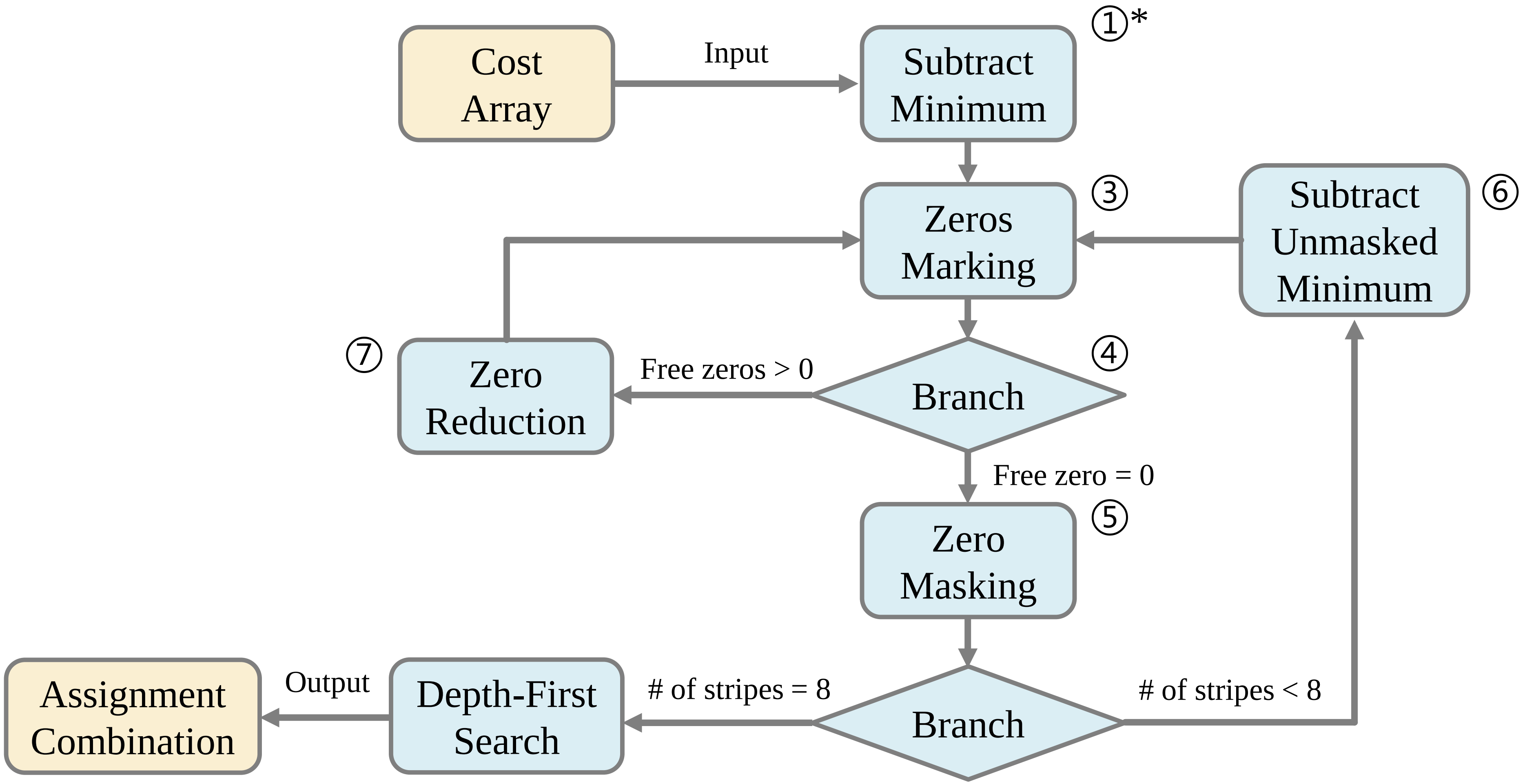
Final Project – Job Assignment Machine

Shu-Hung, Kuo

Hungary Algorithm

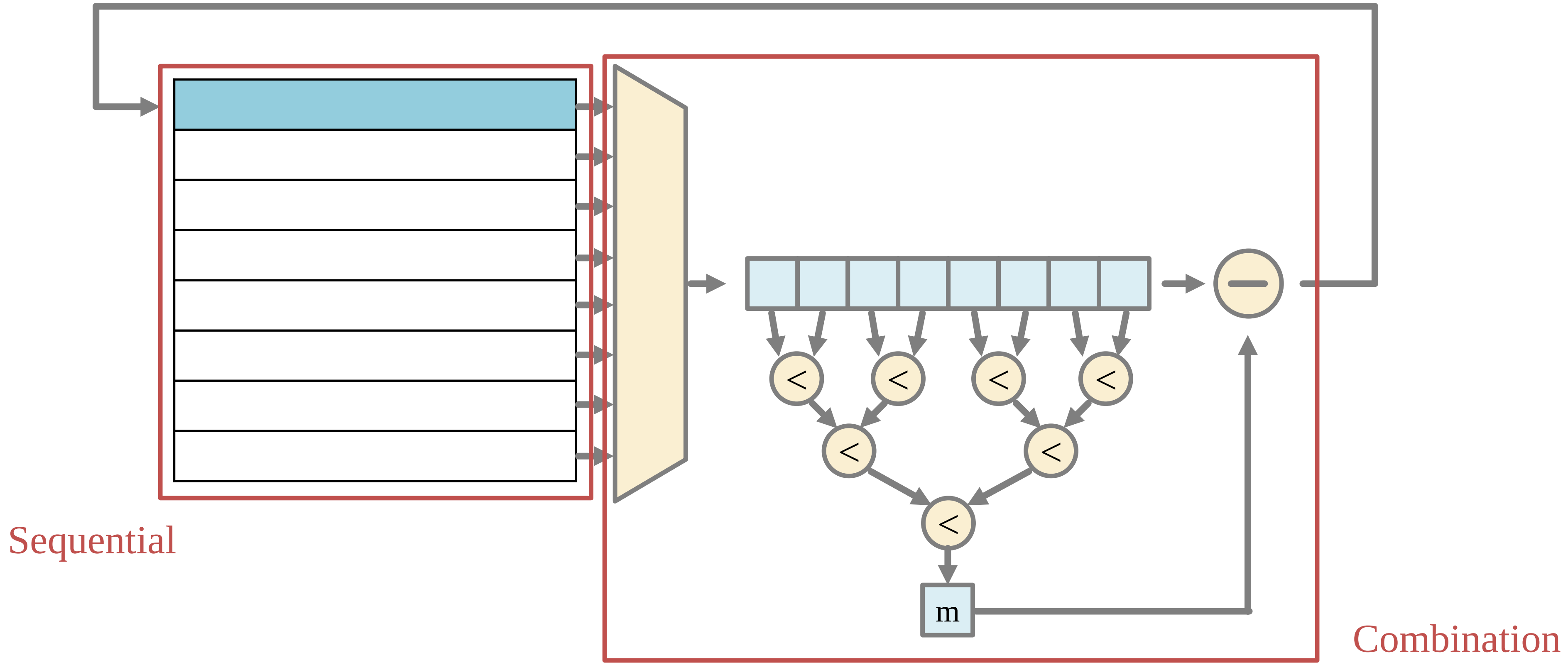
- Reference: https://bbs.csdn.net/topics/390105782



*對應到Reference的步驟

Block diagram color

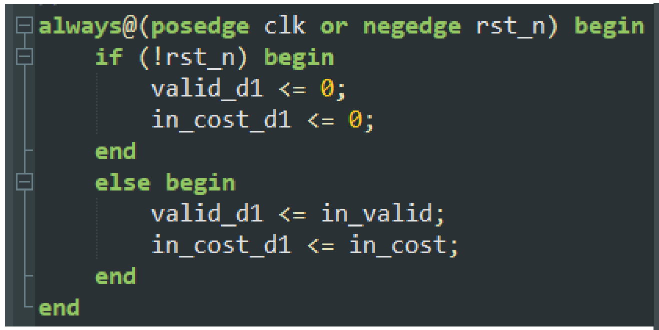
- Sequential Circuits are illustrated in dark color.
- Combination Circuits are illustrated in light color.



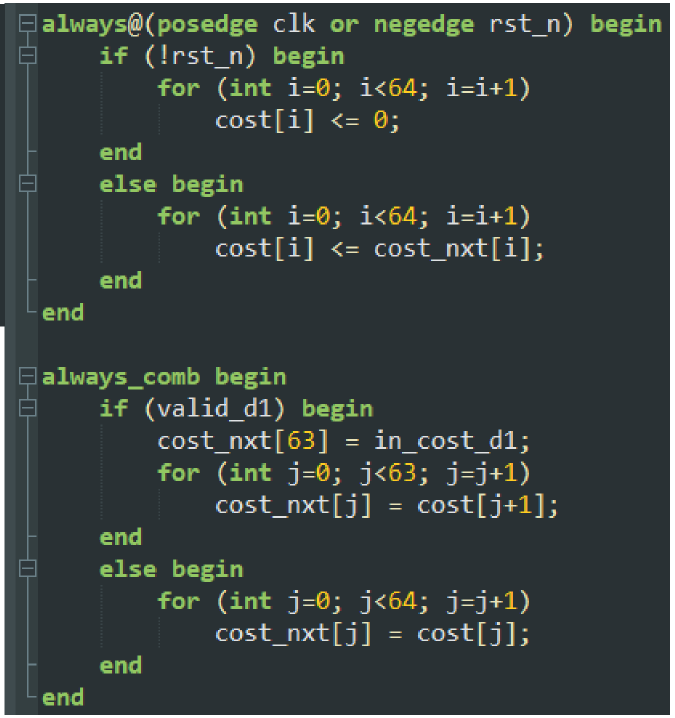
Input Cost Array

- Shift Register

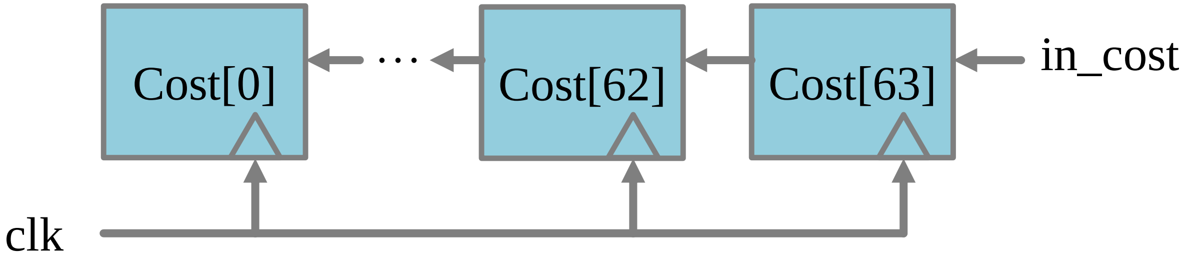
	J1	J2	J3	J4	J5	J6	J7	J8
W1	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7
W2	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}
W3	C_{16}	C_{17}	C_{18}	C_{19}	C_{20}	C_{21}	C_{22}	C_{23}
W4	C_{24}	C_{25}	C_{26}	C_{27}	C_{28}	C_{29}	C_{30}	C_{31}
W5	C_{32}	C_{33}	C_{34}	C_{35}	C_{36}	C_{37}	C_{38}	C_{39}
W6	C_{40}	C_{41}	C_{42}	C_{43}	C_{44}	C_{45}	C_{46}	C_{47}
W7	C_{48}	C_{49}	C_{50}	C_{51}	C_{52}	C_{53}	C_{54}	C_{55}
W8	C_{56}	C_{57}	C_{58}	C_{59}	C_{60}	C_{61}	C_{62}	C_{63}



```
always@(posedge clk or negedge rst_n) begin
    if (!rst_n) begin
        valid_d1 <= 0;
        in_cost_d1 <= 0;
    end
    else begin
        valid_d1 <= in_valid;
        in_cost_d1 <= in_cost;
    end
end
```

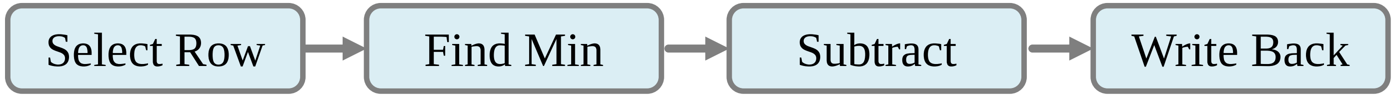


```
always@(posedge clk or negedge rst_n) begin
    if (!rst_n) begin
        for (int i=0; i<64; i=i+1)
            cost[i] <= 0;
    end
    else begin
        for (int i=0; i<64; i=i+1)
            cost[i] <= cost_nxt[i];
    end
end

always_comb begin
    if (valid_d1) begin
        cost_nxt[63] = in_cost_d1;
        for (int j=0; j<63; j=j+1)
            cost_nxt[j] = cost[j+1];
    end
    else begin
        for (int j=0; j<64; j=j+1)
            cost_nxt[j] = cost[j];
    end
end
```



Subtract Minimum

- Step break down
  - (1) 歷遍所有Row，求得各Row的最小值，並將各Row元素減去其最小值
  - (2) 歷遍所有Col，求得各Col的最小值，並將各Col元素減去其最小值
- Row-wise operation
  - 一個一個找太慢，全部一起平行找寫起來太複雜、面積太大
  - 折衷方案，一次找一個Row



Select Row → Find Min → Subtract → Write Back

1	44	33	53	35	35	35	48	61
2	48	63	35	40	32	60	62	40
3	58	55	48	62	32	50	57	41
4	51	55	49	38	51	53	58	50
5	52	40	40	60	38	42	56	56
6	56	52	41	58	32	48	60	58
7	48	63	37	47	61	43	62	53
8	52	33	42	36	59	62	35	55

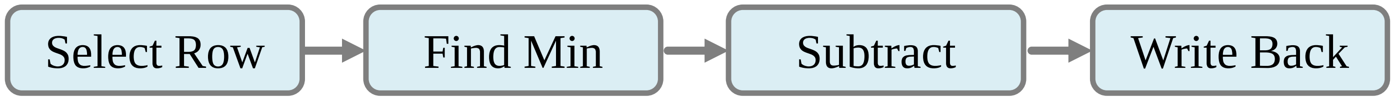
↓ (1)

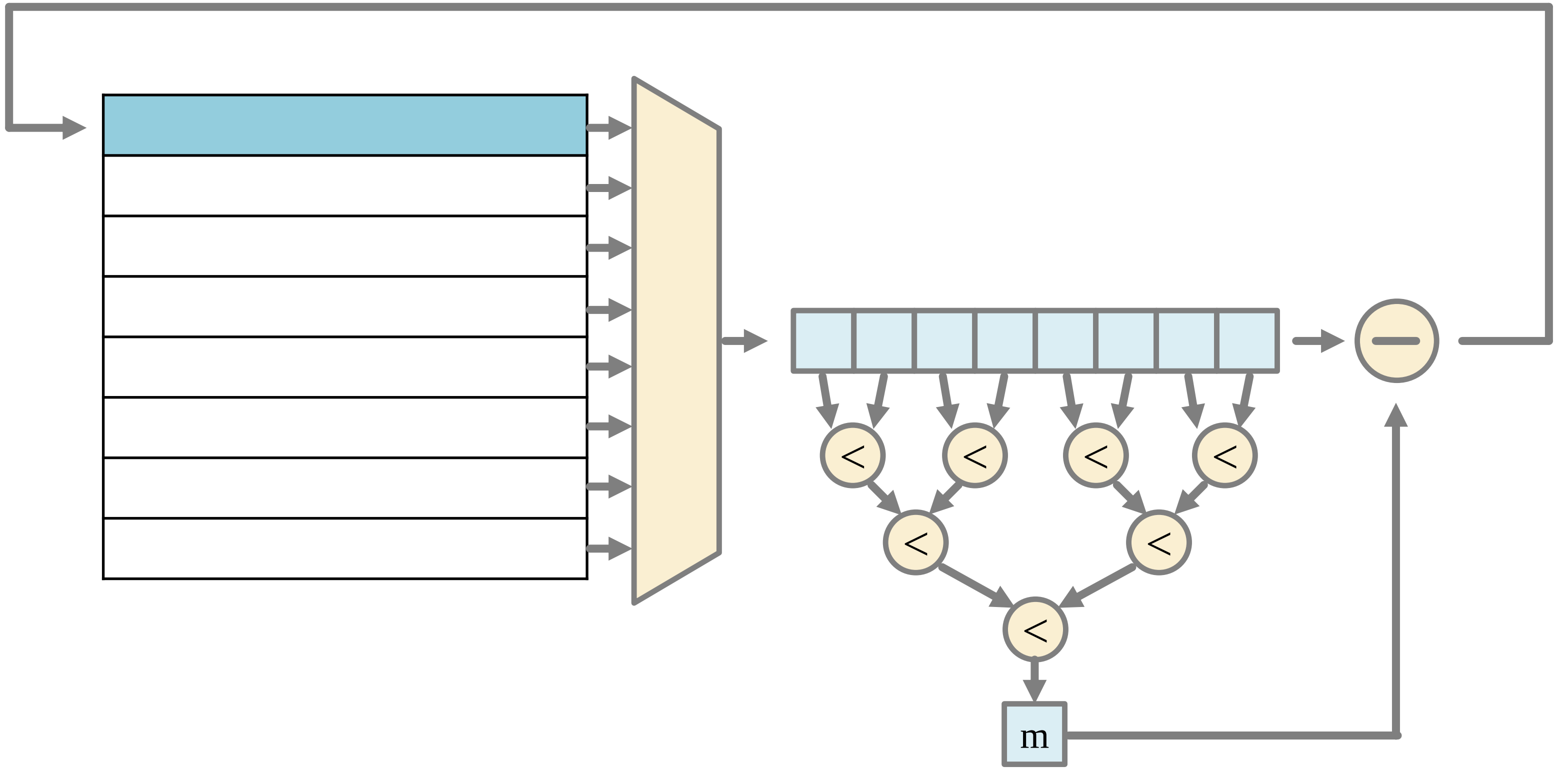
11	0	20	2	2	30	43	58
16	31	3	8	0	45	47	25
26	23	16	30	0	48	55	39
13	17	11	0	13	50	56	48
14	2	2	22	0	29	43	47
24	20	9	26	0	48	60	58
11	26	0	10	24	35	62	53
19	0	9	3	26	29	2	22

↓ (2)

0	0	20	2	2	1	41	36
5	31	3	8	0	16	45	3
15	23	16	30	0	19	53	17
2	17	11	0	13	21	54	26
3	2	2	22	0	0	41	25
13	20	9	26	0	19	58	36
0	26	0	10	24	6	60	31
8	0	9	3	26	0	0	0

Subtract Minimum

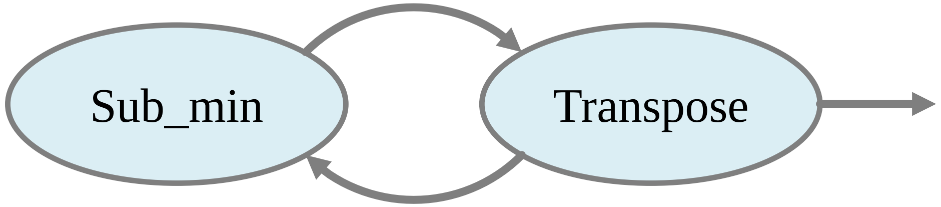


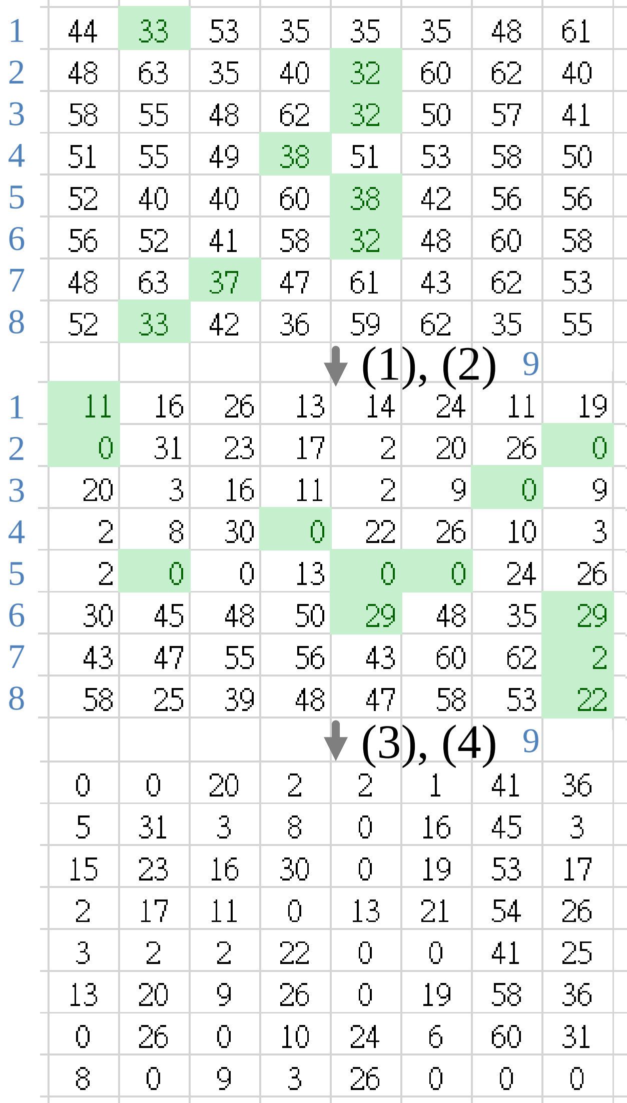


*This path seems long, so I pipeline it in practice.

Subtract Minimum (Simplified)

- Step break down
  - (1) 歷遍所有Row，求得各Row的最小值，並將各Row元素減去其最小值
  - (2) 歷遍所有Col，求得各Col的最小值，並將各Col元素減去其最小值
- Hardware Simplifying
  - (1) 歷遍所有Row，求得各Row的最小值，並將各Row元素減去其最小值
  - (2) Transpose
  - (3) 歷遍所有Row，求得各Row的最小值，並將各Row元素減去其最小值
  - (4) Transpose





1	44	33	53	35	35	35	48	61
2	48	63	35	40	32	60	62	40
3	58	55	48	62	32	50	57	41
4	51	55	49	38	51	53	58	50
5	52	40	40	60	38	42	56	56
6	56	52	41	58	32	48	60	58
7	48	63	37	47	61	43	62	53
8	52	33	42	36	59	62	35	55

↓ (1), (2) 9

1	11	16	26	13	14	24	11	19
2	0	31	23	17	2	20	26	0
3	20	3	16	11	2	9	0	9
4	2	8	30	0	22	26	10	3
5	2	0	0	13	0	0	24	26
6	30	45	48	50	29	48	35	29
7	43	47	55	56	43	60	62	2
8	58	25	39	48	47	58	53	22

↓ (3), (4) 9

0	0	20	2	2	1	41	36
5	31	3	8	0	16	45	3
15	23	16	30	0	19	53	17
2	17	11	0	13	21	54	26
3	2	2	22	0	0	41	25
13	20	9	26	0	19	58	36
0	26	0	10	24	6	60	31
8	0	9	3	26	0	0	0

Subtract Unmasked Minimum

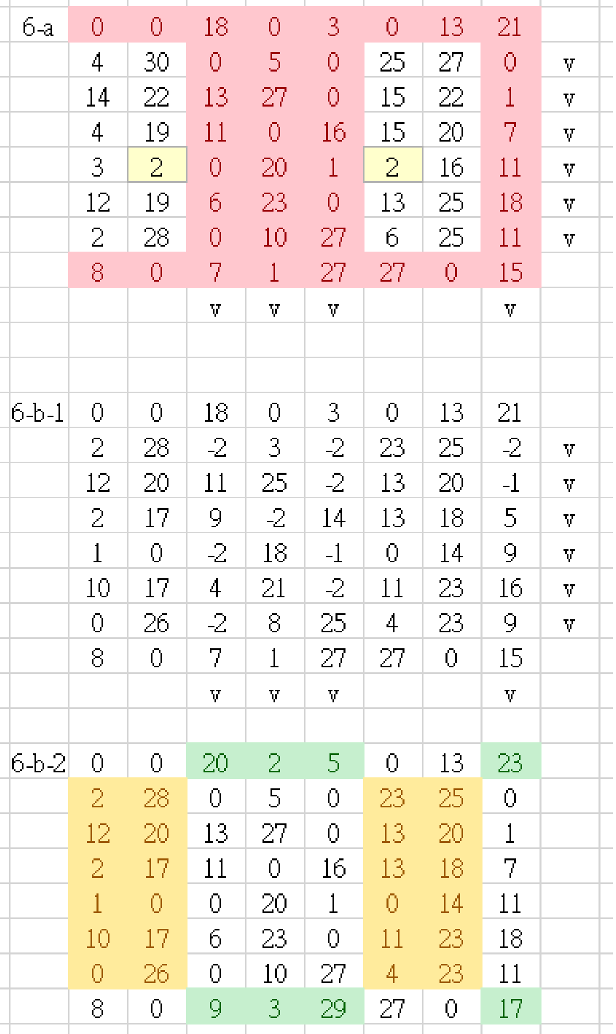
Subtract Minimum ①

Subtract Unmasked Minimum ⑥

These two steps are similar

- Step6:
  - (1) 找到沒被線覆蓋的最小值。
  - (2) 打勾的Row減掉最小值。
  - (3) 打勾的Col加上最小值。

  線沒覆蓋到的減掉最小值(黃)
  被水平、垂直線同時覆蓋的加上最小值(綠)
- Step 1:
  - 每個Row減掉該Row的最小值。
- Step 1 is the case that all elements are unmasked for Step6.



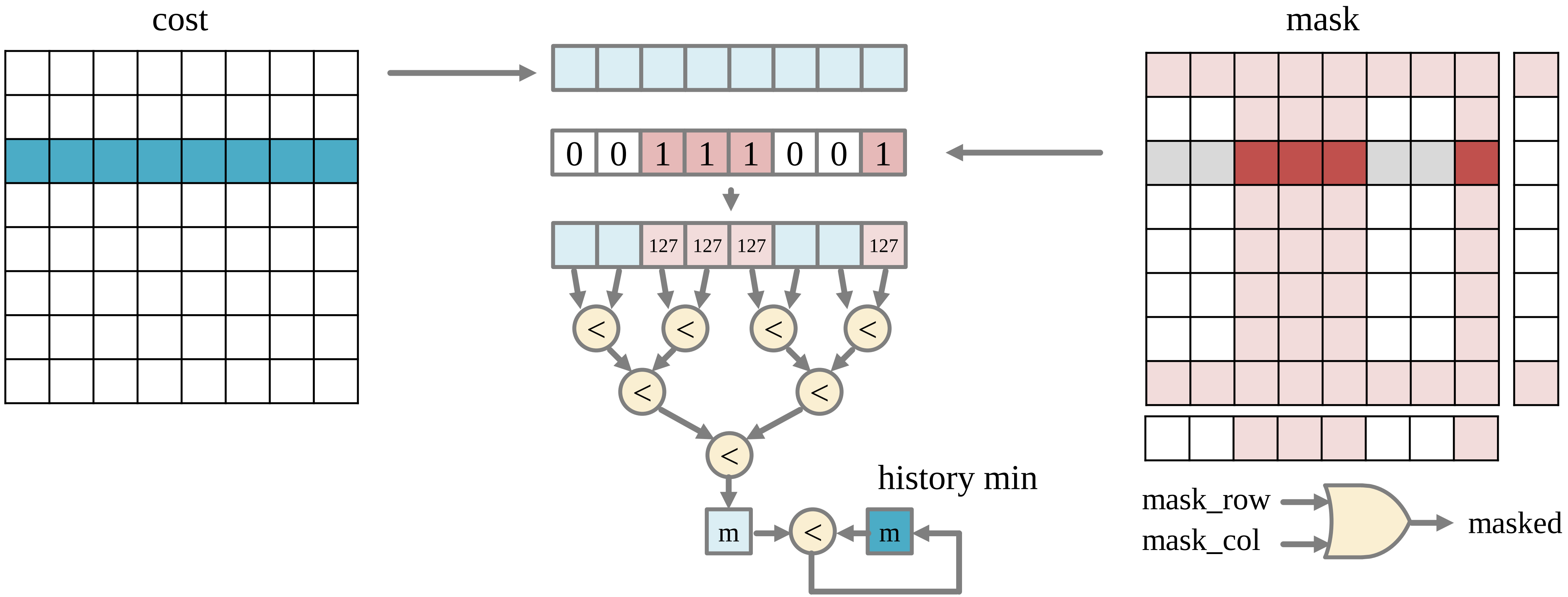
6-a	0	0	18	0	3	0	13	21	
	4	30	0	5	0	25	27	0	v
	14	22	13	27	0	15	22	1	v
	4	19	11	0	16	15	20	7	v
	3	2	0	20	1	2	16	11	v
	12	19	6	23	0	13	25	18	v
	2	28	0	10	27	6	25	11	v
	8	0	7	1	27	27	0	15	
			v	v	v			v	

6-b-1	0	0	18	0	3	0	13	21	
	2	28	-2	3	-2	23	25	-2	v
	12	20	11	25	-2	13	20	-1	v
	2	17	9	-2	14	13	18	5	v
	1	0	-2	18	-1	0	14	9	v
	10	17	4	21	-2	11	23	16	v
	0	26	-2	8	25	4	23	9	v
	8	0	7	1	27	27	0	15	
			v	v	v			v	

6-b-2	0	0	20	2	5	0	13	23
	2	28	0	5	0	23	25	0
	12	20	13	27	0	13	20	1
	2	17	11	0	16	13	18	7
	1	0	0	20	1	0	14	11
	10	17	6	23	0	11	23	18
	0	26	0	10	27	4	23	11
	8	0	9	3	29	27	0	17

Subtract Unmasked Minimum

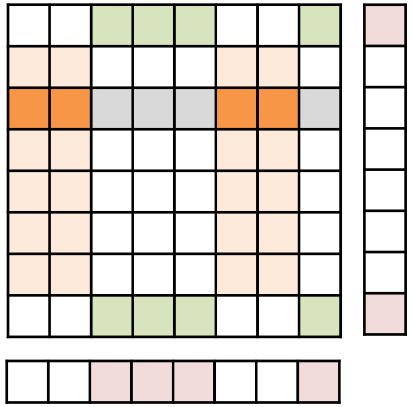
- Find unmasked minimum.
  - The same strategy as step 1, row-wise operation.
  - The masked number are regarded as 127. (maximum of 7-bit number)

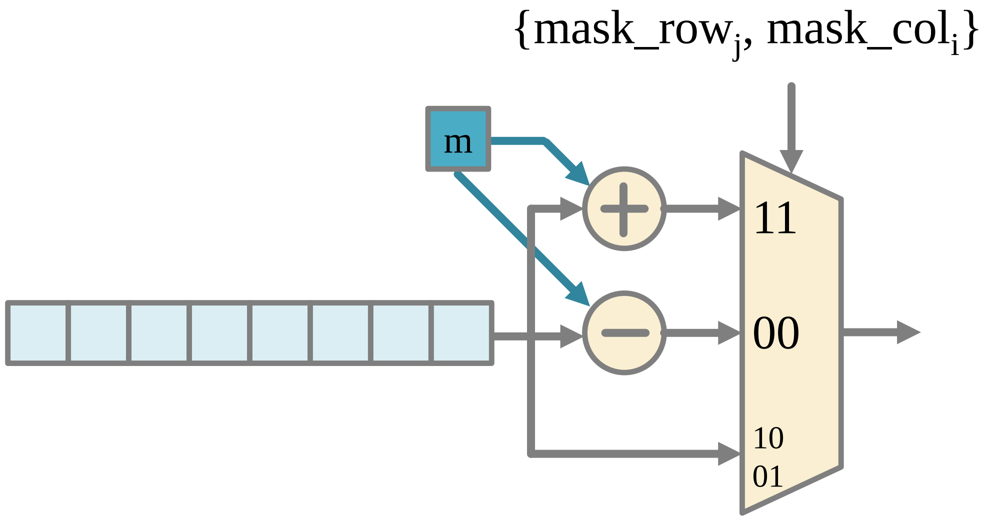


Subtract Unmasked Minimum

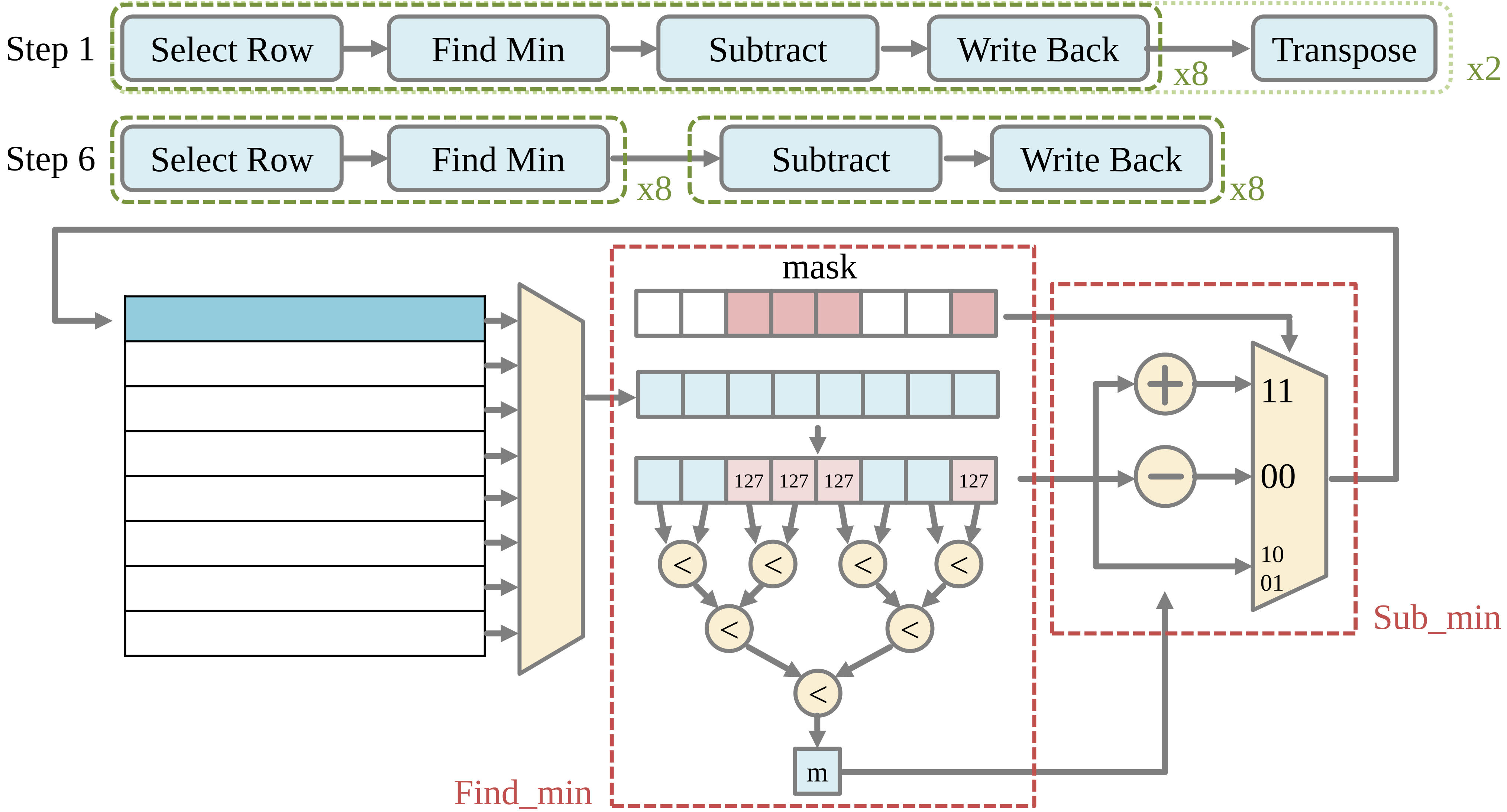
- Unmasked numbers subtract minimum.
- Double masked numbers add minimum.

		mask_row	
		0	1
mask_col	0	unmasked	masked
	1	masked	double masked



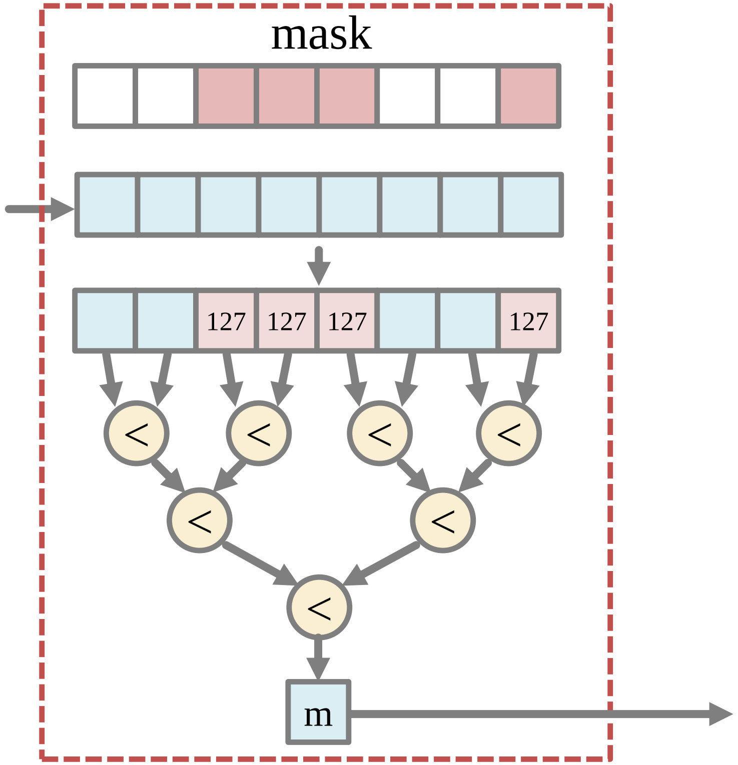


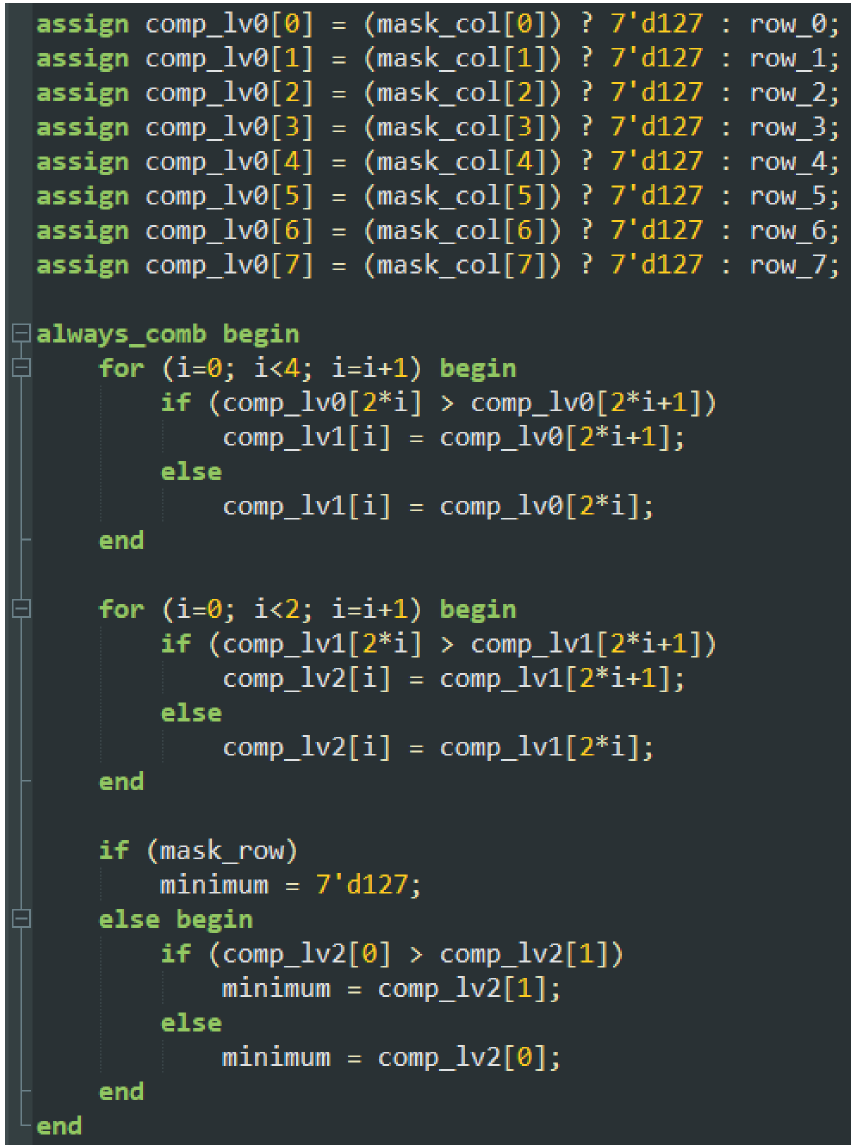
Subtract Unmasked Minimum



*This path seems long, so I pipeline it in practice.

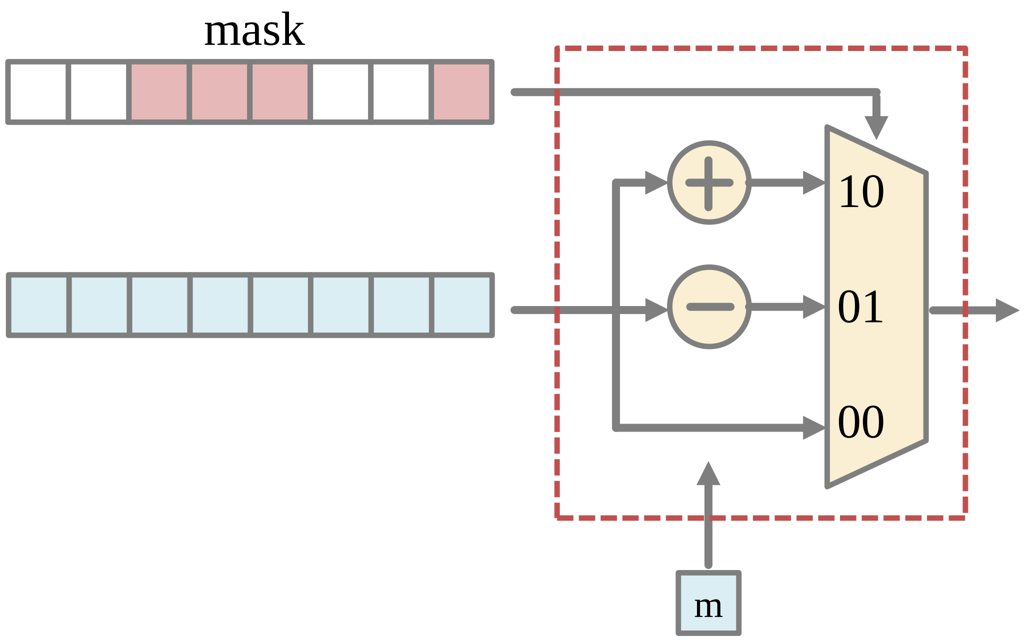
Find_min Submodule

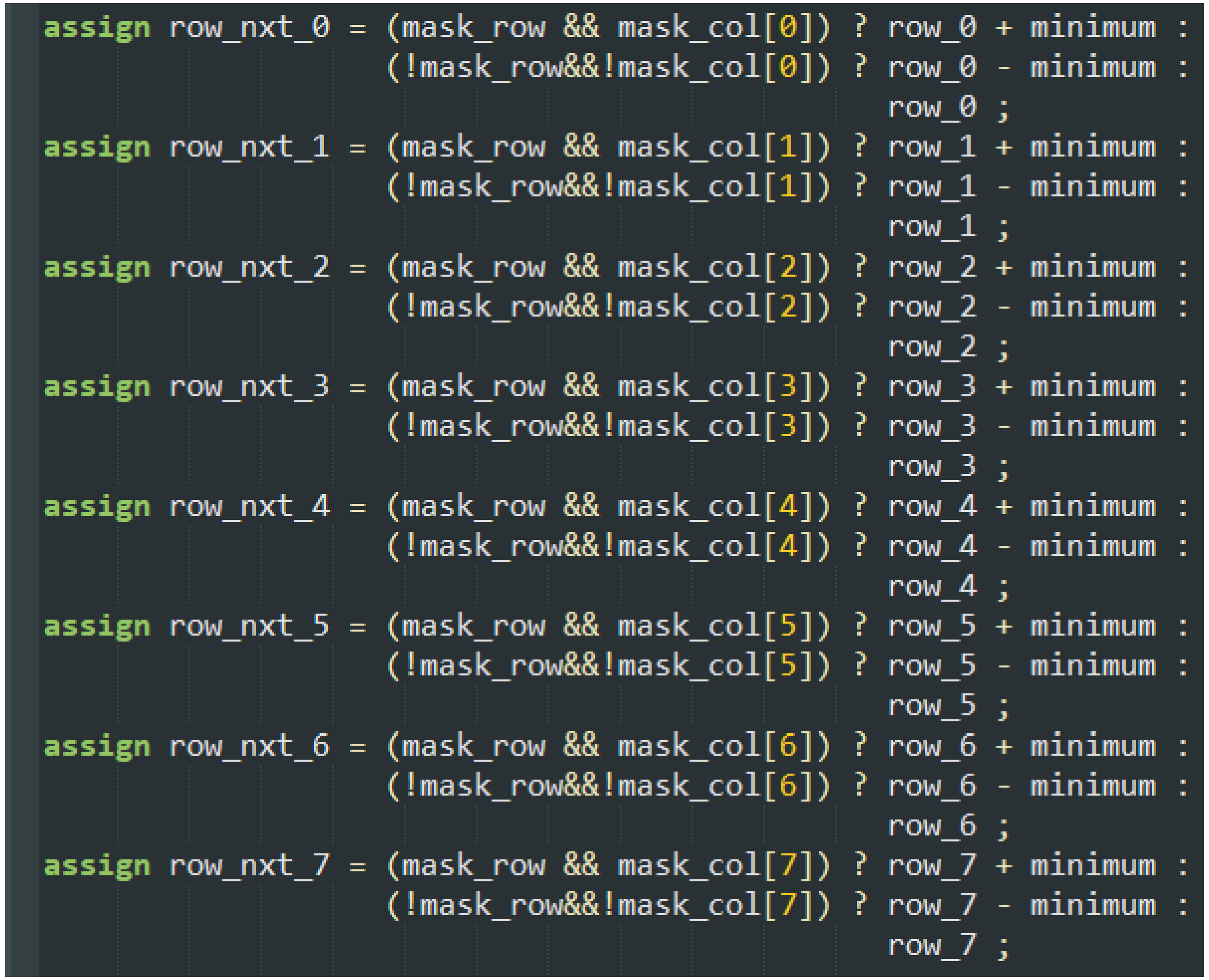




```
assign comp_lv0[0] = (mask_col[0]) ? 7'd127 : row_0;
assign comp_lv0[1] = (mask_col[1]) ? 7'd127 : row_1;
assign comp_lv0[2] = (mask_col[2]) ? 7'd127 : row_2;
assign comp_lv0[3] = (mask_col[3]) ? 7'd127 : row_3;
assign comp_lv0[4] = (mask_col[4]) ? 7'd127 : row_4;
assign comp_lv0[5] = (mask_col[5]) ? 7'd127 : row_5;
assign comp_lv0[6] = (mask_col[6]) ? 7'd127 : row_6;
assign comp_lv0[7] = (mask_col[7]) ? 7'd127 : row_7;

always_comb begin
    for (i=0; i<4; i=i+1) begin
        if (comp_lv0[2*i] > comp_lv0[2*i+1])
            comp_lv1[i] = comp_lv0[2*i+1];
        else
            comp_lv1[i] = comp_lv0[2*i];
    end

    for (i=0; i<2; i=i+1) begin
        if (comp_lv1[2*i] > comp_lv1[2*i+1])
            comp_lv2[i] = comp_lv1[2*i+1];
        else
            comp_lv2[i] = comp_lv1[2*i];
    end

    if (mask_row)
        minimum = 7'd127;
    else begin
        if (comp_lv2[0] > comp_lv2[1])
            minimum = comp_lv2[1];
        else
            minimum = comp_lv2[0];
    end
end
```

Sub_min Submodule

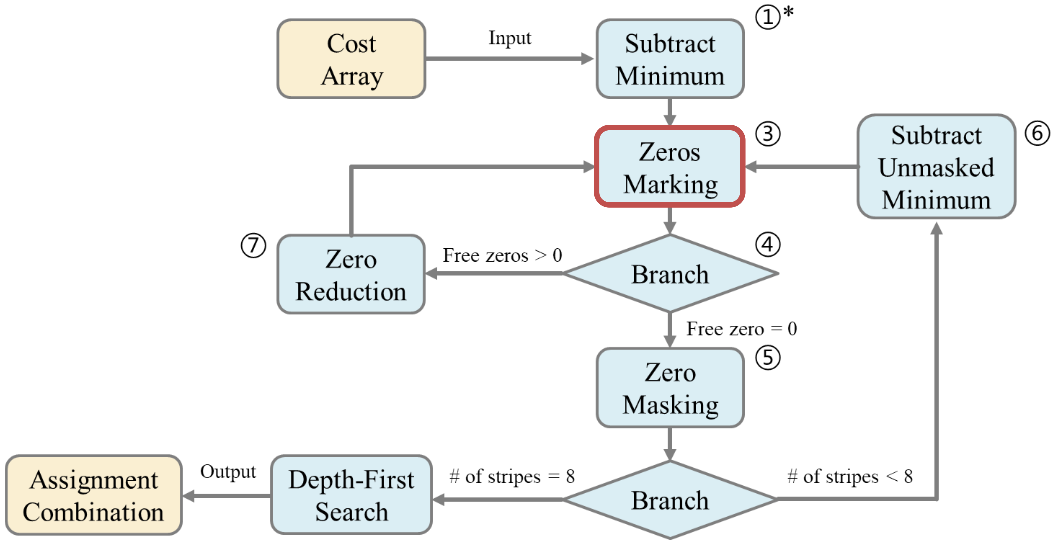




```
assign row_nxt_0 = (mask_row && mask_col[0]) ? row_0 + minimum :
                   (!mask_row&&!mask_col[0]) ? row_0 - minimum :
                                               row_0 ;
assign row_nxt_1 = (mask_row && mask_col[1]) ? row_1 + minimum :
                   (!mask_row&&!mask_col[1]) ? row_1 - minimum :
                                               row_1 ;
assign row_nxt_2 = (mask_row && mask_col[2]) ? row_2 + minimum :
                   (!mask_row&&!mask_col[2]) ? row_2 - minimum :
                                               row_2 ;
assign row_nxt_3 = (mask_row && mask_col[3]) ? row_3 + minimum :
                   (!mask_row&&!mask_col[3]) ? row_3 - minimum :
                                               row_3 ;
assign row_nxt_4 = (mask_row && mask_col[4]) ? row_4 + minimum :
                   (!mask_row&&!mask_col[4]) ? row_4 - minimum :
                                               row_4 ;
assign row_nxt_5 = (mask_row && mask_col[5]) ? row_5 + minimum :
                   (!mask_row&&!mask_col[5]) ? row_5 - minimum :
                                               row_5 ;
assign row_nxt_6 = (mask_row && mask_col[6]) ? row_6 + minimum :
                   (!mask_row&&!mask_col[6]) ? row_6 - minimum :
                                               row_6 ;
assign row_nxt_7 = (mask_row && mask_col[7]) ? row_7 + minimum :
                   (!mask_row&&!mask_col[7]) ? row_7 - minimum :
                                               row_7 ;
```

Zero Marking

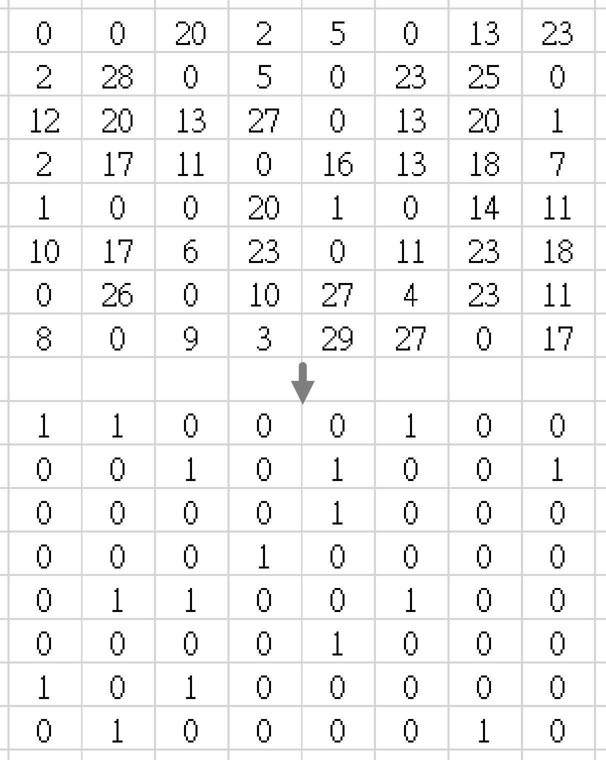
- Step break down
  - (1) 歷遍所有row，找到只含一個Free-zero的Row，該Zero畫圈
    - 此Circled-zero所在的col的其他Zero畫撇。
  - (2) 歷遍所有col，找到只含一個Free-zero的col ，該Zero畫圈
    - 此Circled-zero所在的row的其他Zero畫撇。
- Hardware Simplifying
  - (1) 歷遍所有row，找到只含一個Free-zero的Row，該Zero畫圈
    - 此Circled-zero所在的col的其他Zero畫撇。
  - (2) Transpose



Zero Marking

- Non-zero = 0
- Free-zero = 1
- Circled-zero = 2
- Striped-zero = 3

- 7 bits cost → 2 bits zero mark
  - Smaller area & Shorter time delay

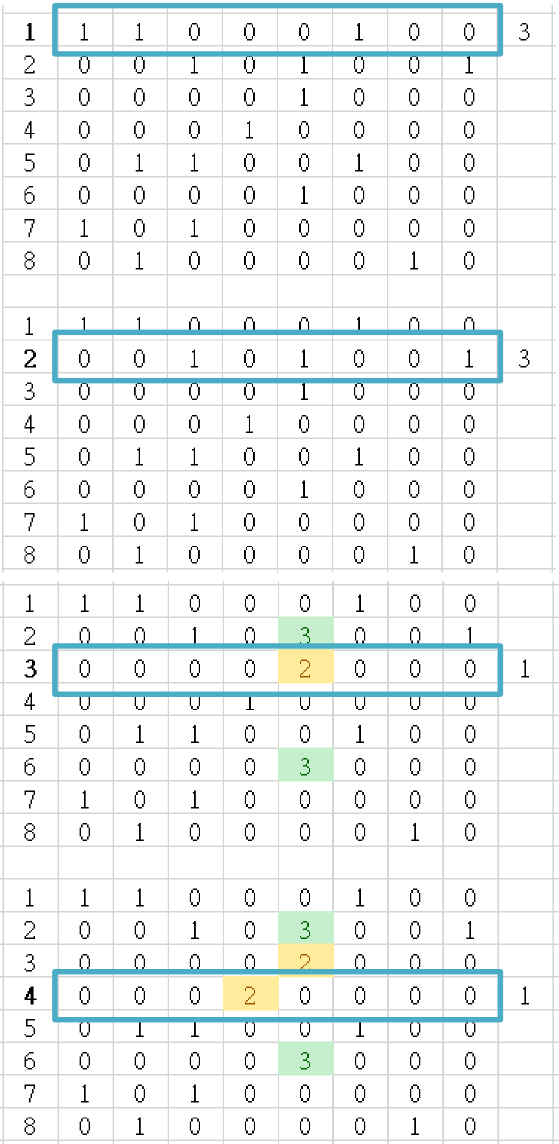


0	0	20	2	5	0	13	23
2	28	0	5	0	23	25	0
12	20	13	27	0	13	20	1
2	17	11	0	16	13	18	7
1	0	0	20	1	0	14	11
10	17	6	23	0	11	23	18
0	26	0	10	27	4	23	11
8	0	9	3	29	27	0	17

↓

1	1	0	0	0	1	0	0
0	0	1	0	1	0	0	1
0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0
0	1	1	0	0	1	0	0
0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0
0	1	0	0	0	0	1	0

Zero Marking



1	1	1	0	0	0	1	0	0	3
2	0	0	1	0	1	0	0	1	
3	0	0	0	0	1	0	0	0	
4	0	0	0	1	0	0	0	0	
5	0	1	1	0	0	1	0	0	
6	0	0	0	0	1	0	0	0	
7	1	0	1	0	0	0	0	0	
8	0	1	0	0	0	0	1	0	

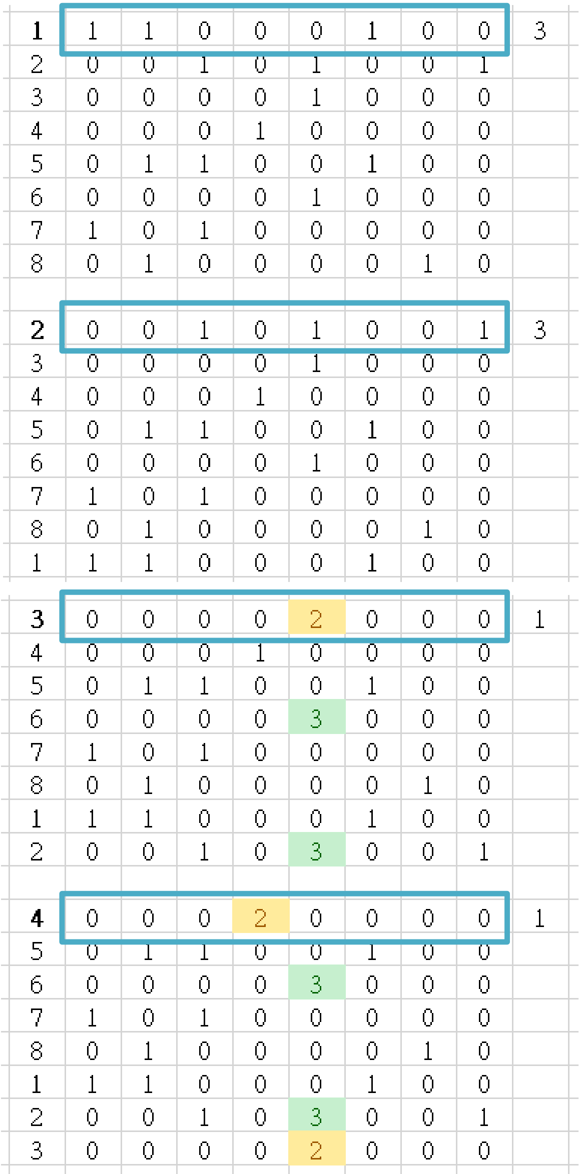
1	1	1	0	0	0	1	0	0	
2	0	0	1	0	1	0	0	1	3
3	0	0	0	0	1	0	0	0	
4	0	0	0	1	0	0	0	0	
5	0	1	1	0	0	1	0	0	
6	0	0	0	0	1	0	0	0	
7	1	0	1	0	0	0	0	0	
8	0	1	0	0	0	0	1	0	

1	1	1	0	0	0	1	0	0	
2	0	0	1	0	3	0	0	1	
3	0	0	0	0	2	0	0	0	1
4	0	0	0	1	0	0	0	0	
5	0	1	1	0	0	1	0	0	
6	0	0	0	0	3	0	0	0	
7	1	0	1	0	0	0	0	0	
8	0	1	0	0	0	0	1	0	

1	1	1	0	0	0	1	0	0	
2	0	0	1	0	3	0	0	1	
3	0	0	0	0	2	0	0	0	
4	0	0	0	2	0	0	0	0	1
5	0	1	1	0	0	1	0	0	
6	0	0	0	0	3	0	0	0	
7	1	0	1	0	0	0	0	0	
8	0	1	0	0	0	0	1	0	

⋮

Simplify the hardware with **shift register.** MUXs are reduced.



1	1	1	0	0	0	1	0	0	3
2	0	0	1	0	1	0	0	1	
3	0	0	0	0	1	0	0	0	
4	0	0	0	1	0	0	0	0	
5	0	1	1	0	0	1	0	0	
6	0	0	0	0	1	0	0	0	
7	1	0	1	0	0	0	0	0	
8	0	1	0	0	0	0	1	0	

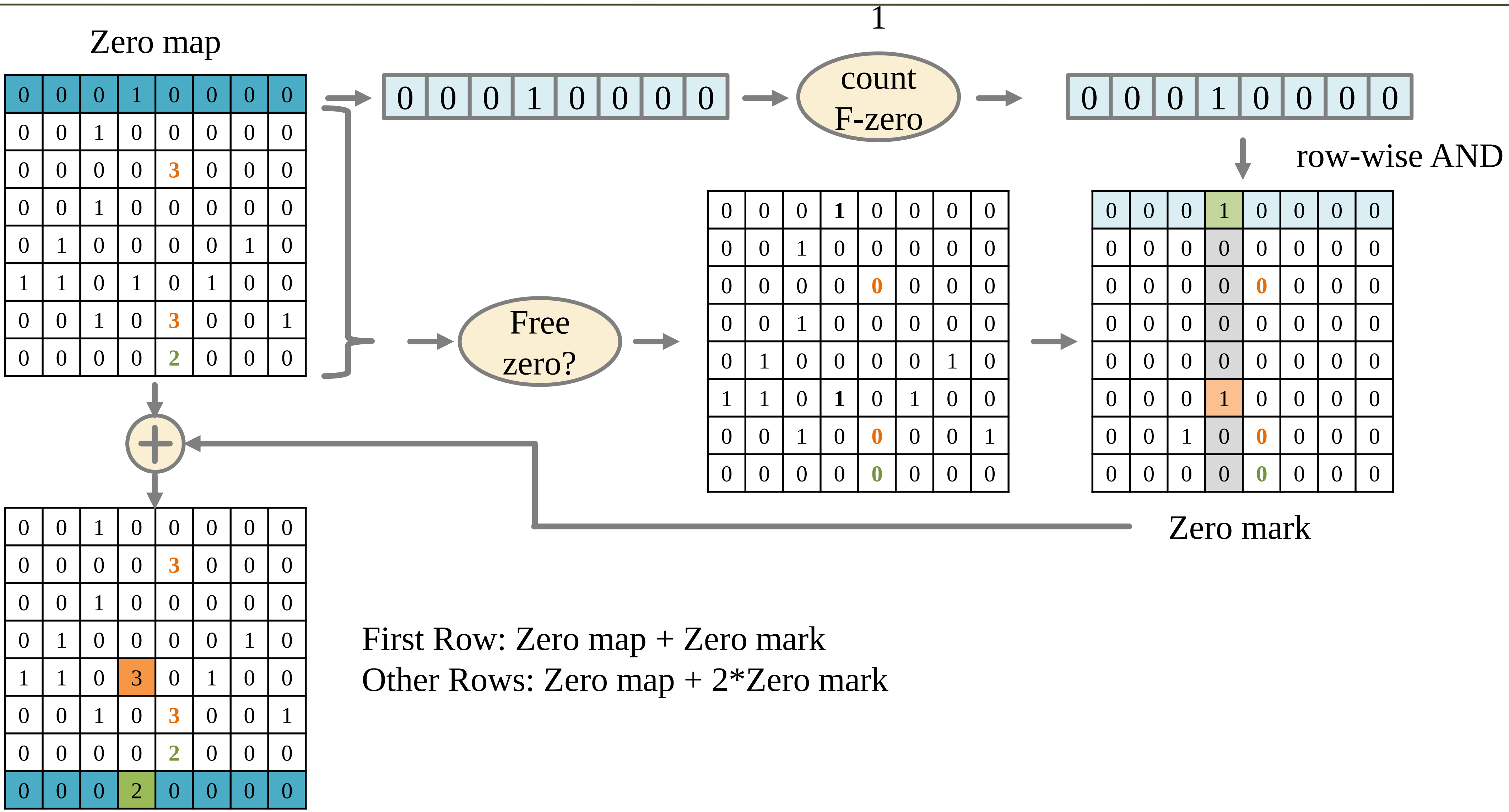
2	0	0	1	0	1	0	0	1	3
3	0	0	0	0	1	0	0	0	
4	0	0	0	1	0	0	0	0	
5	0	1	1	0	0	1	0	0	
6	0	0	0	0	1	0	0	0	
7	1	0	1	0	0	0	0	0	
8	0	1	0	0	0	0	1	0	
1	1	1	0	0	0	1	0	0	

3	0	0	0	0	2	0	0	0	1
4	0	0	0	1	0	0	0	0	
5	0	1	1	0	0	1	0	0	
6	0	0	0	0	3	0	0	0	
7	1	0	1	0	0	0	0	0	
8	0	1	0	0	0	0	1	0	
1	1	1	0	0	0	1	0	0	
2	0	0	1	0	3	0	0	1	

4	0	0	0	2	0	0	0	0	1
5	0	1	1	0	0	1	0	0	
6	0	0	0	0	3	0	0	0	
7	1	0	1	0	0	0	0	0	
8	0	1	0	0	0	0	1	0	
1	1	1	0	0	0	1	0	0	
2	0	0	1	0	3	0	0	1	
3	0	0	0	0	2	0	0	0	

⋮

Zero Marking



Zero map

0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	0	3	0	0	0
0	0	1	0	0	0	0	0
0	1	0	0	0	0	1	0
1	1	0	1	0	1	0	0
0	0	1	0	3	0	0	1
0	0	0	0	2	0	0	0

0	0	0	1	0	0	0	0

count F-zero

0	0	0	1	0	0	0	0

row-wise AND

Free zero?

0	0	0	**1**	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	1	0	0	0	0	1	0
1	1	0	**1**	0	1	0	0
0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0

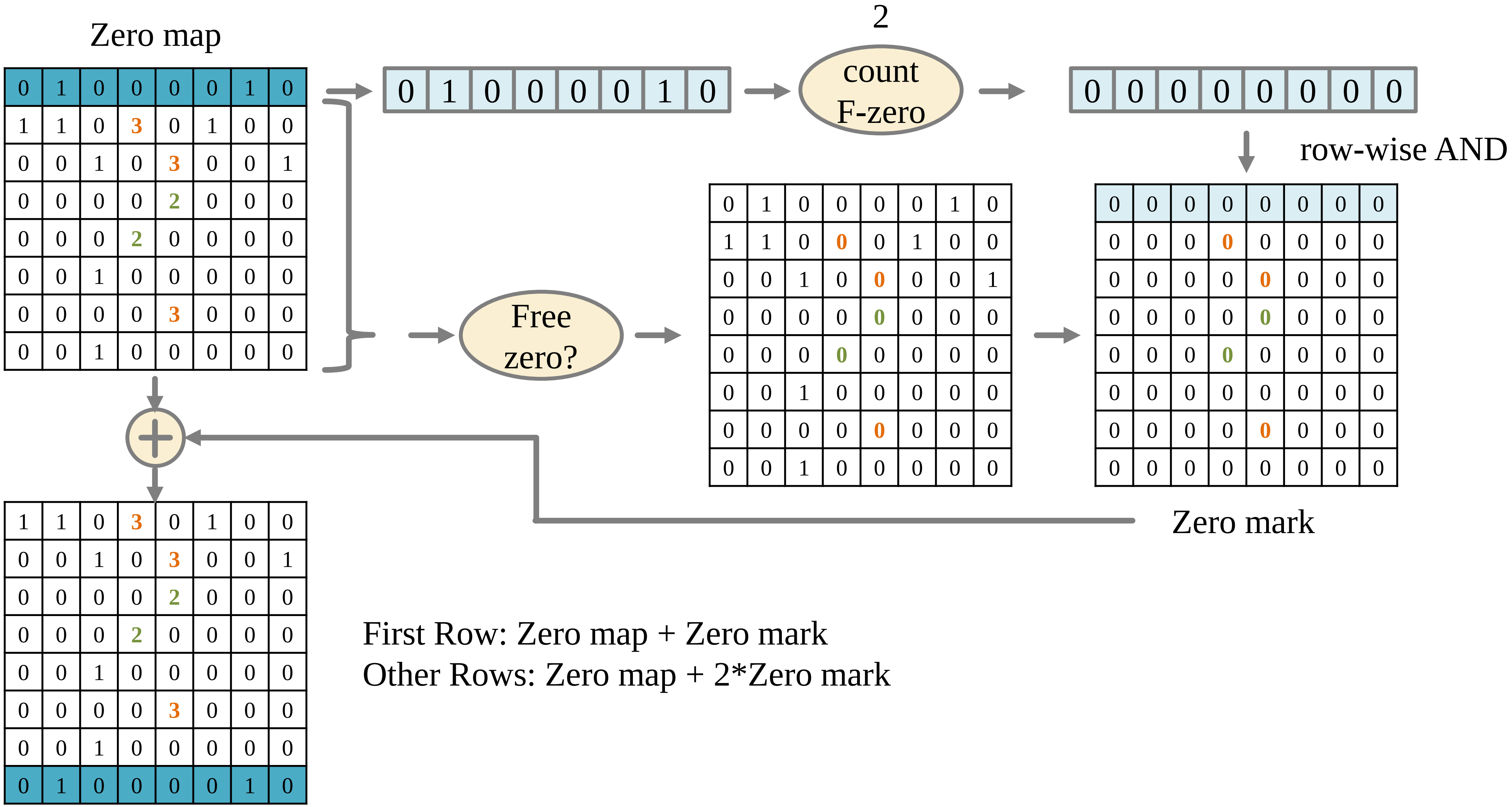
Zero mark

0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0

0	0	1	0	0	0	0	0
0	0	0	0	3	0	0	0
0	0	1	0	0	0	0	0
0	1	0	0	0	0	1	0
1	1	0	3	0	1	0	0
0	0	1	0	3	0	0	1
0	0	0	0	2	0	0	0
0	0	0	2	0	0	0	0

First Row: Zero map + Zero mark
Other Rows: Zero map + 2*Zero mark

Zero Marking



Zero map

0	1	0	0	0	0	1	0
1	1	0	3	0	1	0	0
0	0	1	0	3	0	0	1
0	0	0	0	2	0	0	0
0	0	0	2	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	0	3	0	0	0
0	0	1	0	0	0	0	0

0	1	0	0	0	0	1	0

0	0	0	0	0	0	0	0

0	1	0	0	0	0	1	0
1	1	0	0	0	1	0	0
0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0

Zero mark

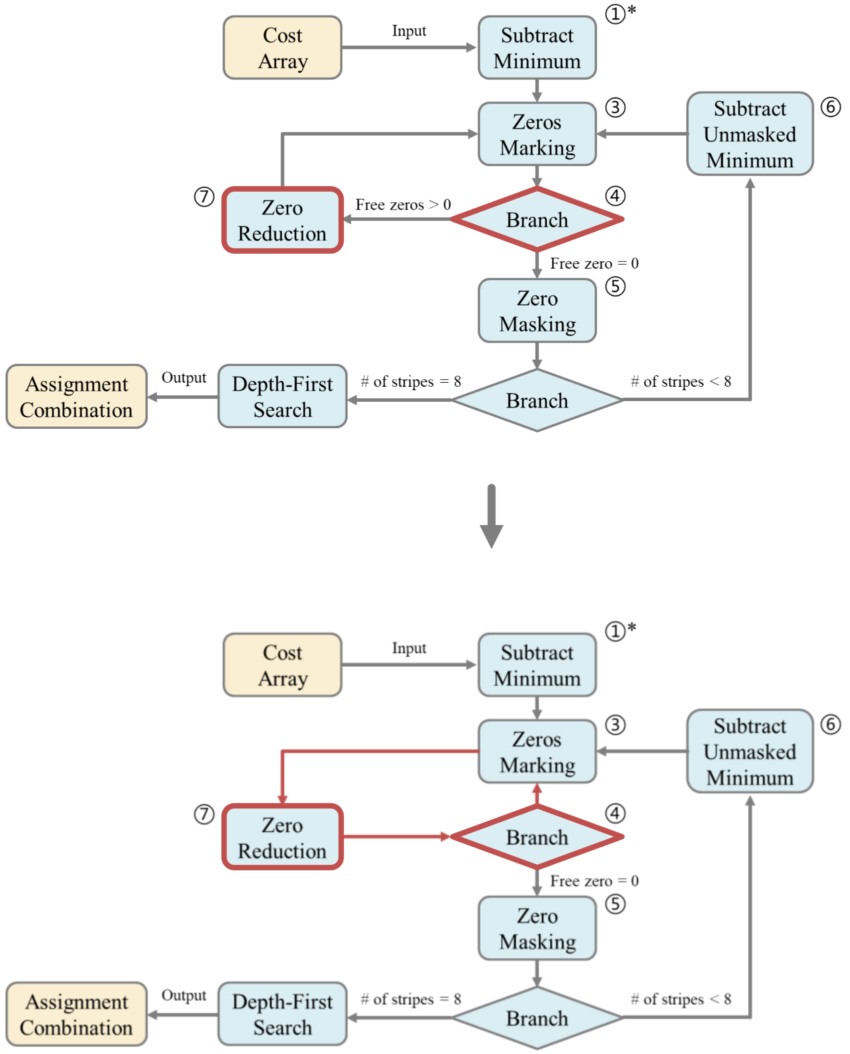
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

1	1	0	3	0	1	0	0
0	0	1	0	3	0	0	1
0	0	0	0	2	0	0	0
0	0	0	2	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	0	3	0	0	0
0	0	1	0	0	0	0	0
0	1	0	0	0	0	1	0

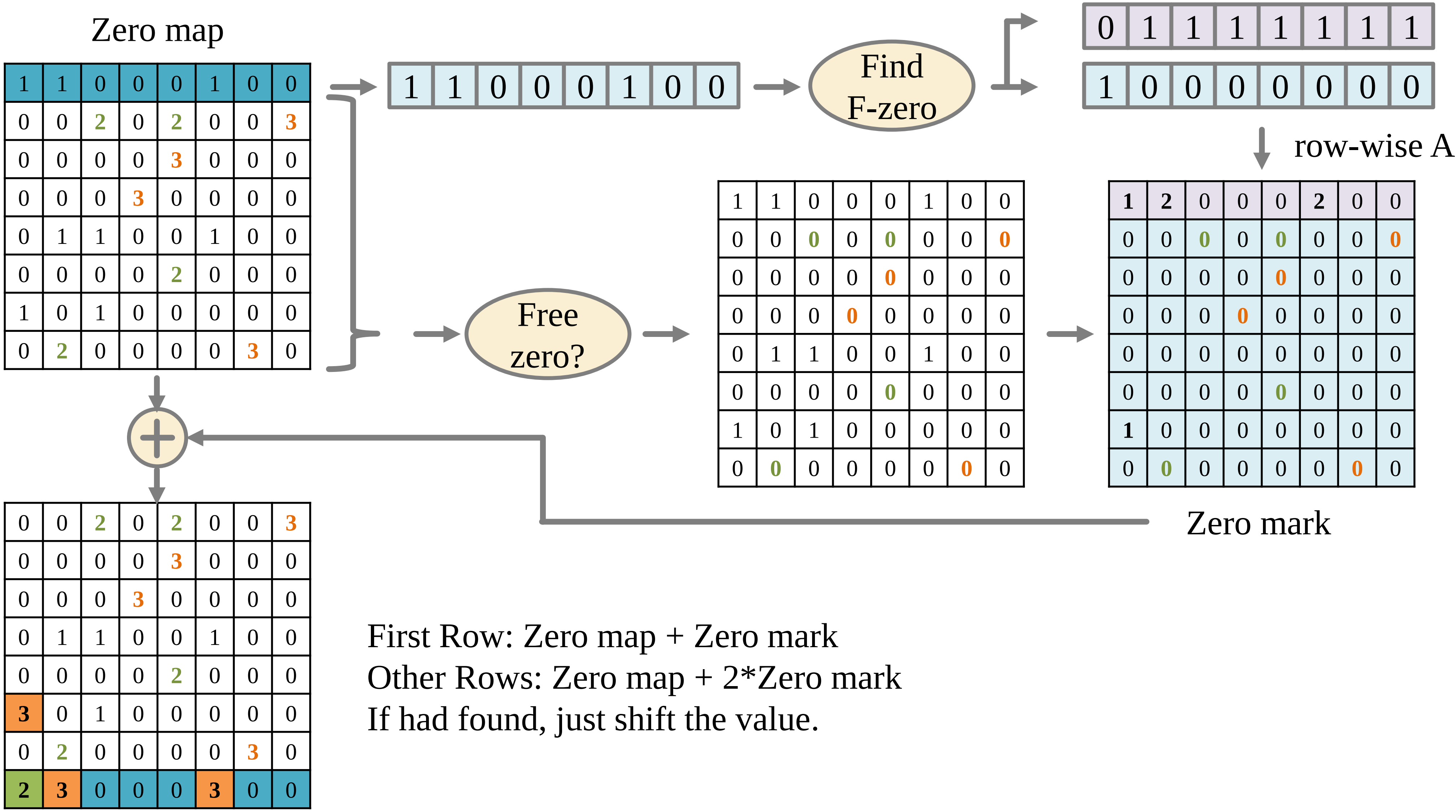
First Row: Zero map + Zero mark
Other Rows: Zero map + 2*Zero mark

Zero Reduction

- Step break down
  - 如果還存在Free-zero，
  - 取第一個還存在Free-zero的Row (or 最少Free-zero的Row)
    - 將該Row第一個Free-zero打圈
    - 將該Free-zero所在的Row/Col上的其他Free-zero都畫撇
  - 回去做Zero Marking
- Hardware Simplifying
  - ~~如果還存在Free-zero，~~
  - 從第一個Row開始往下找Free-zero 硬體比較複雜
  - 取第一個還存在Free-zero的Row ~~(or 最少Free-zero的Row)~~
    - 將該Row第一個Free-zero打圈
    - 將該Free-zero所在的Row/Col上的其他Free-zero都畫撇
  - 找到Free-zero回去做Zero Marking
  - 8個Row找完沒找到Free-zero就去畫線



Zero Reduction



Zero map

1	1	0	0	0	1	0	0
0	0	2	0	2	0	0	3
0	0	0	0	3	0	0	0
0	0	0	3	0	0	0	0
0	1	1	0	0	1	0	0
0	0	0	0	2	0	0	0
1	0	1	0	0	0	0	0
0	2	0	0	0	0	3	0

1	1	0	0	0	1	0	0

Find F-zero

0	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0

row-wise AND

Free zero?

1	1	0	0	0	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1	1	0	0	1	0	0
0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0

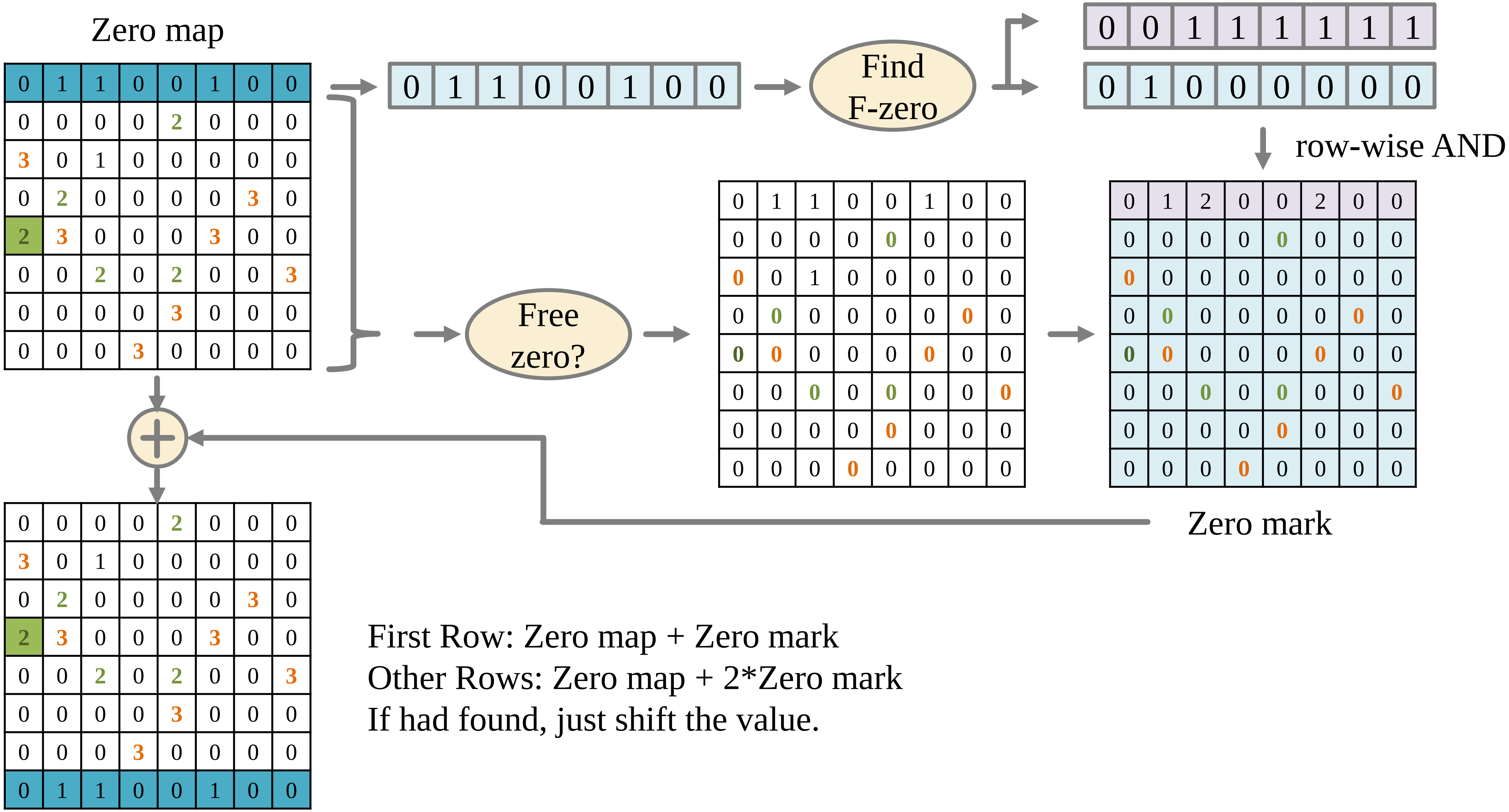
1	**2**	0	0	0	**2**	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Zero mark

0	0	2	0	2	0	0	3
0	0	0	0	3	0	0	0
0	0	0	3	0	0	0	0
0	1	1	0	0	1	0	0
0	0	0	0	2	0	0	0
3	0	1	0	0	0	0	0
0	2	0	0	0	0	3	0
2	**3**	0	0	0	**3**	0	0

First Row: Zero map + Zero mark
Other Rows: Zero map + 2*Zero mark
If had found, just shift the value.

Zero Reduction



Zero map

0	1	1	0	0	1	0	0
0	0	0	0	2	0	0	0
3	0	1	0	0	0	0	0
0	2	0	0	0	0	3	0
2	3	0	0	0	3	0	0
0	0	2	0	2	0	0	3
0	0	0	0	3	0	0	0
0	0	0	3	0	0	0	0

0	1	1	0	0	1	0	0

0	0	1	1	1	1	1	1

0	1	0	0	0	0	0	0

0	1	1	0	0	1	0	0
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

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

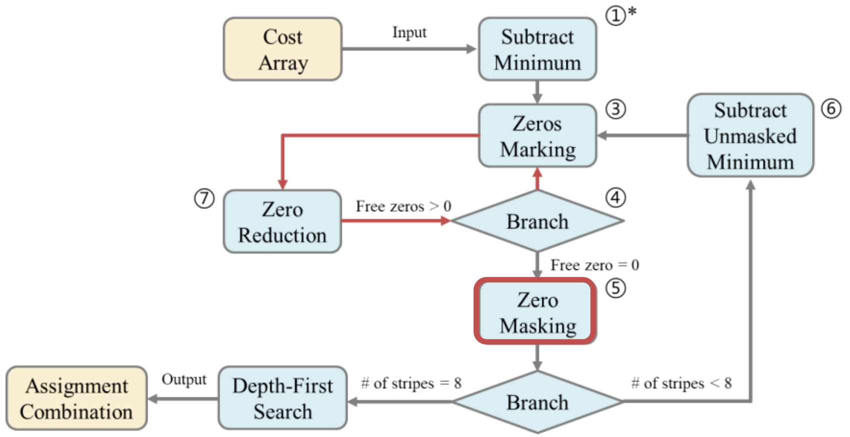
Zero mark

0	0	0	0	2	0	0	0
3	0	1	0	0	0	0	0
0	2	0	0	0	0	3	0
2	3	0	0	0	3	0	0
0	0	2	0	2	0	0	3
0	0	0	0	3	0	0	0
0	0	0	3	0	0	0	0
0	1	1	0	0	1	0	0

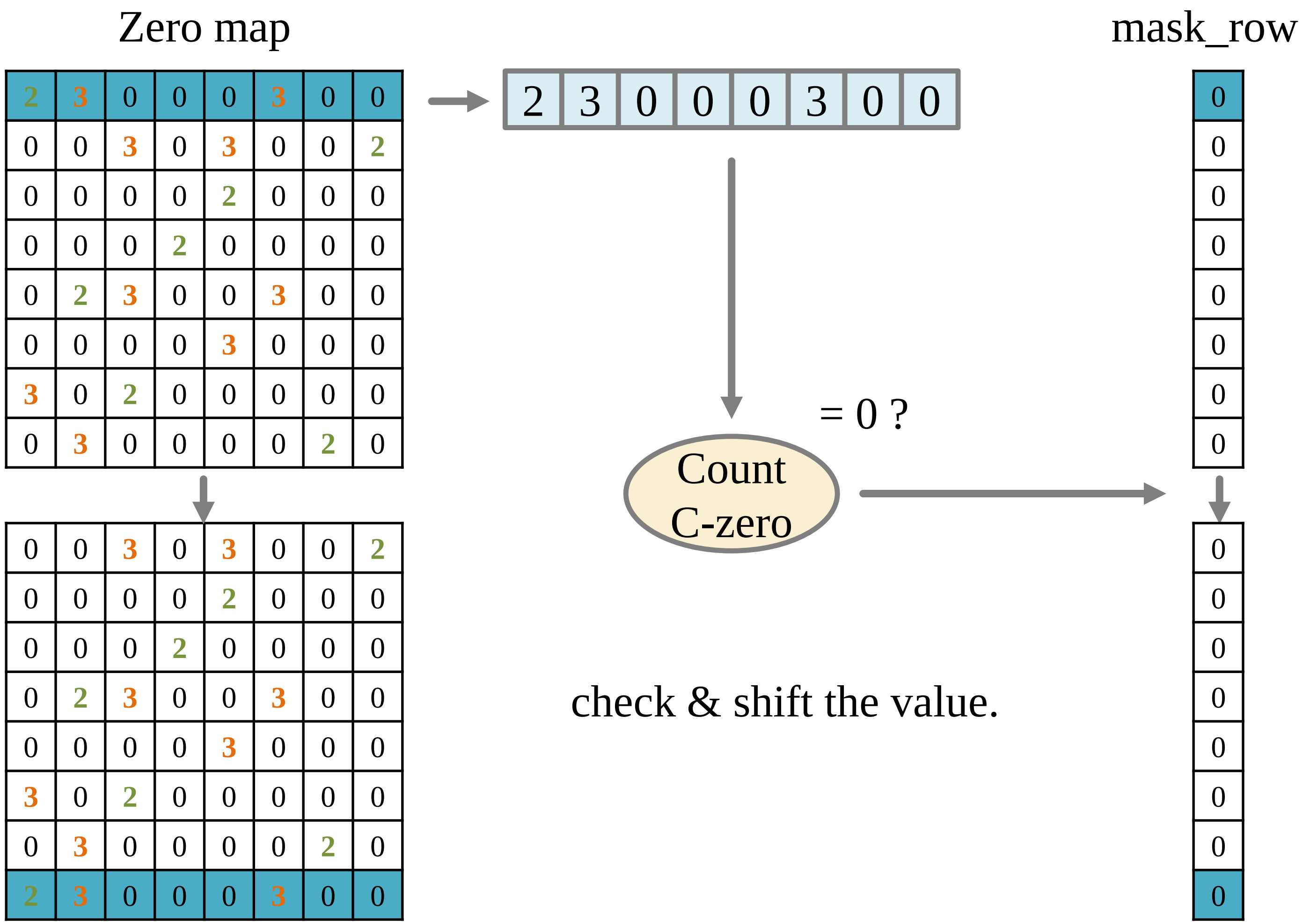
First Row: Zero map + Zero mark
Other Rows: Zero map + 2*Zero mark
If had found, just shift the value.

Zero Masking

- Step Break down
  - (1) 對沒有畫圈零$_2$的Row打勾
  - (2) 對打勾的Row上有畫撇零$_3$的Col打勾
  - (3) 對打勾的Col上有畫圈零$_2$的Row打勾
  - 重複(2)(3)直到沒辦法打勾
- Hardware Simplifying
  - (1) 對沒有畫圈零$_2$的Row打勾
  - (2) 對打勾的Row上有**畫撇零**$_3$的Col打勾
  - (3) Transpose
  - (4) 對打勾的Row上有**畫圈零**$_2$的Col打勾
  - (5) Transpose
  - 重複(2)~(5)直到沒辦法打勾



Zero Masking -1



Zero map

2	3	0	0	0	3	0	0
0	0	3	0	3	0	0	2
0	0	0	0	2	0	0	0
0	0	0	2	0	0	0	0
0	2	3	0	0	3	0	0
0	0	0	0	3	0	0	0
3	0	2	0	0	0	0	0
0	3	0	0	0	0	2	0

2	3	0	0	0	3	0	0

= 0 ?

Count C-zero

check & shift the value.

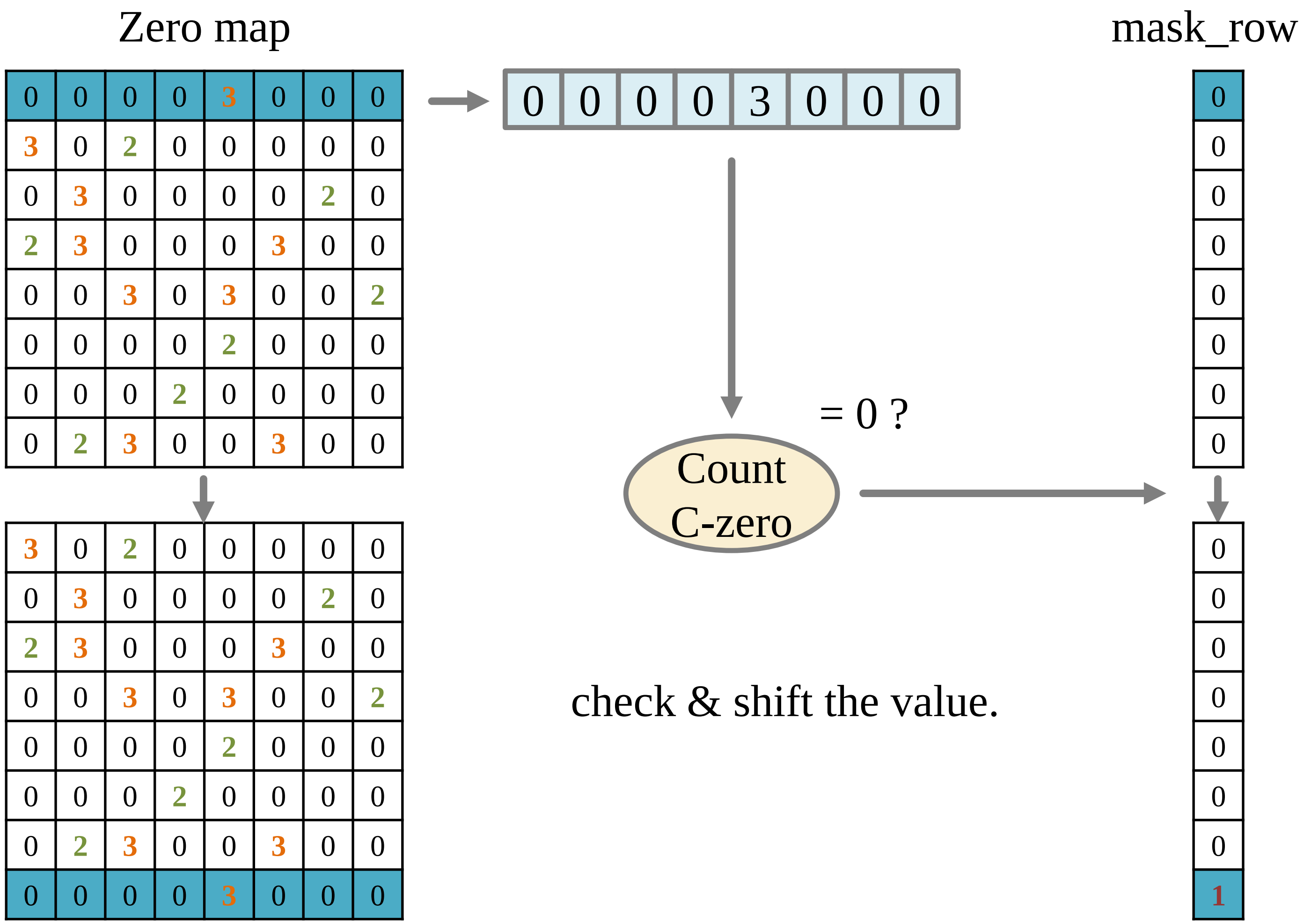
0	0	3	0	3	0	0	2
0	0	0	0	2	0	0	0
0	0	0	2	0	0	0	0
0	2	3	0	0	3	0	0
0	0	0	0	3	0	0	0
3	0	2	0	0	0	0	0
0	3	0	0	0	0	2	0
2	3	0	0	0	3	0	0

mask_row

0
0
0
0
0
0
0
0

0
0
0
0
0
0
0
0

Zero Masking -1



Zero map

0	0	0	0	3	0	0	0
3	0	2	0	0	0	0	0
0	3	0	0	0	0	2	0
2	3	0	0	0	3	0	0
0	0	3	0	3	0	0	2
0	0	0	0	2	0	0	0
0	0	0	2	0	0	0	0
0	2	3	0	0	3	0	0

0	0	0	0	3	0	0	0

3	0	2	0	0	0	0	0
0	3	0	0	0	0	2	0
2	3	0	0	0	3	0	0
0	0	3	0	3	0	0	2
0	0	0	0	2	0	0	0
0	0	0	2	0	0	0	0
0	2	3	0	0	3	0	0
0	0	0	0	3	0	0	0

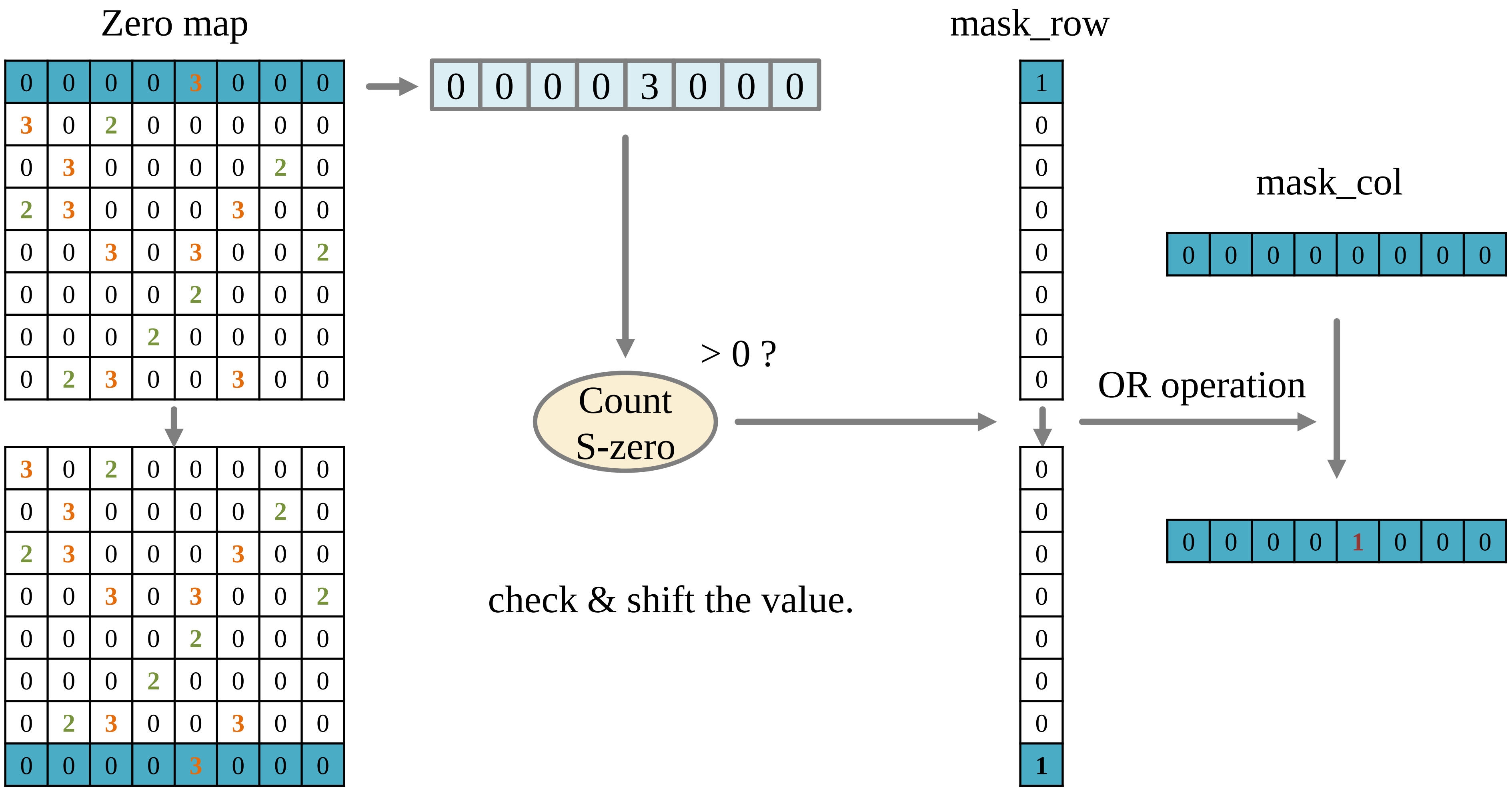
mask_row

0
0
0
0
0
0
0
0

0
0
0
0
0
0
0
1

check & shift the value.

Zero Masking -2



Zero map

0	0	0	0	3	0	0	0
3	0	2	0	0	0	0	0
0	3	0	0	0	0	2	0
2	3	0	0	0	3	0	0
0	0	3	0	3	0	0	2
0	0	0	0	2	0	0	0
0	0	0	2	0	0	0	0
0	2	3	0	0	3	0	0

3	0	2	0	0	0	0	0
0	3	0	0	0	0	2	0
2	3	0	0	0	3	0	0
0	0	3	0	3	0	0	2
0	0	0	0	2	0	0	0
0	0	0	2	0	0	0	0
0	2	3	0	0	3	0	0
0	0	0	0	3	0	0	0

0	0	0	0	3	0	0	0

Count S-zero

> 0 ?

check & shift the value.

mask_row

1	0	0	0	0	0	0	0

0	0	0	0	0	0	0	1

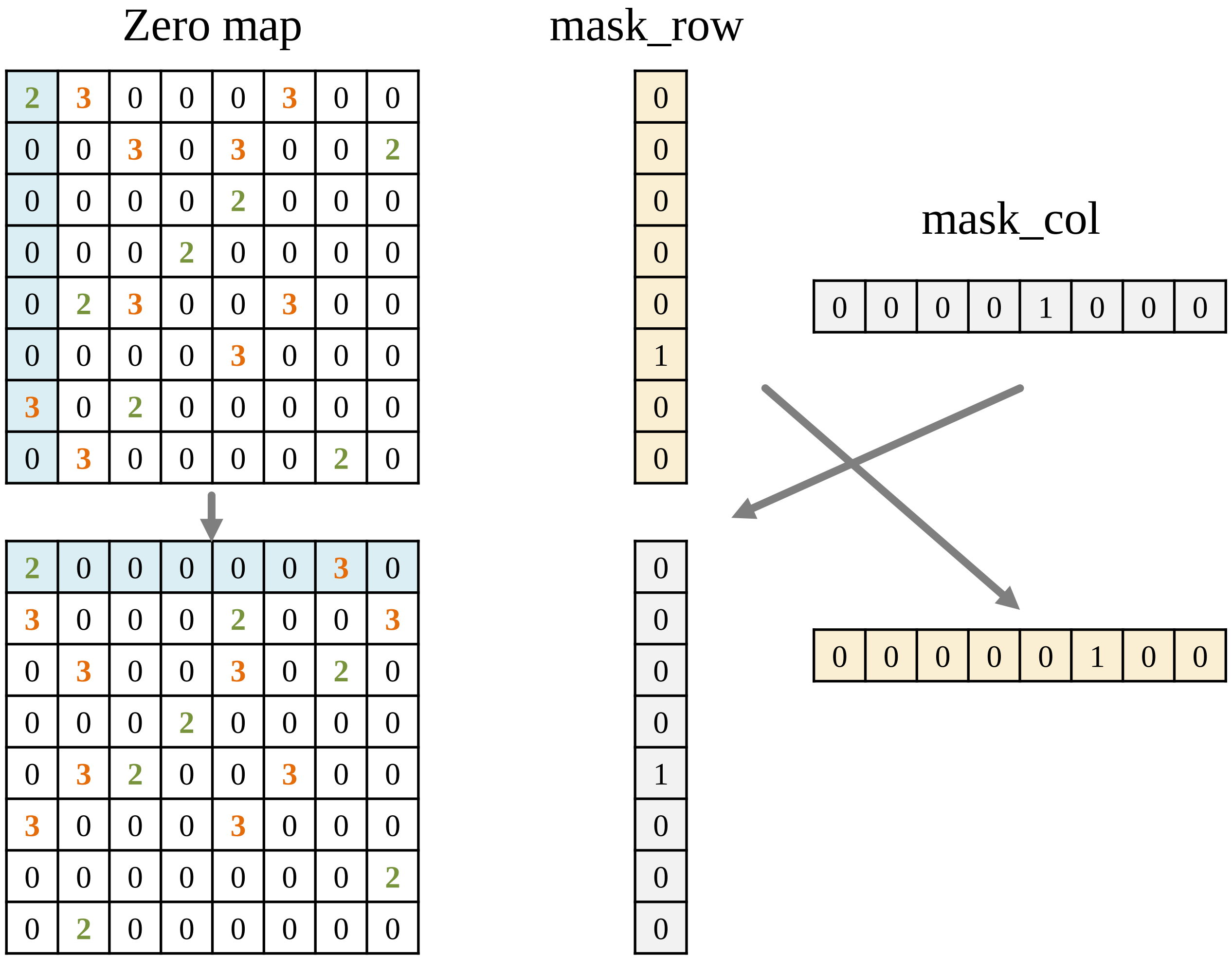
OR operation

mask_col

0	0	0	0	0	0	0	0

0	0	0	0	1	0	0	0

Zero Masking -3,5 (transpose)



Zero map

2	3	0	0	0	3	0	0
0	0	3	0	3	0	0	2
0	0	0	0	2	0	0	0
0	0	0	2	0	0	0	0
0	2	3	0	0	3	0	0
0	0	0	0	3	0	0	0
3	0	2	0	0	0	0	0
0	3	0	0	0	0	2	0

mask_row

0
0
0
0
0
1
0
0

mask_col

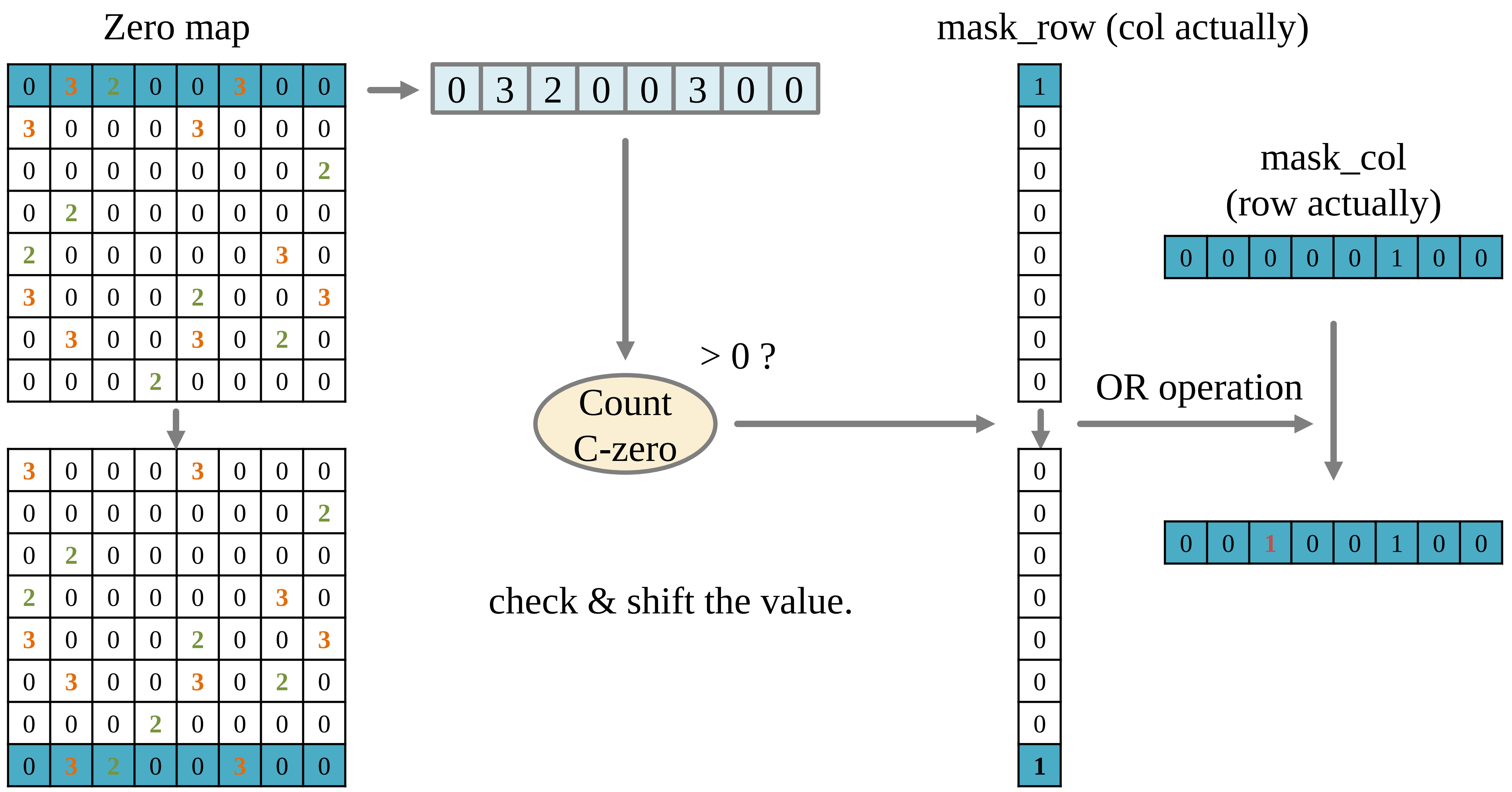
0	0	0	0	1	0	0	0

2	0	0	0	0	0	3	0
3	0	0	0	2	0	0	3
0	3	0	0	3	0	2	0
0	0	0	2	0	0	0	0
0	3	2	0	0	3	0	0
3	0	0	0	3	0	0	0
0	0	0	0	0	0	0	2
0	2	0	0	0	0	0	0

0
0
0
0
1
0
0
0

0	0	0	0	0	1	0	0

Zero Masking -4



Zero map

0	3	2	0	0	3	0	0
3	0	0	0	3	0	0	0
0	0	0	0	0	0	0	2
0	2	0	0	0	0	0	0
2	0	0	0	0	0	3	0
3	0	0	0	2	0	0	3
0	3	0	0	3	0	2	0
0	0	0	2	0	0	0	0

3	0	0	0	3	0	0	0
0	0	0	0	0	0	0	2
0	2	0	0	0	0	0	0
2	0	0	0	0	0	3	0
3	0	0	0	2	0	0	3
0	3	0	0	3	0	2	0
0	0	0	2	0	0	0	0
0	3	2	0	0	3	0	0

0	3	2	0	0	3	0	0

mask_row (col actually)

1
0
0
0
0
0
0
0

0
0
0
0
0
0
0
1

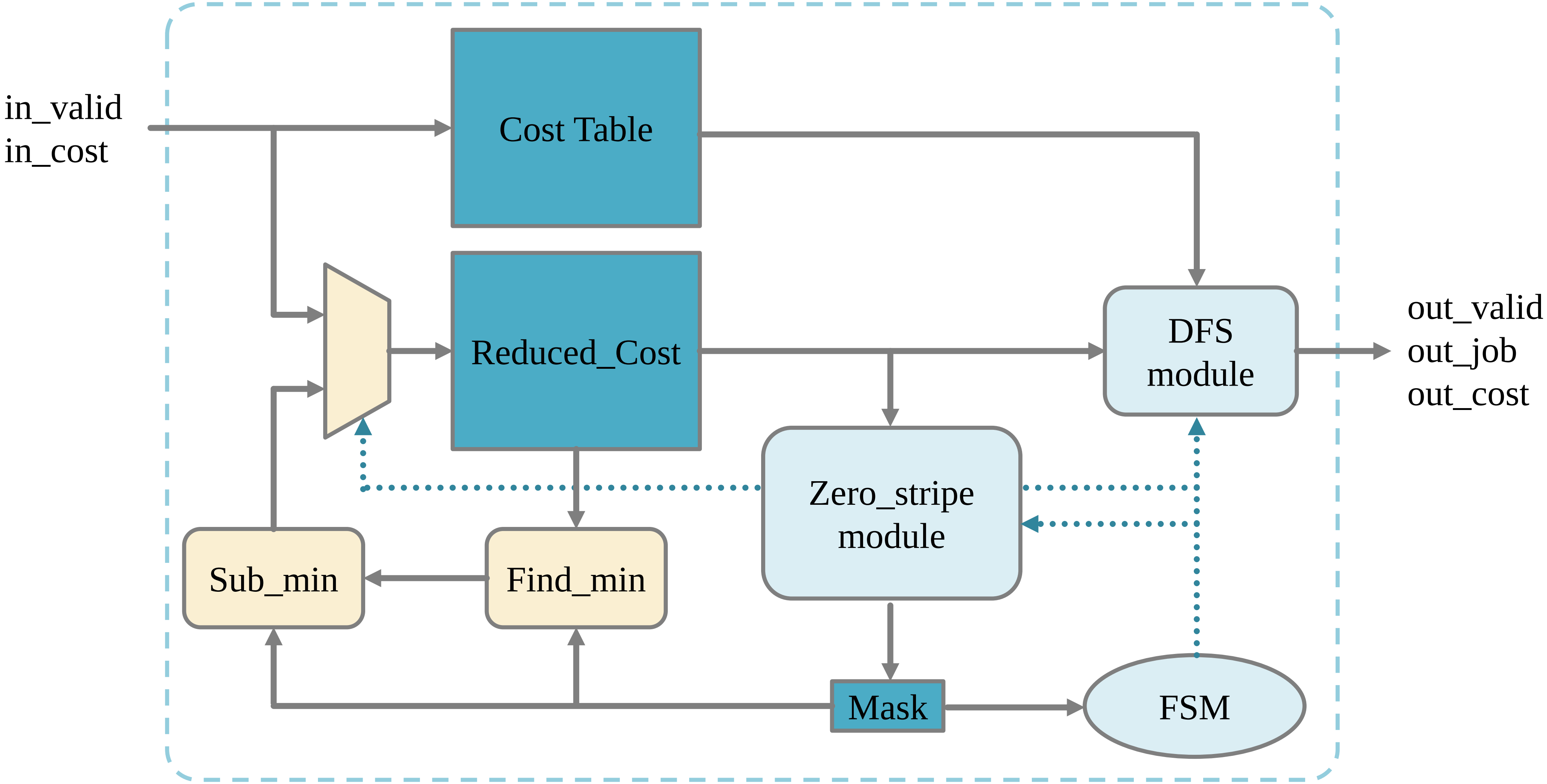
mask_col (row actually)

0	0	0	0	0	1	0	0

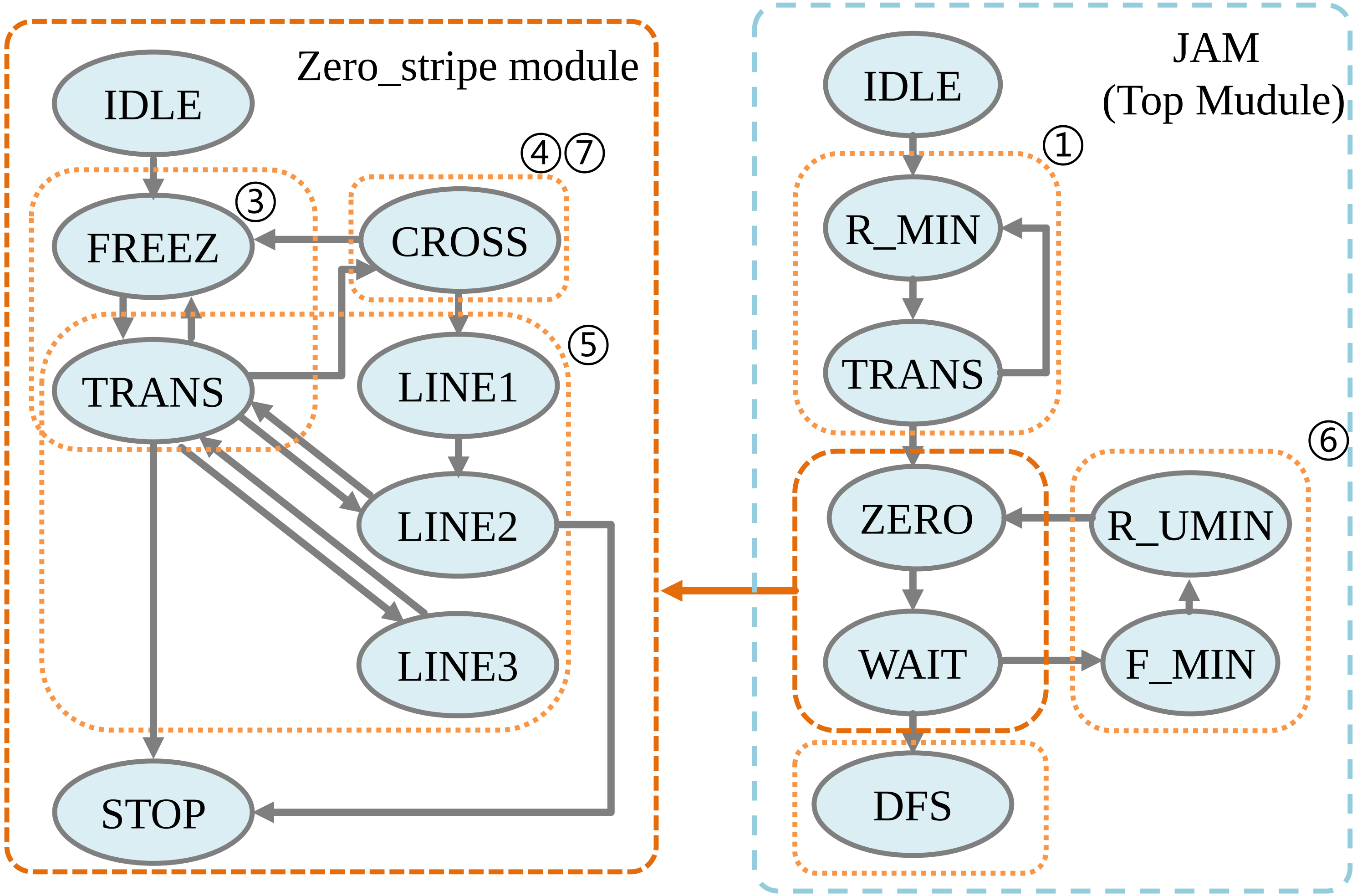
0	0	1	0	0	1	0	0

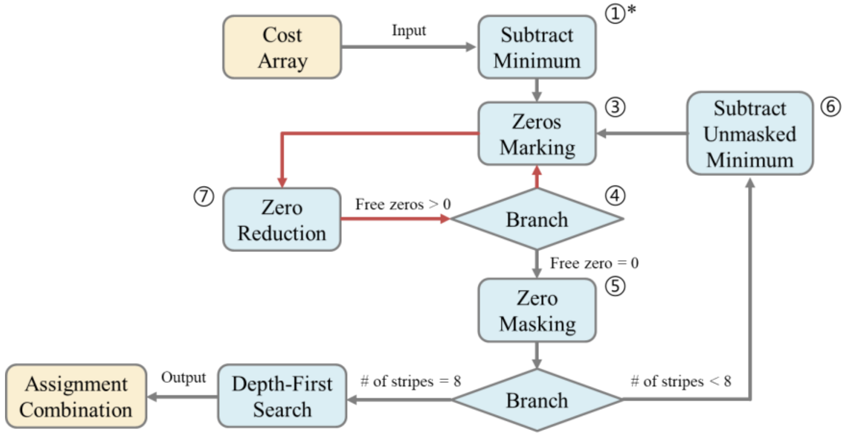
check & shift the value.

Hardware Architecture (Block diagram)



State Diagram





Summary

- Row-wise operation + transpose to simplify hardware.
- Share hardware on some similar step.

Design Flow

- 1. 用軟體先寫出演算法 (助教是用Python, 大約花了兩個晚上)
  - 寫的時候算法要接近硬體在算的方式
  - Debug會比用Verilog上用nWave Debug簡單一點
  - 自己哪裡會算錯在這裡就會知道，寫Verilog照著軟體寫的步驟寫就好
- 2. 寫Pattern
- 3. 寫Design (大約兩個晚上)
  - 因為軟體階段就已經規劃好該怎麼算，所以就只是照寫出來。

Q&A for Report

- 同學:

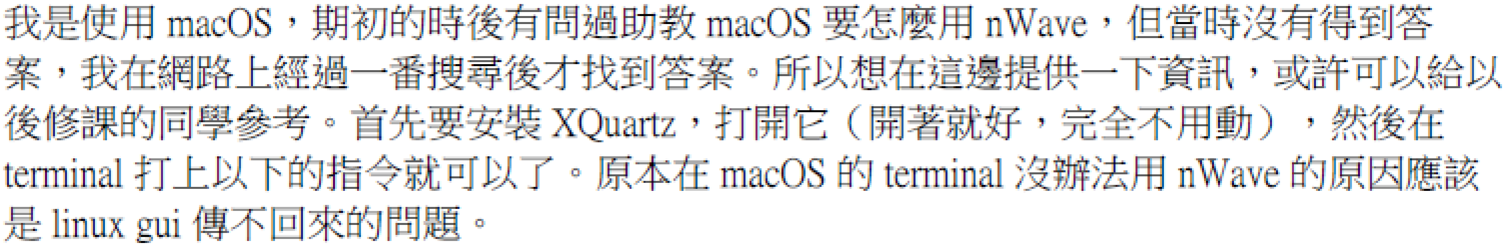
Pipeline 以前的我:

遇到Pipeline以後的我:

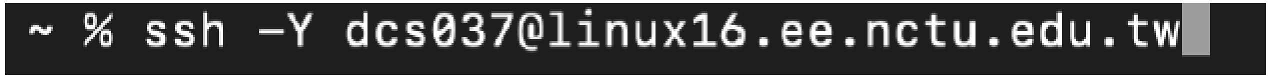
- TA:
  說實話，助教寫這個Final Project幾乎沒用到nWave看波型XD
  一是上一頁所說，你的架構會錯哪裡在軟體測試階段就會知道，
  二是算完的結果是直接用$display印出來看會比nWave慢慢看還快，
- 助教開nWave都是看一下FSM有沒有跳對、哪裡條件出問題。

Q&A for Report

- 同學:



我是使用 macOS，期初的時後有問過助教 macOS 要怎麼用 nWave，但當時沒有得到答案，我在網路上經過一番搜尋後才找到答案。所以想在這邊提供一下資訊，或許可以給以後修課的同學參考。首先要安裝 XQuartz，打開它（開著就好，完全不用動），然後在 terminal 打上以下的指令就可以了。原本在 macOS 的 terminal 沒辦法用 nWave 的原因應該是 linux gui 傳不回來的問題。



```
~ % ssh -Y dcs037@linux16.ee.nctu.edu.tw
```

- TA:
  非常感謝你提供資訊，
  因為助教們都沒有使用macOS的系統，
  所以這方面是完全幫不上忙，
  你的資訊我們會傳下去給下一屆學弟妹的。