OPTIMIZING FPGA-BASED ACCELERATOR DESIGN FOR DEEP CONVOLUTIONAL NEURAL NETWORKS

Surabhi Singh(ss3659)
Mohammad Dohadwala(md874)
Godly k Thomas(gkt27)
YOLO: REAL TIME OBJECT DETECTION SYSTEM

- YOLO.?? YOU ONLY LIVE ONCE
- YOU ONLY LOOK ONCE – State of real-time object detection system
- Can Process upto 200 frames per second
HOW IT WORKS

• YOLO does the following:
  • Applies a single neural network to the resized full image (448X448)
  • Divide image into regions and predicts probability for each region
  • Bounding box for each object is weighted by predicted probabilities

NETWORK DESIGN

- 24 convolutional Layers followed by two fully connected layers
- Each convolutional layer has input and output feature maps

Image Courtesy: See Reference[1]
WHERE IS HLDDA.??

- Optimize a single convolutional layer execution in hardware (FPGA) while execute other convolutional layers in software (ARM)
- Use pragmas learnt in class to increase throughout
- Loop Unrolling, Loop Pipelining, Hardware Tiling for the function executed in hardware.
SINGLE CONVOLUTIONAL LAYER

Weights

Input Feature Map

Output Feature Map
KERNEL COMPUTATION

• Pseudo Code

```c
void gemm_mm(int M, int N, int K, float ALPHA,
              float *A, int lda,
              float *B, int ldb,
              float *C, int ldc)
{
    int i,j,k;
    for(i = 0; i < M; ++i){
        for(k = 0; k < K; ++k){
            register float A_PART = ALPHA*A[i*lda+k];
            for(j = 0; j < N; ++j){
                C[i*ldc+j] += A_PART*B[k*ldb+j];
            }
        }
    }
}
```

KERNEL (WEIGHT) MULTIPLICATION

*Image Courtesy: See Reference[3]*
### IMPLEMENTATION: FEATURE MAP CALCULATION

<table>
<thead>
<tr>
<th>Layer</th>
<th>Convolutional</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>input_fm (N)</td>
<td>3</td>
</tr>
<tr>
<td>output_fm (M)</td>
<td>16</td>
</tr>
<tr>
<td>fm row (R)</td>
<td>448</td>
</tr>
<tr>
<td>fm col (C)</td>
<td>448</td>
</tr>
<tr>
<td>kernel (K)</td>
<td>3</td>
</tr>
<tr>
<td>stride (S)</td>
<td>1</td>
</tr>
<tr>
<td>set #</td>
<td>1</td>
</tr>
<tr>
<td>Weights (MBytes)</td>
<td>0.0033</td>
</tr>
<tr>
<td>Input Feature Map (MBytes)</td>
<td>41.3438</td>
</tr>
<tr>
<td>Output Feature Map (MBytes)</td>
<td>24.5</td>
</tr>
</tbody>
</table>
IMPLEMENTATION: OPTIMIZATION

HARDWARE OPTIMIZATION -> Achieve II = 1 -> Any ideas...??

COMMUNICATION OVERHEAD -> Reduce Latency
CODE OPTIMIZATION

• Code Snippet

```c
void gemm_nn(int M, int N, int K, float ALPHA,
    float *A, int lda,
    float *B, int ldb,
    float *C, int ldc)
{
    int i, j, k;
    for (i = 0; i < M; ++i){
        for (k = 0; k < K; ++k){
            register float A_PART = ALPHA*A[1*lda+k];
            for (j = 0; j < N; ++j){
                C[i*lda+j] = A_PART*B[k*ldb+j];
            }
        }
    }
}
```

Image Courtesy: See Reference[3]
HARDWARE OPTIMIZATION STEPS TO ACHIEVE II = 1

* Stream data into local RAM for inputs (multiple access required)
* Pipeline the dot-product loop, to fully unroll it
* Separate multiply-accumulate in inner loop to force two Floating Point operators

**ACTUAL II = 14**
**EXPECTED II = 1**

RESOURCE CONSTRAINT-LIMITED MEMORY PORTS

ARRAY PARTITION

* Partition local RAMs into N/2 sub-arrays for fully parallel access (dual-port read)
## RESULTS & CONCLUSION

<table>
<thead>
<tr>
<th>Design</th>
<th>CPU Cycles</th>
<th>Speedup</th>
<th>Total Runtime (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average function Runtime</td>
<td>Total 1st layer Runtime</td>
<td></td>
</tr>
<tr>
<td>SW HW BASELINE (ARM+FPGA)</td>
<td>22,135,862</td>
<td>7,415,513,736</td>
<td>1X</td>
</tr>
<tr>
<td>SW BASELINE (ARM)</td>
<td>2,763,683</td>
<td>925,833,786</td>
<td>8X</td>
</tr>
<tr>
<td>SW HW ALTERNATE (ARM+FPGA)</td>
<td>323,961</td>
<td>108,526,936</td>
<td>68X</td>
</tr>
</tbody>
</table>

**Average function runtime**

**Total 1st layer runtime**

**Speedup**

**Total Runtime (s)**
FINAL PROJECT LINK

- https://youtu.be/v9Btl9h6OUM
ACKNOWLEDGEMENT

- Special Thanks to Professor Zhiru Zhang for helping us throughout the Project duration and guiding us when we faced roadblocks.

- We would also like to acknowledge suggestions from Steve Dai and Hsiang Yi, PhD students under Professor Zhiru Zhang.
REFERENCES

- Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks
  *FPGA ’15 Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*
  Chen Zhang, Peng Li, Guangyu Sun, Yijin Guan, Bingjun Xiao, Jason Cong,

- YOLO REAL TIME DETECTION SYSTEM

- YOLO GITHUB LINK
  https://github.com/pjreddie/darknet