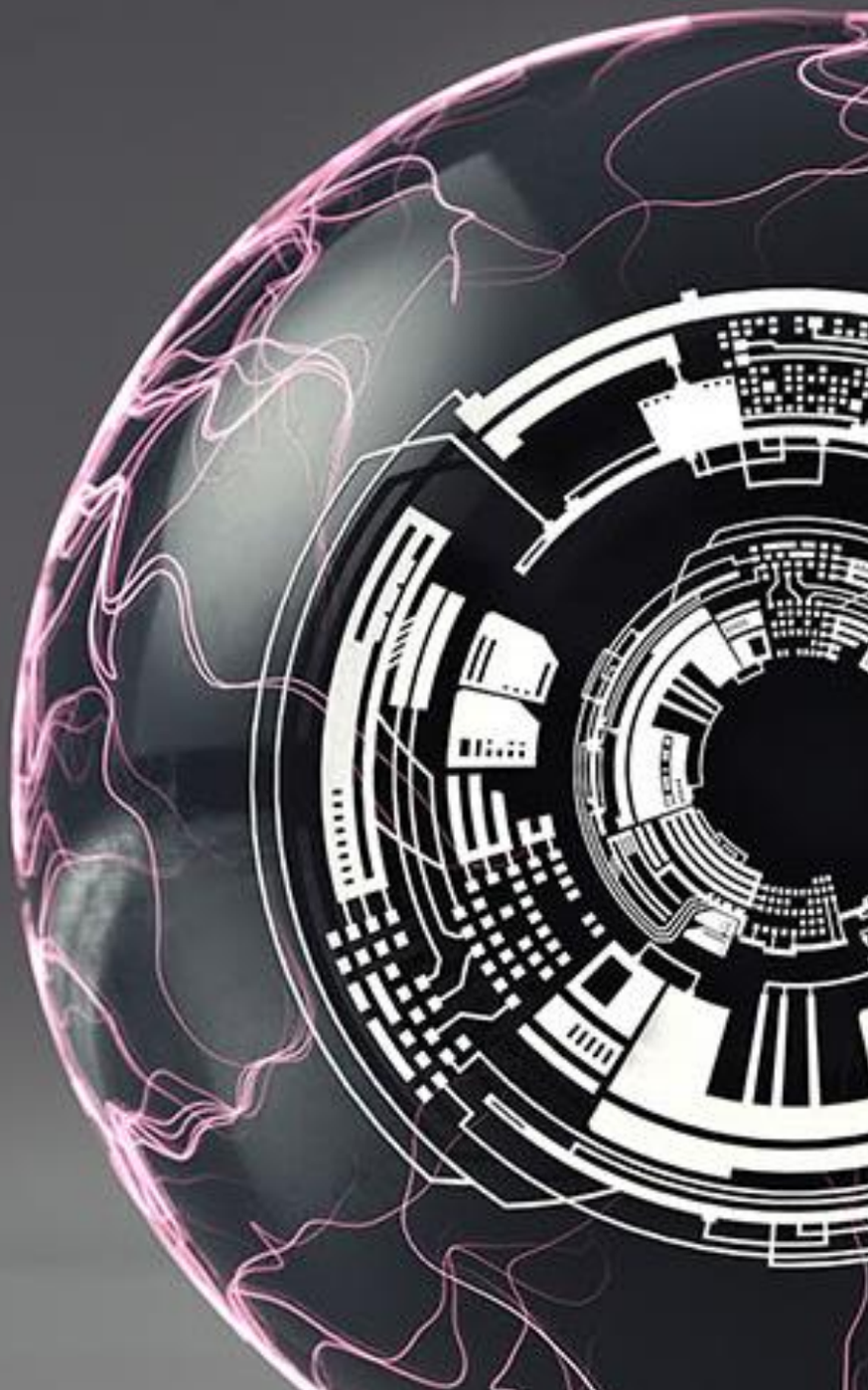

Microsoft
Microsoft Research
Faculty
Summit
2016

Microsoft

Computing with Programmable Logic

Jan Gray
Gray Research LLC
jsgray@acm.org | @jangray | http://fpga.org



Outline

- Introduction
- FPGA 101
- Applications
- Tools
- ‘Software-first’ accelerator design

Computing with programmable logic

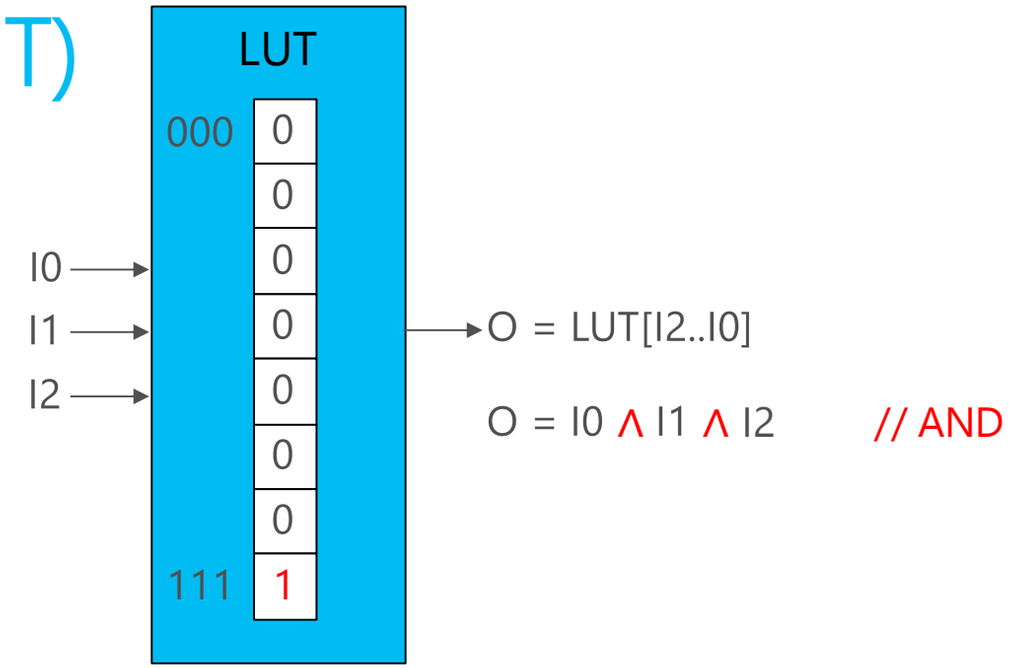
- The autumn of Moore's Law
  - Continued compute scaling → ↓↓↓energy/op → custom hardware
- FPGAs as computer accelerators
  - Massively parallel, specialized, connected, versatile
  - High throughput, low latency, energy efficient
- Great! How do I program one (or a cloud of them)?

What is a Field Programmable Gate Array?

A universal digital logic chip – *build whatever you like*

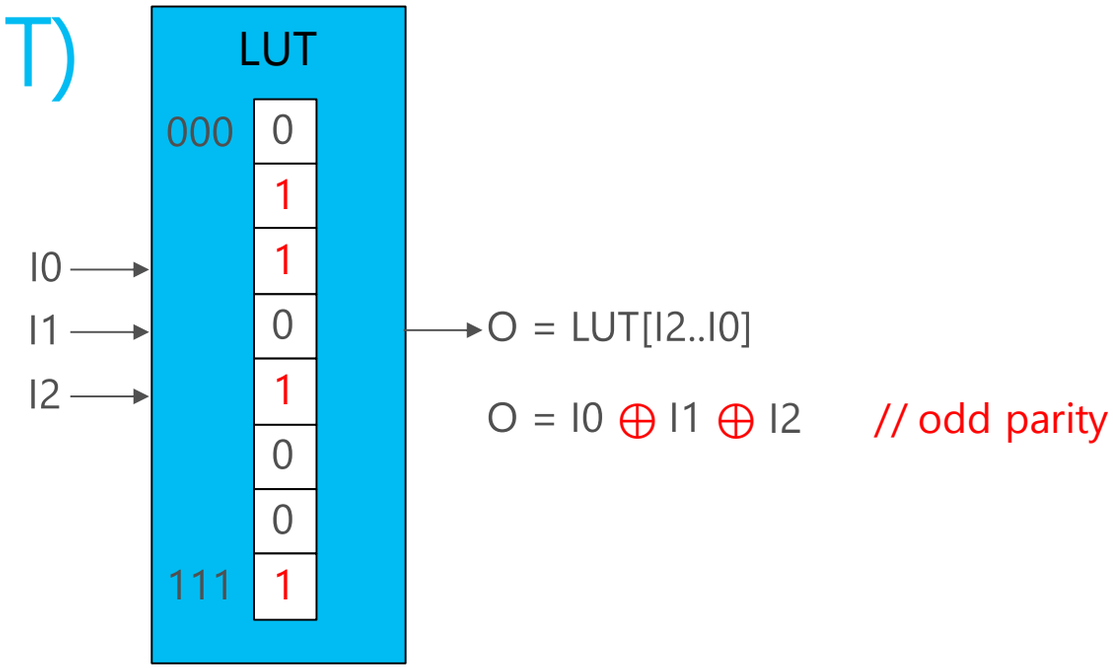
Programmable logic elements

- "Gate": lookup table (LUT)
  - N inputs → 2^N entries



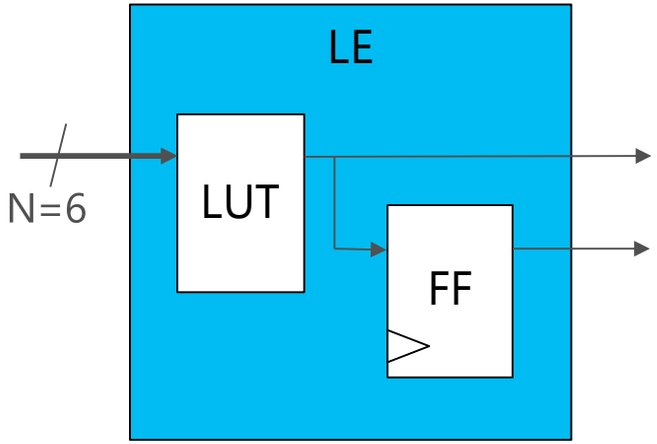
Programmable logic elements

- "Gate": lookup table (LUT)
  - N inputs → 2^N entries



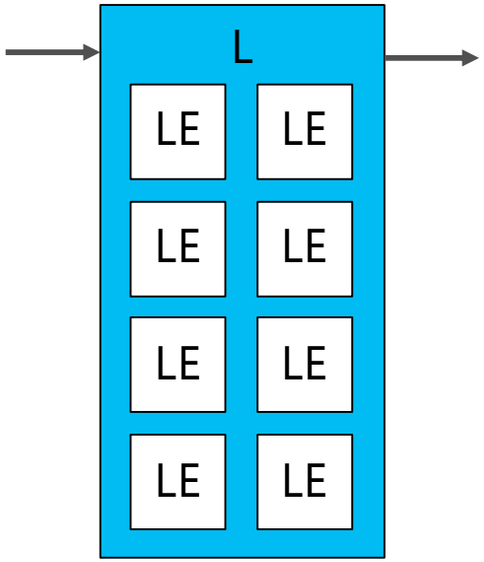
Programmable logic elements

- Logic element (LE)
  - 6-LUT – combinational
  - Flip-flop – synchronous



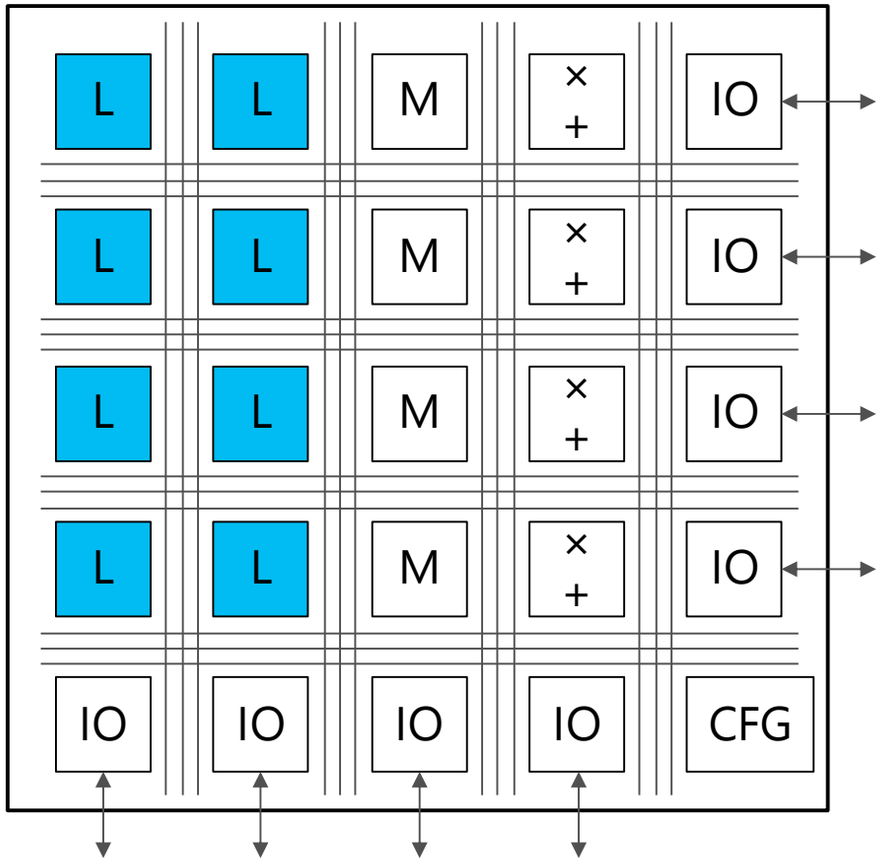
Programmable logic elements

- Configurable logic block
  - Cluster of logic elements
  - Local routing
- For example
  - Up to eight 6-input logic gates (+ 8-bit register)
  - 8-bit ALU
  - 64×8b SRAM



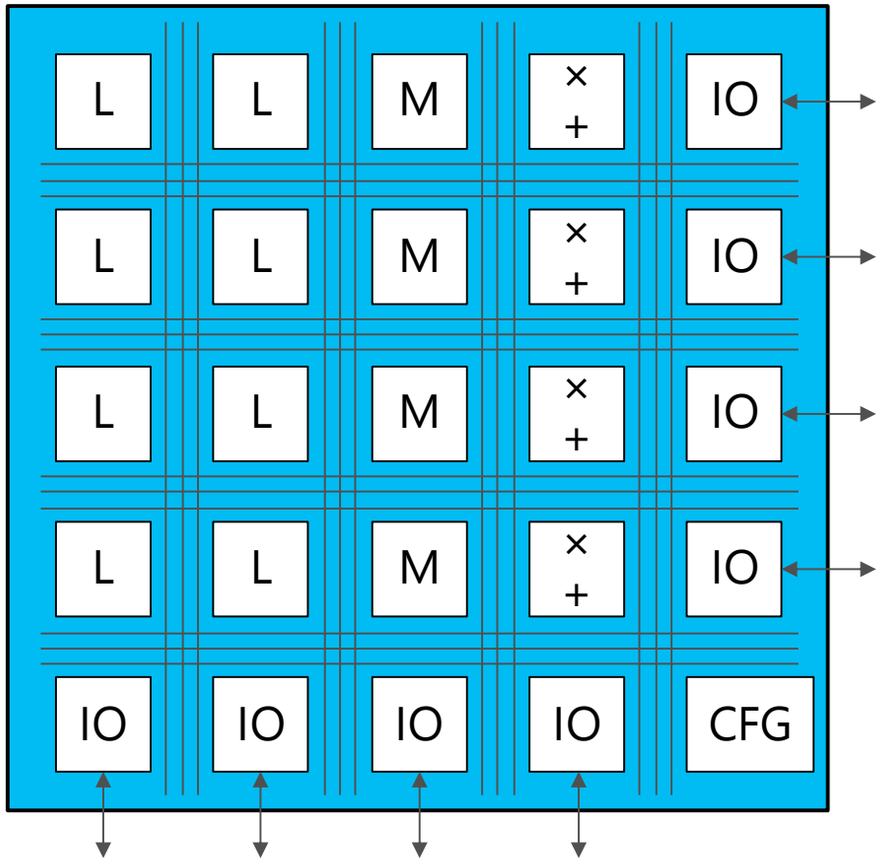
An FPGA is a universal digital logic chip

- Configurable logic blocks



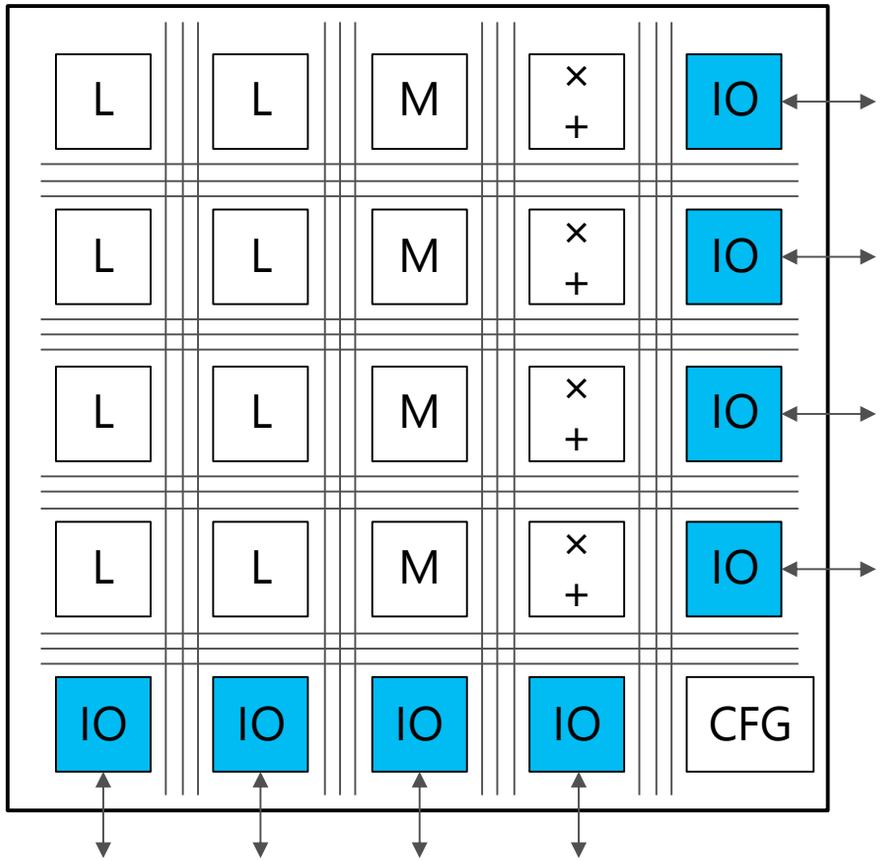
An FPGA is a universal digital logic chip

- Configurable logic blocks
- Interconnect fabric



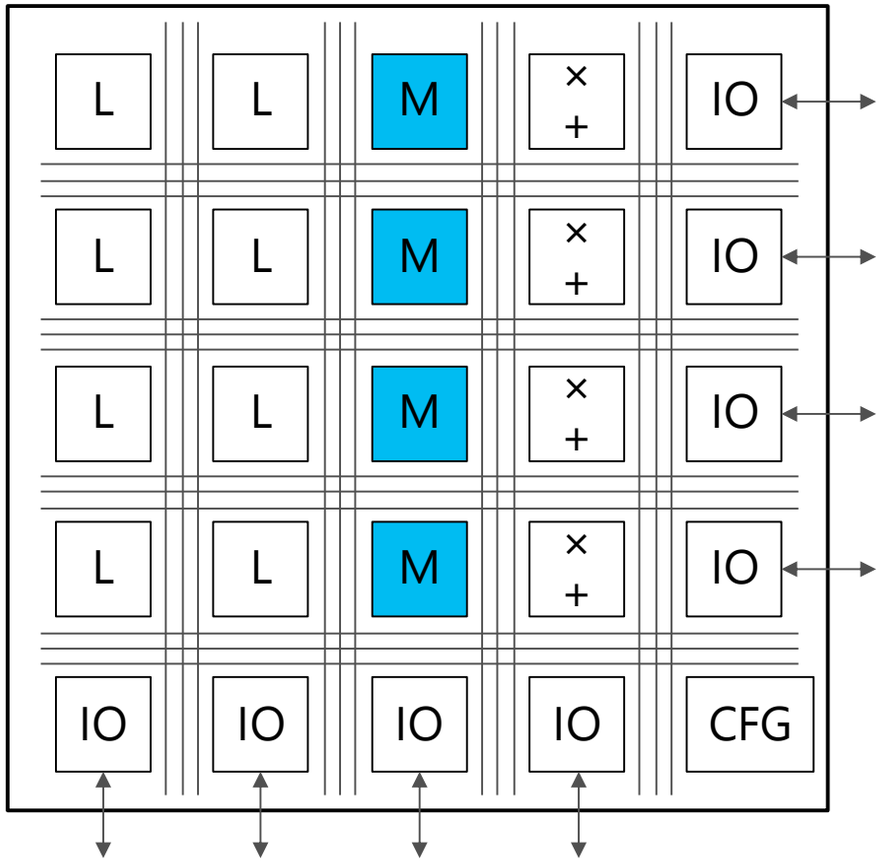
An FPGA is a universal digital logic chip

- Configurable logic blocks
- Interconnect fabric
- I/O blocks



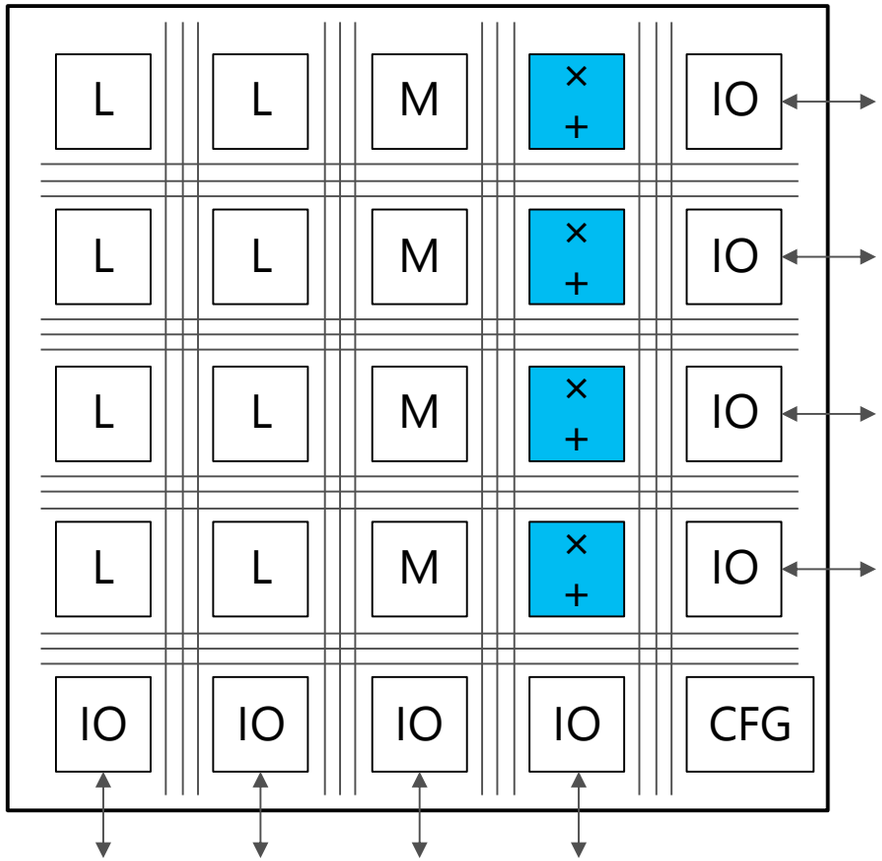
An FPGA is a universal digital logic chip

- Configurable logic blocks
- Interconnect fabric
- I/O blocks
- Fixed logic blocks
  - BRAM: dual-port 1K×36b



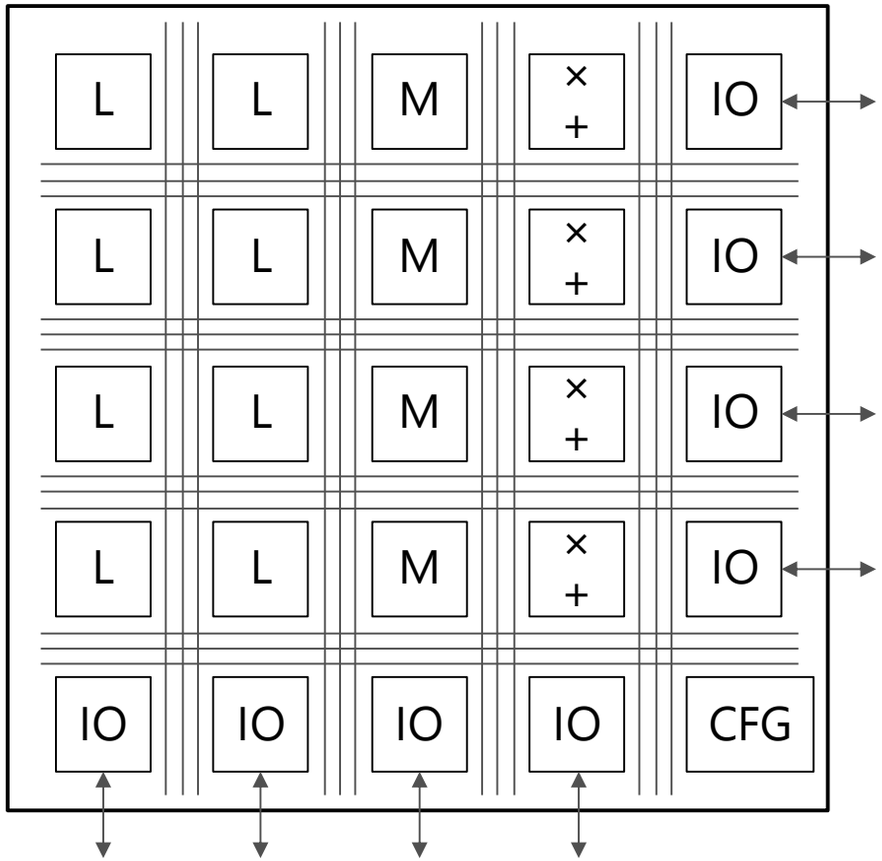
An FPGA is a universal digital logic chip

- Configurable logic blocks
- Interconnect fabric
- I/O blocks
- Fixed logic blocks
  - BRAM: dual-port 1K×36b
  - DSP: fixed/floating multiply-add



An FPGA is a universal digital logic chip

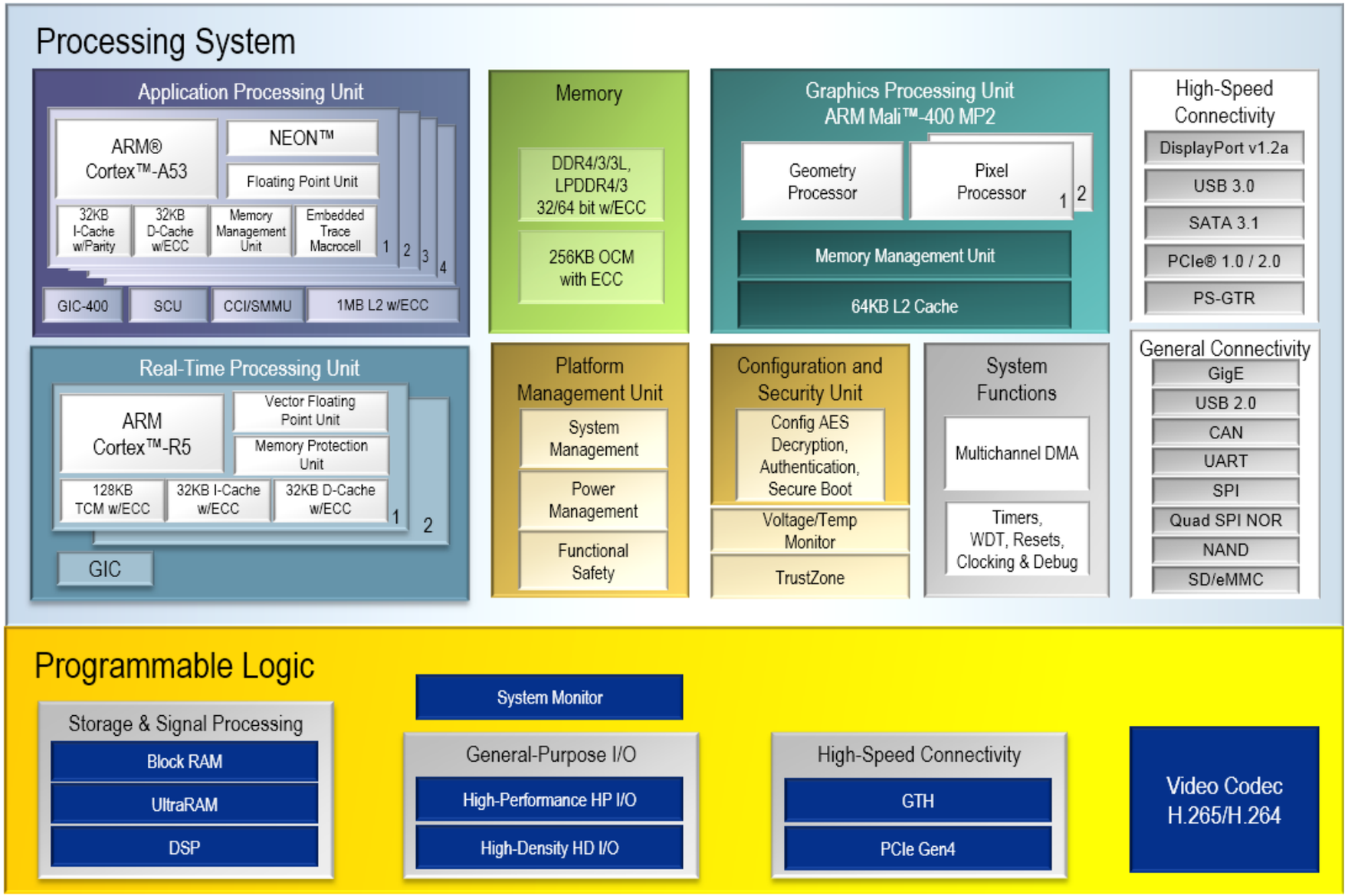
- Configurable logic blocks
- Interconnect fabric
- I/O blocks
- Fixed logic blocks
  - BRAM: dual-port 1K×36b
  - DSP: fixed/floating multiply-add
  - Much more



An FPGA is a **heterogeneous** system on a chip

- ARM SoC
  - 4 ARM-A53 (64b)
  - 2 ARM-R5
  - GPU
  - Memory
  - …

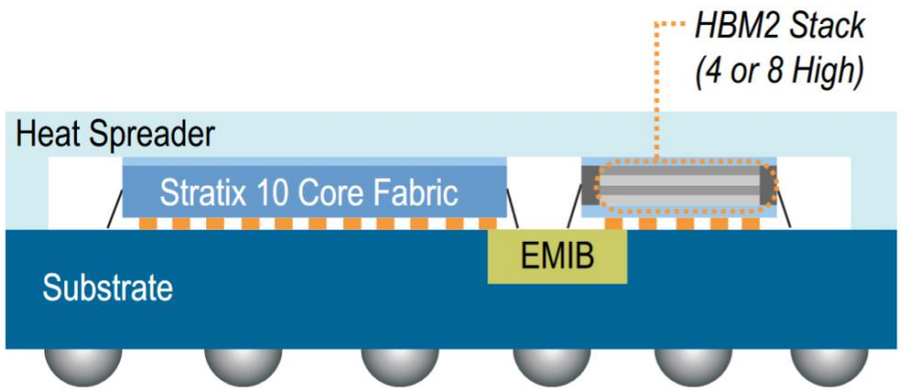
+ FPGA fabric

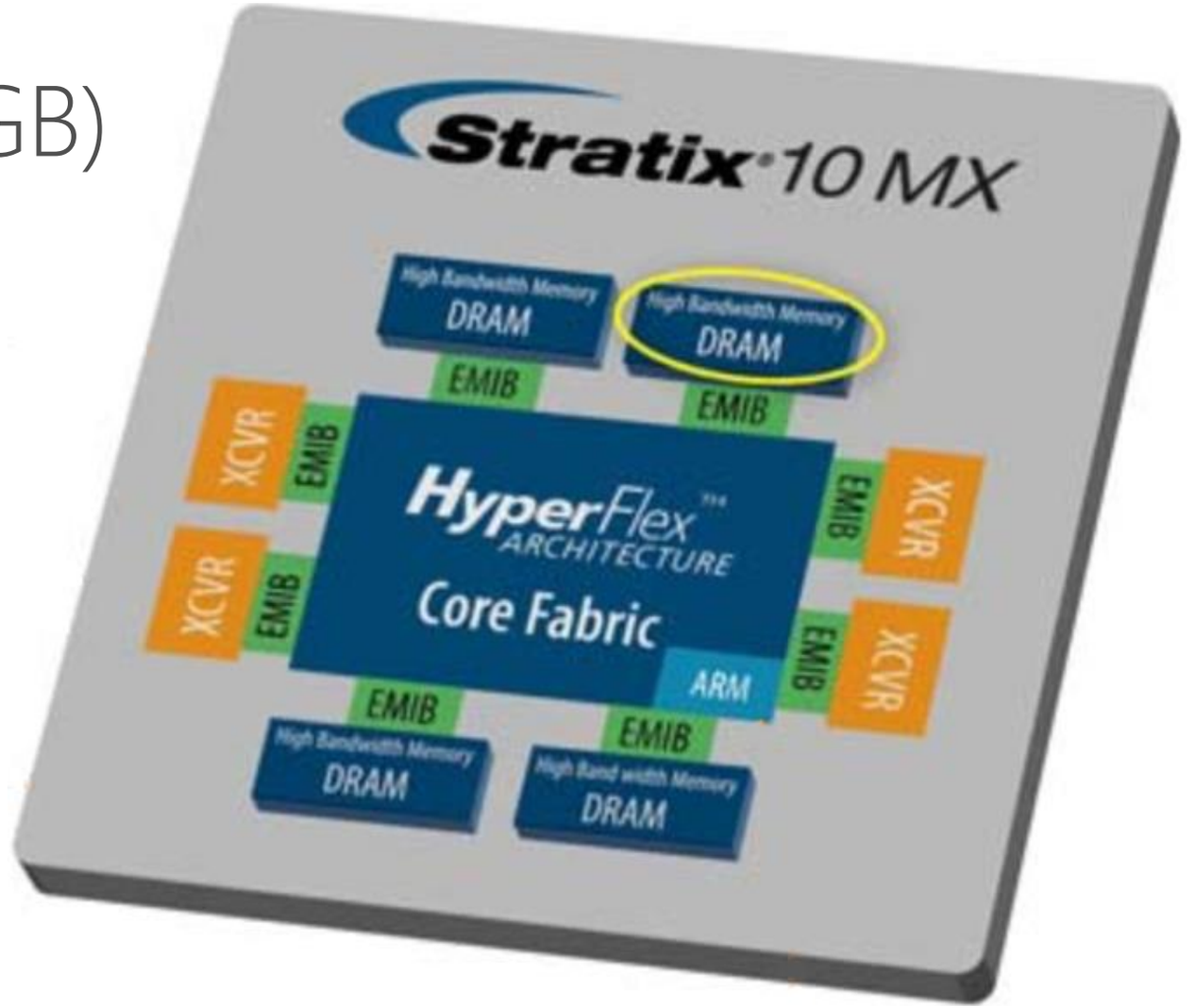


http://www.xilinx.com/content/dam/xilinx/imgs/products/zynq/zynq-ev-block.PNG

An FPGA is a heterogeneous system in package

ARM-SoC-FPGA
+ 4 HBM DRAM stacks (16 GB)





https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/wp/wp-01264-stratix10mx-devices-solve-memory-bandwidth-challenge.pdf

Modern FPGAs are enormous …

- ~2M LUTs, 4M flip-flops
- 50 Mb LUT-RAM + 400 Mb BRAM
- 10,000 DSP blocks
- 12×100G Ethernet
- 128×32Gbps transceivers

(From various 'largest announced' Xilinx and Altera FPGAs)

… with massive parallelism and bandwidth

- 10^{15} bit-op/s
- 10 TFLOPS
- 5000 BRAM + 100K LUT-RAM accesses *per cycle*
- 50 TB/s local RAM bandwidth
- 1 TB/s HBM DRAM bandwidth
- 4000 Gb/s network bandwidth

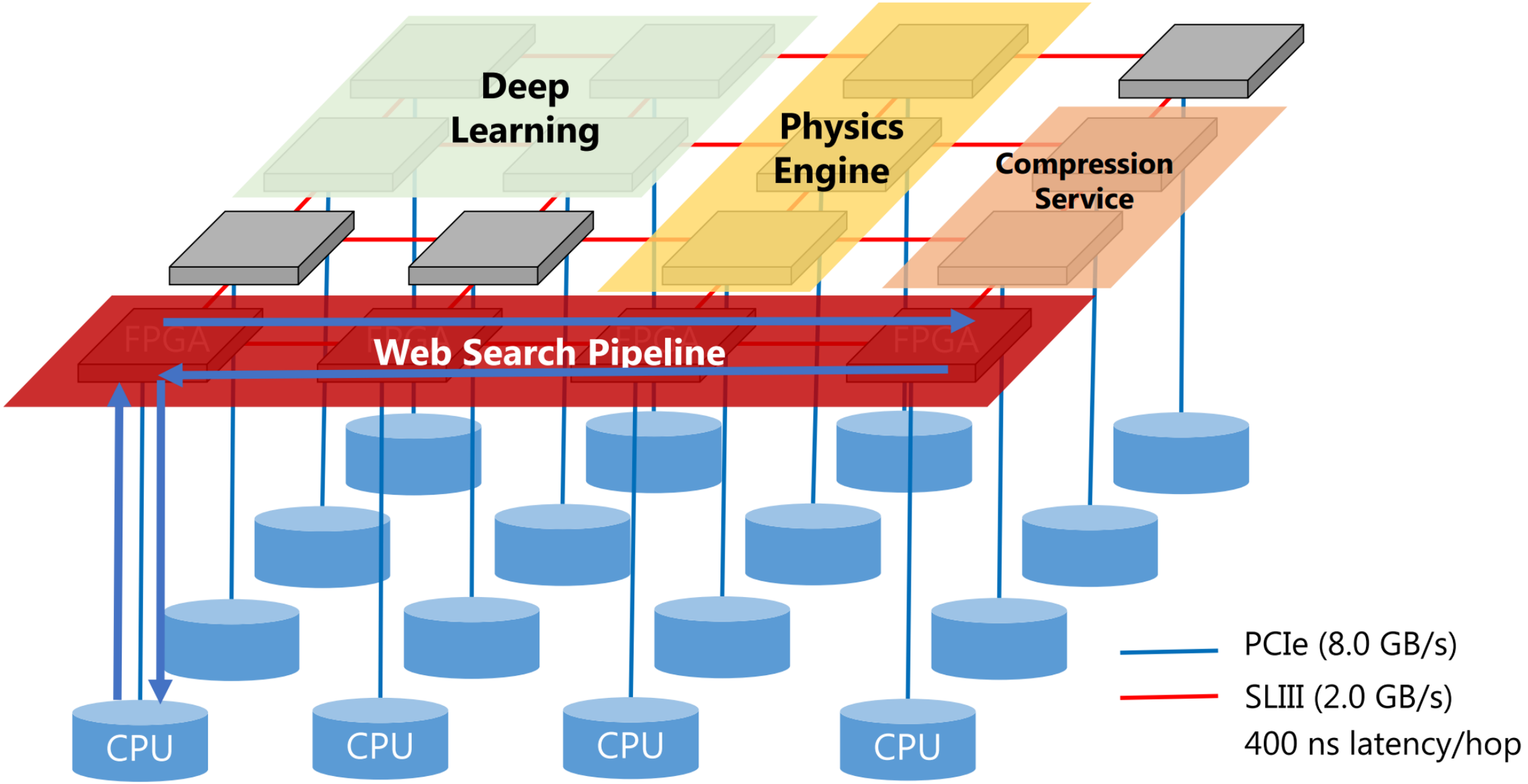
Applications

Xilinx: “next generation smarter systems”

- Cloud computing
  - Deep learning
  - Image and speech recognition
  - Big data analytics
  - Bioinformatics / sequencing
  - Interconnect and storage
- Embedded vision
  - ADAS
  - Machine vision
  - 4K/8K display and transport
  - Medical imaging
  - Surveillance
- Industrial IOT
  - Motion and motor control
  - Predictive maintenance
  - Smart grid / energy
  - Smart medical
  - Real time processing and analytics
- SDN, NFV, programmable data plane
- 5G wireless

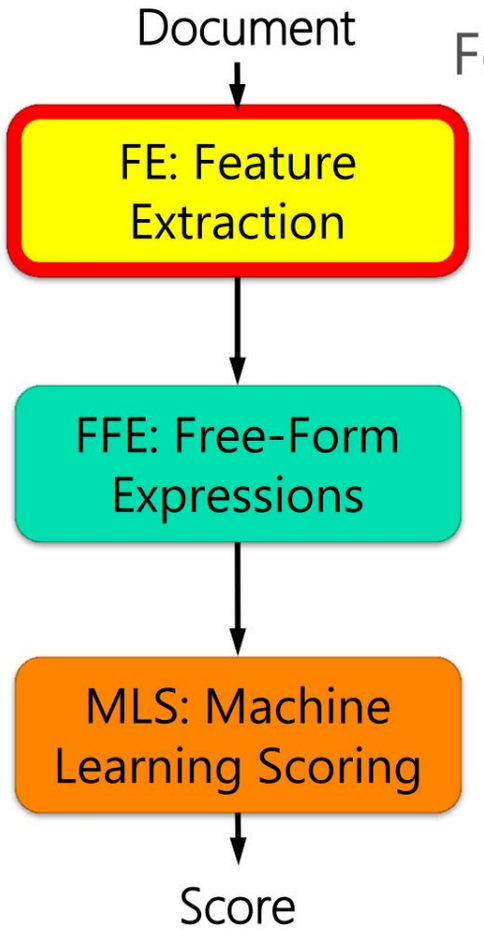
[http://www.xilinx.com/applications/megatrends.html]

Catapult: An Elastic Reconfigurable Fabric for Datacenters



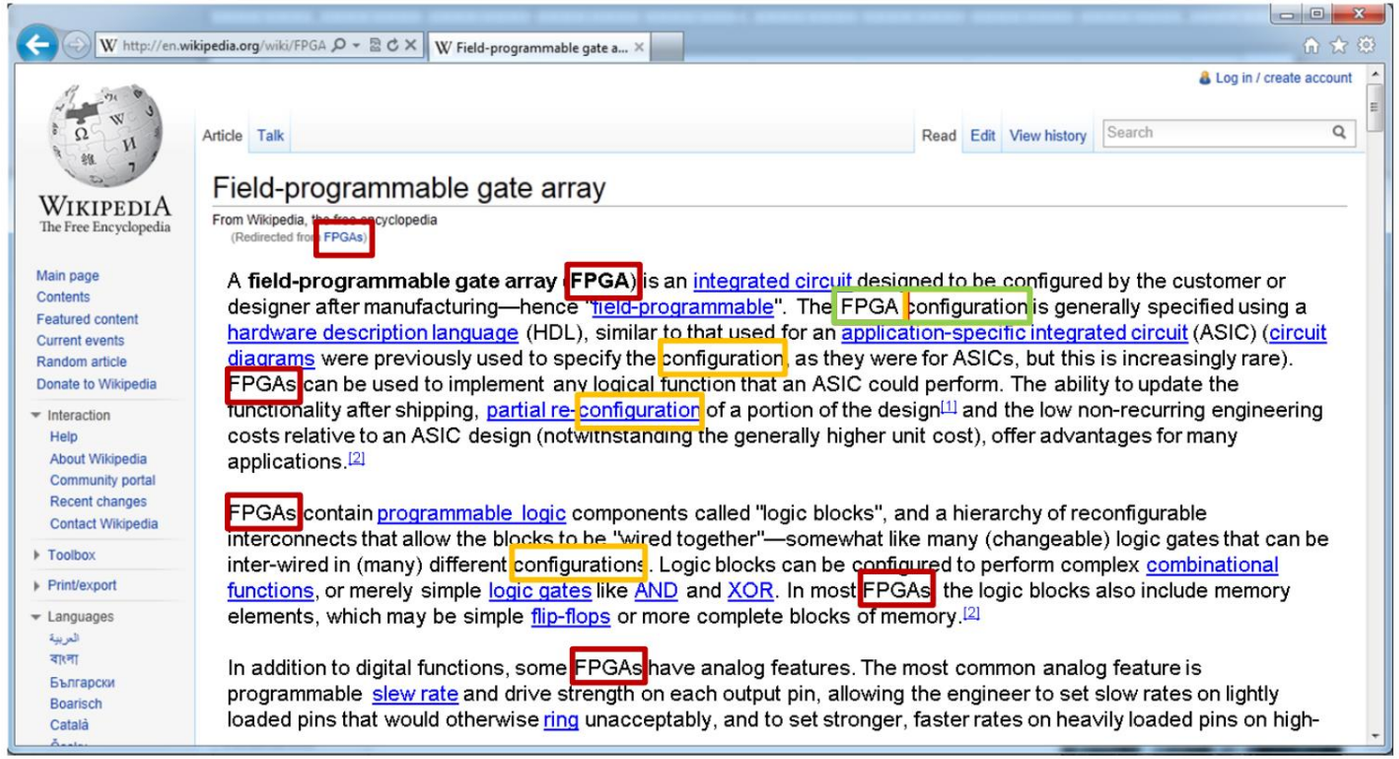
[Andrew Putnam, Doug Burger, et al, MSR, ISCA-41, 2014] [https://www.microsoft.com/en-us/research/project/project-catapult/]

Catapult Bing search query ranking

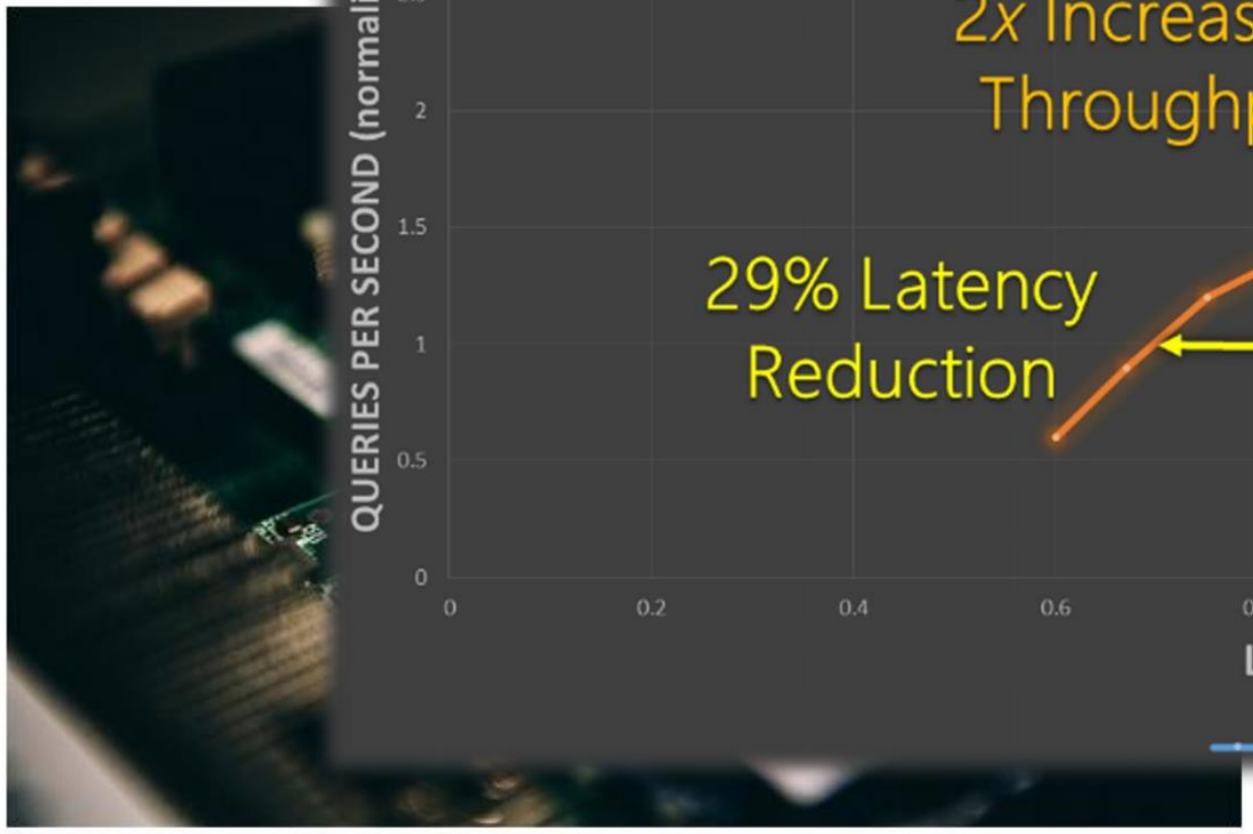


Query: "FPGA Configuration"

Features: **NumberOfOccurrences_0 = 7** | **NumberOfOccurrences_1 = 4** | **NumberOfTuples_0_1 = 1**

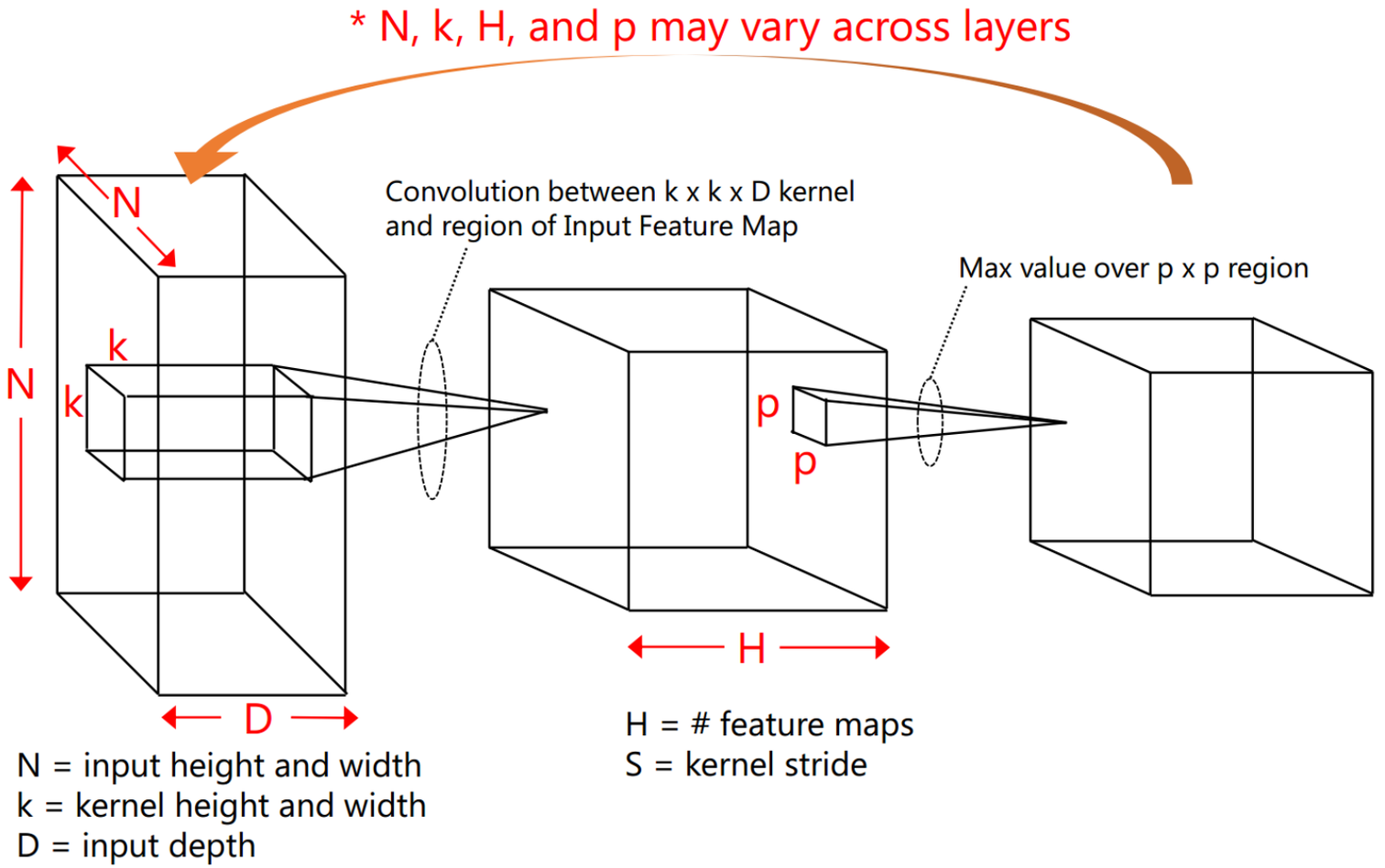


MICROSOFT SUPERCHARGES BING SEARCH WITH PROGRA



http://www.wired.com/2014/06/microsoft-fpga/

3-D Convolution and Max Pooling

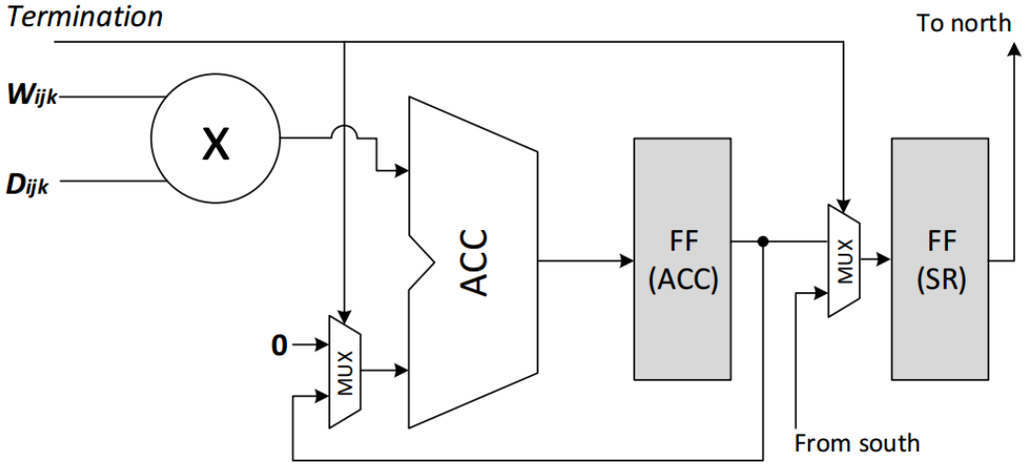


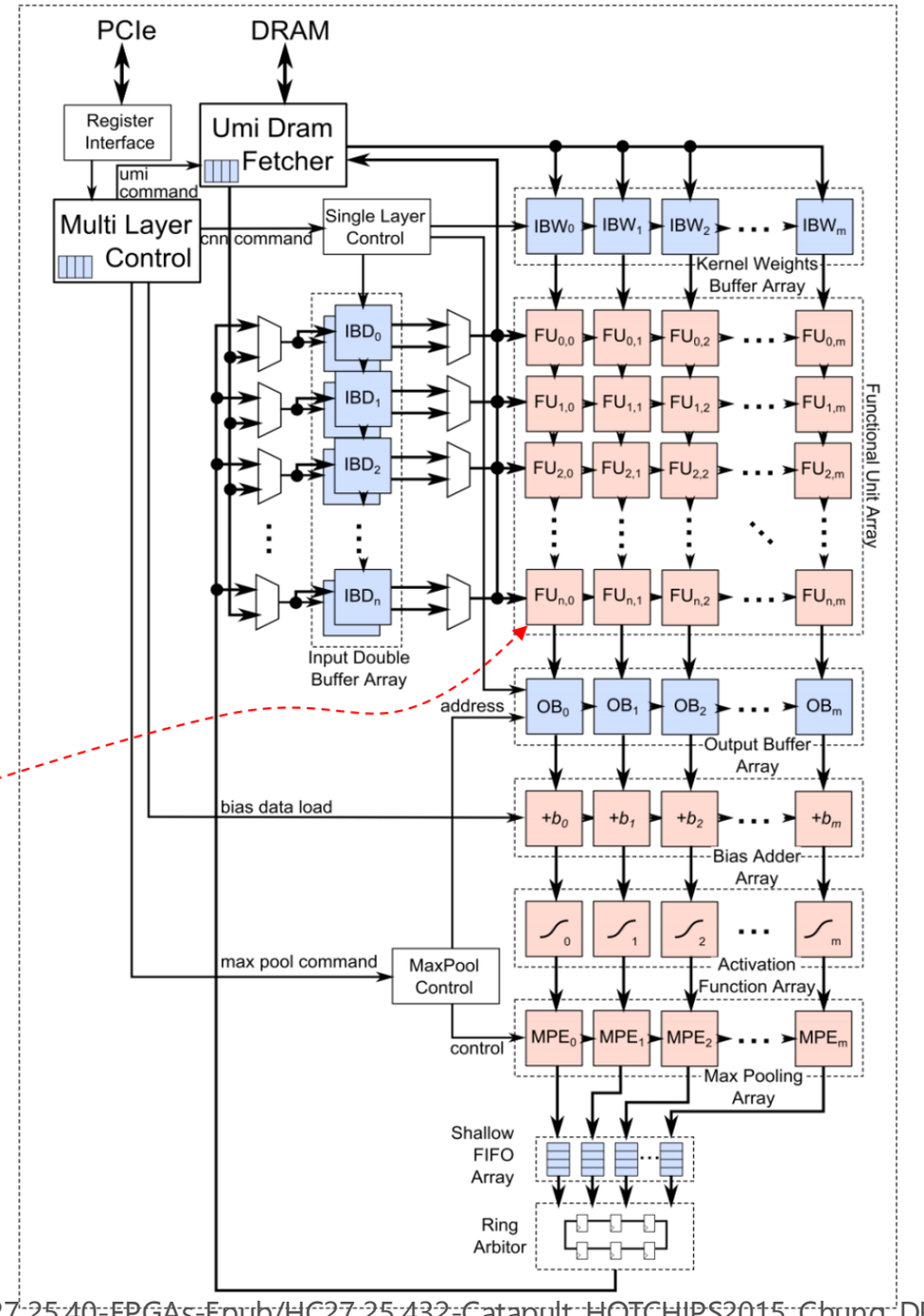
[Eric Chung, MSR]

[Toward Accelerating Deep Learning at Scale Using Specialized Hardware in the Datacenter, Hot Chips 27]

[http://www.hotchips.org/wp-content/uploads/hc_archives/hc27/HC27.25-Tuesday-Epub/HC27.25.40-FPGAs-Epub/HC27.25.432-Catapult_HOTCHIPS2015_Chung_DRAFT_V8.pdf]

Systolic Array Microarchitecture





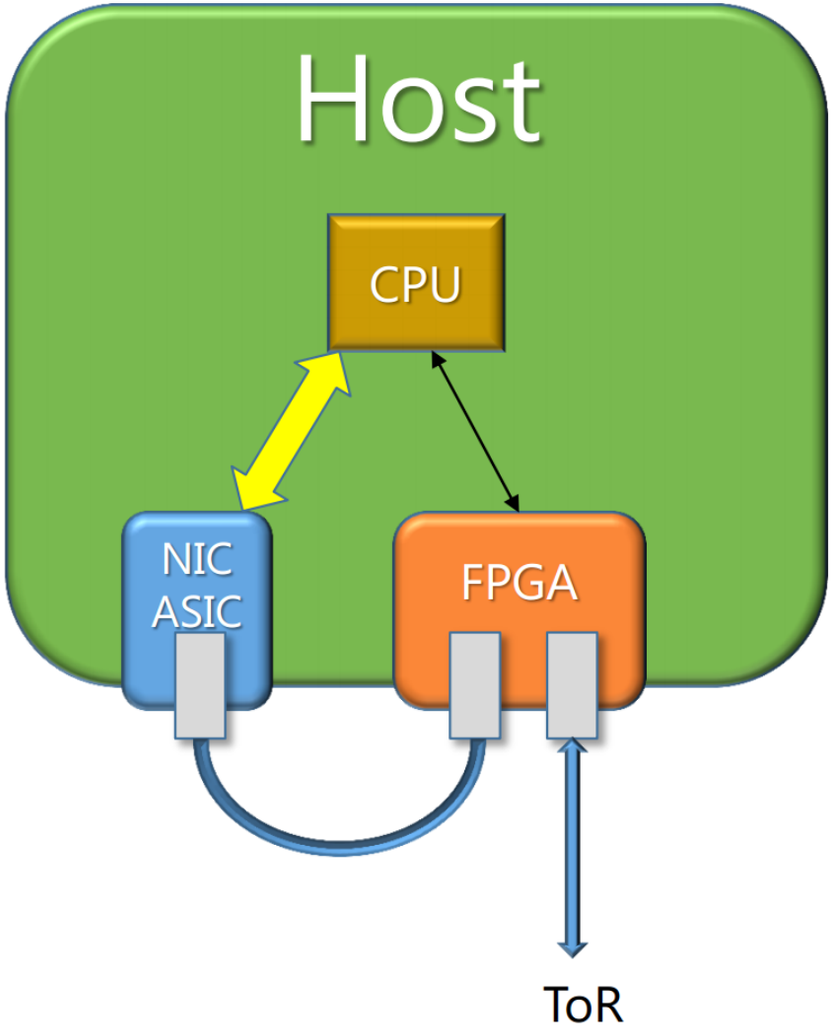
[http://www.hotchips.org/wp-content/uploads/hc_archives/hc27/HC27.25-Tuesday-Epub/HC27.25.40-FPGAs-Epub/HC27.25.432-Catapult_HOTCHIPS2015_Chung_DRAFT_V8.pdf]

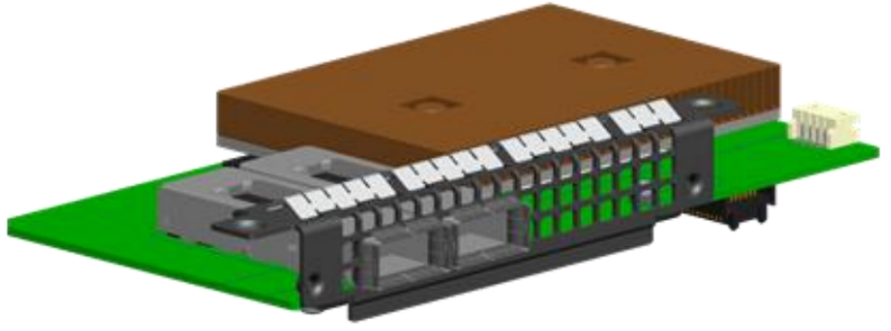
Far better energy efficiency (convnet)

Platform	Library/OS	ImageNet 1K Inference Throughput	Peak TFLOPs	Effective TFLOPs	Estimated Peak Power for **CNN Computation**	Estimated GOPs/J (assuming peak power)
16-core, 2-socket Xeon E5-2450, 2.1GHz	Caffe + Intel MKL Ubuntu 14.04.1*	53 images/s	0.27T	0.074T (27%)	~225W	~0.3
Arria 10 GX1150	Windows Server 2012	~~369 images/s[1]~~ **~880 images/s**	1.366T	~~0.51T (38%)~~ **~1.2T (89%)**	~40W	~~20.6~~ **~30.6**
NervanaSys-32 on NVIDIA Titan X	NervanaSys-32 on Ubuntu 14.0.4	4129 images/s[2]	6.1T	5.75T (94%)	~250W	~23.0

Azure SmartNIC

- Use Catapult FPGAs for reconfigurable functions
  - Already used in Bing
  - Roll out Hardware as we do software
- Programmed using Generic Flow Tables (GFT)
  - Language for programming SDN to hardware
  - Uses connections and structured actions as primitives
- SmartNIC also does Crypto, QoS, storage acceleration, and more…





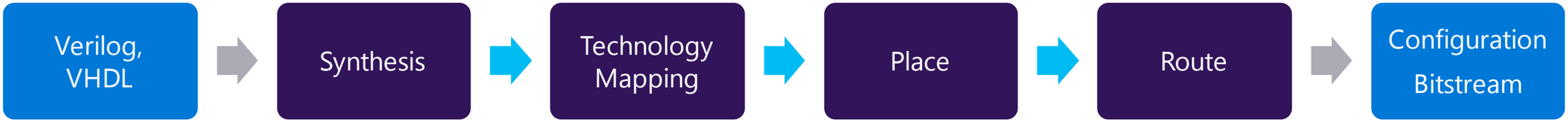
Some themes

- Diverse workloads can run on same FPGA
- Extreme compute and local memory parallelism
- Streaming external data
- Map app to a custom semi-programmable overlay arch

The ascent of FPGA tools

1998: HDL synthesis

- System ::= hierarchy of modules
- Module ::= ports, wires, registers, submodules, processes



- *Does it fit? Does it route? Does it meet timing?*
- A far cry from software dev experience
  - Low level. Lousy languages. Weak abstractions. Limited portability. *Slow!*
- Still the dominant paradigm

2011: High level synthesis: 'C to gates'

- Module ::= HLS C function
- Compiler: HLS C→ data+CFG/DFG → datapath+FSM
- Semi-automates expert strategies and optimizations
- Fast design space exploration → greater productivity

[Jason Cong, UCLA: AutoESL]

HLS example: 16×16 matrix multiply

```
void mm(
    int a[A_ROWS][A_COLS],
    int b[A_COLS][B_COLS],
    int c[A_ROWS][B_COLS])
{



    a_row_loop: for (int i = 0; i < A_ROWS; i++) {
        b_col_loop: for (int j = 0; j < B_COLS; j++) {

            int sum = 0;
            a_col_loop: for (int k = 0; k < A_COLS; k++) {
                sum += a[i][k] * b[k][j];
            }
            c[i][j] = sum;
        }
    }
}
```

Latency: 25121 clocks
DSPs: 3

[http://tcfpga.org/fpga2013/VivadoHLS_Tutorial.pdf]

Matrix multiply: 18b data, parallel dot product

```
#include <ap_int.h>
void mm_18_parallel_dot_product(
    ap_int<18> int a[A_ROWS][A_COLS],
    ap_int<18> int b[A_COLS][B_COLS],
    ap_int<18> int c[A_ROWS][B_COLS])
{
#pragma HLS ARRAY_PARTITION DIM=2 VARIABLE=a complete
#pragma HLS ARRAY_PARTITION DIM=1 VARIABLE=b complete
    a_row_loop: for (int i = 0; i < A_ROWS; i++) {
        b_col_loop: for (int j = 0; j < B_COLS; j++) {
#pragma HLS pipeline
            ap_int<18> int sum = 0;
            a_col_loop: for (int k = 0; k < A_COLS; k++) {
                sum += a[i][k] * b[k][j];
            }
            c[i][j] = sum;
        }
    }
}
```

Latency: 25121 clocks
DSPs: 3

↓

Latency: 260 clocks
DSPs: 16

[http://tcfpga.org/fpga2013/VivadoHLS_Tutorial.pdf]

You must still think like a hardware designer

- Have a good FPGA implementation in mind
- Determine what to unroll, pipeline, partition, …
- Compose HLS modules with rest of system
- Interface system to specific board, host, and software

2015: OpenCL → FPGA!

- OpenCL: for programming heterogeneous machines
- **100% software defined data, compute, synch, …**
- Compiler: kernel → *deeply* pipelined datapath+FSM
- Portable!
- Not *performance* portable

You must still think like a hardware designer

- Have a good FPGA implementation in mind
- Determine what to unroll, pipeline, partition, …
- ~~Compose modules with rest of system~~
- ~~Interface system to specific board, host, and software~~

- Accelerator libraries! – BLAS, OpenCV, ML
- No knowledge of FPGA *or OpenCL* needed to call them

OpenCL→FPGA is necessary but not sufficient

- ~0% of software is OpenCL today – 'dusty deck' C++?
- Bitstream is kernel(s)-specific→ hours per design spin

‘Software-first’ accelerator design

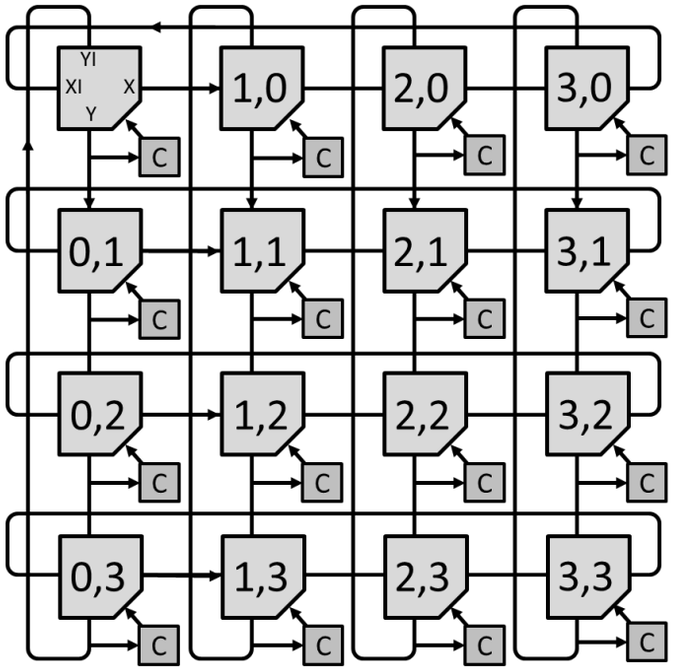
Microsoft Research
Faculty Summit **2016**

Software-first, software-mostly accelerators

- Often easier to build accelerator than integrate it with SW
  - "Big bang" first port
  - Software changes often – accelerator maintenance, agility
- Instead –
  - Compile and run your C++ software directly on *many* soft processors in the FPGA
  - Keep common code base
  - Add custom hardware acceleration as needed
  - More recompiles, fewer FPGA place-and-route jobs

GRVI Phalanx
A massively parallel RISC-V accelerator framework

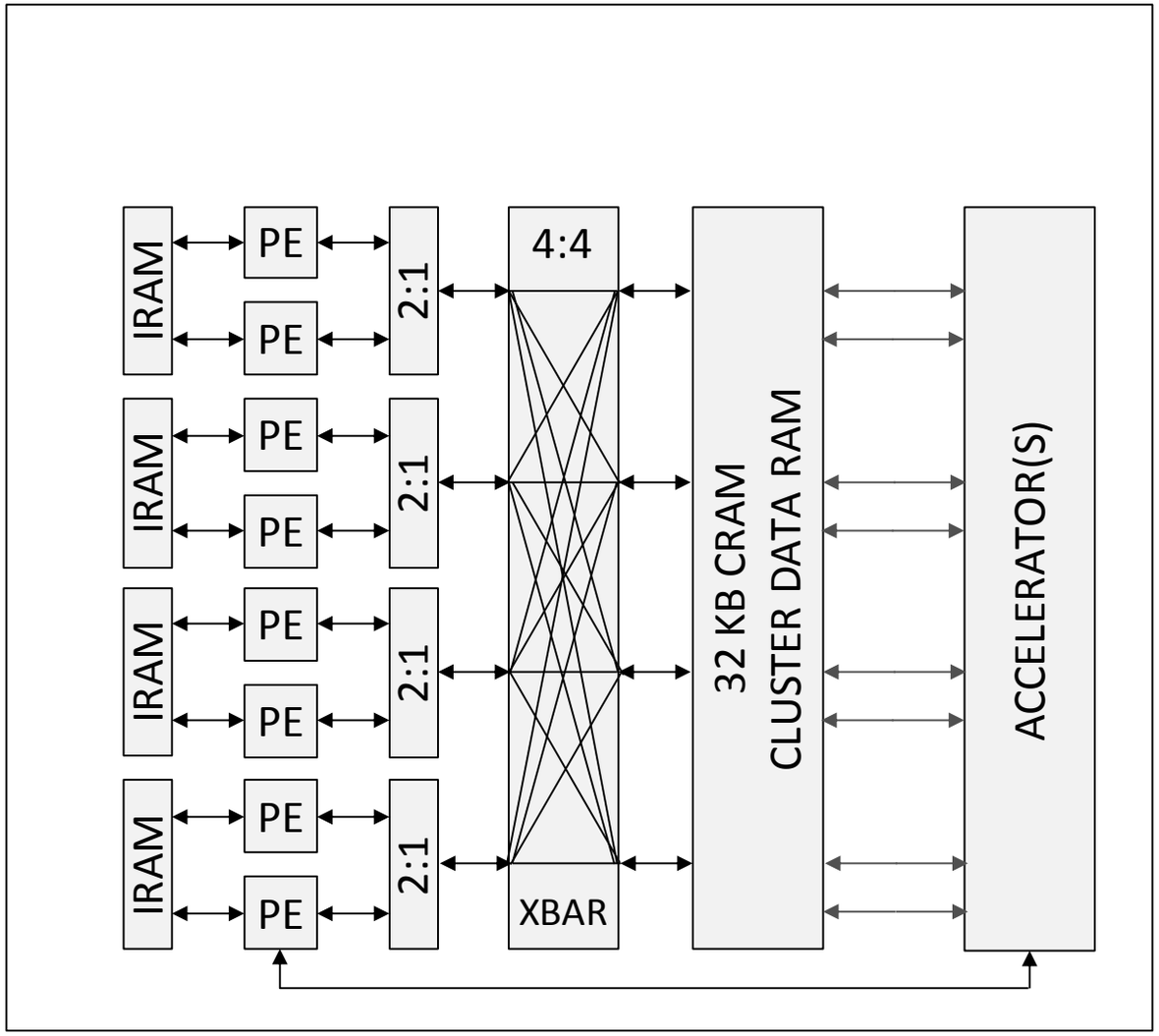
- GRVI: FPGA-efficient RISC-V RV32I soft processor
  - 320 LUTs@375 MHz (KU-2) ≈ 1 MIPS/LUT
- Phalanx: processor/accelerator/IO fabric
  - Clusters of soft CPUs and accelerators, I/O controllers, composed on a Hoplite 2D torus NOC



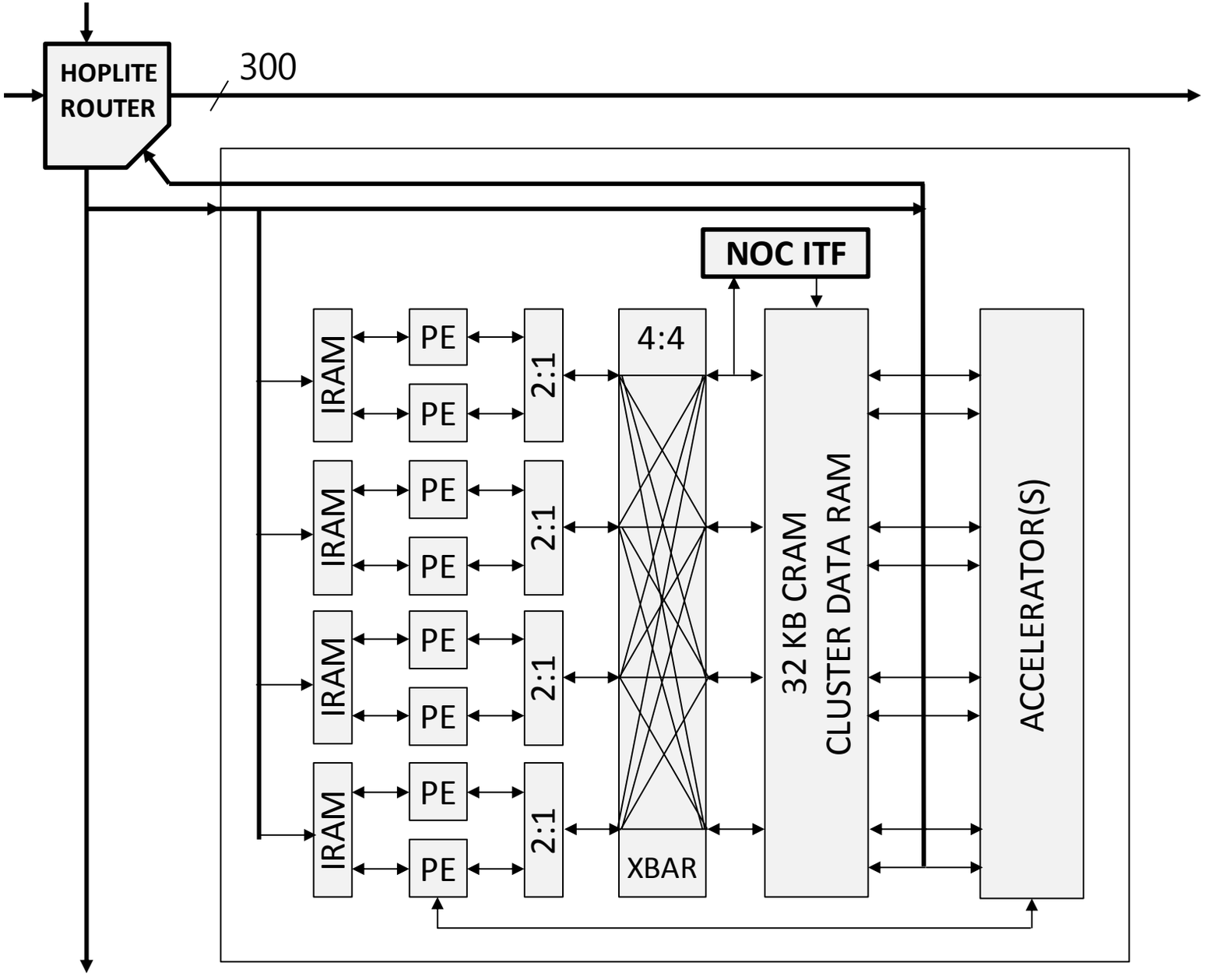
One GRVI processor in a Xilinx KU040 FPGA

{240,000 LUTs + 600 BRAMs} ÷ 320 LUTs ≈ 750 PEs??

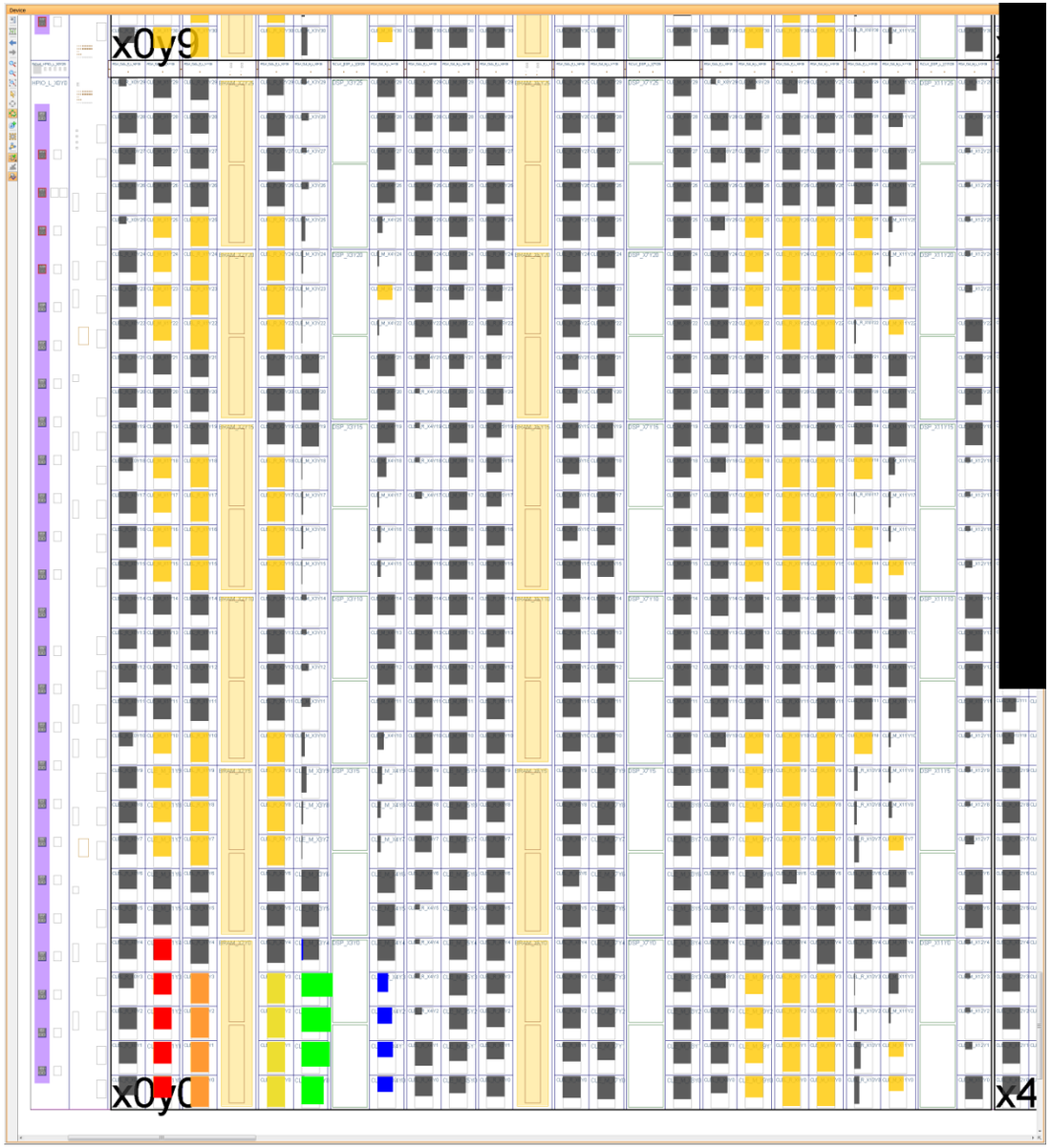
Cluster tile: 8 GRVI PEs + 12 BRAM



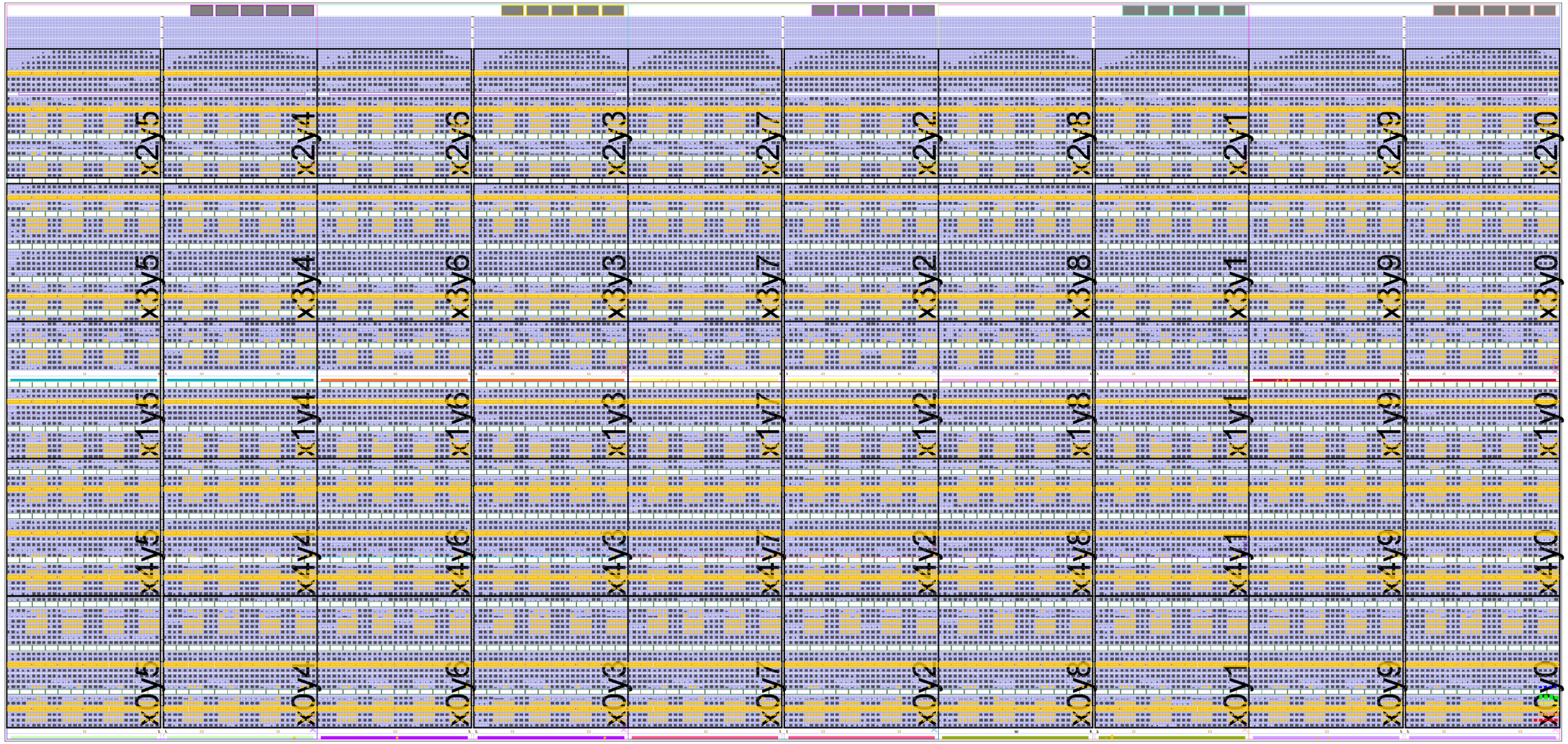
Cluster tile: 8 GRVI PEs + 12 BRAM + NOC router



Cluster tile: 8 GRVI PEs + 12 BRAM + NOC router



10×5 clusters ×8 = 400 GRVI Phalanx



400 RISC-Vs!

- ≤ 100,000 MIPS, 600 GB/s CRAM bandwidth
- 12 W idle / 18 W busy → 30-45 mW/processor, all in
  - Running parallel integer matrix multiplies with Thoth message passing

Accelerated parallel models (aspirational)

- OpenCL kernels – one work group/cluster
- ‘Gatling gun’ packet processing
- Message passing process networks

- *Accelerated*
  - Custom function units, memories, interconnects
  - Custom accelerators on cluster RAM or on NOC

Some research problems for FPGA computing

1. How to **slash PAR times**?
2. How to **specify, debug, profile** complex custom hetero systems?
3. How to **automate energy optimization** FPGA-wide and cloud-wide?
4. How to **share** or **virtualize** accelerators?
5. How to achieve efficient **reliability and fault tolerance** in accelerators?
6. How can HLS or OpenCL tools consistently match **expert quality of results**?
7. What **parallel models** work well with 'software-first' MPPAA overlays?
8. Can an **overlay specification language** improve overlay design and use?
9. Which **FPGA overlays** merit (ASIC) hardening?
10. Role of FPGA acceleration in comp.arch **classroom**?

Thank you!