

Introduction

- Many solutions have been deployed for accelerating deep neural networks in hardware, ranging from ASICs to GPUs to FPGAs. FPGA based designs provide massive parallelism, while being flexible and easily configurable, and being fast and power efficient.
- Among all the operations executed by different types of state-of-the-art neural networks, about 80-90% of the operations are matrix multiplications (also called GEMM - General Matrix Multiply).
- Designing a matrix multiplier utilizing the commonly available building blocks on current FPGAs (CLBs and DSP slices) leads to a slow and inefficient implementation. We propose an architectural modification that adds hard matrix multiplier blocks to the FPGA in order to alleviate this problem.
- Although our proposed changes will make the FPGA slightly less flexible, the benefits obtained are large enough to justify making them, especially with the abundance of AI/ML usecases.

Methodology

Tools used

- VTR 7.0 for FPGA architecture exploration [3]
- Synopsys VCS for Verilog simulations
- Synopsys Design Compiler for ASIC synthesis

Experimental parameters

- Data precision used for experiments was 16-bit fixed point
- Technology node used was 40nm
- The designs mapped to the FPGA were matrix multipliers ranging from 4x4x4 to 64x64x64

Baseline FPGA

- Approximation of Altera Stratix IV FPGA architecture available with VTR
- 40nm island style FPGA with L=4, Fc_in=0.15, Fc_out=0.1, Wilton switches with Fs=3
- CLBs: N=10 fracturable 6-LUT, can operate as two 5-LUTs with shared inputs
- BRAMs: 32 Kbits that can operate in multiple geometries in both single and dual port modes
- DSP slices: Custom designed. Supports MAC mode, multiplier mode and adder mode

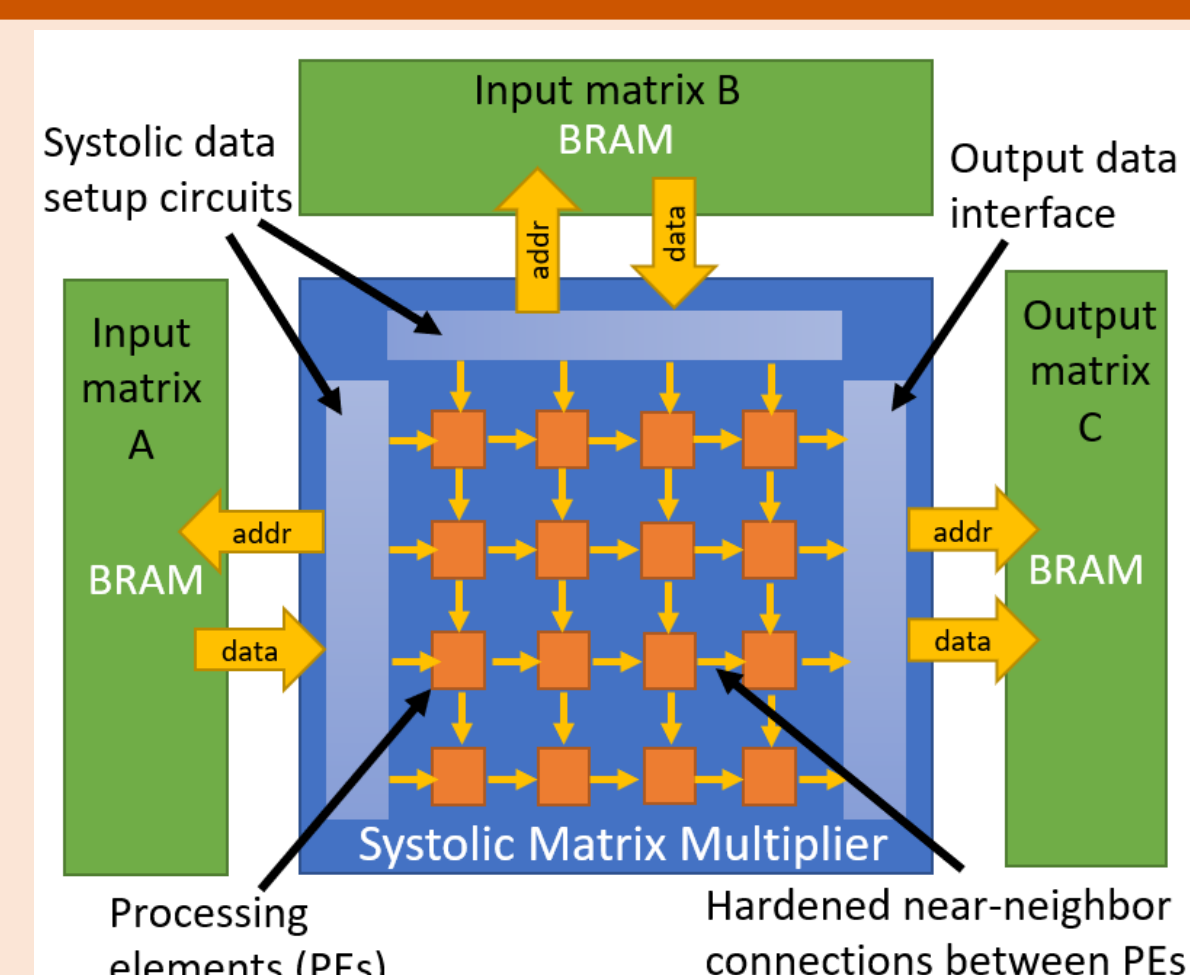
FPGA with matmuls

- Same as above, but DSP slices replaced with matmuls (timing and area obtained from ASIC synthesis and P&R overhead added [5])
- Matmuls use square aspect ratios, contain switch boxes and have pins evenly distributed along perimeter [2] [6]
- Neighboring matmuls (top, bottom, left, right) have direct programmable interconnect.

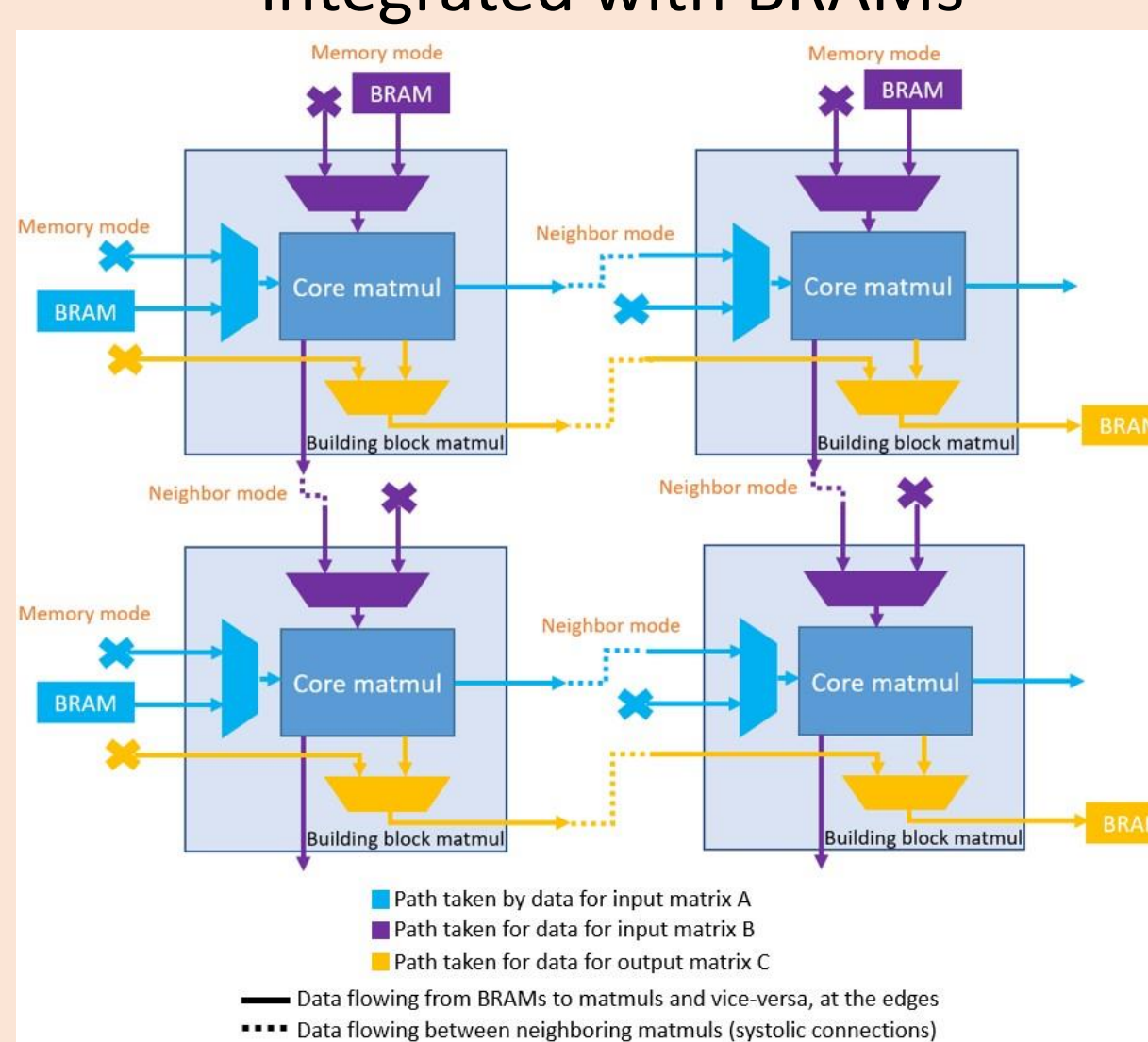
References

- [1] H. T. Kung. 1982. Why Systolic Architectures? Computer.
- [2] C. W. Yu. et al. 2008. The Coarse-Grained / Fine-Grained Logic Interface in FPGAs with Embedded Floating-Point Arithmetic Units. Southern Conference on Programmable Logic.
- [3] Jason Luu et. al. 2014. VTR 7.0: Next Generation Architecture and CAD System for FPGAs. ACM Trans. Reconfigurable Technol. Syst.
- [4] mlperf.org. 2018. MLPerf. <http://www.mlperf.org>
- [5] Chun Ho et. al. 2007. Domain-Specific Hybrid FPGA: Architecture and Floating Point Applications. FPL.
- [6] ChiWai Yu et. al. 2010. Routing optimization for hybrid FPGAs. FPT.

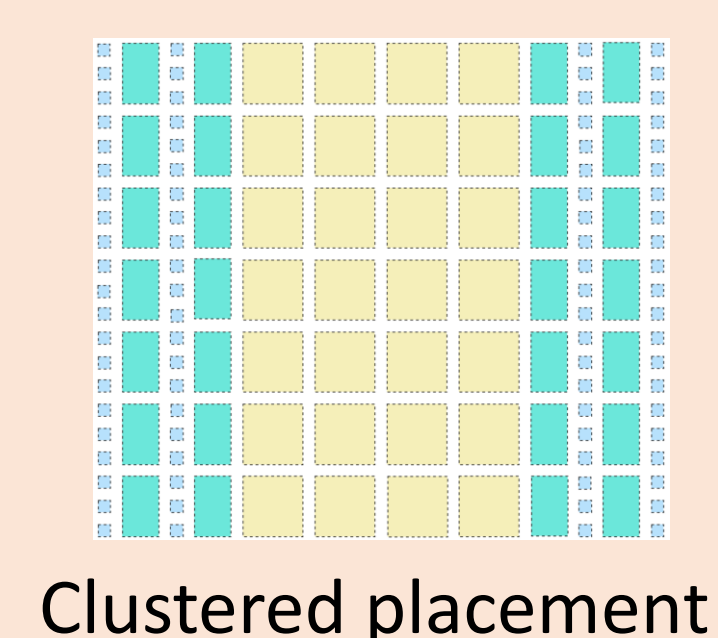
Proposed Architecture



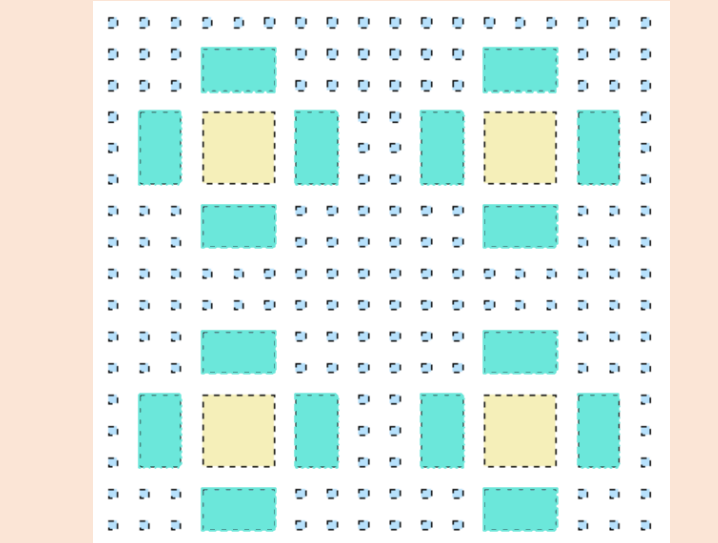
A 4x4x4 systolic matrix multiplier integrated with BRAMs



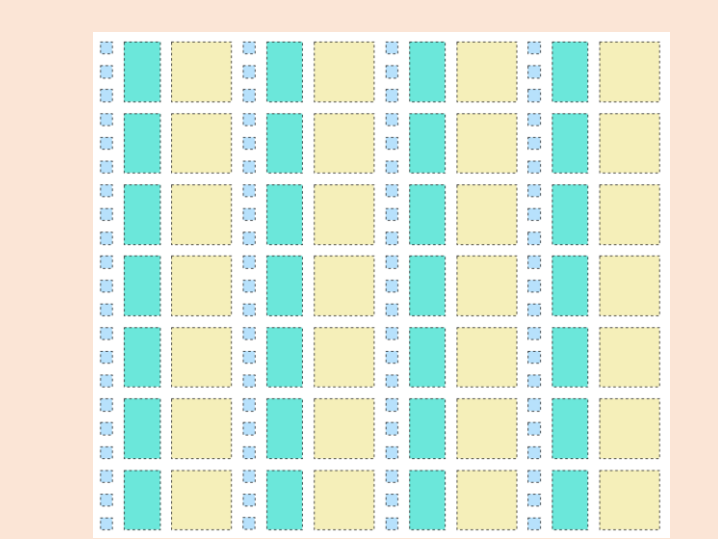
An 8x8x8 matrix multiplier designed by systolically composing two 4x4x4 matmuls



Clustered placement



Surround placement



Columnar placement

We add hard matrix multipliers (“Matmuls”) as building blocks to FPGAs. Implementing a function using specific purpose blocks means better area, speed and power consumption compared to using general purpose blocks.

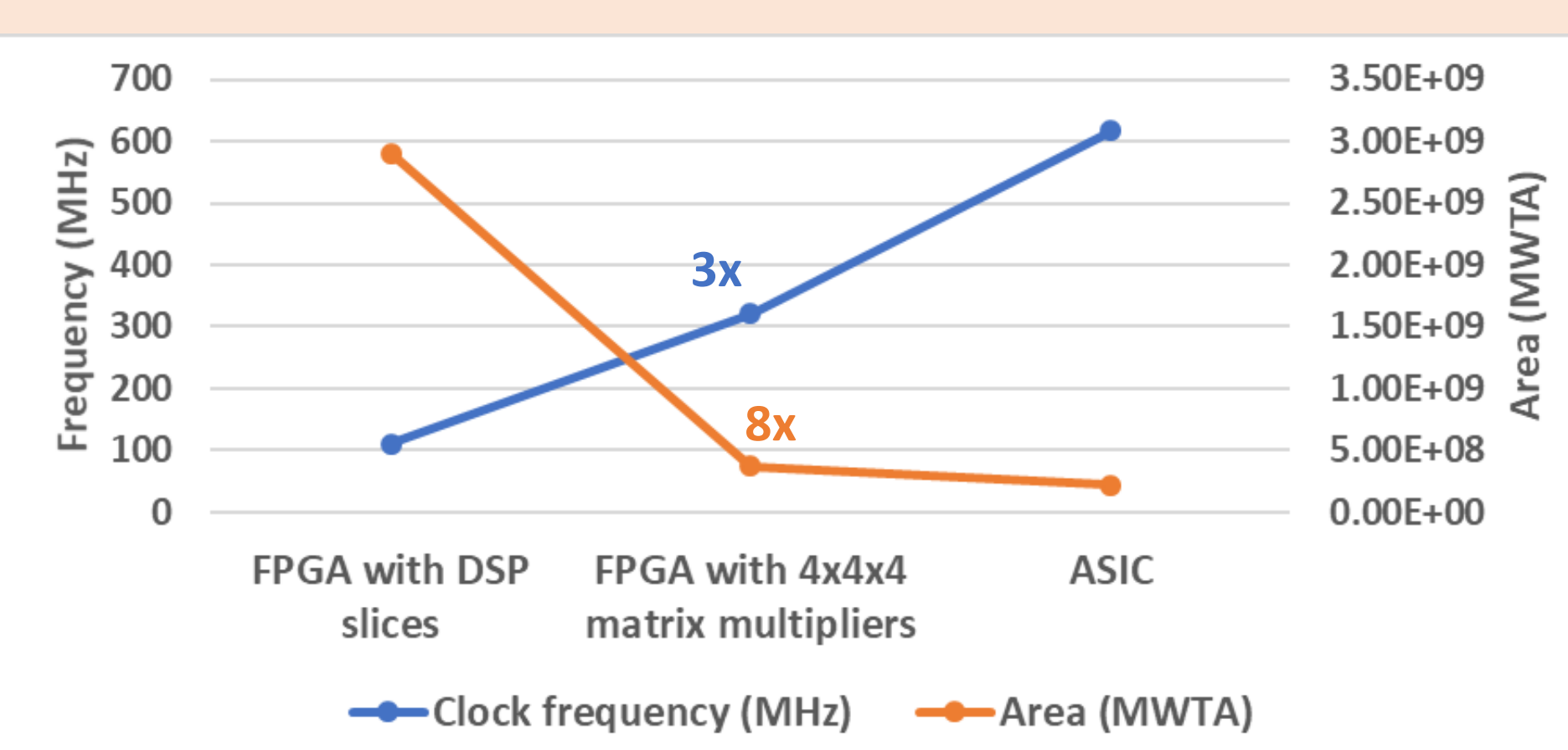
We use systolic architecture because of symmetry, easy composition, near neighbor connections, high compute efficiency. We adapt design R1[1], where inputs move (left to right and top to bottom) and results “stay” in PEs.

We explore many building block sizes: 4x4, 8x8, 16x16, 32x32. Different sizes have different area and delay characteristics. Larger matmuls cause more under-utilization or fragmentation.

For designing large matrix multipliers on an FPGA, we explore regular and systolic composition. We also explore direct programmable interconnect between neighboring interconnect to make composition faster.

We explore various placements of building blocks– Clustered, Surround and Columnar. Clustered placement is asymmetric. Surround placement is recommended in [2]. Columnar placement is commonly used in commercial FPGAs.

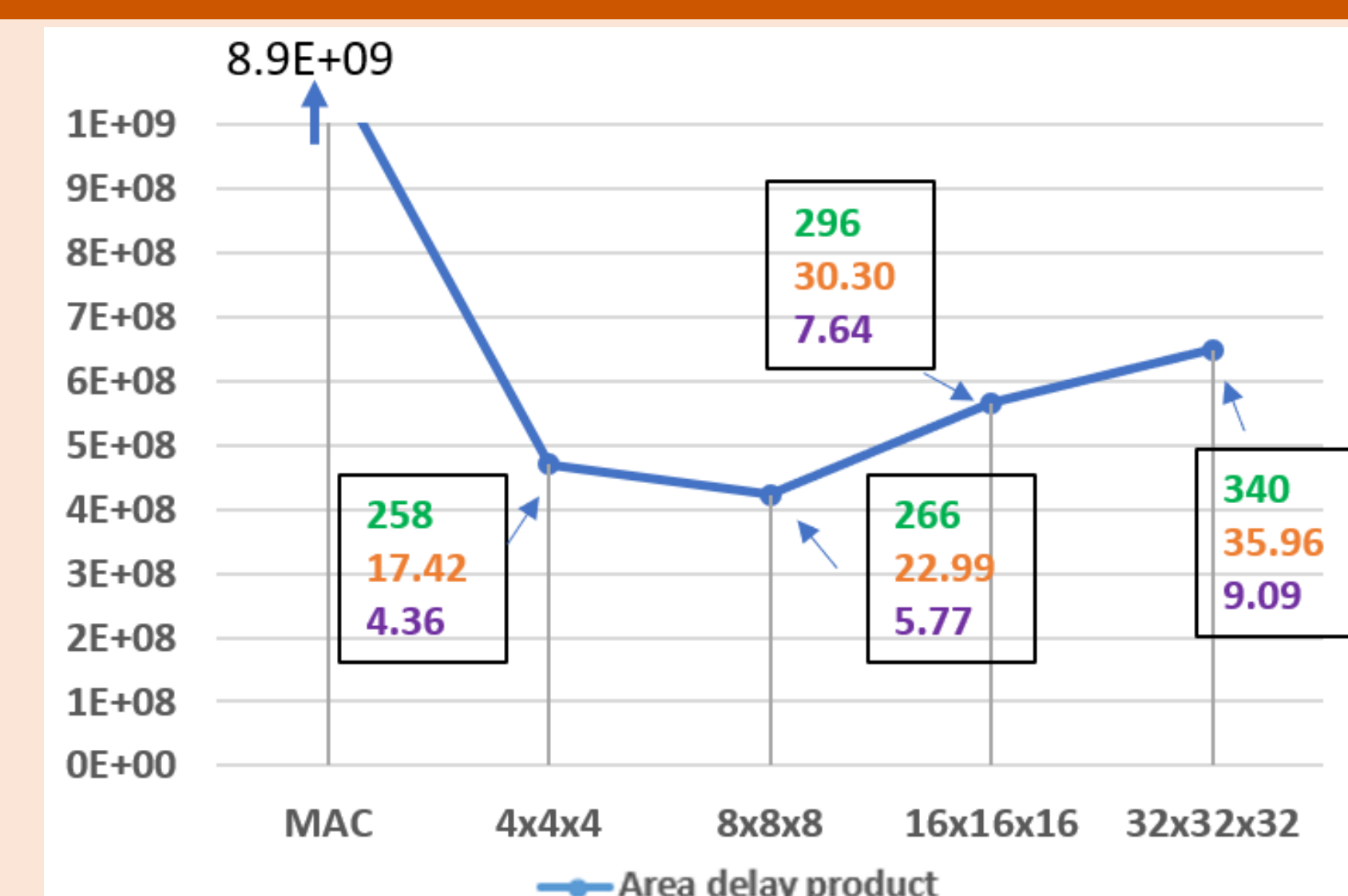
Results



Frequency and area of a 64x64x64 matrix multiplier. We observe that an FPGAs with hard matmuls can bridge the gap between FPGAs with DSPs and ASICs.

Design	Clustered			Surround			Columnar		
	C	A	W	C	A	W	C	A	W
4x4x4	2.5	0.33	66	2.3	1.4	62	2.5	0.33	66
16x16x16	4.0	1.3	288	3.0	2.4	104	2.6	1.0	198
32x32x32	4.1	2.2	410	3.2	3.0	111	2.7	1.7	260
64x64x64	4.2	4.4	496	4.1	5.1	124	3.1	3.7	292

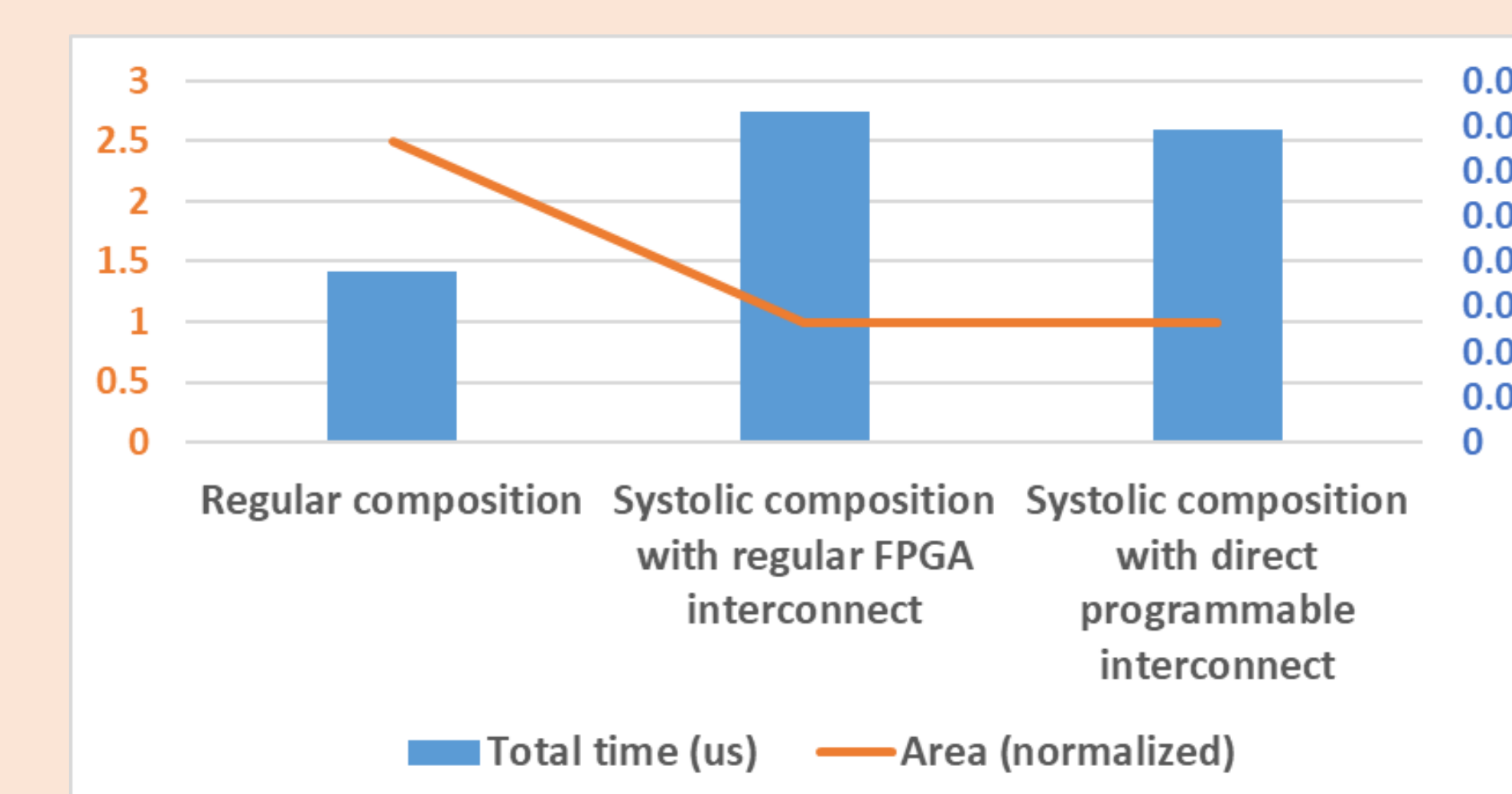
Comparing different placement strategies. C=Critical Path (ns), A=Area(MWTA xE08), W = Channel Width. Clustered is not scalable. Surround yields the lowest channel width, but the critical path is higher for larger designs. Columnar provides lowest area and clock frequency, but yields ML-specific FPGAs



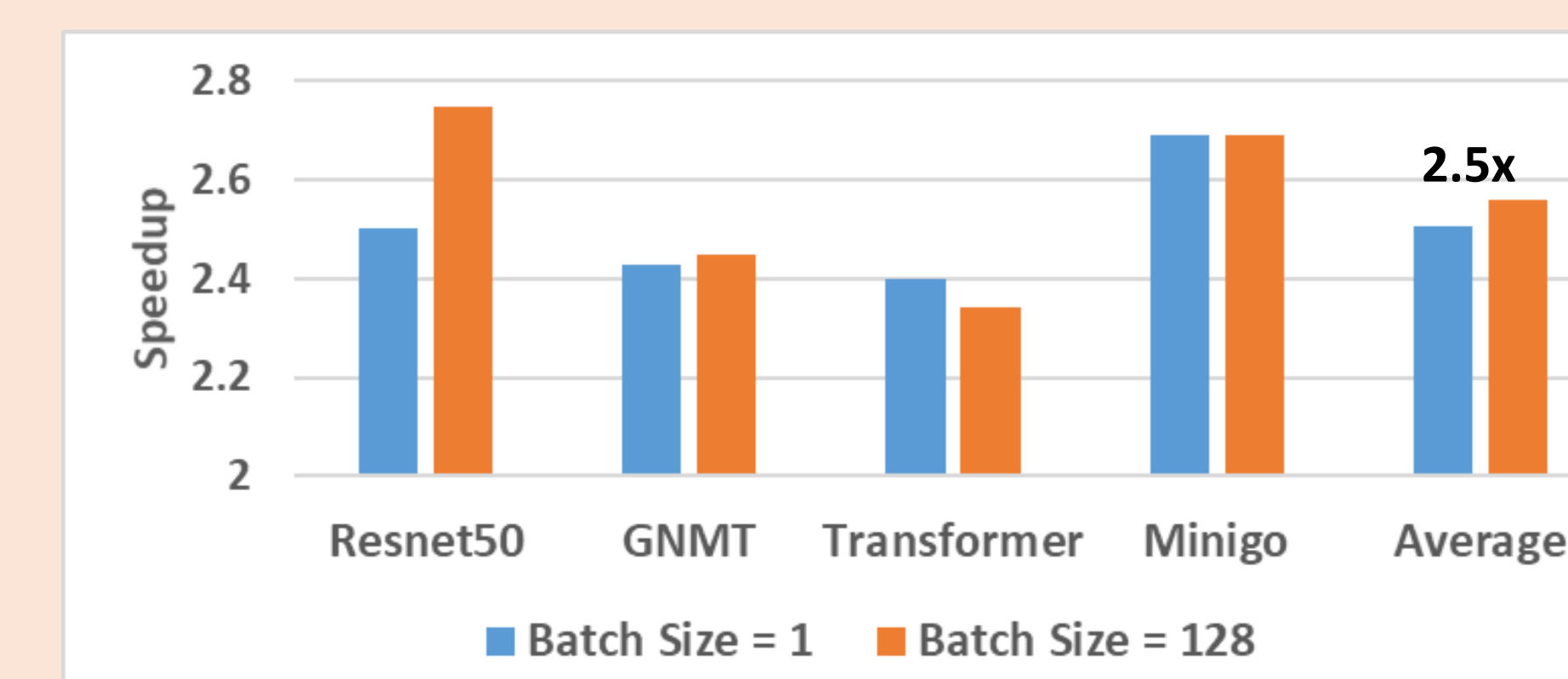
A matrix multiplier with no fragmentation problems (32x32x32) designed using varying granularity of the compute element. Area delay product improves drastically when 4x4x4 matmuls are used.

Matmul	Freq (MHz)	Time (us)	Area (MWTA)	Util
4x4x4	320.77	1.80	3.73E08	0.79
8x8x8	298.14	2.00	3.53E08	0.78
16x16x16	241.79	2.87	3.55E08	0.59
32x32x32	211.90	4.78	3.46E08	0.44

A matrix multiplier with high fragmentation problems (70x70x70) designed using different matmul sizes. Larger matmuls have lower utilization, hence it is better to use smaller matmuls.



Area and time comparison for 8x8x8 matrix multiplier designed using different composition schemes, showing systolic composition is better. We see little benefit from direct interconnect.



End-to-end speedup observed with the proposed architecture compared to baseline FPGA (4x4x4 matmuls and columnar placement)

Conclusions

- Adding hard matrix multipliers to FPGAs can greatly improve the acceleration that FPGA based solutions can provide for AI/ML workloads.
- We recommend adding systolic array based 4x4x4 matrix multiplier blocks to the fabric of FPGAs, because they prove to be the best in the tradeoff between fragmentation/under-utilization and area-delay product.
- Surround placement leads to more versatile FPGAs and can be used in ML-specialized FPGAs. Columnar placement can be used for ML-heavy fabrics that could be a part of a bigger FPGA, the rest of which can have general programmable logic.

- Larger matrix multipliers can be composed systolically by connecting neighboring matmuls. Using direct programmable interconnect doesn't seem to provide much benefit.
- A clock frequency speedup of ~3x and an area improvement of ~8x is observed when a 64x64x64 design is implemented on an FPGA with 4x4x4 hard matrix multiplier blocks vs. on an FPGA with DSP slices.
- We find that providing 4x4x4 hard matrix multiplier blocks in an FPGA speeds up state-of-the-art neural networks from the MLPerf [4] benchmarks by ~2.5x on an average, compared to an FPGA with DSP slices.