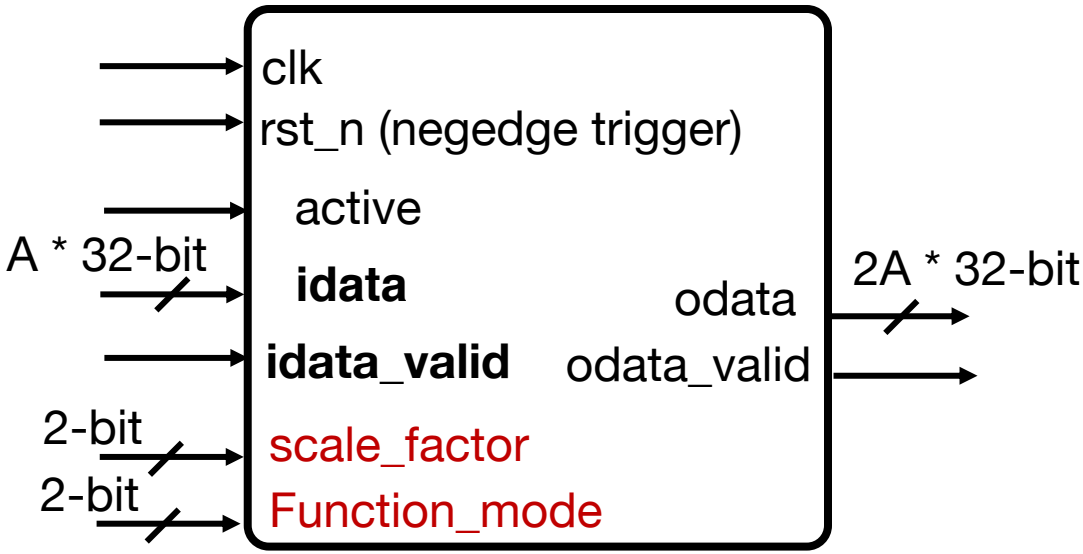


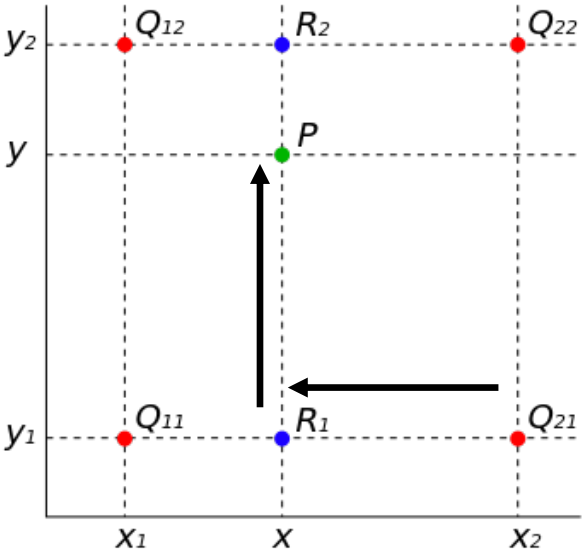
Function_mode
00: Downsample – MaxPooling
01: Downsample – AvgPooling
10: Upsample

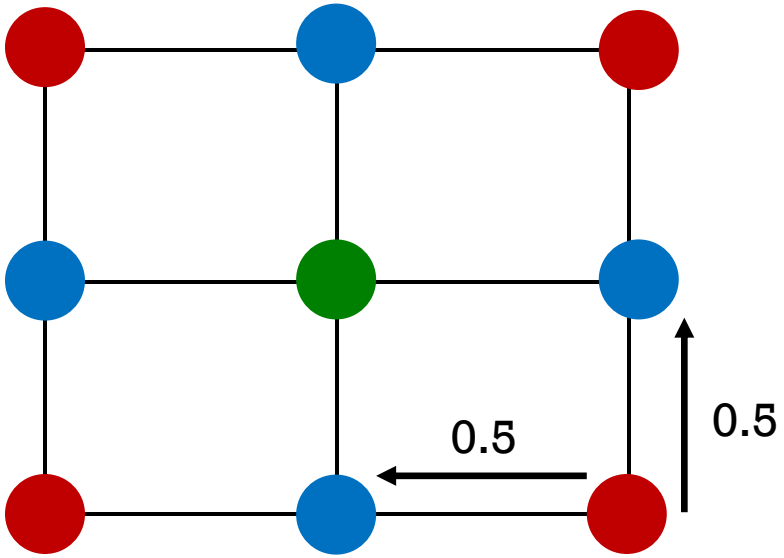
A:
= 8 * 8
or 8 * 4

scale_factor
If (Function_mode[1] == 0) //Downsample
  00: 2x2 kernel, stride 2
  01: 3x3 kernel, stride 2
Else: //Upsample
  00: 2x
  01: 4x

Bilinear Interpolation

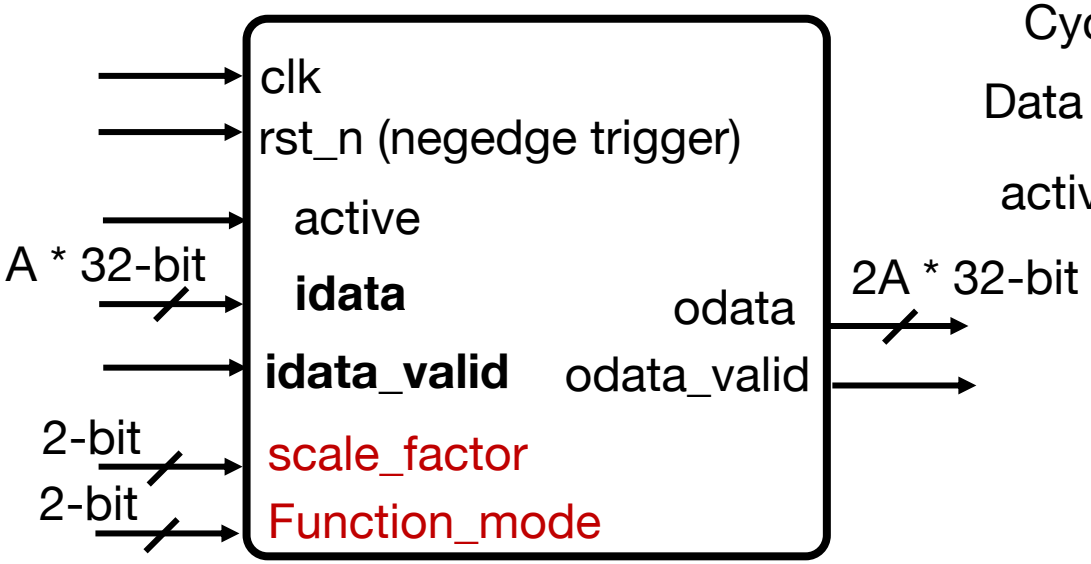
Hardware
computationally cheap interpolation function



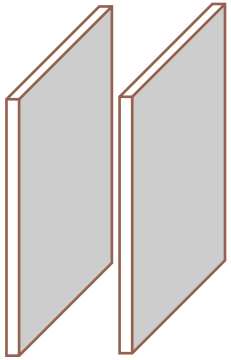


Pytorch source code:
https://github.com/pytorch/pytorch/blob/c4f10e0fe7a299cc434eacacae2d9265f7e48710/aten/src/ATen/native/UpSampleBilinear2d.cpp#L13

Bilinear Interpolation Interface



Cycle	t	t-1
Data Valid	1	1
activate	1	0

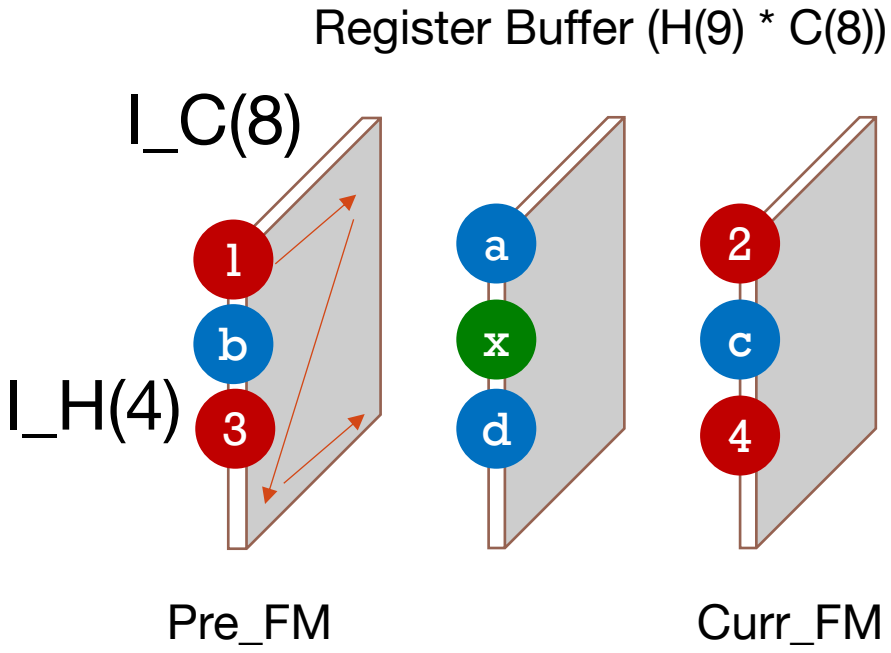


Cycle	0	1	2	3	4
active	0	1	0	1	-
idata	A	B	x	C	-
Idata_valid	valid	valid	0	valid	-
Reg: activate_r			1	0	1
Reg: Curr_FM		A	B	B	C
Reg: Pre_FM			A	A	B

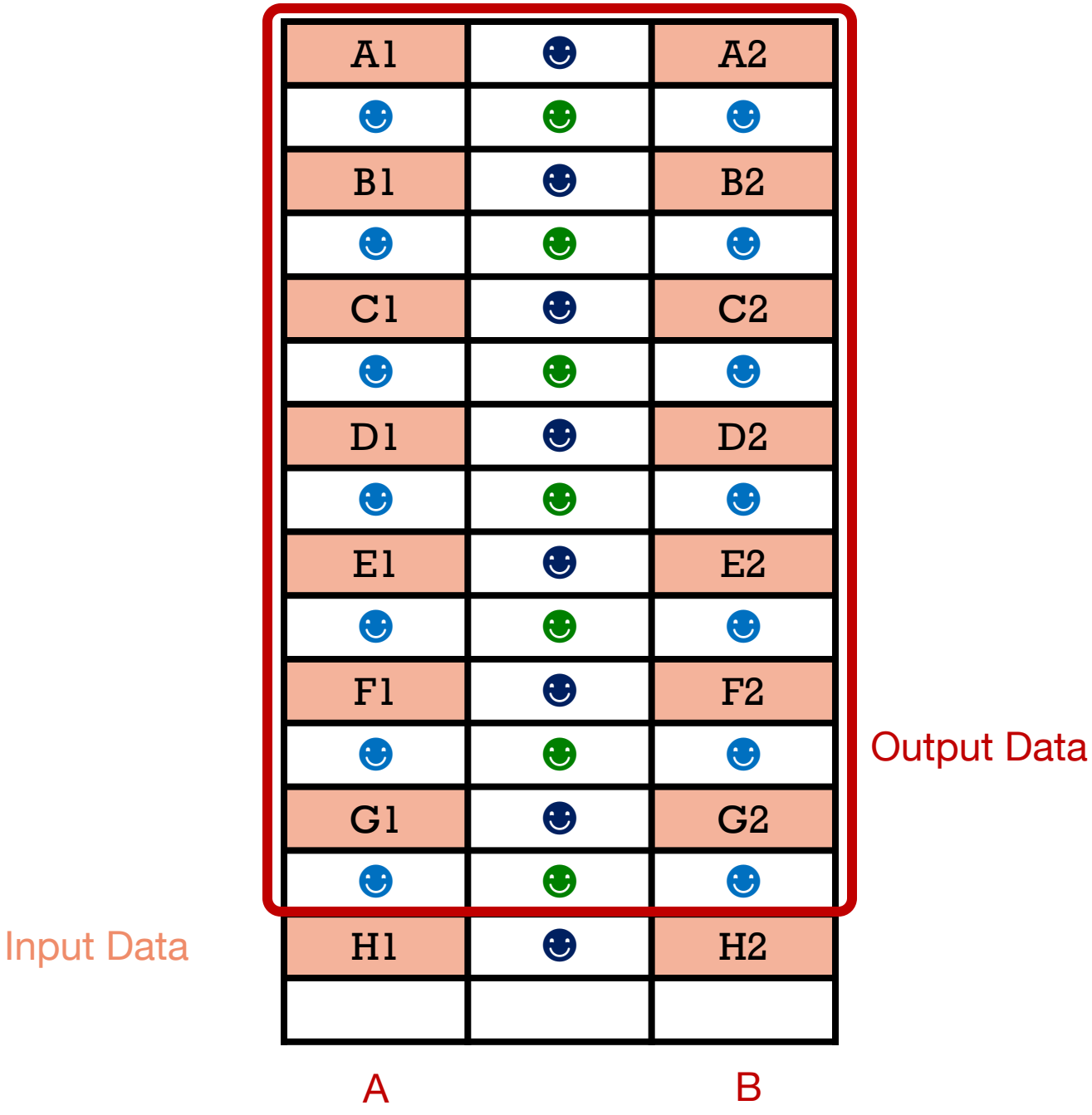
Stage1: 計算 a = (1+2)>>1; b = (1+3)>>1; …

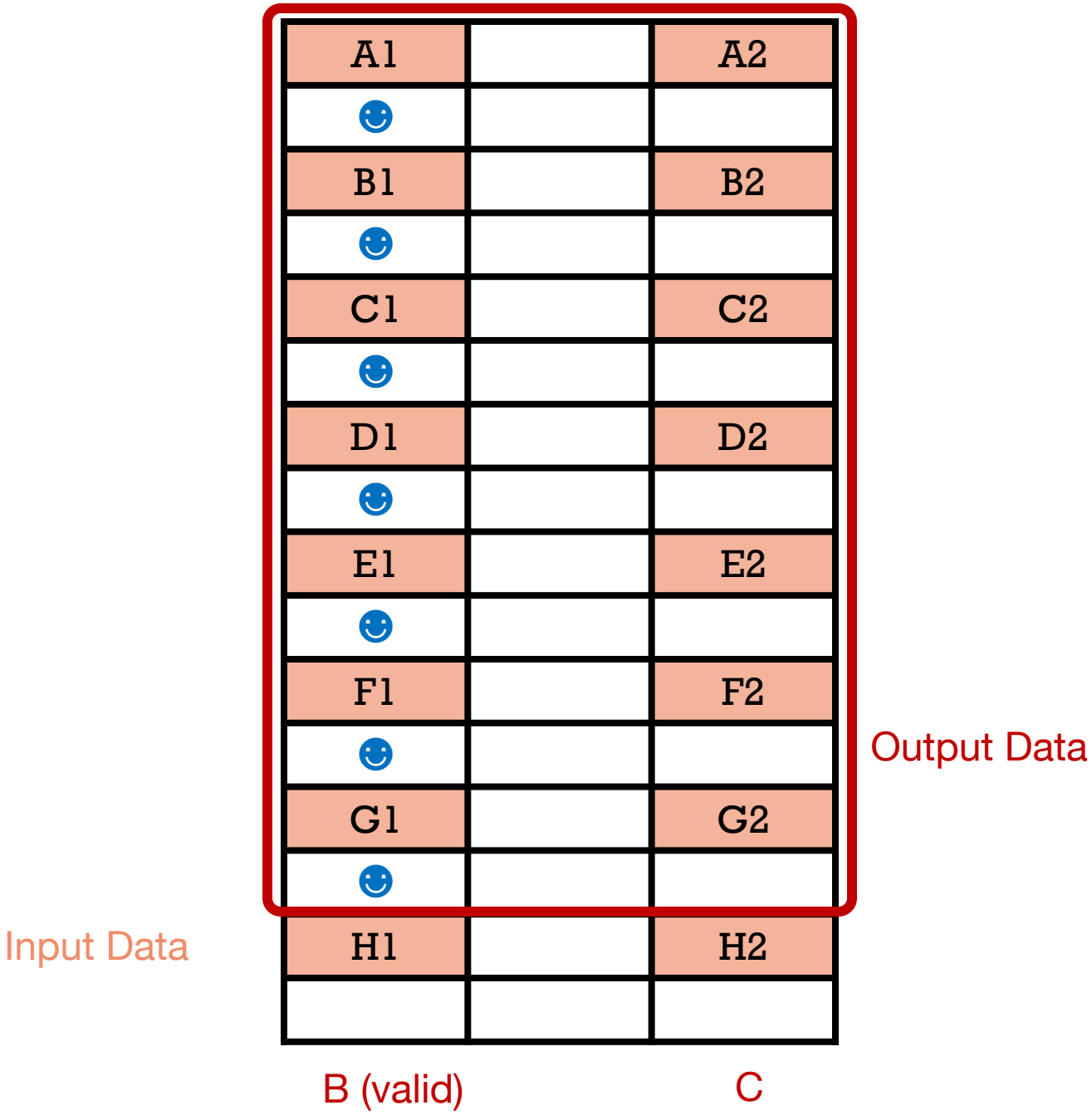
Stage2: 計算 x = (b + c + (1+3)_LSB + (2+4)_LSB) >> 1

Stage3: 假設 input size 為 4(Height) * 8(Channel)，Upsample 2x 後，output size 變為 9 * 8，最後只需輸出 8 (Height) * 8



Cycle	t	t+1	t+2	t+3
activate_r	1	0	1	
Curr_FM	B		C	
Pre_FM	A		B	
stage1	t			t
stage2		t		
odata		A	New	B
odata_valid		1	1	1


A1
A2
B1
B2
C1
C2
D1
D2
E1
E2
F1
F2
G1
G2
H1
H2
Output Data
Input Data
A
B

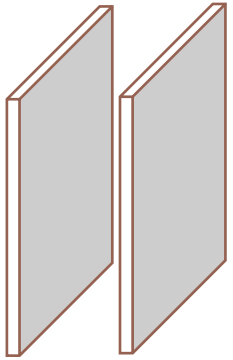

A1
A2
B1
B2
C1
C2
D1
D2
E1
E2
F1
F2
G1
G2
H1
H2
Output Data
Input Data
B (valid)
C

Downsample

- Kernel = 2, 3
- Ex: 2x2 kernel
  - Max Pooling = Max(aa, bb, cc, dd)
  - Avg Pooling = Avg(aa, bb, cc, dd)

aa	bb
cc	dd

Cycle	t	t-1
Data Valid	1	1
activate	1	0



Cycle	0	1	2	3	4
active	0	1	0	1	-
idata	A	B	x	C	-
Idata_valid	valid	valid	0	valid	-
Reg: activate_r			1	0	0
Reg: Curr_FM		A	B	B	C
Reg: Pre_FM			A	A	B

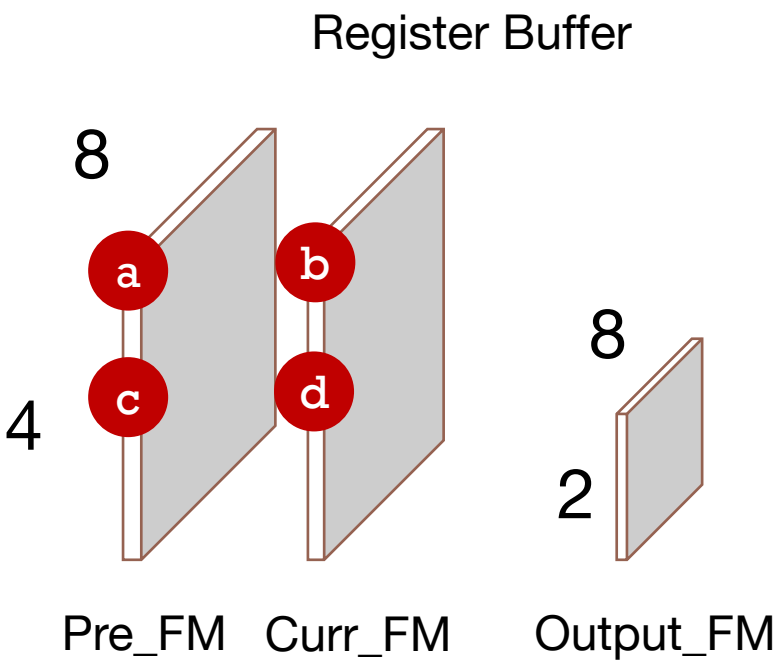
依據 kernel size 與 Pooling 方式，選擇計算結果

Ex: Kernel 2x2, Avg Pooling

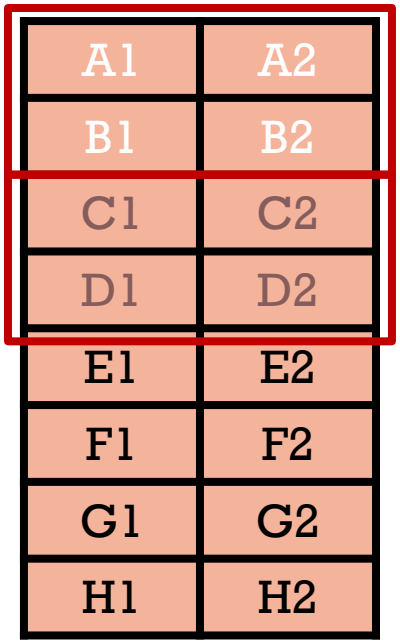
Stage1: 計算 a + b = r1, c + d = r2

Stage2: (r1 + r2) >> 2 = r3

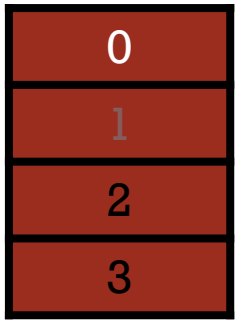
Stage3: store r3 to Output_FM



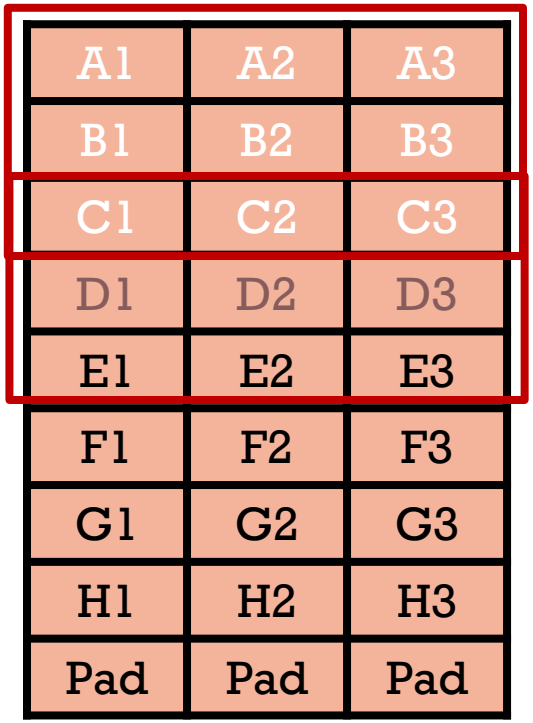
Cycle	t	t+1	t+2	t+3
activate_r	1	0	1	
Curr_FM	B		C	
Pre_FM	A		B	
stage1	r1, r2			…
stage2		r3		
odata			V	
odata_valid			1	



Input Data
(2x2, stride 2)



Output Data



Input Data
(3x3, stride 2)