Advanced Arch Wet-HW1

Chris Shakkour, 208157826, christian.s@campus.technion.ac.il

Nadi Najjar, 211610704, nadi.najjar@campus.technion.ac.il

Contents

CONFIGURATION

MAGIC SUM = (6+4) % 7 = 3

MAGIC FILTER = 4

Systolic array

Arch Feature	Size
ArrayHeight	16
ArrayWidth	16
IfmapSramSz	128KB
FilterSramSz	128KB
OfmapSramSz	128KB
IfmapOffset	0
FilterOffset	10000000
OfmapOffset	20000000
Dataflow	OS

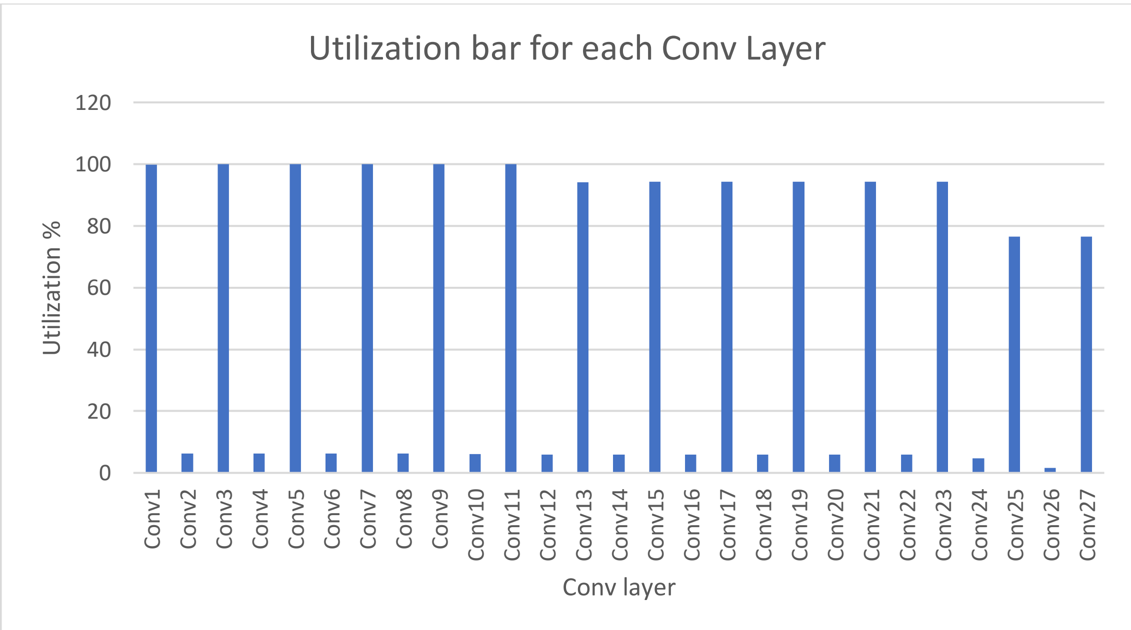
Network topology

Layer name	IFMAP Height	IFMAP Width	Filter Height	Filter Width	Channels	Num Filter	Strides
Conv1	224	224	4	4	3	32	2
Conv2	112	112	4	4	32	1	1
Conv3	112	112	1	1	32	64	1
Conv4	112	112	4	4	64	1	2
Conv5	56	56	1	1	64	128	1
Conv6	56	56	4	4	128	1	1
Conv7	56	56	1	1	128	128	1
Conv8	56	56	4	4	128	1	2
Conv9	28	28	1	1	128	256	1
Conv10	28	28	4	4	256	1	1
Conv11	28	28	1	1	256	256	1
Conv12	28	28	4	4	256	1	2
Conv13	14	14	1	1	256	512	1
Conv14	14	14	4	4	512	1	1
Conv15	14	14	1	1	512	512	1
Conv16	14	14	4	4	512	1	1
Conv17	14	14	1	1	512	512	1
Conv18	14	14	4	4	512	1	1
Conv19	14	14	1	1	512	512	1
Conv20	14	14	4	4	512	1	1
Conv21	14	14	1	1	512	512	1
Conv22	14	14	4	4	512	1	1
Conv23	14	14	1	1	512	512	1
Conv24	14	14	4	4	512	1	2
Conv25	7	7	1	1	512	1024	1
Conv26	7	7	4	4	1024	1	2
Conv27	7	7	1	1	1024	1024	1

QUESTION-1

Following execution results with the upper conv layer and magic mobile net topology.

Conv layer	Utilization %	Cycles
Conv1	99.85	74032
Conv2	6.24	380425
Conv3	99.97	100367
Conv4	6.21	194561
Conv5	99.97	100367
Conv6	6.23	360457
Conv7	99.98	200719
Conv8	6.18	94217
Conv9	99.97	100367
Conv10	6.10	163841
Conv11	99.98	200719
Conv12	6.00	45065
Conv13	94.21	106511
Conv14	5.90	65545
Conv15	94.22	213007
Conv16	5.90	655545
Conv17	94.22	213007
Conv18	5.90	65545
Conv19	94.22	213007
Conv20	5.90	65545
Conv21	94.22	213007
Conv22	5.90	65545
Conv23	94.22	213007
Conv24	4.68	24580
Conv25	76.55	131088
Conv26	1.56	16388
Conv27	76.55	262160



Part-a

The source of severe underutilization is depth-wise convolution layers(first layer of the depth-size separable convolution), the layers with only one filter, since the Systolic array in the OS dataflow format can process 16 different filters with 16 different input features for each iteration when fully utilized, the use of a single filter will keep the rest of the filter channels unutilized hence the systolic array structure is not suitable for such computation hence most of the MAC's inside the systolic array are acting like buffers and not doing actual computational work. To be more precise only one filter is active meaning only one column of the systolic array is doing computational work, hence we expect a utilization of 16/(16x16) = one MAC column (16) divided by overall MAC units (16x16) = ~6.25%.

Part-b

The underutilization layers are the depth-wise convolution layers (first layer of the depth-wise separable layers), the essence of using these layers is to drastically reduce the computational cost.

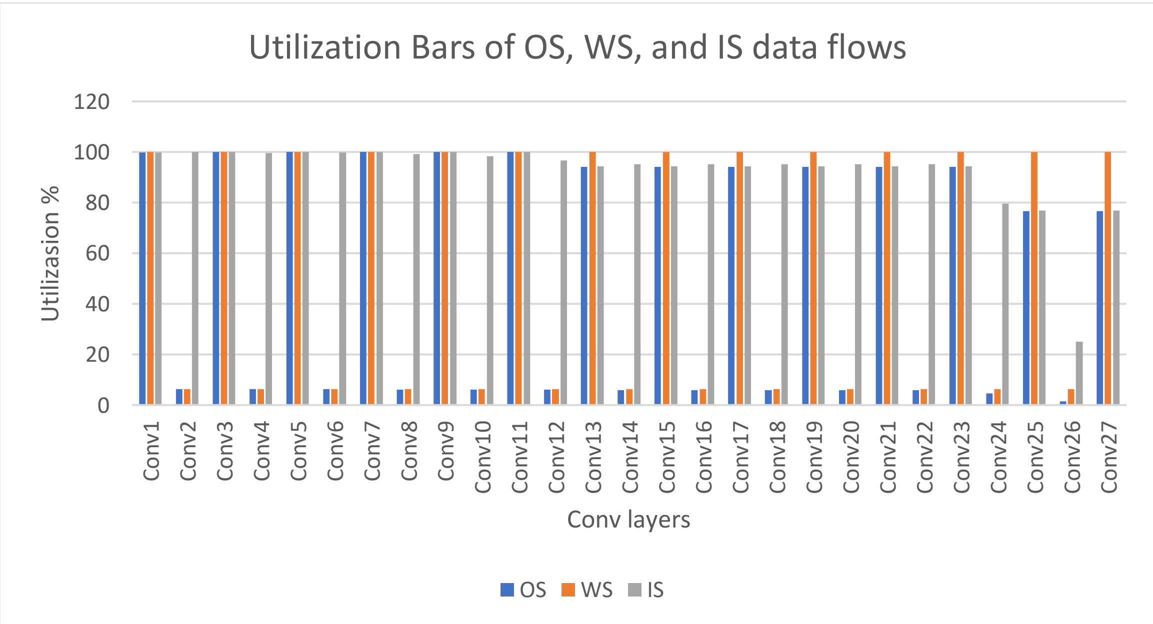
Part-c

The depth-wise separable convolutions which is a form of factorized convolutions which factorize a standard convolution into a depth-wise convolution and a 1×1 convolution called a point-wise convolution. The pointwise convolution then applies a 1×1 convolution to combine the outputs of the depth-wise convolution. This factorization has the effect of drastically reducing computation and model size.

QUESTION-2

For this Question we change the dataflow parameter from OS to WS and IS

Conv layer	Utilization[OS]	Utilization[IS]	Utilization[WS]
Conv1	99.85	99.90	100
Conv2	6.24	99.94	6.25
Conv3	99.97	100	100
Conv4	6.21	99.66	6.25
Conv5	99.97	100	100
Conv6	6.23	99.78	6.25
Conv7	99.98	100	100
Conv8	6.18	99.181	6.25
Conv9	99.97	100	100
Conv10	6.10	98.37	6.25
Conv11	99.98	100	100
Conv12	6.00	96.55	6.25
Conv13	94.21	94.34	100
Conv14	5.90	95.24	6.25
Conv15	94.22	94.34	100
Conv16	5.90	95.24	6.25
Conv17	94.22	94.34	100
Conv18	5.90	95.24	6.25
Conv19	94.22	94.34	100
Conv20	5.90	95.24	6.25
Conv21	94.22	94.34	100
Conv22	5.90	95.24	6.25
Conv23	94.22	94.34	100
Conv24	4.68	79.54	6.25
Conv25	76.55	76.80	100
Conv26	1.56	25.00	6.25
Conv27	76.55	76.80	100



Part-a

i. When computing the depth-wise convolution layers the utilization on the IS compared to the WS and OS is very different, this variance is due to the way the data is flow into the systolic array, the depth-wise layer is characterized with one filter, meaning there is only one filter to compute, in OS dataflow each MAC is responsible for an output feature hence the input features will flow through the rows but the filter weights will flow thru the columns hence the utilization is 16/256 only one column is active… the same goes for the WS dataflow each MAC is pre-loaded with one weight and the rest of the data is flown into the array, the filters we are using are 4x4 hence only 16 weights are loaded into the MAC array giving us a utilization of again 4x4/256. When using the IS dataflow where each MAC is pre-loaded with input features that we have in abundance the whole systolic array can be utilized, and the weights and output features are flown from the rows and columns.

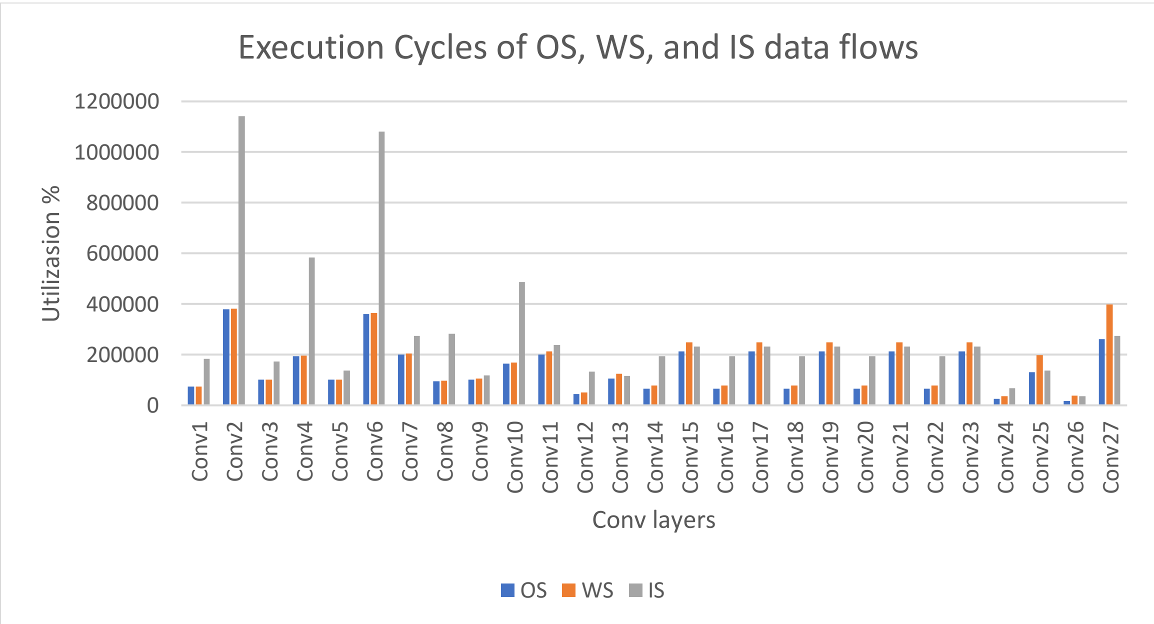
ii. When Comparing OS and WS we get the same utilization when looking at the depth-wise convolution layers. the depth-wise layer is characterized with one filter, meaning there is only one filter to compute, in OS dataflow each MAC is responsible for an output feature hence the input features will flow through the rows but the filter weights will flow thru the columns hence the utilization is 16/256 only one column is active… the same goes for the WS dataflow each MAC is pre-loaded with one weight and the rest of the data is flown into the array, the filters we are using are 4x4 hence only 16 weights are loaded into the MAC array giving us a utilization of again 4x4/256.

When comparing the rest of the layers the ones that are not depth-wise convolution meaning the point-wise and the FC and SoftMax layers at the end the utilization is pretty much the same since again the layers are abundant in the number of weights, input features and output features hence we could always allocate a MAC to work and manipulate the Row and column data flowing into the systolic array.

Part-b

i. Plotting cycles bars graph according to dataflow

Conv layer	Cycles [OS]	Cycles [WS]	Cycles [IS]
Conv1	74032	74214	182682
Conv2	380425	381248	1141024
Conv3	100367	100736	174048
Conv4	194561	195712	582720
Conv5	100367	101888	137200
Conv6	360457	363776	1080448
Conv7	200719	203776	274400
Conv8	94217	97536	281728
Conv9	100367	106496	118776
Conv10	163841	168448	487680
Conv11	200719	212992	237552
Conv12	45065	51712	133376
Conv13	106511	124928	116080
Conv14	65545	78848	193024
Conv15	213007	249856	232160
Conv16	65545	78848	193024
Conv17	213007	249856	232160
Conv18	65545	78848	193024
Conv19	213007	249856	232160
Conv20	65545	78848	193024
Conv21	213007	249856	232160
Conv22	65545	78848	193024
Conv23	213007	249856	232160
Conv24	24580	35328	67584
Conv25	131088	198656	136608
Conv26	16388	37888	36864
Conv27	262160	397312	273216



As seen in the bars plot above the OS dataflow is usually the shortest in execution time, hence, the speedup is expected to be negative…

Conv layer	Speedup % [OS/WS]	Speedup % [OS/IS]
Conv1	-0.25	-59.47
Conv2	-0.22	-66.66
Conv3	-0.37	-42.33
Conv4	-0.59	-66.61
Conv5	-1.49	-26.85
Conv6	-0.91	-66.64
Conv7	-1.50	-26.85
Conv8	-3.40	-66.56
Conv9	-5.76	-15.50
Conv10	-2.73	-66.40
Conv11	-5.76	-15.51
Conv12	-12.85	-66.21
Conv13	-14.74	-8.24
Conv14	-16.87	-66.04
Conv15	-14.75	-8.25
Conv16	-16.87	-66.04
Conv17	-14.75	-8.25
Conv18	-16.87	-66.04
Conv19	-14.75	-8.25
Conv20	-16.87	-66.04
Conv21	-14.75	-8.25
Conv22	-16.87	-66.04
Conv23	-14.75	-8.25
Conv24	-30.42	-63.63
Conv25	-34.01	-4.04
Conv26	-56.75	-55.54
Conv27	-34.02	-4.05

Calculating the overall speedup:

OS cycles = 3948624, WS cycles = 4496166, IS cycles = 7587906

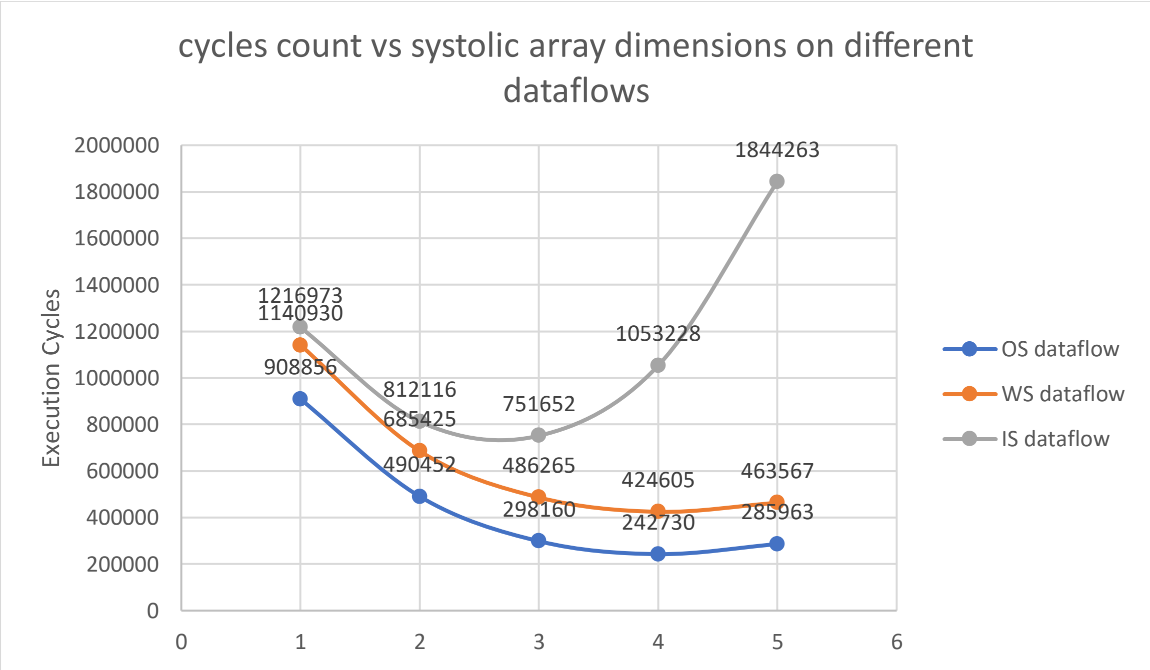
WS Speedup over OS = ((3948624/4496166)-1) *100 = 0.94 %

IS Speedup over OS = ((3948624/7587906)-1) *100 = -40.18 %

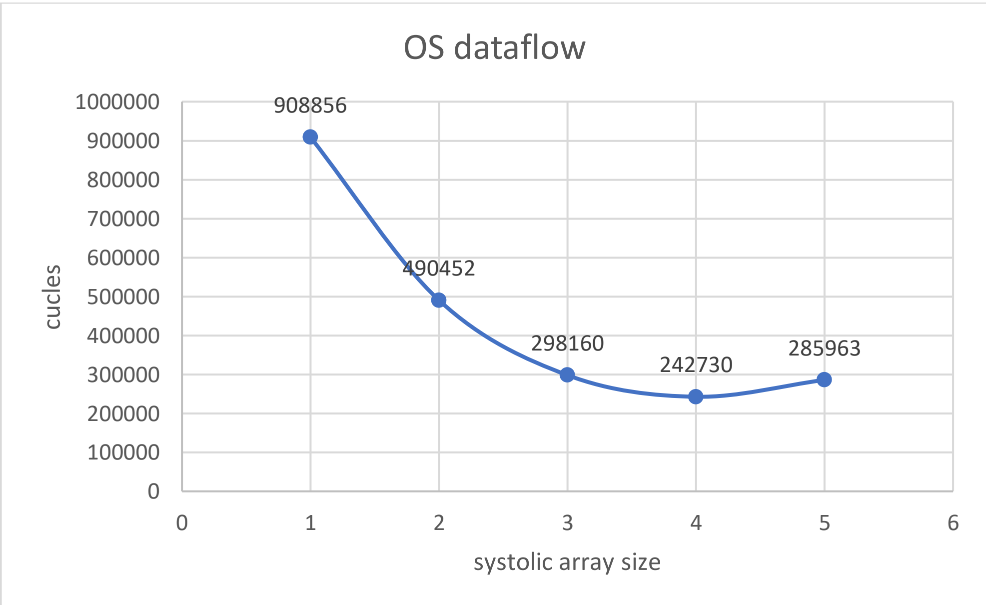
ii. When utilization increases, we expect a decrease in the cycles since more MAC's are doing computations, but this is not always the case. This statement is dependent on both the Hyper parameters of the Layer, IF, OF, Filter size and Etc... and the way the data is flow into the systolic array. For different hyper parameters we would expect different performance results for different dataflows, as seen in the previous question above we can conclude that one would use the largest between the input features the output features and filters to be stationary to ease the data flow manipulation and cut down the memory bandwidth into the systolic array.

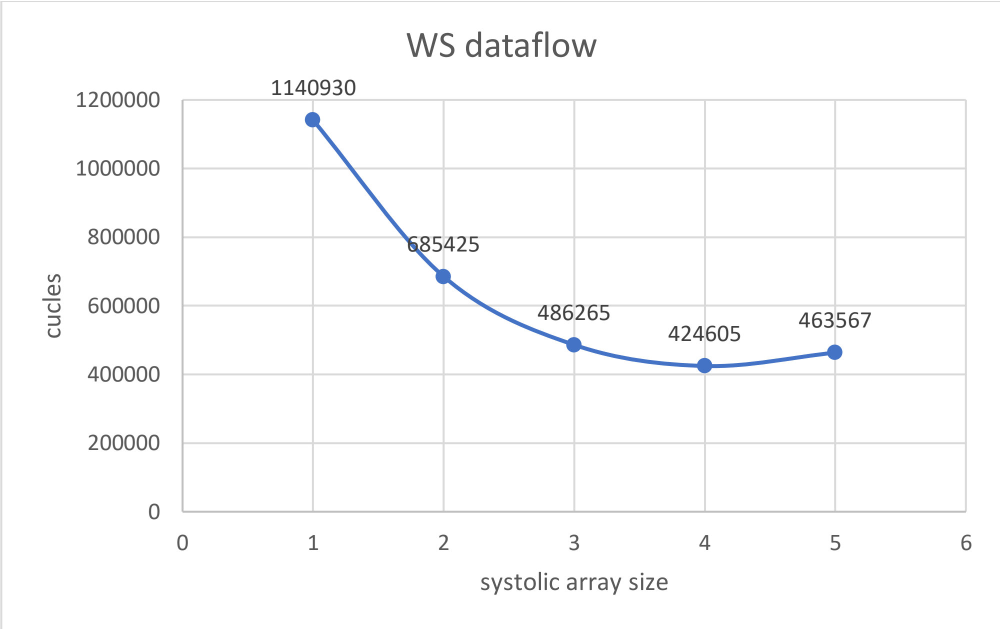
QUESTION-3

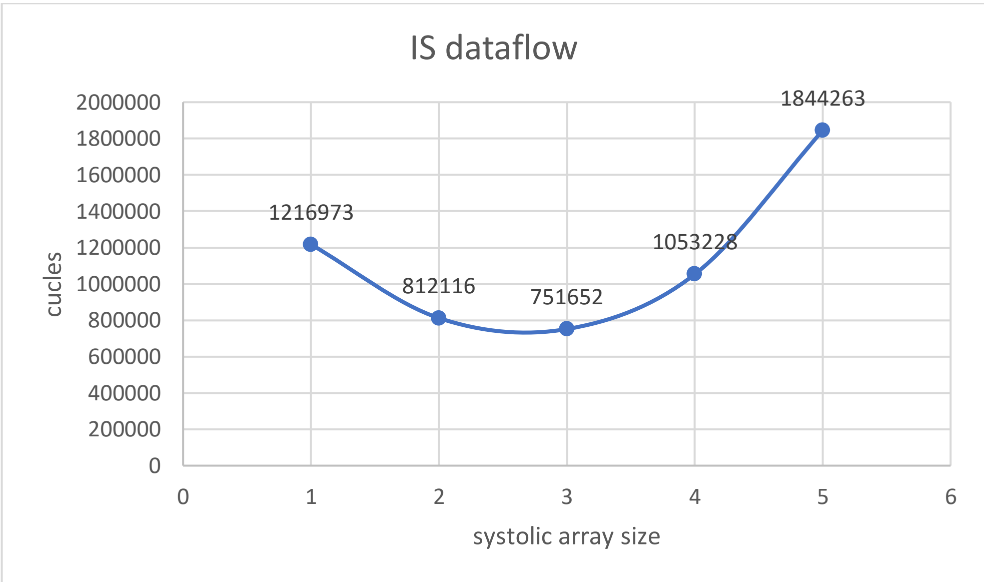
Measurement	Conv layer	Cycles [OS]	Cycles [WS]	Cycles [IS]
1	32x512	908856	1140930	1216973
2	64x256	490452	685425	812116
3	128x128	298160	486265	751652
4	265x64	242730	424605	1053228
5	512x32	285963	463567	1844263



Part-a







First let's take the Stationary input data flow, we can see that after reaching the optimal value 128x128 the cycle time starts to increase drastically since more systolic rows are present that flow the output features but less columns that flow the filters hence lower utilization because the more we progress along the network the more filters are needed and present but the number of columns that can flow filter weights is small hence it will take a long time to finish all the features.

The WS and OS acts almost identical with a slight constant different between the two, the WS and OS performance is affected by the number of MAC's and not directly by the width to column ratio as can be seen from the plots hence the higher the MAC ratio the lower the number of cycles until memory bottlenecks and other control bottlenecks kick in place.

QUESTION-4

Arch Feature	Size
ArrayHeight	16
ArrayWidth	16
IfmapSramSz	4MB/64KB
FilterSramSz	4MB/64KB
OfmapSramSz	4MB/64KB
IfmapOffset	0
FilterOffset	10000000
OfmapOffset	20000000
Dataflow	OS

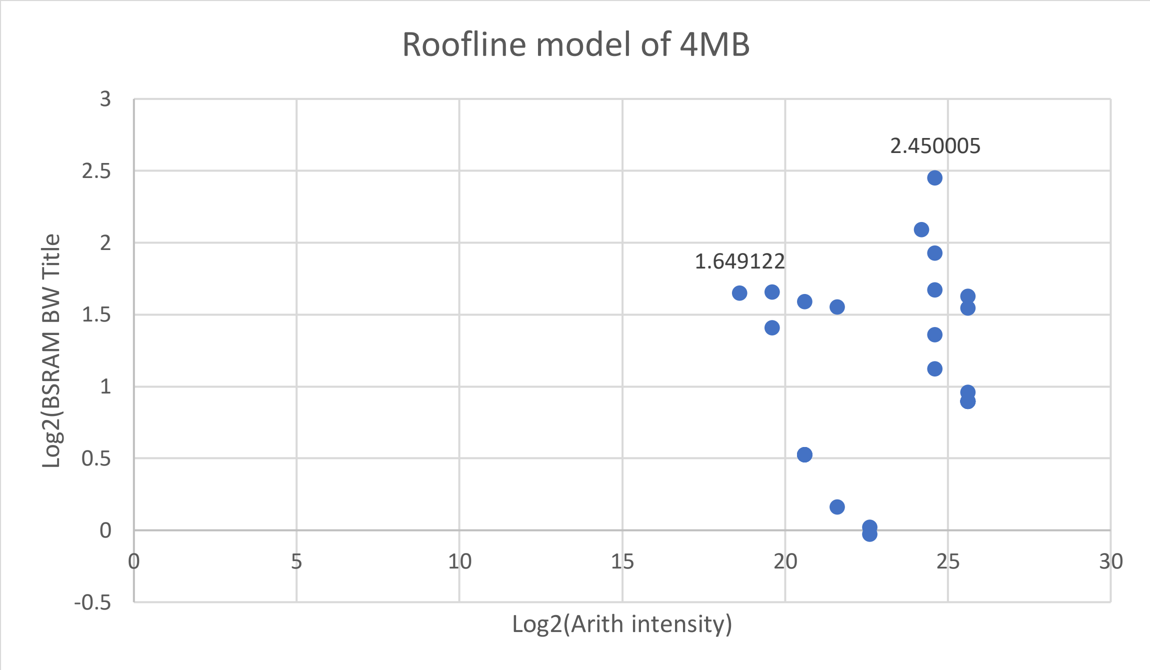
Part-a

Following roof line data base table:

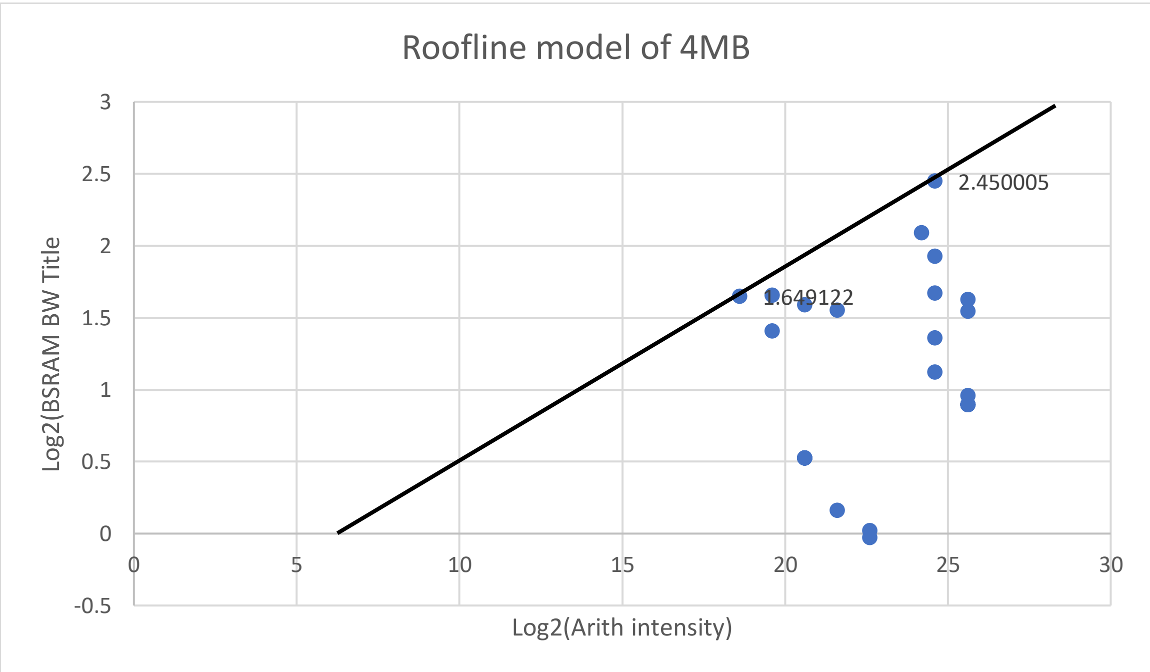
Arithmetic intensity = FH × FW × C × OH × OW × FN

SRAM bandwidth = IF(BW) + Filter(BW) + OF(BW)

Conv layer	Log2(Arithmetic intensity)	Log2(BW)
Conv1	24.19967	2.087885
Conv2	22.61471	-0.02746
Conv3	24.61471	2.450005
Conv4	21.61471	1.552135
Conv5	24.61471	1.927054
Conv6	22.61471	0.019258
Conv7	25.61471	1.544103
Conv8	20.61471	1.588161
Conv9	24.61471	1.357649
Conv10	21.61471	0.15911
Conv11	25.61471	0.956307
Conv12	19.61471	1.653524
Conv13	24.61471	1.119971
Conv14	20.61471	0.523563
Conv15	25.61471	0.89342
Conv16	20.61471	0.523563
Conv17	25.61471	0.89342
Conv18	20.61471	0.523563
Conv19	25.61471	0.89342
Conv20	20.61471	0.523563
Conv21	25.61471	0.89342
Conv22	20.61471	0.523563
Conv23	25.61471	0.89342
Conv24	18.61471	1.649122
Conv25	24.61471	1.66906
Conv26	19.61471	1.407436
Conv27	25.61471	1.626778



Part-b



The roofline BW so that no layer reaches the BW cap is the higher point intersecting the roofline, the point equal to 2.45 in actual BW its 5.46 Bytes per cycle to be more realistic 6 bytes per cycles.

Part-c

The answer is NO this systolic array in not very suited for this type of network since most layers need high memory BW, and the performance is very unstable from layer to layer for example the performance when working with depth-wise layers is very low and require very high SRAM BW.

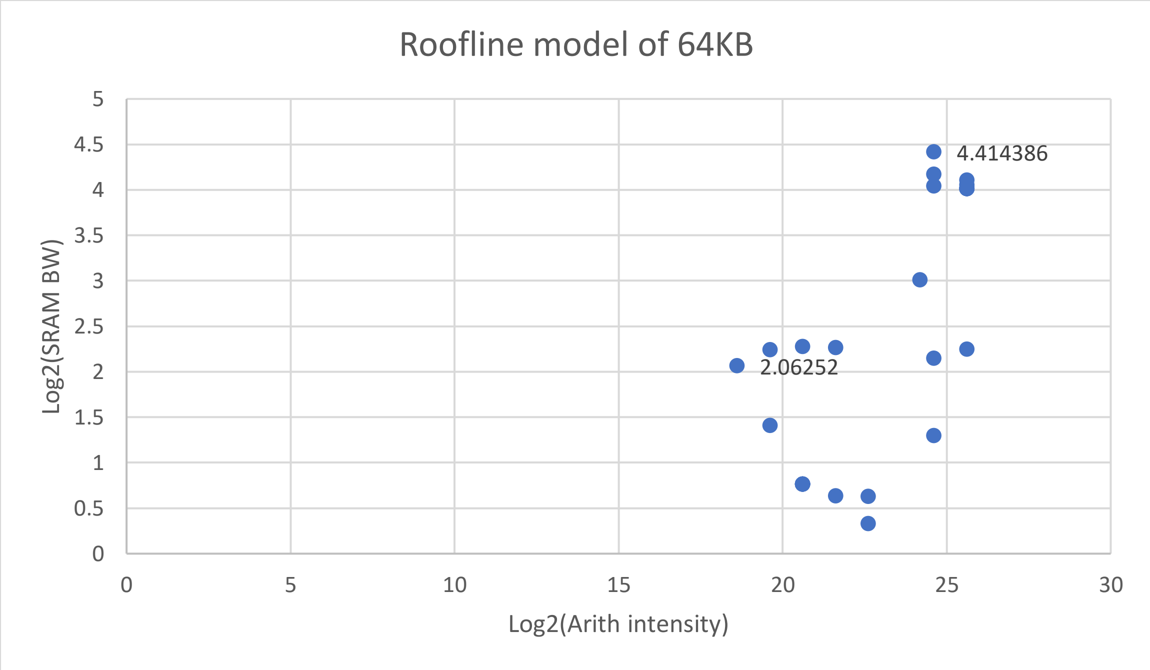
Part-d

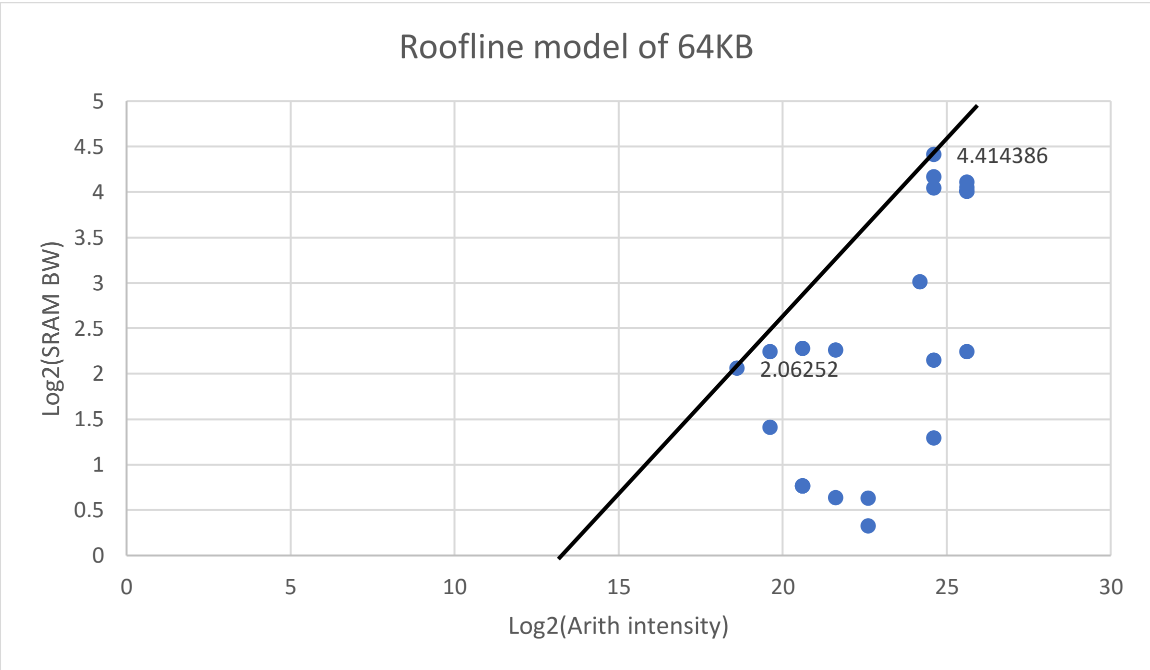
Given SRAM sizes of 64KB Following roof line data base table:

Arithmetic intensity = FH × FW × C × OH × OW × FN

SRAM bandwidth = IF(BW) + Filter(BW) + OF(BW)

Conv layer	Arithmetic intensity	Log2(BW)
Conv1	24.19967	3.01148
Conv2	22.61471	0.3272
Conv3	24.61471	4.414386
Conv4	21.61471	2.261218
Conv5	24.61471	4.167316
Conv6	22.61471	0.62716
Conv7	25.61471	4.107949
Conv8	20.61471	2.275979
Conv9	24.61471	4.042219
Conv10	21.61471	0.634193
Conv11	25.61471	4.050051
Conv12	19.61471	2.242923
Conv13	24.61471	1.294659
Conv14	20.61471	0.764361
Conv15	25.61471	4.009217
Conv16	20.61471	0.764361
Conv17	25.61471	4.009217
Conv18	20.61471	0.764361
Conv19	25.61471	4.009217
Conv20	20.61471	0.764361
Conv21	25.61471	4.009217
Conv22	20.61471	0.764361
Conv23	25.61471	4.009217
Conv24	18.61471	2.06252
Conv25	24.61471	2.148084
Conv26	19.61471	1.407436
Conv27	25.61471	2.243662

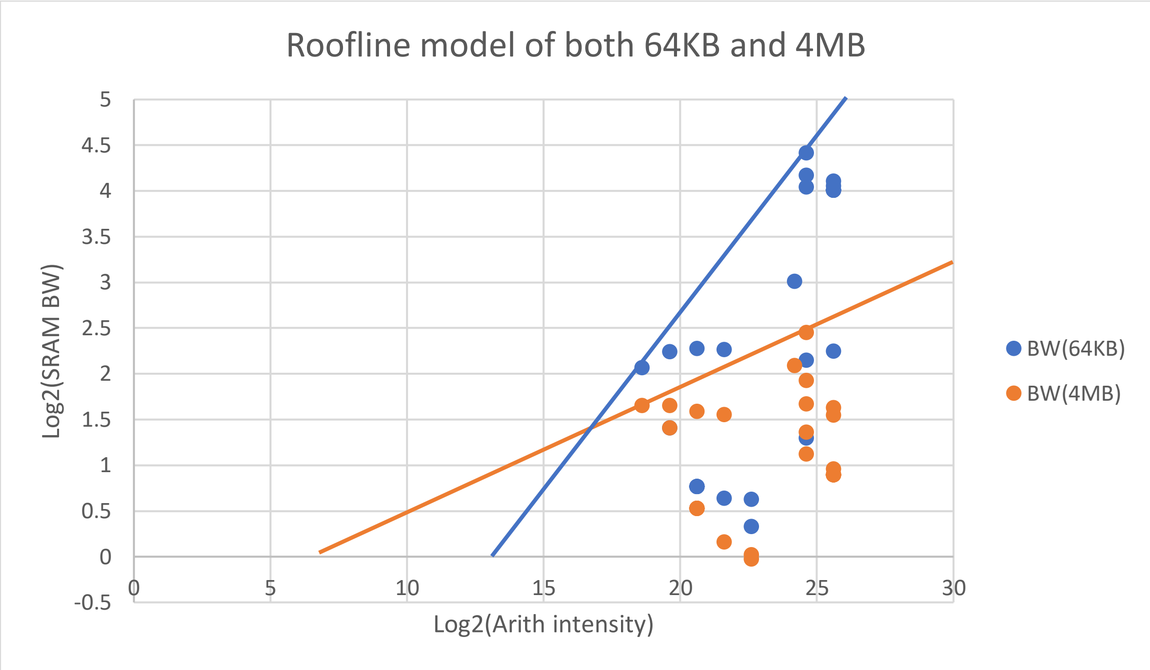




The roofline BW so that no layer reaches the BW cap is the higher point intersecting the roofline, the point equal to 4.41 in actual BW its 21.32 Bytes per cycle to be more realistic 22 bytes per cycles.

Part-e

To answer this question, we plug both points into a plot and compare.



We can see that the BW requirements have shifted up vertically when moving to SRAM's with 64KB of memory instead of 4MB of memory. This behavior is expected since with less memory space we would have to less freedom to store partial sums for later computation hence the dataflow would require to bring from the memory data more often so that each output element that is being calculated could be satisfied completely this requires many filter weight changes as well as many input elements changes causing to read/write the same element many times. unlike when having large amounts of memory space data flow could reuse certain elements and spare SRAM access. Also, since much space is available the systolic array computations could be saved as partial sums for later use hence the compiler has more freedom to optimize the BW reads/writes from the SRAM memories.