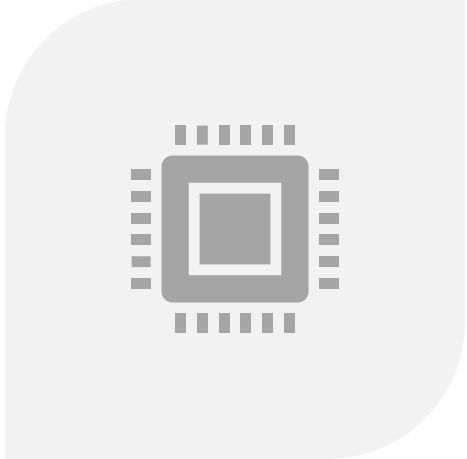
Heterogeneous Computing

Dalton Caron

Montana Technological University

January 13, 2021

EXECUTING PROGRAMS ON COMPUTING PLATFORMS WITH COMPUTING NODES OF DIFFERENT CHARACTERISTICS



HETEROGENEOUS COMPUTING ARCHITECTURES UTILIZE MORE THAN ONE KIND OF PROCESSOR OR CORE

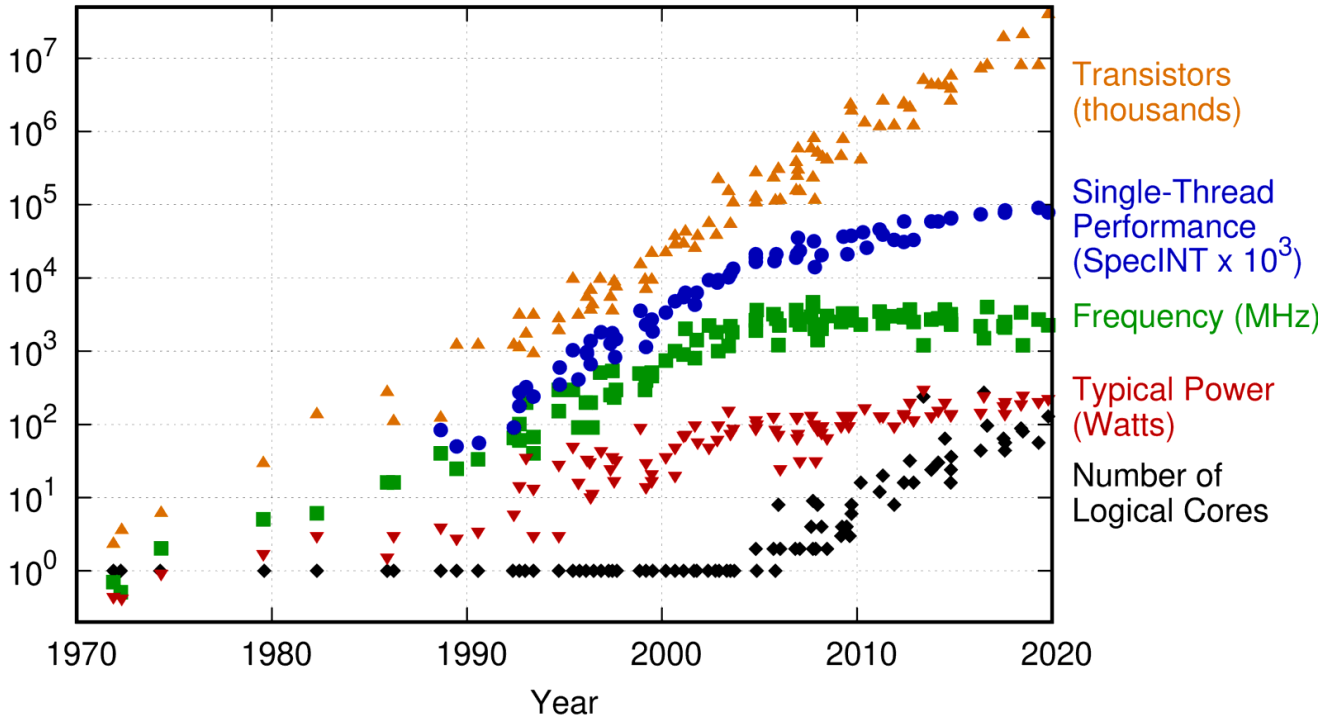
What is Heterogeneous Computing?

My GPU is dusty

Why Heterogeneous Computing?

- Gain performance through specialization at low costs
- I want more computing power

Why: Power Issue



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2019 by K. Rupp

- Power wall
- End of Dennard Scaling
- End of Moore's Law

Hardware Components

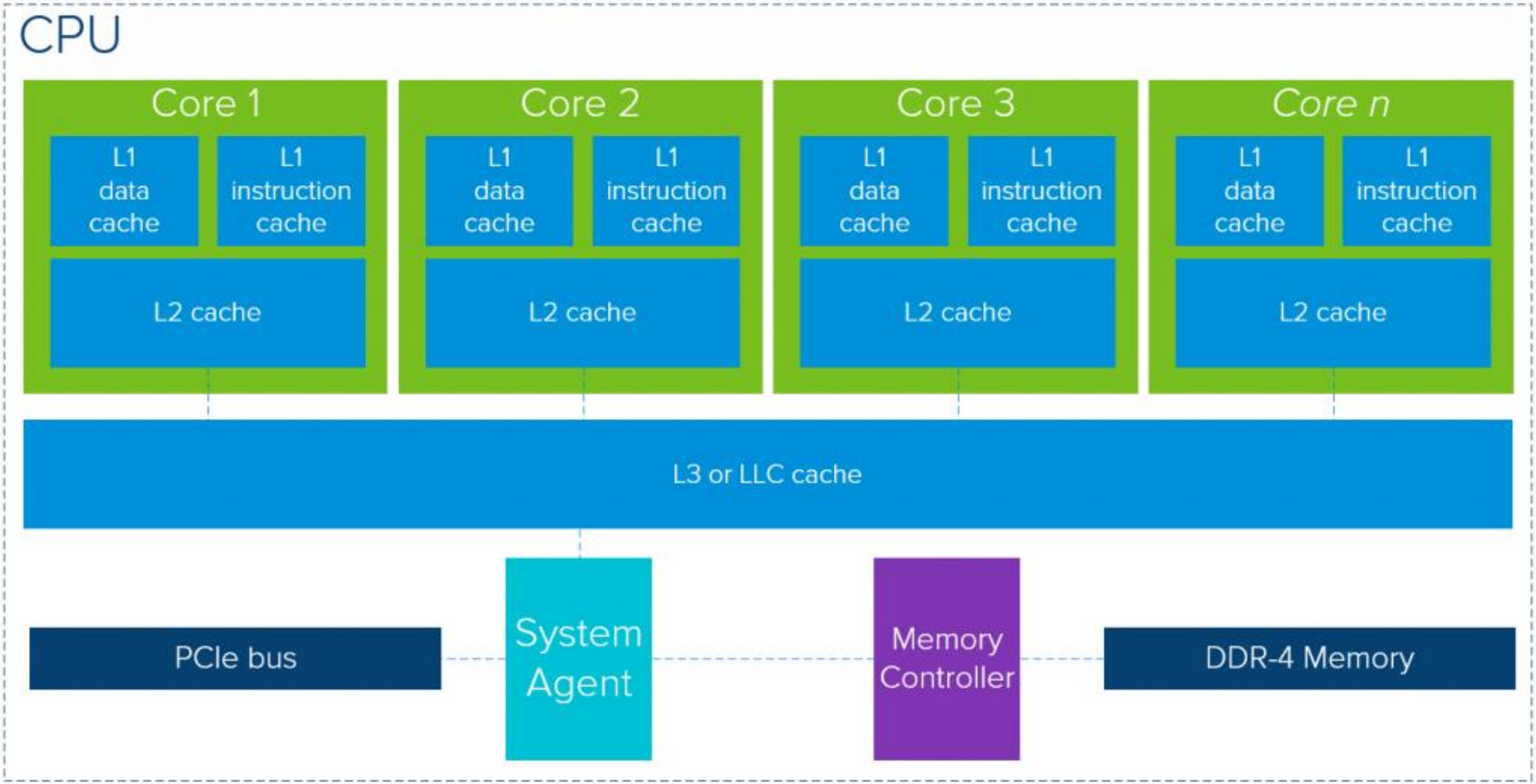
- Other
  - Processors
  - Coprocessors
  - Accelerators

Multicore

- Do we use multicore?
  - Cost
  - Core count and simultaneous multithreading
  - Architecture
  - Scaling
- Bandwidth
- General purpose high latency
- Multiple instruction-multiple data

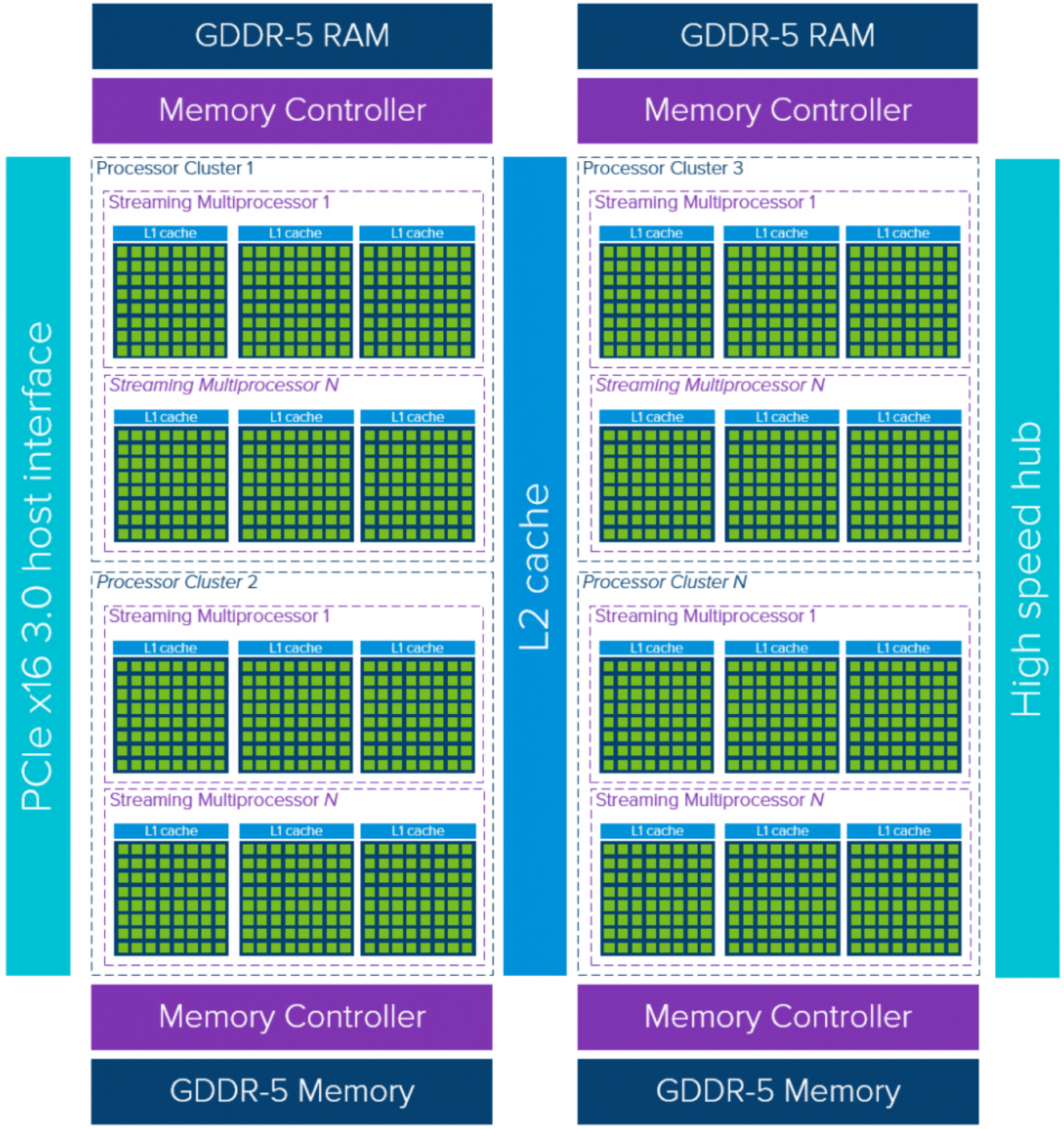
Multiprocessing vs Multithreading

- Multiprocessing
  - Multiple processes executed concurrently
  - Takes place on different processing cores than the host application
  - Improves performance by decomposiong program into parallel tasks
- Multithreading
  - Single process executed concurrently
  - Each thread runs parallel to each other
  - Threads do not allow separated memory areas to offer better performance


CPU
Core 1
L1 data cache
L1 instruction cache
L2 cache
Core 2
L1 data cache
L1 instruction cache
L2 cache
Core 3
L1 data cache
L1 instruction cache
L2 cache
Core n
L1 data cache
L1 instruction cache
L2 cache
L3 or LLC cache
PCIe bus
System Agent
Memory Controller
DDR-4 Memory

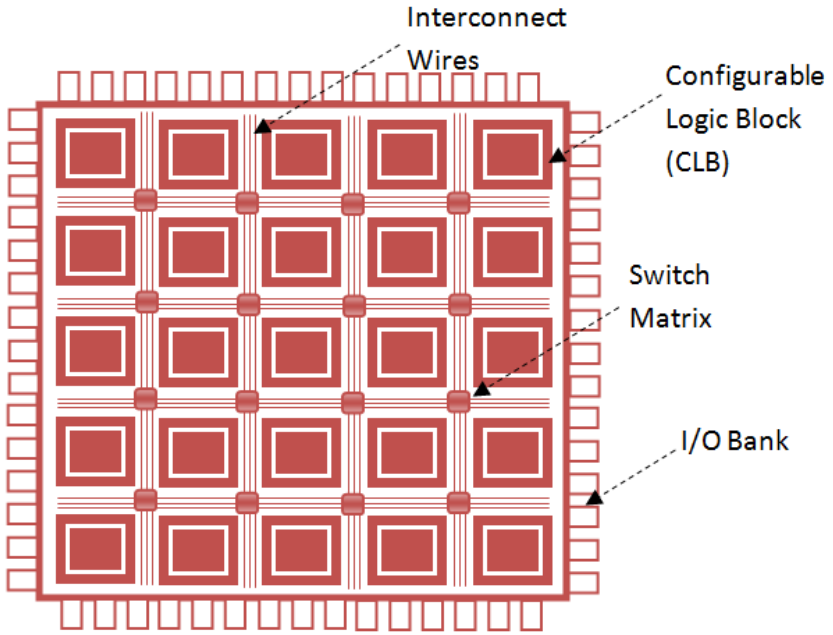
Graphical Processing Unit

- Single instruction-multiple data
- GPU vs Multicore
- Throughput orientated design


GDDR-5 RAM
GDDR-5 RAM
Memory Controller
Memory Controller
PCIe x16 3.0 host interface
Processor Cluster 1
Streaming Multiprocessor 1
L1 cache
L1 cache
L1 cache
Streaming Multiprocessor N
L1 cache
L1 cache
L1 cache
Processor Cluster 2
Streaming Multiprocessor 1
L1 cache
L1 cache
L1 cache
Streaming Multiprocessor N
L1 cache
L1 cache
L1 cache
L2 cache
Processor Cluster 3
Streaming Multiprocessor 1
L1 cache
L1 cache
L1 cache
Streaming Multiprocessor N
L1 cache
L1 cache
L1 cache
Processor Cluster N
Streaming Multiprocessor 1
L1 cache
L1 cache
L1 cache
Streaming Multiprocessor N
L1 cache
L1 cache
L1 cache
High speed hub
Memory Controller
Memory Controller
GDDR-5 Memory
GDDR-5 Memory

Field Programmable Gate Array (FPGA)

- Hardware and software are logically equivalent
- It's just gates
- Speed up functions used constantly during execution
- Dynamically configurable (hence field)

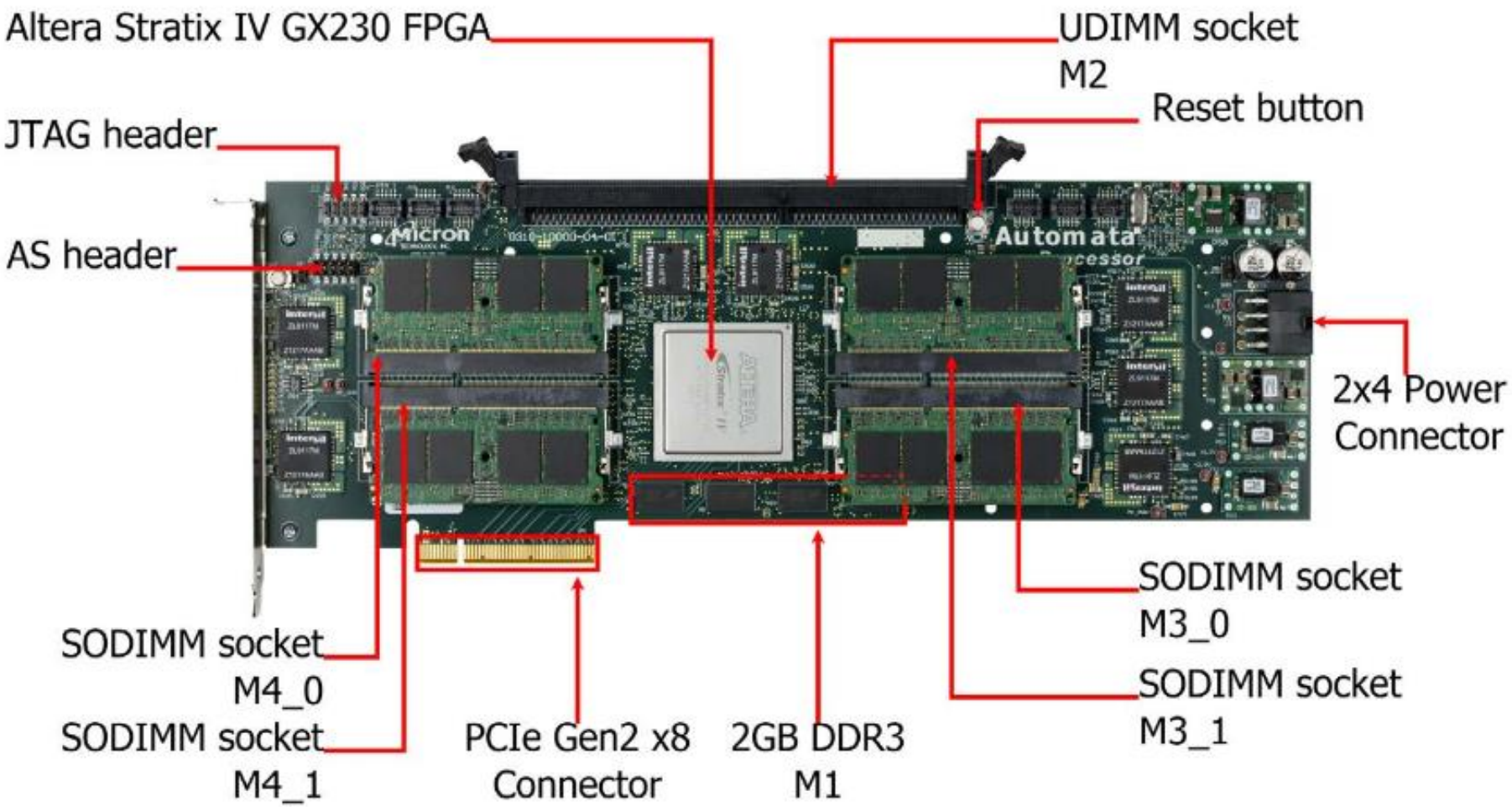


Automata Processors (AP)

- Hardware implementation of nondeterministic finite automaton
- Problems with serial implementation
  - Memory madness
  - Problem space madness
- Multiple active states => parallelism

Automata Processor Development Board

PCIe, 4 Ranks, 32 chips, 1.5M STEs



Memory Array
[256 rows by 49,152 columns]
Input Symbol
8
Input Symbol "Row Decoder"
(8-bits to 2^8 Rows)
Row Enable (2^8-1)
(common)
Row Enable (2^8-2)
(common)
Row Enable (2^8-3)
(common)
Row Enable (2)
(common)
Row Enable (1)
(common)
Row Enable (0)
(common)
State Transition Clock
(common)
STE (0)
STE (1)
STE (2)
STE (3)
STE (4)
STE (y-4)
STE (y-3)
STE (y-2)
STE (y-1)
STE (y)
Mb (2^8-1)
Mb (2^8-2)
Mb (2^8-3)
Mb (2)
Mb (1)
Mb (0)
Logic
Memory Column Bits (1 X 256)
STE(0) Inputs
16
STE(0) Output
Automata Routing Matrix Structure

AP Applications

- Efficient execution regular expressions
- DNA/protein motif search problem (find AGATAA with 2 mutations)
- Association rule mining
- Part of Speech Tagging
  - AKA Brill tagging

Planted Motif Search Problem	Automata Processor	UCONN - BECAT Hornet Cluster
Processors	48 (PCIe Board)+CPU	48 CPU (Cluster/OpenMPI)
Power	245W-315W[1]	>2,000W[1]
Cost	TBD	~$20,000[1]
Performance (25,10)	12.26 minutes[2]	20.5 minutes
Performance (26,11)	13.96 minutes[2]	46.9 hours
Performance (36,16)	36.22 minutes[2]	Unsolved

- Planted Motif Search problem is a leading problem in bioinformatics and is NP Hard. Attempts to find common genomic sequences in noisy data.
- Solutions involving high match lengths and substitution counts are often presented to HPC clusters for processing.
- Independent research predicts the Micron Automata Processor significantly outperforms a multi-core HPC cluster in speed, power and estimated cost.

[1] Micron Technology Estimates, Not including Memory of 4GB DRAM /Core
[2] Research conducted by Georgia Tech (Roy/Aluru)

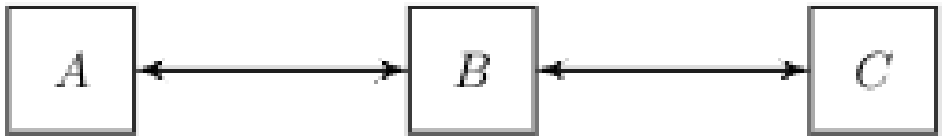
Others

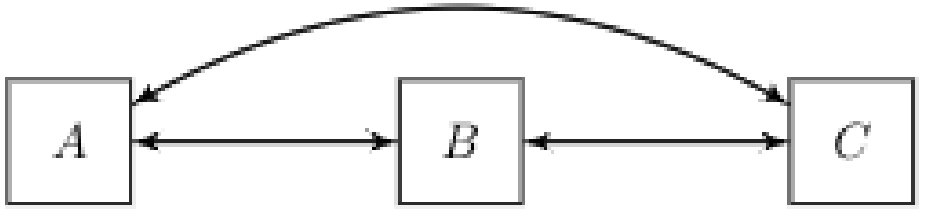
- Neuromorphic chips
- Digital signal processors (DSP)
- Quantum computing (QC)
- Tensor processing units (TPU)

Interconnection

Interconnection: Topology

- Shape of interconnection
- Regular vs irregular shapes
  - Diameter





Granularity

- At what level are components connected?
- Core, Processor, Computer, Blade, Cabinet, ...

Interconnection: Technology

- How to connect two entities together?
- A link is two parallel wires
  - Mutliple links => parallel data transfer
- Bandwidth vs power tradeoff
- Wire delay
- Optical links
  - Less power consumption
  - Arbitrary interconnection
  - Optical broadcast networks

Interconnection: Protocol

- What protocols are used to transfer data over links?
- Hyper Transport
  - Connects CPU to IO or CPU to CPU
  - MABB 51.2 GB/s (v3.1)
- Peripheral Component Interface Express (PCIe)
  - Packet based protocol with up to 32 lanes
  - Bandwidth 4 GB/s (v5.0) per lane
- NVLink
  - Developed by NVIDIA for high-bandwidth requirement of GPUs
  - Bandwidth 300 GB/s (v2.0)

Bandwidth

- Theoretical maximum amount of memory the bus can handle
- Bandwidth is a performance bottleneck and big challenge

Programming Models

The Needs of a Programmer

- Productivity: Ability to finish on time given performance, reliability, and cost requirements
- Portability: Be able to deploy on machines of different characteristics
- The program should look homogeneous
- Runtime must have low overhead

What We Got

- MPI
- OpenMP
- OpenCL
- Heterogeneous System Architecture (HSA)
- CUDA
- Others

MPI

- Defacto programming model for distributed memory architectures
- Distributed memory model
- C++/C library (NO JAVA OR PYTHON)
- Sending messages is costly
- Scatter and gather operations

OpenMP

- Shared memory model
- Notation of private threads
- Now supports other accelerators
- Automatic parallelization option

OpenCL (not GL)

- Maximize portability
- Use all computation resources
- Drive future hardware requirements
- Abstract underlying parallelism

Heterogeneous System Architecture (HSA)

- Main goal is portability
- Not a programming language
- Intermediate representation: HSA Intermediate Language (HSAIL)
- HSAIL converted into target instruction set architectures

CUDA

- Mature GPGPU language
- OpenGL and Direct3D are for graphics!
- Supports OpenCL
- CUDA is like OpenCL

Research Front of Heterogeneous Computing

Processing-in-Memory (PIM) or Near-Data-Processing (NDP)

- What type of processing do we need in memory?
- How is virtual memory affected?
- How are programming models affected?
- Are PIM/NDP machines reliable?
- Does PIM/NDP introduce new security risks?

Exascale Computing Systems

- How do we reduce power consumption?
- Does the programming model need to change?
  - Large amount of concurrency possible
  - Theoretical high capacity and data rates
- Can operating systems scale to these machines?
- Do current algorithms scale to these machines?

Local Research Activities

- Construction of a heterogeneous computing system on campus
  - CPU, GPU, FPGA, and DSP capabilities
- Automatic parallelization of code to different hardware components
  - Compiler based design

Conclusion

References

- M. Zahran, *Heterogeneous Computing*. Morgan & Claypool, 2019.
- D. B. Kirk and W.- mei W. Hwu, *Programming Massively Parallel Processors*. Morgan Kaufmann, 2016.
- D. A. Patterson and J. L. Hennessy, *Computer Organization and Design MIPS Edition*. 5th ed. Morgan Kaufmann, 2014.
- K. Rupp. “Microprocessor Trend Data.” Github. https://github.com/karlrupp/microprocessor-trend-data.
- W. Ke. “An Overview of Micron’s Automata Processor.” University of Virginia CS. https://www.cs.virginia.edu/~skadron/Papers/wang_APoverview_CODES16.pdf.

References

- M. Baboli. “A Comprehensive Evaluation of Direct and Indirect Network-On-Chip Topologies .” Jan. 2014. http://ieomsociety.org/ieom2014/pdfs/451.pdf.
- D. Novillo, “OpenMP and automatic parallelization in GCC.” Red Hat Canada, [Online]. Available: https://www.airs.com/dnovillo/Papers/gcc2006.pdf.
- J. Miller. “On-Chip Optical Communication for Multicore Processors.” MIT Photonics. https://mphotonics.mit.edu/microphotonics-center/meeting-presentations-restricted/spring-2009/713-on-chip-optical-communication-for-multicore-processors/file.

References

- D. Panda. “Programming Models for Exascale Systems.” Ohio State CS. https://mvapich.cse.ohio-state.edu/static/media/publications/slide/hpcac_lugano14_pmodels.pdf.
- “In-Memory Processing.” ScienceDirect. https://www.sciencedirect.com/topics/computer-science/in-memory-processing.