               P01Q11  VMAC       /,    vr1,   vr5,  $0001b
   load weights: Q12  Vload    vr2,    r2,    $5
               P01Q12  VMAC       /,    vr2,   vr5,  $0010b
   load weights: Q13  Vload    vr3,    r2,    $7
               P01Q13  VMAC     vr7,    vr3,   vr4,  $0111b
store output:Y00,Y01,Y02,Y03  Vstore    $0,    r4,   vr7
```

Figure 4: Superscalar strategy(ZoomIn to get details)

cycles that it takes C_{org} and utilization ratio U_{org} are:

$$C_{org} = 4 + 21 \times 16 + 2 = 342 \ cycles$$

$$U_{org} = (9 \times 16)/C_{org} = 0.421$$

With the application of **superscalar**, it will reduce 8 cycles per precedure. So the cycles that it takes C_{ss} and utilization ratio U_{ss} are:

$$C_{ss} = 4 + (21 - 8) \times 16 + 2 = 214 \ cycles$$

$$U_{ss} = (9 \times 16)/C_{org} = 0.673$$

The utilization went up by $(\mathbf{0.673} - \mathbf{0.421})/\mathbf{0.421} = \mathbf{59.6}\%$ after the application of superscalar stratege.