Accelerating inference of YOLOv3-tiny model using FPGA

Anshuman
Arvind Nataraj Sivasankar
Shreyas Tater

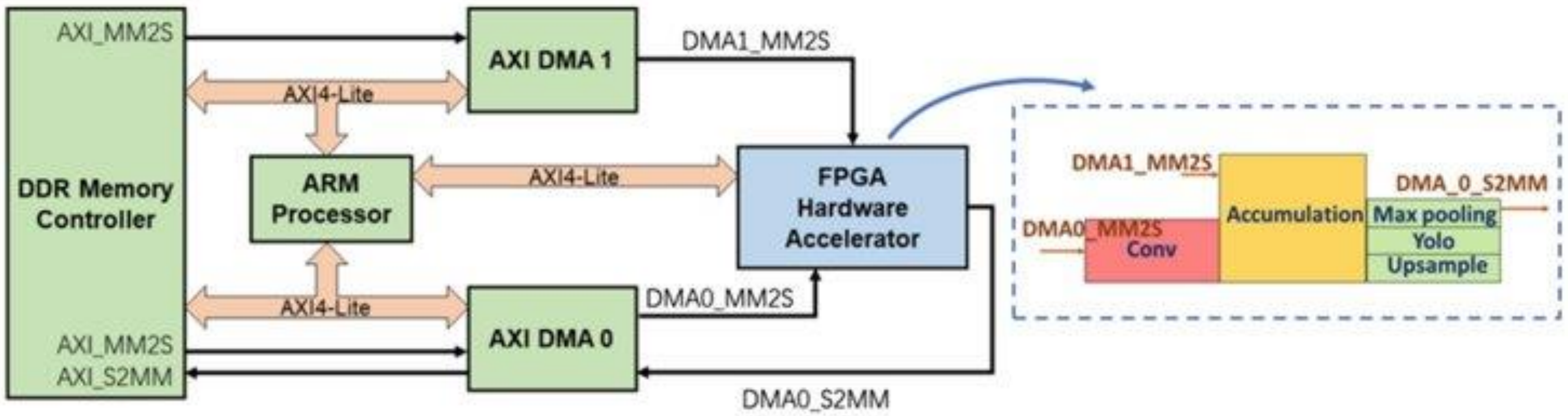
Georgia Tech

Problem

- Deploy YOLOv3-tiny on FPGA with minimal reduction in object detection precision
- Limited resources on FPGA (off chip memory required to store parameters and partial DNN inference computation results)
- Low Latency for real time object detection applications on embedded systems

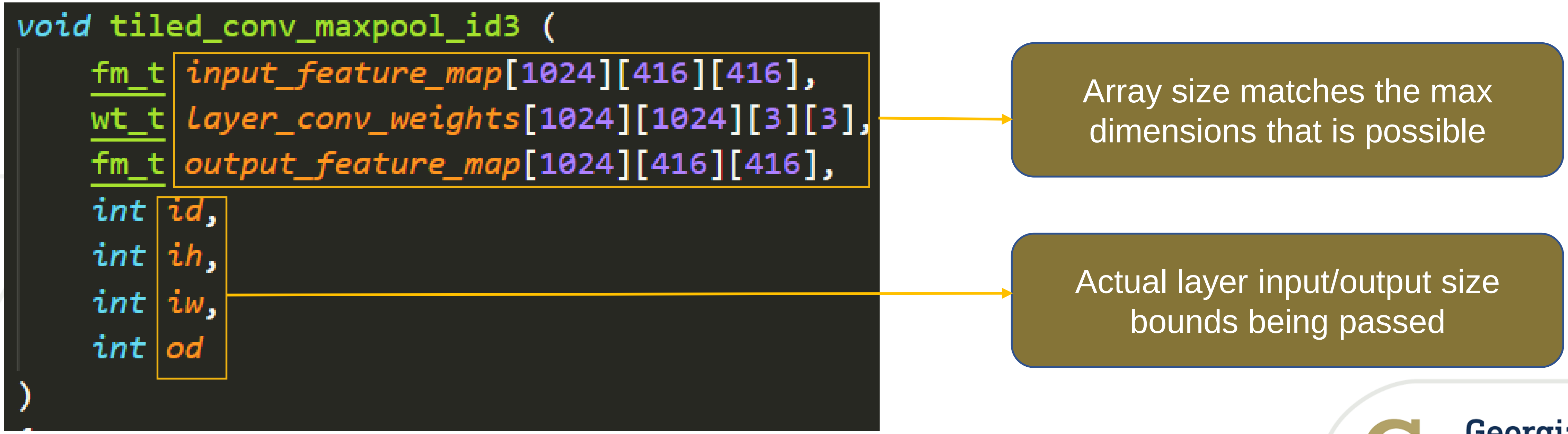
Design

- The FPGA accelerator consists of a three-stage pipeline.
- The first stage of the pipeline supports the execution of the convolution layer, whose output is accumulated in the second stage of the pipeline.
- Depending on the network structure that is executed at a given time, the accumulation results are sent for further processing in the Max pooling, Upsample or Yolo layer



Approach

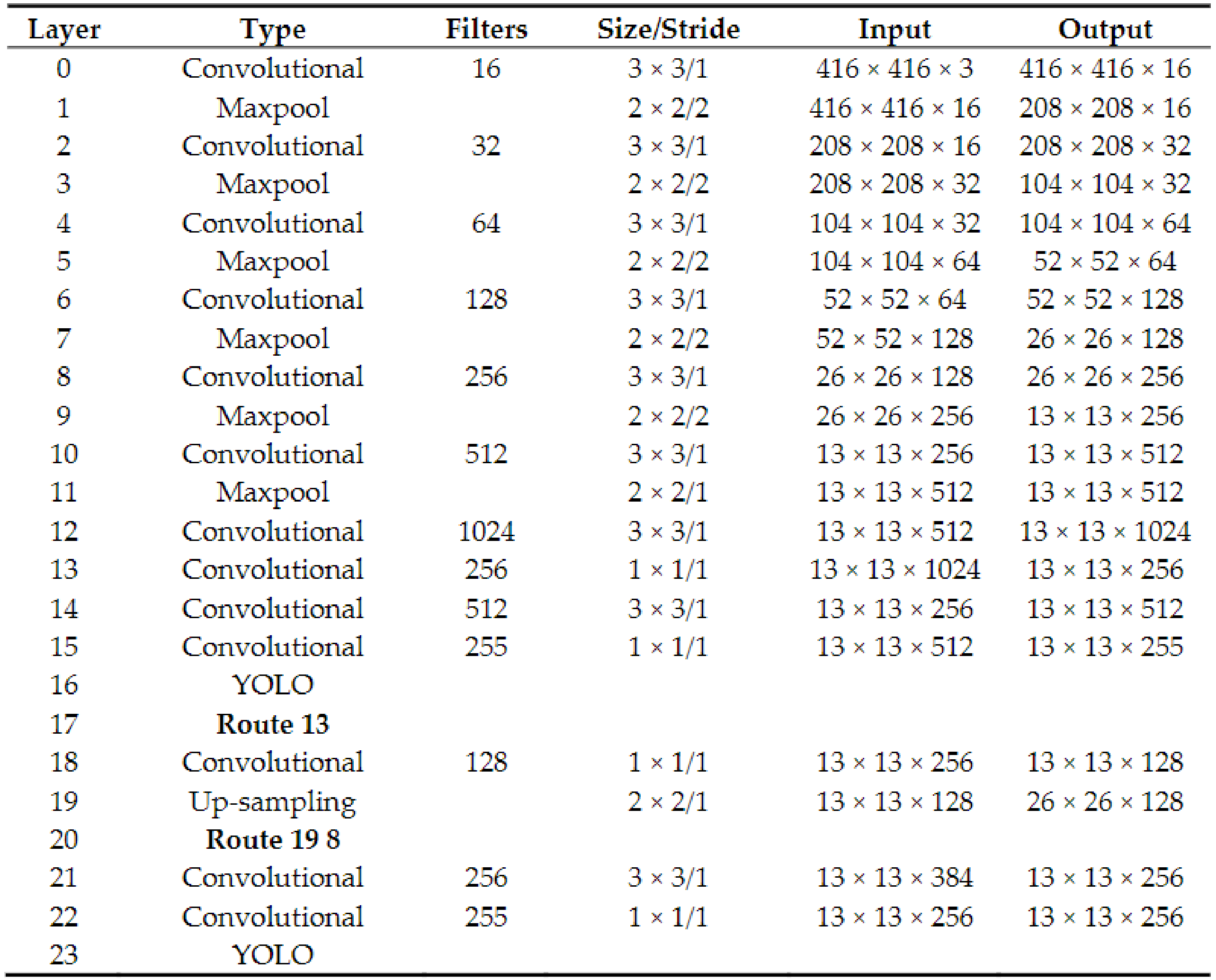
- Worked on a golden C code based off available literature - used *pointers.
- Redid golden C code using Lab2 as base and made modifications to get single layer convolution running for a variable input size.



```
void tiled_conv_maxpool_id3 (
        fm_t input_feature_map[1024][416][416],
        wt_t layer_conv_weights[1024][1024][3][3],
        fm_t output_feature_map[1024][416][416],
        int id,
        int ih,
        int iw,
        int od
)
```

Array size matches the max dimensions that is possible
Actual layer input/output size bounds being passed

Approach

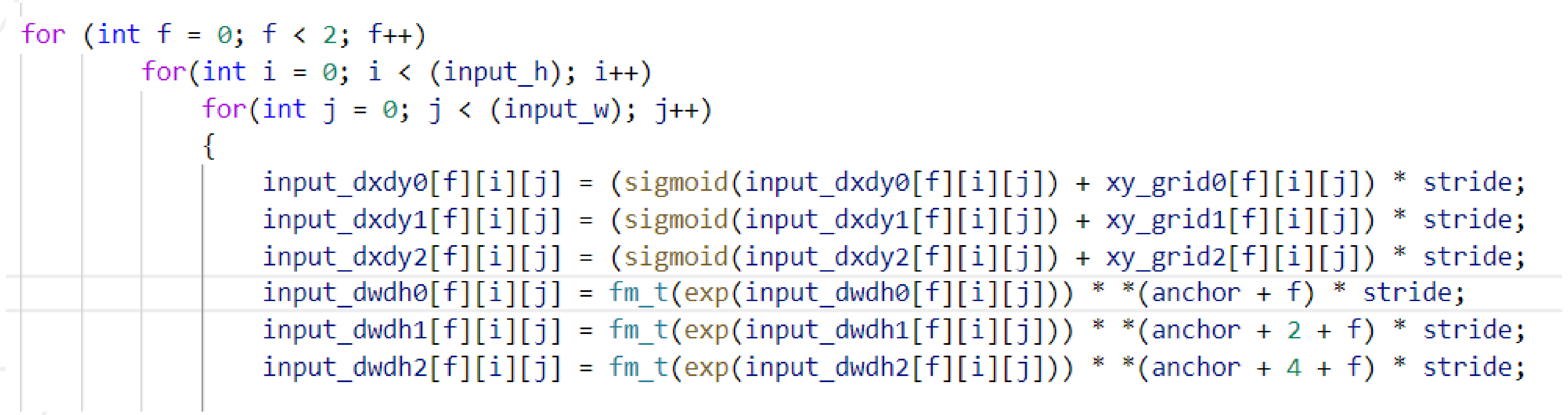


Layer	Type	Filters	Size/Stride	Input	Output
0	Convolutional	16	3 × 3/1	416 × 416 × 3	416 × 416 × 16
1	Maxpool		2 × 2/2	416 × 416 × 16	208 × 208 × 16
2	Convolutional	32	3 × 3/1	208 × 208 × 16	208 × 208 × 32
3	Maxpool		2 × 2/2	208 × 208 × 32	104 × 104 × 32
4	Convolutional	64	3 × 3/1	104 × 104 × 32	104 × 104 × 64
5	Maxpool		2 × 2/2	104 × 104 × 64	52 × 52 × 64
6	Convolutional	128	3 × 3/1	52 × 52 × 64	52 × 52 × 128
7	Maxpool		2 × 2/2	52 × 52 × 128	26 × 26 × 128
8	Convolutional	256	3 × 3/1	26 × 26 × 128	26 × 26 × 256
9	Maxpool		2 × 2/2	26 × 26 × 256	13 × 13 × 256
10	Convolutional	512	3 × 3/1	13 × 13 × 256	13 × 13 × 512
11	Maxpool		2 × 2/1	13 × 13 × 512	13 × 13 × 512
12	Convolutional	1024	3 × 3/1	13 × 13 × 512	13 × 13 × 1024
13	Convolutional	256	1 × 1/1	13 × 13 × 1024	13 × 13 × 256
14	Convolutional	512	3 × 3/1	13 × 13 × 256	13 × 13 × 512
15	Convolutional	255	1 × 1/1	13 × 13 × 512	13 × 13 × 255
16	YOLO				
17	**Route 13**				
18	Convolutional	128	1 × 1/1	13 × 13 × 256	13 × 13 × 128
19	Up-sampling		2 × 2/1	13 × 13 × 128	26 × 26 × 128
20	**Route 19 8**				
21	Convolutional	256	3 × 3/1	13 × 13 × 384	13 × 13 × 256
22	Convolutional	255	1 × 1/1	13 × 13 × 256	13 × 13 × 256
23	YOLO				

- Golden – C
  - Implemented all layers – YOLO, Convolution, Max Pooling, Upsample and Route
  - Used Lab 2 code as base to develop code.
  - Created a 24-layer network

Code Snippets

- YOLO C-Model code



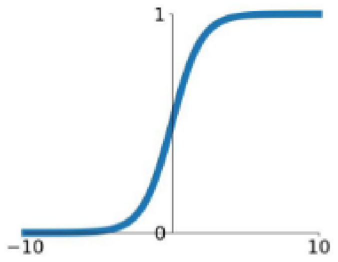
```
for (int f = 0; f < 2; f++)
        for(int i = 0; i < (input_h); i++)
            for(int j = 0; j < (input_w); j++)
            {
                input_dxdy0[f][i][j] = (sigmoid(input_dxdy0[f][i][j]) + xy_grid0[f][i][j]) * stride;
                input_dxdy1[f][i][j] = (sigmoid(input_dxdy1[f][i][j]) + xy_grid1[f][i][j]) * stride;
                input_dxdy2[f][i][j] = (sigmoid(input_dxdy2[f][i][j]) + xy_grid2[f][i][j]) * stride;
                input_dwdh0[f][i][j] = fm_t(exp(input_dwdh0[f][i][j])) * *(anchor + f) * stride;
                input_dwdh1[f][i][j] = fm_t(exp(input_dwdh1[f][i][j])) * *(anchor + 2 + f) * stride;
                input_dwdh2[f][i][j] = fm_t(exp(input_dwdh2[f][i][j])) * *(anchor + 4 + f) * stride;
```

HLS Code Approach

- Had to modify network to implement on FPGA
- Did not implement Batch Normalization on FPGA
- Faced implementation problems for sigmoid and exponential activation – did not implement the YOLO layer on FPGA
- Used Leaky ReLU for convolution

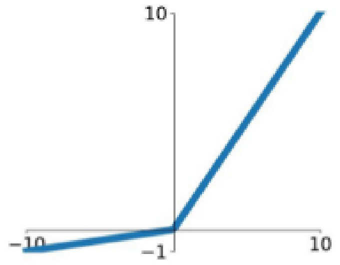
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



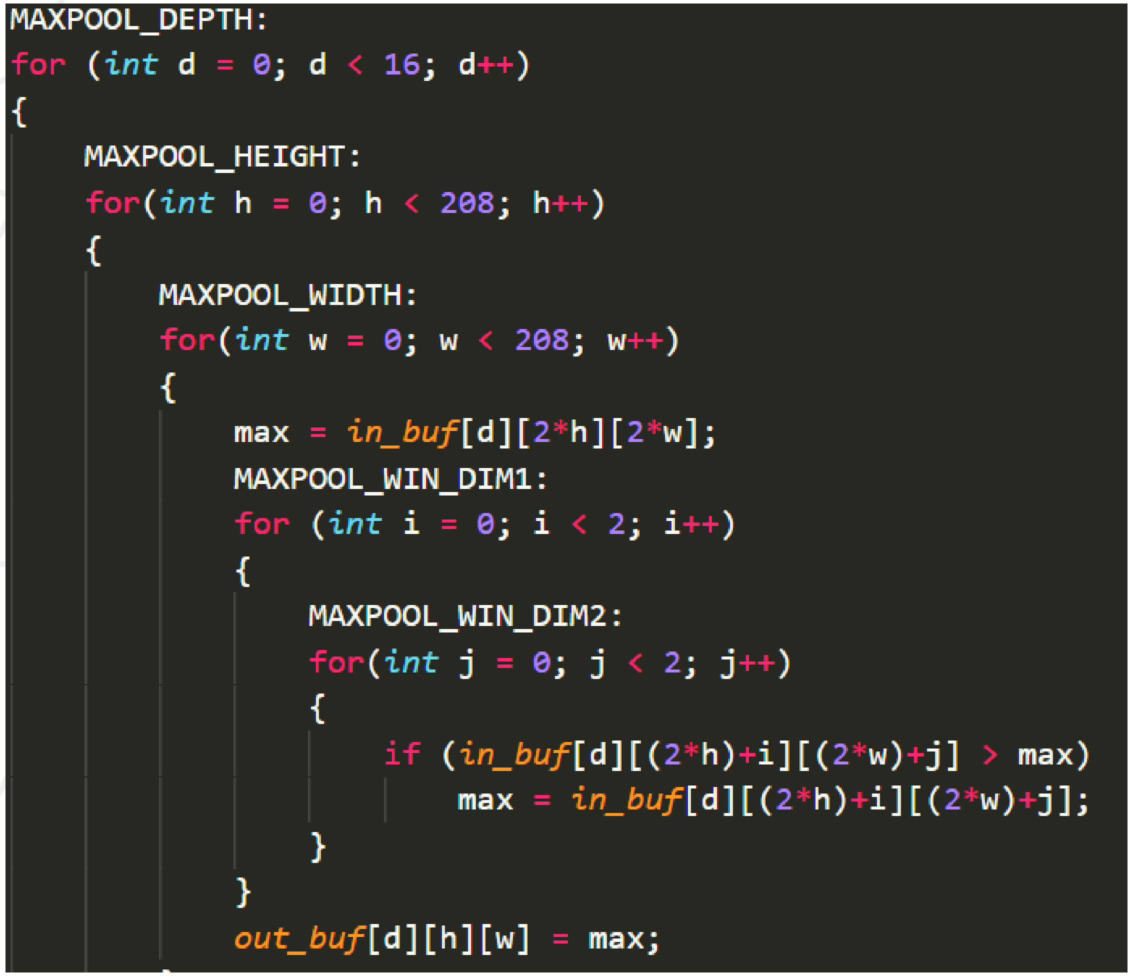
Leaky ReLU

$\max(0.1x, x)$

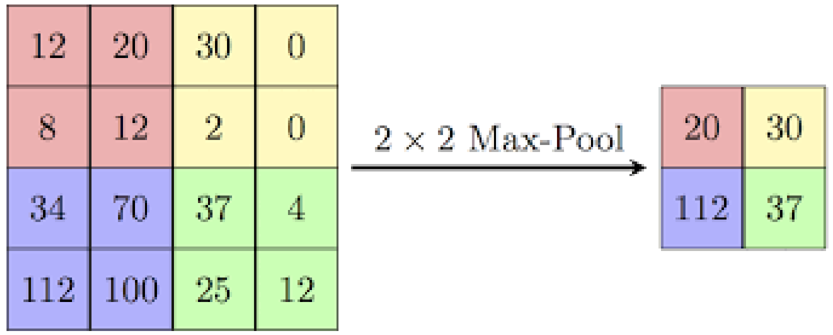


Code Snippets

- Max Pooling

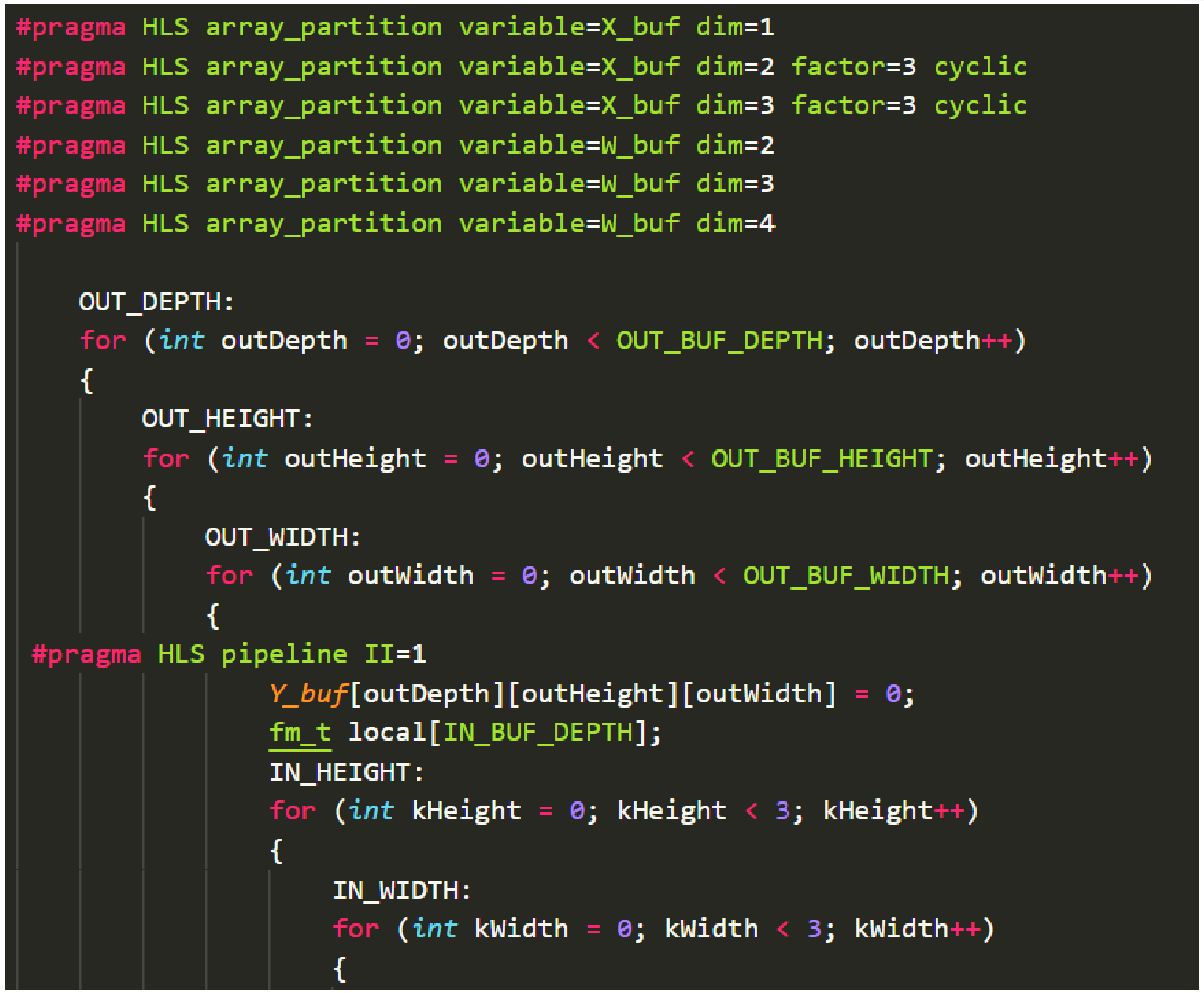


```
MAXPOOL_DEPTH:
for (int d = 0; d < 16; d++)
{
    MAXPOOL_HEIGHT:
    for(int h = 0; h < 208; h++)
    {
        MAXPOOL_WIDTH:
        for(int w = 0; w < 208; w++)
        {
            max = in_buf[d][2*h][2*w];
            MAXPOOL_WIN_DIM1:
            for (int i = 0; i < 2; i++)
            {
                MAXPOOL_WIN_DIM2:
                for(int j = 0; j < 2; j++)
                {
                    if (in_buf[d][(2*h)+i][(2*w)+j] > max)
                        max = in_buf[d][(2*h)+i][(2*w)+j];
                }
            }
            out_buf[d][h][w] = max;
```



Optimization Strategies

- Array Partitioning
- Loop Pipelining
- Loop Unroll

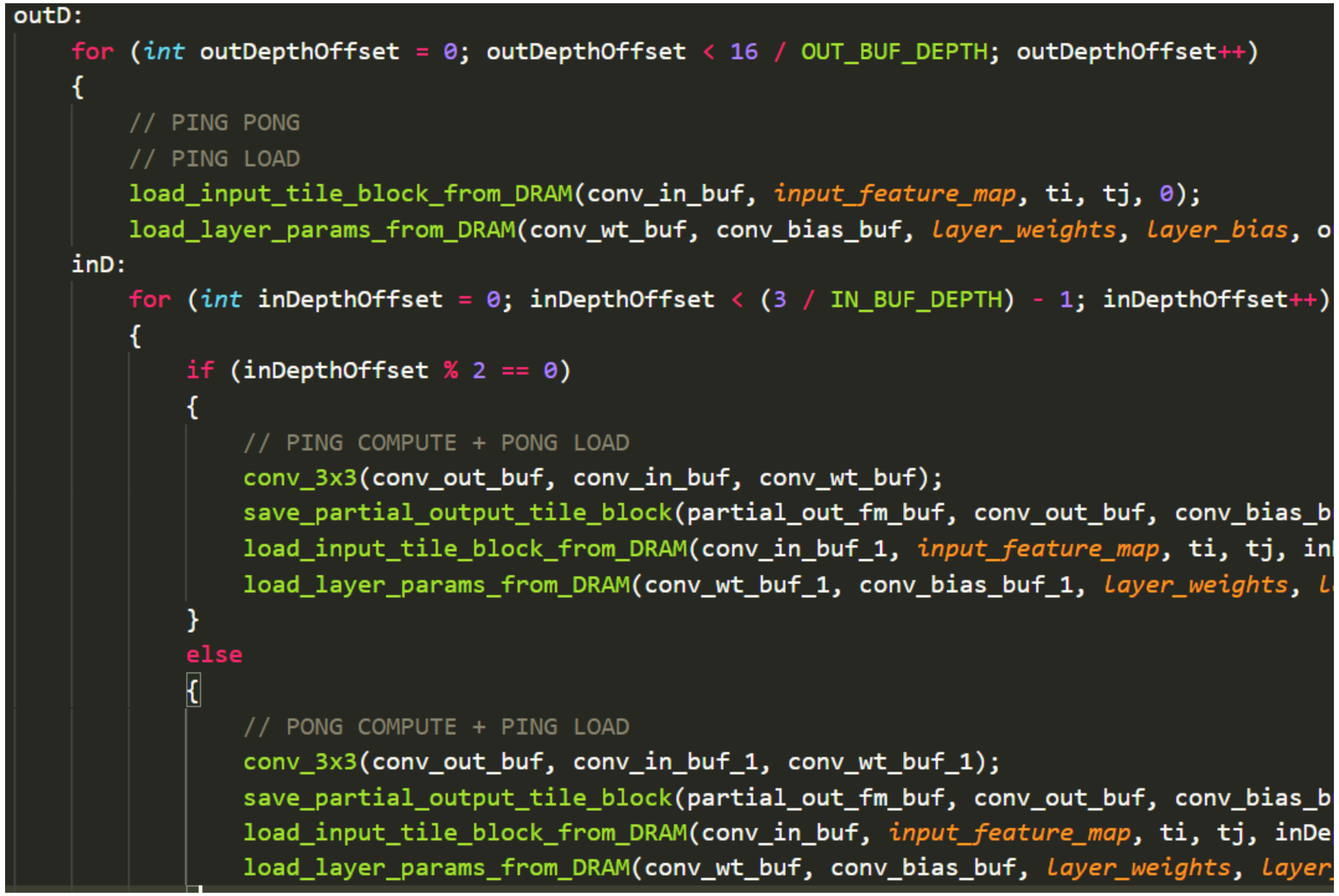


```
#pragma HLS array_partition variable=X_buf dim=1
#pragma HLS array_partition variable=X_buf dim=2 factor=3 cyclic
#pragma HLS array_partition variable=X_buf dim=3 factor=3 cyclic
#pragma HLS array_partition variable=W_buf dim=2
#pragma HLS array_partition variable=W_buf dim=3
#pragma HLS array_partition variable=W_buf dim=4

    OUT_DEPTH:
    for (int outDepth = 0; outDepth < OUT_BUF_DEPTH; outDepth++)
    {
        OUT_HEIGHT:
        for (int outHeight = 0; outHeight < OUT_BUF_HEIGHT; outHeight++)
        {
            OUT_WIDTH:
            for (int outWidth = 0; outWidth < OUT_BUF_WIDTH; outWidth++)
            {
 #pragma HLS pipeline II=1
                Y_buf[outDepth][outHeight][outWidth] = 0;
                fm_t local[IN_BUF_DEPTH];
                IN_HEIGHT:
                for (int kHeight = 0; kHeight < 3; kHeight++)
                {
                    IN_WIDTH:
                    for (int kWidth = 0; kWidth < 3; kWidth++)
                    {
```

Optimization Strategies

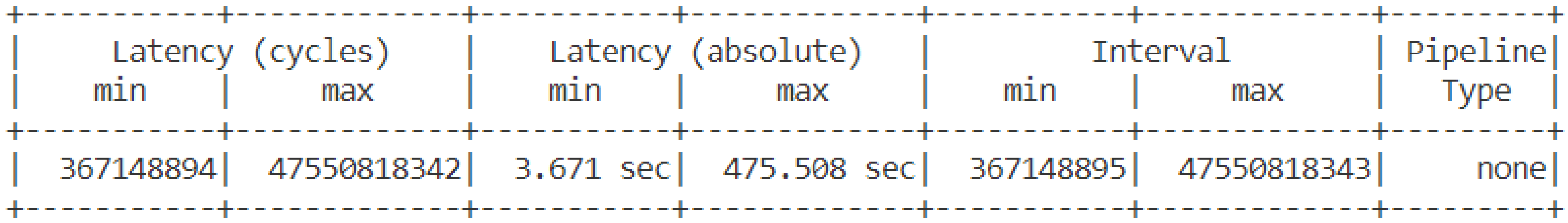
- Loop Tiling
- Ping Pong Buffer



```
outD:
    for (int outDepthOffset = 0; outDepthOffset < 16 / OUT_BUF_DEPTH; outDepthOffset++)
    {
        // PING PONG
        // PING LOAD
        load_input_tile_block_from_DRAM(conv_in_buf, input_feature_map, ti, tj, 0);
        load_layer_params_from_DRAM(conv_wt_buf, conv_bias_buf, layer_weights, layer_bias, o
    inD:
        for (int inDepthOffset = 0; inDepthOffset < (3 / IN_BUF_DEPTH) - 1; inDepthOffset++)
        {
            if (inDepthOffset % 2 == 0)
            {
                // PING COMPUTE + PONG LOAD
                conv_3x3(conv_out_buf, conv_in_buf, conv_wt_buf);
                save_partial_output_tile_block(partial_out_fm_buf, conv_out_buf, conv_bias_b
                load_input_tile_block_from_DRAM(conv_in_buf_1, input_feature_map, ti, tj, in
                load_layer_params_from_DRAM(conv_wt_buf_1, conv_bias_buf_1, layer_weights, l
            }
            else
            {
                // PONG COMPUTE + PING LOAD
                conv_3x3(conv_out_buf, conv_in_buf_1, conv_wt_buf_1);
                save_partial_output_tile_block(partial_out_fm_buf, conv_out_buf, conv_bias_b
                load_input_tile_block_from_DRAM(conv_in_buf, input_feature_map, ti, tj, inDe
                load_layer_params_from_DRAM(conv_wt_buf, conv_bias_buf, layer_weights, layer
```

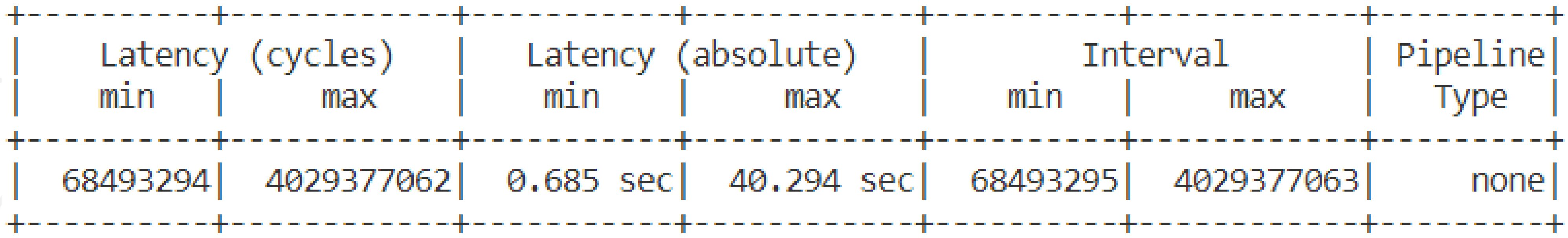
Evaluation

- Unoptimized, Synthesizable Run for all layers:



```
+-----------+-------------+-----------+-------------+-----------+-------------+---------+
|    Latency (cycles)     |   Latency (absolute)    |        Interval         | Pipeline|
|    min    |     max     |    min    |     max     |    min    |     max     |   Type  |
+-----------+-------------+-----------+-------------+-----------+-------------+---------+
|  367148894|  47550818342|  3.671 sec|  475.508 sec|  367148895|  47550818343|     none|
+-----------+-------------+-----------+-------------+-----------+-------------+---------+
```

- Accelerated Run:



```
+----------+------------+----------+------------+----------+------------+---------+
|    Latency (cycles)   |  Latency (absolute)   |       Interval        | Pipeline|
|    min   |     max    |    min   |     max    |    min   |     max    |   Type  |
+----------+------------+----------+------------+----------+------------+---------+
|  68493294|  4029377062| 0.685 sec|  40.294 sec|  68493295|  4029377063|     none|
+----------+------------+----------+------------+----------+------------+---------+
```

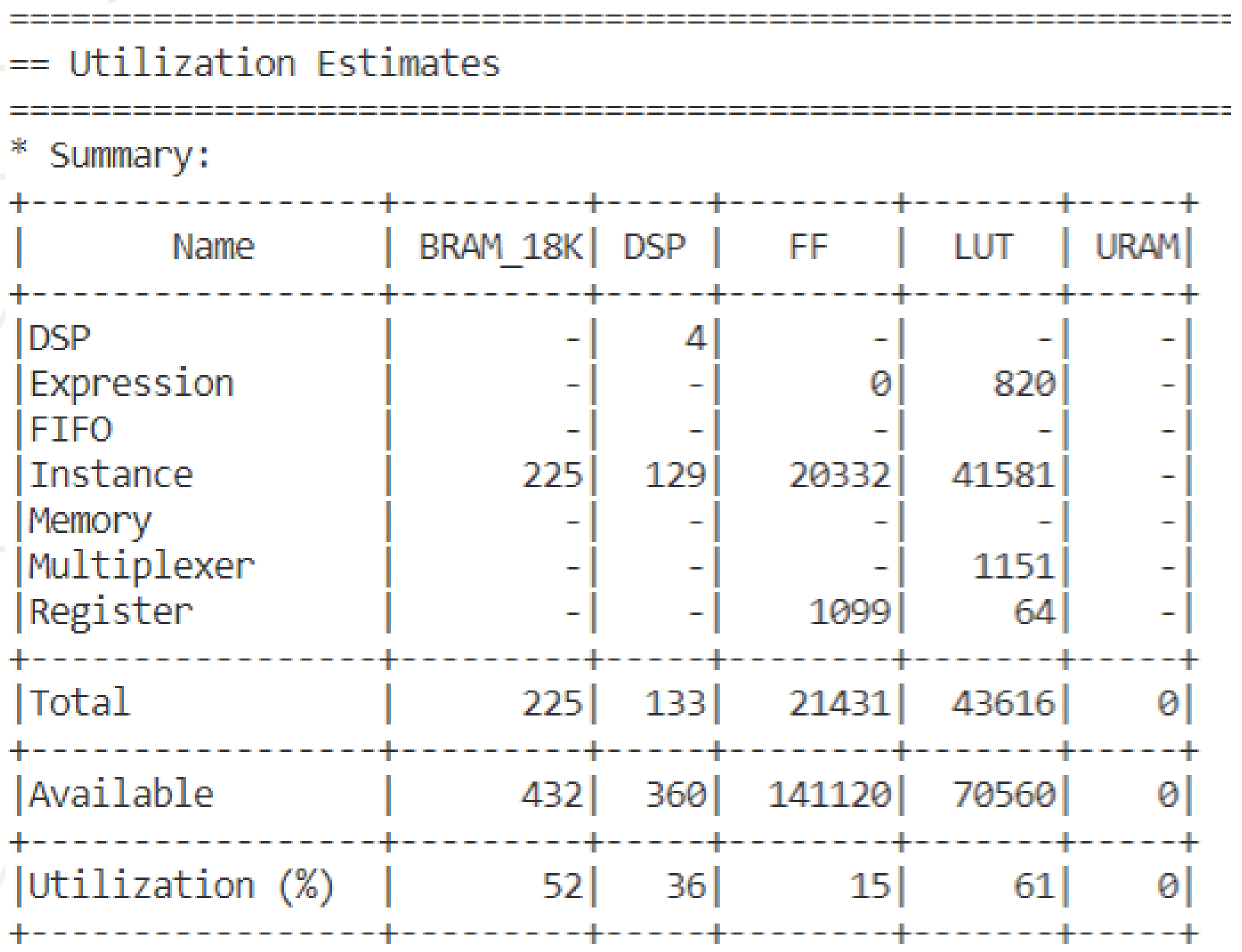
Evaluations – Single layer data

	Latency (sec)	SpeedUp
Unoptimized	3.2	1
Ping Pong Buffer	2.02	1.584158
Loop Unroll + Pipelining	0.186	17.2043
Array Partitioning	0.0549	58.2878

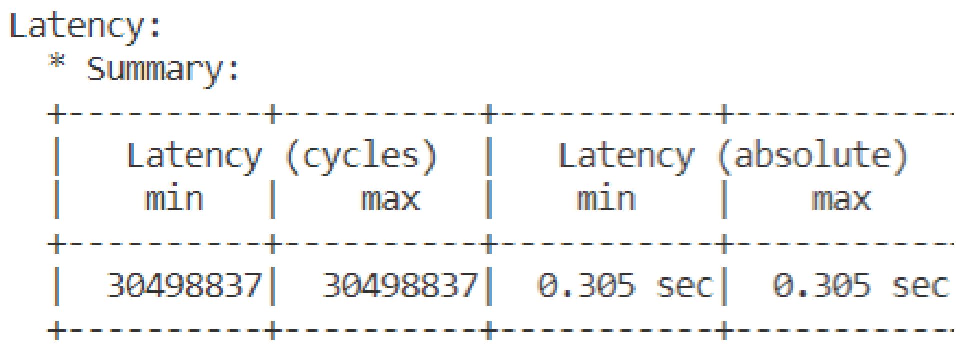
Georgia Tech

Evaluation

- Accelerated run for four layers:



```
================================================================
== Utilization Estimates
================================================================
* Summary:
+-----------------+---------+-----+--------+-------+-----+
|       Name      | BRAM_18K| DSP |   FF   |  LUT  | URAM|
+-----------------+---------+-----+--------+-------+-----+
|DSP              |        -|    4|       -|      -|    -|
|Expression       |        -|    -|       0|    820|    -|
|FIFO             |        -|    -|       -|      -|    -|
|Instance         |      225|  129|   20332|  41581|    -|
|Memory           |        -|    -|       -|      -|    -|
|Multiplexer      |        -|    -|       -|   1151|    -|
|Register         |        -|    -|    1099|     64|    -|
+-----------------+---------+-----+--------+-------+-----+
|Total            |      225|  133|   21431|  43616|    0|
+-----------------+---------+-----+--------+-------+-----+
|Available        |      432|  360|  141120|  70560|    0|
+-----------------+---------+-----+--------+-------+-----+
|Utilization (%)  |       52|   36|      15|     61|    0|
+-----------------+---------+-----+--------+-------+-----+
```



```
Latency:
  * Summary:
  +---------+---------+----------+----------
  |  Latency (cycles) |  Latency (absolute)
  |   min   |   max   |    min   |    max
  +---------+---------+----------+----------
  | 30498837| 30498837| 0.305 sec| 0.305 sec
  +---------+---------+----------+----------
```

Future Actions to try before submitting code

- Our initial approach caused issues with debugging.
- Use independent functions for all convolution layers.
- Literature search for references with YOLO layer implementation on FPGA