December 11, 2021

The University of Texas at Austin

SoC Final Project

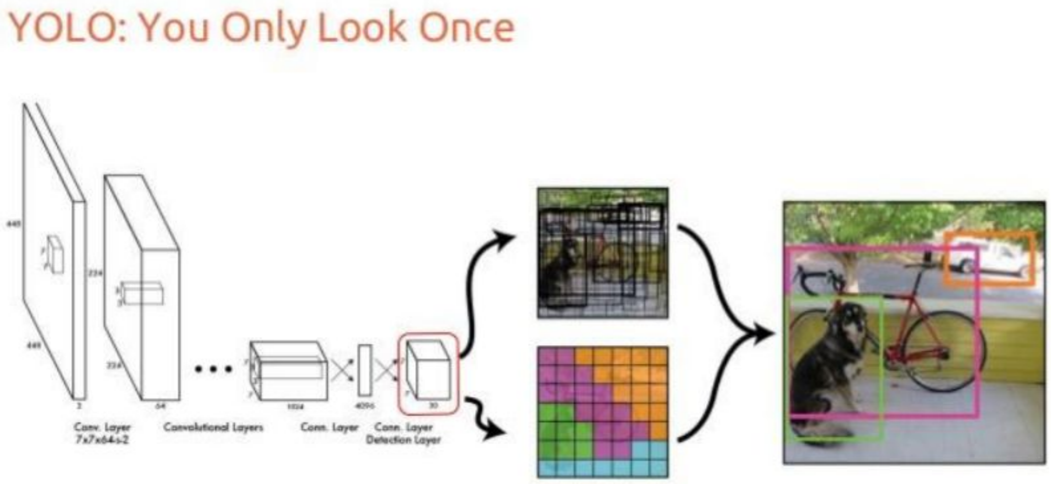
Abhijjith Venkkateshraj, Shreyas Ravishankar

Fall 2021

Contents

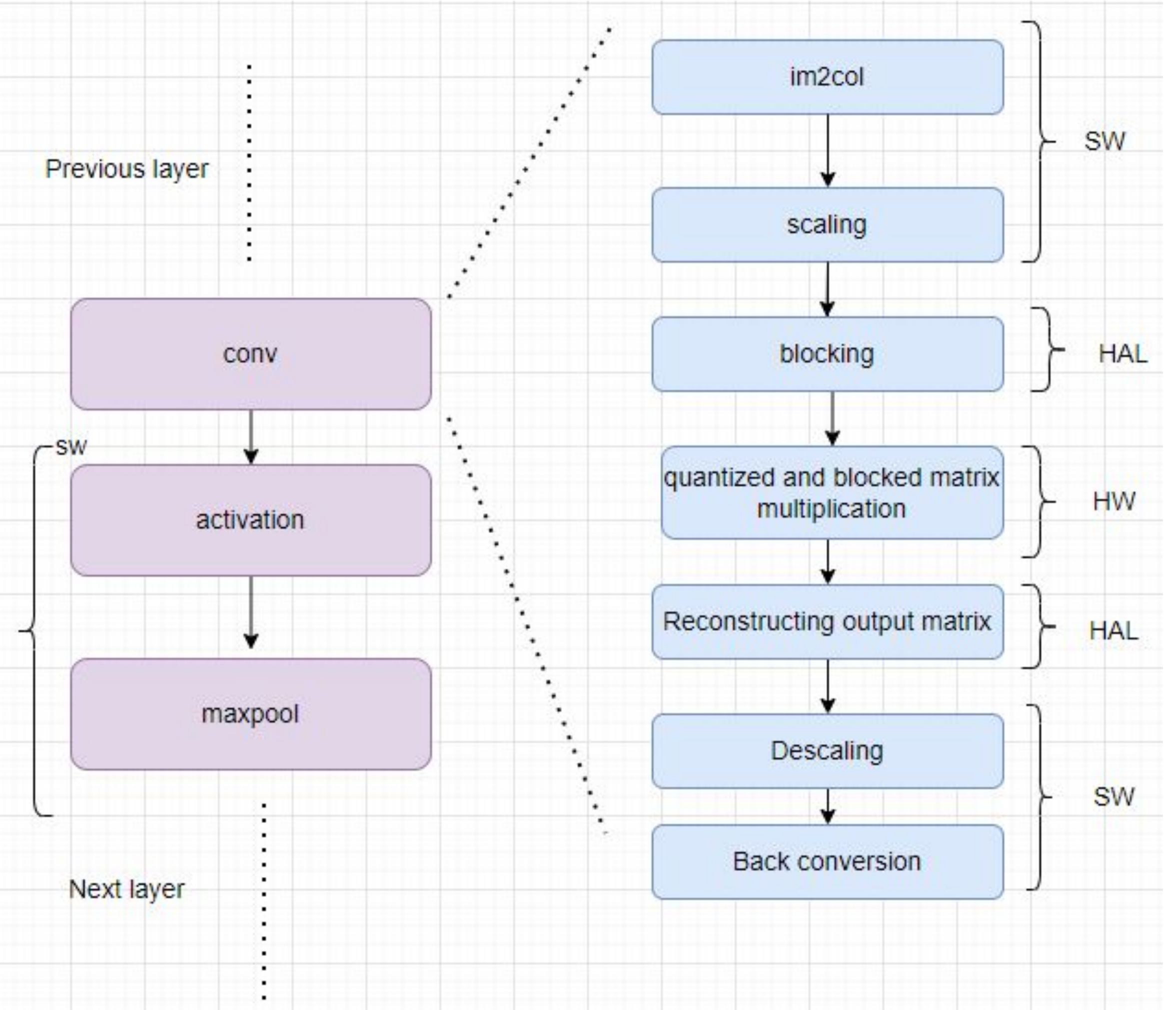
Product Overview

- Object detection using TinyYolo V3
- CNN based object detection
- Marketing requirements
  - Performance- 10fps
  - Accuracy- 50% mAP
  - Area- 0.5mm^2
  - Power < 8mW



System Design

HW/SW Partitioning

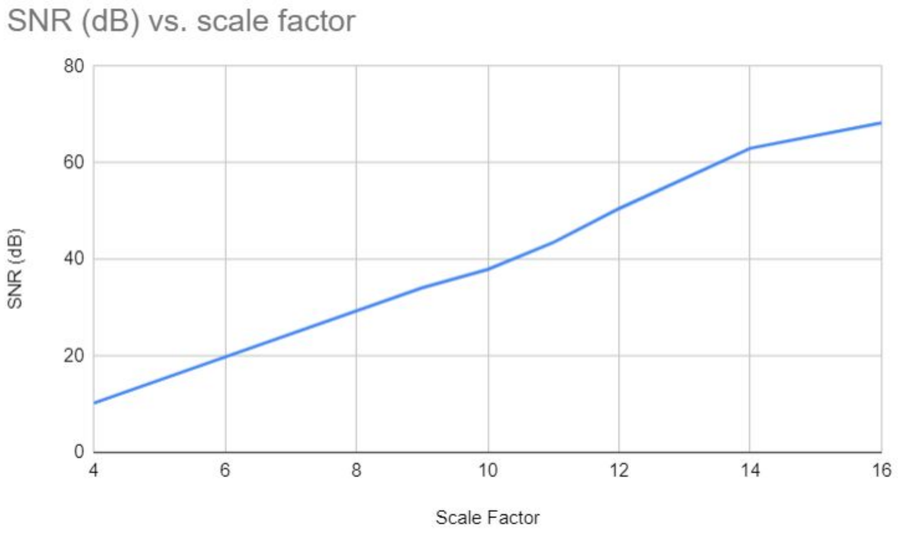


Algorithm Design, Profiling and Optimization

- Pre-processed the weights and images before passing it to the gemm_nn function.
- Involved scaling the arrays and converting them from floating point to fixed point just before the gemm call.
  - Dynamically allocated 2 fixed point arrays storing the weights and images

- Function call stack→ darknet.c (int main function)
  test_detector (detector.c)
  network_predict (network.c)
  forward_network (network.c)
  forward_convolutional_layer (convolutional_layer.c)
  gemm (gemm.c)
  gemm_cpu (gemm.c)
  gemm_nn (gemm.c)

Algorithm Design, Profiling and Optimization



- With a **scale factor of 9**, we got a **latency of 8.99s for gemm_nn** with decent accuracy.
  - SNR experiments might not be a true representation of the dataset used in darknet.
  - The better metric to decide scale factor would be analysing the tradeoff b/w prediction probabilities and latency on the actual image.
  - **Performance ~ 9.522 secs/image**

Algorithm Design, Profiling and Optimization

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 88.22      8.99     8.99     3694     0.00     0.00  gemm_nn
  1.77      9.17     0.18  8845488     0.00     0.00  rand_uniform
  1.47      9.32     0.15     1102     0.00     0.00  stbi__zbuild_huffman
  1.47      9.47     0.15       13     0.01     0.72  forward_convolutional_layee
r
```

```
data/dog.jpg: Predicted in 9522.000000 milli-seconds.
dog: 76%
bicycle: 29%
car: 65%
truck: 47%
truck: 66%
car: 43%
```

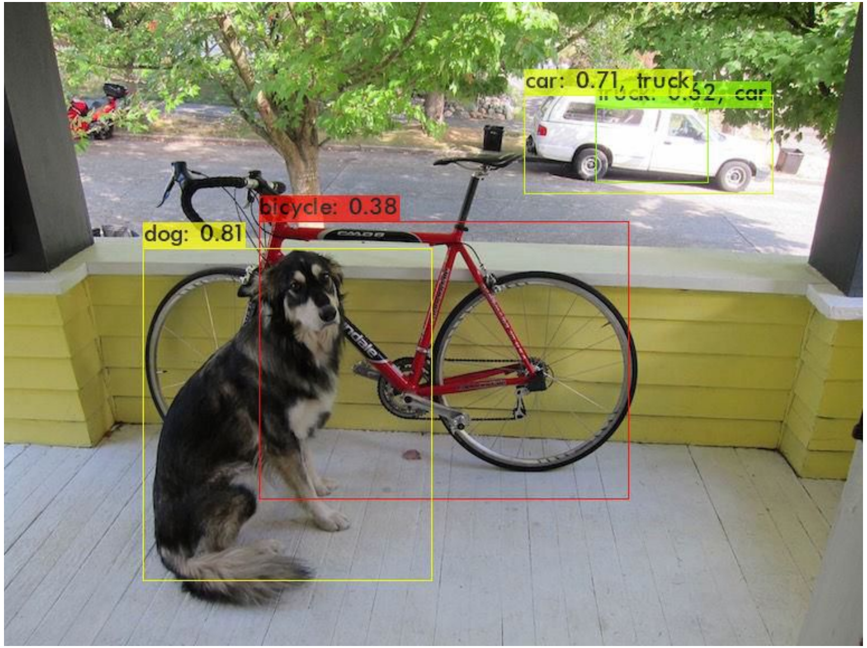
Accuracy

```
 for conf_thresh = 0.25, precision = 0.67, recall = 1.00, F1-score = 0.80
 for conf_thresh = 0.25, TP = 4, FP = 2, FN = 0, average IoU = 63.31 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall

 mean average precision (mAP) = 1.000000, or 100.00 % (4 classes)
Total Detection Time: 10 Seconds
```

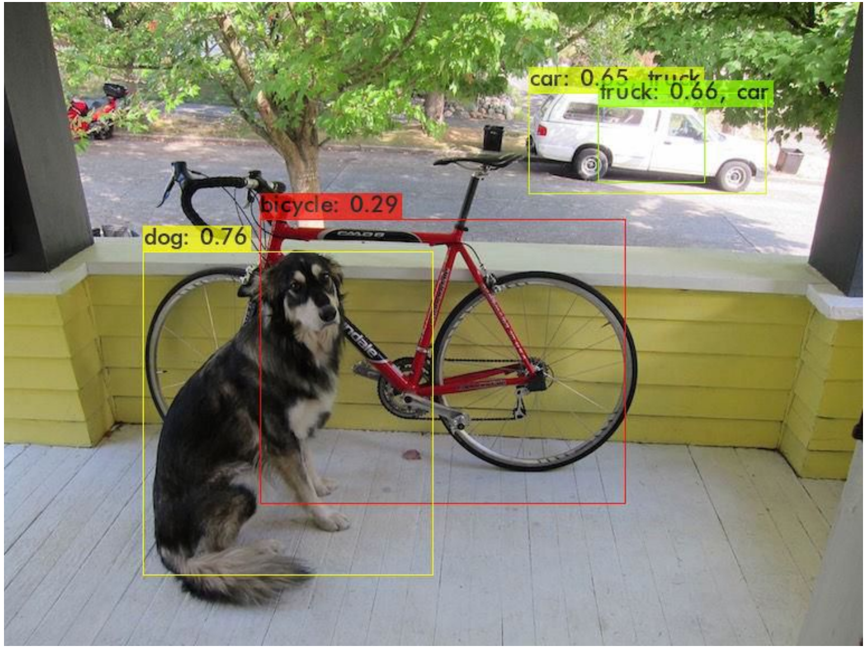
mAP

Algorithm Design, Profiling and Optimization

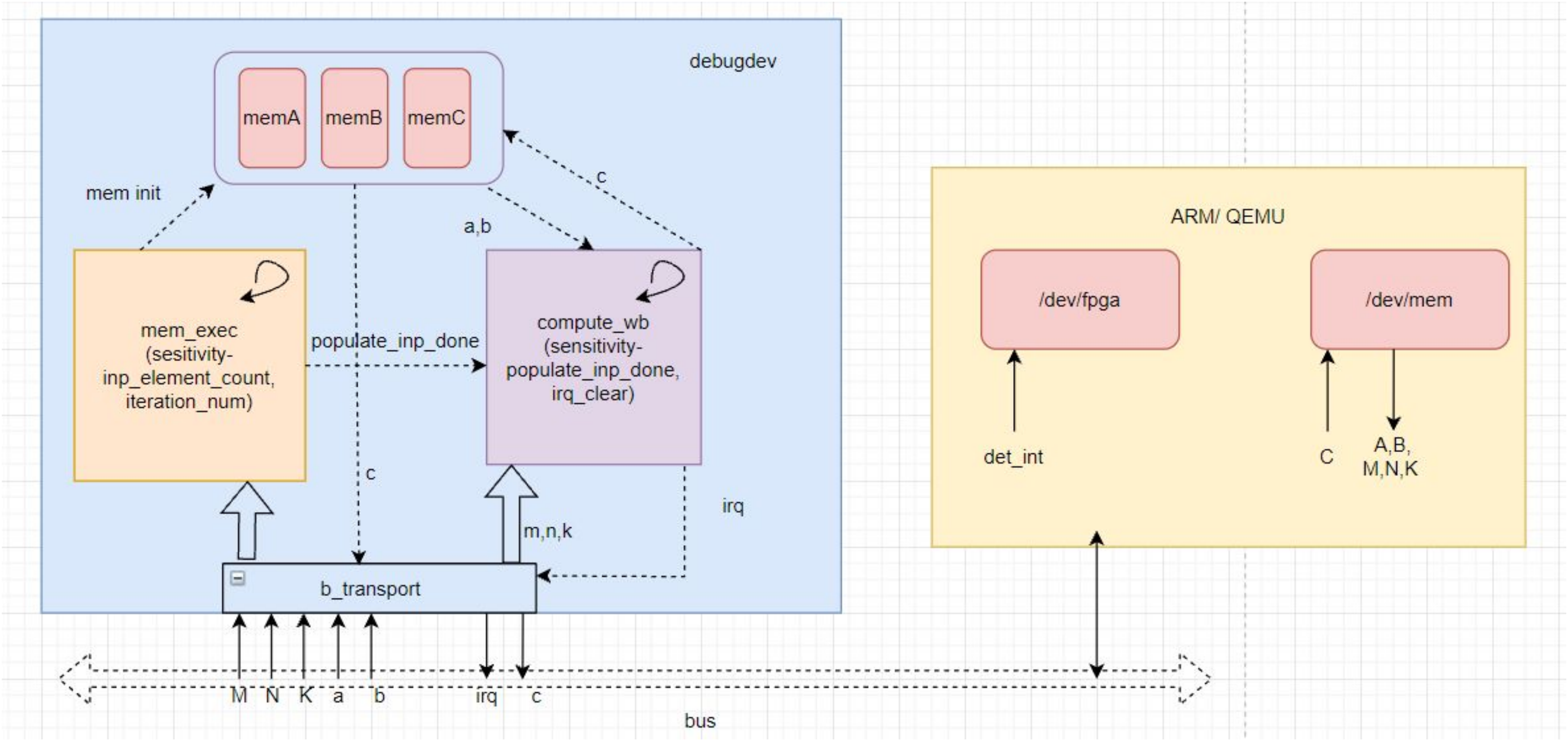
Floating Point



Fixed Point



ARM/QEMU + SystemC Co-Simulation



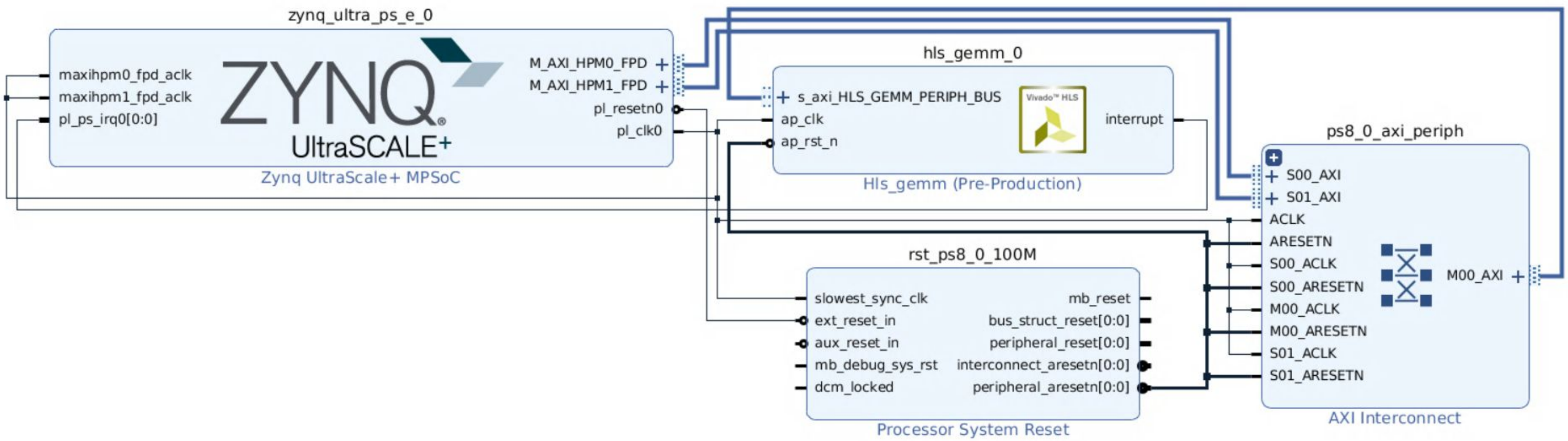
ARM/QEMU + SystemC Co-Simulation

Dimensions, weights and images sent using memory mapped I/O (/dev/mem) and used (/dev/fpga) for interrupts. ~ *3 min/gemm call.*

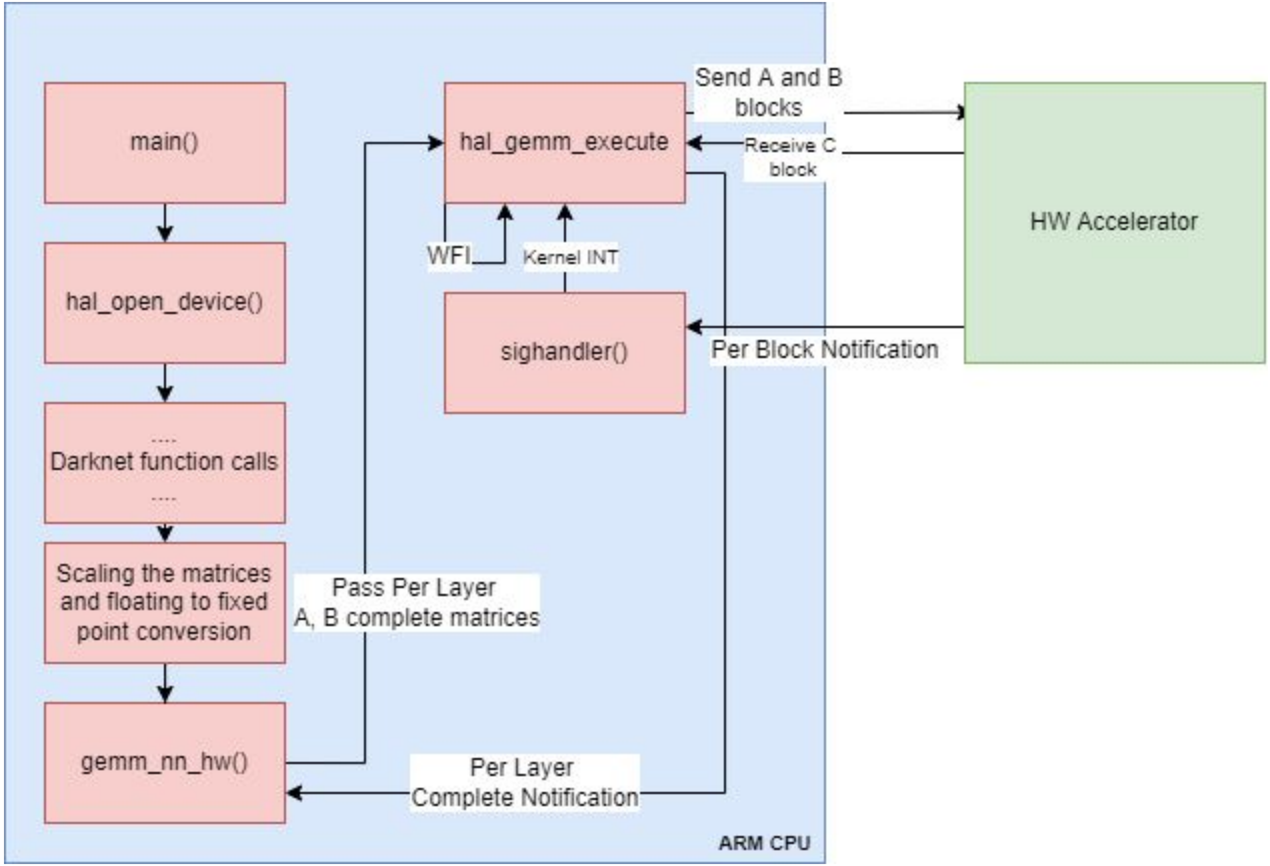
- Since the calls to the GEMM hardware module was taking significant time, we verified the functionality of the model using the below approach
- 1st iteration in each layer invokes gemm_nn_hw and the rest of the iterations in that layer invokes the quantized gemm_nn model (software-only).
- Integrated darknet. **Performance ~ 763 secs/image**

```
data/dog.jpg: Predicted in 763590.025000 milli-seconds.
dog: 76%
bicycle: 29%
car: 65%
truck: 47%
truck: 66%
car: 43%
```

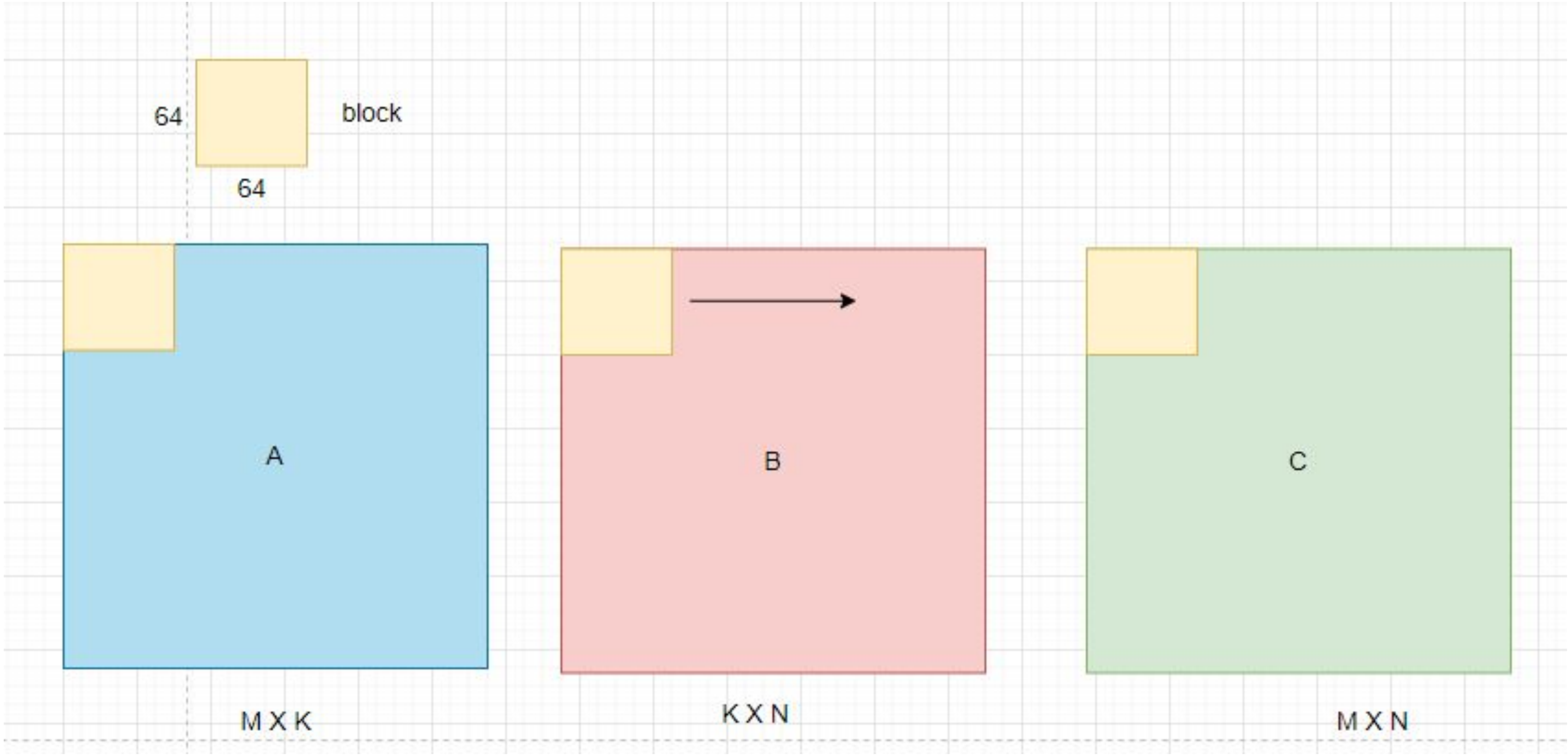
System Architecture



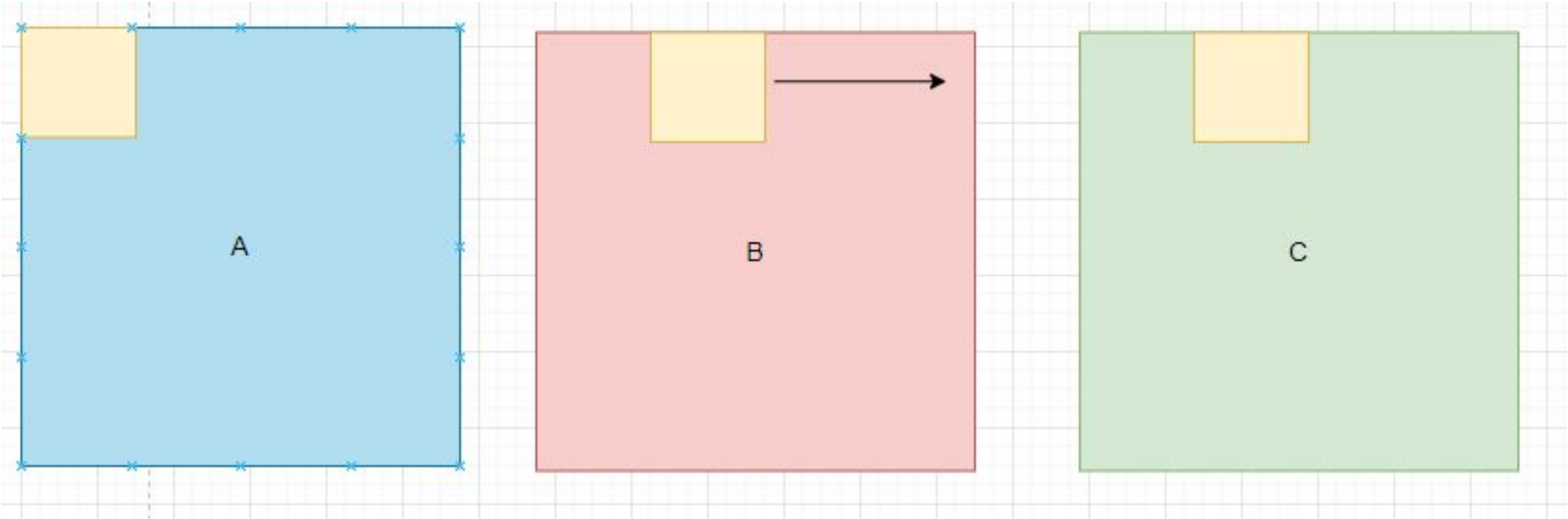
System/SW Architecture - Deep dive



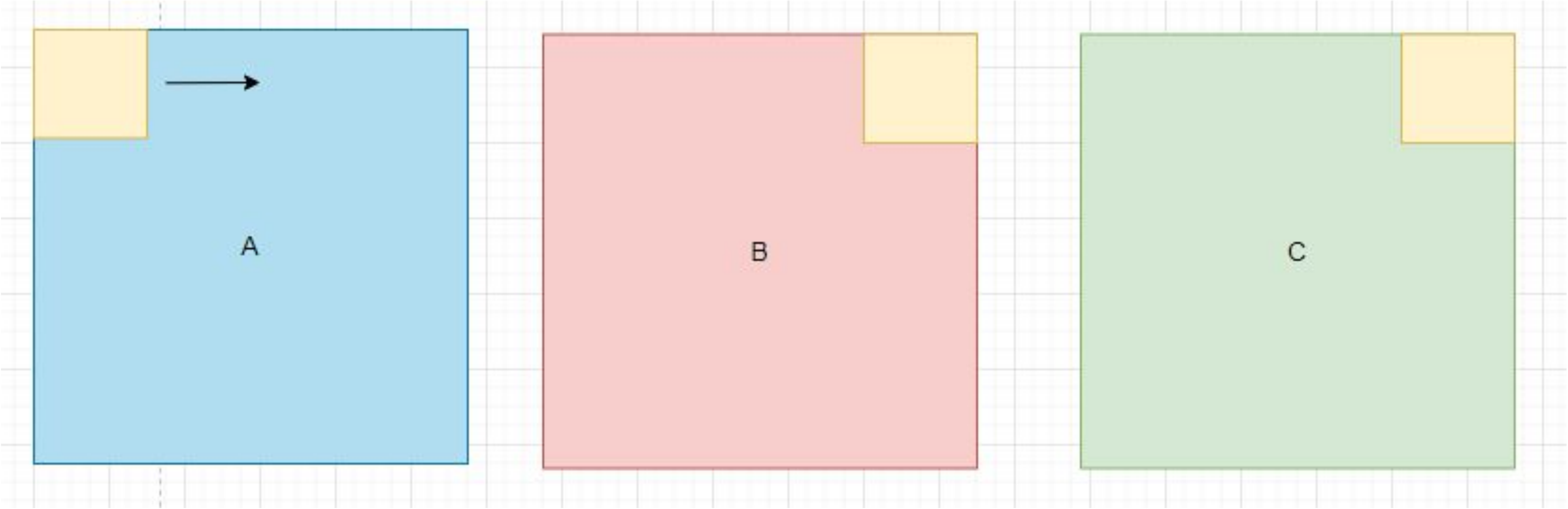
Tiling Approach



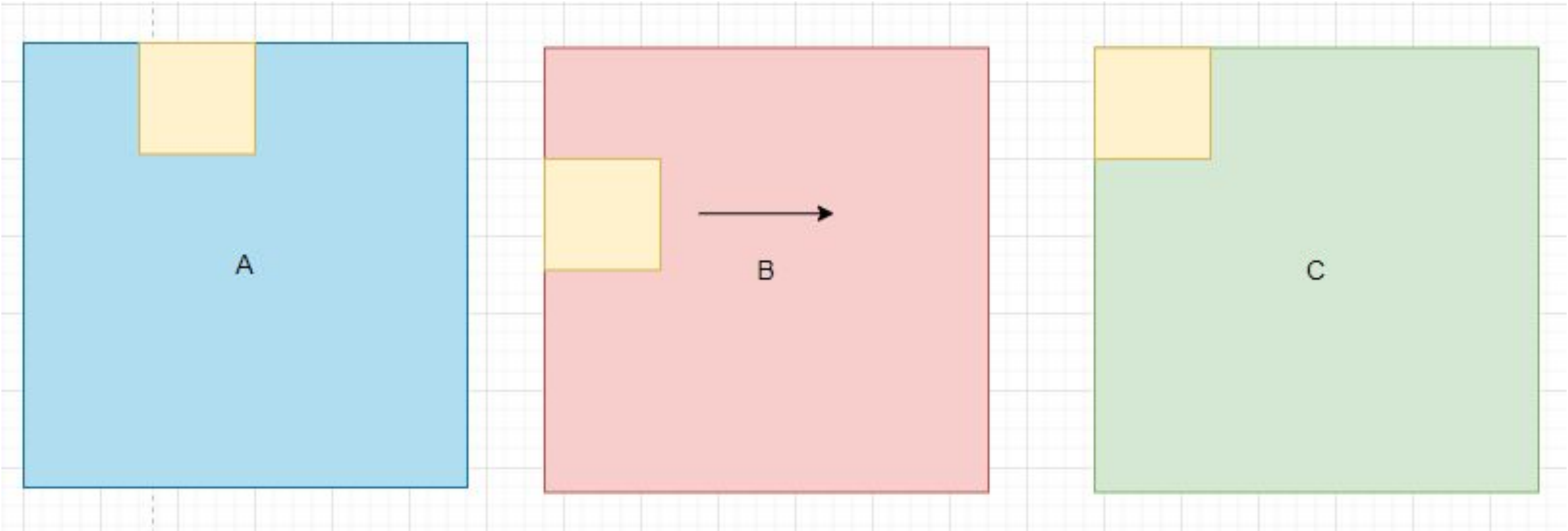
Tiling Approach



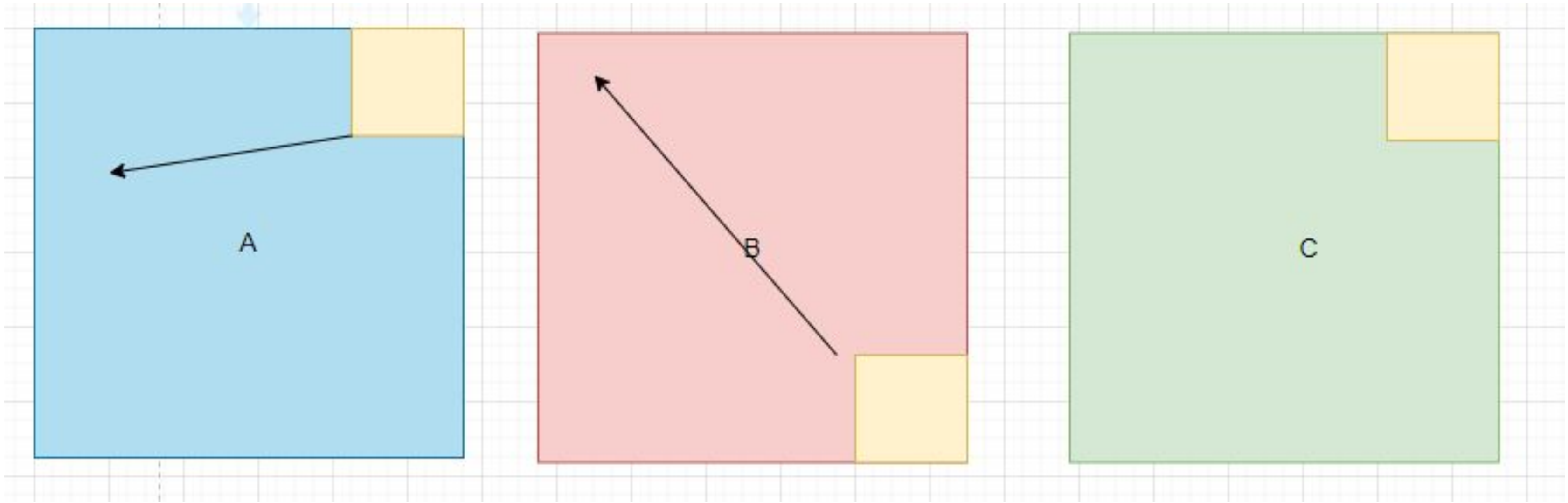
Tiling Approach



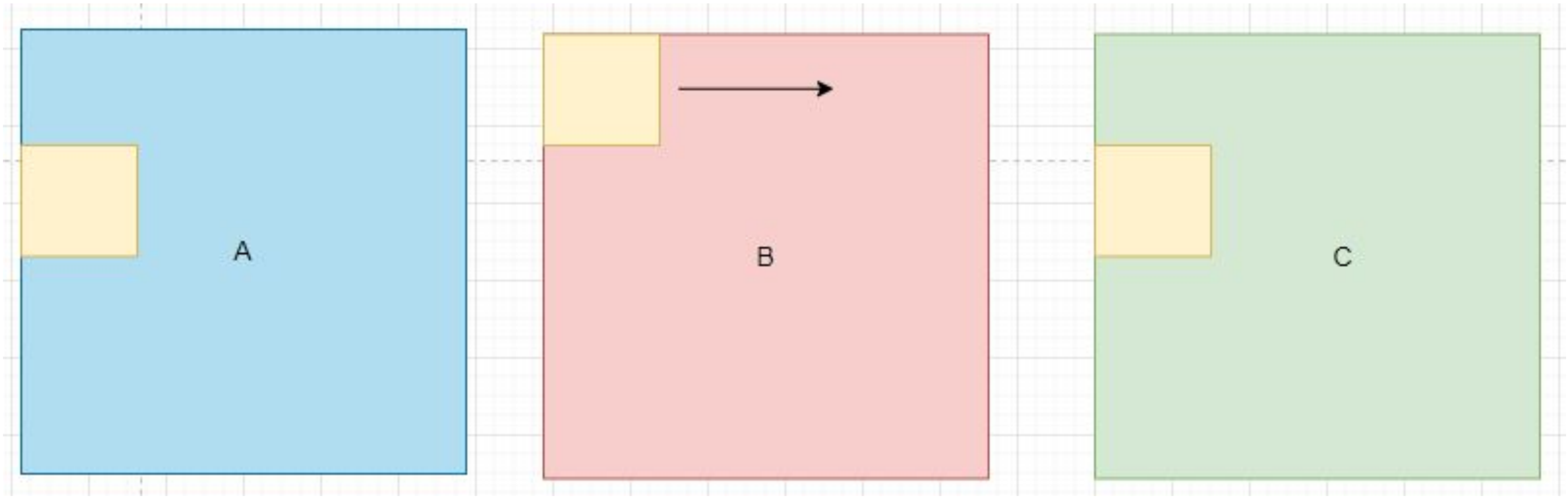
Tiling Approach



Tiling Approach



Tiling Approach



Testing infrastructure

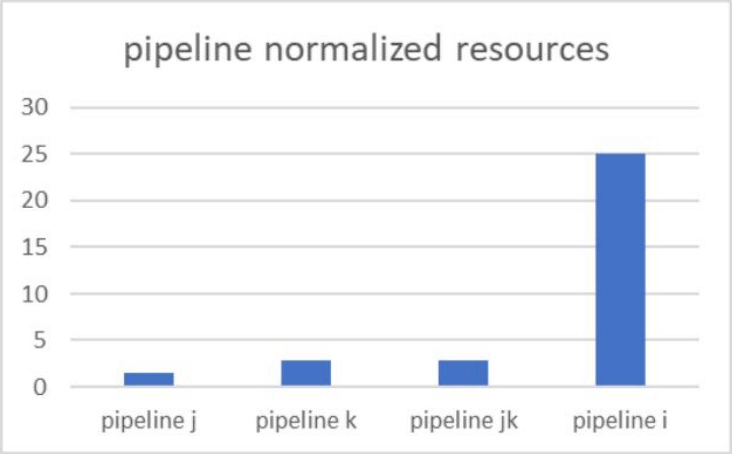
- Python testbench to test correctness of blocking
- Block testing
  - Whether individual blocks of weights and image are written correctly (via memory readback)
- Reconstruction/Layer testing
  - Whether intermediate O/P blocks are correctly reconstructed to the bigger Output matrix
  - Whether each layer’s output is correct or not (functionality between HW and SW GEMM)
- Network testing
  - Whether the final accuracy is as expected or not

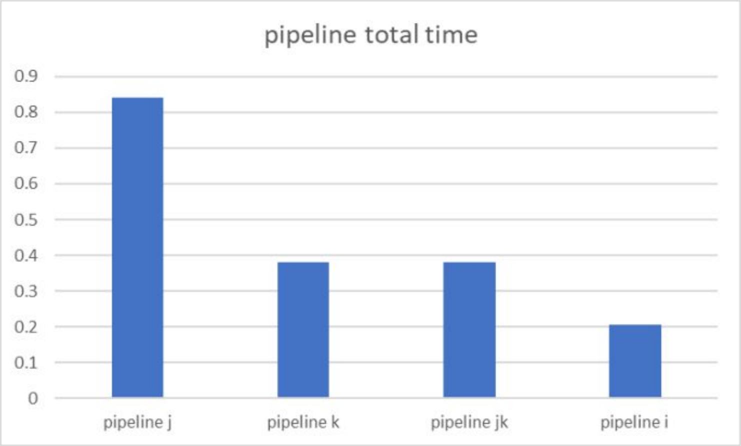
HW Design Approaches

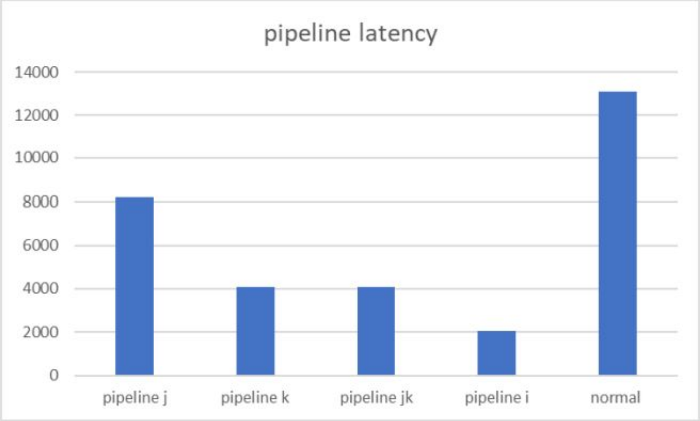
- Pipelining
- Loop unrolling
- Array Partitioning
- Bit Width Optimization

Pipelining

- Pipelining J and K loop doesn't make a difference, so pipelining K loop is sufficient.
- Pipelining the outermost loop gives the most optimization, but at the cost of resources.
- **Pipelining the middle loop (K) gives fair amount of optimization at the same time, saving us on resource usage.**

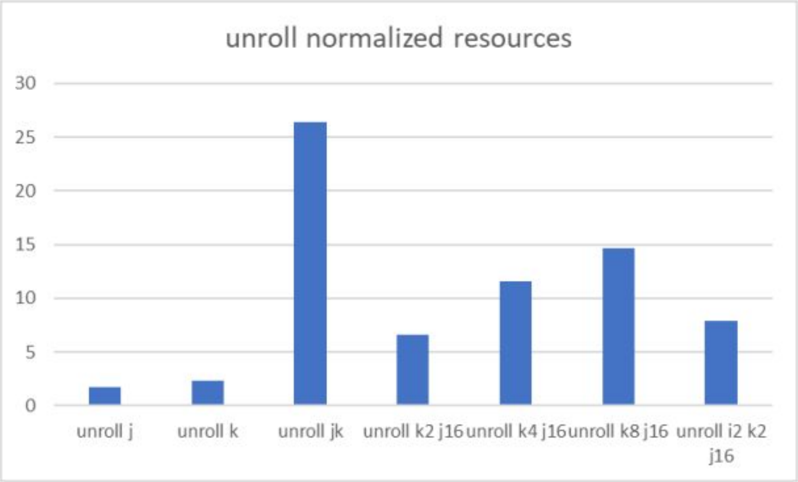


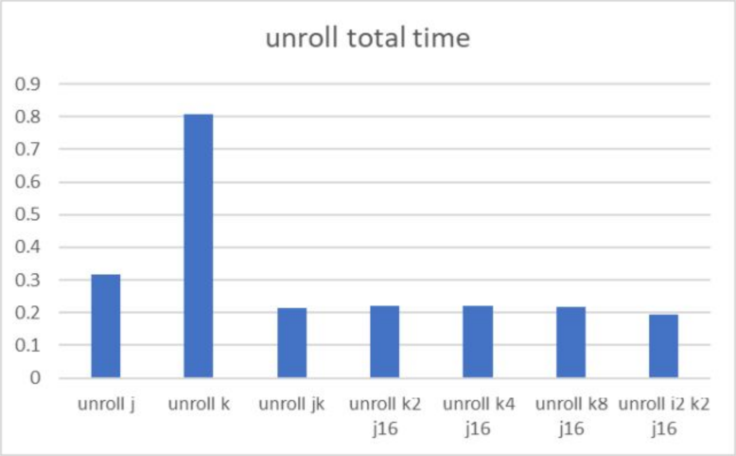


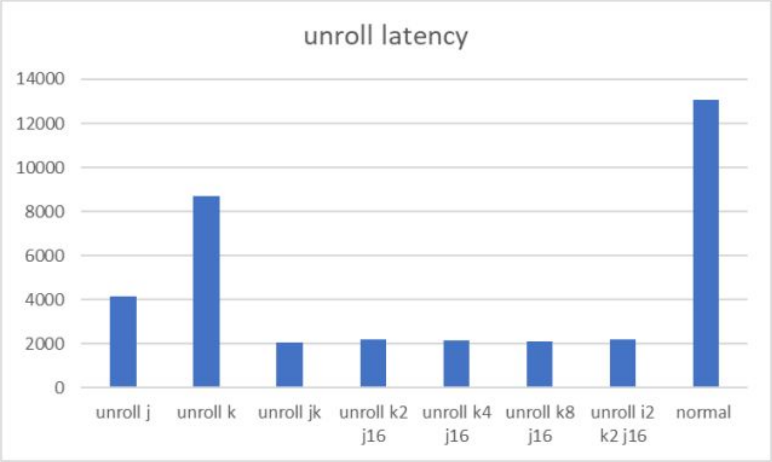


Loop unrolling

- Different loop factors for K shows that unrolling K by more than 2 factor doesn't make a difference in loop latency.
- **Unroll J fully, K by factor of 2.**

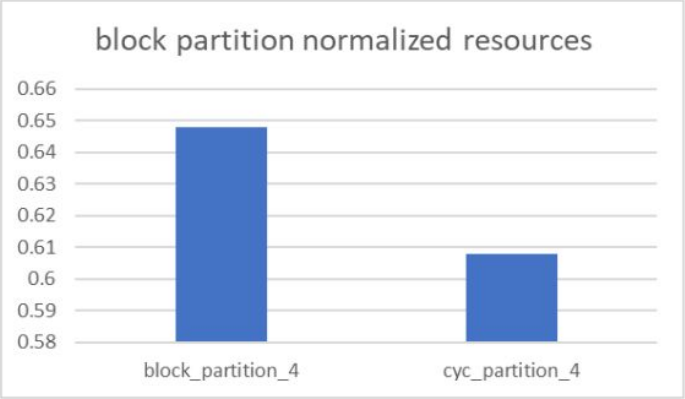


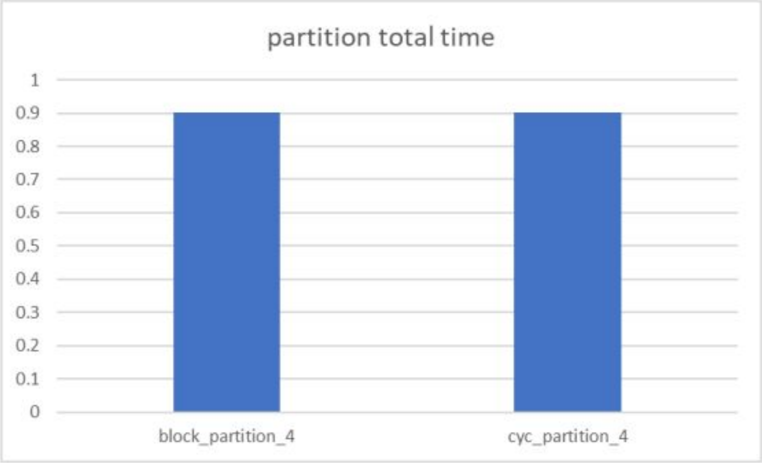


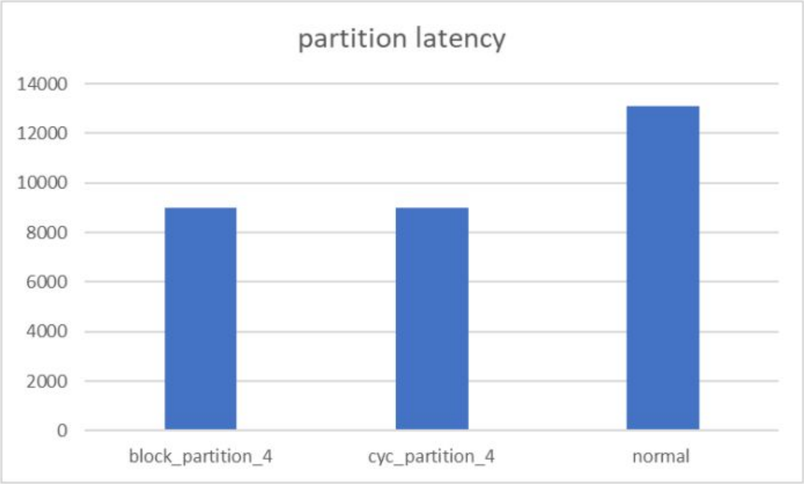


Array Partitioning

- Experimented with cyclic partitioning and block partitioning.
- Saw more benefits when combined with unrolling and pipelining.





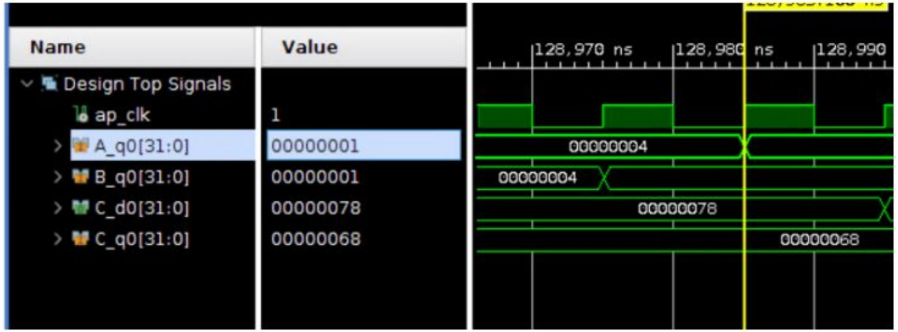


HLS Testing and Co-Simulation

- C simulation done for multiple tests (32), for random numbers generated within the required range based on scale factor.
- C RTL verification - Values observed at various points in the waveform to ensure correct functionality of multiply and accumulate function.

```
for (i = 0; i < num_tests; i++)
{
    for (j = 0; j < max_array_size; j++)
    {
        // Generate random test data, limit dynamic range to 12-bit
        A[j] = (rand() - RAND_MAX / 2) >> (8*sizeof(input_t) - 12);
        B[j] = (rand() - RAND_MAX / 2) >> (8*sizeof(input_t) - 12);
    }
    hls_gemm(A, B, C_hw, 1);
    ref_gemm(A, B, C_sw, 1);
```

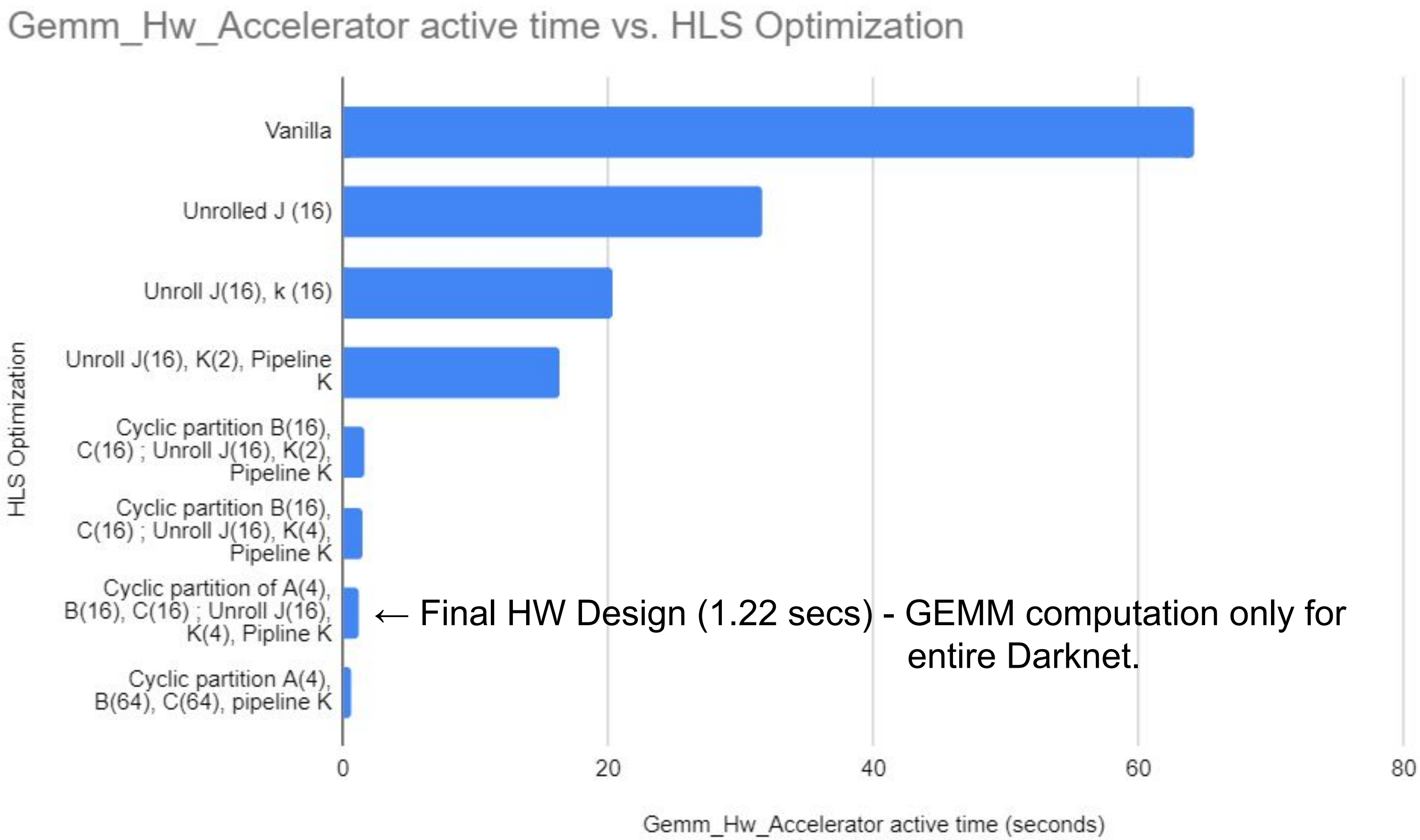
Snippet 1: A value is 0x4, B value is 0x4, and the current value of C is 0x68. So the new value of C would be 0x68 +((0x4)*((0x4))= 0x78.



Block Size Exploration

Dimensions (M,N,K)	**HW optimizations**	**Communication**	**No. of GEMM calls**	**Darknet Performance/image**
16, 16, 16	No optimizations	IOCTL	699232	500 s
16, 16, 16	unrolling and pipelining	MMAP	699232	210 s
64, 64, 64	unrolling and pipelining	MMAP	15191	**62 s**
256, 256,16	unrolling and pipelining	MMAP	6795	524 s
128, 128, 128	unrolling and pipelining	MMAP	3765	179 s

Choosing HW Design for 64*64 GEMM HW Accelerator



← Final HW Design (1.22 secs) - GEMM computation only for entire Darknet.

Final HW/SW Design

- **HW**
  - 64*64 GEMM in HLS
  - Loop unrolling J completely (innermost loop)
  - Pipelining and Loop unrolling K(4) (middle loop)
  - Cyclic partitioning A by 4 factor
  - B and C cyclically partitioned by 16 factor
- **SW**
  - HAL sends blocks of 64*64 using mmap in a cyclic partitioned manner
- Communication is still the bottleneck (consuming ~24 secs).

Performance ~ 25s/image

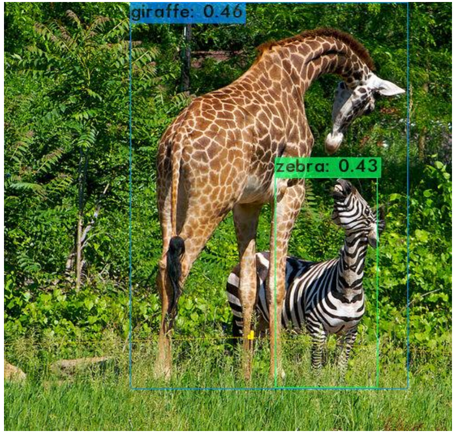
```
================================================================
+ Timing (ns):
    * Summary:
    +--------+-------+----------+------------+
    |  Clock | Target| Estimated| Uncertainty|
    +--------+-------+----------+------------+
    |ap_clk  |   4.00|     3.269|        0.50|
    +--------+-------+----------+------------+

+ Latency (clock cycles):
    * Summary:
    +-------+-------+-------+-------+---------+
    |    Latency    |    Interval   | Pipeline|
    |  min  |  max  |  min  |  max  |   Type  |
    +-------+-------+-------+-------+---------+
    |  16390|  24582|  16390|  24582|   none  |
    +-------+-------+-------+-------+---------+
```

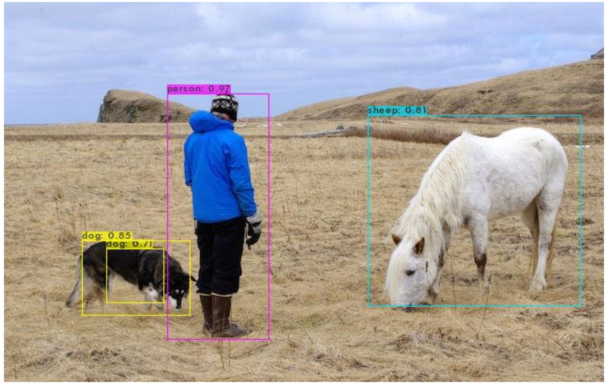
```
================================================================
== Utilization Estimates
================================================================
* Summary:
+-----------------+---------+-------+--------+-------+-----+
|       Name      | BRAM_18K| DSP48E|   FF   |  LUT  | URAM|
+-----------------+---------+-------+--------+-------+-----+
|DSP              |        -|      -|       -|      -|    -|
|Expression       |        -|      -|       0|   6442|    -|
|FIFO             |        -|      -|       -|      -|    -|
|Instance         |       72|    384|   24074|   8896|    -|
|Memory           |        -|      -|       -|      -|    -|
|Multiplexer      |        -|      -|       -|   2813|    -|
|Register         |        -|      -|    8330|      -|    -|
+-----------------+---------+-------+--------+-------+-----+
|Total            |       72|    384|   32404|  18151|    0|
+-----------------+---------+-------+--------+-------+-----+
|Available        |      432|    360|  141120|  70560|    0|
+-----------------+---------+-------+--------+-------+-----+
|Utilization (%)  |       16|    106|      22|     25|    0|
+-----------------+---------+-------+--------+-------+-----+
```

Yolov3 DEMO using Custom Accelerator HLS IP

Metrics/ Validation

Functional Verification and Accuracy

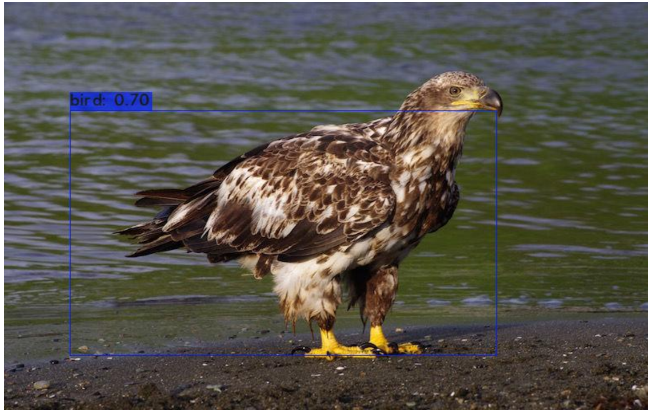





```
data/giraffe.jpg: Predicted in 25431.975
giraffe: 46%
zebra: 43%
```

```
data/person.jpg: Predicted in 25572.405000 milli-seconds.
dog: 85%
dog: 71%
person: 97%
sheep: 81%
```

```
data/eagle.jpg: Predicted in 25417.720000 milli-seconds.
bird: 70%
```



mAP measurement for 2 images

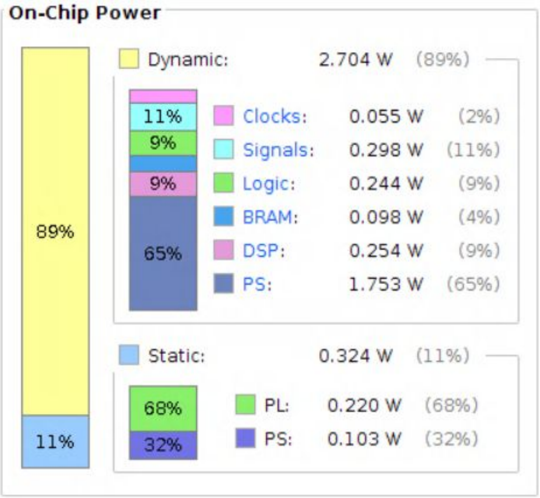
```
 for conf_thresh = 0.25, precision = 0.71, recall = 1.00, F1-score = 0.83
 for conf_thresh = 0.25, TP = 5, FP = 2, FN = 0, average IoU = 67.34 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP) = 1.000000, or 100.00 % (5 classes)
Total Detection Time: 52 Seconds
```

Power and Timing Estimation

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**3.028 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**33.3°C**
Thermal Margin:	66.7°C (24.1 W)
Effective θJA:	2.7°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium

Launch Power Constraint Advisor to find and fix invalid switching activity



Design Timing Summary

Setup

Worst Negative Slack (WNS):	2.303 ns
Total Negative Slack (TNS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	55991

Hold

Worst Hold Slack (WHS):	0.010 ns
Total Hold Slack (THS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	55991

Pulse Width

Worst Pulse Width Slack (WPWS):	3.500 ns
Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	16727

All user specified timing constraints are met.

DPU based Design (Demo)

- State of the art IP (Xilinx) for ML applications
- Vitis- AI toolchain to deploy applications.

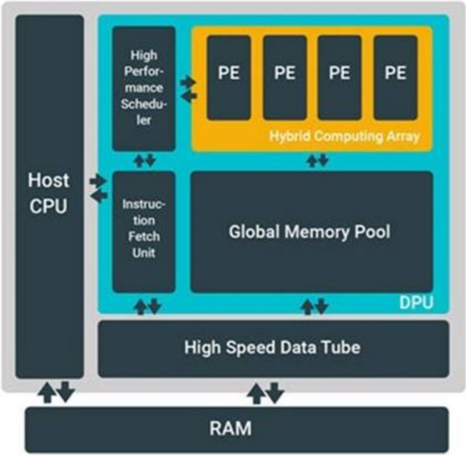
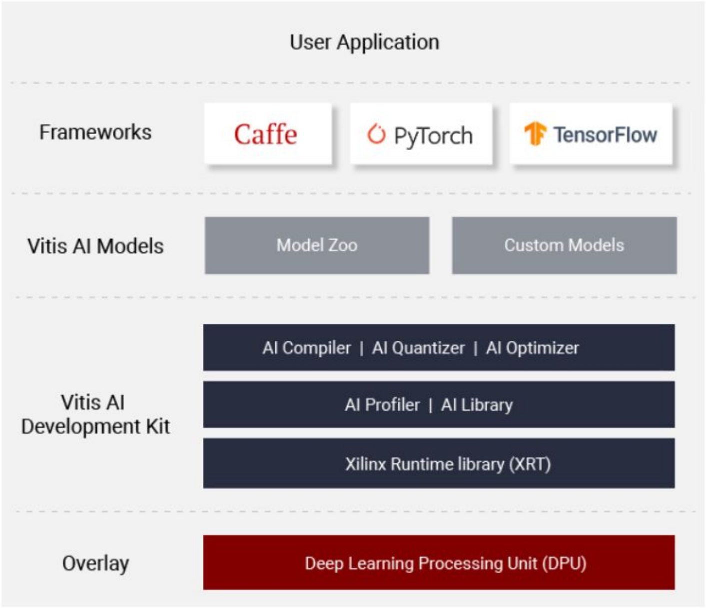
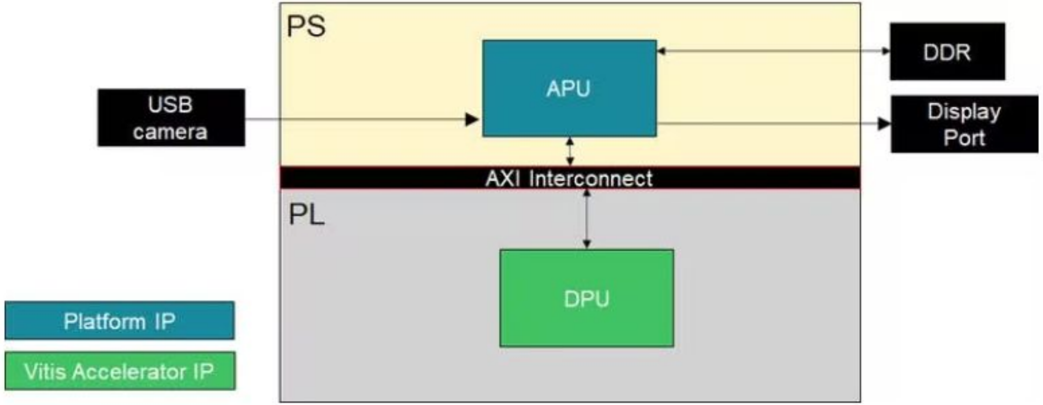


Figure 13: **DPUCZDX8G Architecture**





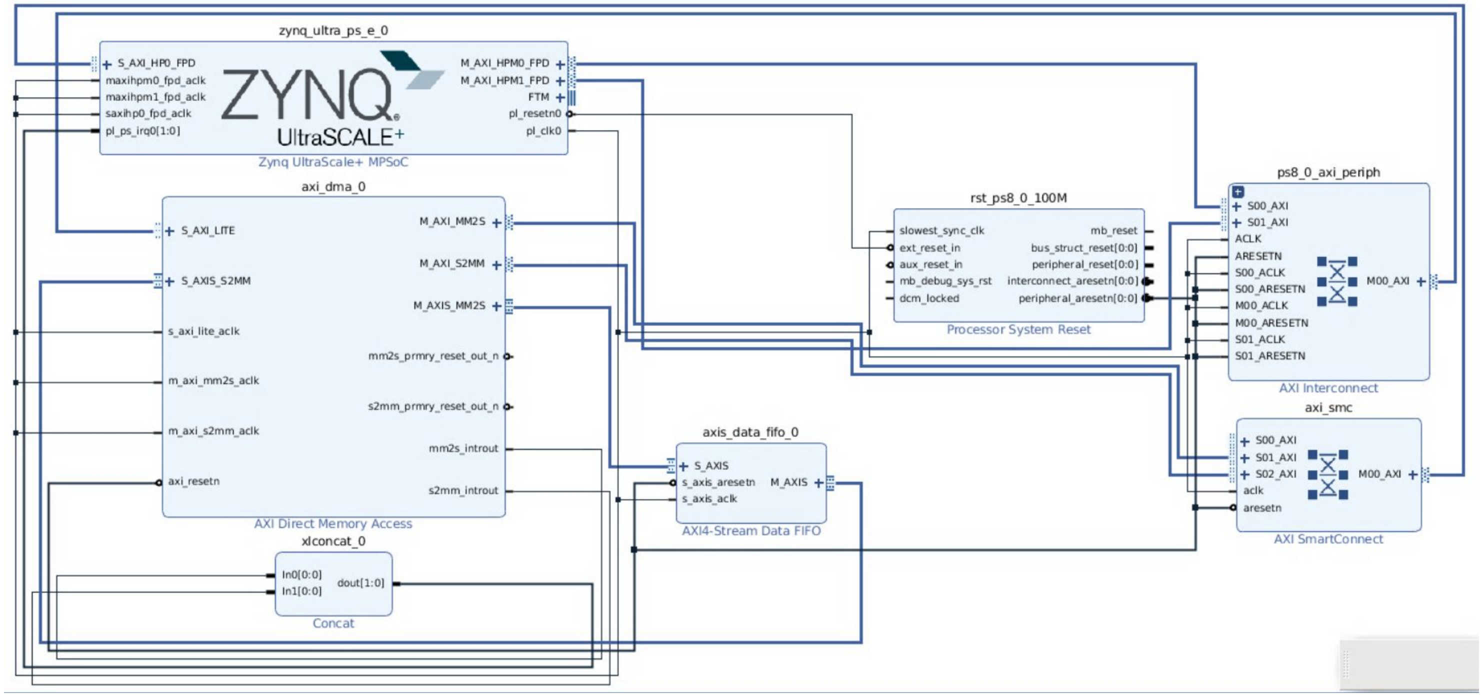
Acknowledgements

Thank you for your help

- Dr. Andreas Gerstlauer
- Alex
- Other Teams

Creative Backlog

DMA based design (Accelerator Master) to communicate with DRAM



- The processor is interrupted only after a layer's gemm computation instead of every block gemm.in
- C matrix (output of every layer) is thus read in one shot from DRAM after the interrupt is received.
- Streaming interface b/w DMA and HW accelerator.
- Processor can rearrange the weights matrix in DRAM, in parallel when gemm is being computed (avoiding rearranging overhead for weights after gemm computation)
- Petalinux is failing during configuration. (Wasn't able to push THIS design to completion)

Final HW Design for a 16*16 GEMM (Lab 3)

- Pipelining Middle (k) Loop
- Unroll the Middle (k) Loop by a factor of 2
- Completely Unroll the lower (j) loop (factor: 16)
- ap_int<16> for inputs and lda; and ap_int<32> for outputs

Frequency - 6.753ns
Cycles- 2178
Total time (16*16)- 1.46us

```
================================================================
== Performance Estimates
================================================================
+ Timing (ns):
    * Summary:
    +--------+-------+---------+-----------+
    |  Clock | Target| Estimated| Uncertainty|
    +--------+-------+---------+-----------+
    |ap_clk  |  10.00|    6.753|       1.25|
    +--------+-------+---------+-----------+

+ Latency (clock cycles):
    * Summary:
    +-----+-----+-----+-----+---------+
    |  Latency  |  Interval | Pipeline|
    | min | max | min | max |   Type  |
    +-----+-----+-----+-----+---------+
    | 2178| 2178| 2178| 2178|   none  |
    +-----+-----+-----+-----+---------+
```

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	35	-	-	-
Expression	-	0	0	1987	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	512	-
Register	-	-	1429	-	-
Total	0	35	1429	2499	0
Available	432	360	141120	70560	0
Utilization (%)	0	9	1	3	0

Software and Hardware Design

Approach 1:

- HW: 16*16 accelerator which implements naive gemm (No optimizations)
- SW: HAL to break down weights/images and send 16*16 matrices over IOCTL
- Tiling logic first implemented and tested in Python
  - Considered dimensions perfectly divisible by block size
  - Added zero padding logic for dimensions not perfectly divisible (zero padding the original array vs padding the block)
  - **Making sure that we are not sending weights repetitively per GEMM layer.**
- **Performance ~500 secs/image**

Approach 2:

- SW: Calling gemm_nn with M=m instead of M=1 for each gemm call (diverted from supposedly parallel thread execution).
- HW: 16*16 accelerator which implements naive gemm (No optimizations)
- **Performance ~310 secs/image**

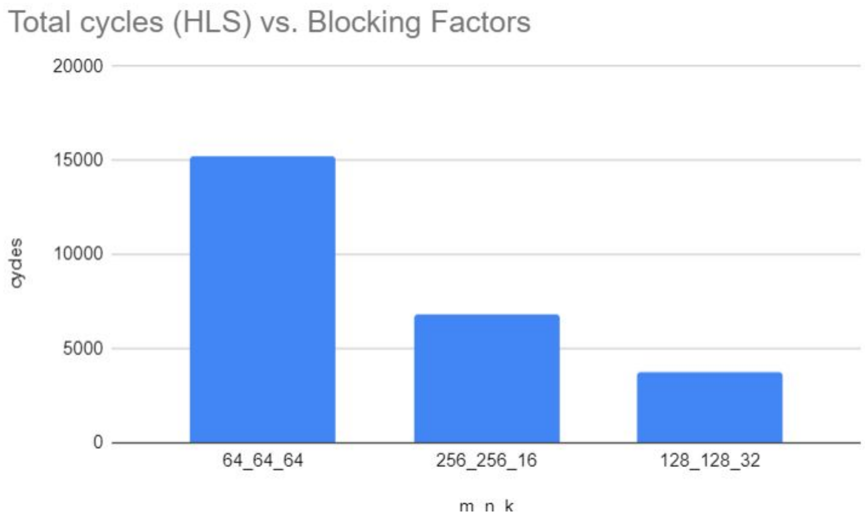
Approach 3:

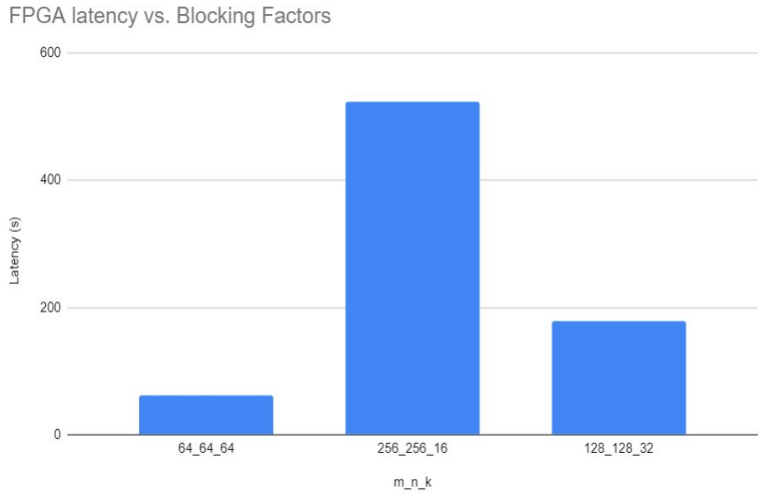
- HW: Took the most optimal design obtained from Lab 3 experiments
  - 16*16
  - Unroll innermost loop (J) fully, Unroll middle (K) loop by factor of 2
  - Pipeline the K loop
  - **Performance ~ 306 sec/image** (communication is still the bottleneck)
- SW optimization- Changed ioctl to mmap
  - **Performance ~ 210 sec/image**

Approach 4: Using varying block size to reduce the communication cost

M	N	K	# GEMM calls	Time(s)
64	64	64	15191	62.3
256	256	16	6795	524
128	128	128	3765	179

- # of GEMM calls not directly proportional to execution time.
- Squared block matrices seemed to perform better as compared to uneven shaped block matrices.
- **64*64 is the chosen block size based on analysis**


Total cycles (HLS) vs. Blocking Factors
20000
15000
10000
5000
0
cycles
64_64_64
256_256_16
128_128_32
m_n_k


FPGA latency vs. Blocking Factors
600
400
200
0
Latency (s)
64_64_64
256_256_16
128_128_32
m_n_k