Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Network

Chen Zhang[1], Peng Li[3], Guangyu Sun[1,2], Yijin Guan[1], Bingjun Xiao[3], Jason Cong[1,2,3]

[1]Peking University
[2]PKU/UCLA Joint Research Institution
[3]University of California, Los Angeles

Convolutional Neural Network

Automotive Safety

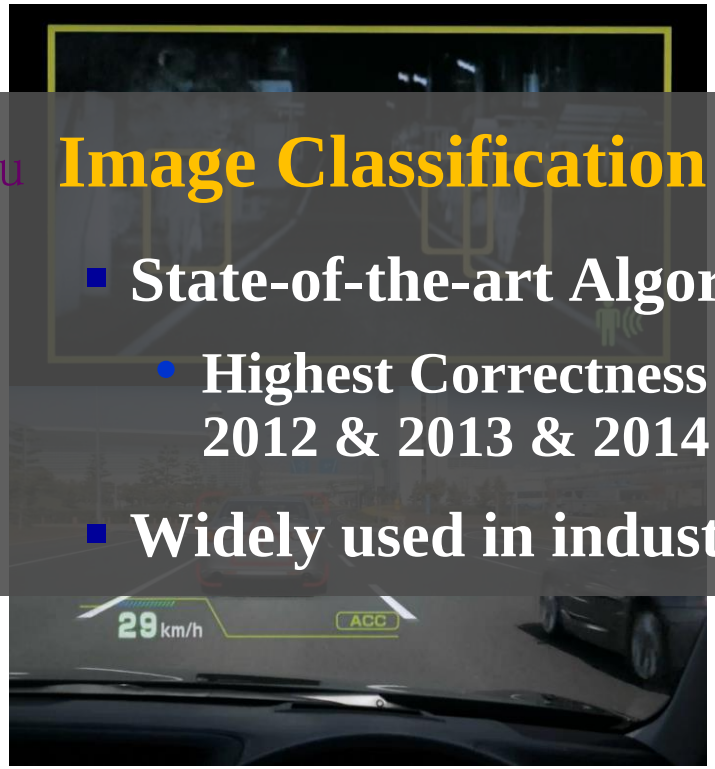


Image Descriptions @ Stanford

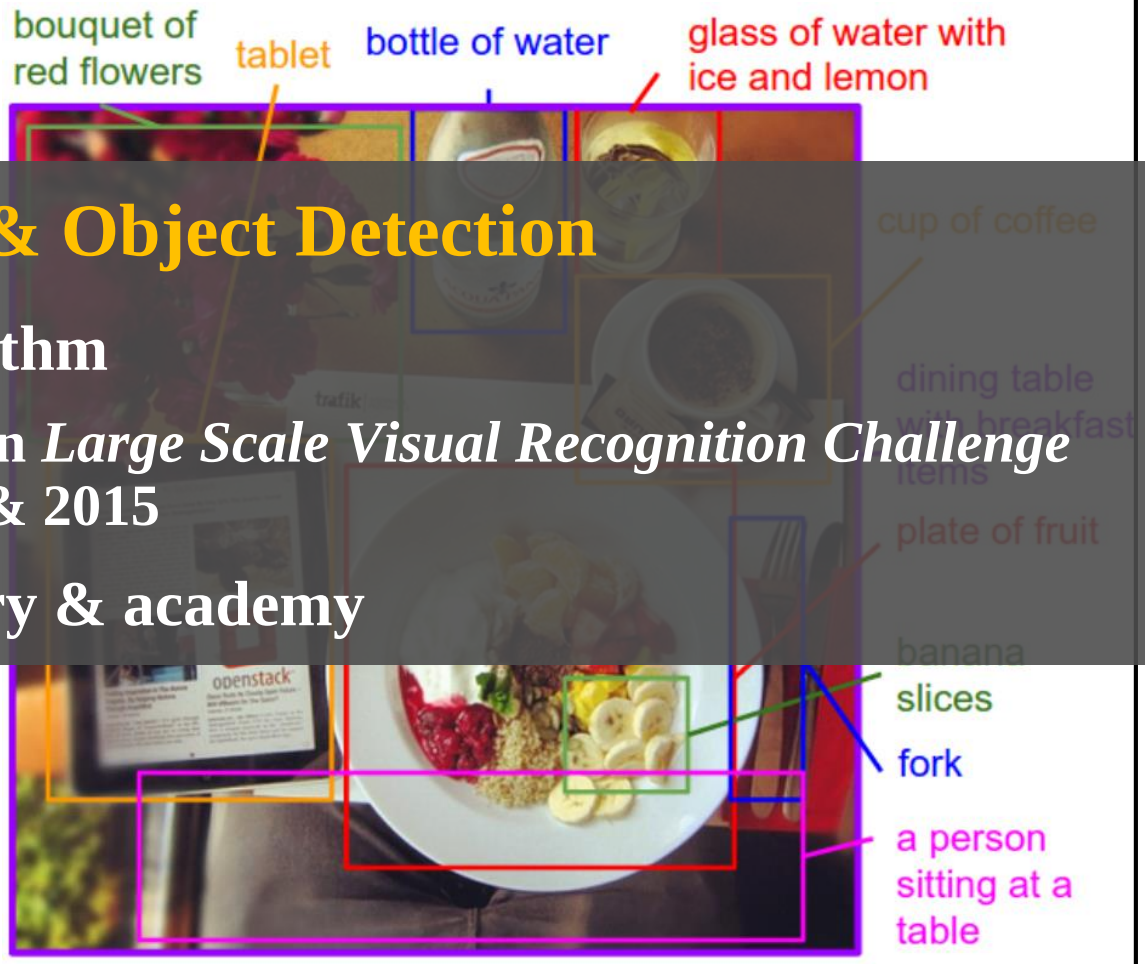
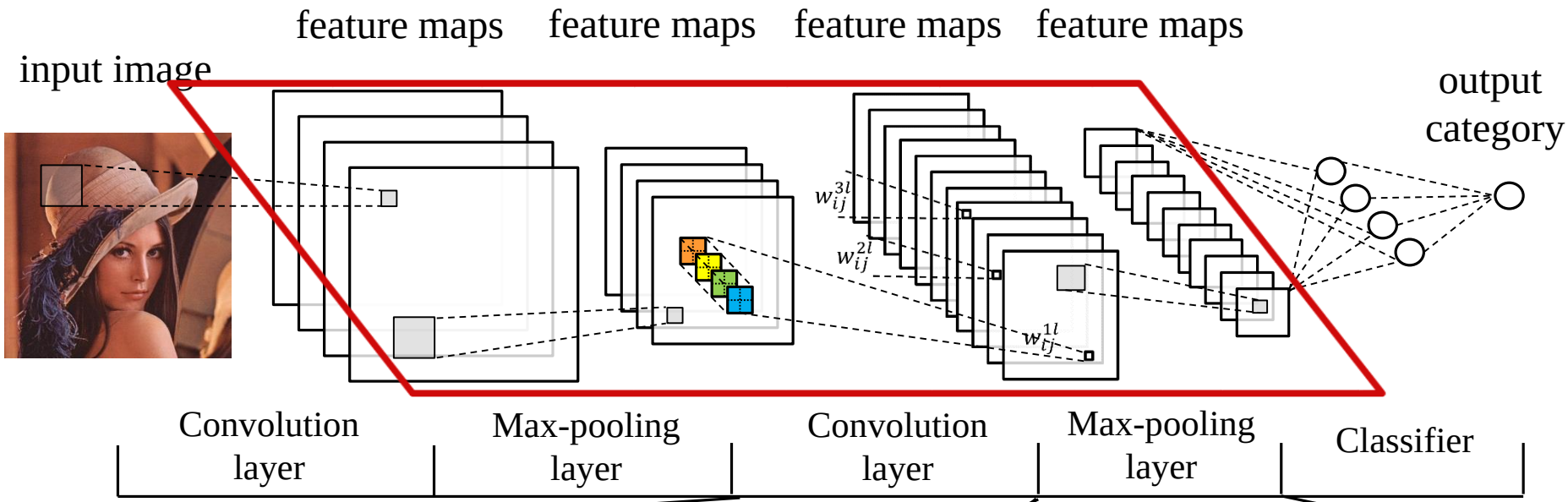
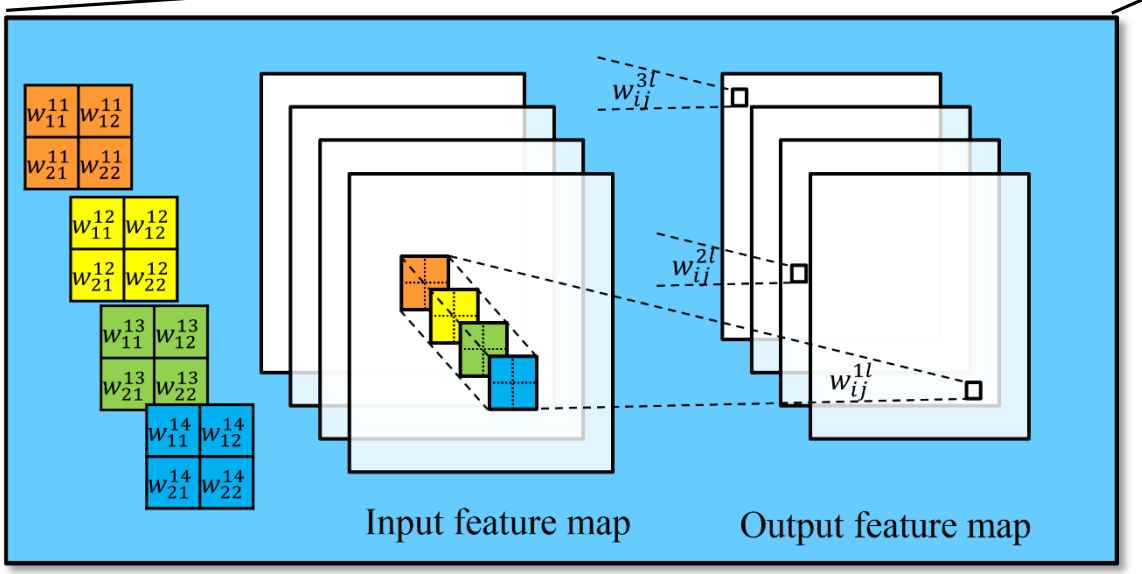


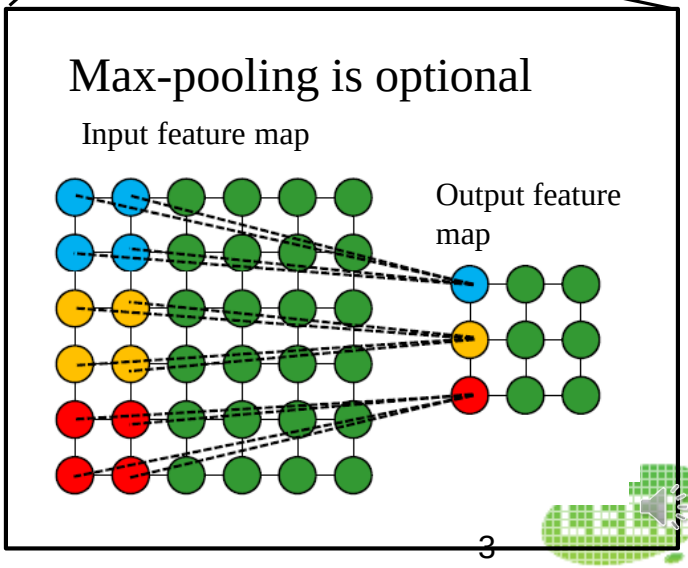
Image Classification & Object Detection

- **State-of-the-art Algorithm**
  - **Highest Correctness in *Large Scale Visual Recognition Challenge* 2012 & 2013 & 2014 & 2015**
- **Widely used in industry & academy**

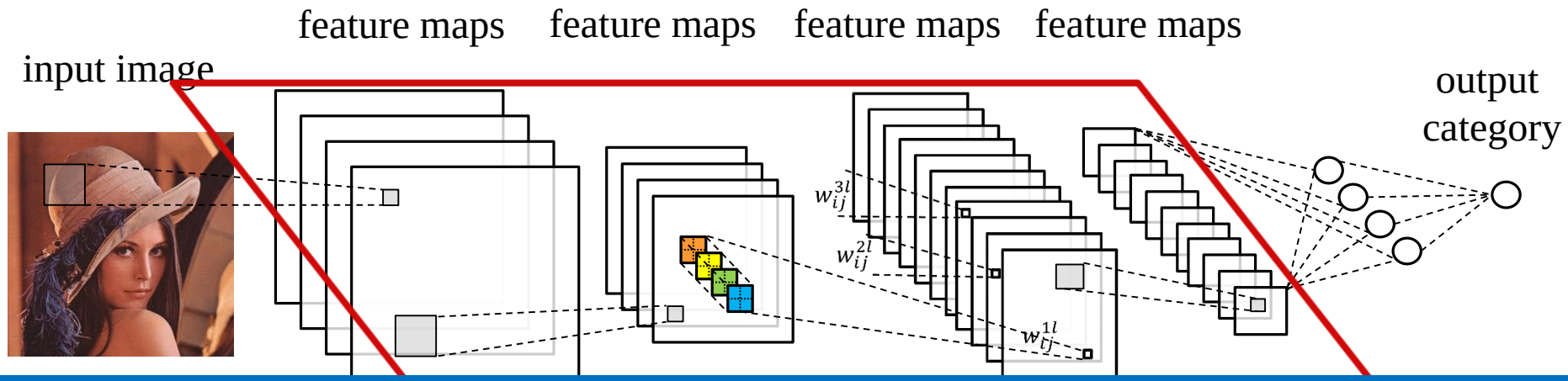
Convolutional Neural Network





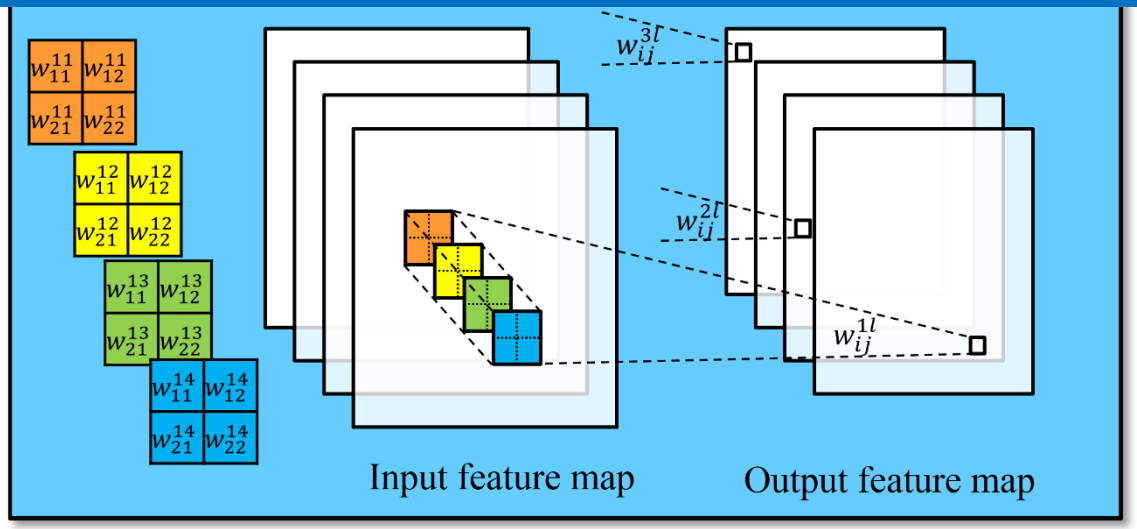


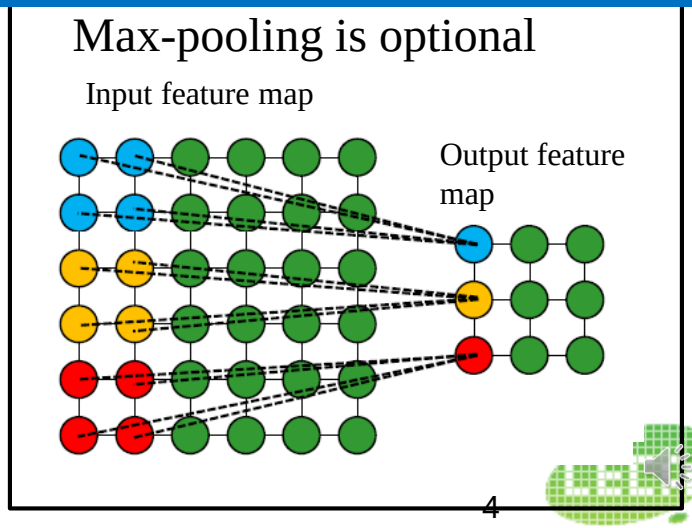
Convolutional Neural Network



Convolutional layers account for over 90% computation

[1] A. Krizhevsky, etc. Imagenet classification with deep convolutional neural networks. NIPS 2012.
[2] J. Cong and B. Xiao. Minimizing computation in convolutional neural networks. ICANN 2014

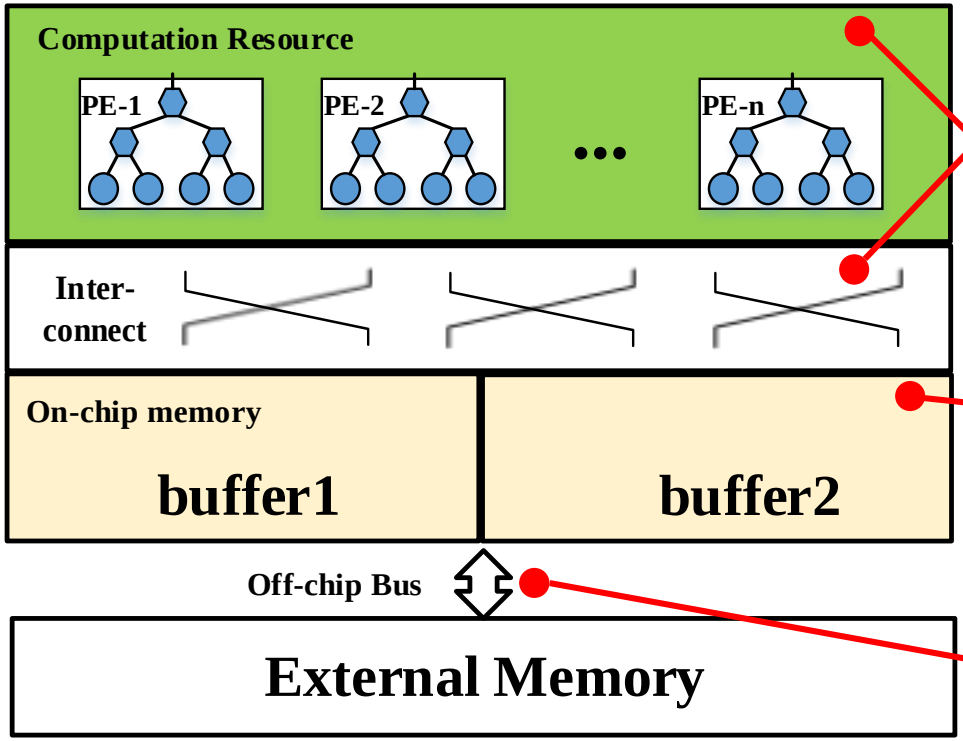




Related Work

1. **CNP: An FPGA-based processor for convolutional networks.** FPL 2009;
2. **A massively parallel coprocessor for convolutional neural networks.** ASAP 2009;
3. **A programmable parallel accelerator for learning and classification.** PACT 2010;
4. **A Dynamically Configurable Coprocessor for Convolutional Neural Networks.** ISCA 2010;
5. **Design and Implementation of an FPGA-based Real-Time Face Recognition System. (short paper)** FCCM 2011;
6. **A massively parallel digital learning processor.** NIPS 2008;
7. **NeuFlow: A Runtime Reconfigurable Dataflow Processor for Vision.** CVPRW 2011;
8. **Accelerating deep neural networks on mobile processor with embedded programmable logic. (Poster)** NIPS 2013;
9. **Memory-Centric Accelerator Design for Convolutional Neural Networks.** ICCD 2013.

Hardware Computing on FPGA



Challenge: Resrc. Util.

Solution: Unroll & Pipeline

+

Challenge: BRAM Limits

Solution: Loop Tiling

+

Challenge: BW Limits

Solution: Data reuse

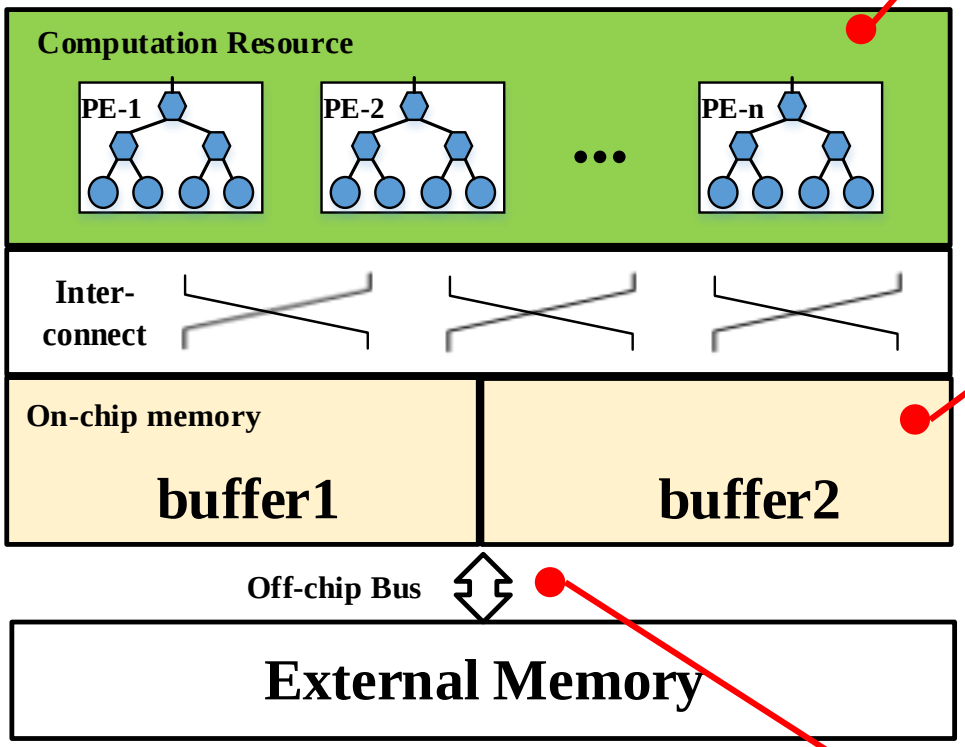
=

Challenge: Co-optimization,
-- match 'computation' & 'communication'

Solution: Roofline Model

Main Contribution

FPGA design in Roofline Model



Computational Perf.

$$= \frac{total\ number\ of\ operations}{execution\ cycles}$$

GFLOP/S

Computation To Communication Ratio

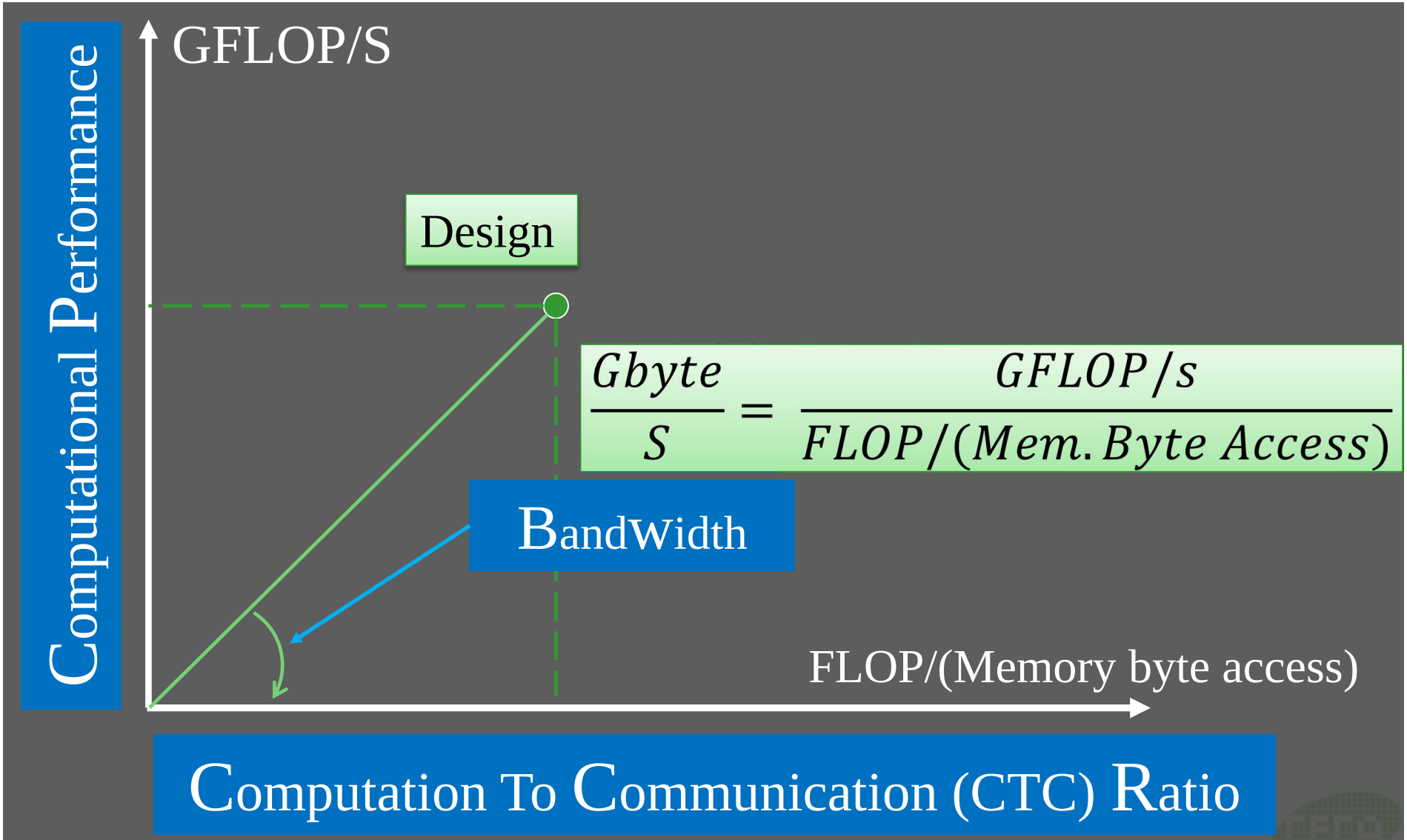
$$= \frac{Total\ number\ of\ operations}{Amount\ of\ external\ data\ access}$$

FLOP / (Mem. Byte Access)

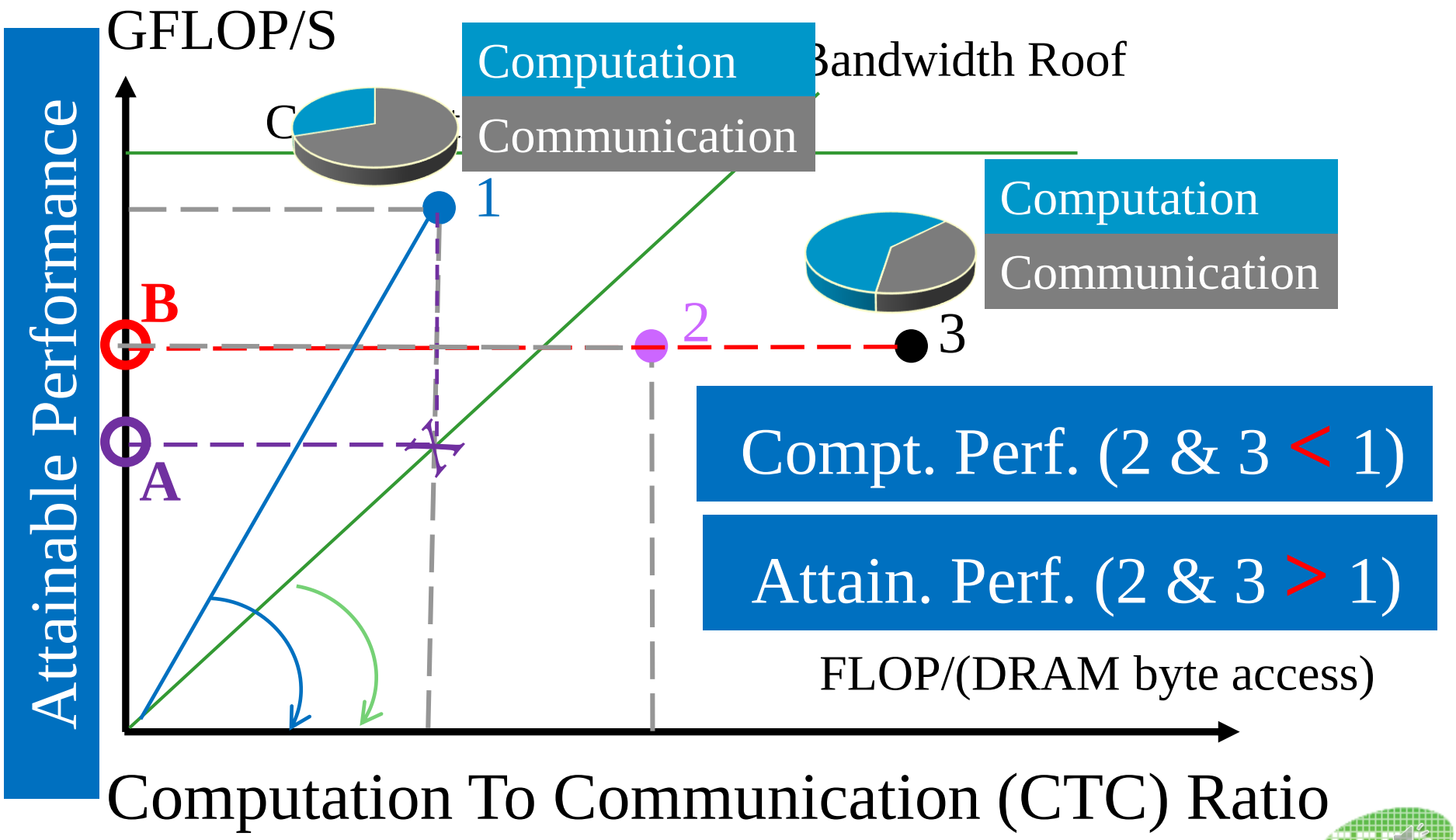
BandWidth

Gbyte/S

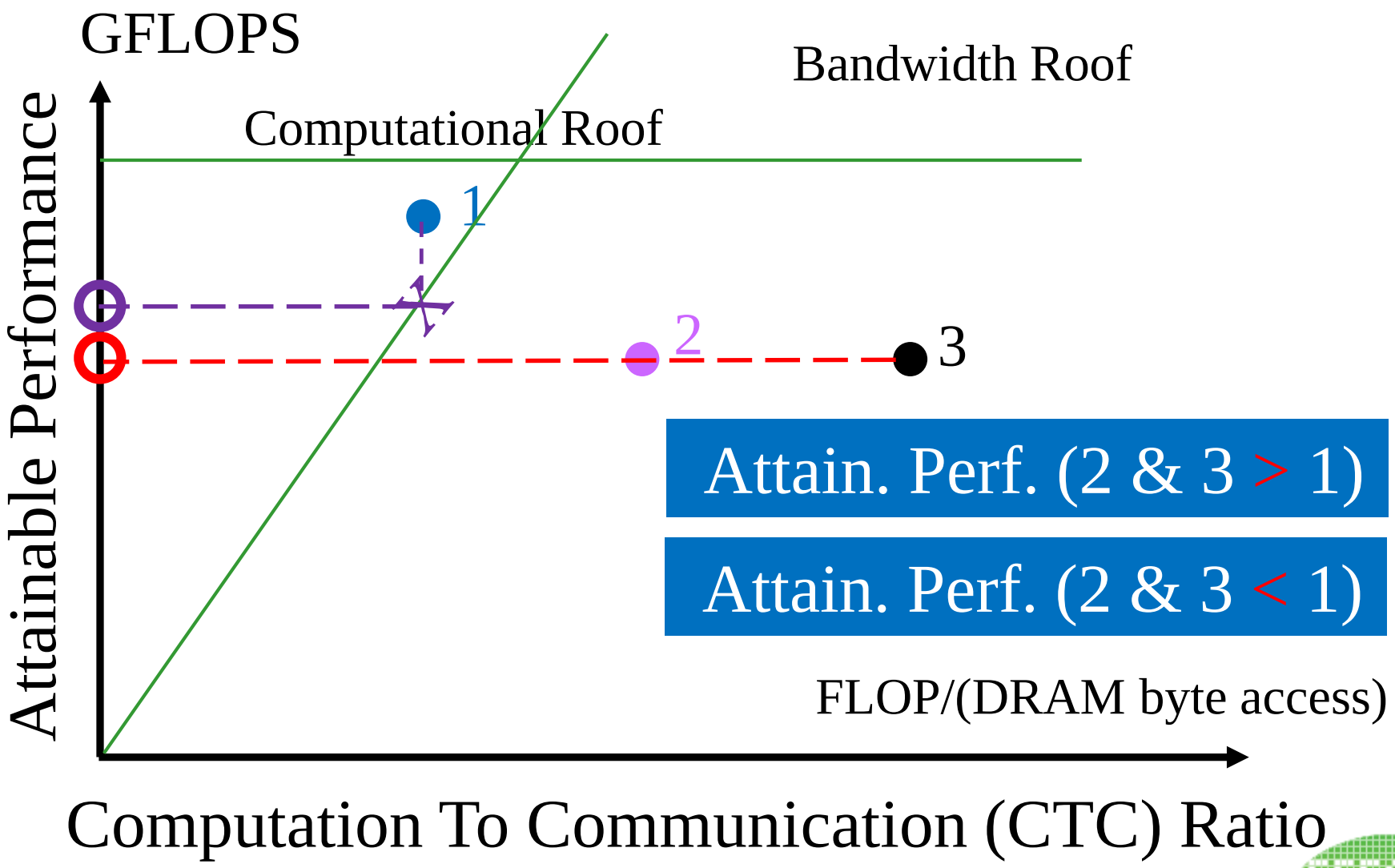
FPGA design in Roofline Model



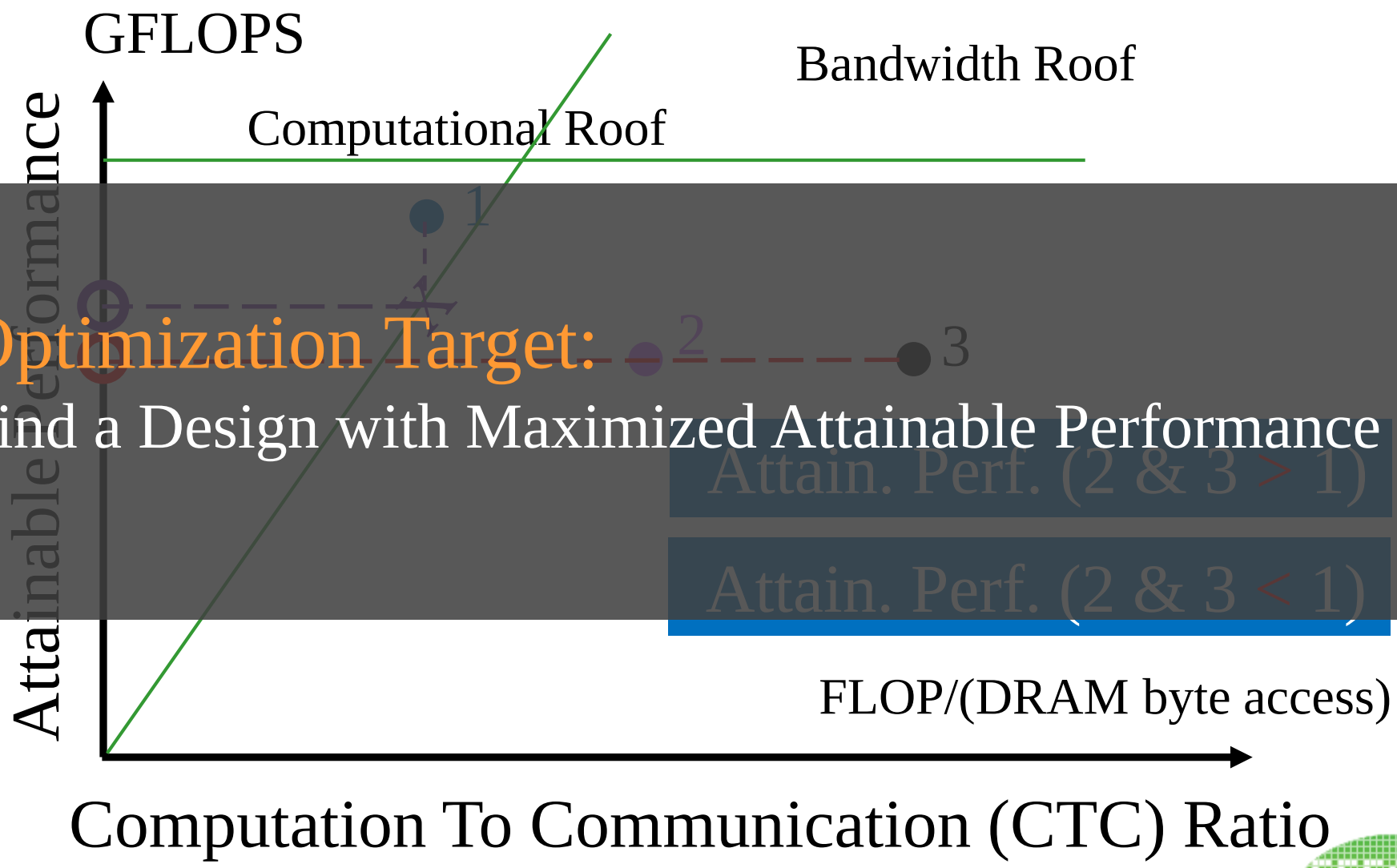
Attainable Performance



Attainable Performance



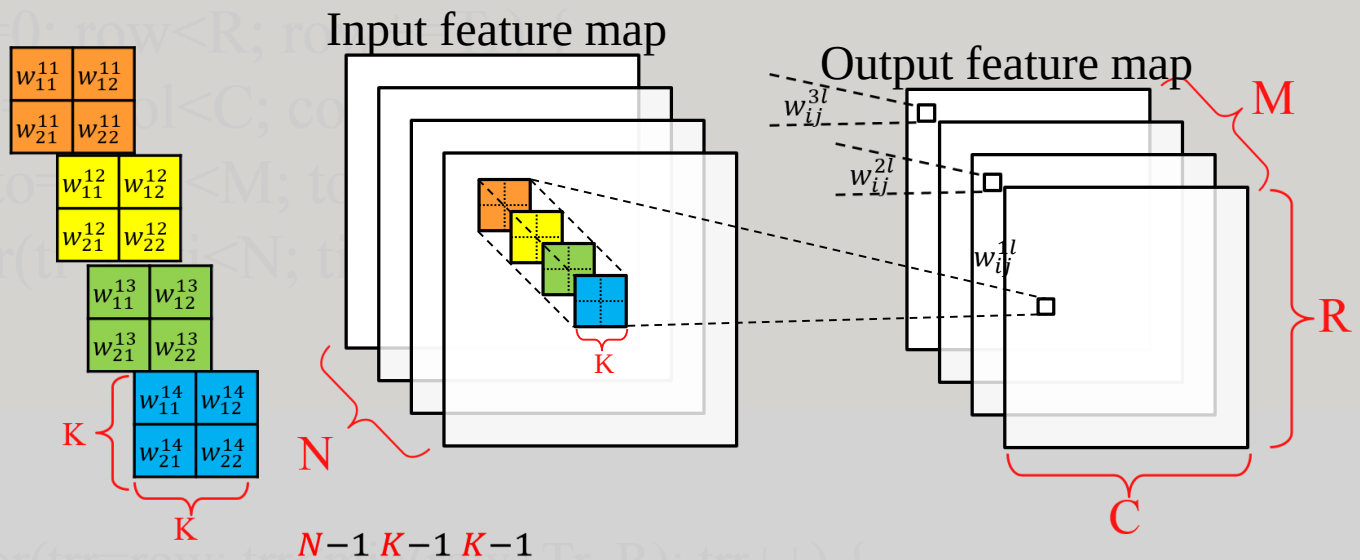
Attainable Performance



Optimization Target:

Find a Design with Maximized Attainable Performance

Loop tiling



$$Y[m][r][c] = \sum_{n=0}^{N-1} \sum_{i=0}^{K-1} \sum_{j=0}^{K-1} W[m][n][i][j] \times X[n][S \times r + i][S \times c + j]$$

```
for(row=0; row<R; row++) {
  for(col=0; col<C; col++) {
    for(to=0; to<M; to++) {
      for(ti=0; ti<N; ti++) {
        for(i=0; i<K; i++) {
          for(j=0; j<K; j++) {
            output_fm[to][row][col] +=
            weights[to][ti][i][j]*input_fm[ti][S*row+i][S*col+j];
}}}}}}
```

R, C, M, N, K, S are all configuration parameters of the convolutional layer

Loop tiling

```
for(row=0; row<R; row+=Tr) {                                  (Tile loop)
  for(col=0; col<C; col+=Tc) {                                (Tile loop)
    for(to=0; to<M; to+=Tm) {                                 (Tile loop)
      for(ti=0; ti<N; ti+=Tn) {                               (Tile loop)
```

Off-chip Data Transfer: Memory Access Optimization

On-chip Data: Computation Optimization

```
        for(trr=row; trr<min(row+Tr, R); trr++) {             (Point loop)
          for(tcc=col; tcc<min(tcc+Tc, C); tcc++) {           (Point loop)
            for(too=to; too<min(to+Tn, M); too++) {           (Point loop)
              for(tii=ti; tii<(ti+Tn, N); tii++) {            (Point loop)
                for(i=0; i<K; i++) {                          (Point loop)
                  for(j=0; j<K; j++) {                        (Point loop)
                    output_fm[to][row][col] +=
                    weights[to][ti][i][j]*input_fm[ti][S*row+i][S*col+j];
          }}}}}}
}}}}
```

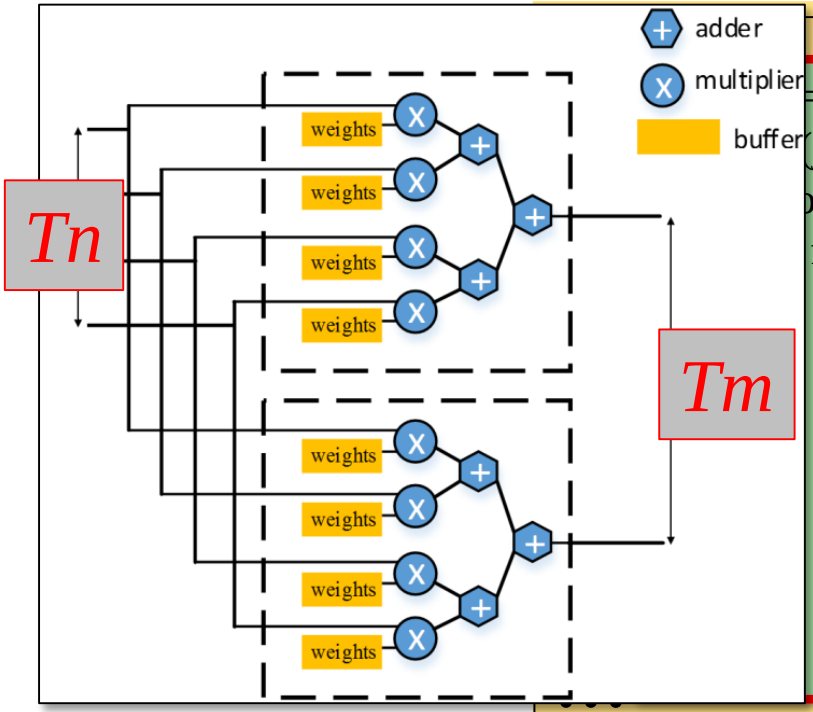
Computation Optimization

(Tile loops)

```
...
for(trr=row; trr<min(row+Tr, R); trr++)                    (Point loop)
  for(tcc=col; tcc<min(tcc+Tc, C); tcc++)                  (Point loop)
    for(too=to; too<min(to+Tn, M); too++)                  (Point loop)
      for(tii=ti; tii<(ti+Tn, N); tii++)                   (Point loop)
        for(i=0; i<K; i++) {
          for(j=0; j<K; j++) {
            output_fm[to][row][col] +=
            weights[to][ti][i][j]*input_fm[ti][S*row+i][S*col+j];
}}}}}}
```

```
for(i=0; i<K; i++) {
  for(j=0; j<K; j++) {
    for(trr=row; trr<min(row+Tr, R); trr++) {
      for(tcc=col; tcc<min(tcc+Tc, C); tcc++) {
#pragma HLS pipeline
        for(too=to; too<min(to+Tm, M); too++) {
#pragma HLS UNROLL
          for(tii=ti; tii<(ti+Tn, N); tii++) {
#pragma HLS UNROLL
            output_fm[to][row][col] +=
            weights[to][ti][i][j]*input_fm[ti][S*row+i][S*col+j];
}}}}}}
```

Computation Optimization



(Tile loops)

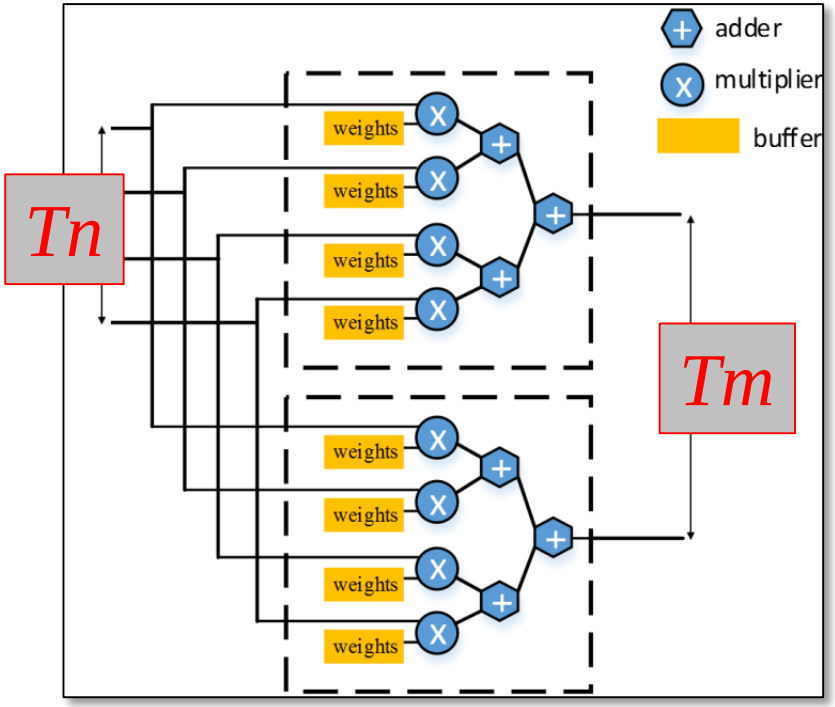
```
=0; i<K; i++) {
j=0; j<K; j++) {
or(trr=row; trr<min(row+Tr, R); trr++) {
 for(tcc=col; tcc<min(tcc+Tc, C); tcc++) {
 HLS pipeline
   for(too=to; too<min(to+Tm, M); too++) {
 HLS UNROLL
      for(tii=ti; tii<(ti+Tn, N); tii++) {
 HLS UNROLL
          output_fm[to][row][col] +=
          weights[to][ti][i][j]*input_fm[ti][S*row+i][S*col+j];
    }}}}}}
```

$$\text{Computational Performance} = \frac{total\ number\ of\ operations}{execution\ cycles}$$

$$\text{execution cycles} = \frac{R}{Tr} \times \frac{C}{Tc} \times \left\lceil \frac{M}{Tm} \right\rceil \times \left\lceil \frac{N}{Tn} \right\rceil \times (K \times K \times Tc \times Tr + P)$$

$$\approx \left\lceil \frac{M}{Tm} \right\rceil \times \left\lceil \frac{N}{Tn} \right\rceil \times R \times C \times K \times K$$

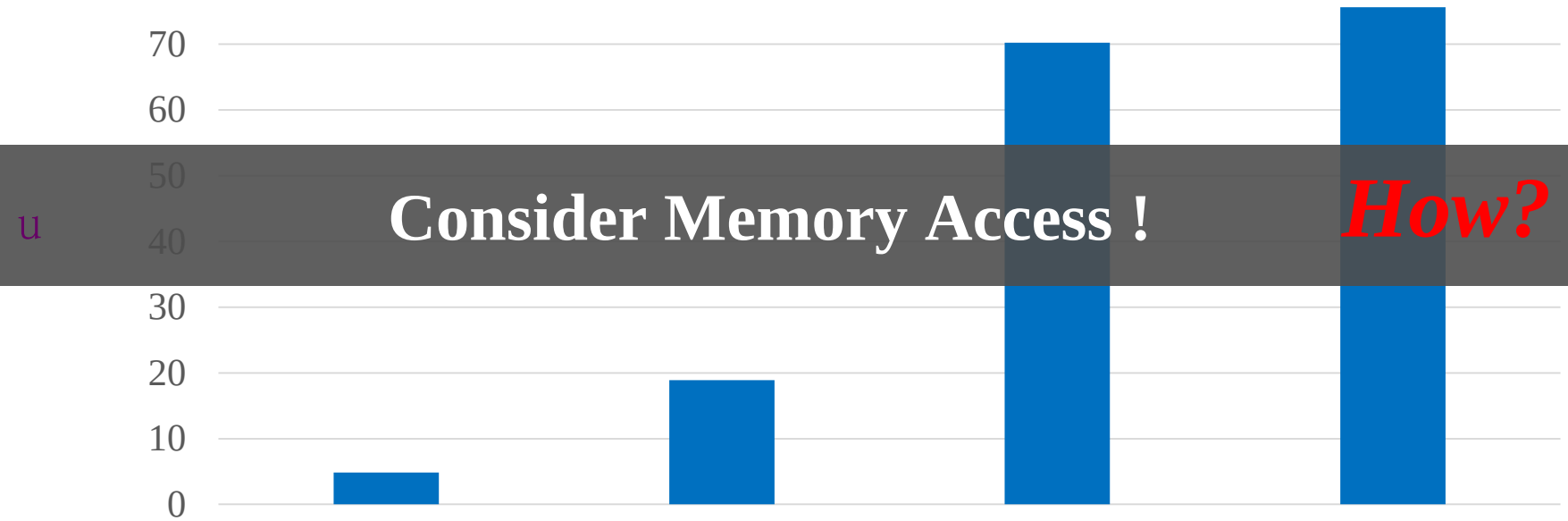
Computational Performance



	Design 1	Design 2	Design 3	Design 4
Output (Tm)	5	10	20	30
Input (Tn)	5	10	20	15
DSPs	125	500	2000	2250

Computational Performance

$$\text{Computational Performance} = \frac{total\ number\ of\ operations}{execution\ cycles}$$



Consider Memory Access !

How?

	Design 1	Design 2	Design 3	Design 4
Output (Tm)	5	10	20	30
Input (Tn)	5	10	20	15
DSPs	125	500	2000	2250

Memory Access Optimization

```
for(row=0; row<R; row+=Tr) {
  for(col=0; col<C; col+=Tc) {
    for(to=0; to<M; to+=Tm) {
      for(ti=0; ti<N; ti+=Tn) {
        load output feature map
        S: foo(output_fm(to, row, col));
        store output feature map
      }
}}}
```

```
for(trr=row; trr<min(row+Tr, R); trr++)
  for(tcc=col; tcc<min(tcc+Tc, C); tcc++)
    for(too=to; too<min(to+Tn, M); too++)
      for(tii=ti; tii<(ti+Tn, N); tii++)
        for(i=0; i<K; i++) {
          for(j=0; j<K; j++) {
            output_fm[to][row][col] +=
              weights[to][ti][i][j]*
              input_fm[ti][S*row+i][S*col+j];
}}}}}}
```

Memory Access Optimization

```
1  for(row=0; row<R; row+=Tr) {
2    for(col=0; col<C; col+=Tc) {
3      for(to=0; to<M; to+=Tm) {
4        for(ti=0; ti<N; ti+=Tn) {
     load output feature map
     S: foo(output_fm(to, row, col));
     store output feature map
     }
}}}
```

```
1  for(row=0; row<R; row+=Tr) {
2    for(col=0; col<C; col+=Tc) {
3      for(to=0; to<M; to+=Tm) {
     load output feature map
4        for(ti=0; ti<N; ti+=Tn) {
     S: foo(output_fm(to, row, col));
     }
     store output feature map
}}}
```

Before local memory promotion

'*output_fm*' memory access

$$= 2 \times \frac{R}{Tr} \times \frac{C}{Tc} \times \boxed{\frac{M}{Tm} \times \frac{N}{Tn}} \times Size_{array}$$

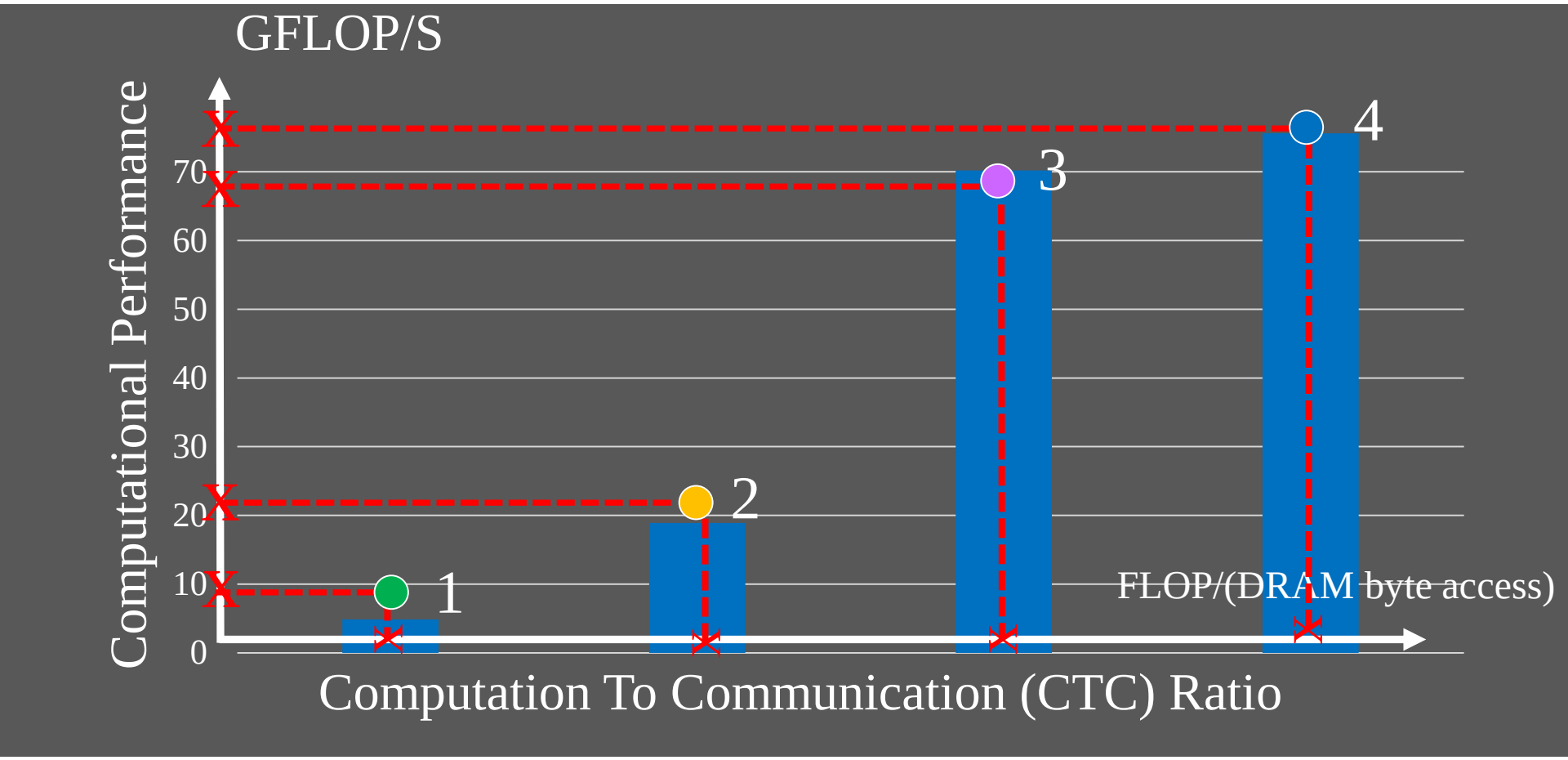
After local memory promotion

'*output_fm*' memory access

$$= 2 \times \frac{R}{Tr} \times \frac{C}{Tc} \times \frac{M}{Tm} \times Size_{array}$$

$$\textit{Computation to Communication Ratio} = \frac{\textit{Total number of operations}}{\textit{Total amount of external data access}}$$

Computation To Communication Ratio



	Design 1	Design 2	Design 3	Design 4
Output (Tm)	5	10	20	30
Input (Tn)	5	10	20	15

Design Space

Computation Engine:

Constraints for CNN configurations:

$Tm \in (Integer, 1 < Tm < M)$ N=128

$Tn \in (Integer, 1 < Tn < N)$ N=192

Constraints for FPGA resource:

$Tn \times Tm \in (Integer, 1 < Tm \times Tn < \#\ of\ PE)$

# of PE = 450

Legal Solutions of Tm & Tn:

2097

×

Communication:

of memory access methods

Legal Solutions:

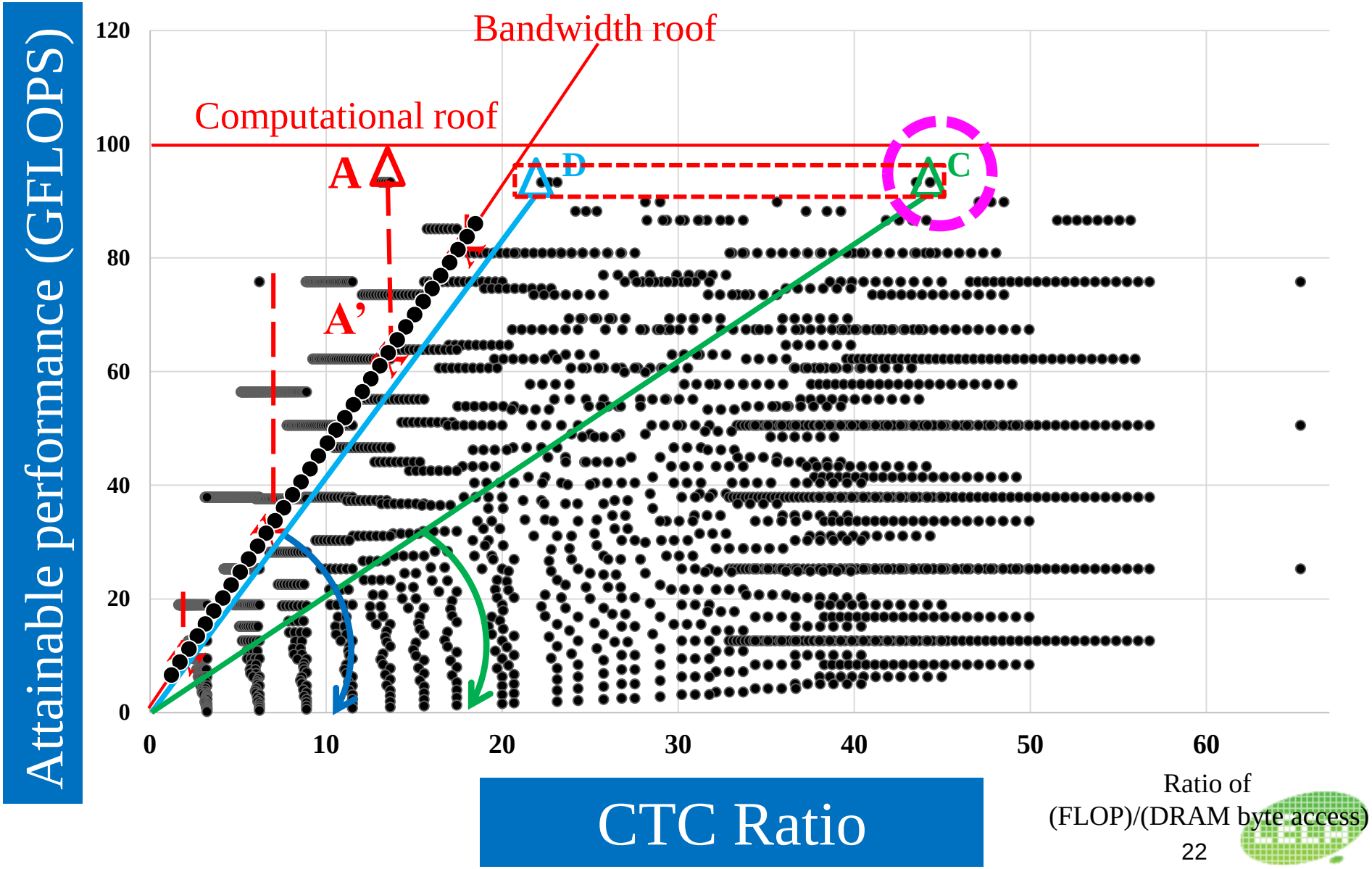
3

=

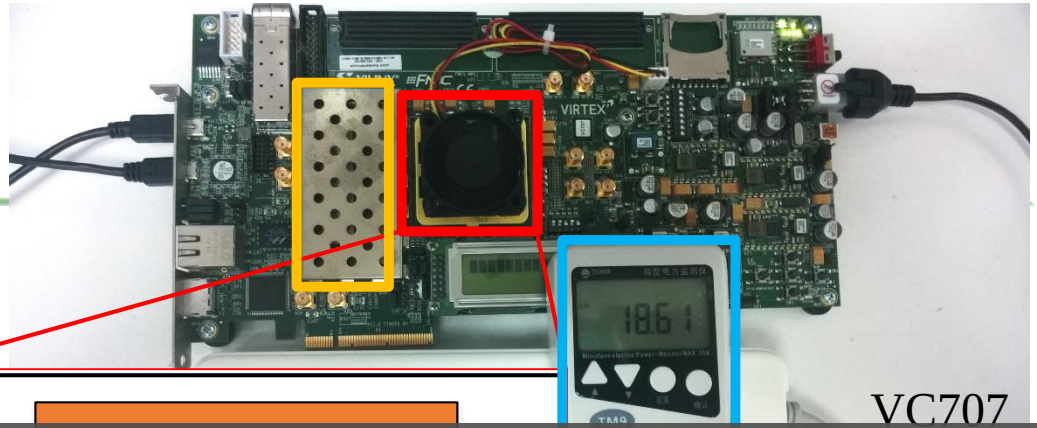
Total Legal Solutions:

6291

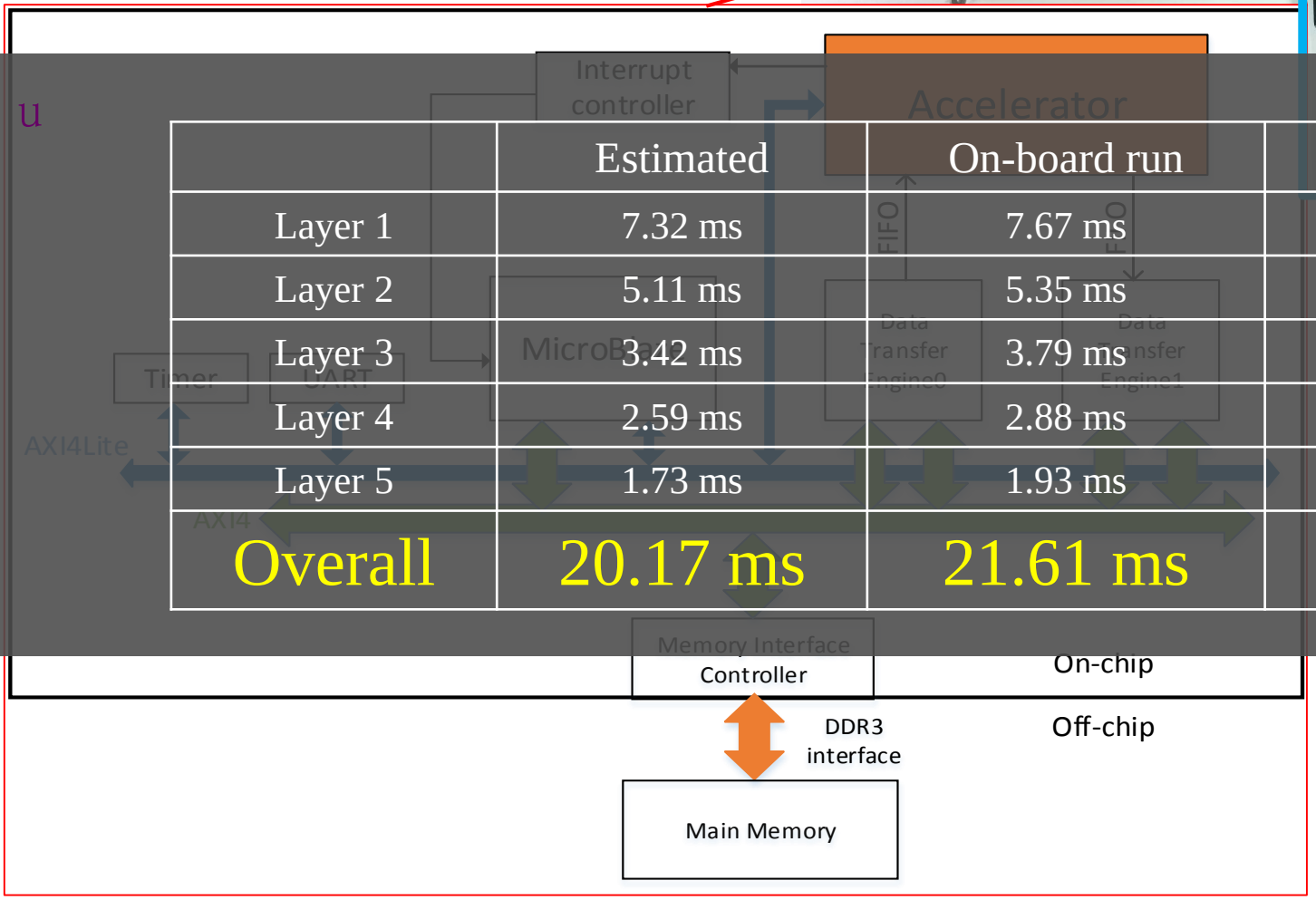
Design Space Exploration



System Overview



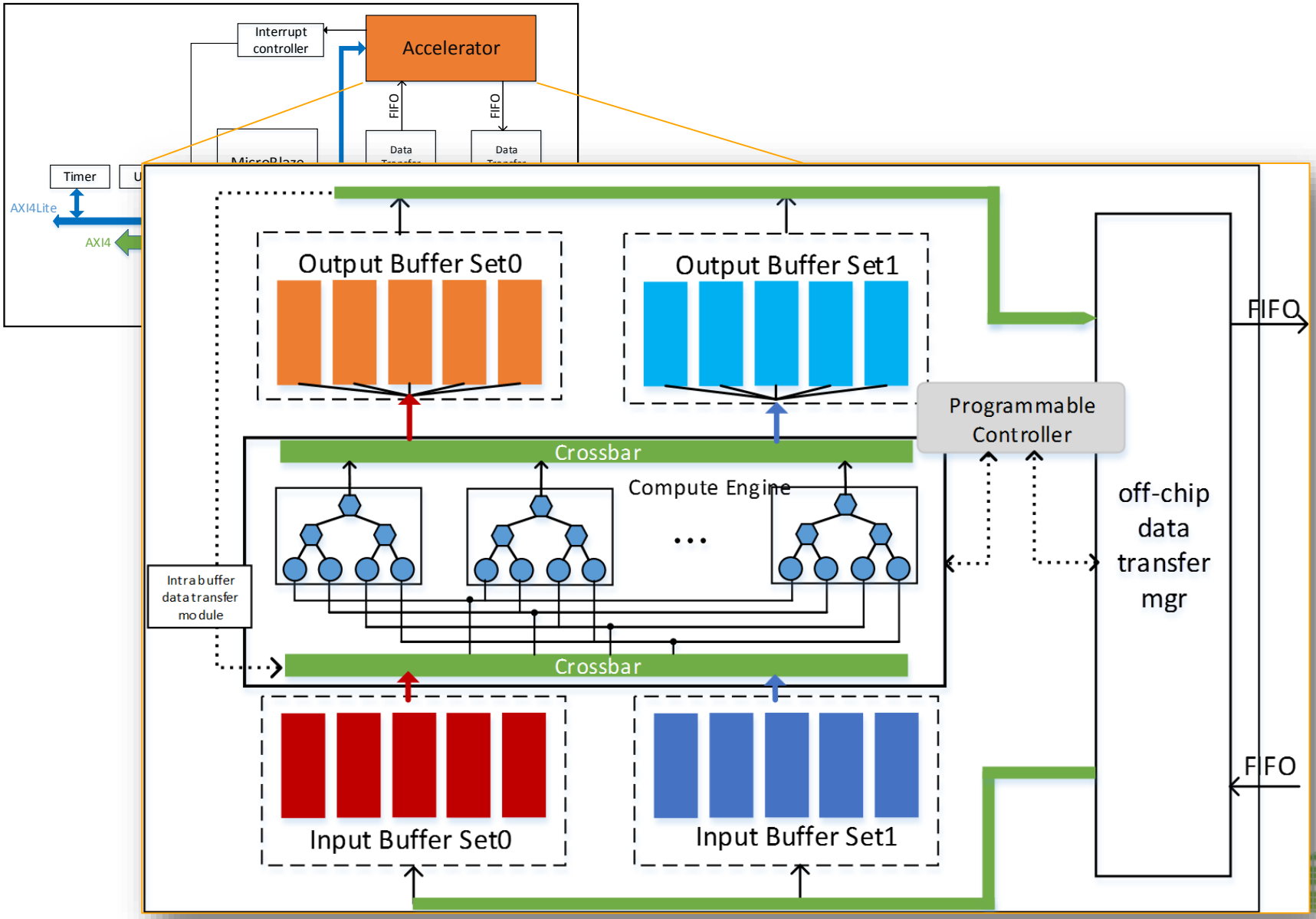
VC707



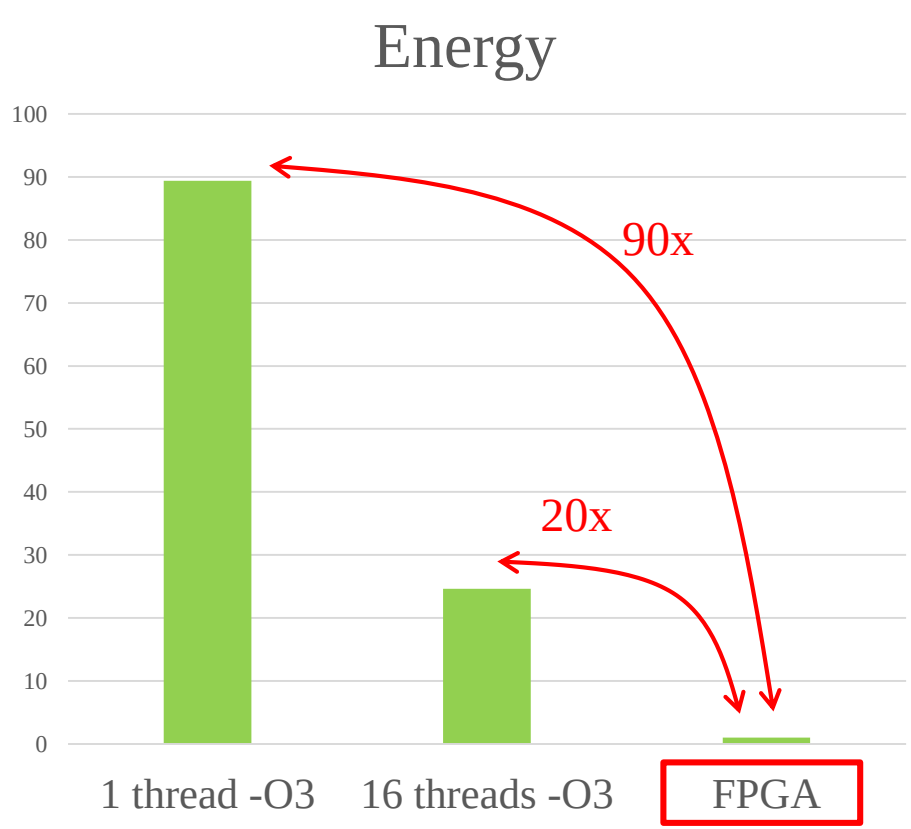
	Estimated	On-board run	Difference
Layer 1	7.32 ms	7.67 ms	~5%
Layer 2	5.11 ms	5.35 ms	~5%
Layer 3	3.42 ms	3.79 ms	~9%
Layer 4	2.59 ms	2.88 ms	~10%
Layer 5	1.73 ms	1.93 ms	~10%
Overall	20.17 ms	21.61 ms	~7%

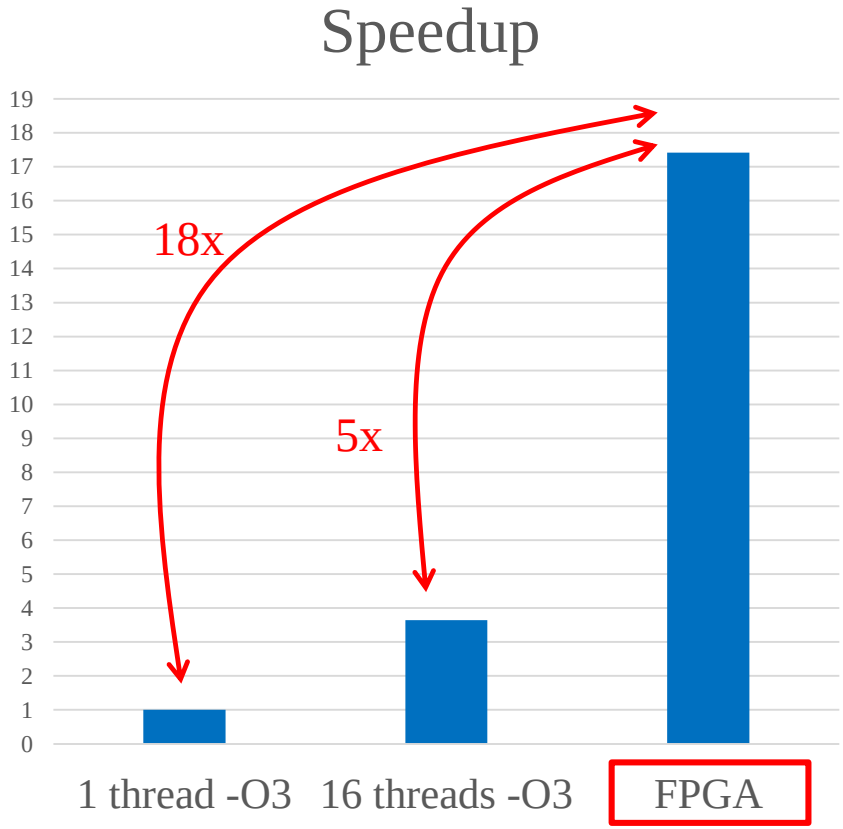
Vivado 2013.4
Vivado HLS 2013.4

Accelerator



Experimental Results: vs. CPU



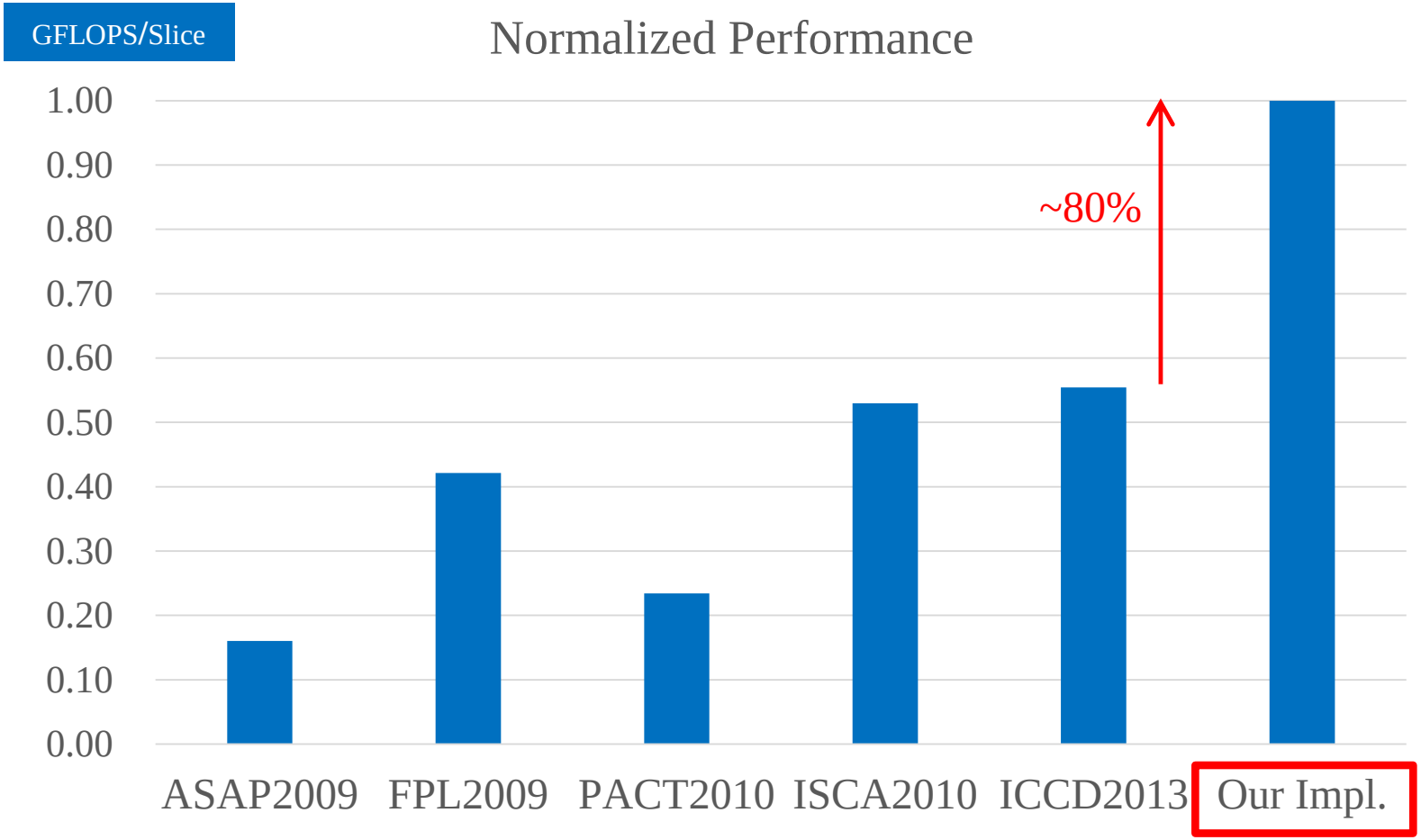


CPU	Xeon E5-2430 (32nm)	16 cores	2.2 GHz	gcc 4.7.2 –O3 OpenMP 3.0
FPGA	Virtex7-485t (28nm)	448 PEs	100MHz	Vivado 2013.4 Vivado HLS 2013.4

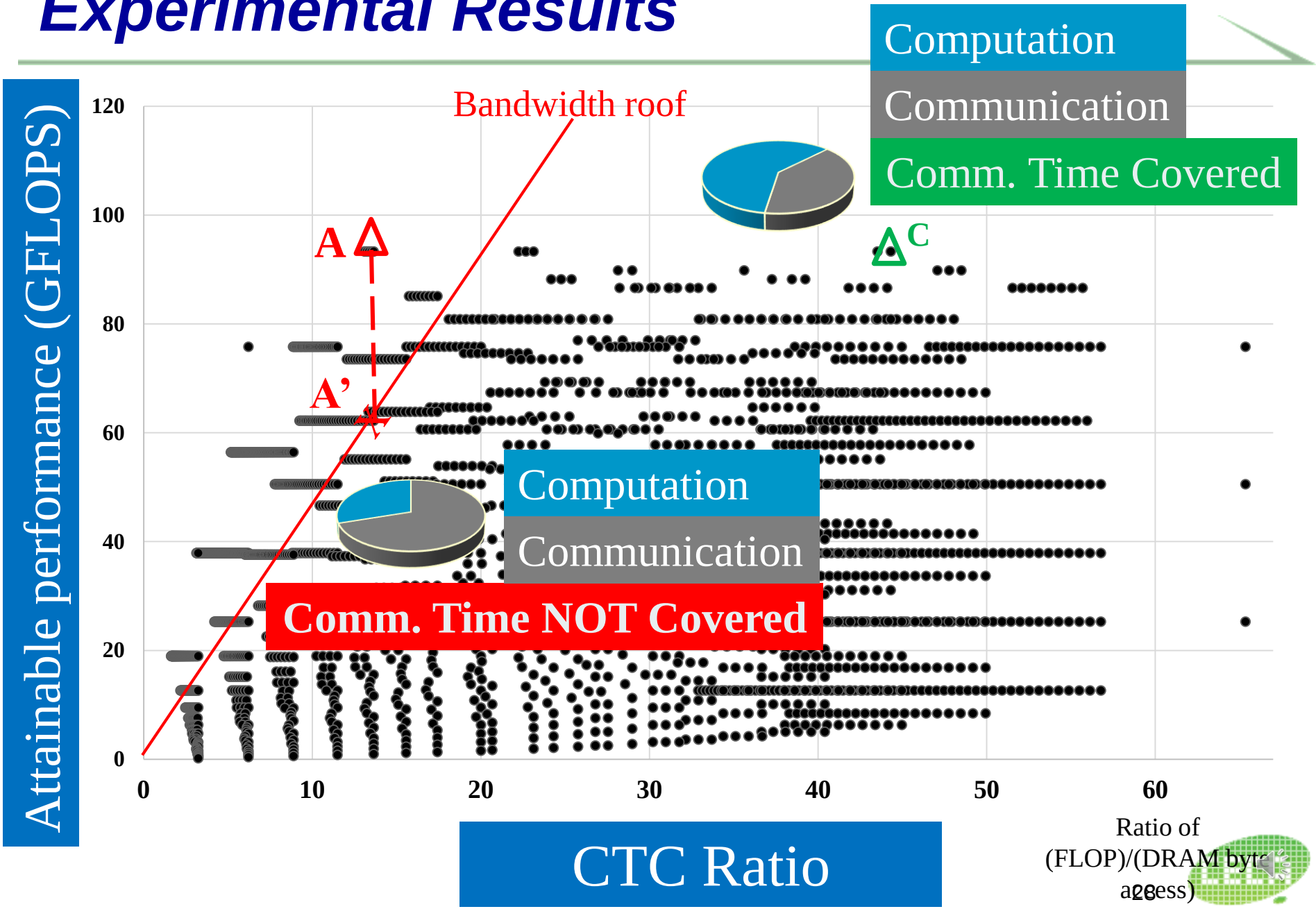
Experimental Results: vs. Other FPGAs

FPL2009	ASAP2009	PACT2010	ISCA2010	ICCD2013	Our Impl.
Virtex 4	Virtex 5	Virtex 5	Virtex 5	Virtex 6	Virtex 7
125MHz	115MHz	125MHz	200MHz	150MHz	100MHz
5.25 GOPS	6.74 GOPS	7 GOPS	16 GOPS	17 GOPS	61.6 GOPS

Experimental Results: vs. Other FPGAs



Experimental Results



Conclusions

An accelerator for convolutional neural network

- **Contribution:**
  - **Accurate Analytical model for computation & communication**
  - **Find the best solution with roofline model**
- **Result:**
  - **On-board run implementation**
  - **~3.5x better performance over other FPGA implementations**
  - **~80% performance/area improvement**

Thank You

Q & A