AIgebraic Enhancements for Systolic Arrays

Trevor Pogue

Department of Electrical and Computer Engineering
McMaster University

November 22 2024

Overview

- Backround
- Deep Learning Accelerator System Architecture
- Fast Inner-Product Algorithms and Hardware Architectures
- Karatsuba Matrix Multiplication Algorithm and Hardware Architectures
- Strassen Hardware Architectures
- Conclusion and Future Work

Overview

- **Backround**
- Deep Learning Accelerator System Architecture
- Fast Inner-Product Algorithms and Hardware Architectures
- Karatsuba Matrix Multiplication Algorithm and Hardware Architectures
- Strassen Hardware Architectures
- Conclusion and Future Work

Background

- Why use hardware acceleration for deep learning (DL):
  - Increasing demand in environments like real-time and datacenters alike, requires efficiency
  - Requires billions of operations
  - Breaks down into same parallelizable compute patterns - great fit for hardware acceleration

Background

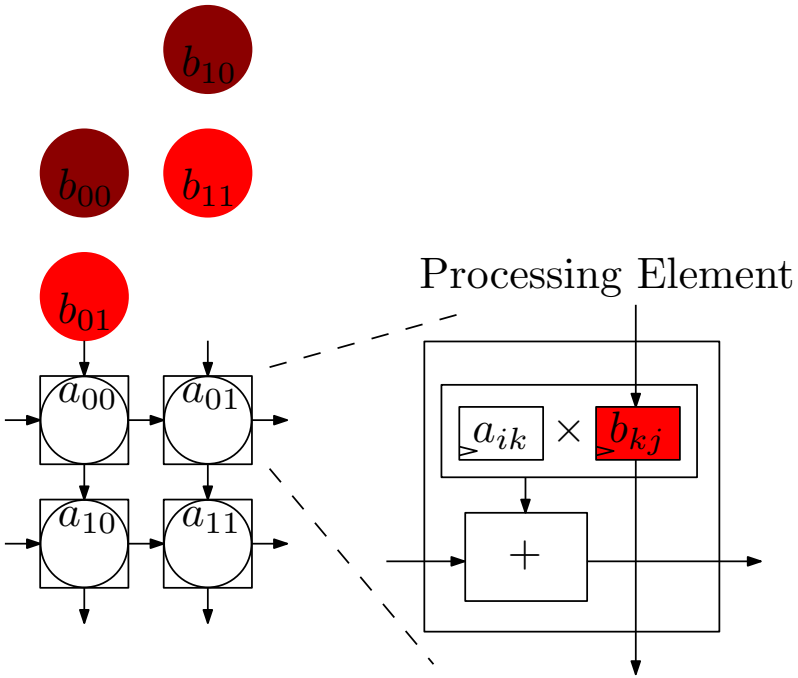
- Why use hardware acceleration for deep learning (DL):
  - Increasing demand in environments like real-time and datacenters alike, requires efficiency
  - Requires billions of operations
  - Breaks down into same parallelizable compute patterns - great fit for hardware acceleration

- Previous approaches for deep learning hardware acceleration:
  - Quantization
  - Sparse/pruned NNs
  - Hardware architecture design automation
  - Hardware-oriented NN model design automation

Background

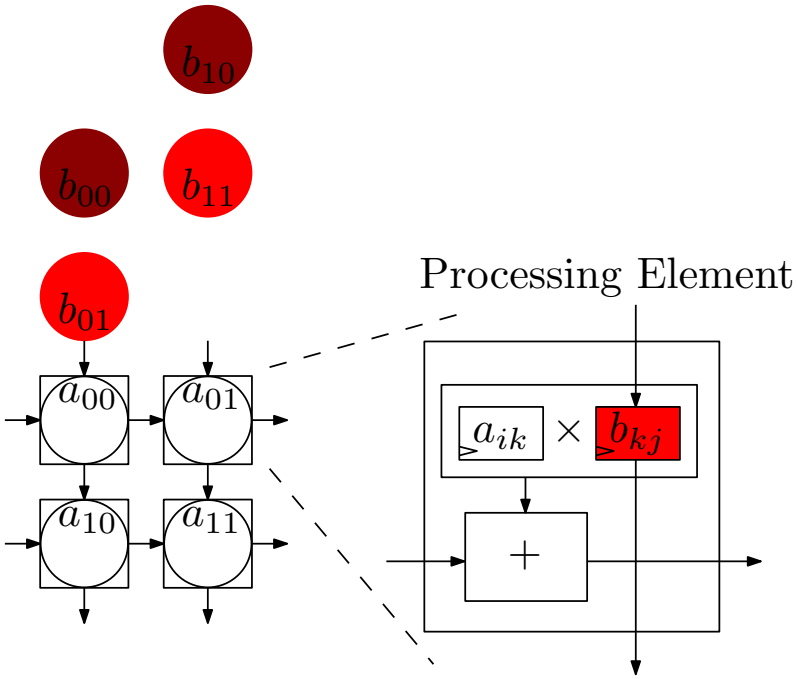
- Why use hardware acceleration for deep learning (DL):
  - Increasing demand in environments like real-time and datacenters alike, requires efficiency
  - Requires billions of operations
  - Breaks down into same parallelizable compute patterns - great fit for hardware acceleration
- Previous approaches for deep learning hardware acceleration:
  - Quantization
  - Sparse/pruned NNs
  - Hardware architecture design automation
  - Hardware-oriented NN model design automation
- Chosen under-explored approach: Advancements and application of fast matrix multiplication in custom hardware designs
  - Expensive portion of most neural networks (NN) decomposes to GEMM
  - NN's algebra is performed using reduced complexity GEMM
  - Less explored route for continuing progress

- 2D array of multiply-accumulate (MAC) units



[1] Norman P. Jouppi et al. "In-Datacenter Performance Analysis of a Tensor Processing Unit". In: *SIGARCH Comput. Archit. News* 45.2 (June 2017), 1–12

- 2D array of multiply-accumulate (MAC) units

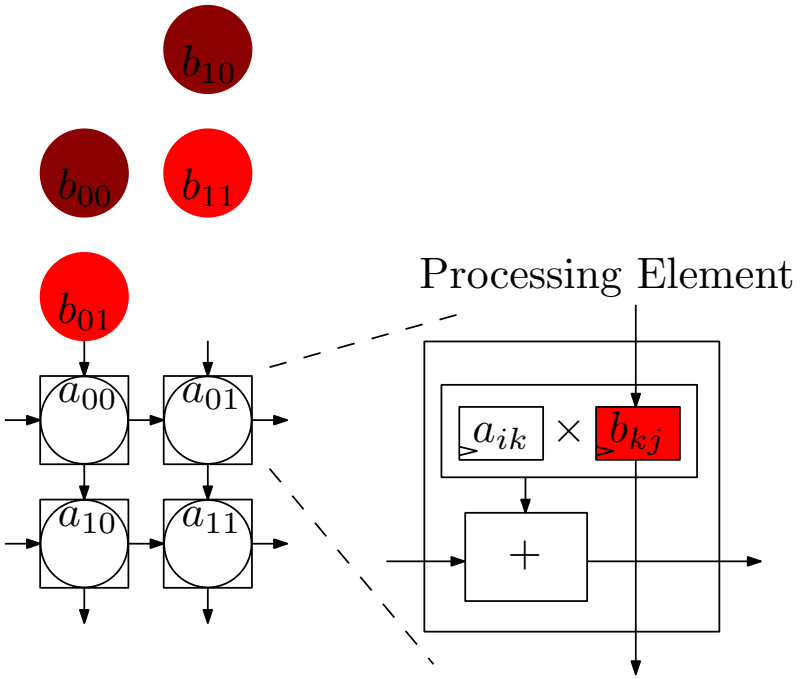


Benefits

- Reduced reads from memory
  - Reduced memory bandwidth and power consumption
- Local and regular interconnections between processors
  - Increases max clock frequency

[1] Norman P. Jouppi et al. "In-Datacenter Performance Analysis of a Tensor Processing Unit". In: *SIGARCH Comput. Archit. News* 45.2 (June 2017), 1–12

- 2D array of multiply-accumulate (MAC) units



Benefits

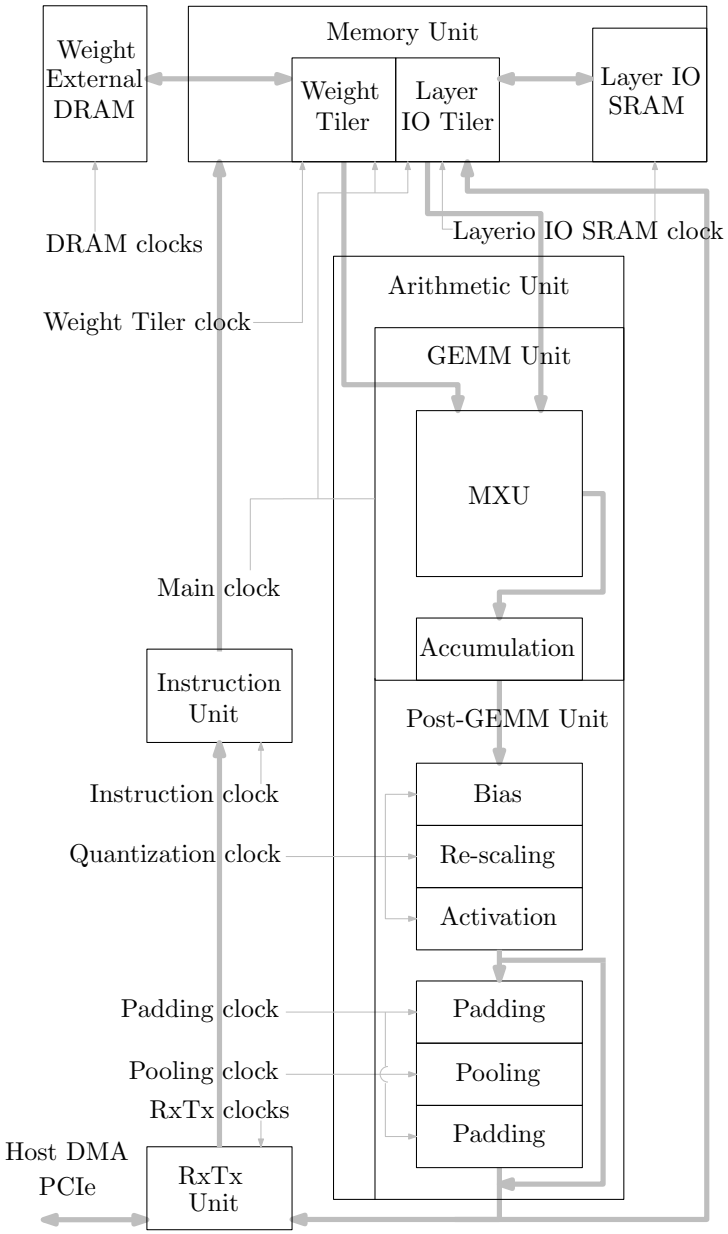
- Reduced reads from memory
  - Reduced memory bandwidth and power consumption
- Local and regular interconnections between processors
  - Increases max clock frequency
- Has been implemented commercially in Google's Tensor Processing Unit (TPU) [1]

[1] Norman P. Jouppi et al. "In-Datacenter Performance Analysis of a Tensor Processing Unit". In: *SIGARCH Comput. Archit. News* 45.2 (June 2017), 1–12

Overview

- Backround
- **Deep Learning Accelerator System Architecture**
- Fast Inner-Product Algorithms and Hardware Architectures
- Karatsuba Matrix Multiplication Algorithm and Hardware Architectures
- Strassen Hardware Architectures
- Conclusion and Future Work

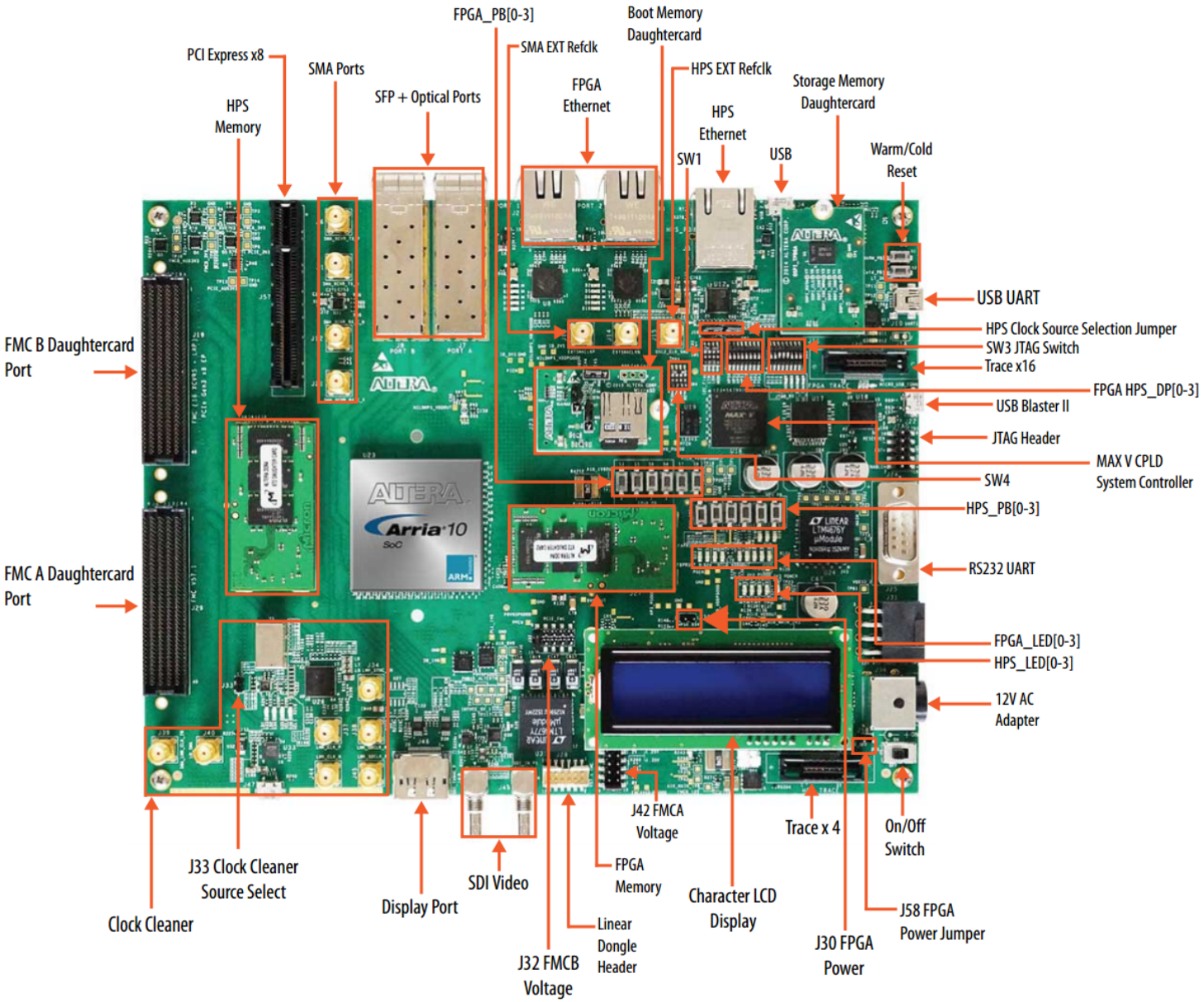
Implemented DL Accelerator System Design Overview



System overview

- Matrix Multiply Unit (MXU) - Systolic Array
- Post-GEMM Unit - NN-specific operations
- Memory Unit - Memory access control, On-chip memory
- Weight DRAM (external memory)
- RxTx Unit - PCIe interface to host
- Instruction Unit

System Design - Used FPGA Platform



https://rocketboards.org/foswiki/Documentation/Arria10SoCGSRD

Overview

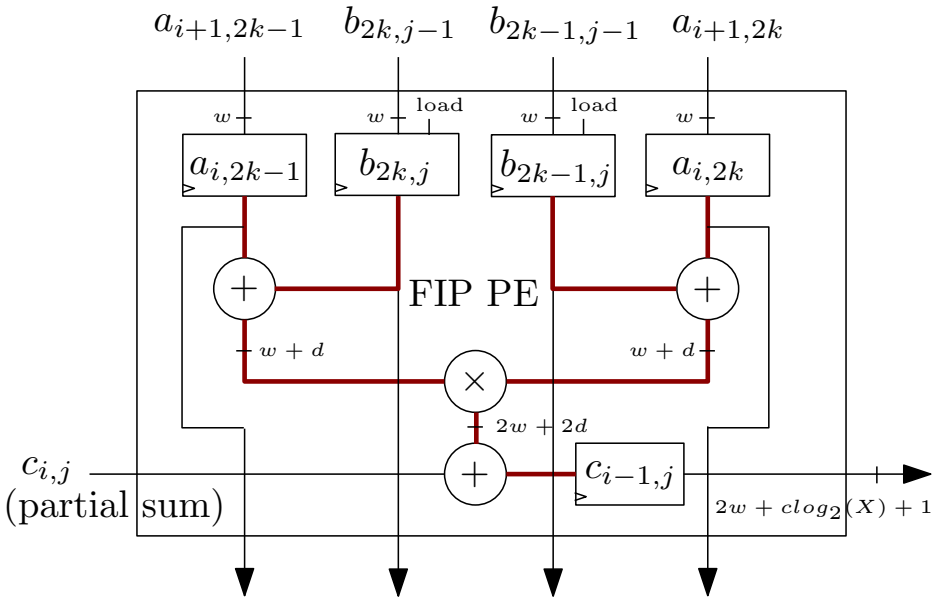
- Backround
- Deep Learning Accelerator System Architecture
- **Fast Inner-Product Algorithms and Hardware Architectures**
- Karatsuba Matrix Multiplication Algorithm and Hardware Architectures
- Strassen Hardware Architectures
- Conclusion and Future Work

Winograd's Fast Inner Product (FIP) [2]:

$$c_{i,j} = \sum_{k=0}^{n/2-1} (a_{i,2k+1} + b_{2k,j})(a_{i,2k} + b_{2k+1,j}) - \alpha_i - \beta_j$$

$$\alpha_i = \sum_{j=0}^{n/2-1} a_{i,2j} \cdot a_{i,2j+1}$$

$$\beta_j = \sum_{i=0}^{n/2-1} b_{2i,j} \cdot b_{2i+1,j}$$



[2] S. Winograd. "A New Algorithm for Inner Product". In: *IEEE Trans. Comput.* C-17.7 (1968), pp. 693–694

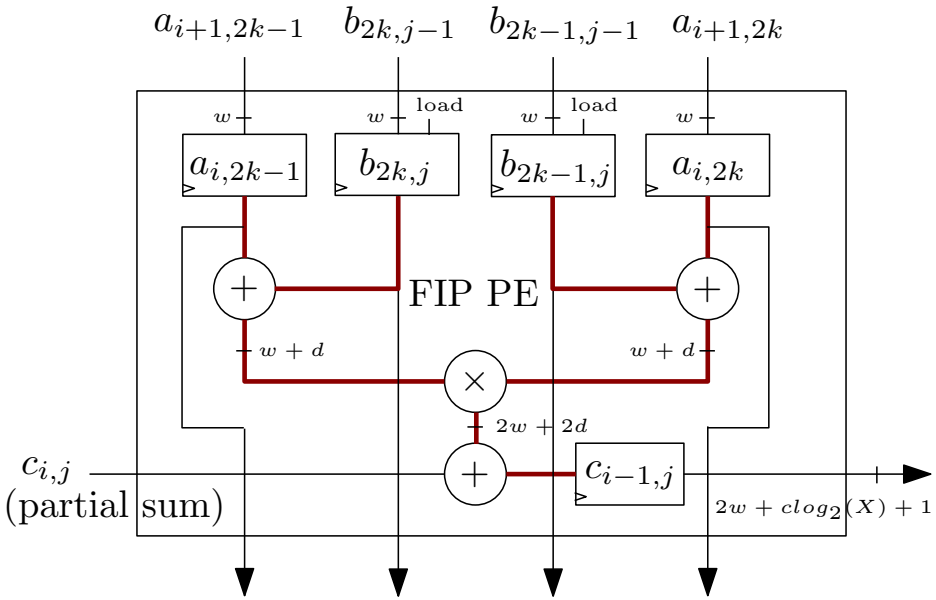
[3] Trevor E. Pogue and Nicola Nicolici. "Fast Inner-Product Algorithms and Architectures for Deep Neural Network Accelerators". In: *IEEE Trans. Comput.* 73.2 (2024), pp. 495–509

Winograd's Fast Inner Product (FIP) [2]:

$$c_{i,j} = \sum_{k=0}^{n/2-1} (a_{i,2k+1} + b_{2k,j})(a_{i,2k} + b_{2k+1,j}) - \alpha_i - \beta_j$$

$$\alpha_i = \sum_{j=0}^{n/2-1} a_{i,2j} \cdot a_{i,2j+1}$$

$$\beta_j = \sum_{i=0}^{n/2-1} b_{2i,j} \cdot b_{2i+1,j}$$



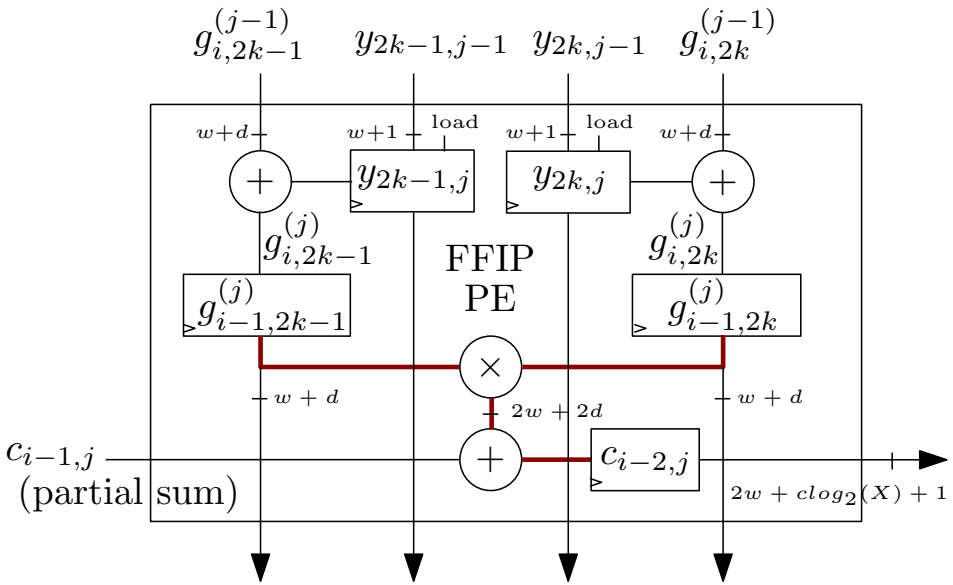
Proposed Free-pipeline FIP (FFIP) [3]:

$$g^{(j)}_{i,2k-1} = a_{i,2k} + y_{2k-1,j} \quad \text{for } j = 1$$

$$g^{(j)}_{i,2k} = a_{i,2k-1} + y_{2k,j} \quad \text{for } j = 1$$

$$g^{(j)}_{i,k} = g^{(j-1)}_{i,k} + y_{k,j} \quad \text{for } j > 1$$

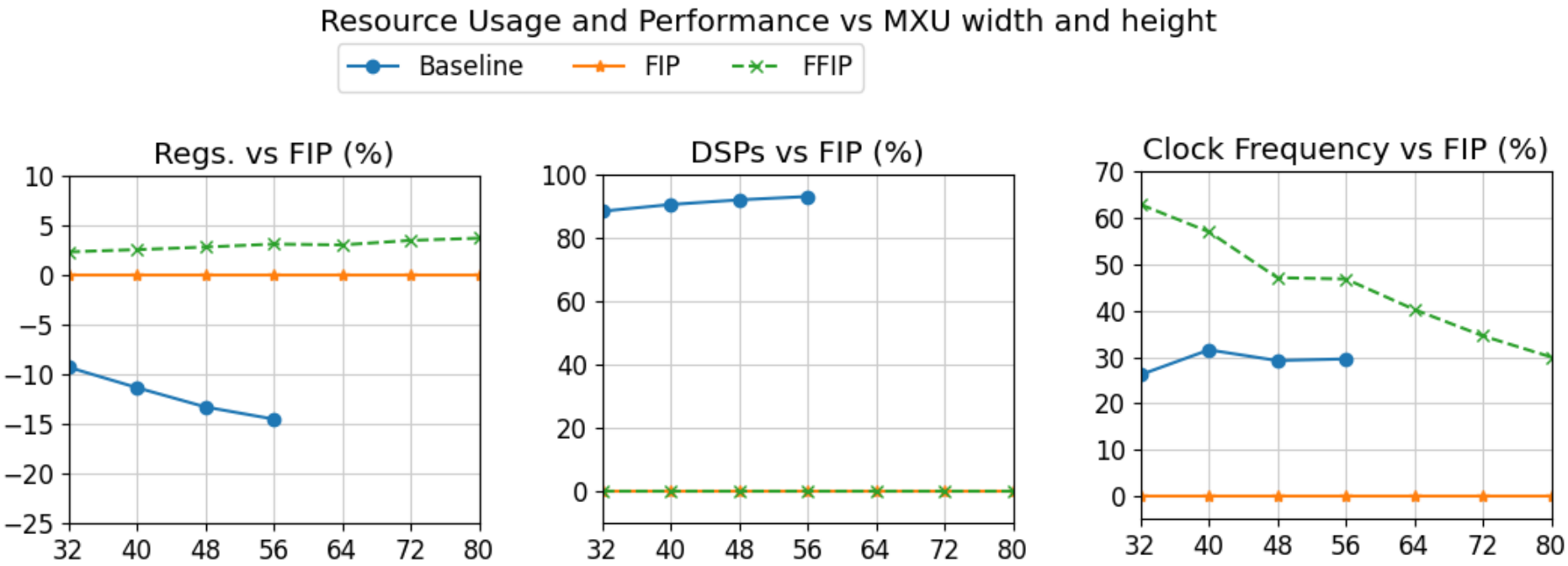
$$y_{i,j} = \begin{cases} b_{ij} & \text{for } j = 1 \\ b_{ij} - b_{i,j-1} & \text{for } j > 1\,. \end{cases}$$



[2] S. Winograd. "A New Algorithm for Inner Product". In: *IEEE Trans. Comput.* C-17.7 (1968), pp. 693–694

[3] Trevor E. Pogue and Nicola Nicolici. "Fast Inner-Product Algorithms and Architectures for Deep Neural Network Accelerators". In: *IEEE Trans. Comput.* 73.2 (2024), pp. 495–509

FFIP Results



- *LUT/ALM resources share a similar curve as registers
- *Memory resources are equivalent for all designs

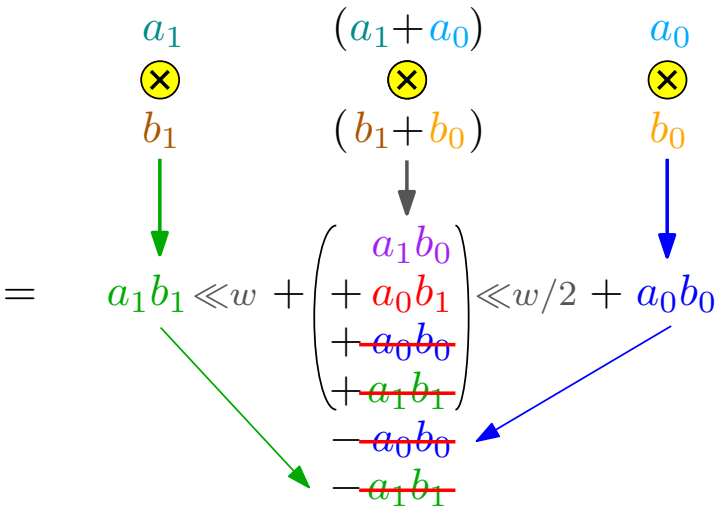
Overview

- Backround
- Deep Learning Accelerator System Architecture
- Fast Inner-Product Algorithms and Hardware Architectures
- **Karatsuba Matrix Multiplication Algorithm and Hardware Architectures**
- Strassen Hardware Architectures
- Conclusion and Future Work

Proposed Karatsuba Matrix Multiplication (KMM)

2-Digit Karatsuba Scalar Multiplication (KSM_2)

$$
\begin{aligned}
& a \times b \\
&= (a_1 \ll w/2 + a_0) \times (b_1 \ll w/2 + b_0)
\end{aligned}
$$



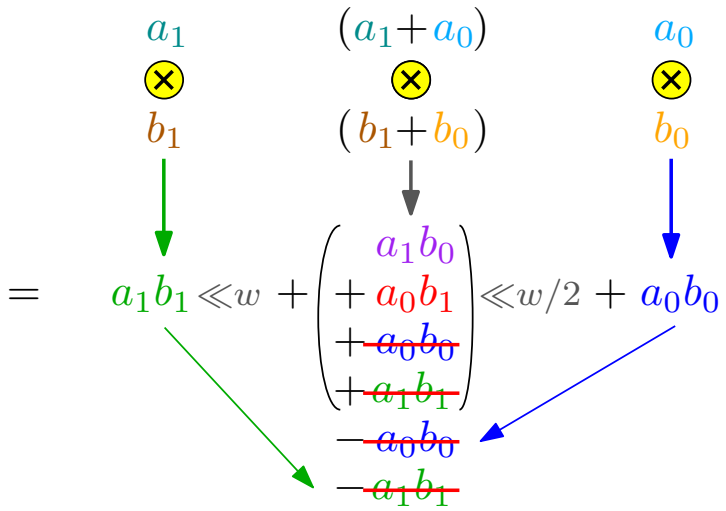
- [4] Requires 3 single-digit mults instead of 4
- But requires 3 extra additions, increasing overall #operation

[4] Anatolii Alekseevich Karatsuba and Yu P Ofman. "Multiplication of many-digital numbers by automatic computers". In: *Proc. Doklady Akademii Nauk*. Vol. 145. 2. Russian Academy of Sciences. 1962, pp. 293–294

Proposed Karatsuba Matrix Multiplication (KMM)

2-Digit Karatsuba Scalar Multiplication (KSM_2)

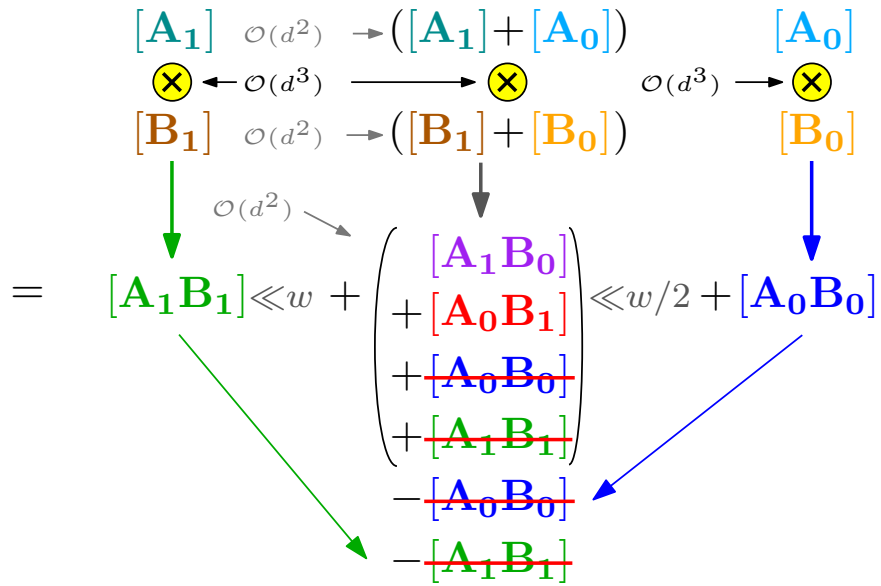
$$\begin{aligned} & a \times b \\ &= (a_1 \ll w/2 + a_0) \times (b_1 \ll w/2 + b_0) \end{aligned}$$



- [4] Requires 3 single-digit mults instead of 4
- But requires 3 extra additions, increasing overall #operation

2-Digit Karatsuba Matrix Multiplication (KMM_2)

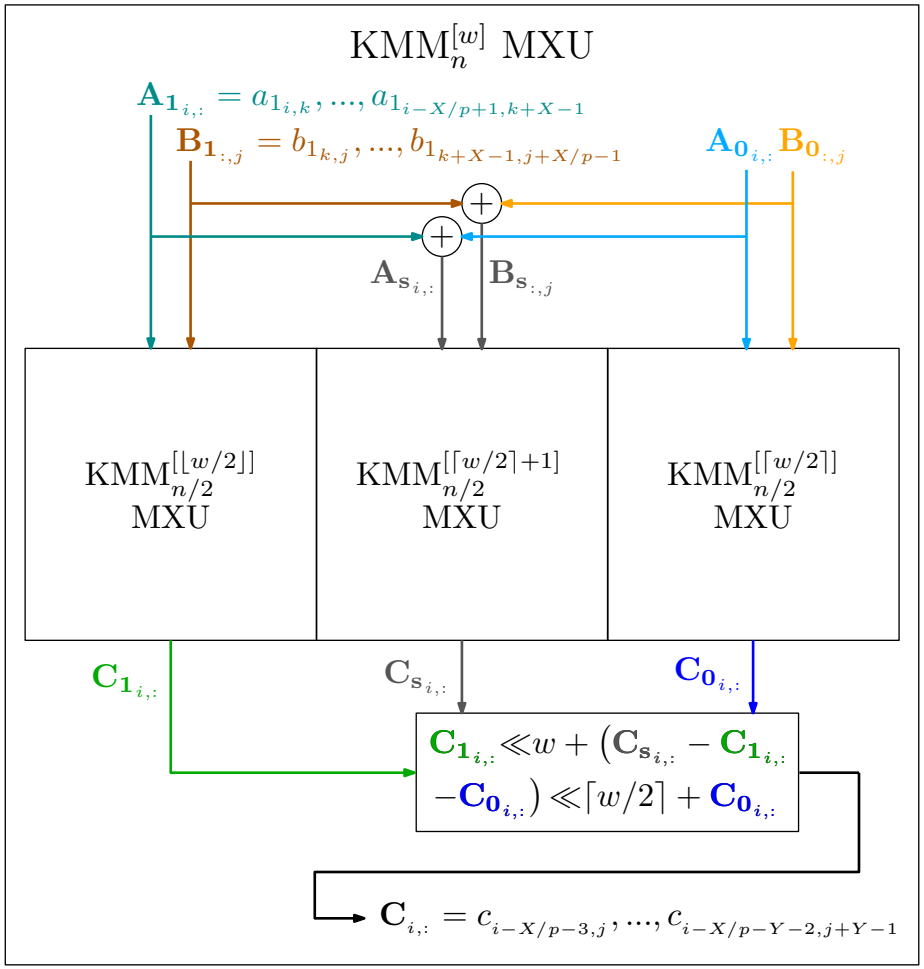
$$\begin{aligned} & [\mathbf{A}] \times [\mathbf{B}] \\ &= ([\mathbf{A_1}] \ll w/2 + [\mathbf{A_0}]) \times ([\mathbf{B_1}] \ll w/2 + [\mathbf{B_0}]) \end{aligned}$$



- d = matrix height/width
- Increase in number of additions with complexity $\mathcal{O}(d^2)$ is now insignificant relative to the reduction of 3 instead of 4 single-digit MM of complexity $\mathcal{O}(d^3)$

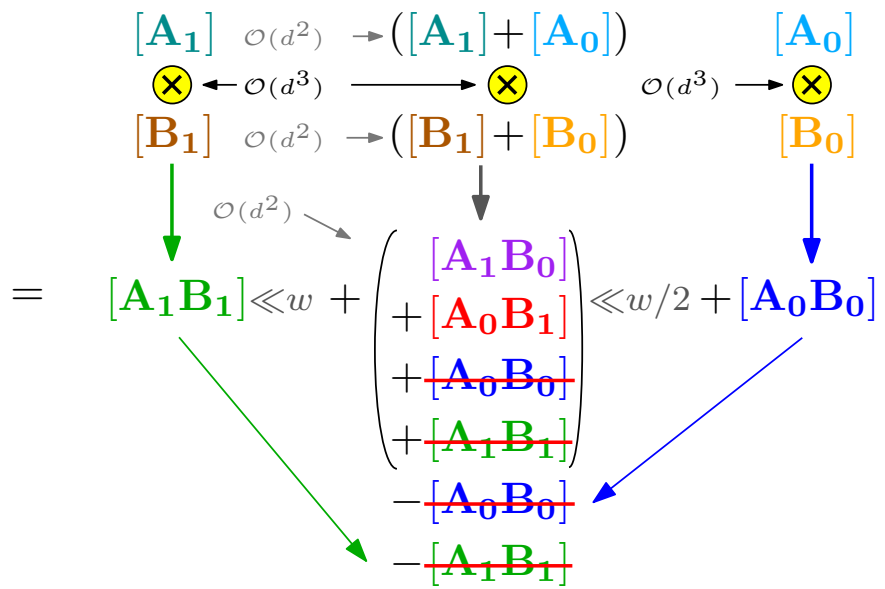
[4] Anatolii Alekseevich Karatsuba and Yu P Ofman. "Multiplication of many-digital numbers by automatic computers". In: *Proc. Doklady Akademii Nauk*. Vol. 145. 2. Russian Academy of Sciences. 1962, pp. 293–294

Proposed KMM Hardware Architecture



2-Digit Karatsuba Matrix Multiplication (KMM_2)

$$[\mathbf{A}] \times [\mathbf{B}]$$
$$= ([\mathbf{A_1}] \ll w/2 + [\mathbf{A_0}]) \times ([\mathbf{B_1}] \ll w/2 + [\mathbf{B_0}])$$



- d = matrix height/width
- Increase in number of additions with complexity $\mathcal{O}(d^2)$ is now insignificant relative to the reduction of 3 instead of 4 single-digit MM of complexity $\mathcal{O}(d^3)$

KMM Results

$$\text{Area}(\text{ADD}^{[w]}) = w \text{ AU}$$
$$\text{Area}(\text{FF}^{[w]}) = 0.7\ w \text{ AU}$$
$$\text{Area}(\text{MULT}^{[w]}) = w^2 \text{ AU}$$

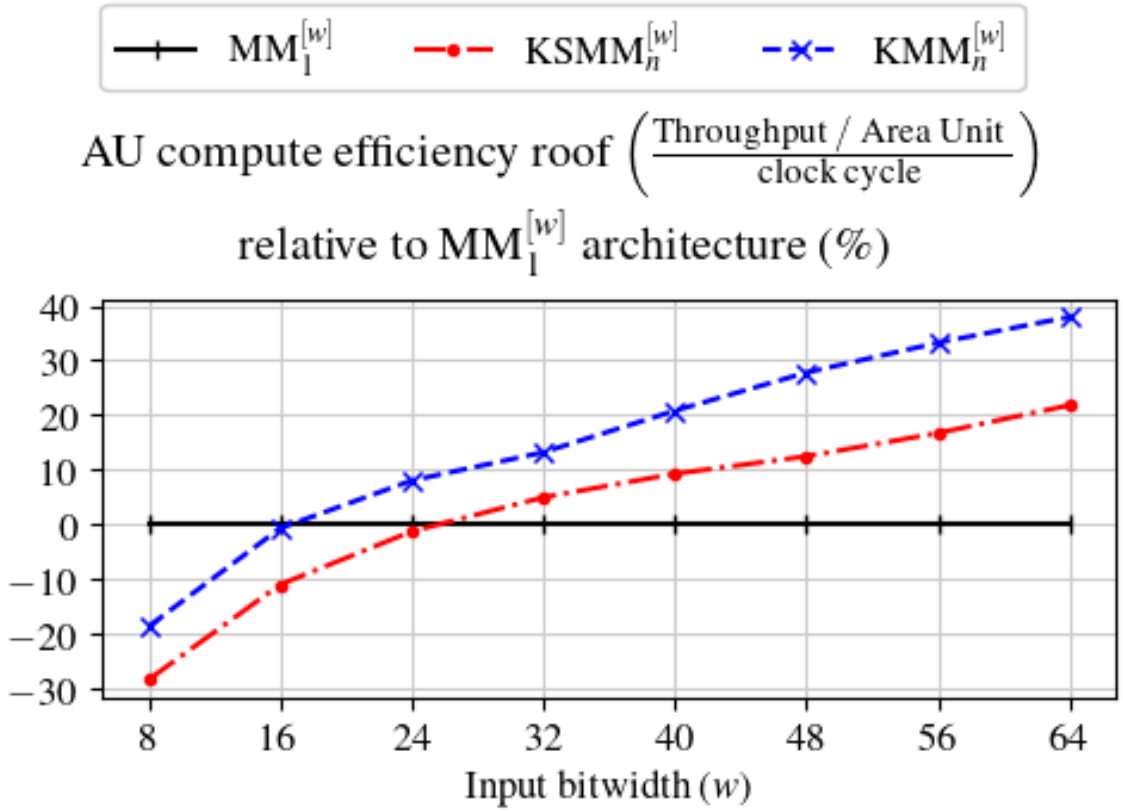


Figure (1) Maximum achievable AU compute efficiencies for the fixed-precision MM_1, KSMM_n, and KMM_n architectures.

Overview

- Backround
- Deep Learning Accelerator System Architecture
- Fast Inner-Product Algorithms and Hardware Architectures
- Karatsuba Matrix Multiplication Algorithm and Hardware Architectures
- **Strassen Hardware Architectures**
- Conclusion and Future Work

Strassen Matrix Multiplication (SMM) [5]

Traditional 4-tile MM requires 8 tile MMs:

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{21} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{bmatrix}$$

Strassen [5] requires 7 tile MMs:

$$\begin{aligned} T_1 &= A_{11} + A_{22} & S_1 &= B_{11} + B_{22} & Q_1 &= T_1 \cdot S_1 \\ T_2 &= A_{21} + A_{22} & S_2 &= B_{11} & Q_2 &= T_2 \cdot S_2 \\ T_3 &= A_{11} & S_3 &= B_{12} - B_{22} & Q_3 &= T_3 \cdot S_3 \\ T_4 &= A_{22} & S_4 &= B_{21} - B_{11} & Q_4 &= T_4 \cdot S_4 \\ T_5 &= A_{11} + A_{12} & S_5 &= B_{22} & Q_5 &= T_5 \cdot S_5 \\ T_6 &= A_{21} - A_{11} & S_6 &= B_{11} + B_{12} & Q_6 &= T_6 \cdot S_6 \\ T_7 &= A_{12} - A_{22} & S_7 &= B_{21} + B_{22} & Q_7 &= T_7 \cdot S_7 \end{aligned}$$

$$\begin{aligned} C_{11} &= Q_1 + Q_4 - Q_5 + Q_7 \\ C_{12} &= Q_3 + Q_5 \\ C_{21} &= Q_2 + Q_4 \\ C_{22} &= Q_1 - Q_2 + Q_3 + Q_6 \end{aligned}$$

[5] Volker Strassen. "Gaussian elimination is not optimal". In: *Numer. Math.* 13.4 (1969), pp. 354–356

Proposed SMM Hardware Architecture

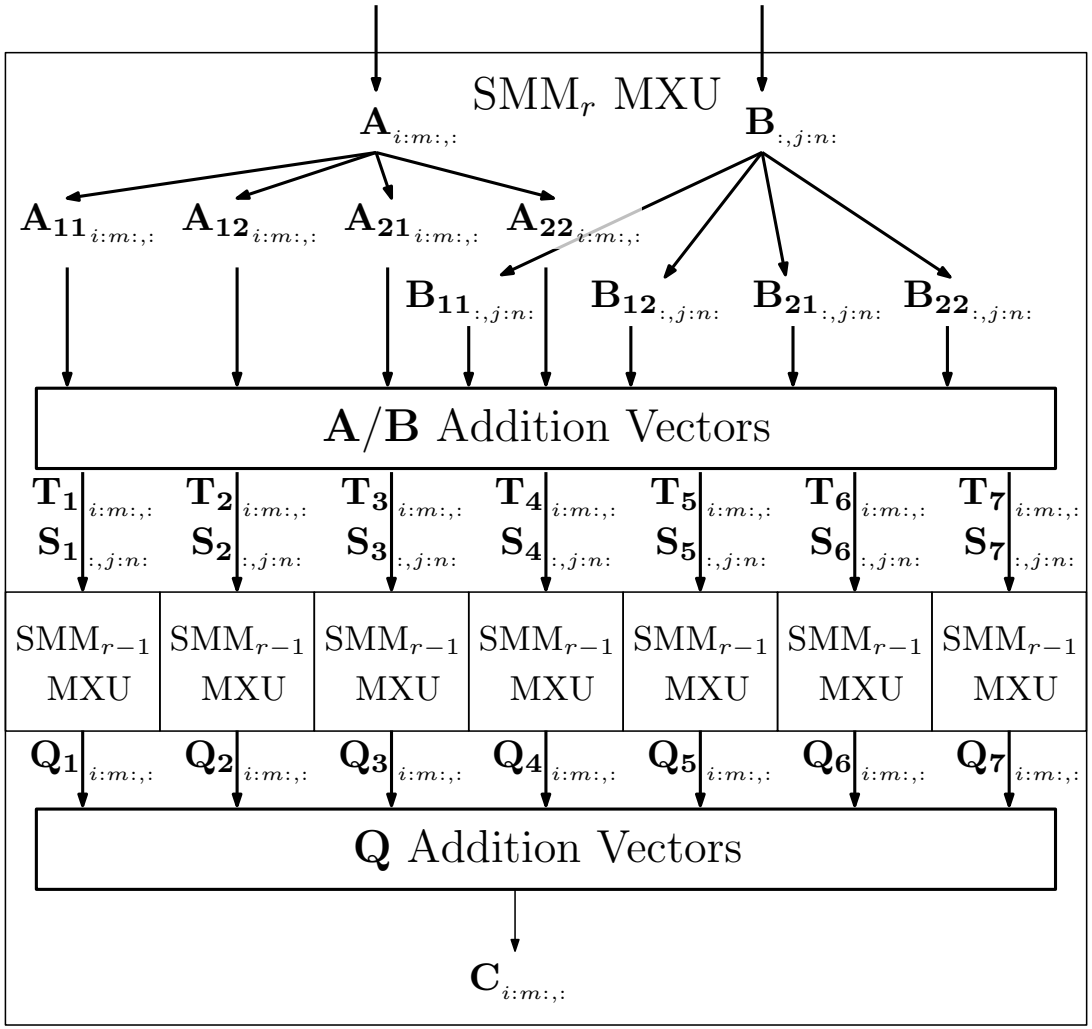


Figure (2) SMM$_r$ systolic-array architecture for implementing r levels of Strassen recursion in hardware.

- Unlike CPU/GPU implementations, extra additions & data movements are performed in parallel with the MMs
- Eliminates extra execution time needed for these steps

FFIP+SMM Results

Table (1) FFIP+SMM$_r$ architectures in a DNN accelerator compared with prior state-of-the-art accelerators.

	TNNLS '22 [6]		TCAD '22 [7]		Entropy '22 [8]		TCAD '24 [9]		Proposed FFIP+SMM$_1$ 32×32		
FPGA	Arria 10 GX 1150		Arria 10 GX 1150		Arria 10 GX 1150		Stratix 10 GX650		Arria 10 GX 1150		
ALMs	304K		304K		303K		152K		216K		
Registers	889K		890K		-		567K		627K		
Memories	2334		2334		1953		2056		2713		
DSPs	1473		1473		1503		1024		1518		
Frequency (MHz)	200		220		172		200		313		
Input bitwidth (fixed-point)	8	8	8	8	8	8	8	8	8	8	8
Model	ResNet-50	VGG 16	Bayes ResNet18	Bayes VGG11	RCNN ResNet50	RCNN VGG16	ResNet-50	ResNet-152	ResNet-50	ResNet-101	ResNet-152
Throughput (GOPS)	1519	1295	1590	534	719	865	800	794	4006	4397	4568
$\frac{\text{mults/multiplier}}{\text{clock cycle}}$ [1]	0.645	0.550	0.639	0.206	0.696	0.837	0.977	0.969	1.674	1.837	1.908

[1] Multiplier compute efficiency, measures how efficiently multipliers are utilized. It can surpass 1 in the proposed designs due to the algebraic enhancements.

[6] Shuanglong Liu et al. "Toward full-stack acceleration of deep convolutional neural networks on FPGAs". In: *IEEE Trans. Neural Netw. Learn. Syst.* 33.8 (2022), pp. 3974–3987

[7] Hongxiang Fan et al. "FPGA-based Acceleration for Bayesian Convolutional Neural Networks". In: *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* 41.12 (2022), pp. 5343–5356

[8] Jianjing An et al. "An OpenCL-Based FPGA Accelerator for Faster R-CNN". In: *Entropy* 24.10 (2022), p. 1346

[9] Kui Dai et al. "DCP-CNN: Efficient Acceleration of CNNs With Dynamic Computing Parallelism on FPGA". In: *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* (2024)

Overview

- Backround
- Deep Learning Accelerator System Architecture
- Fast Inner-Product Algorithms and Hardware Architectures
- Karatsuba Matrix Multiplication Algorithm and Hardware Architectures
- Strassen Hardware Architectures
- **Conclusion and Future Work**

Conclusion and Future Work

Future Work

- Floating-point algorithms and architectures
- Non-systolic-array architectures
- Toom-Cook Matrix Multiplication
- Transformer acceleration

Conclusion and Future Work

Future Work

- Floating-point algorithms and architectures
- Non-systolic-array architectures
- Toom-Cook Matrix Multiplication
- Transformer acceleration

Conclusion

- Contributes to the field of DL and MM acceleration through under-explored direction
- Proposes new efficient MM algorithms and/or their systolic-array hardware architectures
- Increases performance-per-area capabilities of hardware accelerators

Conclusion and Future Work

Future Work

- Floating-point algorithms and architectures
- Non-systolic-array architectures
- Toom-Cook Matrix Multiplication
- Transformer acceleration

Conclusion

- Contributes to the field of DL and MM acceleration through under-explored direction
- Proposes new efficient MM algorithms and/or their systolic-array hardware architectures
- Increases performance-per-area capabilities of hardware accelerators

Thank You! Questions?