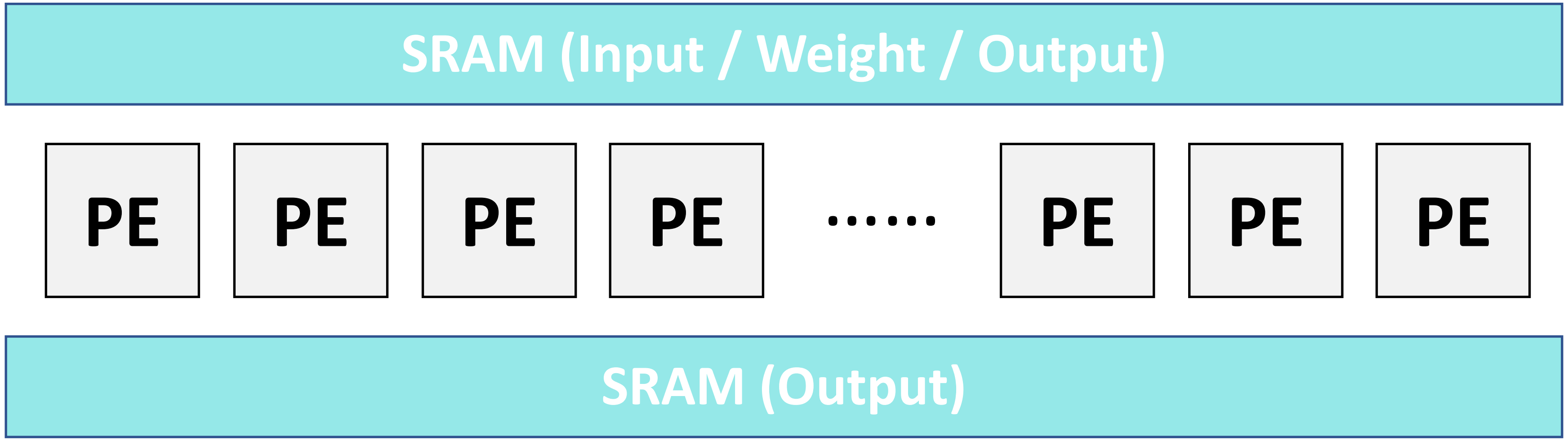
AI On Chip 2022
Assignment 4

MAC Implementation & Analyzation

PE(Processing Element)

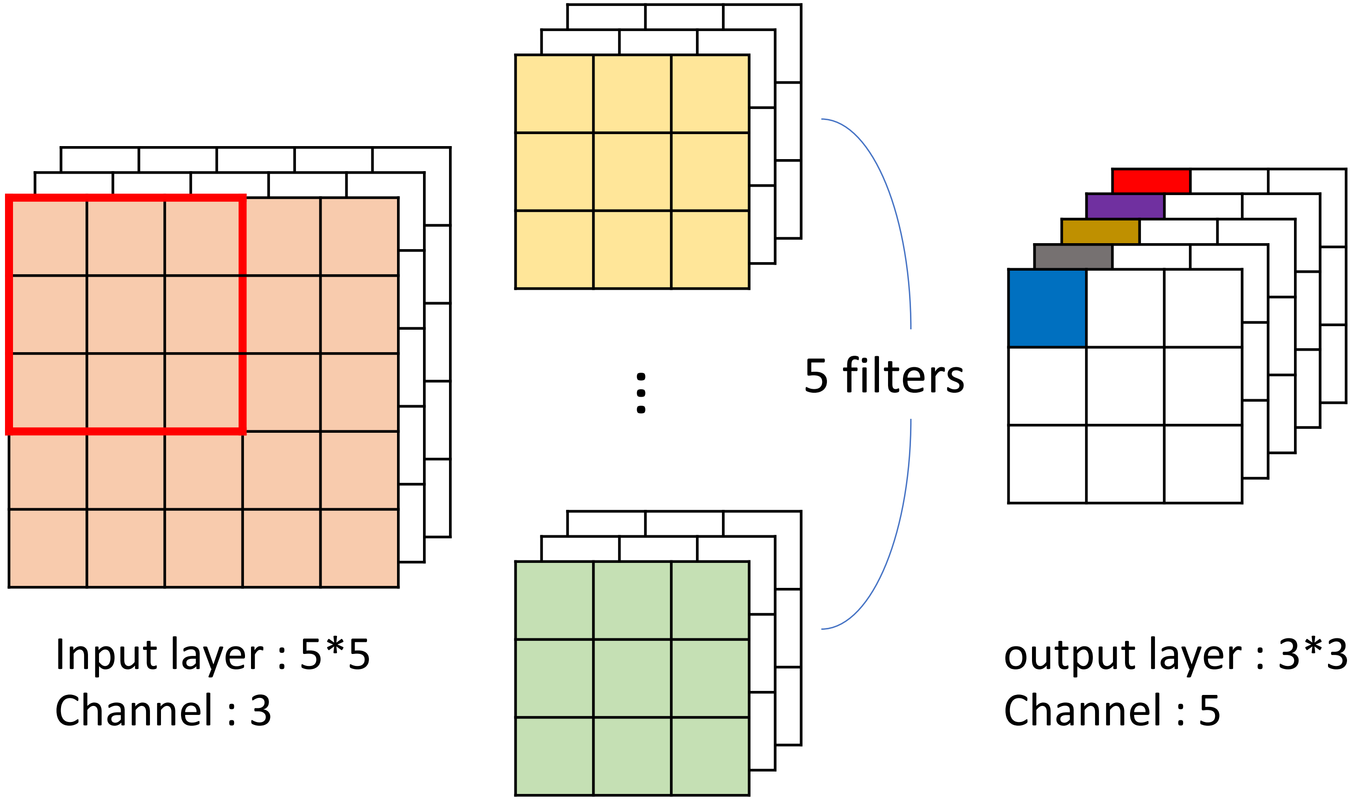
- It's the computation unit in AI Accelerator
- There are a lot of PEs in AI Accelerator
- PE loads data from SRAM, compute and save the result back to SRAM
- SRAM can be divided into 3 types according to the data it stores
  - Input feature map
  - Weight
  - Output feature map

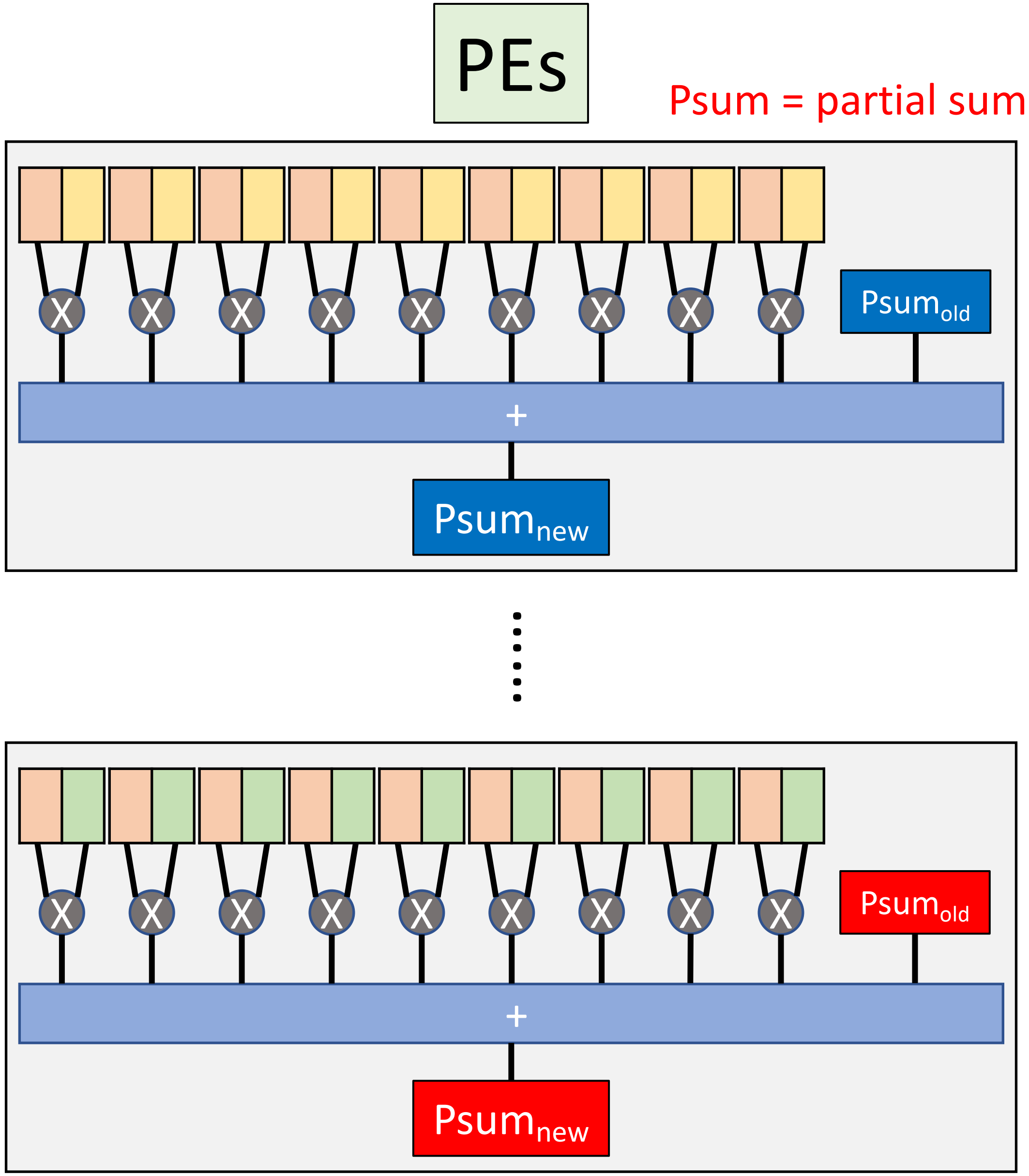


PE(Processing Element)

- It's the computation unit in AI Accelerator
- There are a lot of PEs in AI Accelerator

- **Take 3*3, 1 stride Convolution for example**

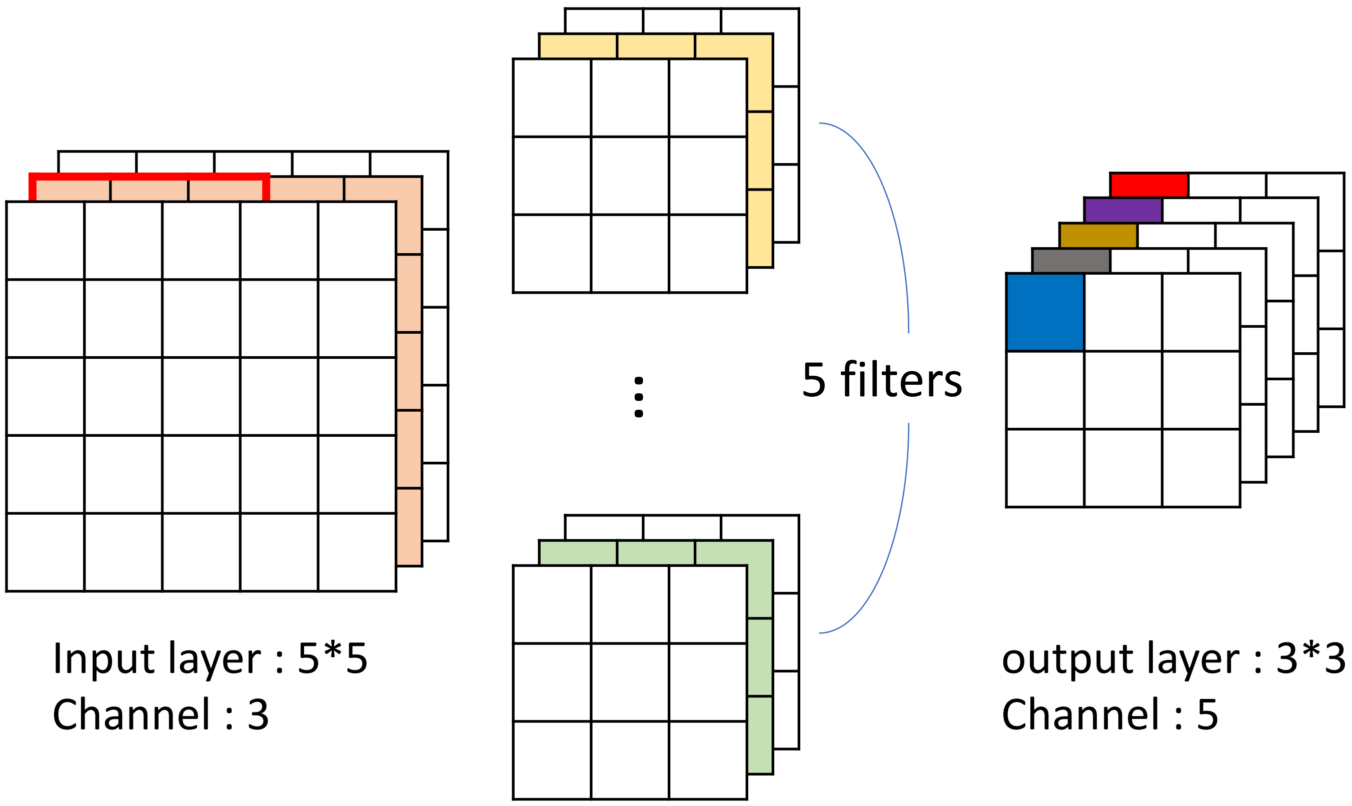


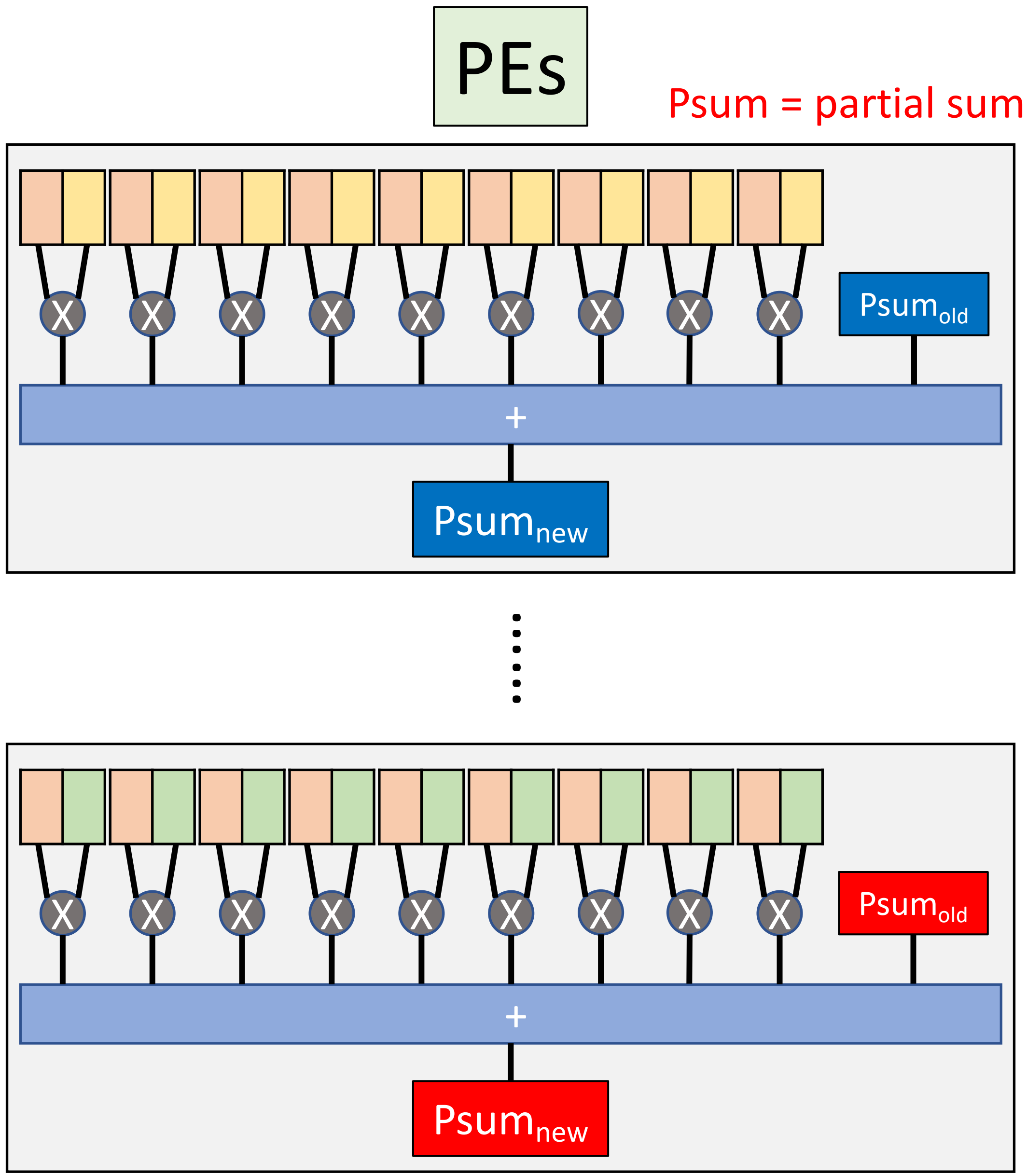


PE(Processing Element)

- It's the computation unit in AI Accelerator
- There are a lot of PEs in AI Accelerator

- **Take 3*3, 1 stride Convolution for example**

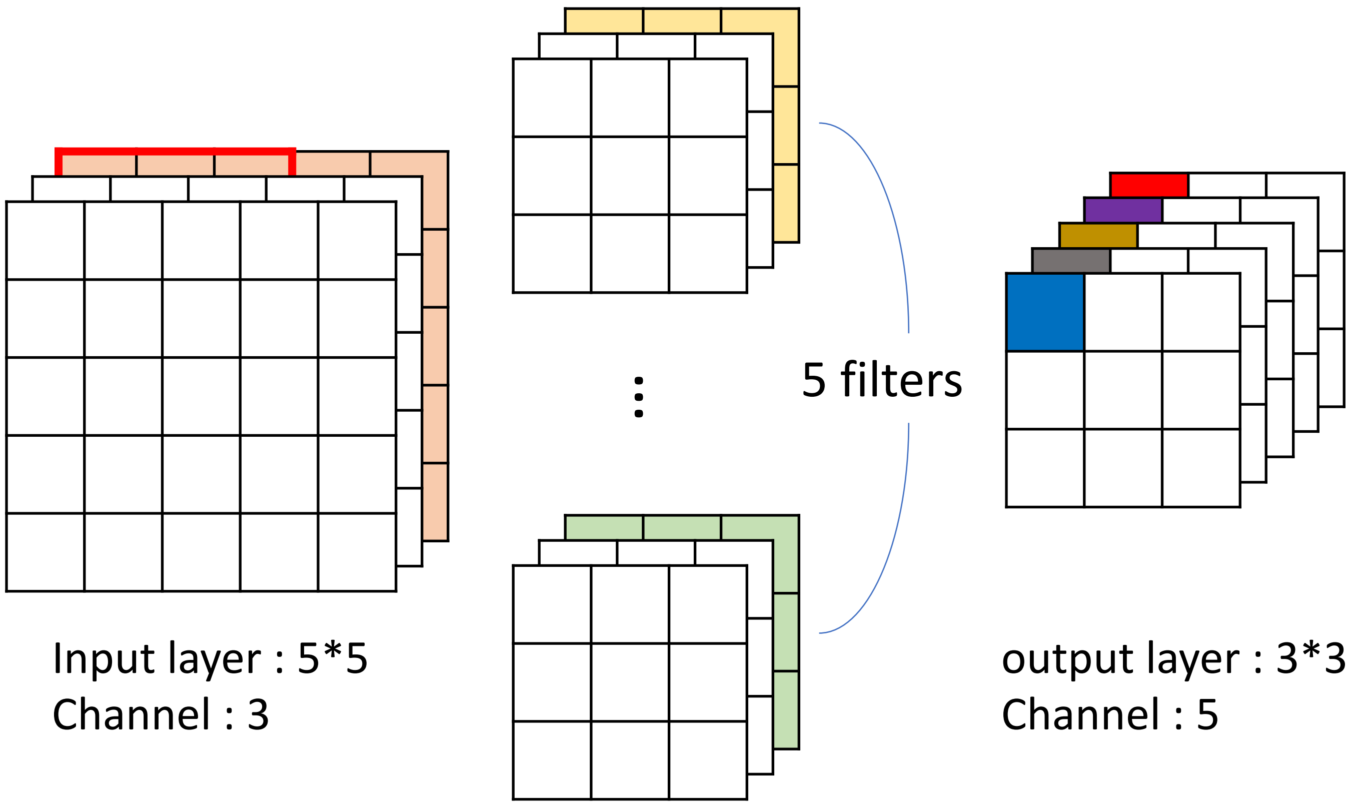


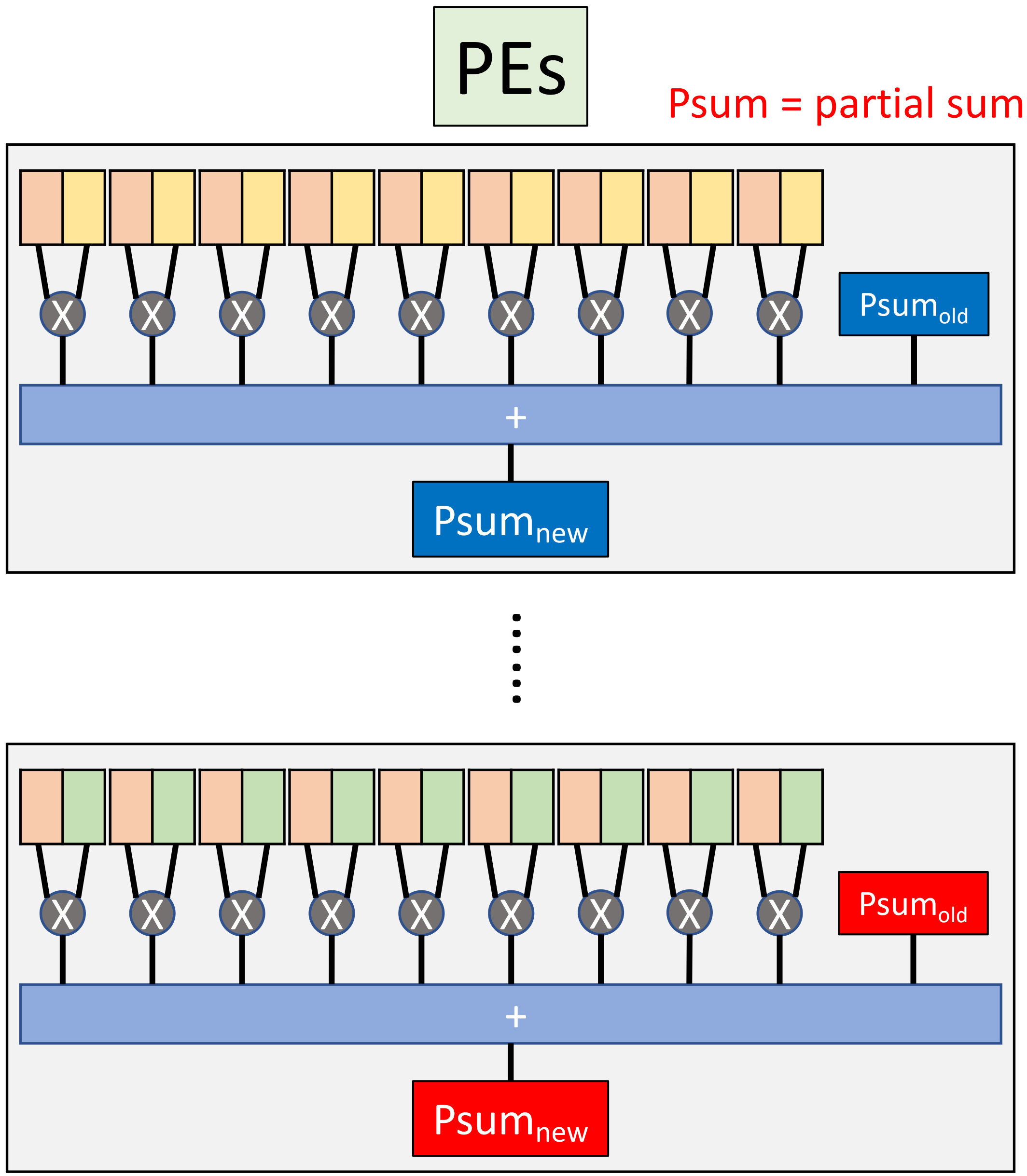


PE(Processing Element)

- It's the computation unit in AI Accelerator
- There are a lot of PEs in AI Accelerator

- **Take 3*3, 1 stride Convolution for example**

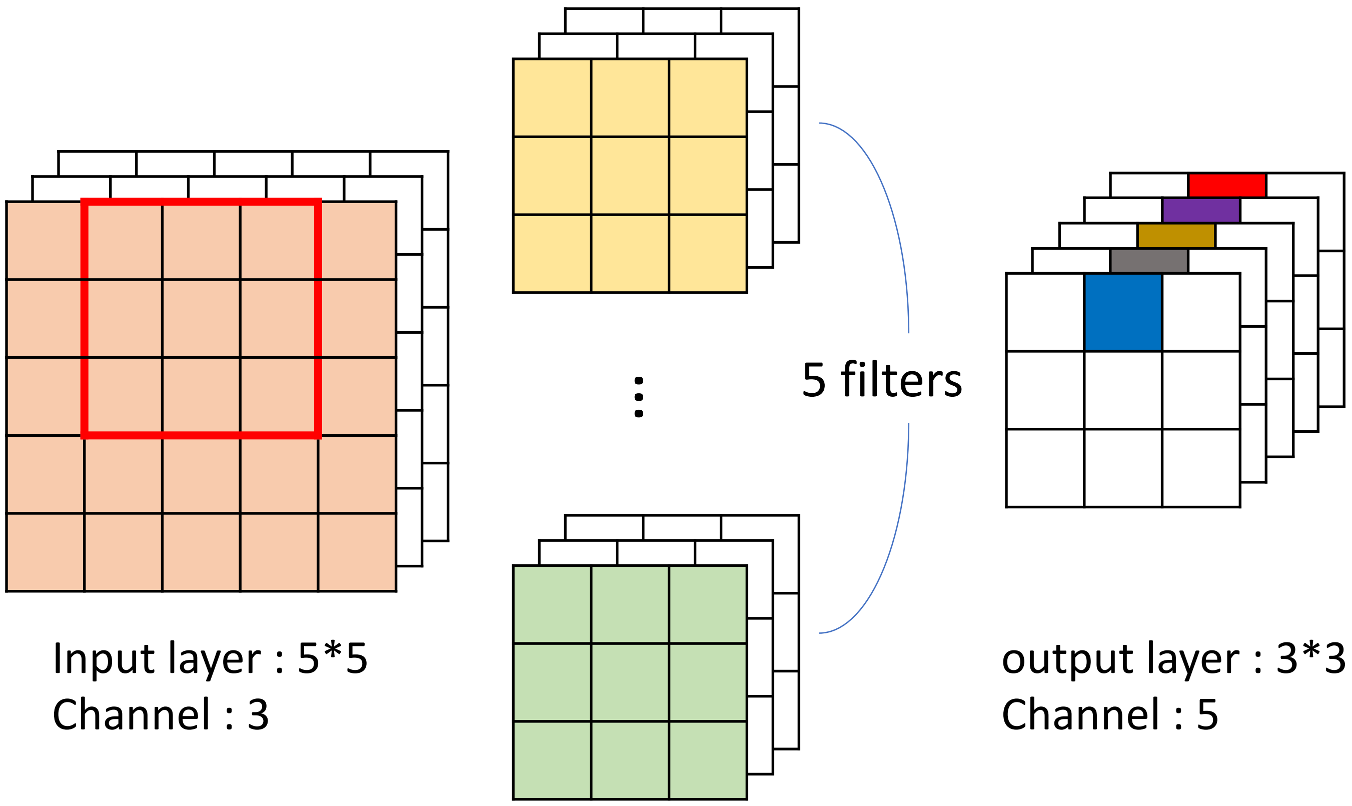


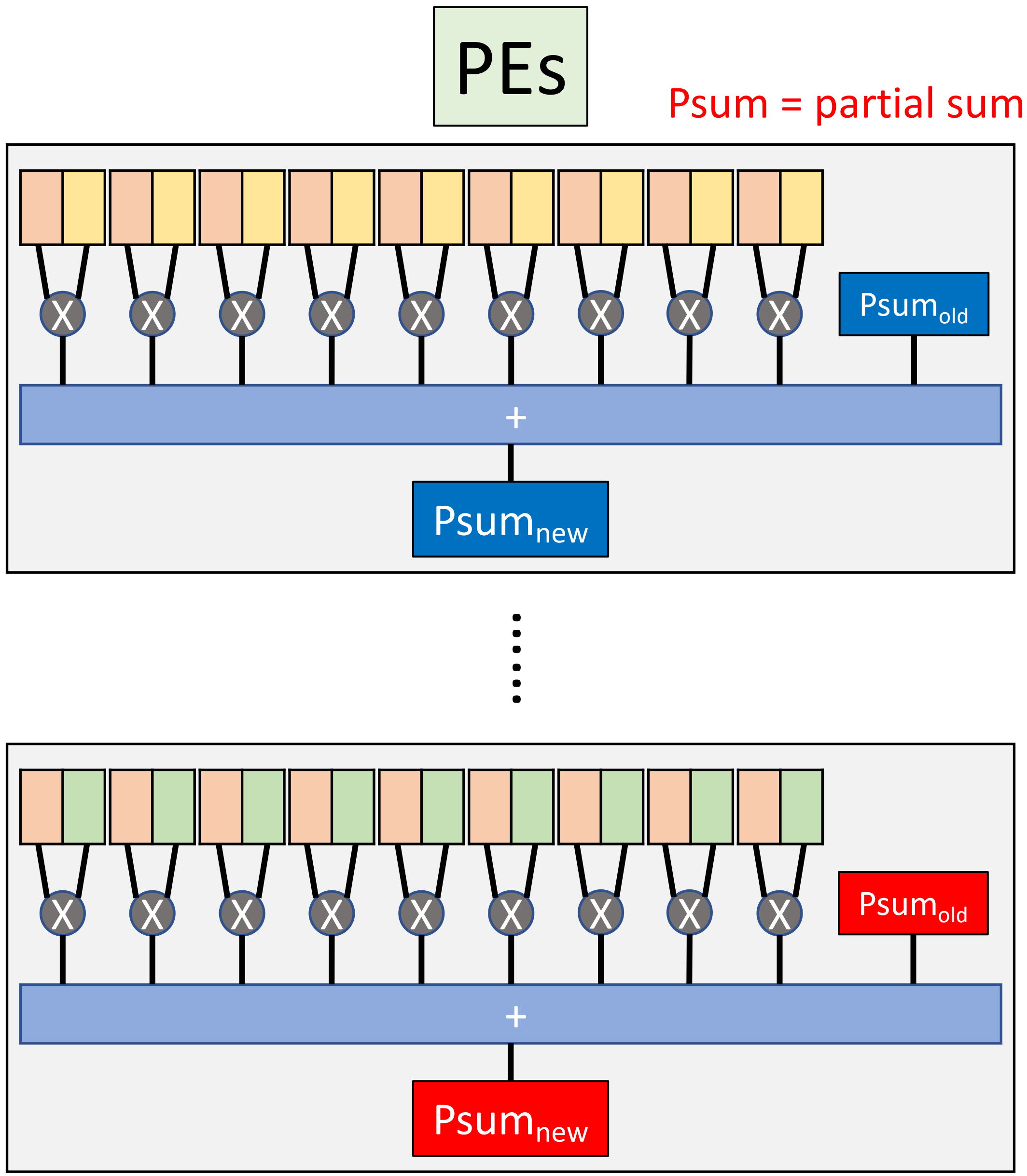


PE(Processing Element)

- It's the computation unit in AI Accelerator
- There are a lot of PEs in AI Accelerator

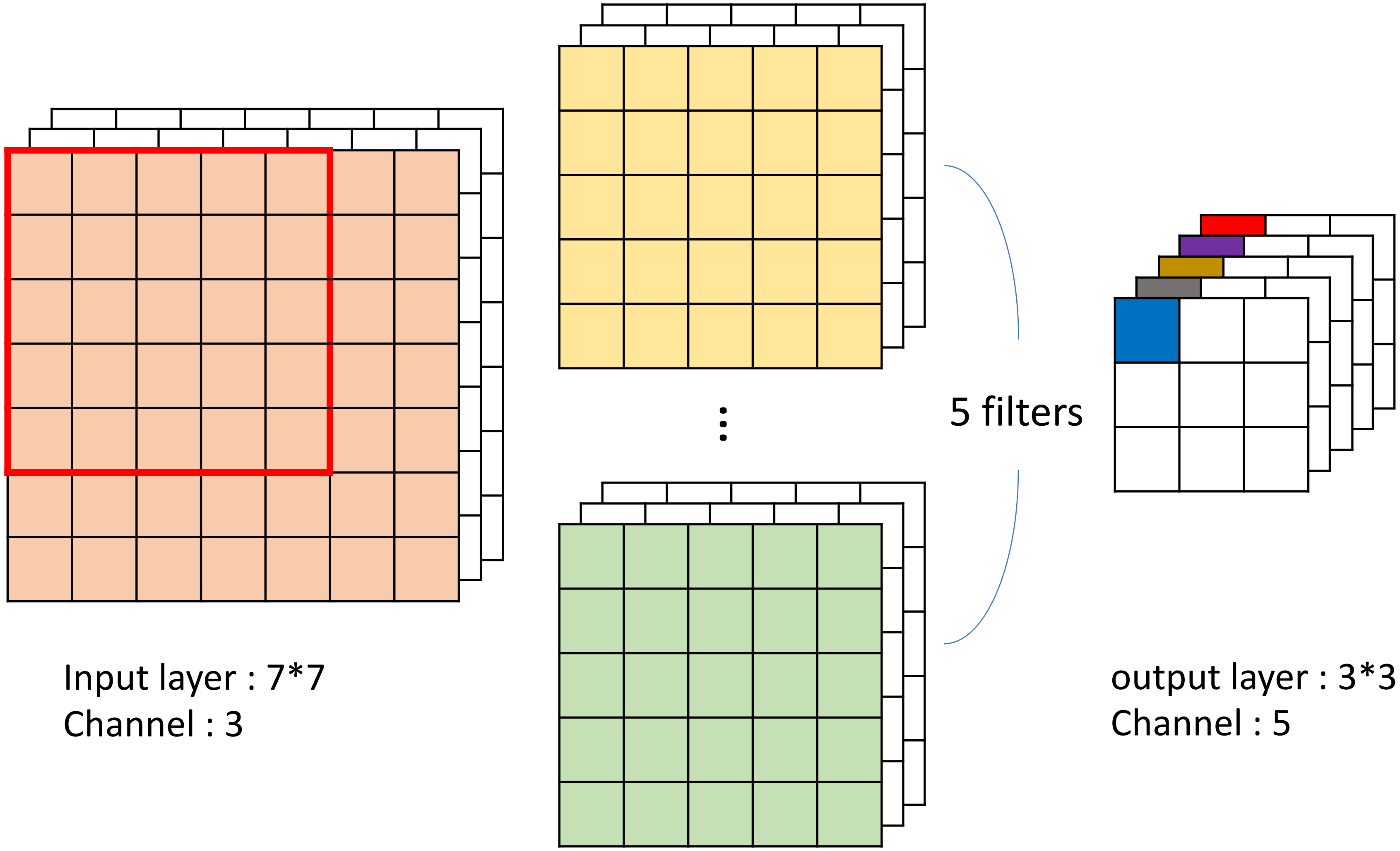
- **Take 3*3, 1 stride Convolution for example**





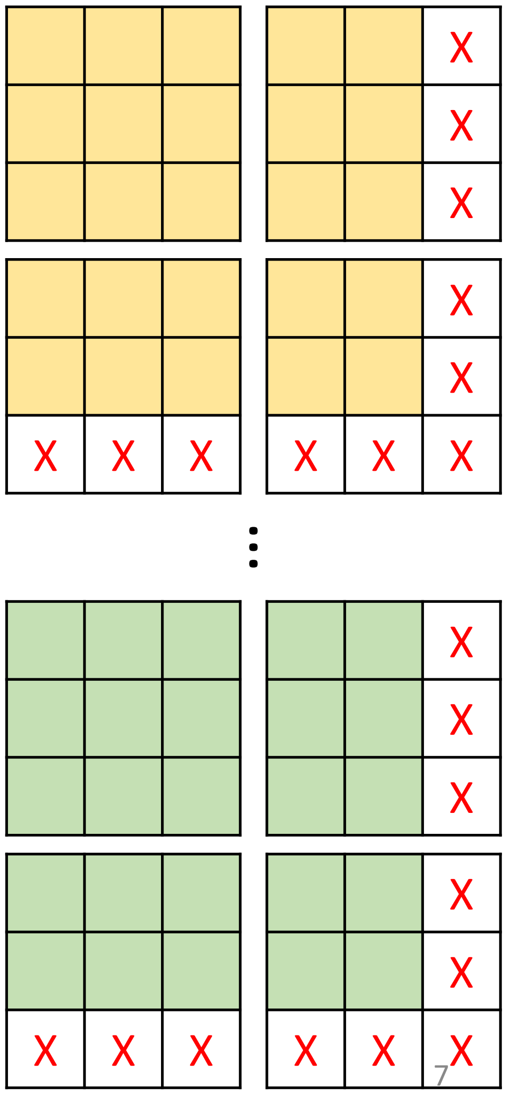
PE(Processing Element)

- How about 5*5 convolution in these 3*3 PEs ?



4 PEs each convolution

This method will waste
a lot of computation opportunities

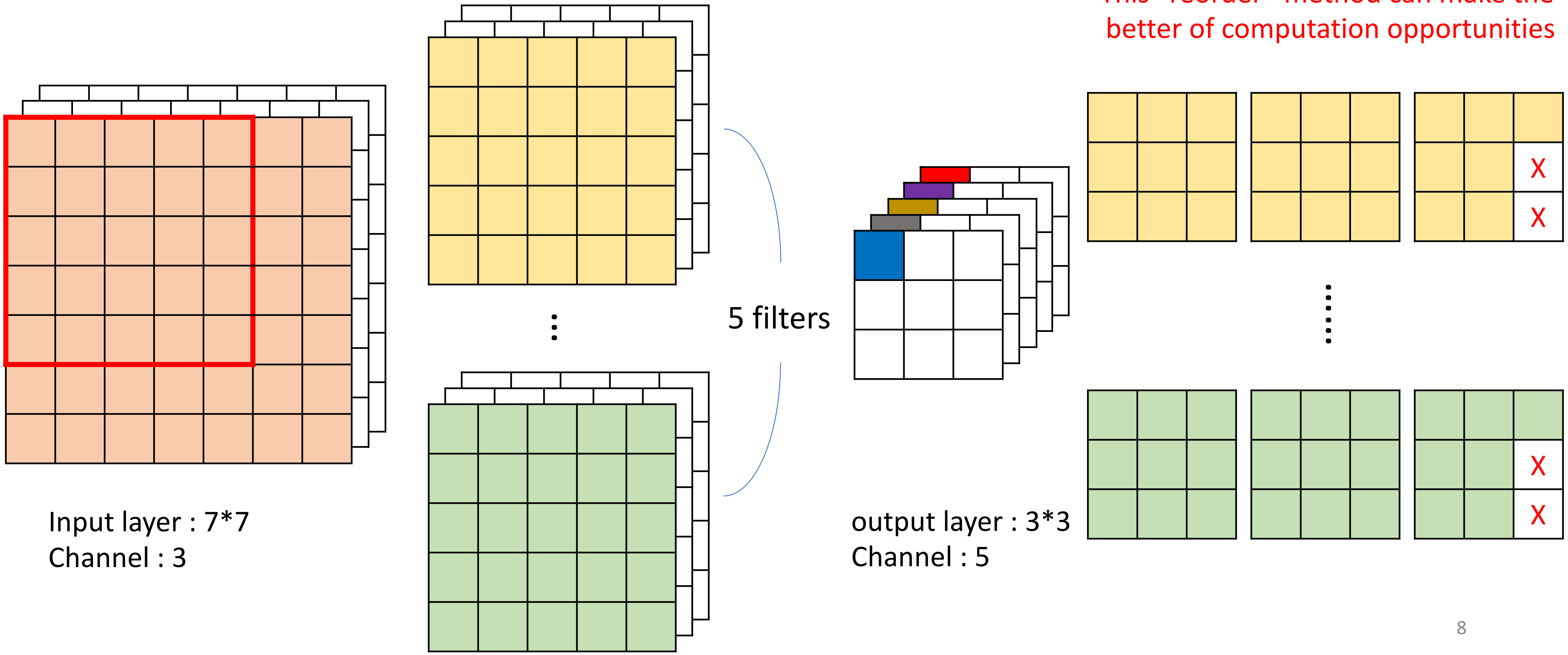


PE(Processing Element)

- How about 5*5 convolution in these 3*3 PEs ?

3 PEs each convolution

This "reorder" method can make the better of computation opportunities



Input layer : 7*7
Channel : 3

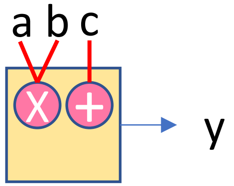
output layer : 3*3
Channel : 5

PE(Processing Element)

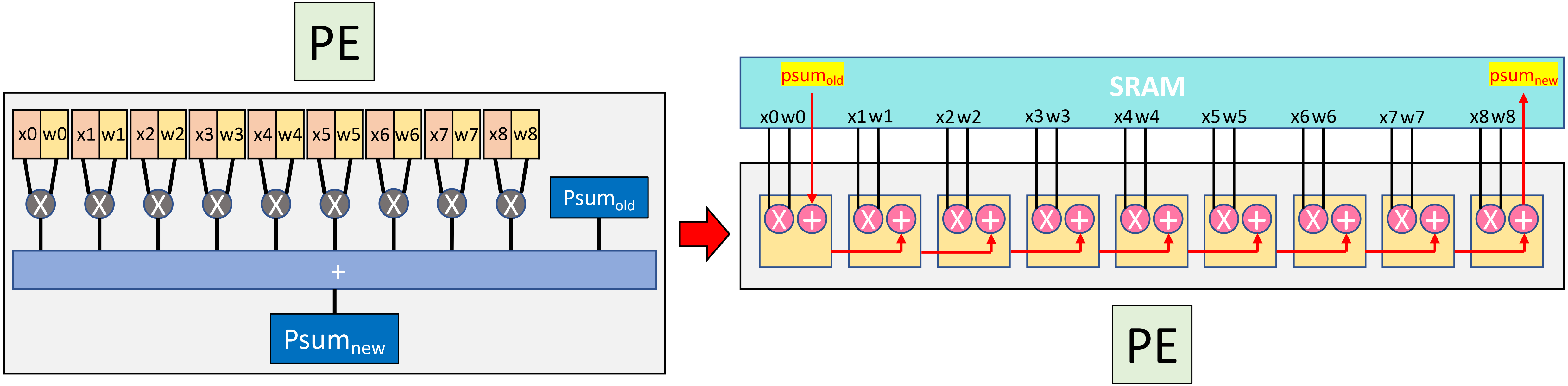
- We can view 3*3 PE as 9 MAC operations

- MAC (Multiply Accumulate)

$$y \leftarrow a * b + c$$

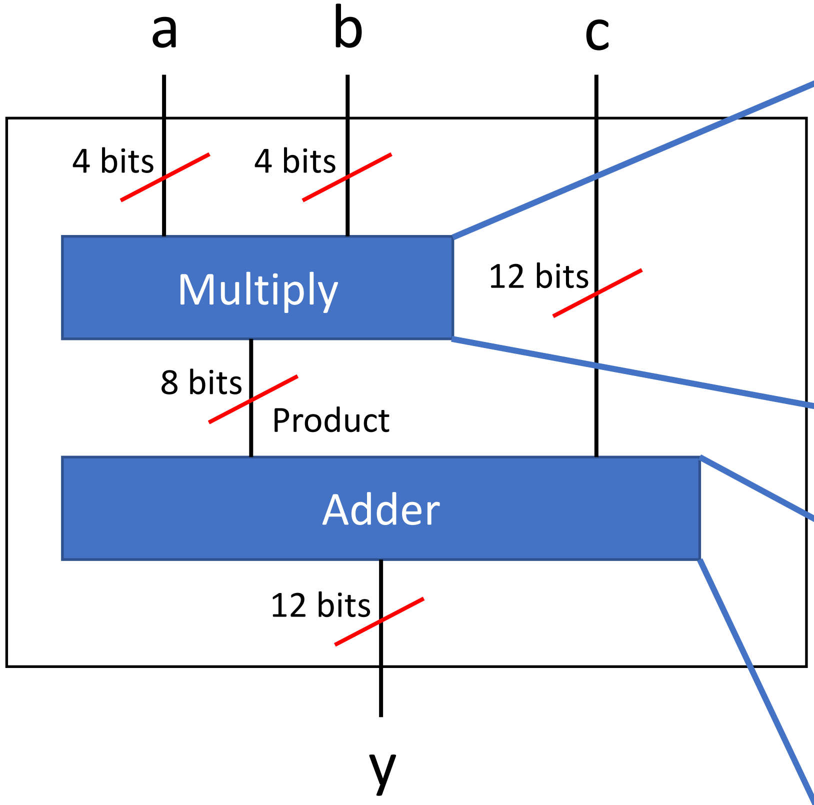


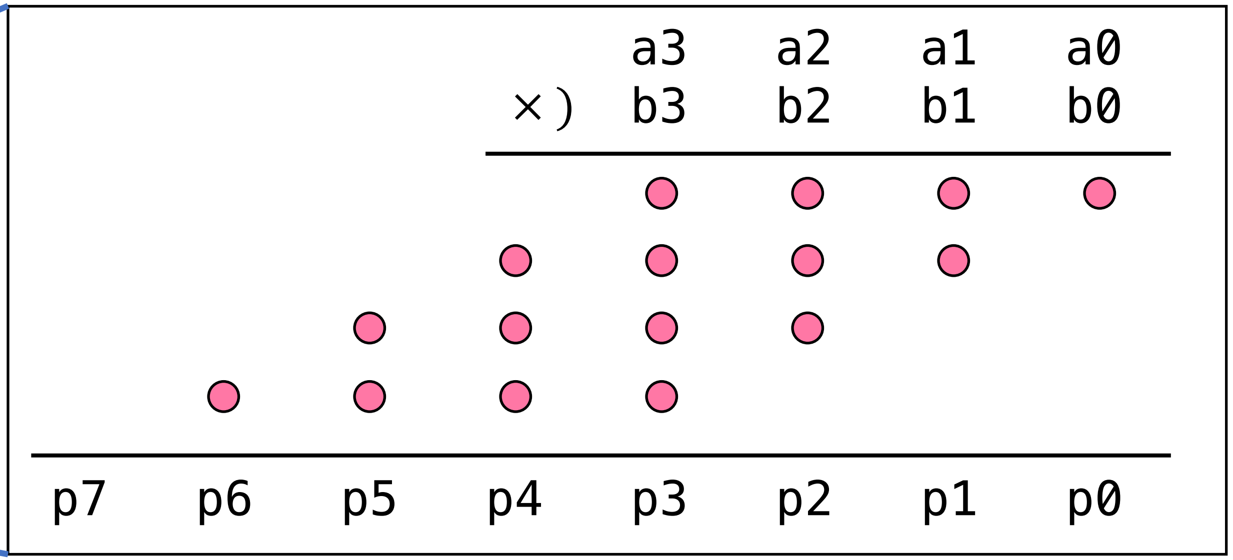
$$
\begin{aligned}
psum_{mid} &\leftarrow x0 * w0 + psum_{old} \\
psum_{mid} &\leftarrow x1 * w1 + psum_{mid} \\
psum_{mid} &\leftarrow x2 * w2 + psum_{mid} \\
psum_{mid} &\leftarrow x3 * w3 + psum_{mid} \\
psum_{mid} &\leftarrow x4 * w4 + psum_{mid} \\
psum_{mid} &\leftarrow x5 * w5 + psum_{mid} \\
psum_{mid} &\leftarrow x6 * w6 + psum_{mid} \\
psum_{mid} &\leftarrow x7 * w7 + psum_{mid} \\
psum_{new} &\leftarrow x8 * w8 + psum_{mid}
\end{aligned}
$$



MAC (Multiply Accumulate)

$$y\ (12bits) \leftarrow a\ (4bits) \times b(4bits) + c(12bits)$$





	0	0	0	0	p7	p6	p5	p4	p3	p2	p1	p0
+)	c11	c10	c9	c8	c7	c6	c5	c4	c3	c2	c1	c0
y12	y11	y10	y9	y8	y7	y6	y5	y4	y3	y2	y1	y0

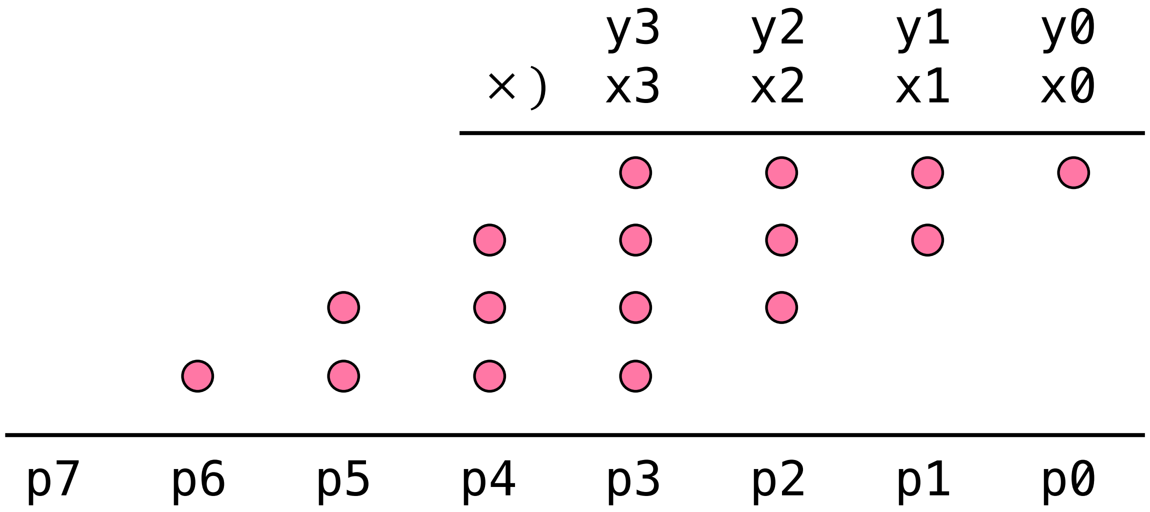
Just ignore it, if it overflow, it will become noise.

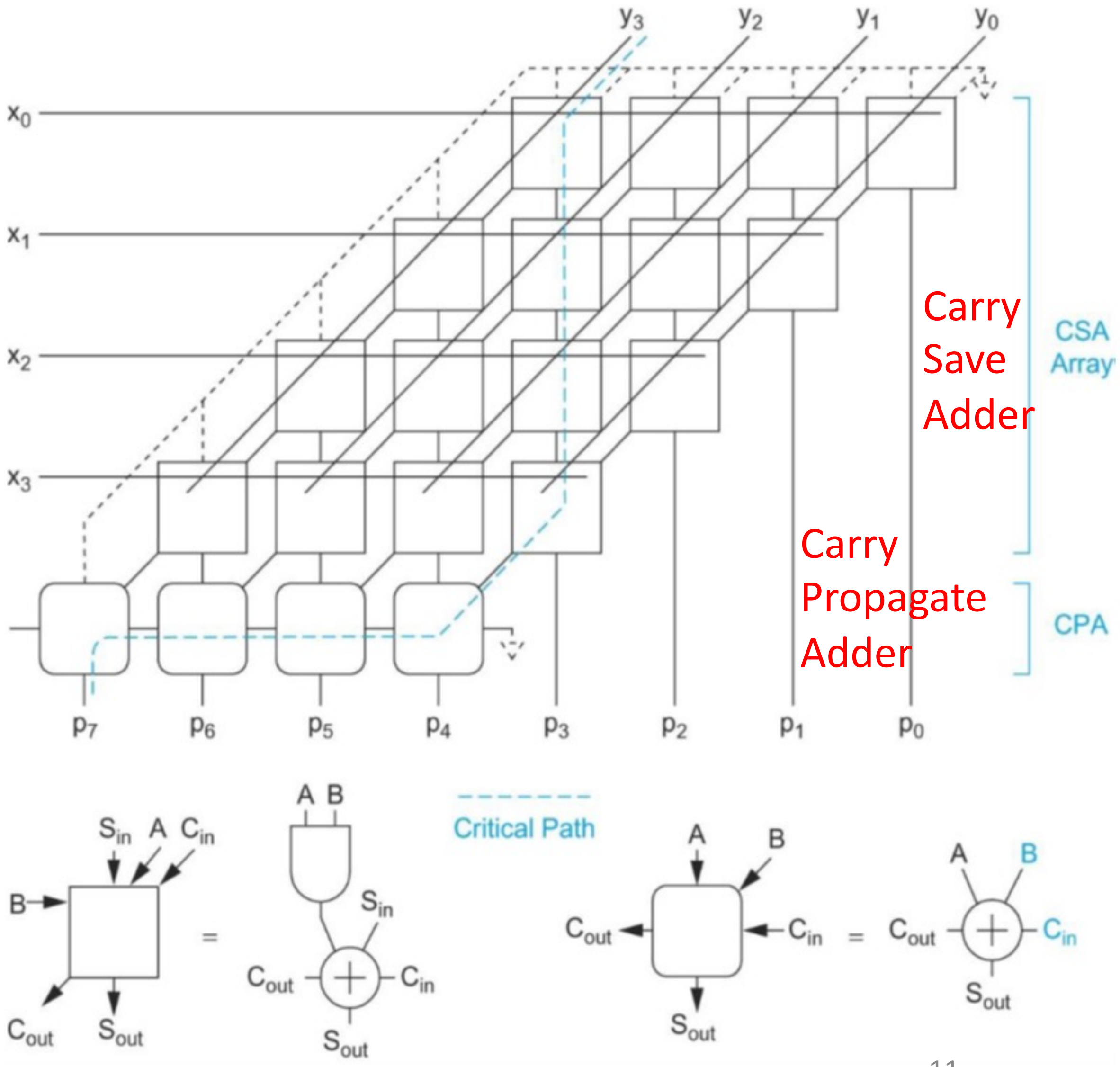
MAC (Multiply Accumulate)

If you don't know what is CSA / CPA
Please see the next page first

Multiply (3 steps)

1. Partial Product Generator (PPG)
   - Do AND operation on each bit of x & y
   - Produce the pink points
2. Partial Product Reduction Tree (PPRT)
   - Efficiently add all pink points, we use CSA
3. Carry-Propagate Adder (CPA)
   - Last Adder





CPA vs CSA

Take 3 numbers addition as example

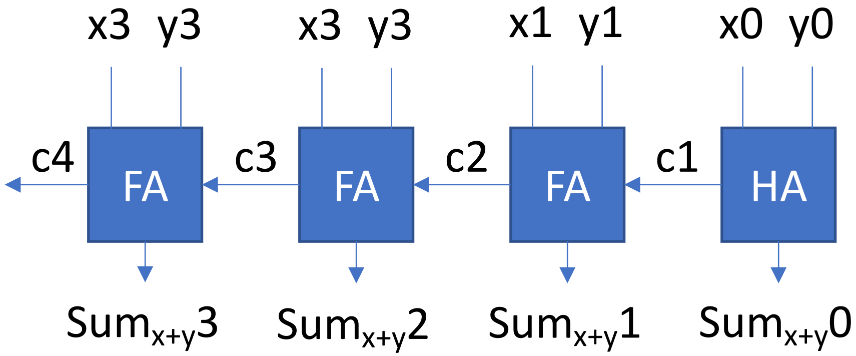
$$
\begin{array}{rl}
 & x3\ x2\ x1\ x0 \\
+) & y3\ y2\ y1\ y0 \\
+) & z3\ z2\ \ z1\ z0
\end{array}
$$

M = How many numbers to add = 3
N = How many bits in a number = 4

Carry-Propagate Adder (delay : (M-1) * N FAs)

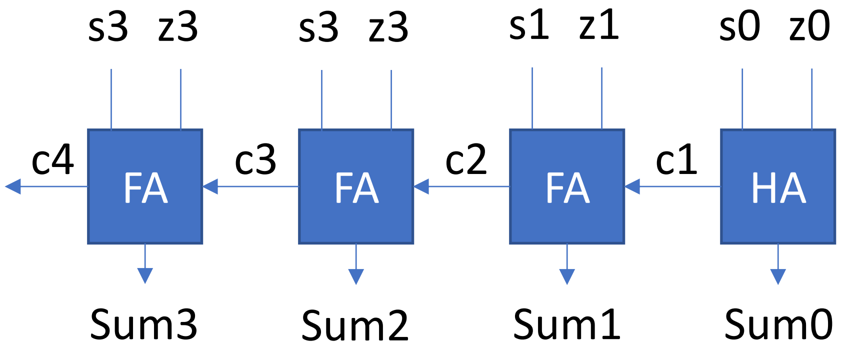
Step 1 : $Sum_{x+y} = x + y$

Time : N FAs



Step 2 : $Sum = Sum_{x+y} + z$

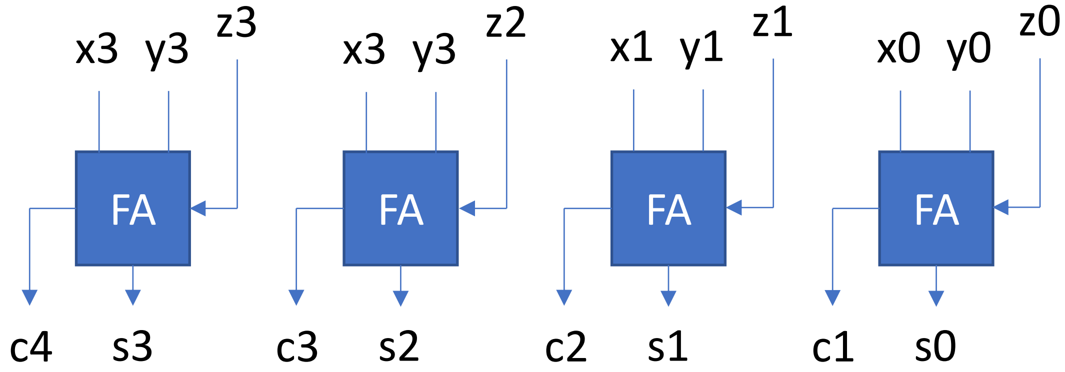
Time : N FAs



Carry-Save Adder (delay : (M-2) + N FAs)

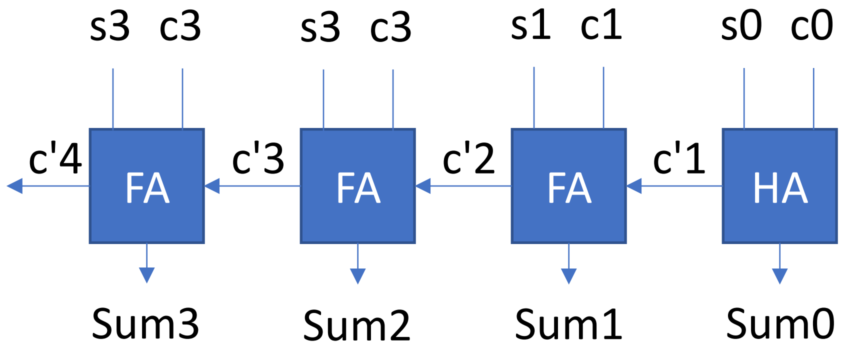
Step 1 : Each bit does FA independently

Time : 1 FAs



Step 2 : Do Carry-Propagate, Sum = c + s

Time : N FAs



$$
\begin{array}{rl}
 & s3\ s2\ s1\ s0 \\
+) & c3\ c2\ c1\ \ 0
\end{array}
$$

Supplement : Another PPRT Method, Wallace Tree

Take 8*8 multiply as example

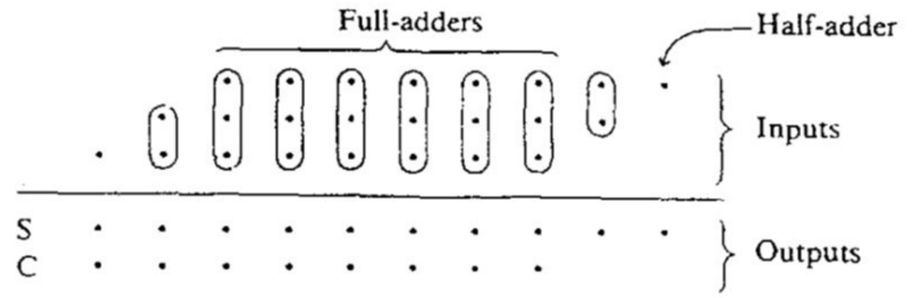
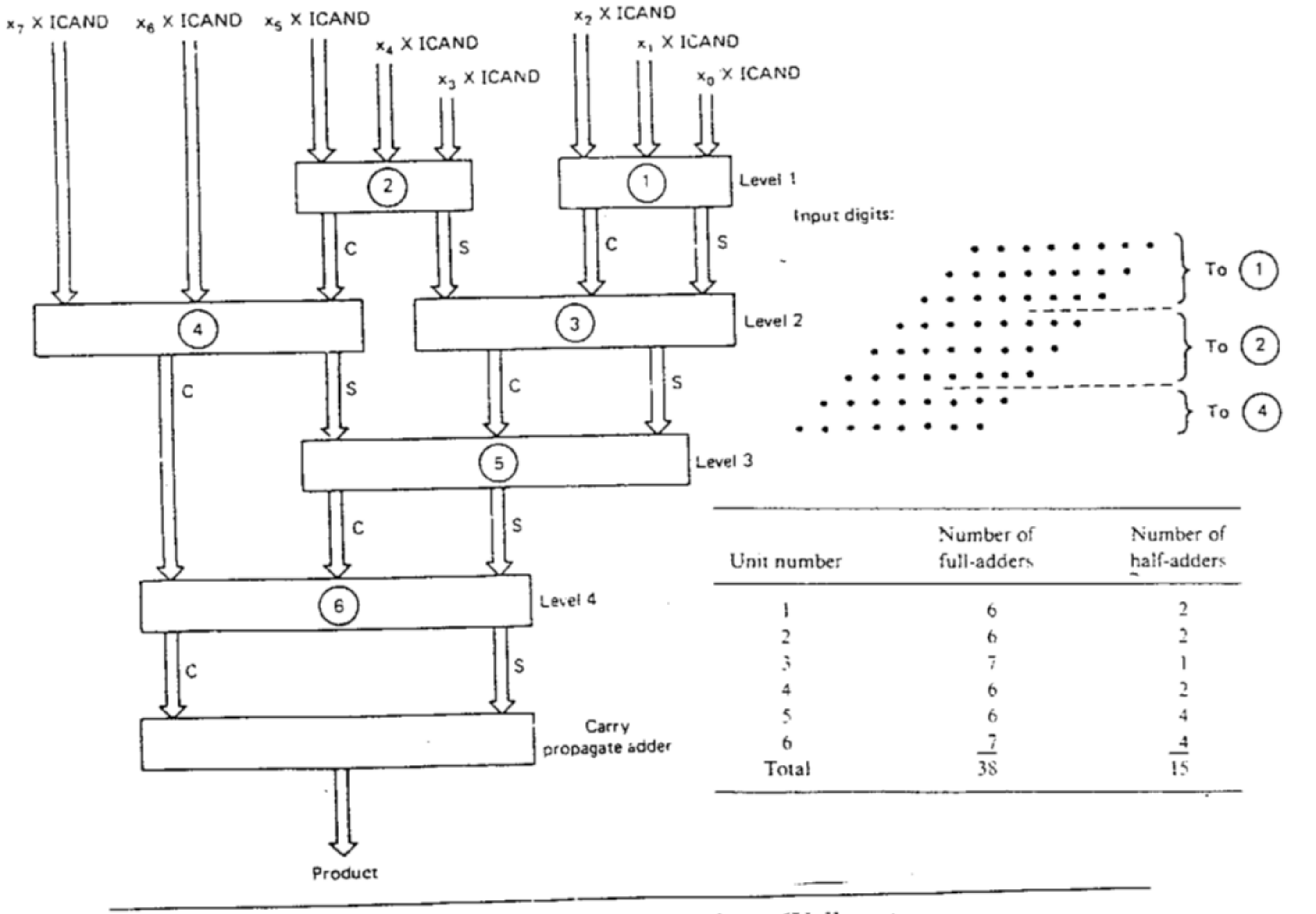


Figure 6.25 Three vectors producing two vectors at the outputs.

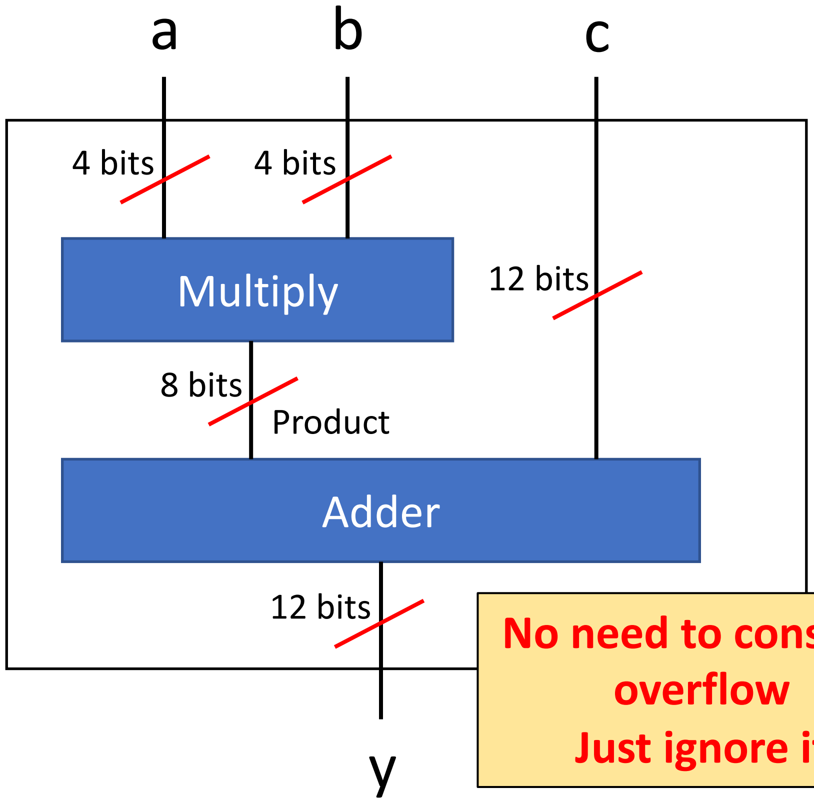


Unit number	Number of full-adders	Number of half-adders
1	6	2
2	6	2
3	7	1
4	6	2
5	6	4
6	7	4
Total	38	15

Figure 6.26 8-bit*8-bit multiplication structure using a Wallace tree.

Homework (Verilog & Report)

1. Implement Half-adder & Full-adder
2. Implement a 4-bit MAC mentioned in this ppt
   - You can only use the components below
     - Half-Adder / Full-Adder you implemented
     - Verilog Built-in AND gate
   - Pass all test data
   - Analyze how many Half Adder and Full Adder you use
   - Analyze the delay / latency of your 4-bit MAC



No need to consider overflow
Just ignore it

3. Use 4-bit MAC to combine a 8-bit MAC
   - Pass all test data
   - Explain what's the difference between
     ① A primitive 8-bit MAC (same method as 4-bit MAC you implemented, but (a, b, c) = (8, 8, 24) bits)
     ② The combined 8-bit MAC (combined by 4-bit MAC)

4. Use primitive 8-bit MAC to explain the difference before and after using Wallace Tree
   - Please explain what is Wallace Tree first
   - Just think and explain, do not need to implement

Homework – Verilog Component Format

1. HalfAdder
   - File name : HA.v
   - Module name : HA
2. FullAdder
   - File name : FA.v
   - Module name : FA
3. 4-bit MAC
   - File name : MAC_4bit.v
   - Module name : MAC_4bit
4. 8-bit MAC
   - File name : MAC_8bit.v
   - Module name : MAC_8bit

4-bit MAC

```
`include “HA.v”
`include “FA.v”
module MAC_4bit (
    input [3:0] a,
    input [3:0] b,
    input [11:0] c,
    output [11:0] result,
    output cout
);

/* Here is your code */

endmodule
```

8-bit MAC

```
`include “MAC_4bit.v”
module MAC_8bit (
    input [7:0] a,
    input [7:0] b,
    input [23:0] c,
    output [23:0] result,
    output cout
);

/* Here is your code */

endmodule
```

Homework – Report Format

Please turn the report into PDF

1. 4-bit MAC
    - Screenshot all pass screen
    - The analyzation of your 4-bit MAC
        - Simply introduce your 4-bit MAC
        - (Hardware Resource) Count how many HA & FA you used
        - (Time Delay) Count the delay / latency of your 4-bit MAC (you can use FA as time unit)

2. 8-bit MAC
    - Screenshot all pass screen
    - Detailly introduce how you combined it through 4-bit MAC you implemented
    - Explain what's the difference (hardware resource / delay) between
        ① A primitive 8-bit MAC (same method as 4-bit MAC you implemented, but (a, b, c) = (8, 8, 24) bits)
        ② The combined 8-bit MAC (combined by 4-bit MAC you implemented)

3. Wallace Tree
    - Introduce what is Wallace Tree
    - Explain the difference before and after using Wallace Tree with the primitive 8-bit MAC

Homework – File Structure

AOC4_StudentID_Name → AOC4_StudentID_Name.zip

- AOC4_StudentID_Name
  - AOC4_Report_StudentID_Name.pdf
  - HA.v
  - FA.v
  - MAC_4bit.v
  - MAC_8bit.v

Please turn the report into PDF