国防科技大学
National university of defense technology

Towards a Uniform Template-based Architecture for Accelerating 2D and 3D CNNs on FPGA

Junzhong Shen, You Huang, Zelong Wang, Yuran Qiao, Mei Wen, Chunyuan Zhang

National University of Defense Technology, China

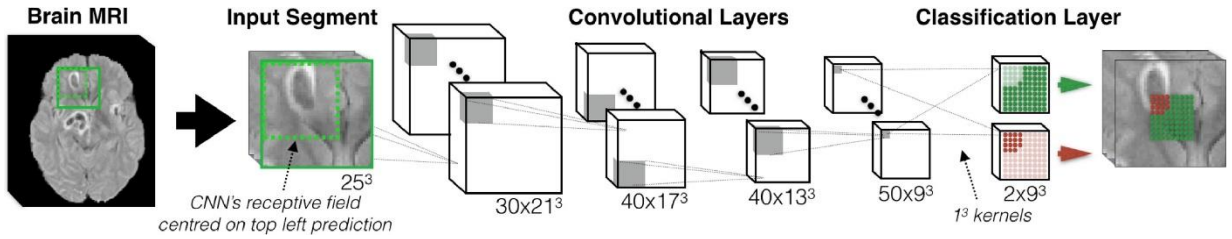
Feb 26, 2018

Outline

- 3D CNNs
- Motivation
- Template-based methodology
- Design Space Exploration
- Evaluations
- Future Work

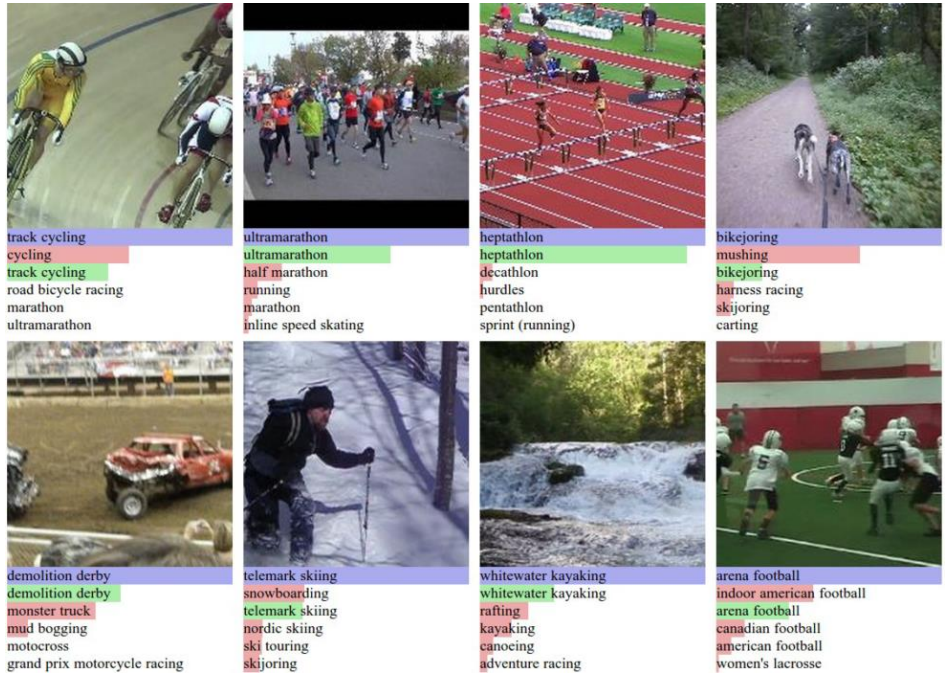
3D CNNs: Efficient method for complicated computer vision applications

- Wide range of applications…



Medical image analysis [1]

Human action recognition [2]



Video classification [3]

Features of 3D CNNs

- Evaluation on a classical 3D CNN model (C3D [4])
  - CONV layers are computationally-intensive
  - FC layers are memory intensive
  - Large amount of intermediate data
  - Large number of computations

→ Similar to 2D CNN

→ More sensitive to memory bandwidth

→ Require higher computation power

Layers	Ops (GFLOPS)	Data Transfer(MB)		Time (ms)
		In+Out	Weights	
CONV	38.4(99.9%)	99.0(27.7%)	17.7(26.7%)	31.9(97.3%)
ReLU	0.0(0.0%)	96.7(27.1%)	0.0(0.0%)	0.0(2.3%)
Pool	0.0(0%)	161.0(45.1%)	0.0(0.0%)	0.0(0.2%)
FC	0.0(0.1%)	0.1%(0.0%)	**48.8(73.3%)**	0.7(0.2%)

Outline

- 3D CNNs
- Motivations
- Template-based methodology
- Design Space Exploration
- Evaluations
- Future Work

Motivations

- Advantages of FPGAs in accelerating CNNs
  - Reconfigurability
  - Higher computation ability, more energy-efficient
  - High-level Synthesis (HLS) tools
- Few work focus on accelerating 3D CNN
  - Ignoring the increasing popularity of 3D CNN
  - Higher computational complexity, greater memory demands
- 2D and 3D CNNs share similar computation pattern
  - Unify 2D/3D CNNs into **a single acceleration framework**
  - Using **uniform templates** for accelerator design

Outline

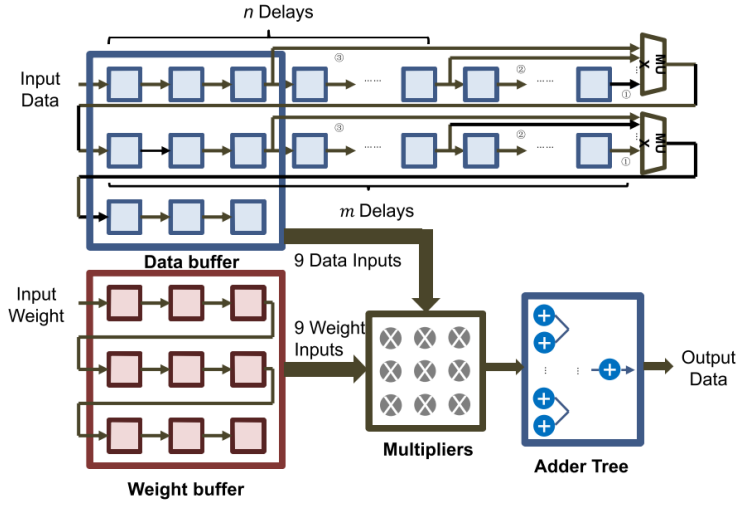
- 3D CNNs
- Motivation
- Template-based methodology
- Design Space Exploration
- Evaluations
- Future Work

Template-based Methodology

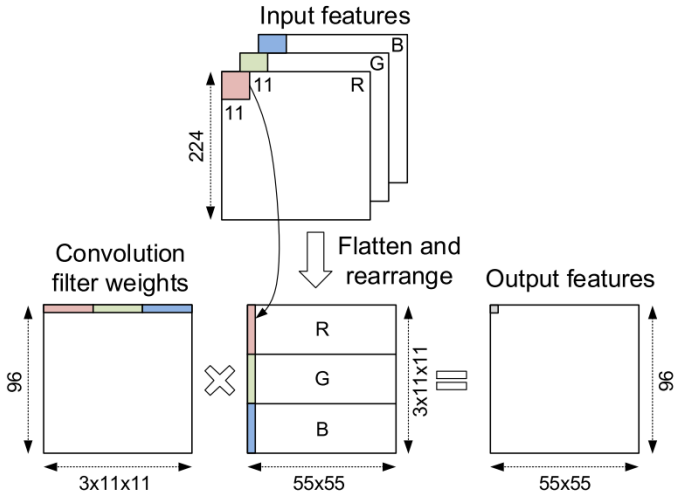
- Overall Flow
  - Algorithm selection
  - Common operations extraction
  - Template designs
  - Template-based architecture
- Advantages
  - Generate accelerators in a short period of time
  - Scalability of template-based design
  - Make it easy to exploit the fine-grained parallelism of CNN algorithms

Template-based Methodology

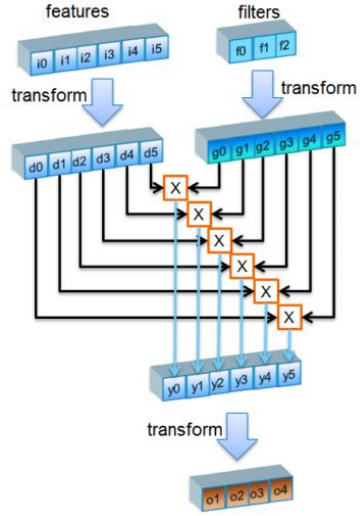
- Algorithm selection



Ordinary convolution [5]



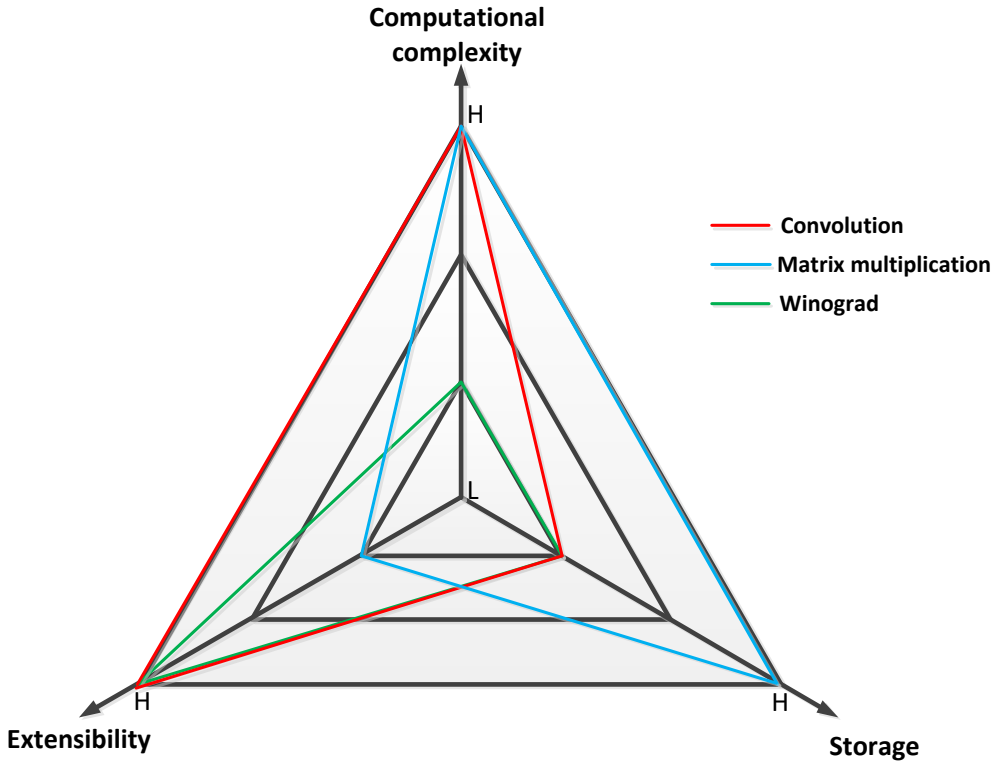
Convert to Matrix Multiplication [6]



Winograd algorithm [7]

Template-based Methodology

- Algorithm selection
  - Computational complexity
  - Extensibility
  - Storage (data replication)

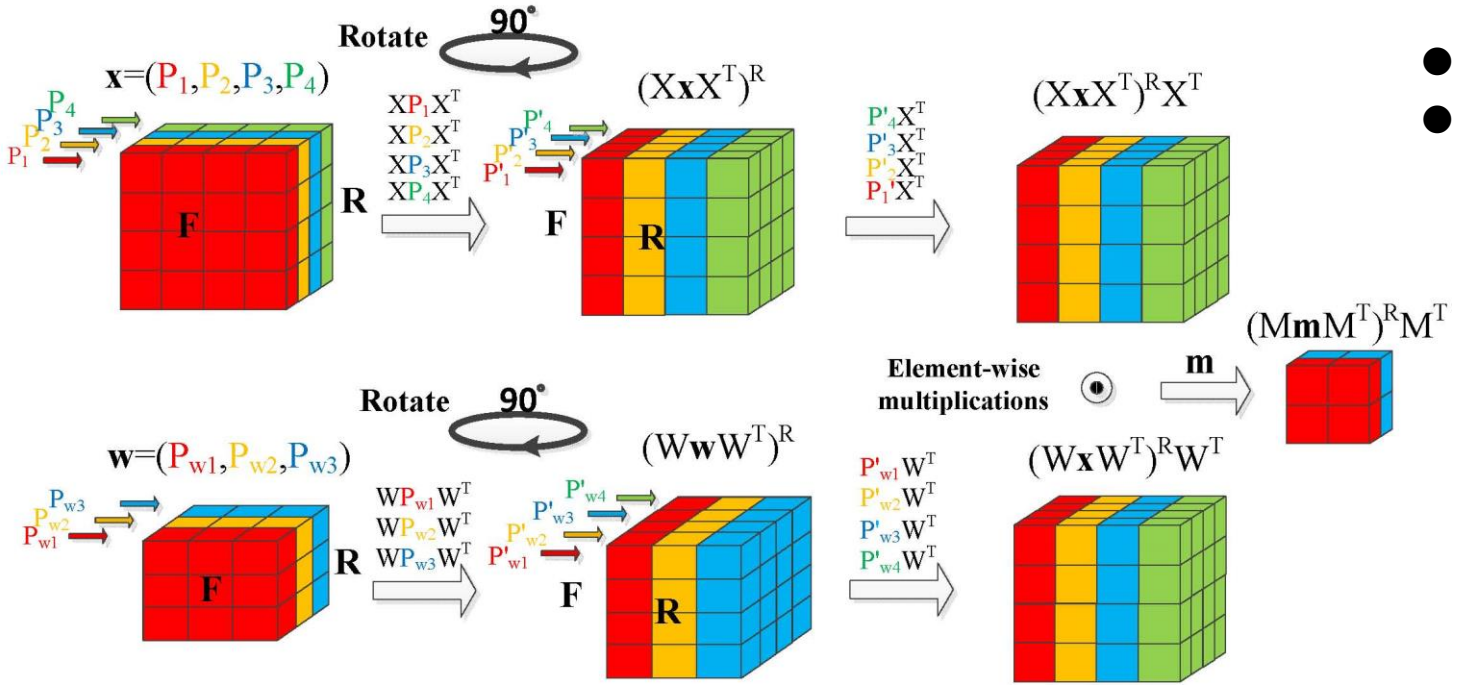


	F(2,3)		$F(2^2,3^2)$		$F(2^3,3^3)$	
Algorithms	muls	adds	muls	adds	muls	Adds
Ordinary	6	4	36	32	216	208
Winograd	4	11	16	77	64	419

Over 3x saving in number of multiplications !

Template-based Methodology

- Winograd algorithm extension



Process of 3D Winograd algorithm ($F(2^3,3^3)$)

- X, W and M are constant matrices
- ⊙ is denoted as element-wise multiplications

1D: $$y = M[(Xx) \odot (Ww)]$$

2D: $$Y = M[(XxX^T) \odot (WwW^T)]M^T$$

3D: $$Z = (M((XxX^T)^R X^T \odot (WwW^T)^R W^T)M^T)^R M^T$$

Template-based Methodology

- Common operations extraction

```
L1: for(row=0;row<R; row+=2){
L2:  for(col=0;col<C; col+=2){
L3:   for(m=0; m<M; m++){
L4:    for(n=0; n<N; n++){
        Load(In, n);  //4*4 input  tile
        Load(W, m ,n); //3*3 filter tile
        2D_Trans_X(In, Tin);
        2D_Trans_W(W, Tw);
        2D_Mul(Tin,Tw, P);
        2D_Trans_M(P, Tp);
        2D_Accumulate(Sum,Tp);
    }
    Send(Out,Sum,m); //2*2 output tile
}}}
```

(a) CONV layers of 2D CNN

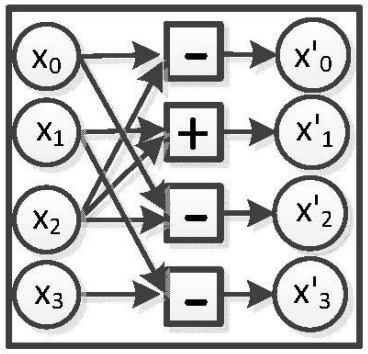
```
L1:for(dep=0;dep<Z;dep+=2){
L2: for(row=0;row<R; row+=2){
L3:  for(col=0;col<C; col+=2){
L4:   for(m=0; m<M; m++){
L5:    for(n=0; n<N; n++){
        Load(In, n);  //4*4*4 input  tile
        Load(W, m ,n); //3*3*3 filter tile
        2D_Trans_X(In, Tin);
        2D_Trans_W(W, Tw);
        Rotate(Tin, Tin^R);
        Rotate(Tw, Tw^R);
        1D_Trans_X(Tin^R, Tin^1 );
        1D_Trans_W(Tw^R, Tw^1 );
        3D_Mul(Tin^1,Tw^1, P);
        2D_Trans_M(P, Tp);
        Rotate(Tp, Tp^R);
        1D_Trans_M(Tp^R, Tp^1);
        3D_Accumulate(Tp^1,Sum);
    }
    Send(Out, Sum,m); //2*2*2 output tile
}}}}
```

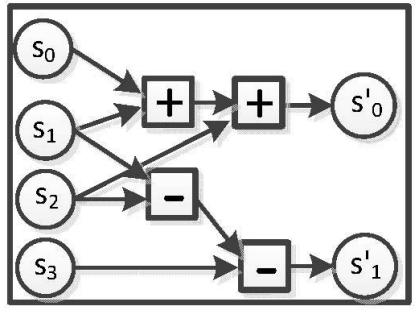
(b) CONV layers of 3D CNN

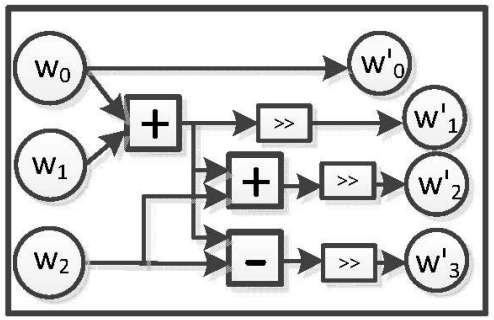
- Both 2D and 3D Winograd transformations can be split into 1D Winograd transformations
- The operator-level parallelism of the element-wise multiplications on the 3D tiles can be performed repeatedly by multiplications on the 2D tiles
- Accumulations of a 3D tile can be performed by accumulations of the 2D tiles

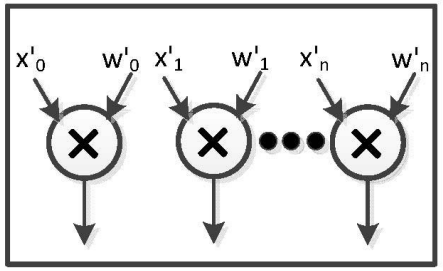
Template-based Methodology

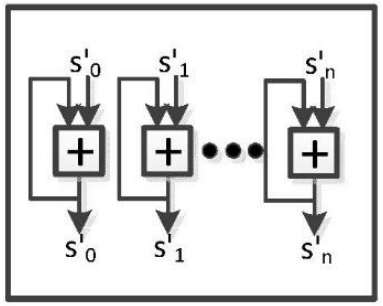
- Template design
  - Building blocks for Winograd algorithm
    - Transformation
    - Element-wise multiplication
    - Accumulation
  - Simple and resource saving
    - Adders
    - Multipliers
    - Shifter





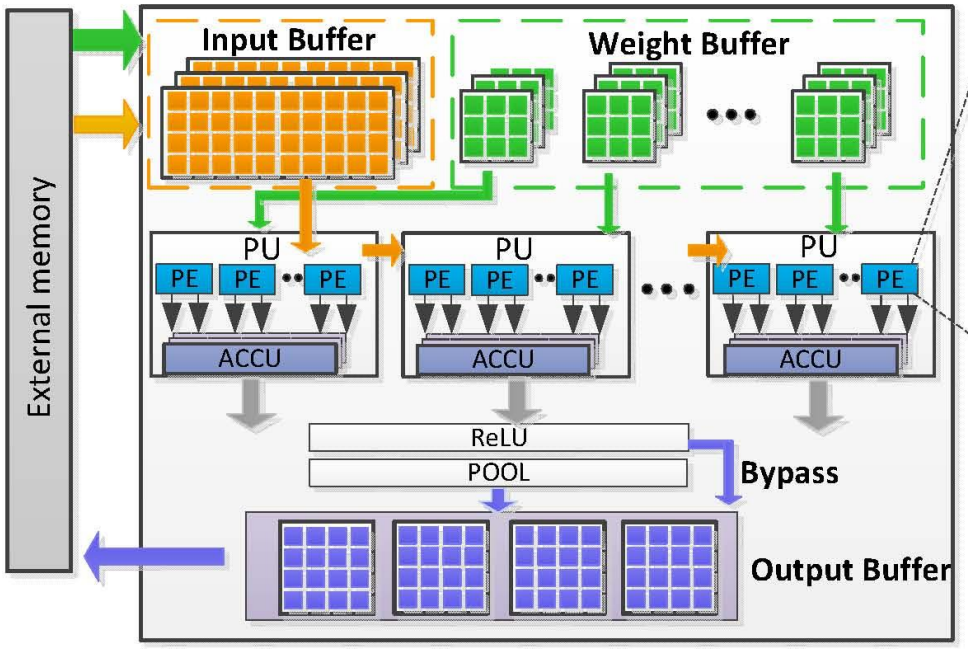






Template-based Methodology

- Architecture

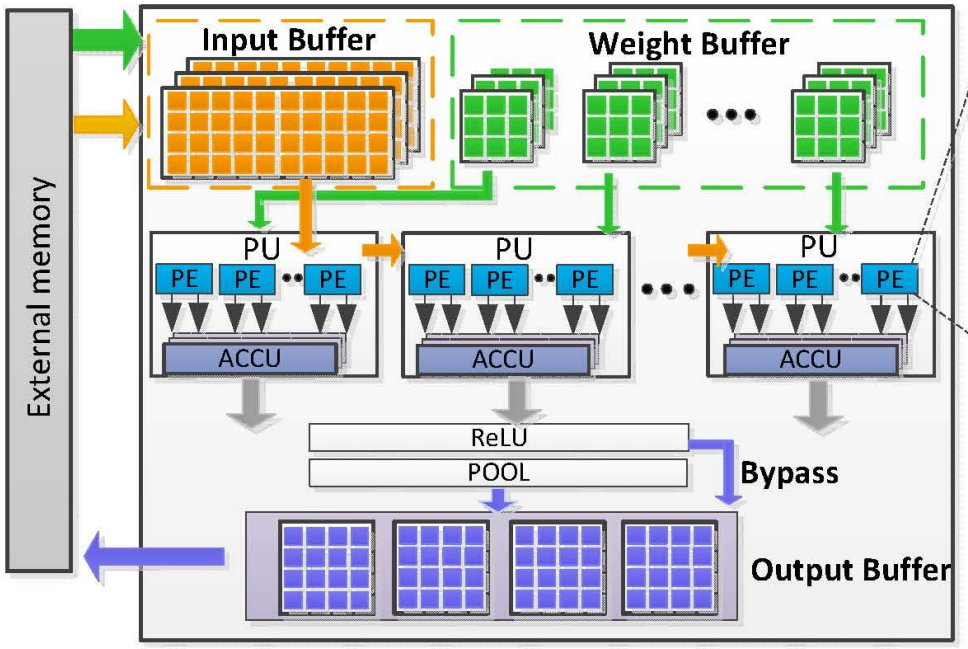


(a)

- On-chip Buffers
  - Tiled input/output feature maps
  - Tiled weights
- Computation Engines
  - Two levels of hierarchy
    - PEs
    - PUs
  - Parallelism
    - Inter kernel parallelism
    - Inter output parallelism

Template-based Methodology

- Architecture



(a)

HLS description of computation engine

```
Computation_Engine<To,Ti>(Tz,Tr,Tc):
L1:for(tz=0;tz < Tz;tz+=2){//loop flattened (only for 3D CNN)
 L2: for(tr=0;tr < Tr;tr+=2){//loop flattened
  L3:  for(tc=0;tc < Tc;tc+=2){
#Pragma HLS PIPELINE
   Load(In,Bin); //load input tiles Bin from In[Ti][Td*Th*Tw]
   Load(W,Bw); //load filter tiles from W[To][Ti][Ksize]
   L4:  for(to=0;to < To; to++){
#Pragma HLS UNROLL
     PU(Bin,Bw,Bout,to,tz,tr,tc){  //parallelism in PUs
    L5:    for(ti=0;ti < Ti; ti++){
#Pragma HLS UNROLL
       PE(Bin,Bw,to,ti); //parallelism in PEs
}
Accumulations(Bout,to,ti); }}
store(Bout,Out);//store output tiles Bout to Out[To][Tz*Tr*Tc]
}}}
```

Parallel strategies

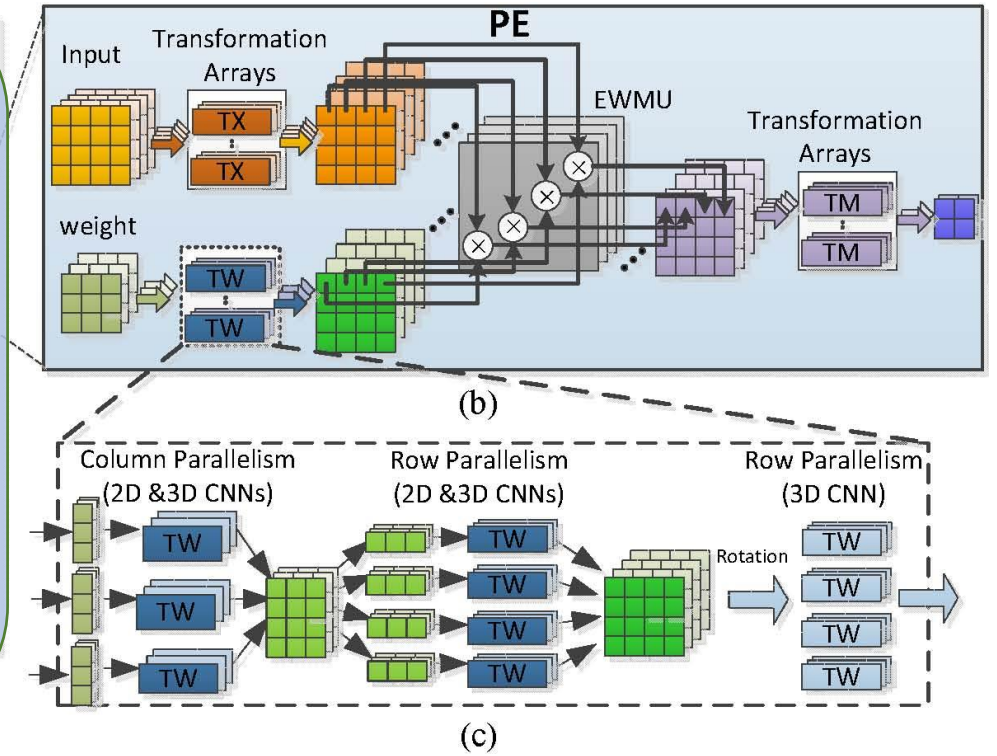
(c)

Template-based Methodology

- Architecture

PE Design

- Performing the Winograd algorithms
  - 2D/3D Winograd
- Organized by proposed templates
  - Transformation arrays
  - Element-Wise Multiplication Unit (EWMU)
- Parallelism
  - Intra kernel (Operation-level) parallelism
  - Intra Output parallelism



Template-based Methodology

- Memory access optimization
  - External memory access

Table 3: Tiling strategies

	Tiling Strategies	# of Transfers*	Burst Length
1	$T_z <= Z, T_c < C, T_r <= R$	$T_d * T_h * \frac{W}{T_w}$	$\frac{T_w * Bw_{on}}{Bw_{off}}$
2	$T_z <= Z, T_c = C, T_r < R$	T_d	$\frac{T_h * W * Bw_{on}}{Bw_{off}}$
3	$T_z <= Z, T_c = C, T_r = R$	1	$\frac{T_d * H * W * Bw_{on}}{Bw_{off}}$

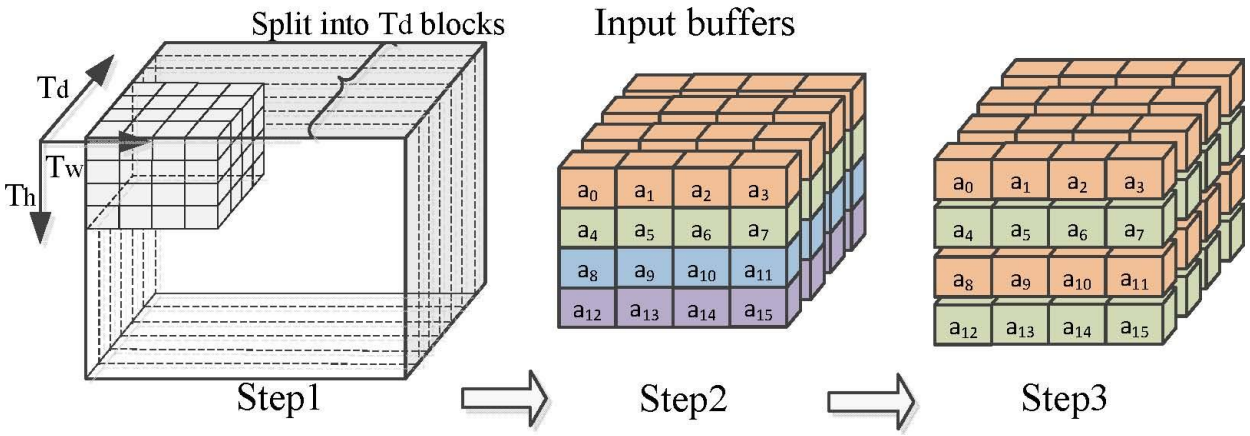
* $T_w = T_c * S - S + K, T_h = T_r * S - S + K, T_d = T_z * S - S + K.$

- Strategy2 for CONV layers with large input feature maps
- Strategy3 for CONV layers with small input feature maps

- Increase both Bw_{on} and Bw_{off} to facilitate fast DRAM-BRAM transfer
- Use multiple memory ports

Template-based Methodology

- Memory access optimization
  - On-chip memory access



- Problems
  - Severe on-chip memory access conflicts
  - It is infeasible to fully split the memory blocks into register files
- Solution
  - Step-by-step optimization strategy

Outline

- 3D CNNs
- Motivations
- Template-based Methodology
- Design Space Exploration
- Evaluations
- Future Work

Design Space Exploration

- Uniform analytical model for 2D/3D CNNs
  - Computational Roof

$$CR = \frac{OPs}{EC_w} = \frac{2 \times M \times N \times Ksize \times m^{dim}}{\lceil \frac{M}{T_o} \rceil \times \lceil \frac{N}{T_i} \rceil \times I}.$$

  - Performance model

$$\begin{cases} T^i_{trans} = \frac{(V_{in} + V_w) \times Data\_Width}{BW_{eff}} \\ T^o_{trans} = \frac{V_{out} \times Data\_Width}{BW_{eff}}, \\ T_{com} = \frac{T_z \times T_r \times T_c \times I}{m^{dim} \times Freq} \\ T_{total} = \frac{Z}{T_z} \times \frac{R}{T_r} \times \frac{C}{T_c} \times (\lceil \frac{M}{T_o} \rceil \times \lceil \frac{N}{T_i} \rceil \times \max\{T_{com}, T^i_{trans}\} + T^o_{trans}) \end{cases}$$

- Definition

$$dim = \begin{cases} 2, \text{ 2D CNN} \\ 3, \text{ 3D CNN} \end{cases}$$

$Z = Tz = 1$ for 2D CNN

Outline

- 3D CNNs
- Motivations
- Template-based methodology
- Design Space Exploration
- Evaluations
- Future Work

Evaluations

- Cross-layer parameters
  - Minimal overall performance degradation
  - Optimal on-chip and off-chip bandwidth
- Resource utilization
  - DSPs are no longer the limiting resource
  - LUTs dominates the resource utilization

Table 4: Uniform cross-layer parameters

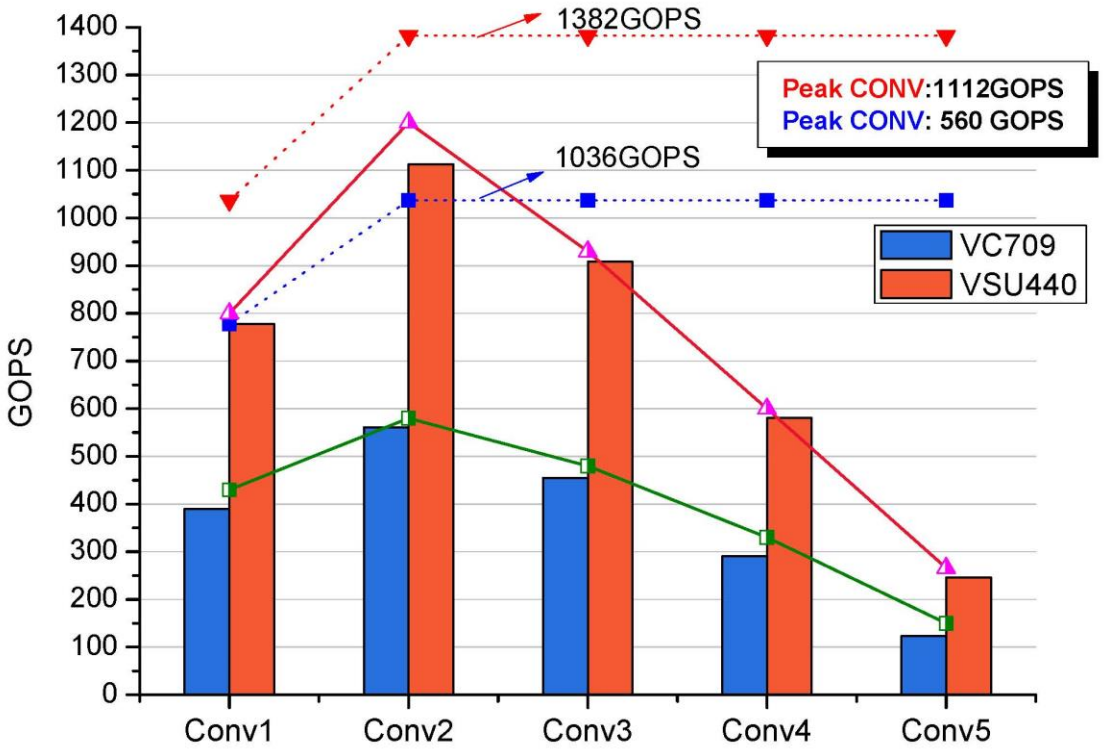
Networks	Winograd	T_i	T_o	Bw_{off}	Bw_{on}	# of Ports
VGG16	$F(2^2, 3^2)$	4	64	256bit	64bit	4
C3D	$F(2^3, 3^3)$	4	32	256bit	64bit	4

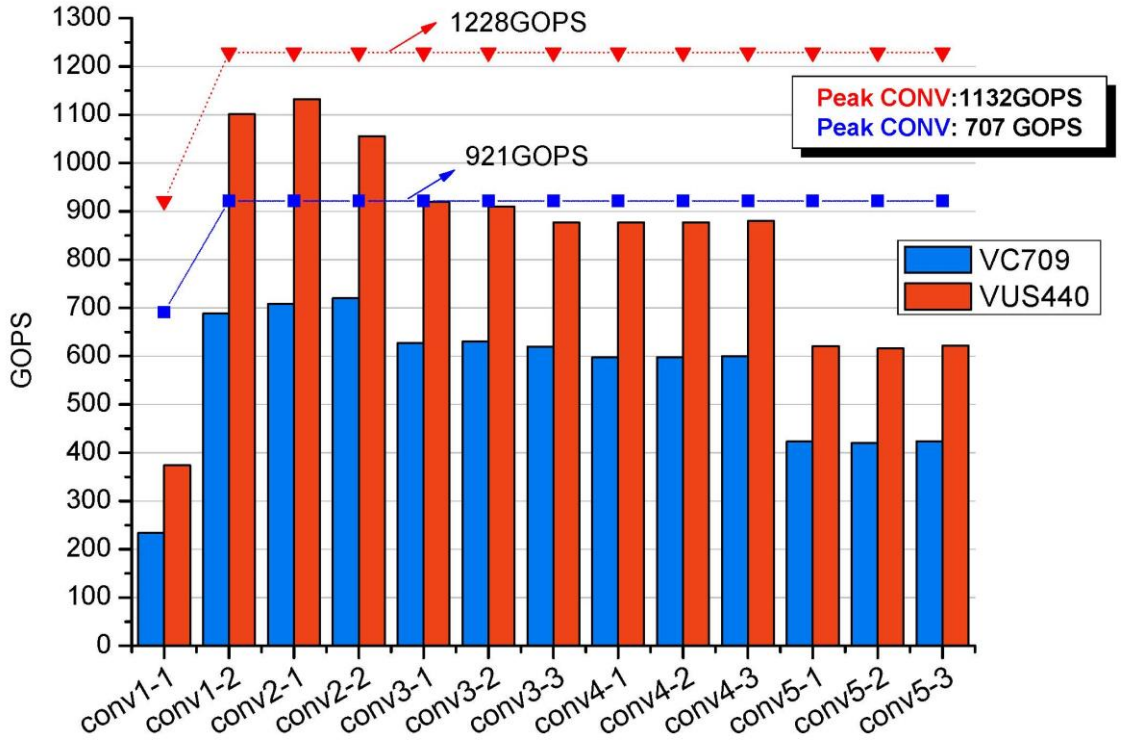
Table 5: FPGA resource utilization

Device		Resource	DSP	BRAM	LUT	FF
VC709		Available	3600	2940	433K	866K
	2D	Used	1376	1232	175K	202K
		Utilization	38%	42%	40%	23%
	3D	Used	1536	1508	242K	286K
		Utilization	42%	52%	56%	33%
VUS440		Available	2880	5040	2532K	5065K
	2D	Used	1376	1232	170K	189K
		Utilization	48%	24%	6.7%	3.7%
	3D	Used	1536	1476	209K	285K
		Utilization	53%	30%	8.3%	5.6%

Evaluation Results

- VGG and C3D





Evaluation Results

- Performance and Energy Efficiency Comparison (2D CNN)

	[5]	[6]	[8]	[9]	[7]	[10]	Our work	
FPGA	Xilinx XC7Z045	Altera Stratix-V	Xilinx Virtex 690t	Arria10 GX1150	Arria10 GX1150	Arria10 GX1150	Xilinx Virtex 690t	Xilinx VCU440
Frequency	150	120	150	150	303	385	150	200
CNN	VGG	VGG	VGG	VGG	AlexNet	VGG	VGG	VGG
Precision	16-bit fixed	8-16 bits fixed	16-bit fixed	8-16 bits fixed	16-bit float	16-bit fixed	16-bit fixed	16-bit fixed
DSP Utilization	780(87%)	727(37%)	2833(78%)	1518(100 %)	1476(97 %)	2756(91%)	1376(38%)	1376(48%)
Throughput (Gops)	137	118	354	645	1382	1790	570	821
DSP Efficiency	0.18	0.16	0.12	0.43	0.98	0.65	0.41	0.60

Evaluation Results

- Performance and Energy Efficiency Comparison (3D CNN)

Platforms	CPU	GPU		FPGA	
Device	E5-2680	K40	GTX1080	VC709	VUS440
Technology	22nm	28nm	16nm	28nm	20nm
Power(W)	115	250	180	25	26
CONV(Gops)	60.3	1206.5	4375.7	474.3	940.6
CNN(Gops)	58.7	1174.0	4101.9	430.7	784.7
Latency(ms)	656.2	32.8	8.8	89.4	49.1
Speedup	1x	20.0x	69.9x	7.3x	13.4x
Gops/W	0.5 (1x)	4.7 (9.2x)	22.8 (44.6x)	17.1 (33.8x)	30.2 (60.3x)

Outline

- 3D CNNs
- Motivations
- Related Works
- Template-based methodology
- Evaluations
- Future work

Future Work

- Evaluations on the Winograd algorithm with different size
  - $F(4^3,3^3)$, $F(6^3,3^3)$ …
- Design uniform templates and architecture for compressed models
  - Sparse CNNs (Pruning, Quantization)
- Other fast algorithms
  - FFT
  - …

Thank You

shenjunzhong@nudt.edu.cn

国防科技大学
National university of defense technology

References

[1] Kamnitsas, Konstantinos, et al. "Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation." *Medical image analysis* 36 (2017): 61-78.

[2] Liu, Zhi, Chenyang Zhang, and Yingli Tian. "3d-based deep convolutional neural network for action recognition with depth sequences." Image and Vision Computing 55 (2016): 93-100.

[3] Karpathy, Andrej, et al. "Large-scale video classification with convolutional neural networks." Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. 2014.

[4] Tran, Du, et al. "Learning spatiotemporal features with 3d convolutional networks." *Computer Vision (ICCV), 2015 IEEE International Conference on*. IEEE, 2015.

[5] Qiu, Jiantao, et al. "Going deeper with embedded fpga platform for convolutional neural network." *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2016.

[6] Suda, Naveen, et al. "Throughput-optimized OpenCL-based FPGA accelerator for large-scale convolutional neural networks." *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2016.

[7] Aydonat, Utku, et al. "An OpenCL™ deep learning accelerator on Arria 10." *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2017.

[9] Ma, Yufei, et al. "Optimizing loop operation and dataflow in FPGA acceleration of deep convolutional neural networks." Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. ACM, 2017.

[10] Zhang, Jialiang, and Jing Li. "Improving the performance of OpenCL-based FPGA accelerator for convolutional neural network." Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. ACM, 2017.