THE UNIVERSITY OF TEXAS AT AUSTIN
UT ECE
ELECTRICAL & COMPUTER ENGINEERING

HPC on a Reconfigurable Substrate with Machine Learning Support

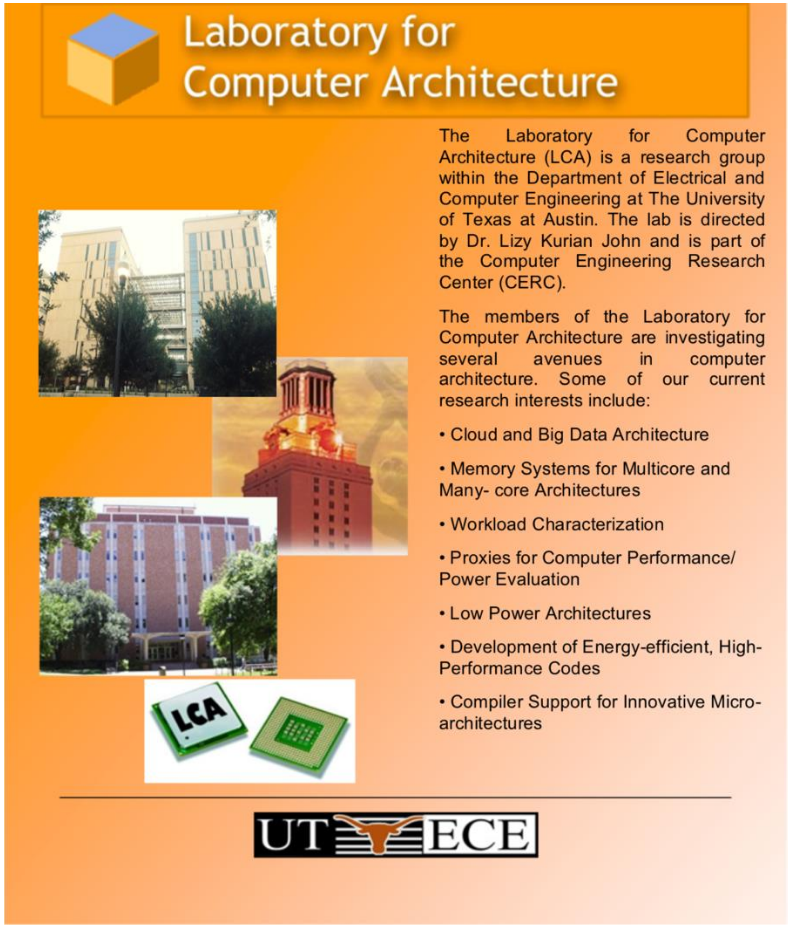
Lizy K. John
Laboratory for Computer Architecture (LCA)
The University of Texas at Austin

Thanks to University of Texas that gave me the chance to have colleagues such as 2019 Nobel winner, Prof. John Goodenough and 2023 Turing award winner Bob Metcalfe.

2019 Nobel Winner
John Goodenough

John B. Goodenough
Virginia H. Cockrell Centennial Chair
of Engineering
Tuesday, October 17, 2017
11:00 AM to 12:30 PM
EER 0.904 (Mulva Auditorium)

2023 Turing
Award Winner
Bob Metcalfe



Laboratory for Computer Architecture

The Laboratory for Computer Architecture (LCA) is a research group within the Department of Electrical and Computer Engineering at The University of Texas at Austin. The lab is directed by Dr. Lizy Kurian John and is part of the Computer Engineering Research Center (CERC).

The members of the Laboratory for Computer Architecture are investigating several avenues in computer architecture. Some of our current research interests include:

- Cloud and Big Data Architecture
- Memory Systems for Multicore and Many- core Architectures
- Workload Characterization
- Proxies for Computer Performance/ Power Evaluation
- Low Power Architectures
- Development of Energy-efficient, High-Performance Codes
- Compiler Support for Innovative Micro-architectures

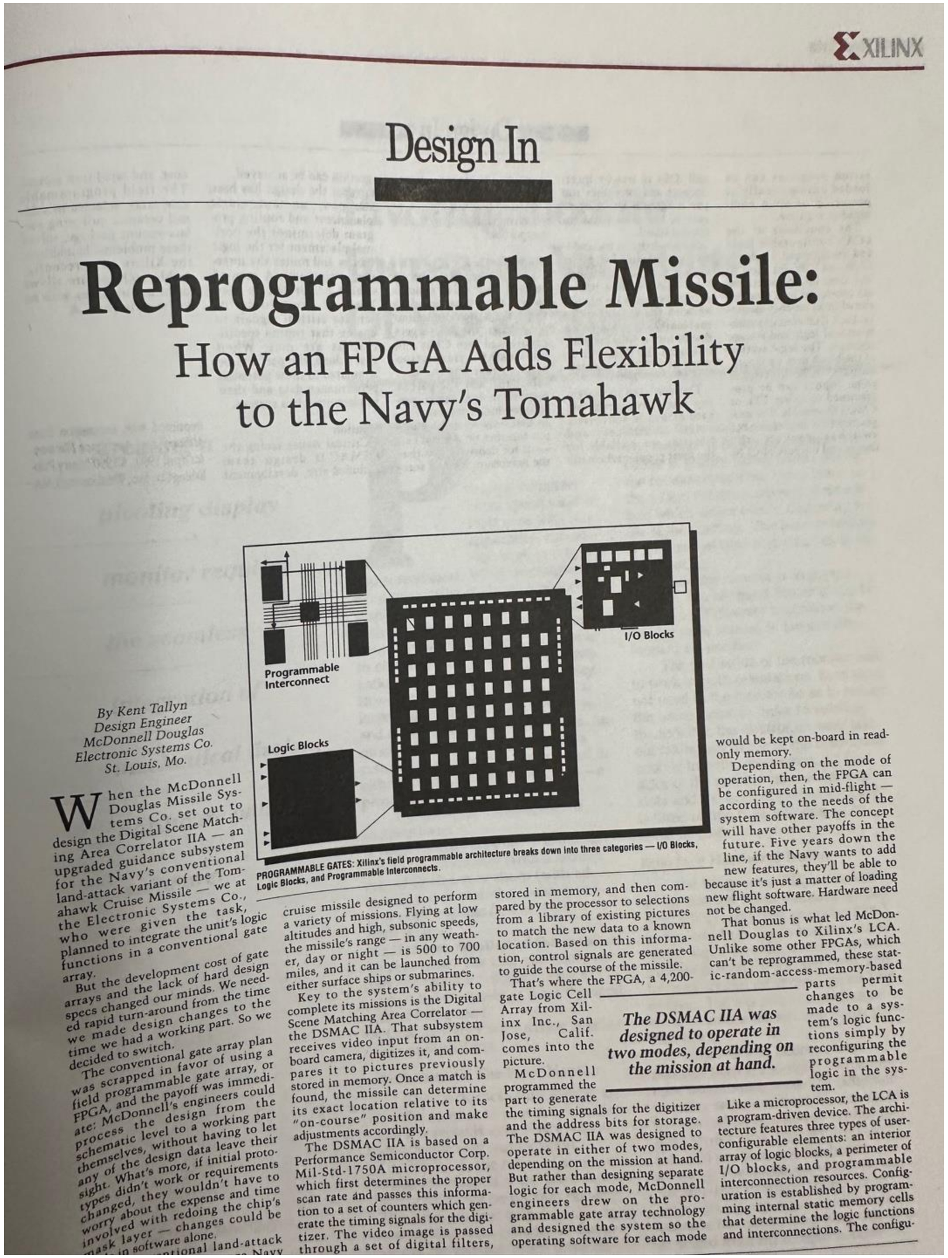
LCA research was supported in part by
Semiconductor Research Corporation (SRC) Tasks 3015.001 and 3148.001,
National Science Foundation (NSF) Grant #2326894,
and NVIDIA Applied Research Accelerator Program Grant.

HPC and Reconfigurable Substrates have Changed a lot since 1990s

Arrival of FPGAs

FPGA Based Reconfigurable Computing

1993


XILINX
Programmable Interconnect
I/O Blocks
Logic Blocks

Design In

Reprogrammable Missile:

How an FPGA Adds Flexibility to the Navy's Tomahawk

By Kent Tallyn
Design Engineer
McDonnell Douglas
Electronic Systems Co.
St. Louis, Mo.

PROGRAMMABLE GATES: Xilinx's field programmable architecture breaks down into three categories — I/O Blocks, Logic Blocks, and Programmable Interconnects.

When the McDonnell Douglas Missile Systems Co. set out to design the Digital Scene Matching Area Correlator IIA — an upgraded guidance subsystem for the Navy's conventional land-attack variant of the Tomahawk Cruise Missile — we at the Electronic Systems Co., who were given the task, planned to integrate the unit's logic functions in a conventional gate array.

But the development cost of gate arrays and the lack of hard design specs changed our minds. We needed rapid turn-around from the time we made design changes to the time we had a working part. So we decided to switch.

The conventional gate array plan was scrapped in favor of using a field programmable gate array, or FPGA, and the payoff was immediate: McDonnell's engineers could process the design from the schematic level to a working part themselves, without having to let any of the design data leave their sight. What's more, if initial prototypes didn't work or requirements changed, they wouldn't have to worry about the expense and time involved with redoing the chip's mask layer — changes could be made in software alone.

The conventional land-attack Navy cruise missile designed to perform a variety of missions. Flying at low altitudes and high, subsonic speeds, the missile's range — in any weather, day or night — is 500 to 700 miles, and it can be launched from either surface ships or submarines.

Key to the system's ability to complete its missions is the Digital Scene Matching Area Correlator — the DSMAC IIA. That subsystem receives video input from an onboard camera, digitizes it, and compares it to pictures previously stored in memory. Once a match is found, the missile can determine its exact location relative to its "on-course" position and make adjustments accordingly.

The DSMAC IIA is based on a Performance Semiconductor Corp. Mil-Std-1750A microprocessor, which first determines the proper scan rate and passes this information to a set of counters which generate the timing signals for the digitizer. The video image is passed through a set of digital filters, stored in memory, and then compared by the processor to selections from a library of existing pictures to match the new data to a known location. Based on this information, control signals are generated to guide the course of the missile.

That's where the FPGA, a 4,200-gate Logic Cell Array from Xilinx Inc., San Jose, Calif. comes into the picture.

The DSMAC IIA was designed to operate in two modes, depending on the mission at hand.

McDonnell programmed the part to generate the timing signals for the digitizer and the address bits for storage. The DSMAC IIA was designed to operate in either of two modes, depending on the mission at hand. But rather than designing separate logic for each mode, McDonnell engineers drew on the programmable gate array technology and designed the system so the operating software for each mode would be kept on-board in read-only memory.

Depending on the mode of operation, then, the FPGA can be configured in mid-flight — according to the needs of the system software. The concept will have other payoffs in the future. Five years down the line, if the Navy wants to add new features, they'll be able to because it's just a matter of loading new flight software. Hardware need not be changed.

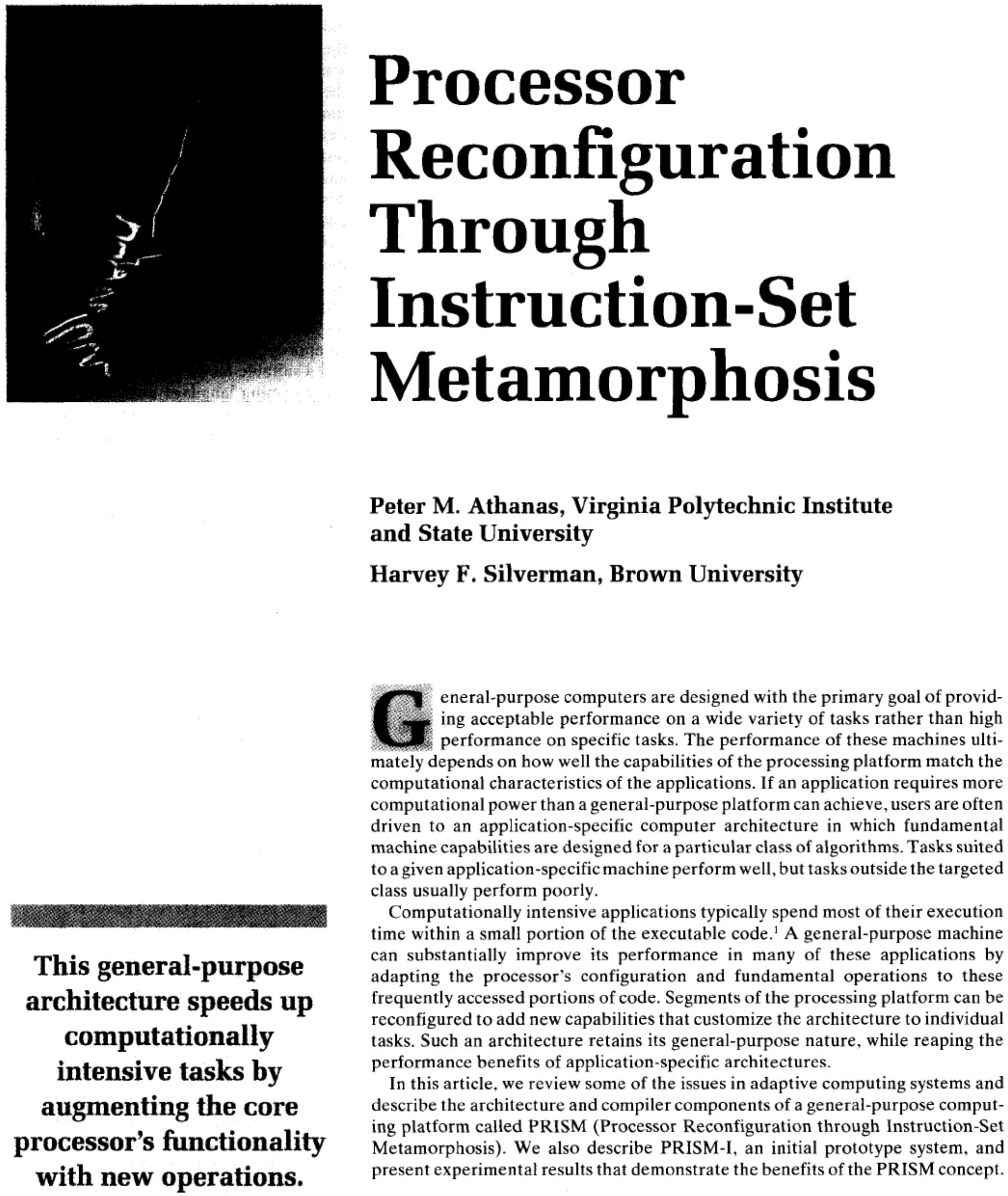
That bonus is what led McDonnell Douglas to Xilinx's LCA. Unlike some other FPGAs, which can't be reprogrammed, these static-random-access-memory-based parts permit changes to be made to a system's logic functions simply by reconfiguring the programmable logic in the system.

Like a microprocessor, the LCA is a program-driven device. The architecture features three types of user-configurable elements: an interior array of logic blocks, a perimeter of I/O blocks, and programmable interconnection resources. Configuration is established by programming internal static memory cells that determine the logic functions and interconnections. The configu-

Instruction Set Metamorphosis with FPGAs

- **IEEE Computer Magazine, March 1993**

IEEE Computer Magazine, March 1993



Processor Reconfiguration Through Instruction-Set Metamorphosis

Peter M. Athanas, Virginia Polytechnic Institute and State University

Harvey F. Silverman, Brown University

This general-purpose architecture speeds up computationally intensive tasks by augmenting the core processor's functionality with new operations.

General-purpose computers are designed with the primary goal of providing acceptable performance on a wide variety of tasks rather than high performance on specific tasks. The performance of these machines ultimately depends on how well the capabilities of the processing platform match the computational characteristics of the applications. If an application requires more computational power than a general-purpose platform can achieve, users are often driven to an application-specific computer architecture in which fundamental machine capabilities are designed for a particular class of algorithms. Tasks suited to a given application-specific machine perform well, but tasks outside the targeted class usually perform poorly.

Computationally intensive applications typically spend most of their execution time within a small portion of the executable code.[1] A general-purpose machine can substantially improve its performance in many of these applications by adapting the processor's configuration and fundamental operations to these frequently accessed portions of code. Segments of the processing platform can be reconfigured to add new capabilities that customize the architecture to individual tasks. Such an architecture retains its general-purpose nature, while reaping the performance benefits of application-specific architectures.

In this article, we review some of the issues in adaptive computing systems and describe the architecture and compiler components of a general-purpose computing platform called PRISM (Processor Reconfiguration through Instruction-Set Metamorphosis). We also describe PRISM-I, an initial prototype system, and present experimental results that demonstrate the benefits of the PRISM concept.

PRISM (Athanas, 1993 March)

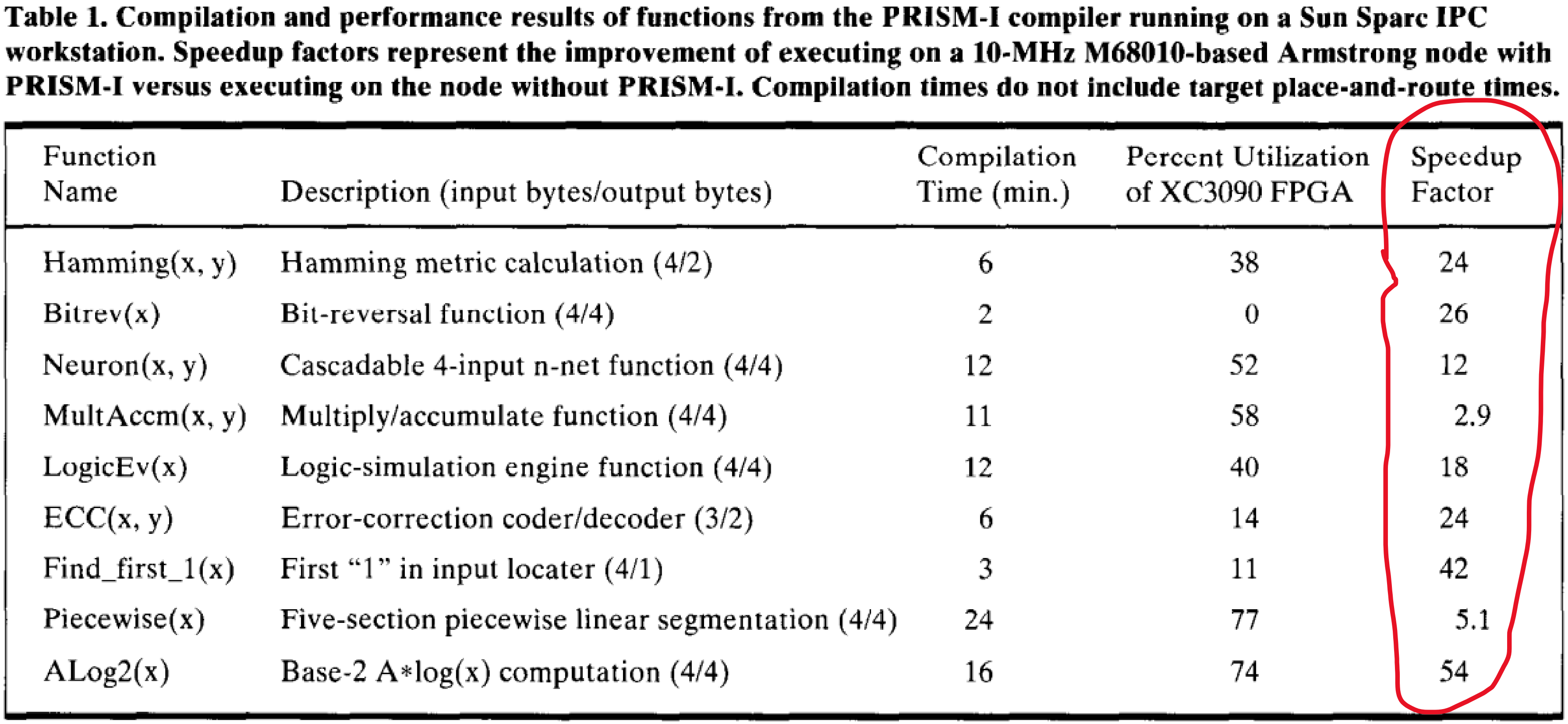
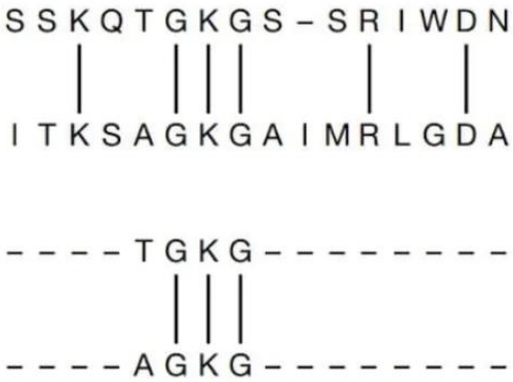


Table 1. Compilation and performance results of functions from the PRISM-I compiler running on a Sun Sparc IPC workstation. Speedup factors represent the improvement of executing on a 10-MHz M68010-based Armstrong node with PRISM-I versus executing on the node without PRISM-I. Compilation times do not include target place-and-route times.

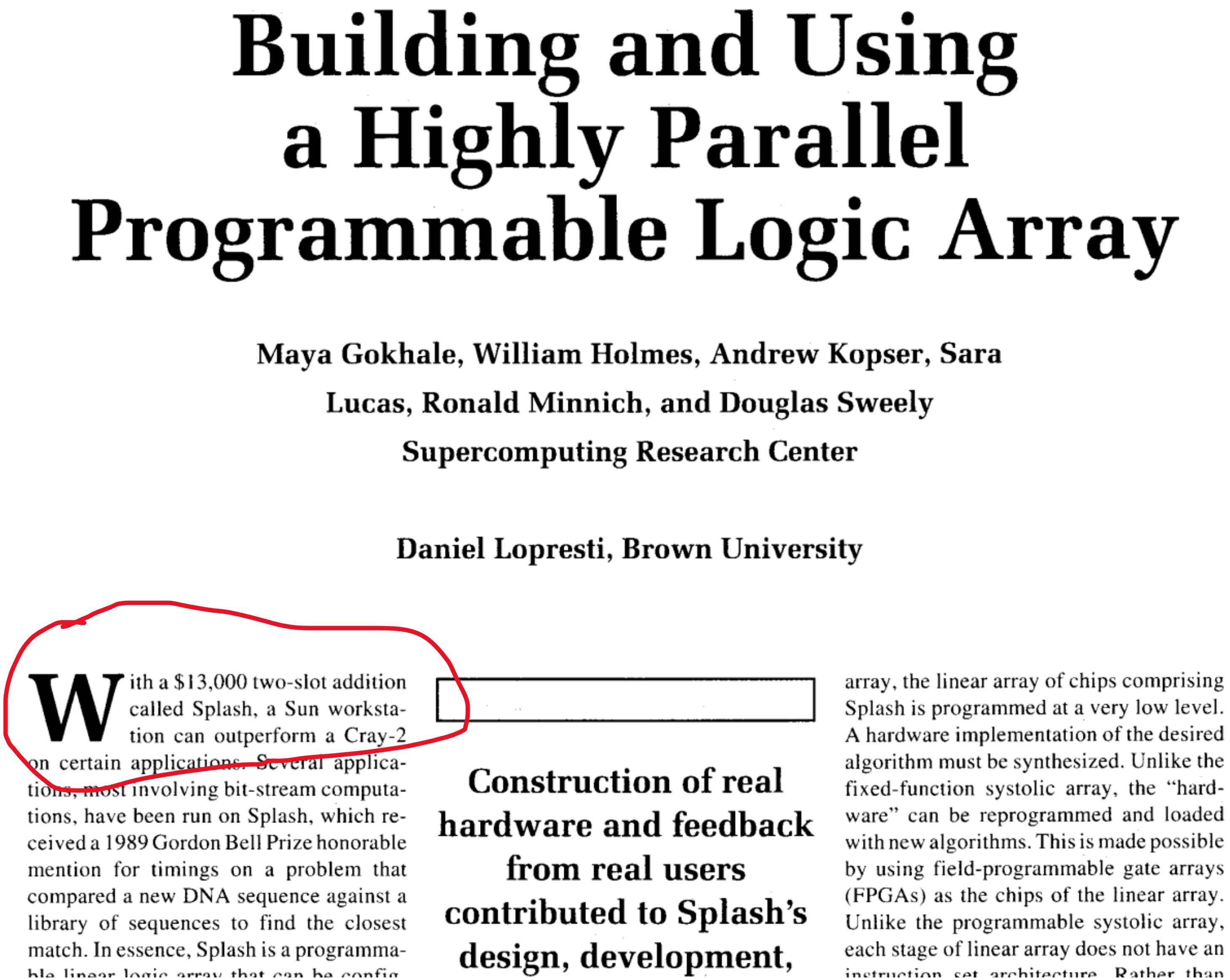
Function Name	Description (input bytes/output bytes)	Compilation Time (min.)	Percent Utilization of XC3090 FPGA	Speedup Factor
Hamming(x, y)	Hamming metric calculation (4/2)	6	38	24
Bitrev(x)	Bit-reversal function (4/4)	2	0	26
Neuron(x, y)	Cascadable 4-input n-net function (4/4)	12	52	12
MultAccm(x, y)	Multiply/accumulate function (4/4)	11	58	2.9
LogicEv(x)	Logic-simulation engine function (4/4)	12	40	18
ECC(x, y)	Error-correction coder/decoder (3/2)	6	14	24
Find_first_1(x)	First "1" in input locater (4/1)	3	11	42
Piecewise(x)	Five-section piecewise linear segmentation (4/4)	24	77	5.1
ALog2(x)	Base-2 A∗log(x) computation (4/4)	16	74	54

Sequence Comparison using SPLASH (Gokhale, 1991)



```
S S K Q T G K G S – S R I W D N
    |     | | |       |     |
I T K S A G K G A I M R L G D A

– – – – T G K G – – – – – – – –
          | | |
– – – – A G K G – – – – – – – –
```



Building and Using a Highly Parallel Programmable Logic Array

Maya Gokhale, William Holmes, Andrew Kopser, Sara Lucas, Ronald Minnich, and Douglas Sweely
Supercomputing Research Center

Daniel Lopresti, Brown University

With a $13,000 two-slot addition called Splash, a Sun workstation can outperform a Cray-2 on certain applications. Several applications, most involving bit-stream computations, have been run on Splash, which received a 1989 Gordon Bell Prize honorable mention for timings on a problem that compared a new DNA sequence against a library of sequences to find the closest match. In essence, Splash is a programmable linear logic array that can be config-

Construction of real hardware and feedback from real users contributed to Splash's design, development,

array, the linear array of chips comprising Splash is programmed at a very low level. A hardware implementation of the desired algorithm must be synthesized. Unlike the fixed-function systolic array, the "hardware" can be reprogrammed and loaded with new algorithms. This is made possible by using field-programmable gate arrays (FPGAs) as the chips of the linear array. Unlike the programmable systolic array, each stage of linear array does not have an instruction set architecture. Rather than

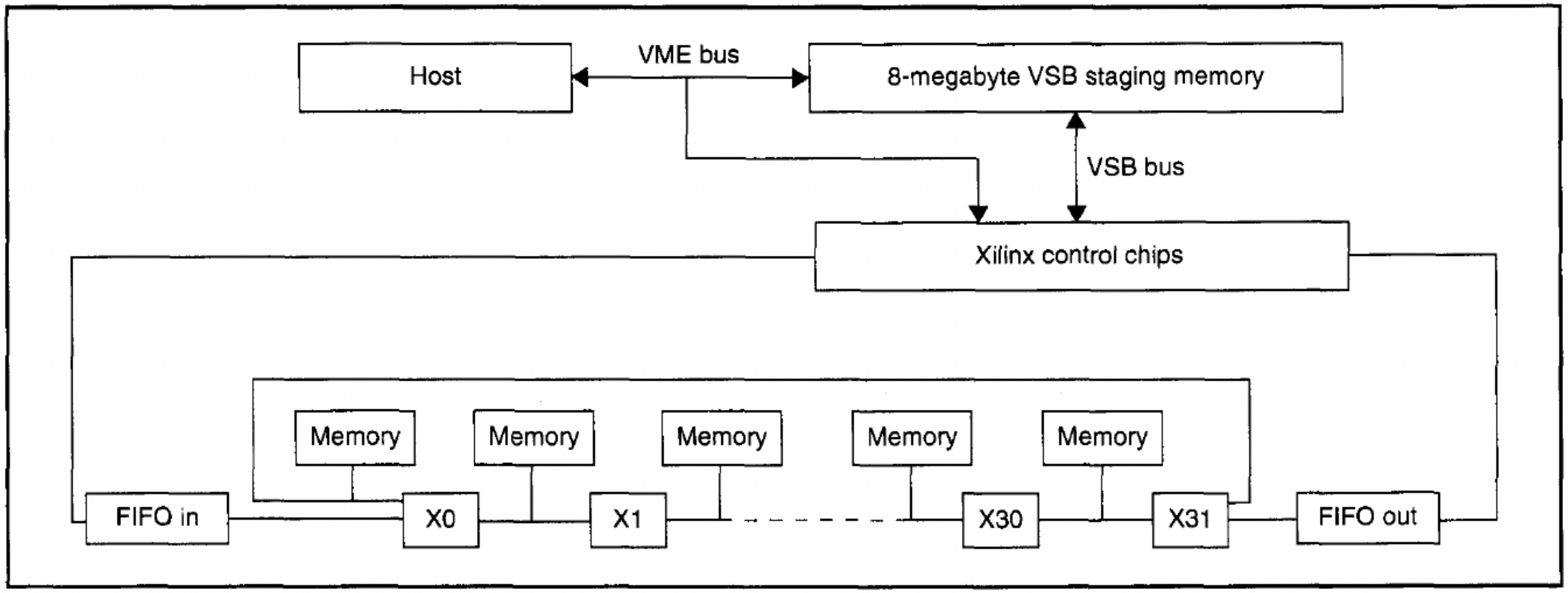


Figure 1. The 32-stage linear array.

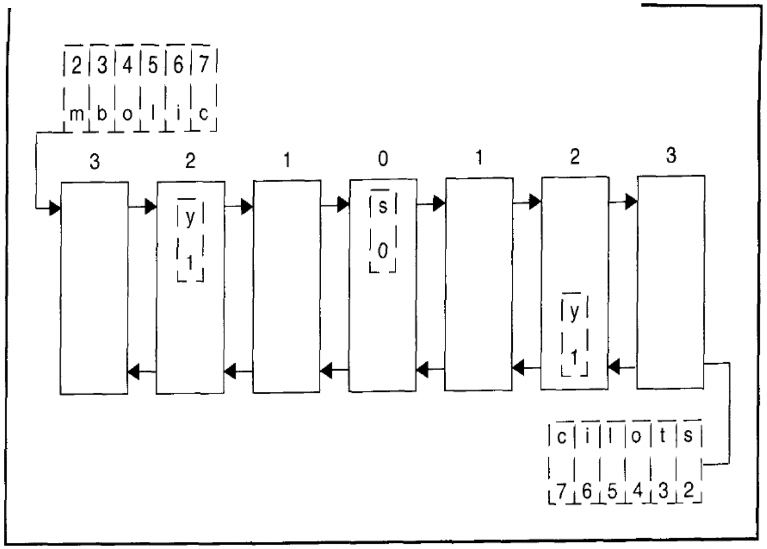
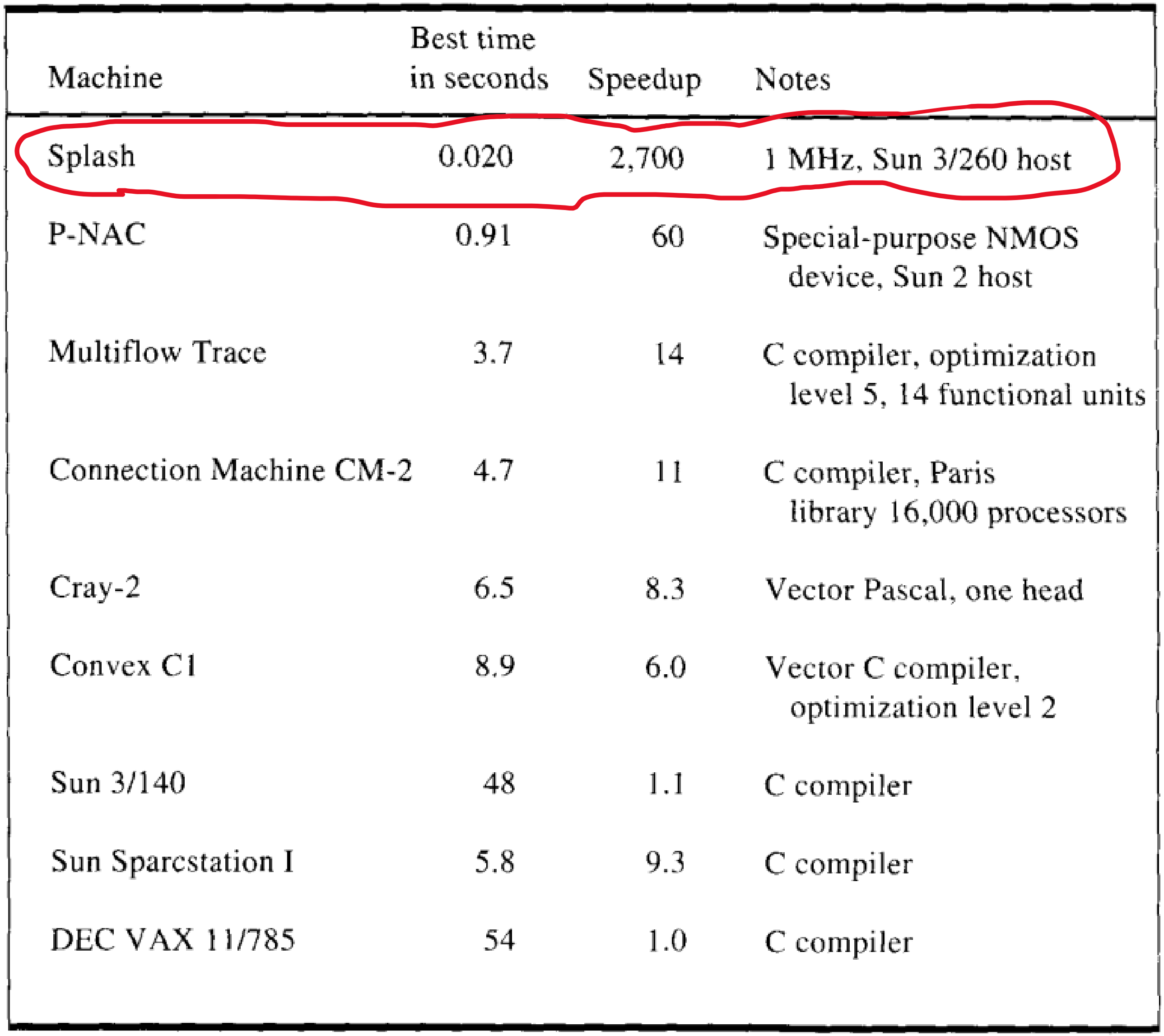


Figure 7. A linear systolic array for sequence comparison.

Table 1. Benchmark results for 100 comparisons of 100-long sequences.



Machine	Best time in seconds	Speedup	Notes
Splash	0.020	2,700	1 MHz, Sun 3/260 host
P-NAC	0.91	60	Special-purpose NMOS device, Sun 2 host
Multiflow Trace	3.7	14	C compiler, optimization level 5, 14 functional units
Connection Machine CM-2	4.7	11	C compiler, Paris library 16,000 processors
Cray-2	6.5	8.3	Vector Pascal, one head
Convex C1	8.9	6.0	Vector C compiler, optimization level 2
Sun 3/140	48	1.1	C compiler
Sun Sparcstation I	5.8	9.3	C compiler
DEC VAX 11/785	54	1.0	C compiler

FPGA Evolution

Sea of CLBs

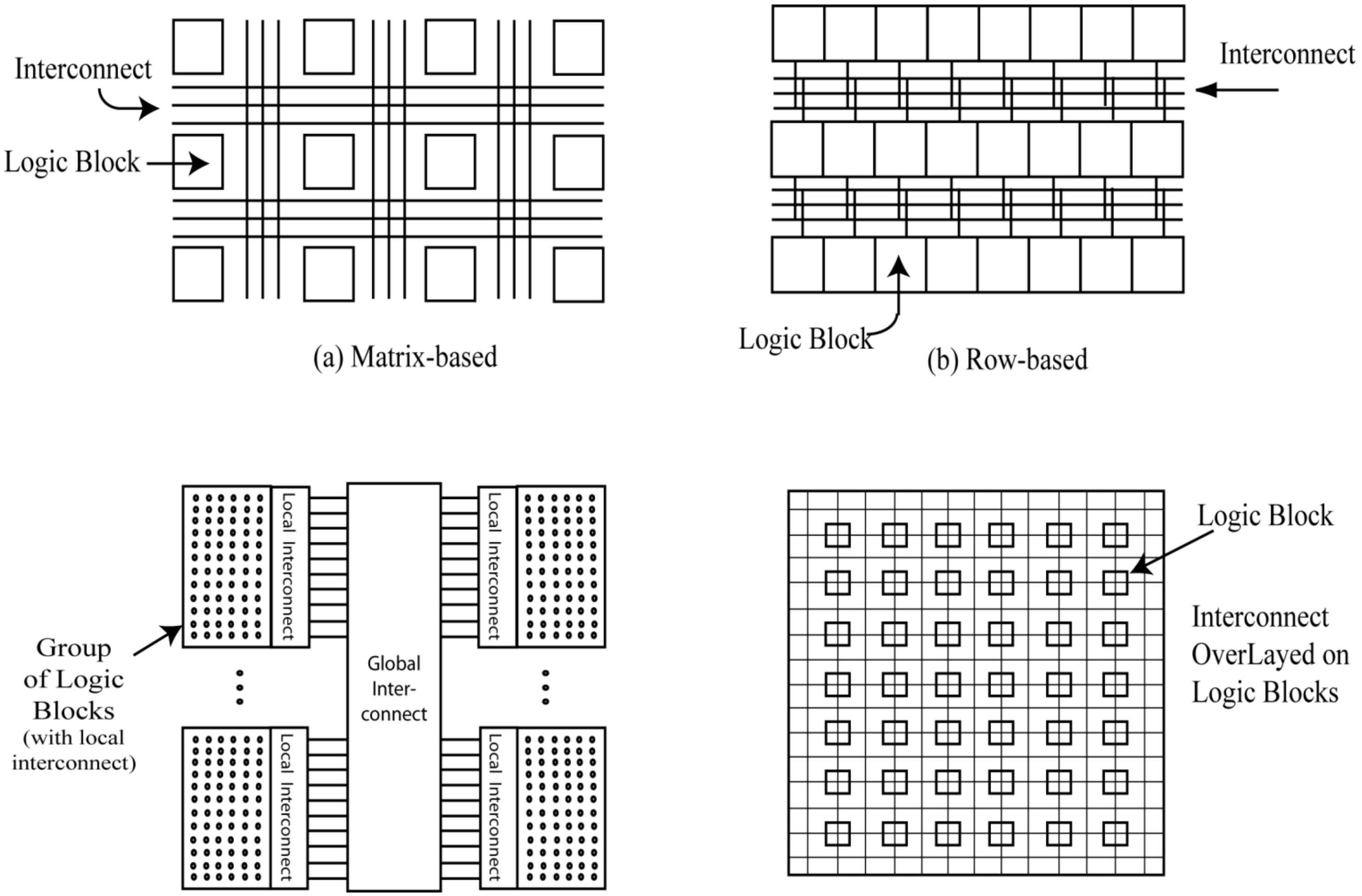
Block RAMs

Embedded CPUs

DSP Slices (Sea of MACs)

ML Specific FPGAs (Xilinx Versal, Intel TensorBlocks)

Architectures for FPGAs

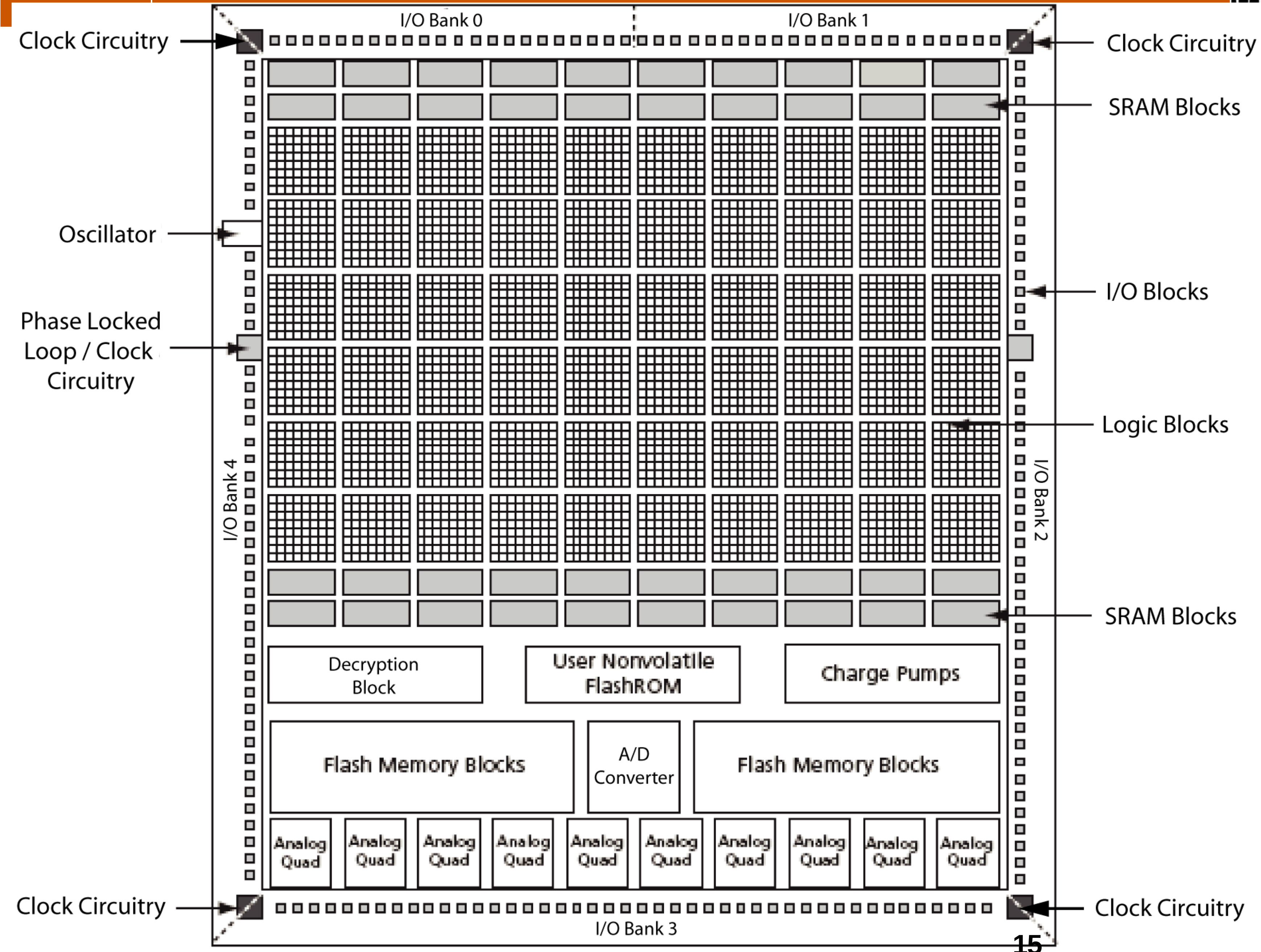


(a) Matrix-based

(b) Row-based

(c) Hierarchical

(d) Sea-of-Gates


Clock Circuitry
I/O Bank 0
I/O Bank 1
Clock Circuitry
SRAM Blocks
Oscillator
I/O Blocks
Phase Locked Loop / Clock Circuitry
Logic Blocks
I/O Bank 4
I/O Bank 2
SRAM Blocks
Decryption Block
User Nonvolatile FlashROM
Charge Pumps
Flash Memory Blocks
A/D Converter
Flash Memory Blocks
Analog Quad
Analog Quad
Analog Quad
Analog Quad
Analog Quad
Analog Quad
Analog Quad
Analog Quad
Analog Quad
Analog Quad
Clock Circuitry
I/O Bank 3
Clock Circuitry

HPC has changed too

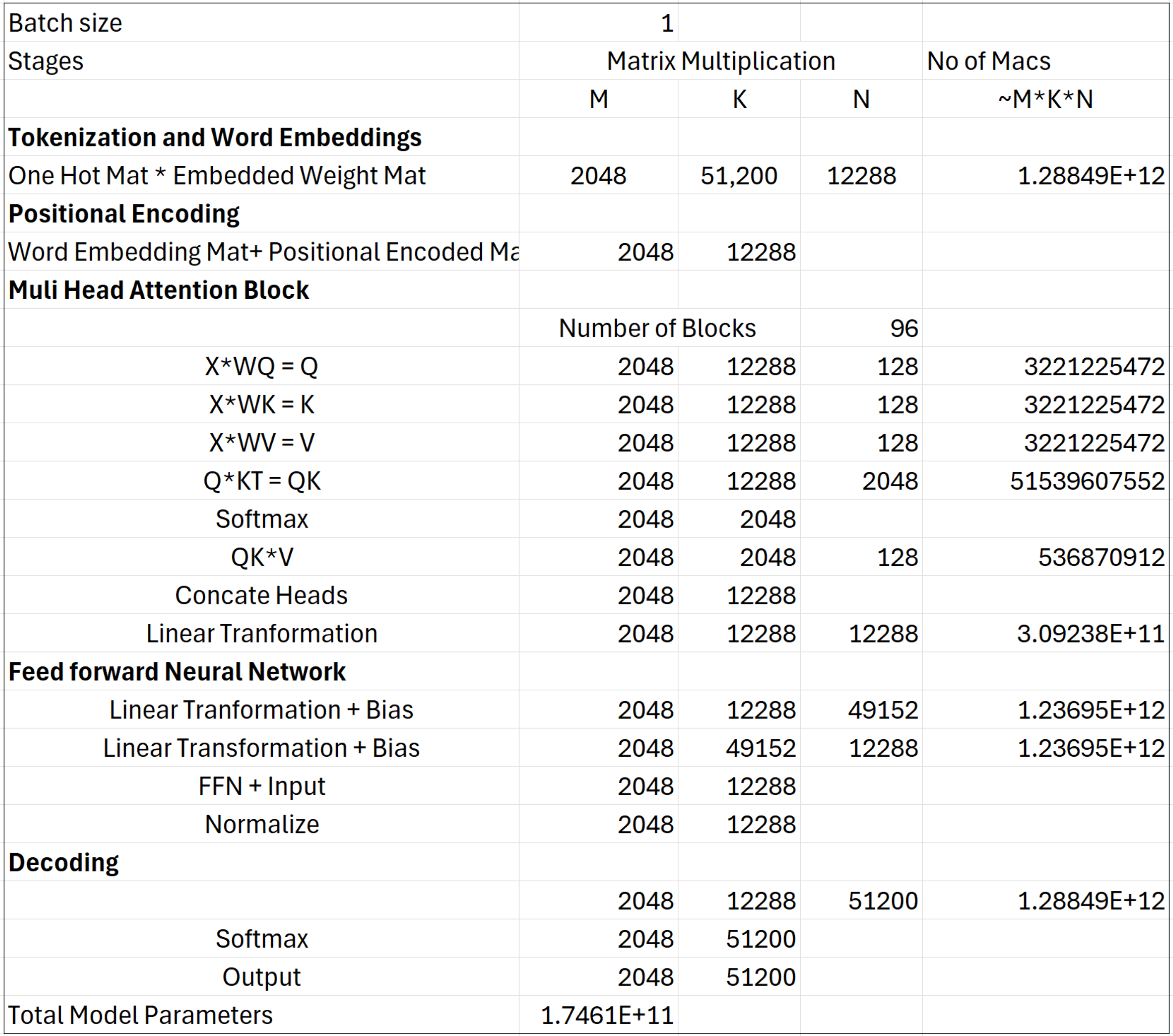
AI is the new HPC

AI is taking over as the primary technology used to tackle complex computational problems

AI is becoming the key tool for performing complex simulations and data analysis

Impressive ability to handle large datasets and intricate models.

GPT3 Model Parameters and Multiplications



Batch size	1			
Stages	Matrix Multiplication			No of Macs
	M	K	N	~M*K*N
Tokenization and Word Embeddings				
One Hot Mat * Embedded Weight Mat	2048	51,200	12288	1.28849E+12
Positional Encoding				
Word Embedding Mat+ Positional Encoded Ma	2048	12288		
Muli Head Attention Block				
	Number of Blocks		96	
X*WQ = Q	2048	12288	128	3221225472
X*WK = K	2048	12288	128	3221225472
X*WV = V	2048	12288	128	3221225472
Q*KT = QK	2048	12288	2048	51539607552
Softmax	2048	2048		
QK*V	2048	2048	128	536870912
Concate Heads	2048	12288		
Linear Tranformation	2048	12288	12288	3.09238E+11
Feed forward Neural Network				
Linear Tranformation + Bias	2048	12288	49152	1.23695E+12
Linear Transformation + Bias	2048	49152	12288	1.23695E+12
FFN + Input	2048	12288		
Normalize	2048	12288		
Decoding				
	2048	12288	51200	1.28849E+12
Softmax	2048	51200		
Output	2048	51200		
Total Model Parameters	1.7461E+11			

All roads lead to GEMM

GEMM has been the bread and butter of HPC

HPC done via AI

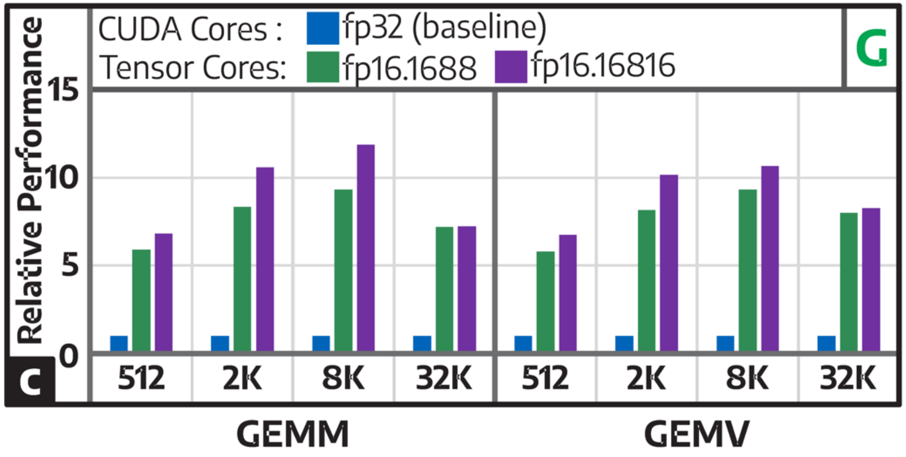
HPC done in conjunction with AI

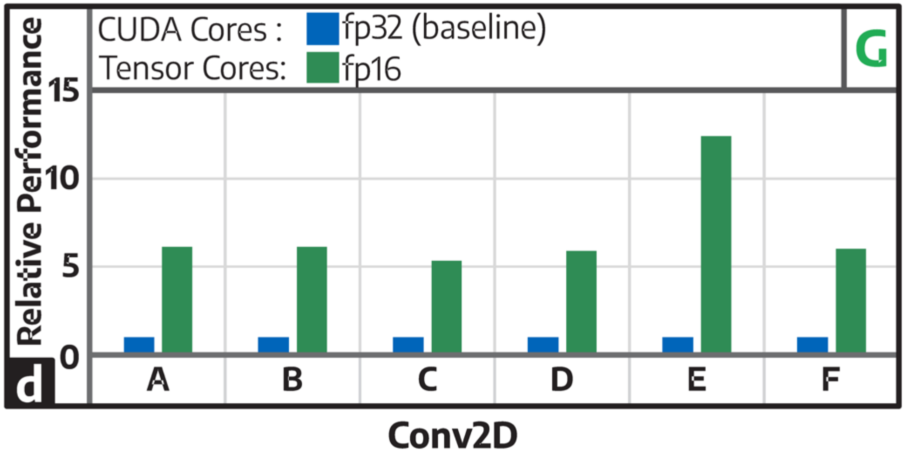
Whether HPC or AI,

All roads lead to GEMM

Programmable Matrix Accelerators – Tensor Cores

- Average speed-up on FP16 Tensor Cores compared to FP32 CUDA Cores:
  - **GEMM**: 7.69 × (hmma.1688), 9.14x (hmma.16816)
  - **GEMV**: 7.82 × (hmma.1688), 8.96x (hmma.16816)
  - **Conv2D**: 6.99 × (hmma.1688)



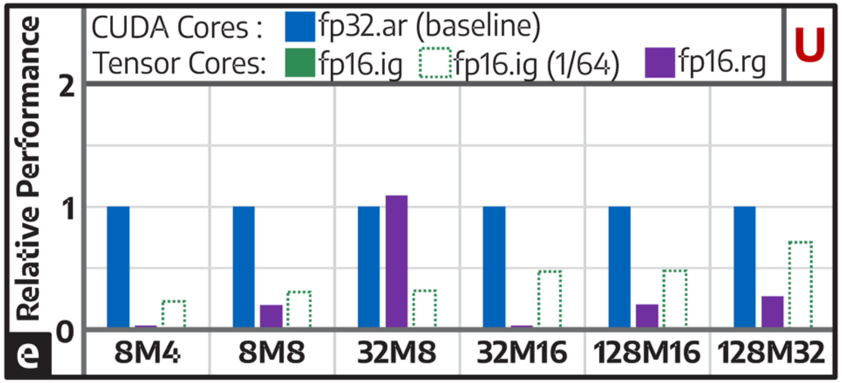


Reshaping Matrix Accelerators to do other Functions

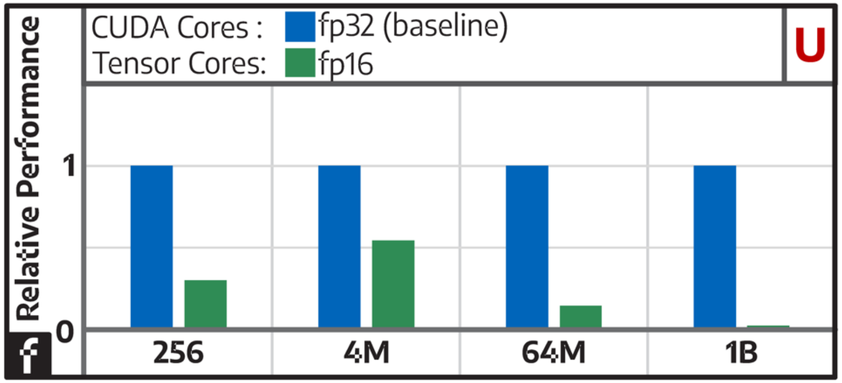
- In general, FIR and ElWiseAdd see performance degradation on Tensor Cores despite transformation.

- By default, they cannot run on Tensor Cores.

- Average speed-up on FP16 Tensor Cores compared to FP32 CUDA Cores:
  - **FIR**: 0.30 × (reshaped GEMV), 0.01x (implicit GEMV)
  - **ElWiseAdd**: 0.25 ×

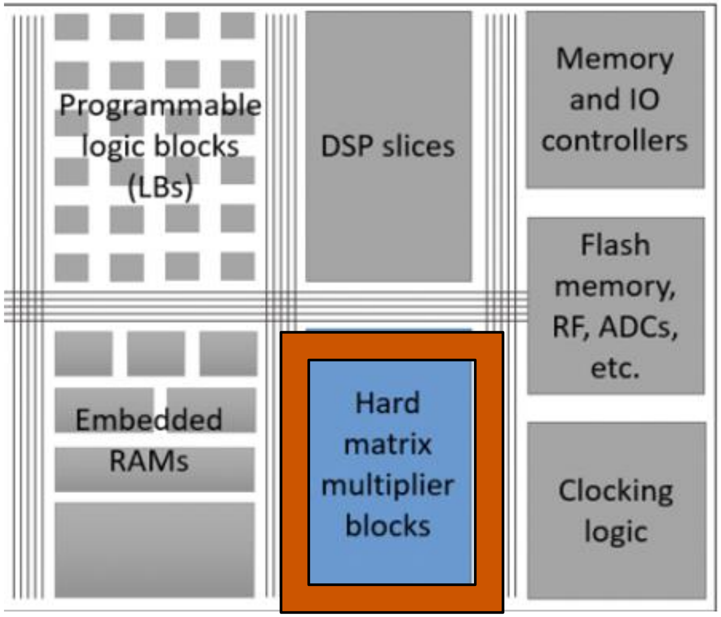


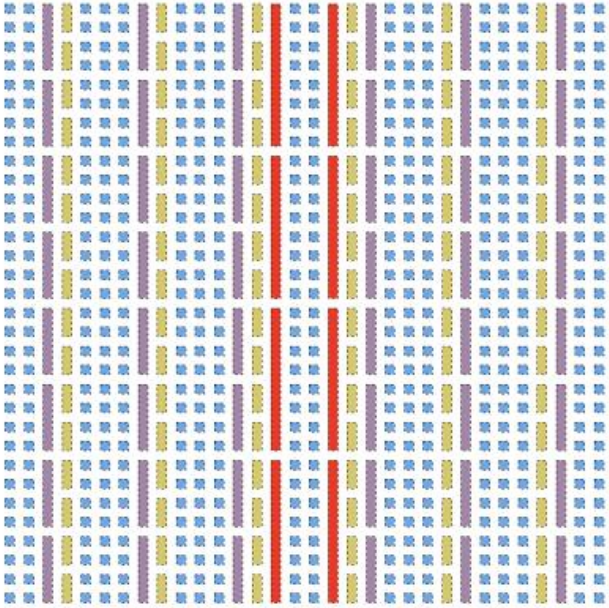
FIR



ElWiseAdd

Tensor Slices: Hardening ML Specific Blocks


Programmable logic blocks (LBs)
DSP slices
Memory and IO controllers
Flash memory, RF, ADCs, etc.
Embedded RAMs
Hard matrix multiplier blocks
Clocking logic

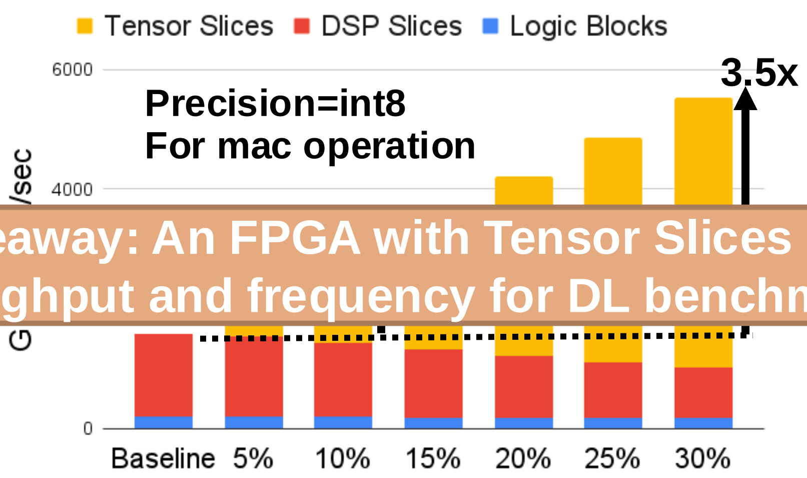


General goal – Higher performance and Lower energy

Arora et al., Tensor Slices to the Rescue: Supercharging ML Acceleration on FPGAs, FPGA 2021

Compute Throughput and Frequency Improvement

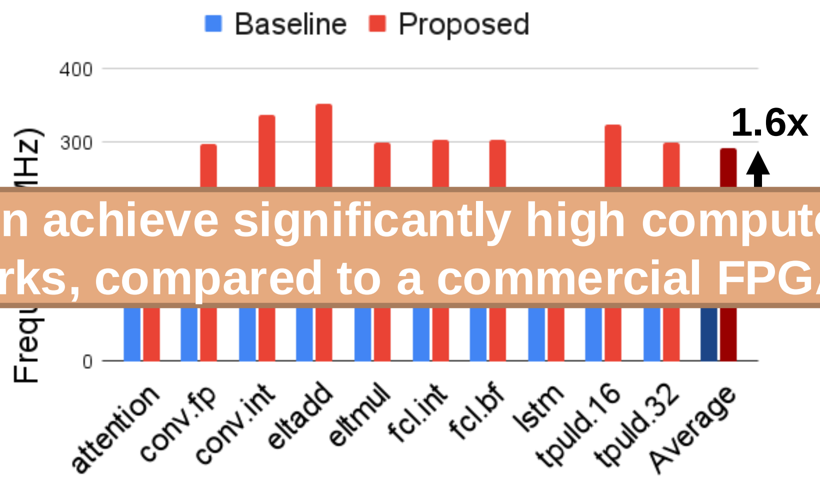
Higher is better



Precision=int8
For mac operation

3.5x

Higher is
better



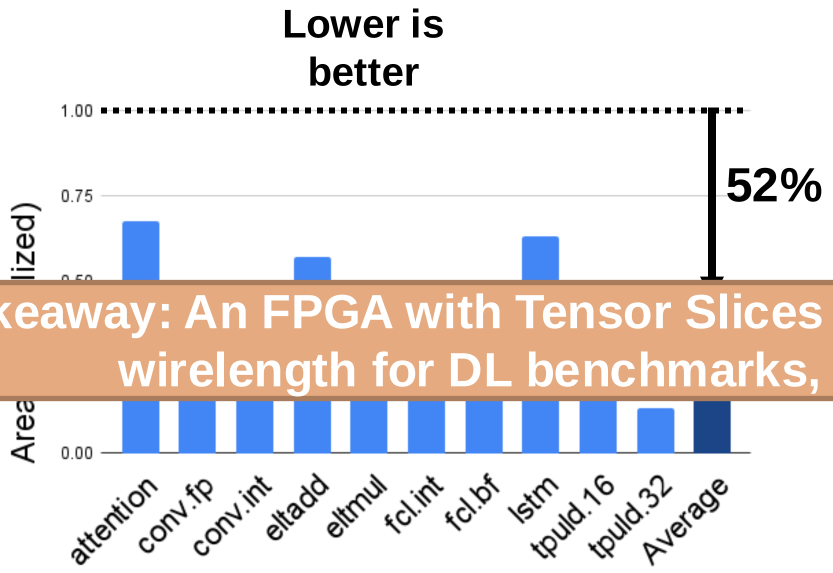
1.6x

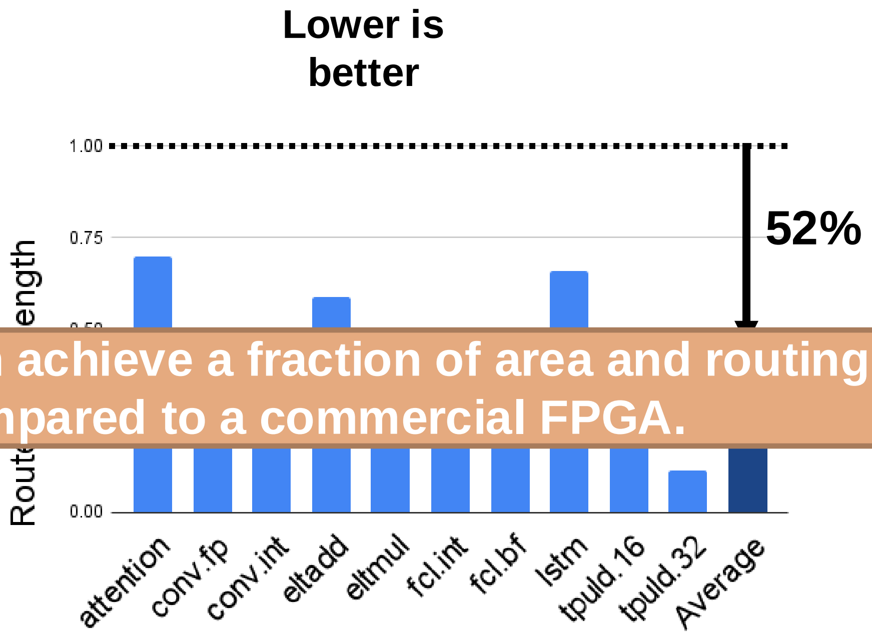
Takeaway: An FPGA with Tensor Slices can achieve significantly high compute throughput and frequency for DL benchmarks, compared to a commercial FPGA.

Percent of area converted to tensor slices

Not extra area

Area and Routing Wirelength Reduction

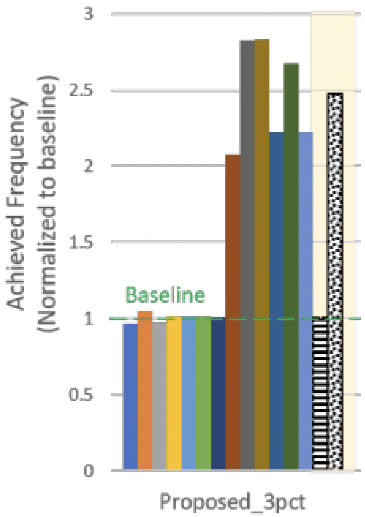




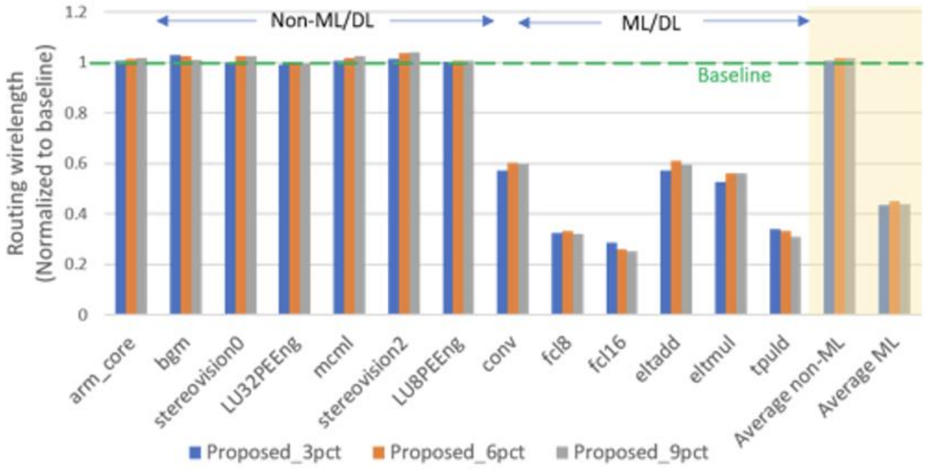
Takeaway: An FPGA with Tensor Slices can achieve a fraction of area and routing wirelength for DL benchmarks, compared to a commercial FPGA.

Tensor Slices: Non-ML Benchmarks do not slowdown

Higher is better

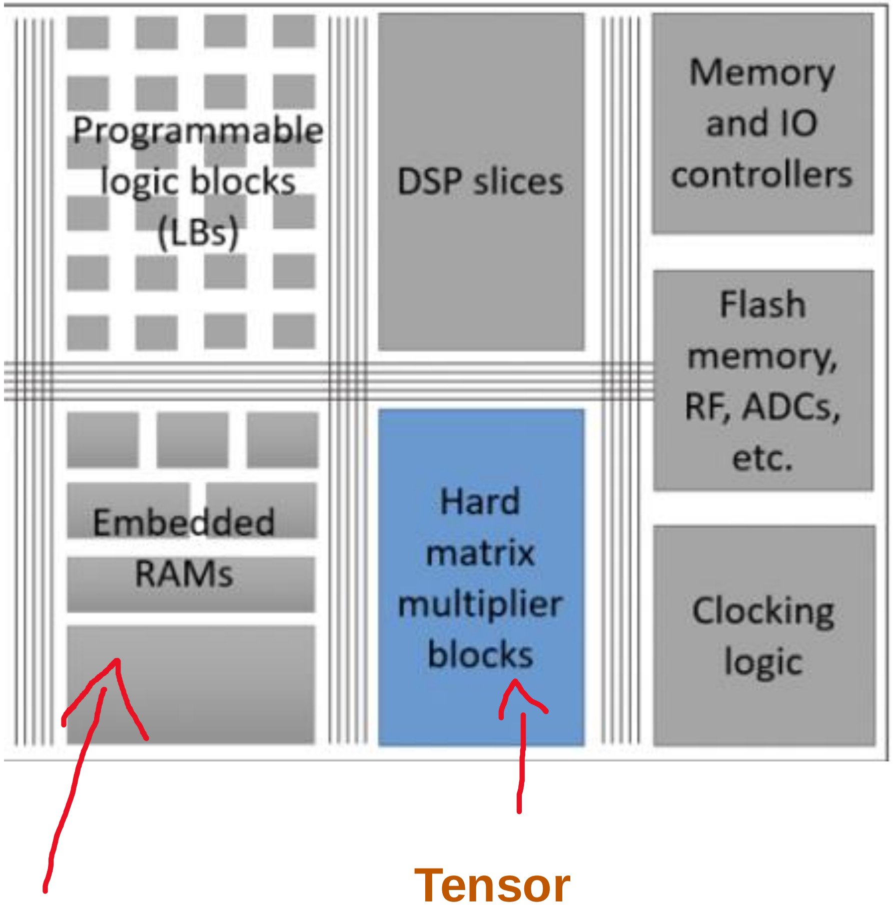


Lower is better



First 7 bars are for non-ML Benchmarks
Next 6 are for ML benchmarks
Last 2 bars are averages.
Last bar is average for ML

Intelligent Compute Fabrics: Supercharging ML Acceleration on FPGAs – Compute-RAM Slices

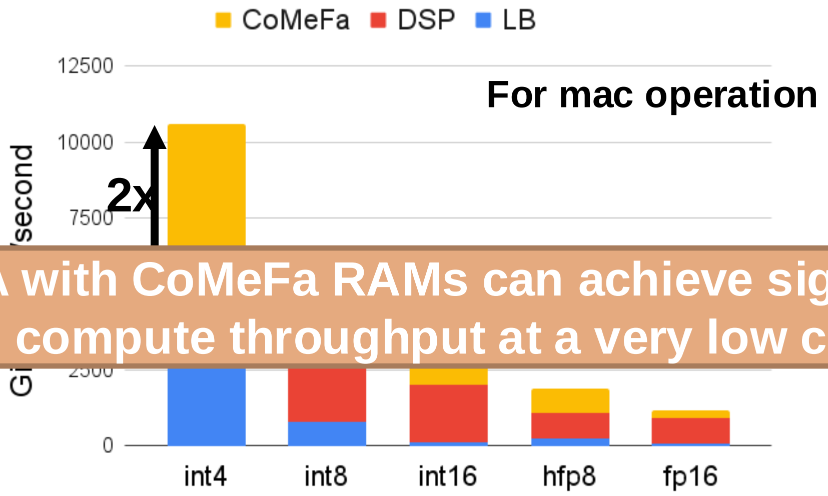

Programmable logic blocks (LBs)
DSP slices
Memory and IO controllers
Flash memory, RF, ADCs, etc.
Embedded RAMs
Hard matrix multiplier blocks
Clocking logic

Compute in RAMs

Tensor Slices

Compute Throughput Improvement

Higher is better



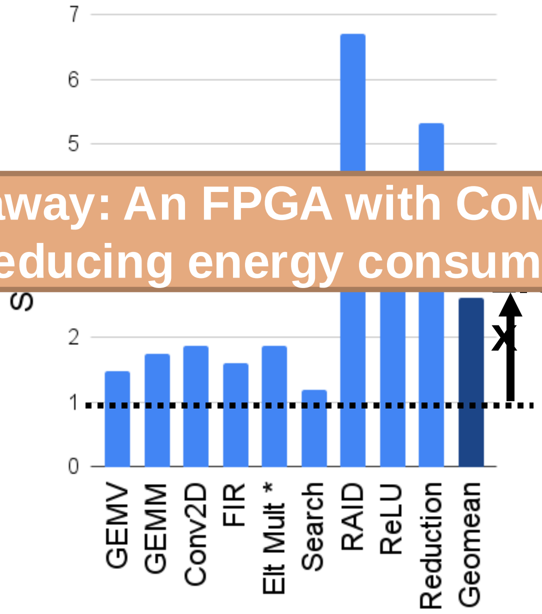
For mac operation

Takeaway: An FPGA with CoMeFa RAMs can achieve significant improvement in compute throughput at a very low cost.

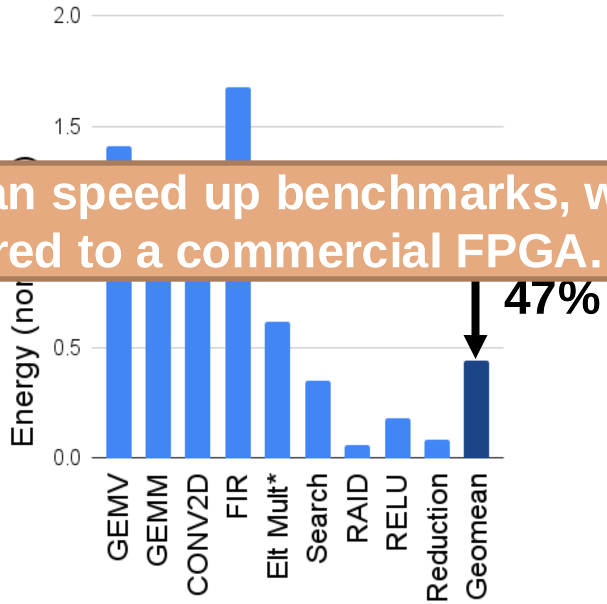
3.8% area overhead at the chip level

Speedup and Energy Reduction

Higher is better



Lower is better

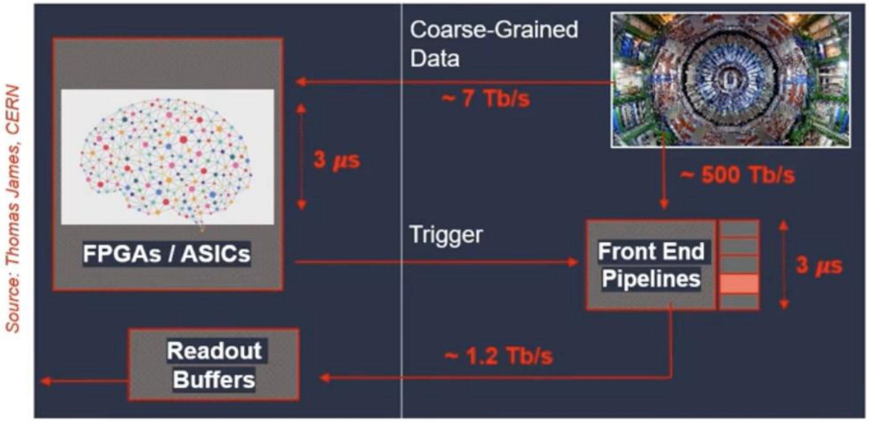


Takeaway: An FPGA with CoMeFa RAMs can speed up benchmarks, while reducing energy consumption, compared to a commercial FPGA.

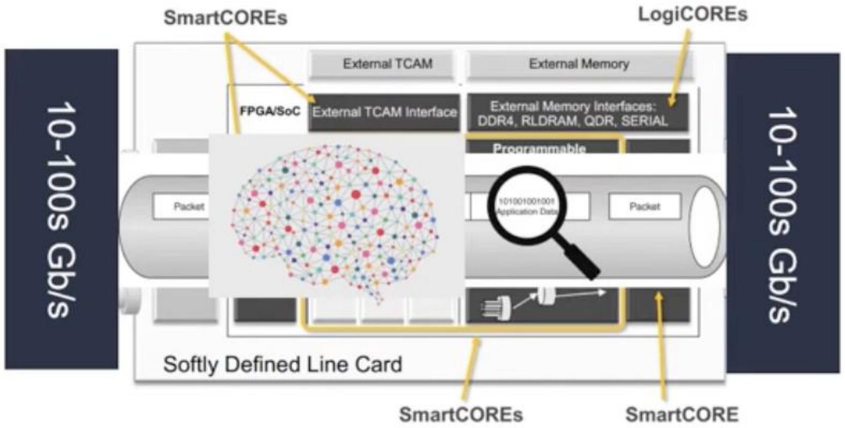
Era of Chiplets

2.5D and 3D Chiplets

DNNs in Extreme Throughput Applications



CERN CMS Experiment



Network Intrusion Detection

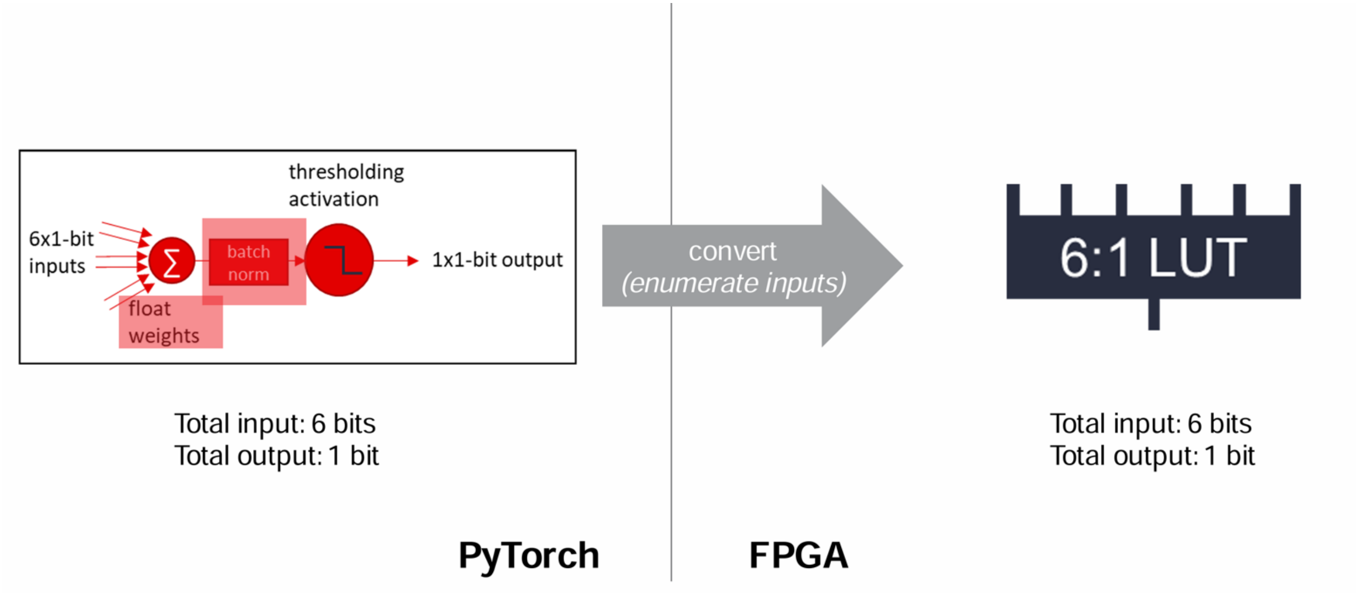
How do we mix DNNs into extreme-throughput applications?

- Need DNNs running at 100Ms of FPS, sub-microsecond latency

Slide from LogicNets presentation from AMD/Xilinx

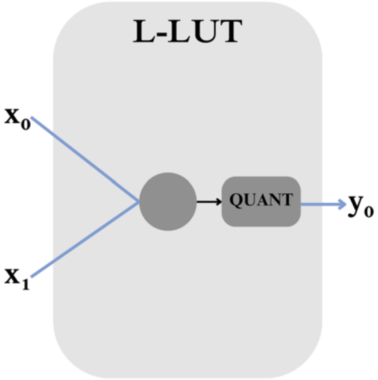
LogicNets

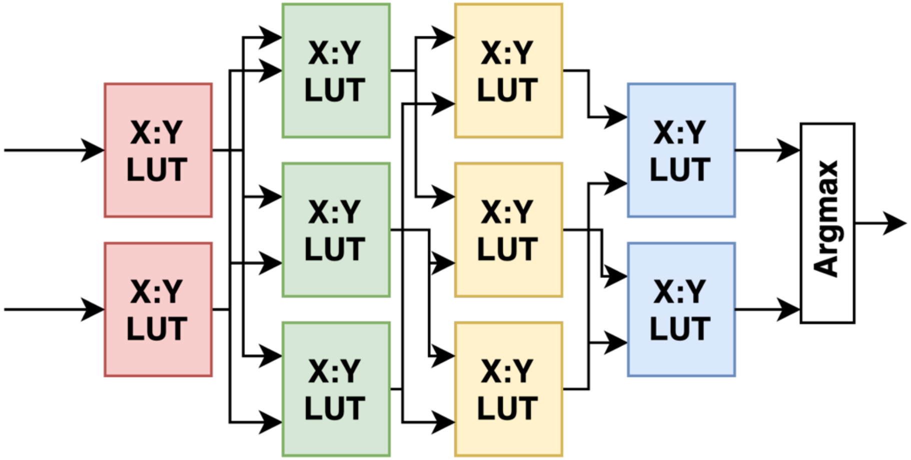
- **LogicNets (Umuroglu et al., 2020)**:
  - Trains sparse DNNs with binary inputs and activations.
  - After training: converts neurons into LUTs by going through all possible IO combinations.



LogicNets

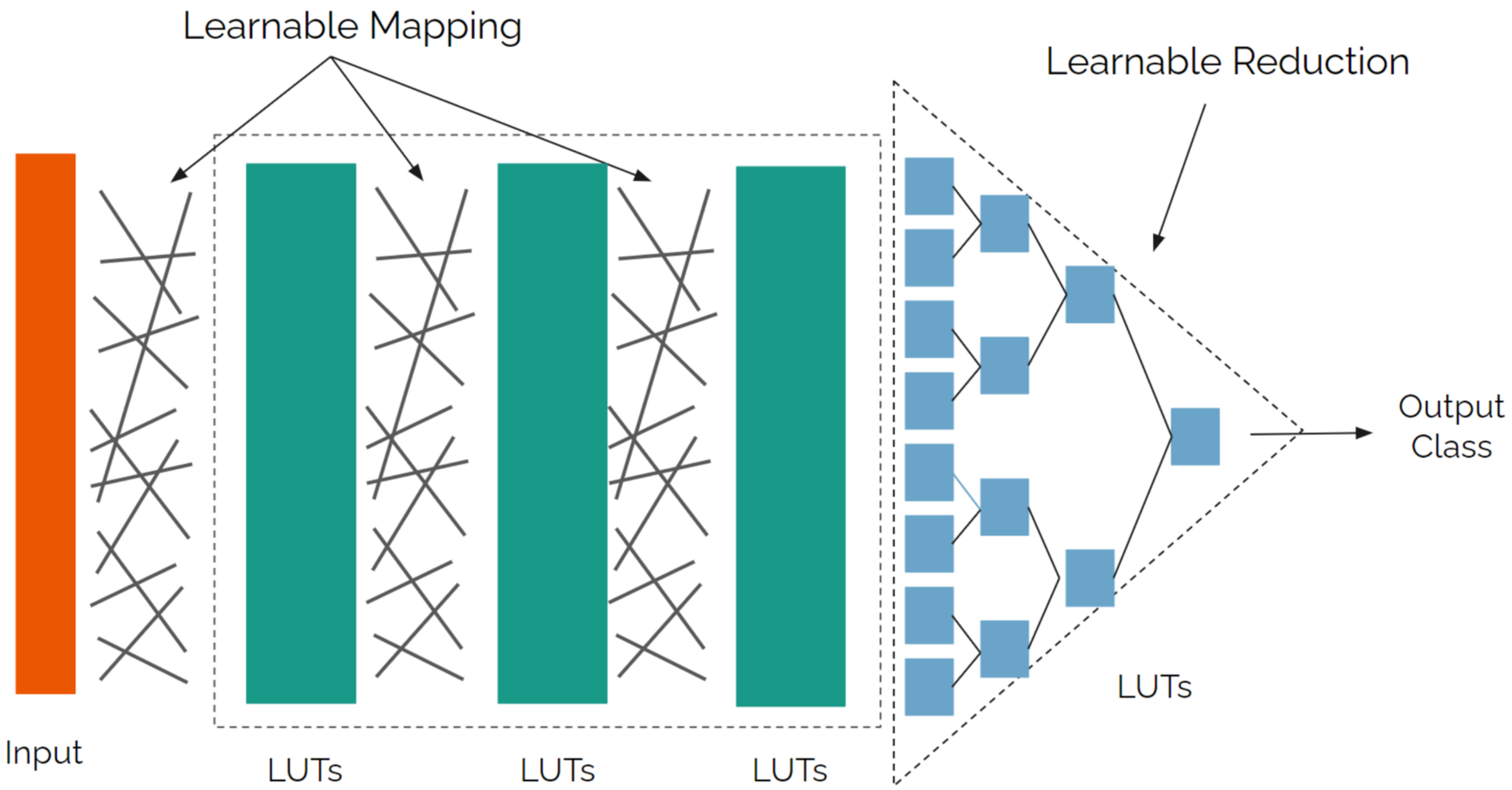
- **LogicNets (Umuroglu et al., 2020)**:
  - Trains sparse DNNs with binary inputs and activations.
  - After training: converts neurons into LUTs by going through all possible IO combinations.





DWNs

- Differentiable Weightless Neural Networks (DWN) (ICML 2024)



Reconfigurable Chiplets

What should be on the reconfigurable chiplet?

CLB Chiplets

Neural Network Chiplets

DSP Chiplets

Memory Chiplets

Reconfigurable Tensor Cores (V*V, M*V, M*M)

AI Chiplets (TensorSlices, PIMs)

Thoughts on Reconfigurable Chiplets

Memory-Heavy Chiplet Configurations

Compute-Heavy Chiplet Configurations

Fine-Grain Reconfiguration (High Overhead)

Coarse-Grain Reconfiguration (Medium Overhead)

Large-Grain Reconfiguration (Small Overhead)

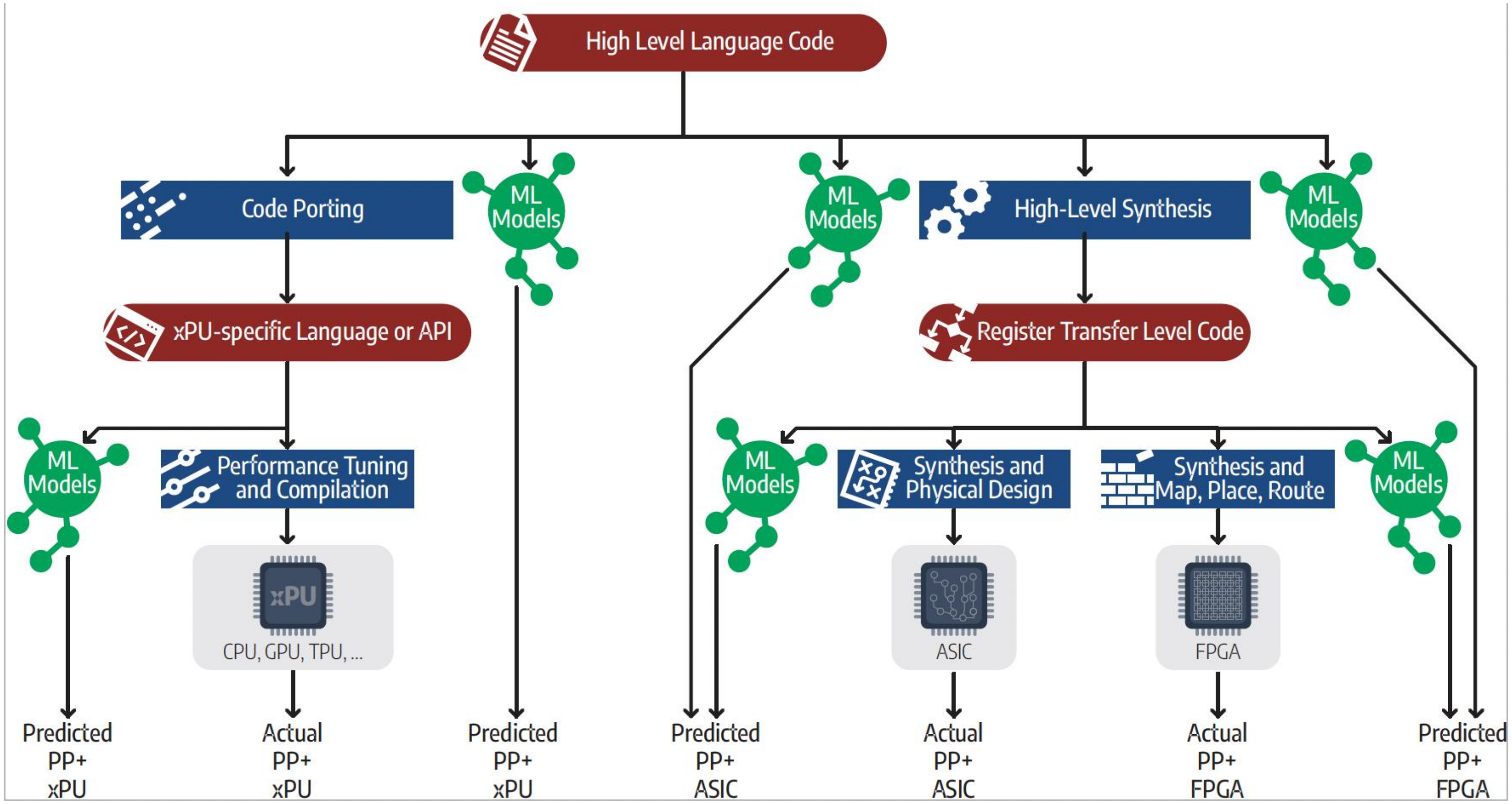
It’s all about the granularity

It’s all about the interconnect

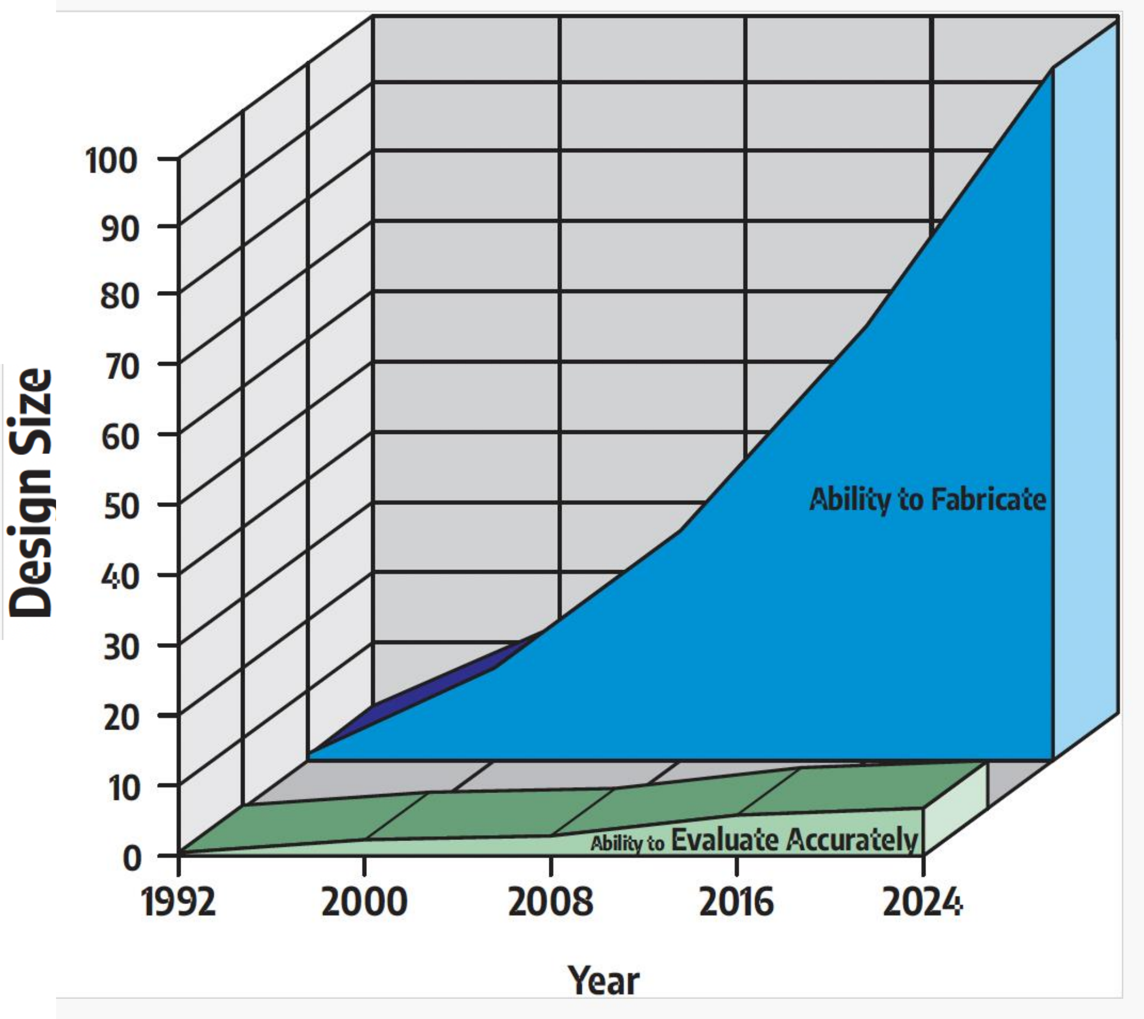
It’s all about the scale

It’s all about the mapping of applications to the heterogeneous reconfigurable substrate

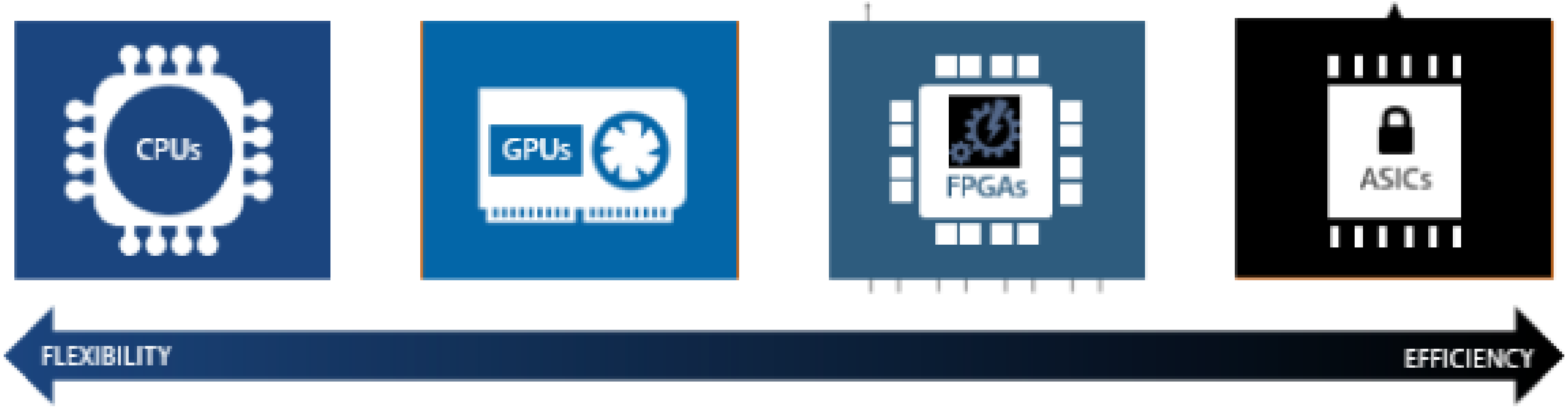
Importance of Mapping for
HPC on Reconfigurable Heterogeneous Substrate

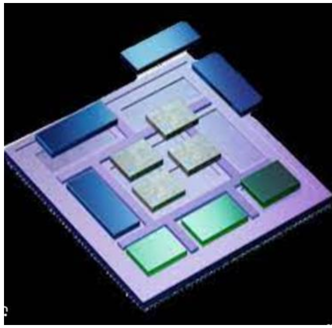


It’s all about the ability to model and evaluate



Chips and Chiplets for AI and ML and HPC





Chips will mix chiplets of CPUs, GPUs, FPGA-like blocks, ASIC-like blocks, HBMs, etc.

High Throughput, Low Power, Low Latency

Summary

Reconfigurable Large Scale Substrates for AI and ML and HPC seem viable

It will be all about the granularity

It will be all about the overheads of reconfiguration

It will be all about the interconnect

It will be all about the mapping

It will be all about the ability to model and evaluate

Hope we will make computing more energy efficient

[Source: Yoshua Bengio, ISCA TIML 2017]

- **Energy efficiency**: the brain is about **500,000 x** more energy efficient than an **Nvidia P100 GPU (ISCA 2017)**
- **13,000 x** more energy efficient than **H100**

VS

Thank You! Questions?

Laboratory for Computer Architecture (LCA)
The University of Texas at Austin
lca.ece.utexas.edu