

Homework Assignment #2

Instructor: Chixiao Chen

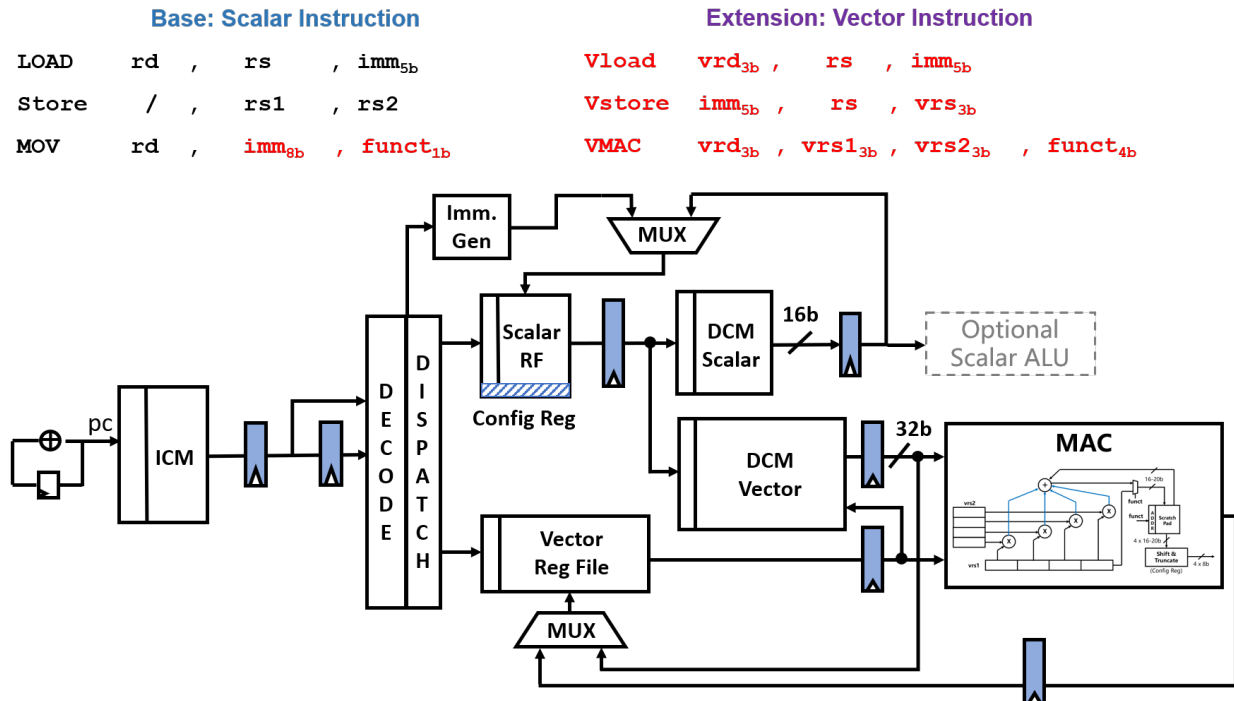
Name: Chunyu Wang, FudanID: 20210860017

- 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 L^AT_EX 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.



Implementatino of a VMAC module:

```

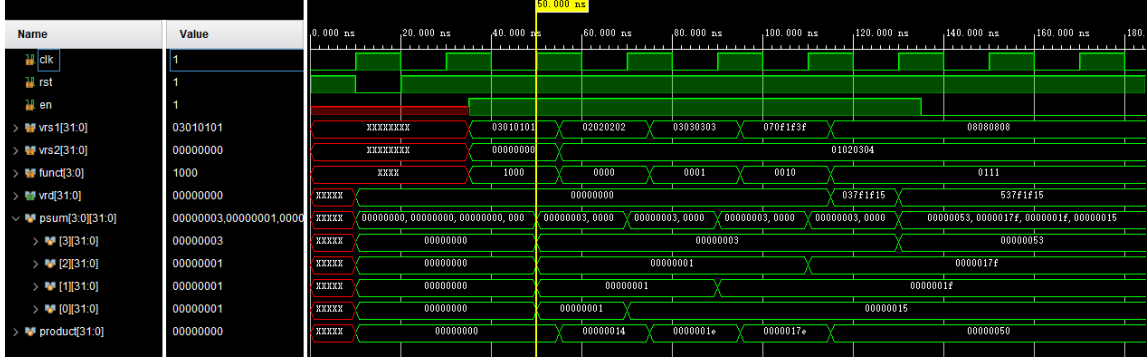
1 //timescale 1ns / 1ps
2 ////////////////////////////////////////
3 // Author: Wang Chunyu
4 // Description:
5 // Additional Comments:
6 ////////////////////////////////////////
7 module VMAC(
8     input          clk,rst,en,
9     input [31:0]   vrs1,
10    input [31:0]   vrs2,
11    input [3:0]    funct,
12
13    output reg [31:0] vrd
14 );
15 reg [31:0] psum[3:0];
16 reg [31:0] product;
17 integer i;
18
19 always @(*) begin
20     if (!funct[3]) begin //mac, do not set bias
21         product=0;
22         for(i=0;i<4;i=i+1)
23             product=product+$signed(vrs1[i*8+:8])*$signed(vrs2[i*8+:8]);
24
25         //output, shift and truncate
26         if(funct[2]==1'b1) begin
27             for(i=0;i<4;i=i+1) begin
28                 // shift 0, truncate, the lower 8 bit as output. overflow handling, remain sign bit
29                 if((~psum[i][31]) && (psum[i]>31'b00000000000000000000000000000000))
30                     vrd[i*8+:8]=8'b01111111;
31                 else if((psum[i][31]) && (psum[i]<31'b11111111111111111111111111111111))
32                     vrd[i*8+:8]=8'b10000000;
33                 else
34                     vrd[i*8+:8]=psum[i][7:0];
35             end
36         end
37         else begin
38             vrd=0;
39         end
40     end
41     else begin
42         product=0;
43         vrd=0;
44     end
45 end
46 // write psum (scratch pad)
47 always@(posedge clk or negedge rst) begin
48     if(!rst) begin
49         for(i=0;i<4;i=i+1)
50             psum[i]<=0;
51     end
52     else if(en) begin
53         if(funct[3]==1'b1) // set bias
54             for(i=0;i<4;i=i+1)
55                 psum[i]<=$signed(vrs1[i*8+:8]);
56         else begin// 4 mac
57             psum[funct[1:0]]<=psum[funct[1:0]]+product;
58         end
59     end
60 end
61 endmodule

```

VMAC simulation:

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

(a)



(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**.

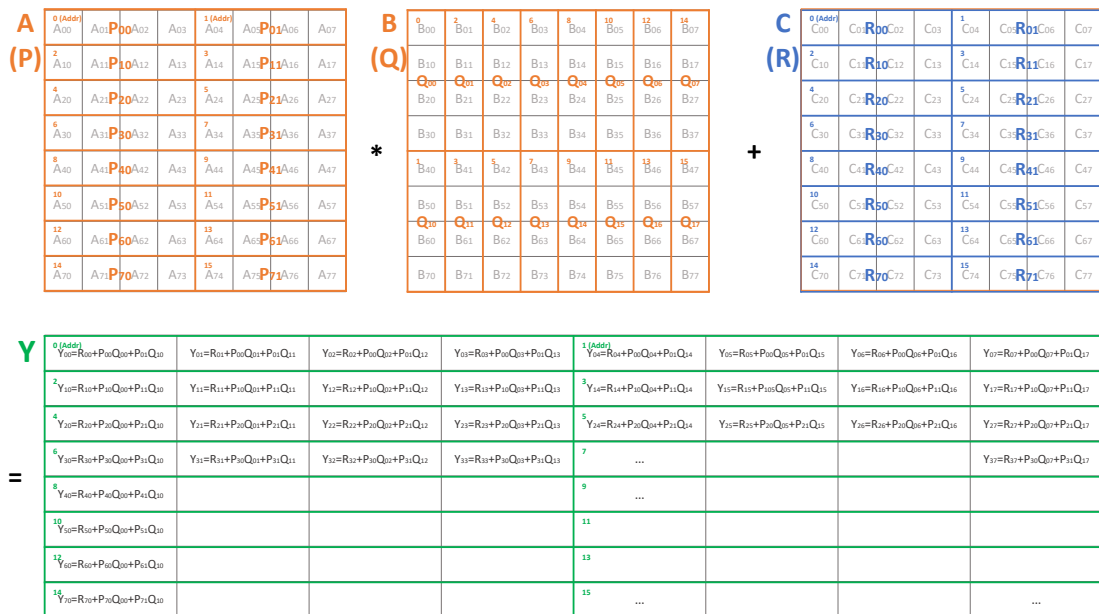


Figure 2: 8x8 matrix multiplication using a 4x4 VMAC (ZoomIn to get details)

Assembly Code:

```

1  MOV     r1,    $0, [AB],    $0    // address bias (C)R
2  MOV     r2,    $0, [AW],    $0    // address weights (B)Q
3  MOV     r3,    $0, [AX],    $0    // address input X (A)P
4  MOV     r4,    $0, [AY],    $0    // address output Y = AB+C
5
6  Vload   vr0,   r1,    $0          // load bias: R00
7  VMAC    /,    vr0,   /, $1000b    // mac init, set bias
8  Vload   vr0,   r2,    $0          // load weights: Q00
9  Vload   vr1,   r2,    $2          // load weights: Q01
10 Vload   vr2,   r2,    $4          // load weights: Q02
11 Vload   vr3,   r2,    $6          // load weights: Q03
12 Vload   vr4,   r3,    $0          // load input: P00
13 VMAC    /,    vr0,   vr4, $0000b
14 VMAC    /,    vr1,   vr4, $0001b
15 VMAC    /,    vr2,   vr4, $0010b
16 VMAC    /,    vr3,   vr4, $0011b
17 Vload   vr0,   r2,    $1          // load weights: Q10
18 Vload   vr1,   r2,    $3          // load weights: Q11
19 Vload   vr2,   r2,    $5          // load weights: Q12
20 Vload   vr3,   r2,    $7          // load weights: Q13
21 Vload   vr4,   r3,    $1          // load input: P01
22 VMAC    /,    vr0,   vr4, $0000b
23 VMAC    /,    vr1,   vr4, $0001b
24 VMAC    /,    vr2,   vr4, $0010b
25 VMAC    vr7,   vr3,   vr4, $0111b
26 Vstore  $0,    r4,    vr7        // store output:Y00,Y01,Y02,Y03
27
28 Vload   vr0,   r1,    $1          // load bias: R01
29 VMAC    /,    vr0,   /, $1000b    // mac init, set bias
30 Vload   vr0,   r2,    $8          // load weights: Q04
31 Vload   vr1,   r2,    $10         // load weights: Q05
32 Vload   vr2,   r2,    $12         // load weights: Q06
33 Vload   vr3,   r2,    $14         // load weights: Q07
34 Vload   vr4,   r3,    $0          // load input: P00
35 VMAC    /,    vr0,   vr4, $0000b
36 VMAC    /,    vr1,   vr4, $0001b
37 VMAC    /,    vr2,   vr4, $0010b
38 VMAC    /,    vr3,   vr4, $0011b
39 Vload   vr0,   r2,    $9          // load weights: Q14
40 Vload   vr1,   r2,    $11         // load weights: Q15
41 Vload   vr2,   r2,    $13         // load weights: Q16
42 Vload   vr3,   r2,    $15         // load weights: Q17
43 Vload   vr4,   r3,    $1          // load input: P01
44 VMAC    /,    vr0,   vr4, $0000b
45 VMAC    /,    vr1,   vr4, $0001b
46 VMAC    /,    vr2,   vr4, $0010b
47 VMAC    vr7,   vr3,   vr4, $0111b
48 Vstore  $1,    r4,    vr7        // store output:Y04,Y05,Y06,Y07
49
50 Vload   vr0,   r1,    $2          // load bias: R10
51 VMAC    /,    vr0,   /, $1000b    // mac init, set bias
52 Vload   vr0,   r2,    $0          // load weights: Q00
53 Vload   vr1,   r2,    $2          // load weights: Q01
54 Vload   vr2,   r2,    $4          // load weights: Q02
55 Vload   vr3,   r2,    $6          // load weights: Q03
56 Vload   vr4,   r3,    $2          // load input: P10
57 VMAC    /,    vr0,   vr4, $0000b
58 VMAC    /,    vr1,   vr4, $0001b
59 VMAC    /,    vr2,   vr4, $0010b
60 VMAC    /,    vr3,   vr4, $0011b
61 Vload   vr0,   r2,    $1          // load weights: Q10
62 Vload   vr1,   r2,    $3          // load weights: Q11

```

```

63 Vload  vr2,  r2,  $5          // load weights: Q12
64 Vload  vr3,  r2,  $7          // load weights: Q13
65 Vload  vr4,  r3,  $3          // load input: P11
66 VMAC    /,  vr0,  vr4,  $0000b
67 VMAC    /,  vr1,  vr4,  $0001b
68 VMAC    /,  vr2,  vr4,  $0010b
69 VMAC  vr7,  vr3,  vr4,  $0111b
70 Vstore  $2,  r4,  vr7        // store output:Y10,Y11,Y12,Y13
71
72 Vload  vr0,  r1,  $3          // load bias: R11
73 VMAC    /,  vr0,  /,  $1000b  // mac init, set bias
74 Vload  vr0,  r2,  $8          // load weights: Q04
75 Vload  vr1,  r2,  $10         // load weights: Q05
76 Vload  vr2,  r2,  $12         // load weights: Q06
77 Vload  vr3,  r2,  $14         // load weights: Q07
78 Vload  vr4,  r3,  $2          // load input: P10
79 VMAC    /,  vr0,  vr4,  $0000b
80 VMAC    /,  vr1,  vr4,  $0001b
81 VMAC    /,  vr2,  vr4,  $0010b
82 VMAC    /,  vr3,  vr4,  $0011b
83 Vload  vr0,  r2,  $9          // load weights: Q14
84 Vload  vr1,  r2,  $11         // load weights: Q15
85 Vload  vr2,  r2,  $13         // load weights: Q16
86 Vload  vr3,  r2,  $15         // load weights: Q17
87 Vload  vr4,  r3,  $3          // load input: P11
88 VMAC    /,  vr0,  vr4,  $0000b
89 VMAC    /,  vr1,  vr4,  $0001b
90 VMAC    /,  vr2,  vr4,  $0010b
91 VMAC  vr7,  vr3,  vr4,  $0111b
92 Vstore  $3,  r4,  vr7        // store output:Y14,Y15,Y16,Y17
93
94 Vload  vr0,  r1,  $4          // load bias: R20
95 VMAC    /,  vr0,  /,  $1000b  // mac init, set bias
96 Vload  vr0,  r2,  $0          // load weights: Q00
97 Vload  vr1,  r2,  $2          // load weights: Q01
98 Vload  vr2,  r2,  $4          // load weights: Q02
99 Vload  vr3,  r2,  $6          // load weights: Q03
100 Vload  vr4,  r3,  $4          // load input: P20
101 VMAC    /,  vr0,  vr4,  $0000b
102 VMAC    /,  vr1,  vr4,  $0001b
103 VMAC    /,  vr2,  vr4,  $0010b
104 VMAC    /,  vr3,  vr4,  $0011b
105 Vload  vr0,  r2,  $1          // load weights: Q10
106 Vload  vr1,  r2,  $3          // load weights: Q11
107 Vload  vr2,  r2,  $5          // load weights: Q12
108 Vload  vr3,  r2,  $7          // load weights: Q13
109 Vload  vr4,  r3,  $5          // load input: P21
110 VMAC    /,  vr0,  vr4,  $0000b
111 VMAC    /,  vr1,  vr4,  $0001b
112 VMAC    /,  vr2,  vr4,  $0010b
113 VMAC  vr7,  vr3,  vr4,  $0111b
114 Vstore  $4,  r4,  vr7        // store output:Y20,Y21,Y22,Y23
115
116 Vload  vr0,  r1,  $5          // load bias: R21
117 VMAC    /,  vr0,  /,  $1000b  // mac init, set bias
118 Vload  vr0,  r2,  $8          // load weights: Q04
119 Vload  vr1,  r2,  $10         // load weights: Q05
120 Vload  vr2,  r2,  $12         // load weights: Q06
121 Vload  vr3,  r2,  $14         // load weights: Q07
122 Vload  vr4,  r3,  $4          // load input: P20
123 VMAC    /,  vr0,  vr4,  $0000b
124 VMAC    /,  vr1,  vr4,  $0001b
125 VMAC    /,  vr2,  vr4,  $0010b

```

```

126 VMAC      /,   vr3,   vr4, $0011b
127 Vload    vr0,   r2,   $9      // load weights: Q14
128 Vload    vr1,   r2,   $11     // load weights: Q15
129 Vload    vr2,   r2,   $13     // load weights: Q16
130 Vload    vr3,   r2,   $15     // load weights: Q17
131 Vload    vr4,   r3,   $5      // load input: P21
132 VMAC      /,   vr0,   vr4, $0000b
133 VMAC      /,   vr1,   vr4, $0001b
134 VMAC      /,   vr2,   vr4, $0010b
135 VMAC    vr7,   vr3,   vr4, $0111b
136 Vstore   $5,   r4,   vr7      // store output:Y24,Y25,Y26,Y27
137
138 Vload    vr0,   r1,   $6      // load bias: R30
139 VMAC      /,   vr0,   /,   $1000b // mac init, set bias
140 Vload    vr0,   r2,   $0      // load weights: Q00
141 Vload    vr1,   r2,   $2      // load weights: Q01
142 Vload    vr2,   r2,   $4      // load weights: Q02
143 Vload    vr3,   r2,   $6      // load weights: Q03
144 Vload    vr4,   r3,   $6      // load input: P30
145 VMAC      /,   vr0,   vr4, $0000b
146 VMAC      /,   vr1,   vr4, $0001b
147 VMAC      /,   vr2,   vr4, $0010b
148 VMAC      /,   vr3,   vr4, $0011b
149 Vload    vr0,   r2,   $1      // load weights: Q10
150 Vload    vr1,   r2,   $3      // load weights: Q11
151 Vload    vr2,   r2,   $5      // load weights: Q12
152 Vload    vr3,   r2,   $7      // load weights: Q13
153 Vload    vr4,   r3,   $7      // load input: P31
154 VMAC      /,   vr0,   vr4, $0000b
155 VMAC      /,   vr1,   vr4, $0001b
156 VMAC      /,   vr2,   vr4, $0010b
157 VMAC    vr7,   vr3,   vr4, $0111b
158 Vstore   $6,   r4,   vr7      // store output:Y30,Y31,Y32,Y33
159
160 Vload    vr0,   r1,   $7      // load bias: R31
161 VMAC      /,   vr0,   /,   $1000b // mac init, set bias
162 Vload    vr0,   r2,   $8      // load weights: Q04
163 Vload    vr1,   r2,   $10     // load weights: Q05
164 Vload    vr2,   r2,   $12     // load weights: Q06
165 Vload    vr3,   r2,   $14     // load weights: Q07
166 Vload    vr4,   r3,   $6      // load input: P30
167 VMAC      /,   vr0,   vr4, $0000b
168 VMAC      /,   vr1,   vr4, $0001b
169 VMAC      /,   vr2,   vr4, $0010b
170 VMAC      /,   vr3,   vr4, $0011b
171 Vload    vr0,   r2,   $9      // load weights: Q14
172 Vload    vr1,   r2,   $11     // load weights: Q15
173 Vload    vr2,   r2,   $13     // load weights: Q16
174 Vload    vr3,   r2,   $15     // load weights: Q17
175 Vload    vr4,   r3,   $7      // load input: P31
176 VMAC      /,   vr0,   vr4, $0000b
177 VMAC      /,   vr1,   vr4, $0001b
178 VMAC      /,   vr2,   vr4, $0010b
179 VMAC    vr7,   vr3,   vr4, $0111b
180 Vstore   $7,   r4,   vr7      // store output:Y34,Y35,Y36,Y37
181
182 Vload    vr0,   r1,   $8      // load bias: R40
183 VMAC      /,   vr0,   /,   $1000b // mac init, set bias
184 Vload    vr0,   r2,   $0      // load weights: Q00
185 Vload    vr1,   r2,   $2      // load weights: Q01
186 Vload    vr2,   r2,   $4      // load weights: Q02
187 Vload    vr3,   r2,   $6      // load weights: Q03
188 Vload    vr4,   r3,   $8      // load input: P40

```

```

189 VMAC      /,  vr0,  vr4, $0000b
190 VMAC      /,  vr1,  vr4, $0001b
191 VMAC      /,  vr2,  vr4, $0010b
192 VMAC      /,  vr3,  vr4, $0011b
193 Vload     vr0,  r2,   $1      // load weights: Q10
194 Vload     vr1,  r2,   $3      // load weights: Q11
195 Vload     vr2,  r2,   $5      // load weights: Q12
196 Vload     vr3,  r2,   $7      // load weights: Q13
197 Vload     vr4,  r3,   $9      // load input: P41
198 VMAC      /,  vr0,  vr4, $0000b
199 VMAC      /,  vr1,  vr4, $0001b
200 VMAC      /,  vr2,  vr4, $0010b
201 VMAC      vr7,  vr3,  vr4, $0111b
202 Vstore    $8,   r4,   vr7      // store output:Y40,Y41,Y42,Y43
203
204 Vload     vr0,  r1,   $9      // load bias: R41
205 VMAC      /,  vr0,  /,   $1000b // mac init, set bias
206 Vload     vr0,  r2,   $8      // load weights: Q04
207 Vload     vr1,  r2,  $10      // load weights: Q05
208 Vload     vr2,  r2,  $12      // load weights: Q06
209 Vload     vr3,  r2,  $14      // load weights: Q07
210 Vload     vr4,  r3,   $8      // load input: P40
211 VMAC      /,  vr0,  vr4, $0000b
212 VMAC      /,  vr1,  vr4, $0001b
213 VMAC      /,  vr2,  vr4, $0010b
214 VMAC      /,  vr3,  vr4, $0011b
215 Vload     vr0,  r2,   $9      // load weights: Q14
216 Vload     vr1,  r2,  $11      // load weights: Q15
217 Vload     vr2,  r2,  $13      // load weights: Q16
218 Vload     vr3,  r2,  $15      // load weights: Q17
219 Vload     vr4,  r3,   $9      // load input: P41
220 VMAC      /,  vr0,  vr4, $0000b
221 VMAC      /,  vr1,  vr4, $0001b
222 VMAC      /,  vr2,  vr4, $0010b
223 VMAC      vr7,  vr3,  vr4, $0111b
224 Vstore    $9,   r4,   vr7      // store output:Y44,Y45,Y46,Y47
225
226 Vload     vr0,  r1,  $10      // load bias: R50
227 VMAC      /,  vr0,  /,   $1000b // mac init, set bias
228 Vload     vr0,  r2,   $0      // load weights: Q00
229 Vload     vr1,  r2,   $2      // load weights: Q01
230 Vload     vr2,  r2,   $4      // load weights: Q02
231 Vload     vr3,  r2,   $6      // load weights: Q03
232 Vload     vr4,  r3,  $10      // load input: P50
233 VMAC      /,  vr0,  vr4, $0000b
234 VMAC      /,  vr1,  vr4, $0001b
235 VMAC      /,  vr2,  vr4, $0010b
236 VMAC      /,  vr3,  vr4, $0011b
237 Vload     vr0,  r2,   $1      // load weights: Q10
238 Vload     vr1,  r2,   $3      // load weights: Q11
239 Vload     vr2,  r2,   $5      // load weights: Q12
240 Vload     vr3,  r2,   $7      // load weights: Q13
241 Vload     vr4,  r3,  $11      // load input: P51
242 VMAC      /,  vr0,  vr4, $0000b
243 VMAC      /,  vr1,  vr4, $0001b
244 VMAC      /,  vr2,  vr4, $0010b
245 VMAC      vr7,  vr3,  vr4, $0111b
246 Vstore    $10,  r4,   vr7      // store output:Y50,Y51,Y52,Y53
247
248 Vload     vr0,  r1,  $11      // load bias: R51
249 VMAC      /,  vr0,  /,   $1000b // mac init, set bias
250 Vload     vr0,  r2,   $8      // load weights: Q04
251 Vload     vr1,  r2,  $10      // load weights: Q05

```

```

252 Vload vr2, r2, $12 // load weights: Q06
253 Vload vr3, r2, $14 // load weights: Q07
254 Vload vr4, r3, $10 // load input: P50
255 VMAC /, vr0, vr4, $0000b
256 VMAC /, vr1, vr4, $0001b
257 VMAC /, vr2, vr4, $0010b
258 VMAC /, vr3, vr4, $0011b
259 Vload vr0, r2, $9 // load weights: Q14
260 Vload vr1, r2, $11 // load weights: Q15
261 Vload vr2, r2, $13 // load weights: Q16
262 Vload vr3, r2, $15 // load weights: Q17
263 Vload vr4, r3, $11 // load input: P51
264 VMAC /, vr0, vr4, $0000b
265 VMAC /, vr1, vr4, $0001b
266 VMAC /, vr2, vr4, $0010b
267 VMAC vr7, vr3, vr4, $0111b
268 Vstore $11, r4, vr7 // store output:Y54,Y55,Y56,Y57
269
270 Vload vr0, r1, $12 // load bias: R60
271 VMAC /, vr0, /, $1000b // mac init, set bias
272 Vload vr0, r2, $0 // load weights: Q00
273 Vload vr1, r2, $2 // load weights: Q01
274 Vload vr2, r2, $4 // load weights: Q02
275 Vload vr3, r2, $6 // load weights: Q03
276 Vload vr4, r3, $12 // load input: P60
277 VMAC /, vr0, vr4, $0000b
278 VMAC /, vr1, vr4, $0001b
279 VMAC /, vr2, vr4, $0010b
280 VMAC /, vr3, vr4, $0011b
281 Vload vr0, r2, $1 // load weights: Q10
282 Vload vr1, r2, $3 // load weights: Q11
283 Vload vr2, r2, $5 // load weights: Q12
284 Vload vr3, r2, $7 // load weights: Q13
285 Vload vr4, r3, $13 // load input: P61
286 VMAC /, vr0, vr4, $0000b
287 VMAC /, vr1, vr4, $0001b
288 VMAC /, vr2, vr4, $0010b
289 VMAC vr7, vr3, vr4, $0111b
290 Vstore $12, r4, vr7 // store output:Y60,Y61,Y62,Y63
291
292 Vload vr0, r1, $13 // load bias: R61
293 VMAC /, vr0, /, $1000b // mac init, set bias
294 Vload vr0, r2, $8 // load weights: Q04
295 Vload vr1, r2, $10 // load weights: Q05
296 Vload vr2, r2, $12 // load weights: Q06
297 Vload vr3, r2, $14 // load weights: Q07
298 Vload vr4, r3, $12 // load input: P60
299 VMAC /, vr0, vr4, $0000b
300 VMAC /, vr1, vr4, $0001b
301 VMAC /, vr2, vr4, $0010b
302 VMAC /, vr3, vr4, $0011b
303 Vload vr0, r2, $9 // load weights: Q14
304 Vload vr1, r2, $11 // load weights: Q15
305 Vload vr2, r2, $13 // load weights: Q16
306 Vload vr3, r2, $15 // load weights: Q17
307 Vload vr4, r3, $13 // load input: P61
308 VMAC /, vr0, vr4, $0000b
309 VMAC /, vr1, vr4, $0001b
310 VMAC /, vr2, vr4, $0010b
311 VMAC vr7, vr3, vr4, $0111b
312 Vstore $13, r4, vr7 // store output:Y64,Y65,Y66,Y67
313
314 Vload vr0, r1, $14 // load bias: R70

```



```

315 VMAC    /,  vr0,    /,  $1000b    // mac init, set bias
316 Vload   vr0,  r2,    $0           // load weights: Q00
317 Vload   vr1,  r2,    $2           // load weights: Q01
318 Vload   vr2,  r2,    $4           // load weights: Q02
319 Vload   vr3,  r2,    $6           // load weights: Q03
320 Vload   vr4,  r3,   $14           // load input: P70
321 VMAC    /,  vr0,  vr4, $0000b
322 VMAC    /,  vr1,  vr4, $0001b
323 VMAC    /,  vr2,  vr4, $0010b
324 VMAC    /,  vr3,  vr4, $0011b
325 Vload   vr0,  r2,    $1           // load weights: Q10
326 Vload   vr1,  r2,    $3           // load weights: Q11
327 Vload   vr2,  r2,    $5           // load weights: Q12
328 Vload   vr3,  r2,    $7           // load weights: Q13
329 Vload   vr4,  r3,   $15           // load input: P71
330 VMAC    /,  vr0,  vr4, $0000b
331 VMAC    /,  vr1,  vr4, $0001b
332 VMAC    /,  vr2,  vr4, $0010b
333 VMAC    vr7,  vr3,  vr4, $0111b
334 Vstore  $14,  r4,  vr7           // store output:Y70,Y71,Y72,Y73
335
336 Vload   vr0,  r1,   $15           // load bias: R71
337 VMAC    /,  vr0,    /,  $1000b    // mac init, set bias
338 Vload   vr0,  r2,    $8           // load weights: Q04
339 Vload   vr1,  r2,   $10           // load weights: Q05
340 Vload   vr2,  r2,   $12           // load weights: Q06
341 Vload   vr3,  r2,   $14           // load weights: Q07
342 Vload   vr4,  r3,   $14           // load input: P70
343 VMAC    /,  vr0,  vr4, $0000b
344 VMAC    /,  vr1,  vr4, $0001b
345 VMAC    /,  vr2,  vr4, $0010b
346 VMAC    /,  vr3,  vr4, $0011b
347 Vload   vr0,  r2,    $9           // load weights: Q14
348 Vload   vr1,  r2,   $11           // load weights: Q15
349 Vload   vr2,  r2,   $13           // load weights: Q16
350 Vload   vr3,  r2,   $15           // load weights: Q17
351 Vload   vr4,  r3,   $15           // load input: P71
352 VMAC    /,  vr0,  vr4, $0000b
353 VMAC    /,  vr1,  vr4, $0001b
354 VMAC    /,  vr2,  vr4, $0010b
355 VMAC    vr7,  vr3,  vr4, $0111b
356 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 **A**, **B**, **B**, **Y**, it uses $4 + 21 \times 16 = 340$ instructions. As all instructions can be executed sequentially,

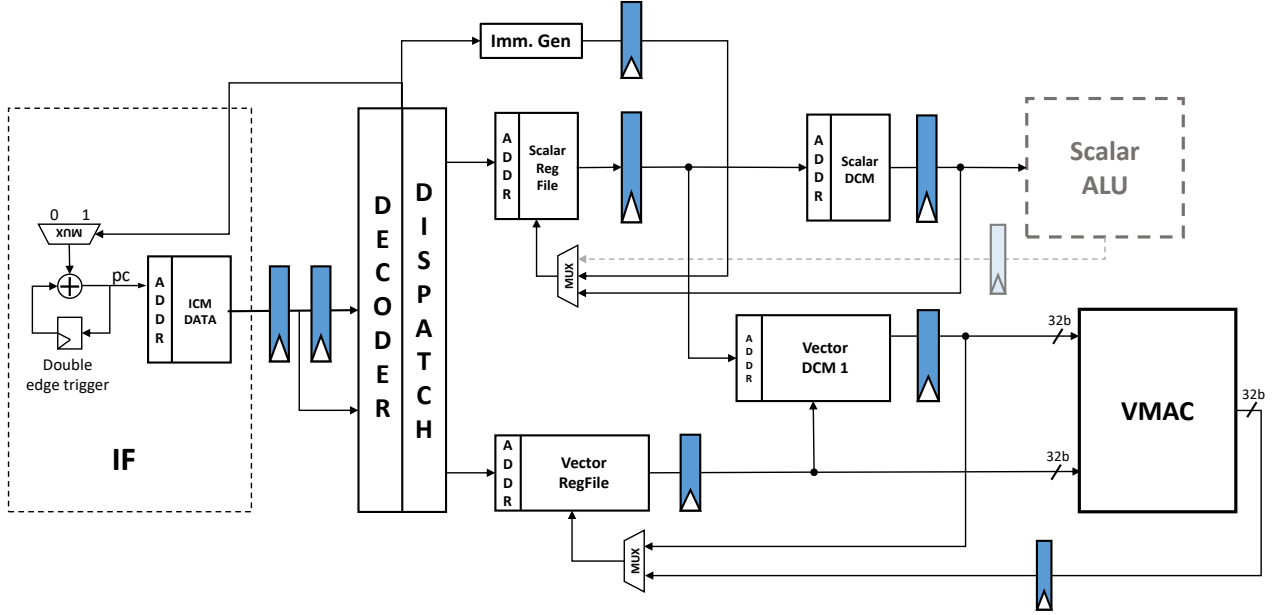


Figure 3: Hardware architecture

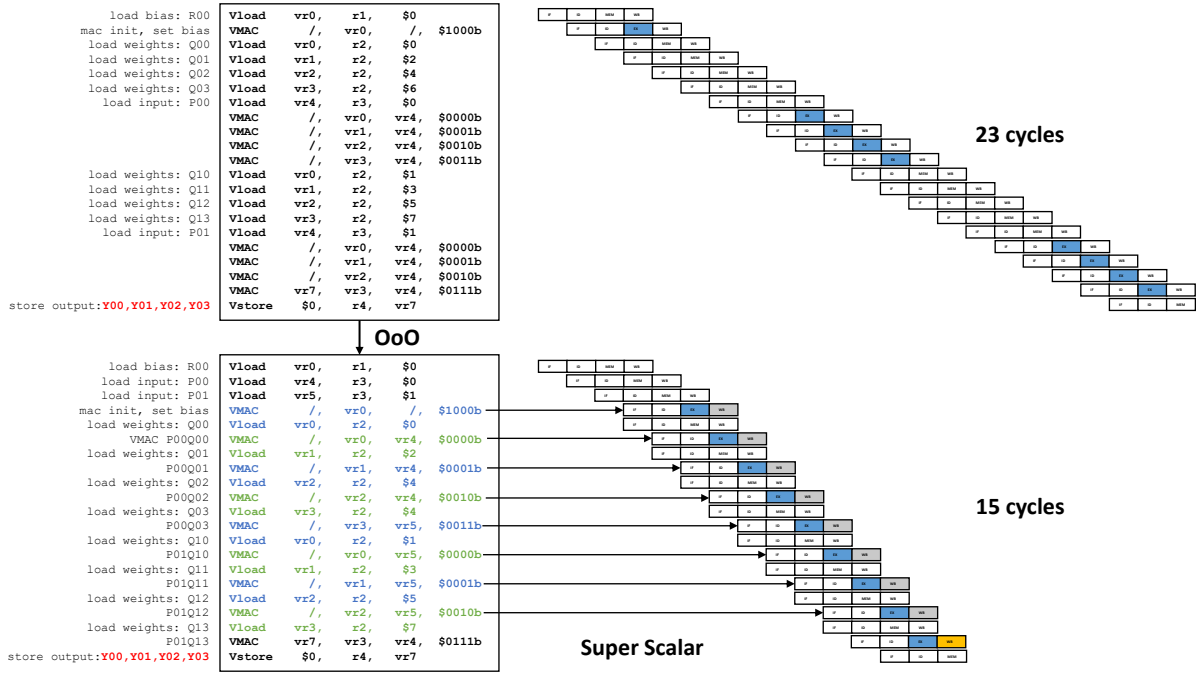


Figure 4: Superscalar strategy (ZoomIn to get details)

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

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

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

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

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

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

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