Peking University

Center for Energy-efficient Computing and Applications

Exploring Heterogeneous Algorithms for Accelerating Deep Convolutional Neural Networks on FPGAs

Qingcheng Xiao[1], Yun Liang[1], Liqiang Lu[1], Shengen Yan[2,3] and Yu-Wing Tai[3]

[1]Center for Energy-efficient Computing and Applications, EECS, Peking University

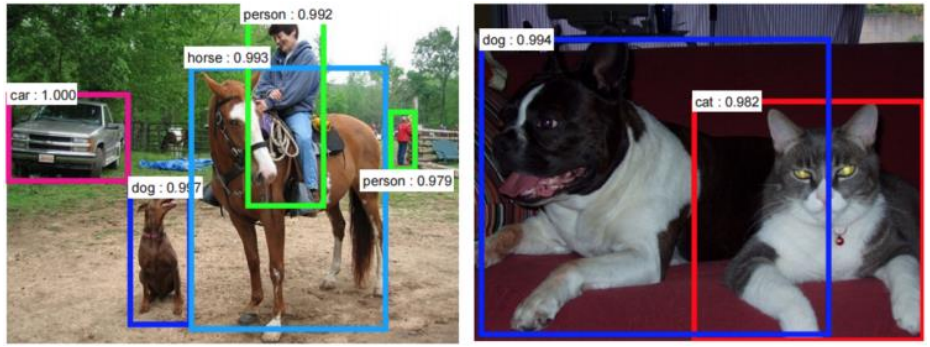
[2]Department of Information Engineering, Chinese University of Hong Kong

[3]SenseTime Group Limited

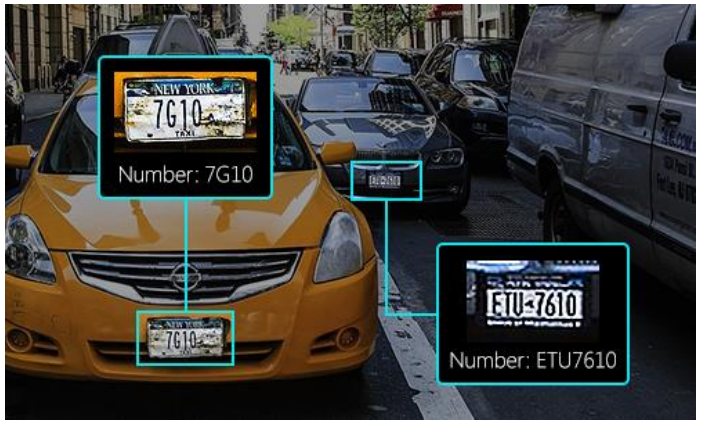
{walkershaw,ericlyun,luliqiang}@pku.edu.cn, {yanshengen,yuwing}@gmail.com

Background

Segmentation

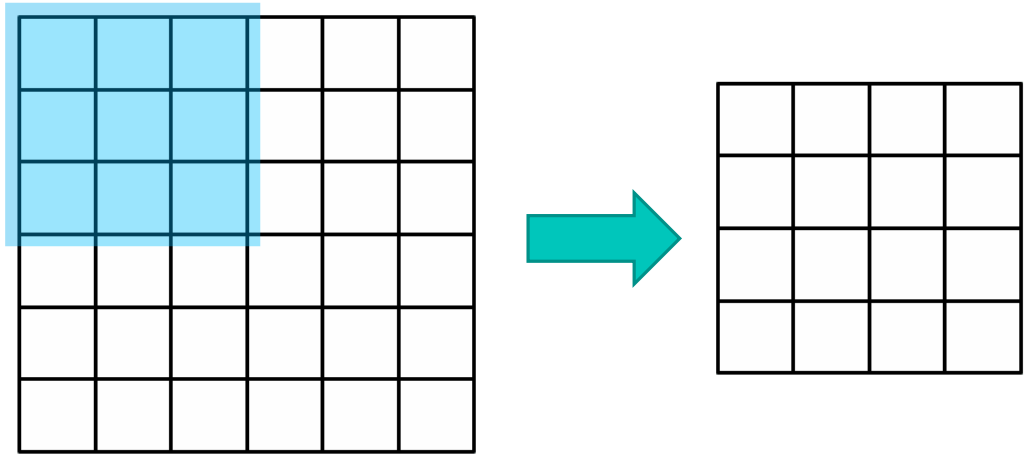


Detection



ADAS

Conventional Convolution



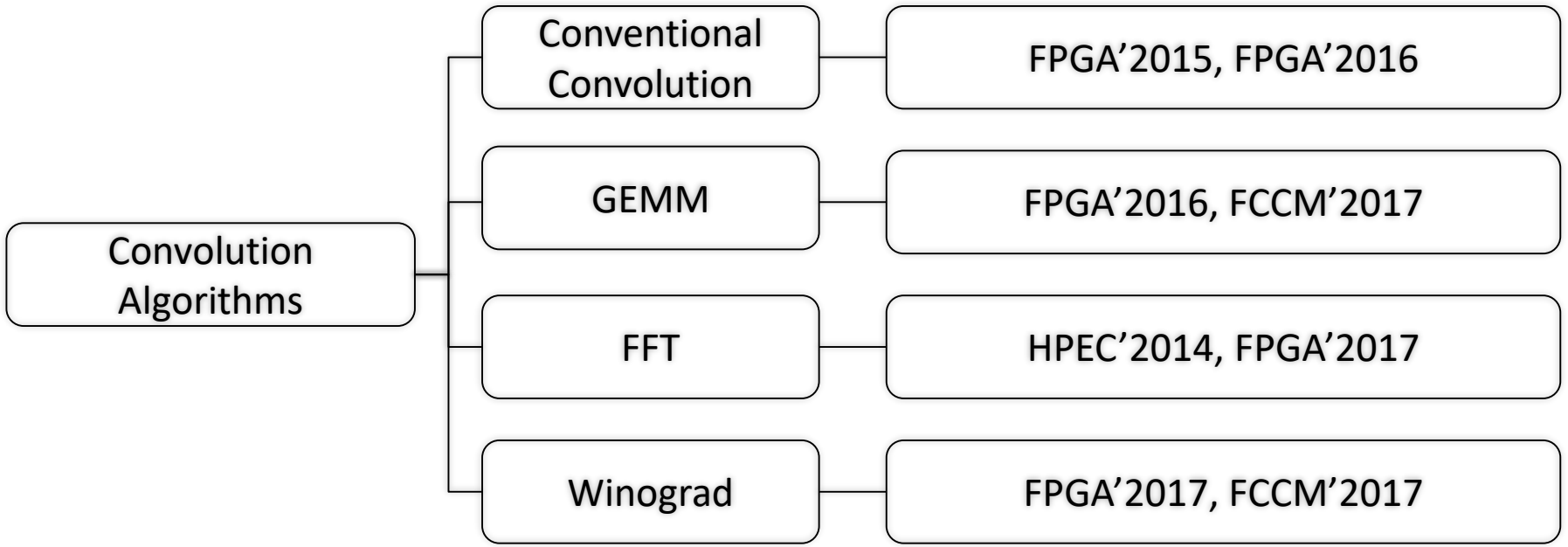
Input feature map

Output feature map

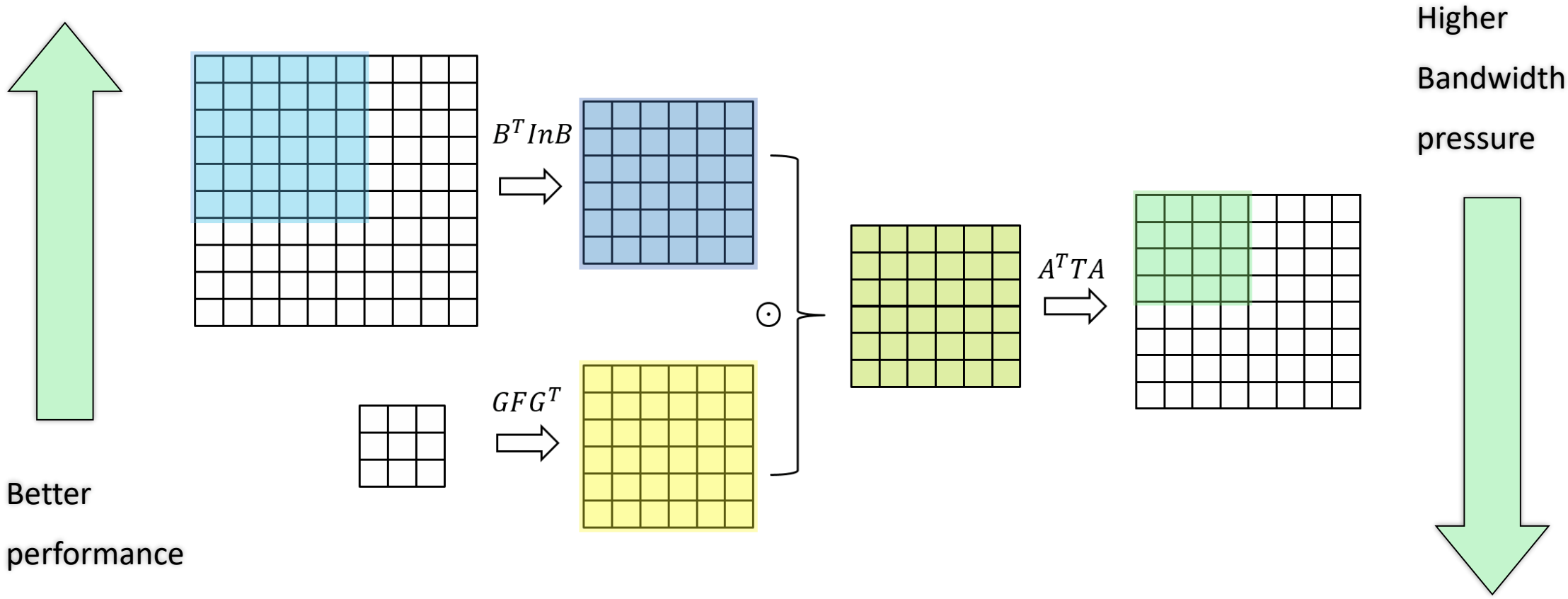
- Techniques
  - Tiling
  - Parallelism in different dimensions
  - Memory access pattern optimization

$$Y[i][j][n] = \sum_{m=0}^{M-1} \sum_{u=0}^{K-1} \sum_{v=0}^{K-1} D[i * S + u][j * S + v][m] \times G[n][u][v][m]$$

Emerging convolution algorithms

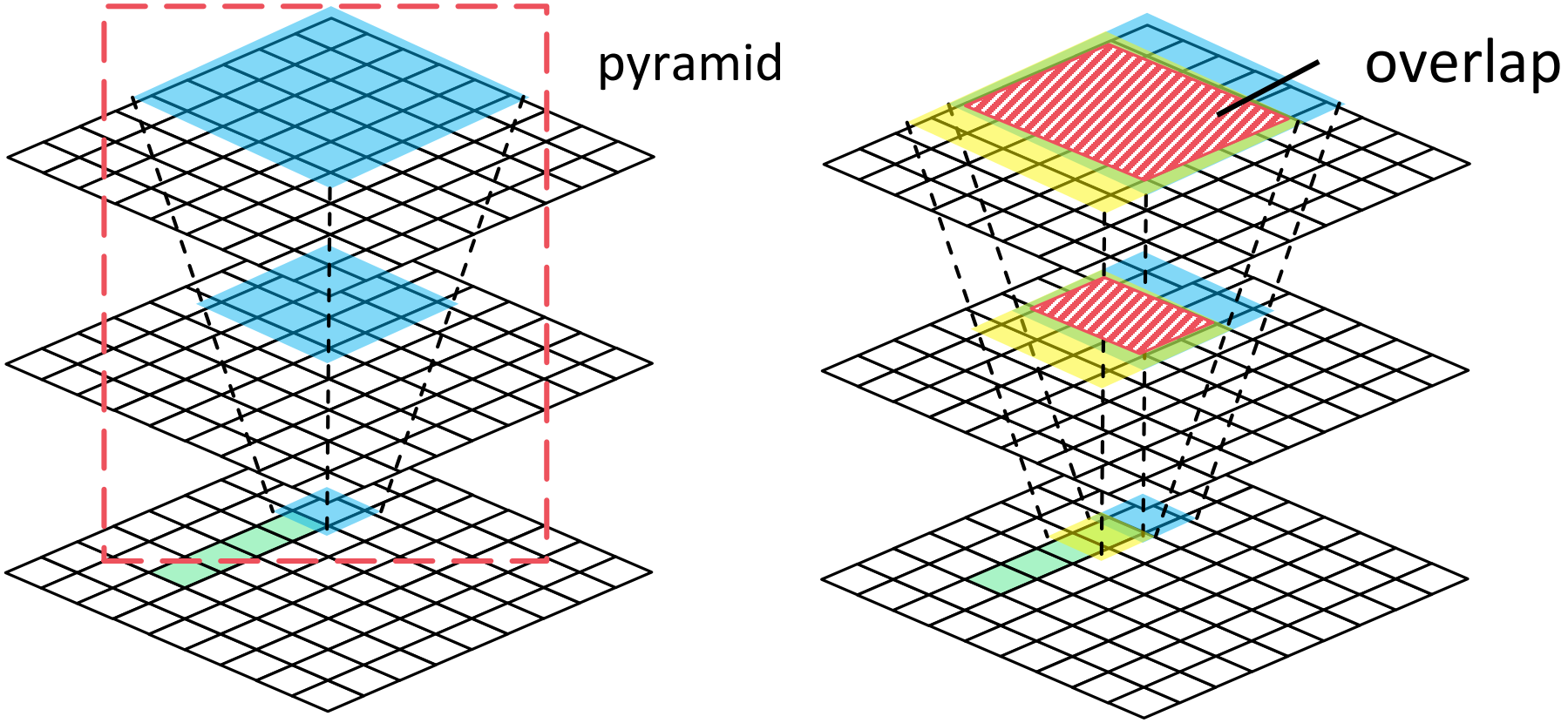


Winograd Convolution



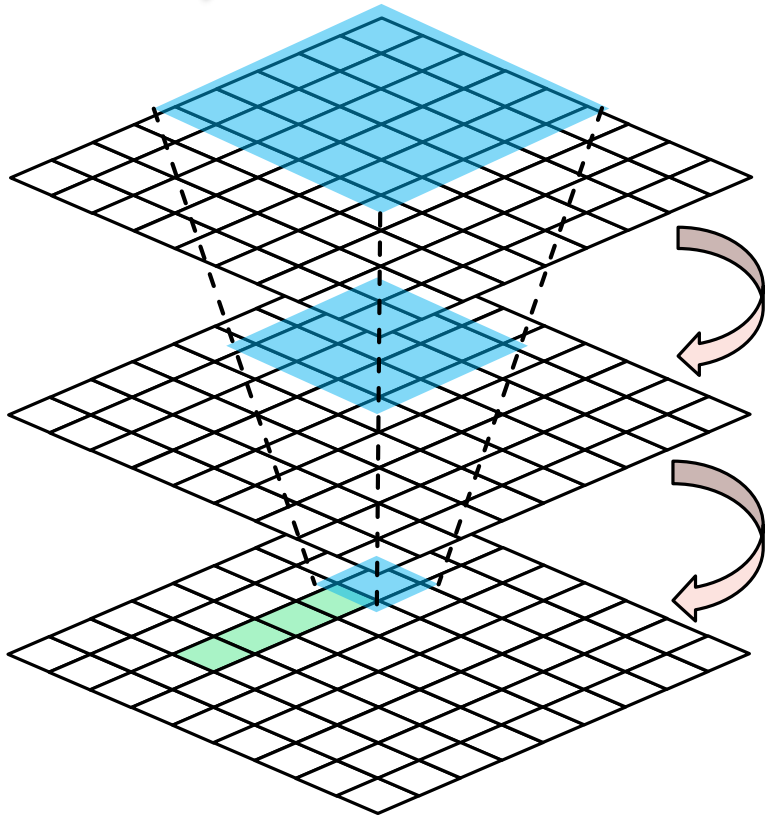
Architecture Design

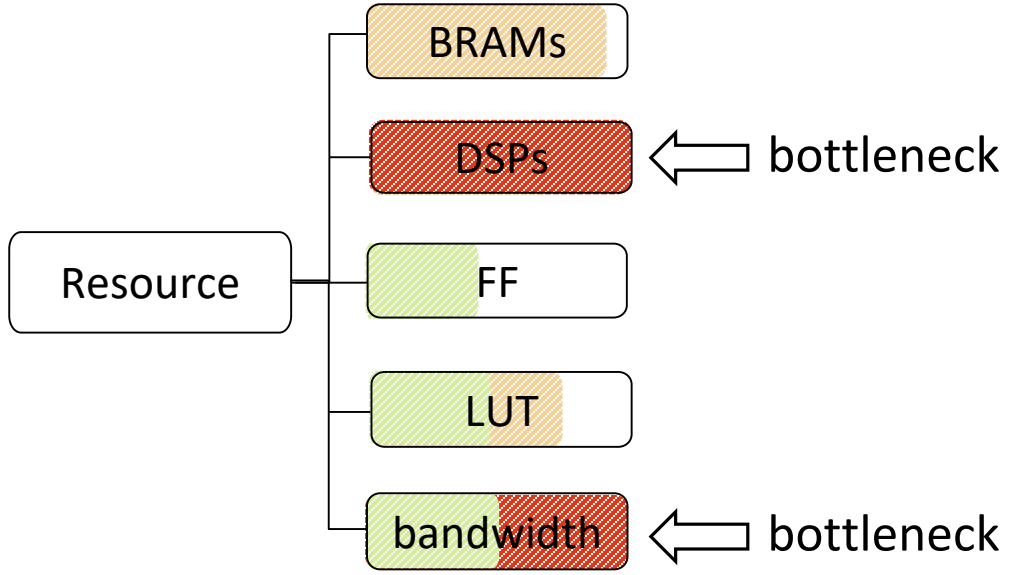
- Fusion Architecture



Why heterogeneous algorithms?

- parallelize the computation as much as possible, until either the computation resources or memory bandwidth are exhausted

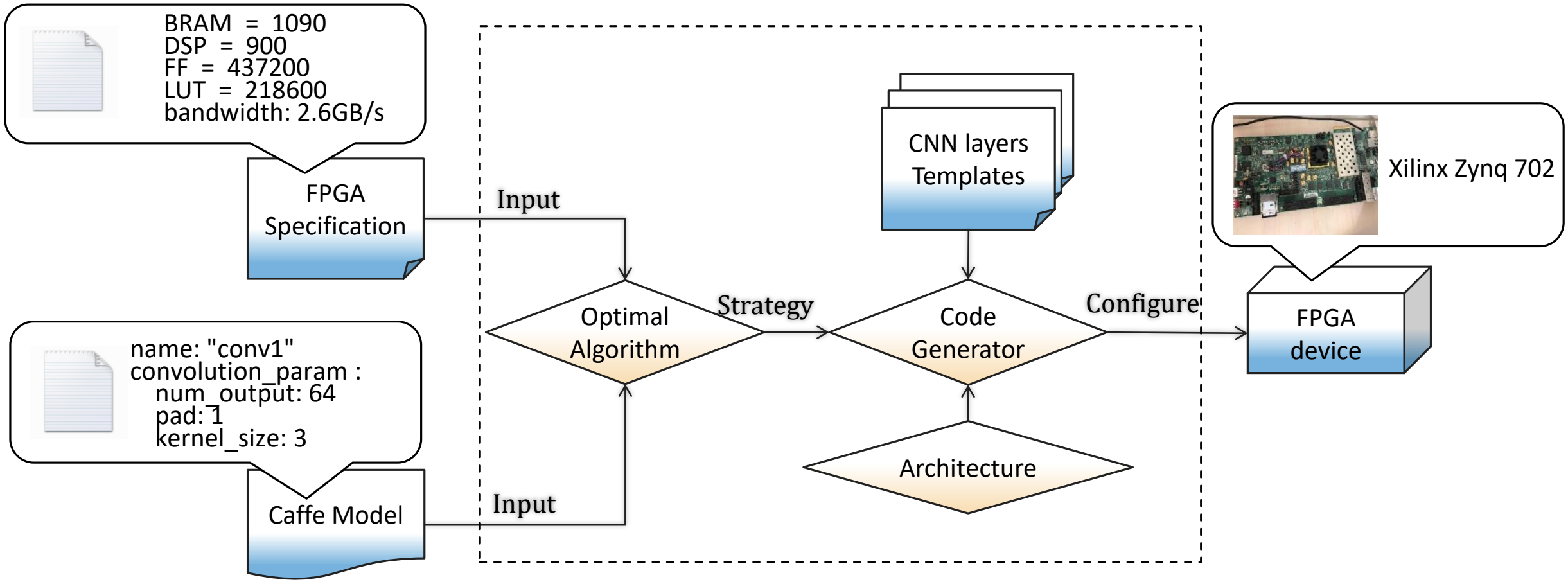




Less wasted resource, higher performance

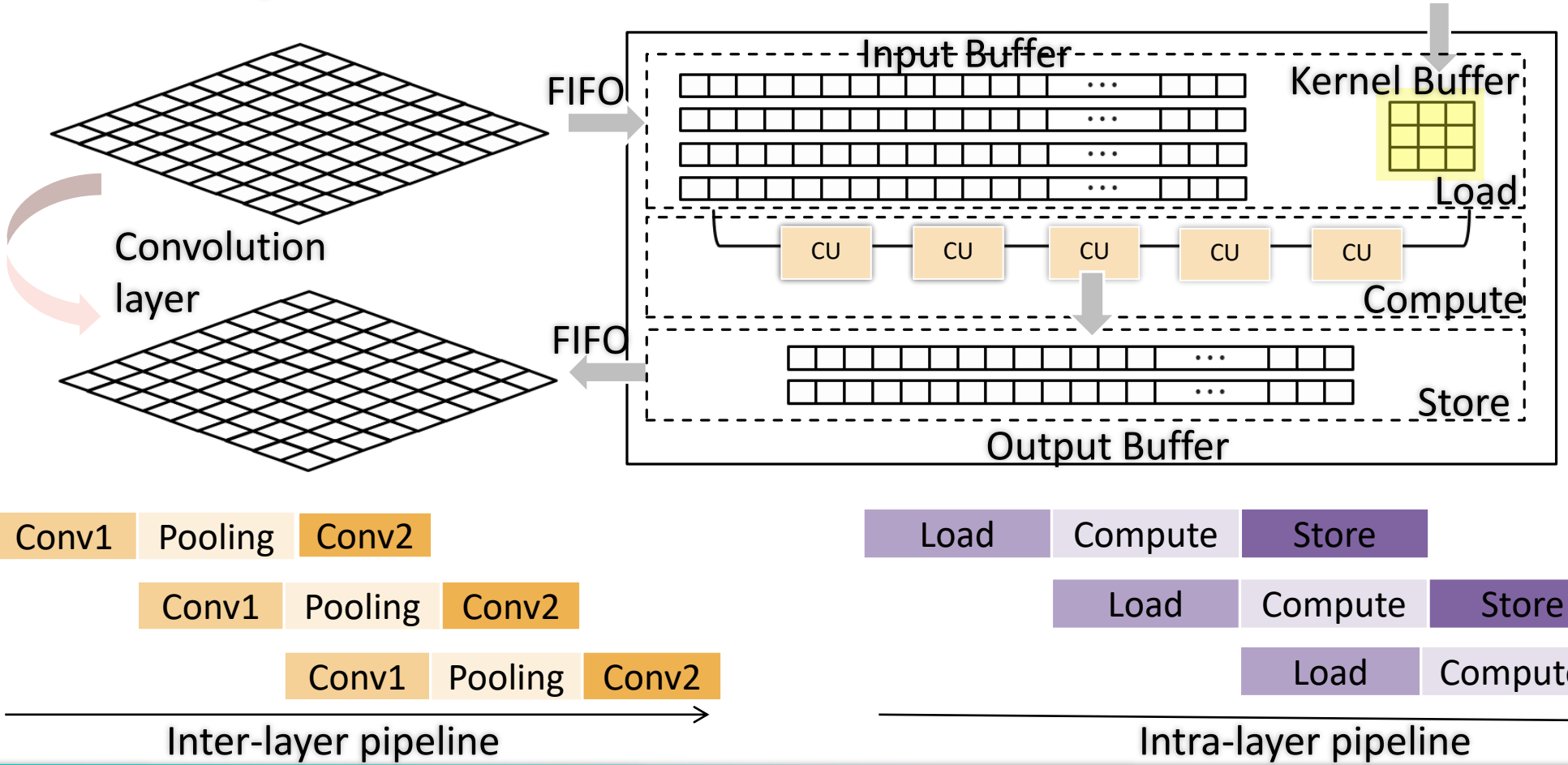
Only exhaust one dimension of resource, leaving others under-utilized

Framework Overview



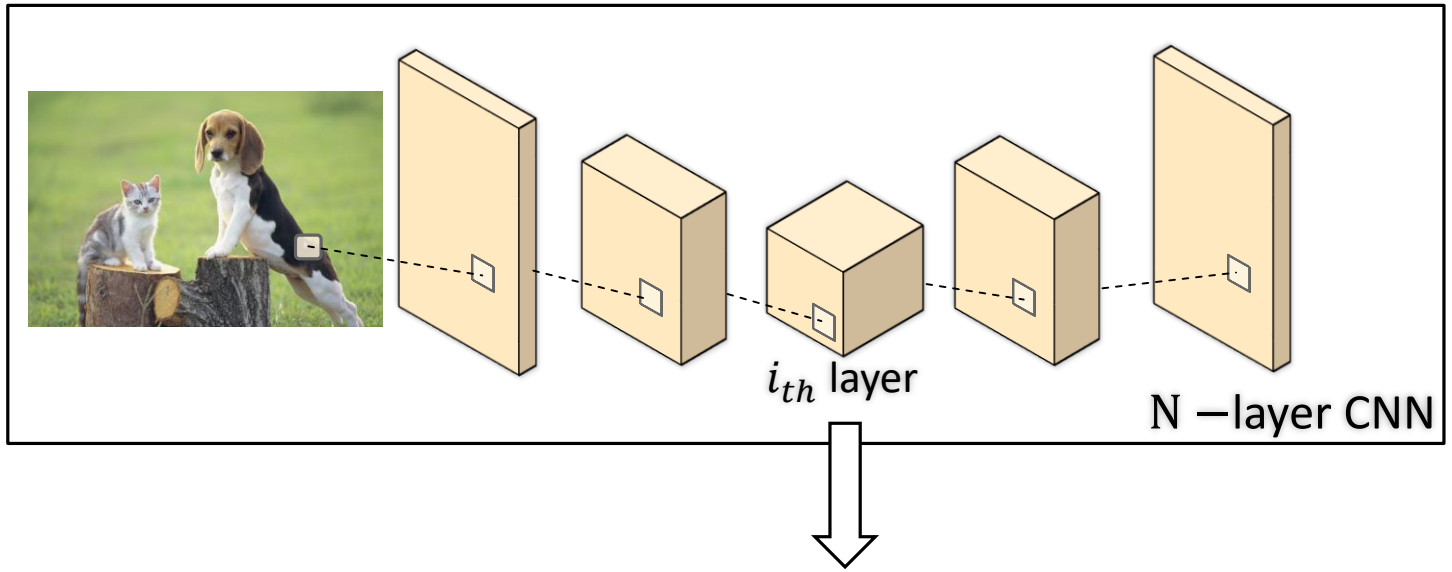
Architecture Design

- Line buffer design



Optimal Algorithm

- Problem definition



$$C_i = < g_i, algo_i, p_i >$$

A strategy $S = \{ C_i \mid 1 \le i \le N \}$

Input:

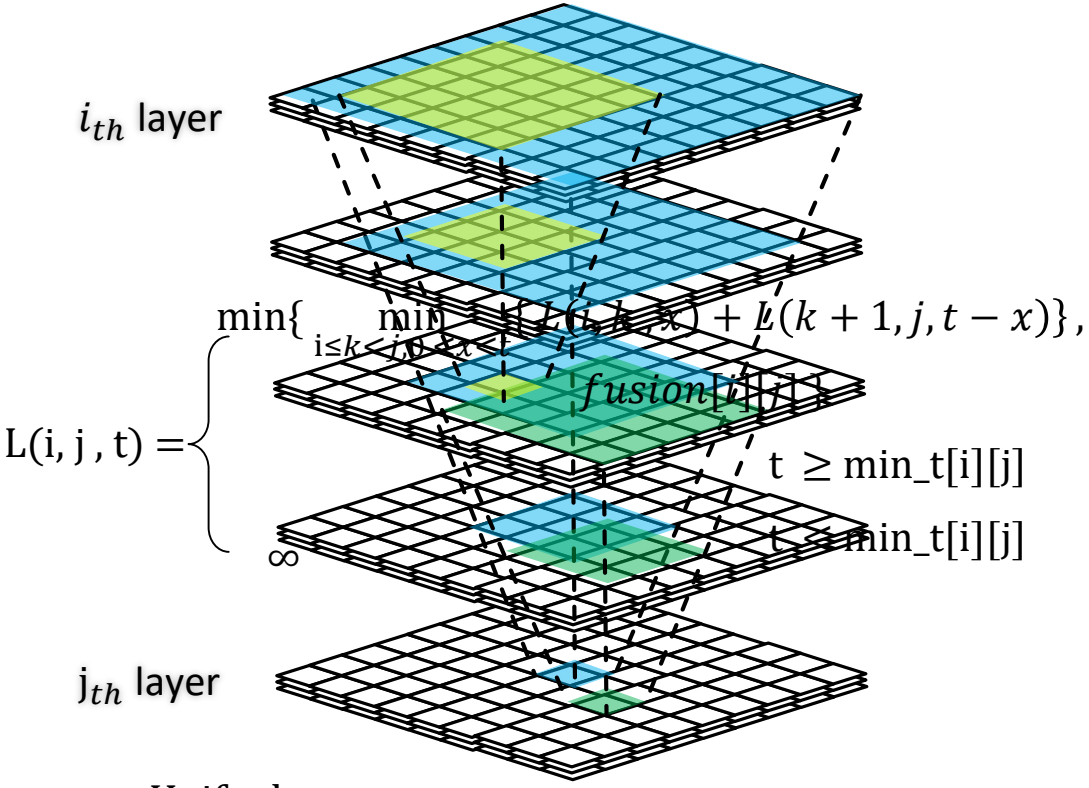
- an N-layer CNN
- Resource constraint *R*
- Transfer constraint *T*

Output:

The optimal strategy *S* that minimizes end-to-end latency

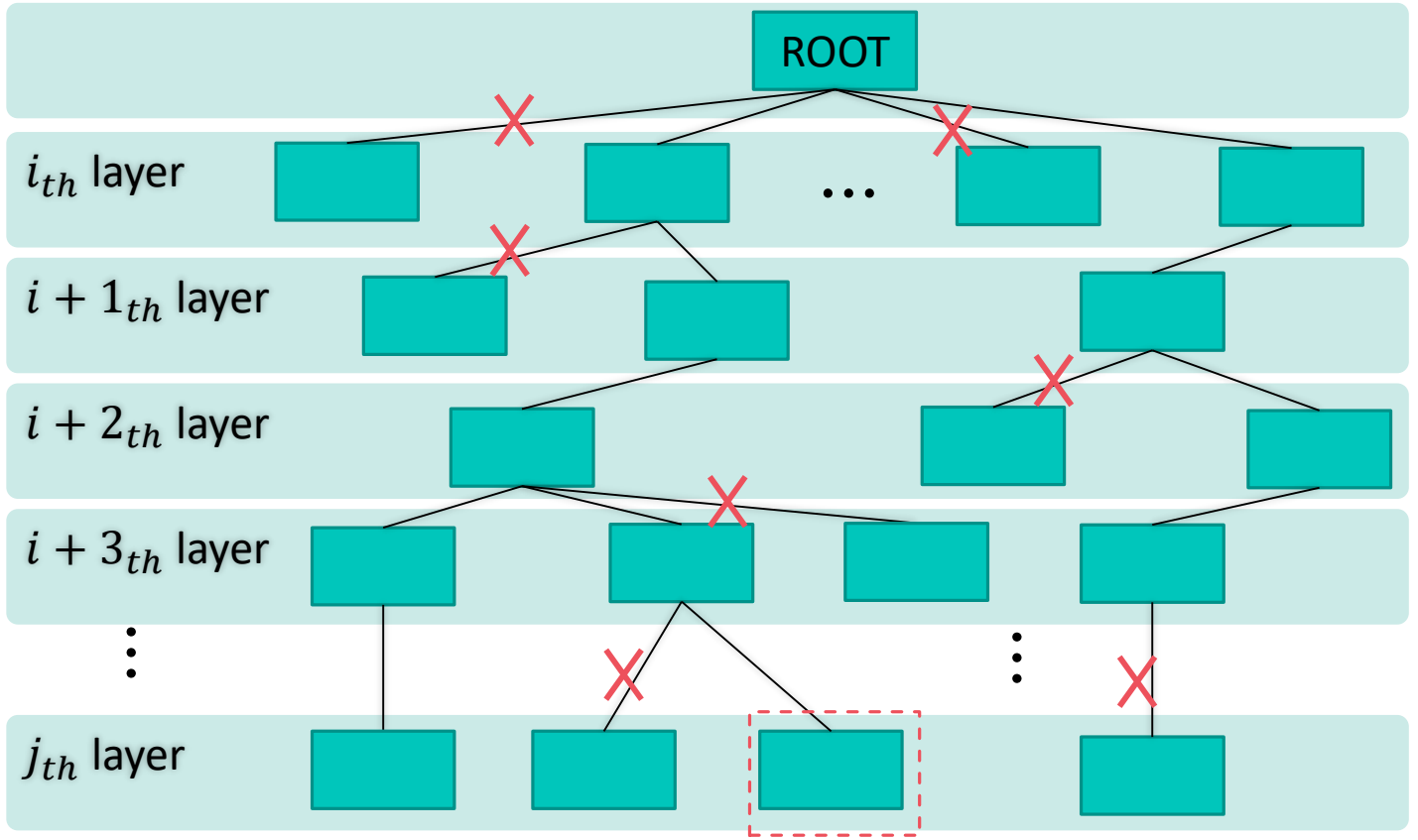
Optimal Algorithm

- Dynamic programming

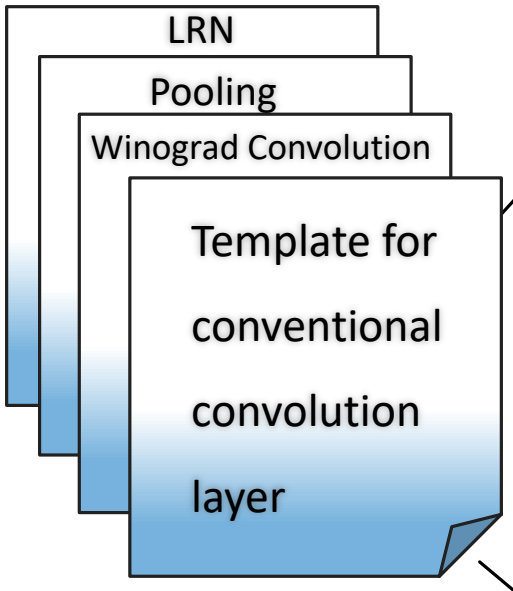


- Unify them as a group
- Find a middle layer to split them into two groups

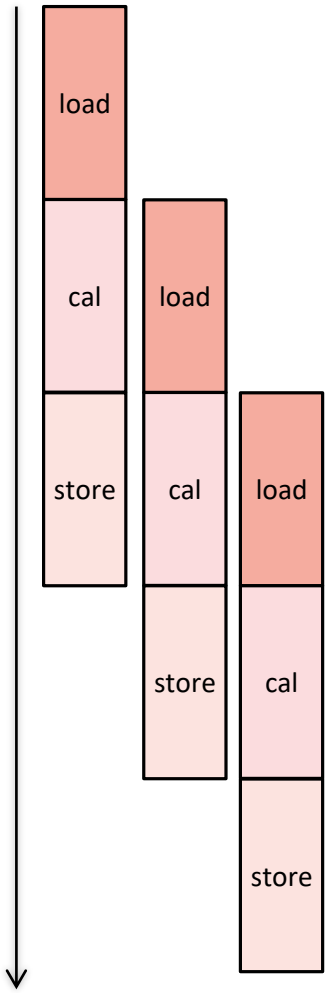
- Branch-and-bound



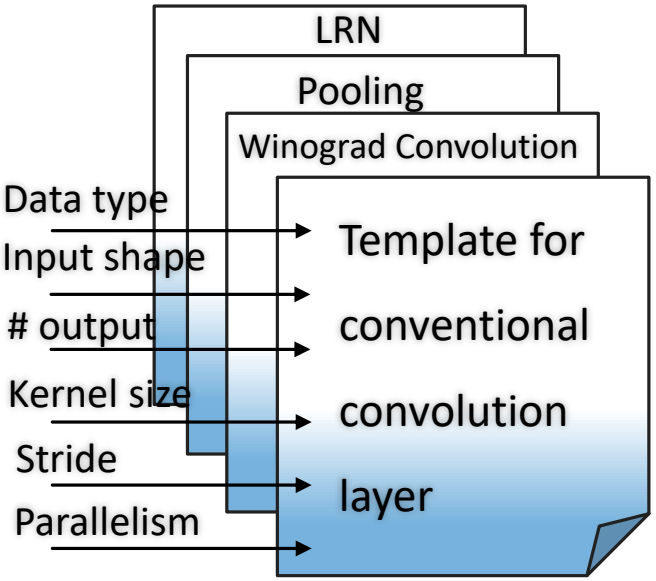
Code Generator




```
#pragma HLS ARRAY_PARTITION
#pragma HLS PIPELINE
Conventional_conv<…>(input, output, weight){
        init(input, line_buf);
        for(row < output_h){
                load(input, line_buf);
                compute(line_buf, weight, output_buf);
                store(output_buf,  output);
        }
}
```

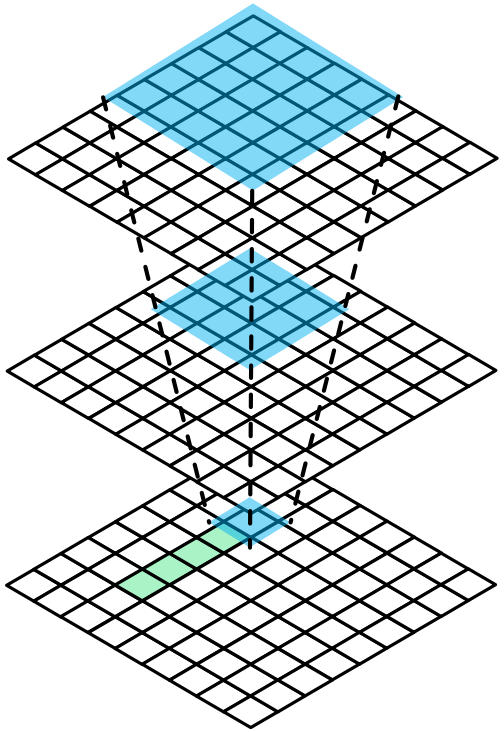


Code Generator



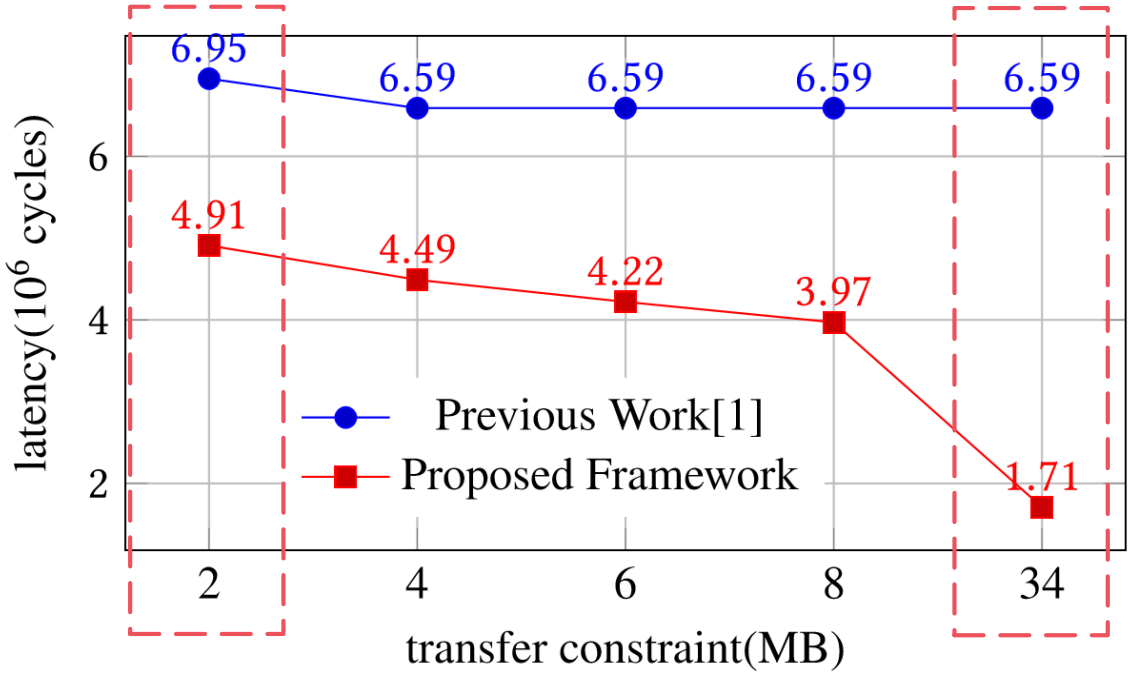
Optimal strategy *S*

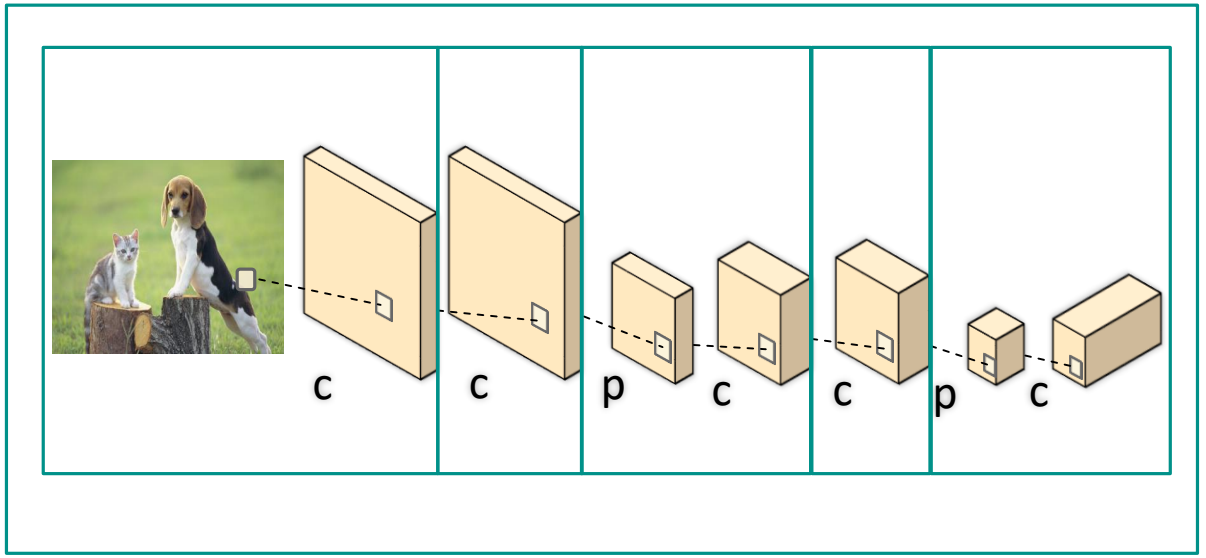
```
Compute_group<…>(input, output, K_1, K_2, K_3){
    #pragma HLS DATAFLOW
    padding<…>(input, conv_in_1);
    conventional_conv<…>(conv_in_1, conv_out_1, K_1);
    padding<…>(conv_out_1, conv_in_2);
    winograd_conv<…>(conv_in_2, conv_out_2, K_2);
    pooling<…>(conv_out_2, pooling_out);
    padding<…>(pooling_out, conv_in_3);
    conventional_conv<…>(conv_in_3, output, K_3);
}
```



Case Study of VGG

- Xilinx Zynq ZC706

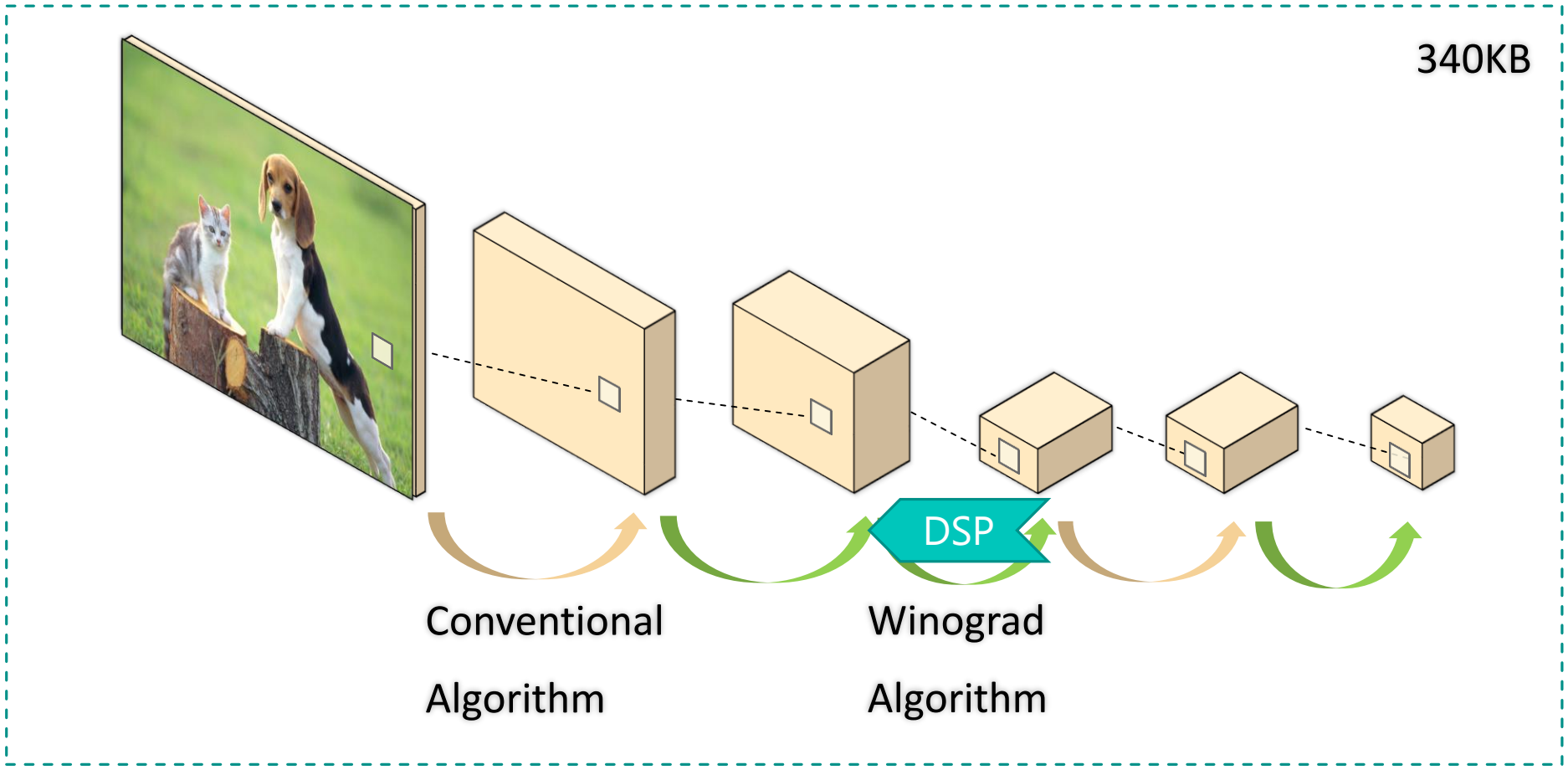




660 GOPS effective performance

[1] M. Alwani, H. Chen, M. Ferdman, and P. Milder. Fused-layer cnn accelerators. In MICRO, 2016.

Case Study of AlexNet



Conclusion

- Accelerating CNNs on FPGAs
- A framework that helps in exploring heterogeneous algorithms
- line-buffer-based architecture / dynamic programming algorithm / code generator

Q & A

Thank you all! Any questions？