SpiNNaker2 MAC Array (16x2) Performance Report

Note: this performance report is based on VGG-19 model with the best mapping strategy.

- **Mapping strategy:**

Convolution layer: operator fusion + data reuse
Fully-connected layer: operator fusion
→ Operator fusion will decrease the size of output activation
→ Data reuse will reduce bandwidth problem of dram
(Next slide has details information for the mapping strategy)

- **Splitting scheme difference between MAC array (16*4) and MAC array (16*2)**

No different

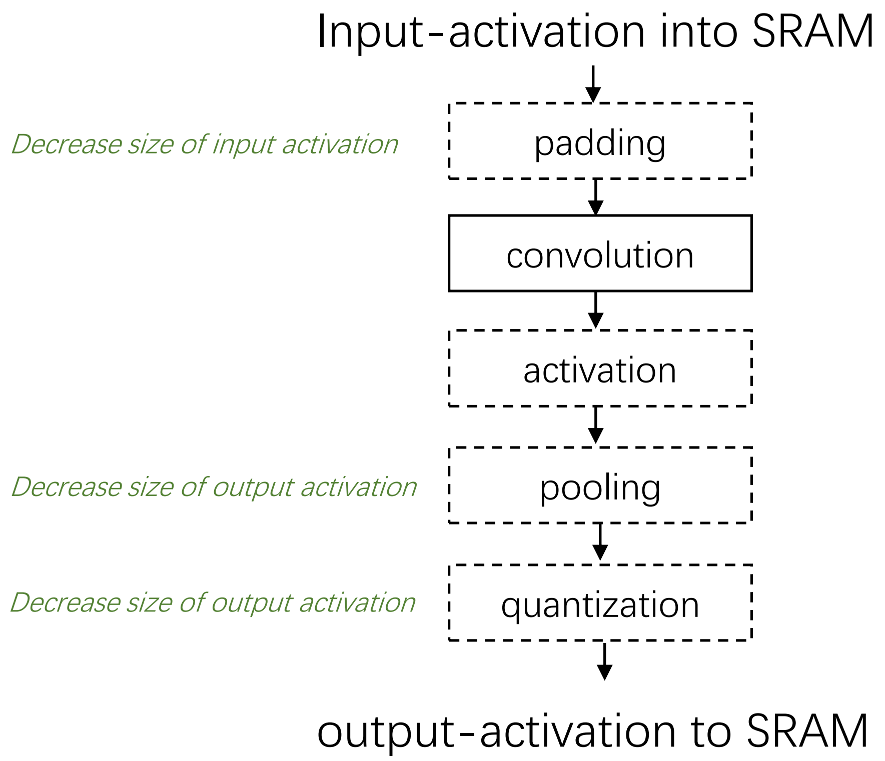
- **Simulator difference between MAC array (16*4) and MAC array (16*2)**

MAC array size is different 16*4 → 16 *2;
The prefetching of filter for CONV is different.

For 16*4_MAC_array, each 4 clocks send a fetching request.
For 16*2_MAC_array, each 8 clocks send a fetching request.

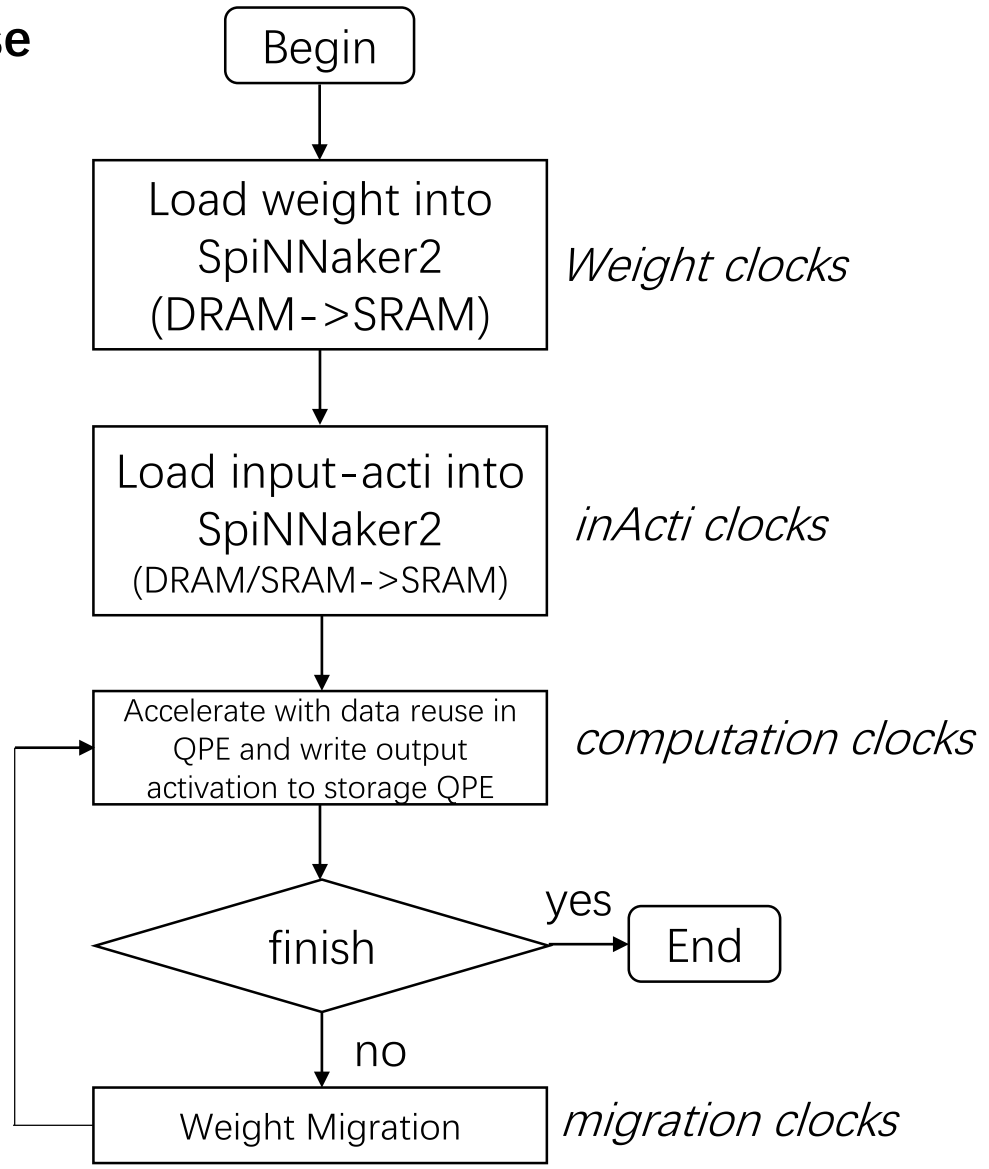
Mapping strategy Convolution layer: operator fusion + data reuse

Operator Fusion


Input-activation into SRAM
Decrease size of input activation
padding
convolution
activation
Decrease size of output activation
pooling
Decrease size of output activation
quantization
output-activation to SRAM

Note that padding, activation, pooling and quantization is not simulated. It is assume to be done immediately (0 clock).

Data reuse

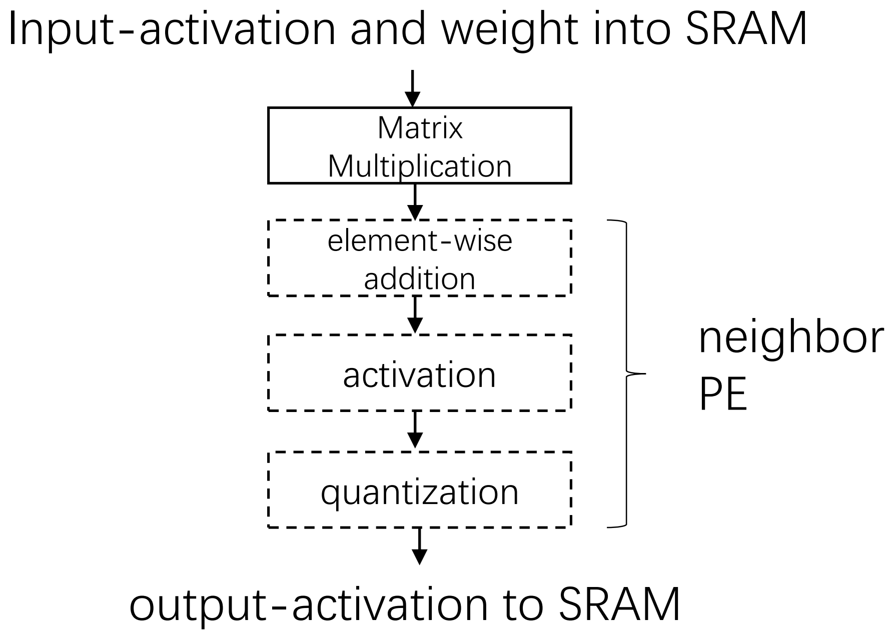

Begin
Load weight into SpiNNaker2 (DRAM->SRAM)
Weight clocks
Load input-acti into SpiNNaker2 (DRAM/SRAM->SRAM)
inActi clocks
Accelerate with data reuse in QPE and write output activation to storage QPE
computation clocks
finish
yes
End
no
Weight Migration
migration clocks

Mapping strategy Fully-connected layer: operator fusion

For fully-connected layer, operator fusion and data reuse cannot coexist. As for the convolution, its result (the output activation of CONV will be the input activation of FC) will be stored in QPE. Therefore, data reuse in FC has no meaning, because only input-activation could be reused and due to NoC bandwidth is larger than DRAM bandwidth, reuse input-activation will not increase the system performance. **The whole system performance of FC wil be limited by DRAM bandwidth.**

Due to the DRAM bandwidth, most the Pes in SpiNNaker2 will be idle. Theoretically, 8 PEs can fully utilize 4 DRAM.

However, because FC is splitted, some computation (element-wise addition) muss be done ARM core. In order to eliminate this part, some idle PEs (not the PE used for FC) will be used to accelerate the element-wise addition. Moreover, these PEs will also be used to accelerate non-linearity and quantization.



Result for VGG-19

MAC array Row = 4， Column = 16

"VGG-CONV_1" :	37982
	weight_clock: 497
	inActi_clock: 8374
	weight_migration: 0
	computation: 29111 (including write outActi from SRAM to SRAM)
"VGG-CONV_6" :	214575
	weight_clock: 9611
	inActi_clock: 23344
	weight_migration: 2432*4=9728
	computation: 171892 (including write outActi from SRAM to SRAM)
"VGG-CONV_8":	413733
	weight_clock: 19114
	inActi_clock: 49389
	weight_migration: 4736*4=18944
	computation: 326286 (including write outActi from SRAM to SRAM)
"VGG-FC_19" :	132044

Result for VGG-19

MAC array Row = 2， Column = 16

"VGG-CONV_1" :	49153
	weight_clock: 495
	inActi_clock: 8374
	weight_migration: 0
	computation: 40284 (including write outActi from SRAM to SRAM)
"VGG-CONV_6" :	327914
	weight_clock: 9611
	inActi_clock: 23342
	weight_migration: 2432*4=9728
	computation: 285233 (including write outActi from SRAM to SRAM)
"VGG-CONV_8":	639999
	weight_clock: 19115
	inActi_clock: 49386
	weight_migration: 4736*4=18944
	computation: 552554 (including write outActi from SRAM to SRAM)
"VGG-FC_19" :	131864

Result for VGG-19

MAC array Row = 4， Column = 8

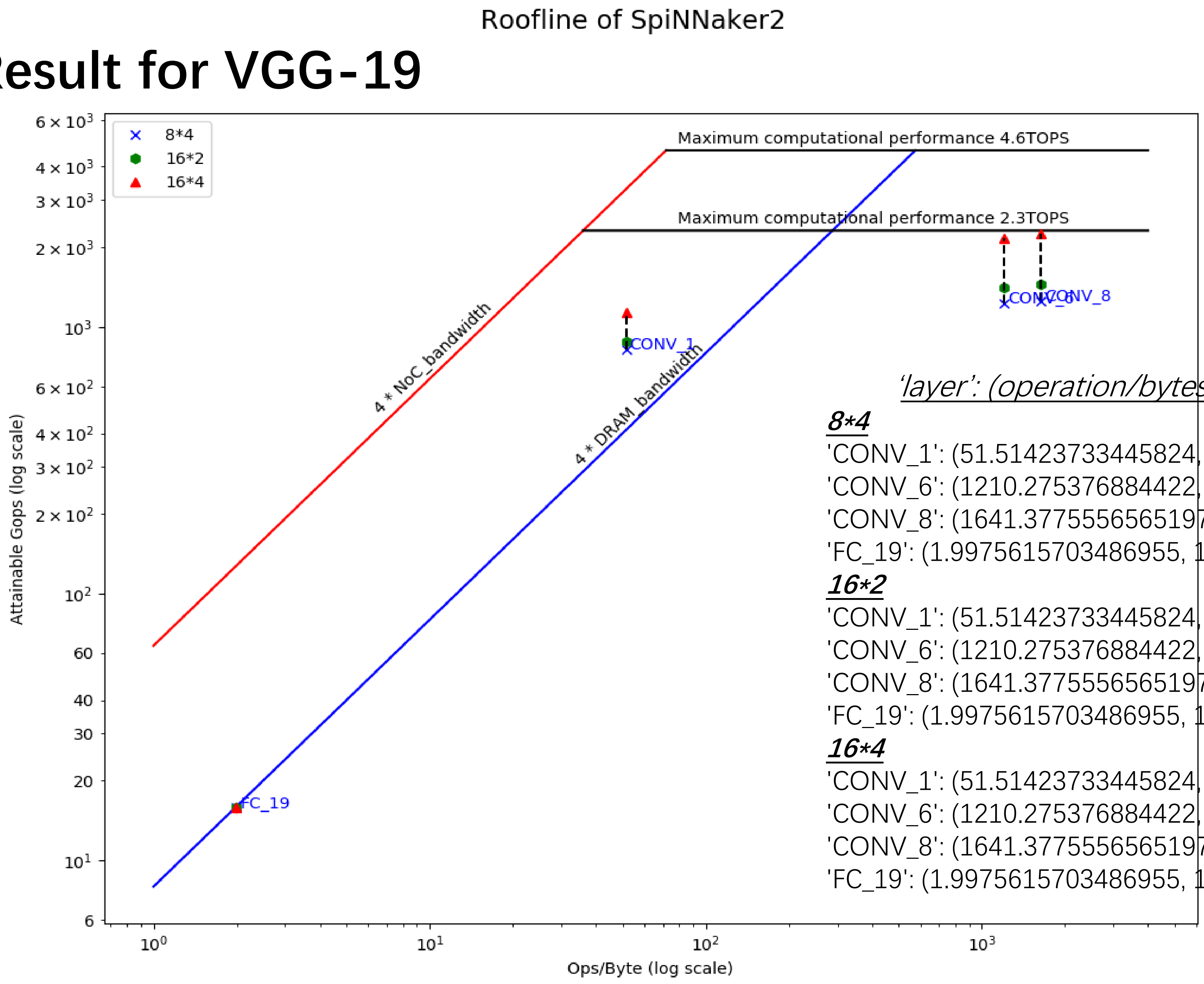
"VGG-CONV_1" : 52504
weight_clock: 493
inActi_clock: 8373
weight_migration: 0
computation: 43638 (including write outActi from SRAM to SRAM)

"VGG-CONV_6" : 376141
weight_clock: 9611
inActi_clock: 23342
weight_migration: 2433*4=9732
computation: 333456 (including write outActi from SRAM to SRAM)

"VGG-CONV_8": 736468
weight_clock: 19115
inActi_clock: 49388
weight_migration: 4736*4=18944
computation: 649021 (including write outActi from SRAM to SRAM)

"VGG-FC_19" : 132816

Result for VGG-19



'layer': (operation/bytes, attainable Gops)

8*4

'CONV_1': (51.51423733445824, 825.6906902331251)
'CONV_6': (1210.275376884422, 1229.3847679460628)
'CONV_8': (1641.3775556565197, 1255.783050994748)
'FC_19': (1.9975615703486955, 15.789904830743284)

16*2

'CONV_1': (51.51423733445824, 881.9820560291334)
'CONV_6': (1210.275376884422, 1410.192965228688)
'CONV_8': (1641.3775556565197, 1445.0710579235279)
'FC_19': (1.9975615703486955, 15.903900988897652)

16*4

'CONV_1': (51.51423733445824, 1141.3844452635458)
'CONV_6': (1210.275376884422, 2155.0600768961904)
'CONV_8': (1641.3775556565197, 2235.3644306835567)
'FC_19': (1.9975615703486955, 15.882221077822544)

Result for ResNet-50

MAC array Row = 4， Column = 16

"RES-CONV_1":	99623
	weight_clock: 971
	inActi_clock: 11061
	weight_migration: 0
	computation: 87591 (including write outActi from SRAM to SRAM)
"CONV_SC_14":	115207
	weight_clock: 6926
	inActi_clock: 18284
	weight_migration: 634 (half) * 2 = 1268
	computation: 89363 (including write outActi from SRAM to SRAM)
"CONV_19":	44987
	weight_clock: 7735
	inActi_clock: 15453
	weight_migration: 0
	computation: 21799 (including write outActi from SRAM to SRAM)
"CONV_42":	47116
	weight_clock: 25364
	inActi_clock: 11049
	weight_migration: 0
	computation: 10703 (including write outActi from SRAM to SRAM)

Result for ResNet-50

MAC array Row = 2,， Column = 16

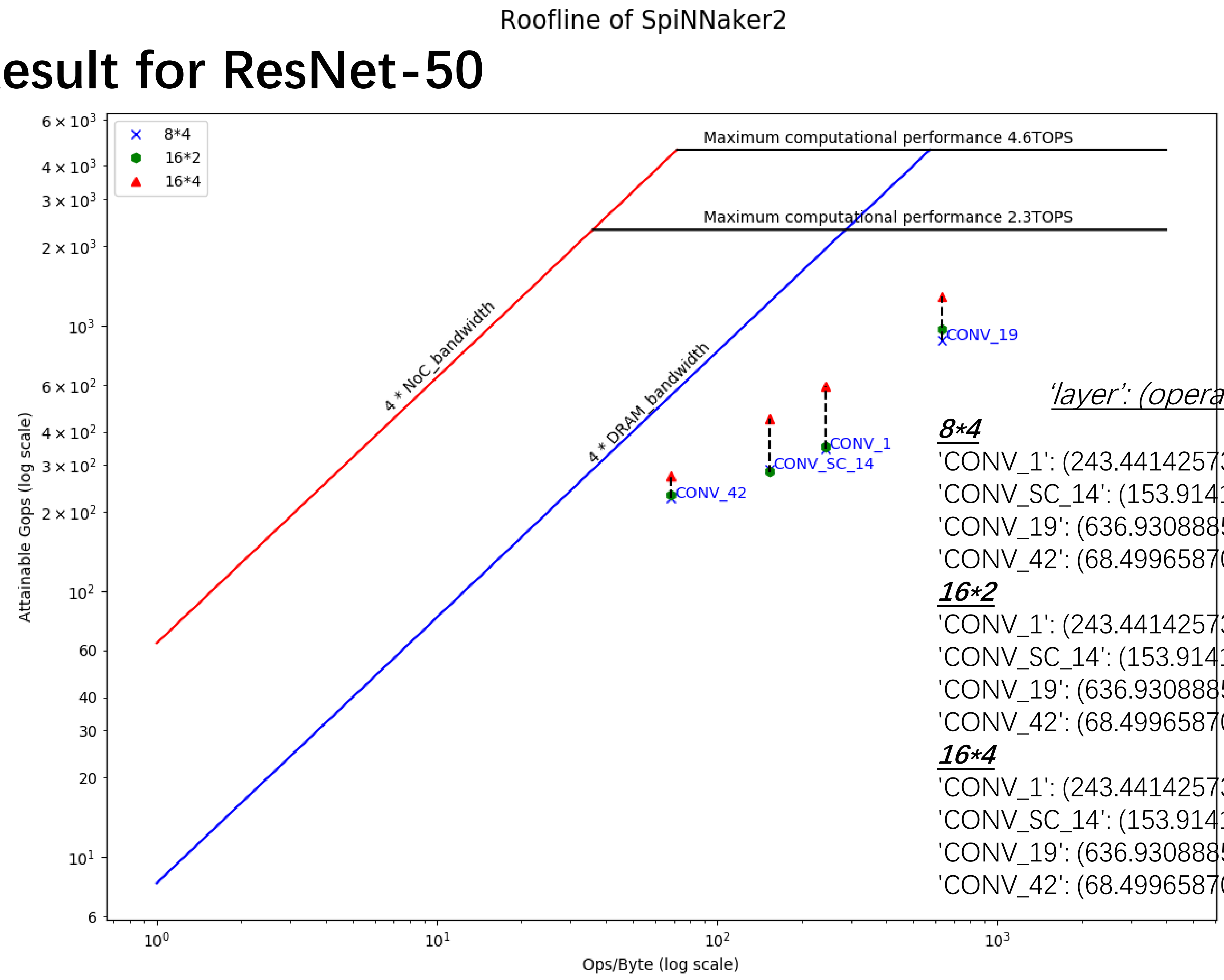
"RES-CONV_1":	168451
	weight_clock: 968
	inActi_clock: 11061
	weight_migration: 0
	computation: 156422 (including write outActi from SRAM to SRAM)
"CONV_SC_14":	180906
	weight_clock: 6936
	inActi_clock: 18290
	weight_migration: 634 (half) * 2 = 1268
	computation: 154412 (including write outActi from SRAM to SRAM)
"CONV_19":	59216
	weight_clock: 7736
	inActi_clock: 15455
	weight_migration: 0
	computation: 36025 (including write outActi from SRAM to SRAM)
"CONV_42":	55317
	weight_clock: 25365
	inActi_clock: 11051
	weight_migration: 0
	computation: 18901 (including write outActi from SRAM to SRAM)

Result for ResNet-50

MAC array Row = 4， Column = 8

"RES-CONV_1":	170628
	weight_clock: 969
	inActi_clock: 11060
	weight_migration: 0
	computation: 158599 (including write outActi from SRAM to SRAM)
"CONV_SC_14":	176935
	weight_clock: 6898
	inActi_clock: 18295
	weight_migration: 634 (half) * 2 = 1268
	computation: 150474 (including write outActi from SRAM to SRAM)
"CONV_19":	65233
	weight_clock: 7739
	inActi_clock: 15454
	weight_migration: 0
	computation: 42040 (including write outActi from SRAM to SRAM)
"CONV_42":	57366
	weight_clock: 25366
	inActi_clock: 11050
	weight_migration: 0
	computation: 20950 (including write outActi from SRAM to SRAM)

Result for ResNet-50



<u>*'layer': (operation/bytes, attainable Gops)*</u>

8*4

'CONV_1': (243.44142573799928, 345.8223503762571)
'CONV_SC_14': (153.91411042944785, 290.3903919518467)
'CONV_19': (636.9308885754584, 886.0967915012341)
'CONV_42': (68.49965870307167, 223.91409545723948)

16*2

'CONV_1': (243.44142573799928, 350.2916337688705
'CONV_SC_14': (153.91411042944785, 284.0161409792931)
'CONV_19': (636.9308885754584, 976.1340178330182
'CONV_42': (68.49965870307167, 232.20810962271992)

16*4

'CONV_1': (243.44142573799928, 592.3027413348323)
'CONV_SC_14': (153.91411042944785, 445.9817893009973)
'CONV_19': (636.9308885754584, 1284.8767866272476)
'CONV_42': (68.49965870307167, 272.6261991680108)}

Conclusion

- Decreasing computing resource to be half, the attainable computing power is not half. → Row=2 has higher computing resource utilization. It could better to balance the performance between different components. Or in other word, difference components match each other better.

- Decreasing computing resource to be half, the NoC's memory bandwidth problem is not so serious as Row=4. Current simulation is under $f_{NoC} = 2 * f_{pe}$. If $f_{NoC} < 2 * f_{pe}$, I think computing resource utilization of Row=2 is higher.

- Decreasing computing resource to be half, FC could get a little bit higher computing power, because the data amount increased by the alignment is decreased, which saving the times to output the result from MAC array to neighbor PE. However, It doesn’t affect CONV.

- **16∗2 is better than 8∗4.**