CprE 488 – Embedded Systems Design

Lecture 8 – Hardware Acceleration

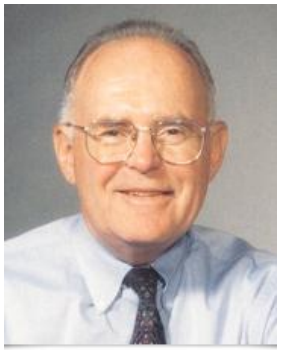
Joseph Zambreno
Electrical and Computer Engineering
Iowa State University

www.ece.iastate.edu/~zambreno
rcl.ece.iastate.edu

First, solve the problem. Then, write the code. – John Johnson

Motivation: Moore's Law

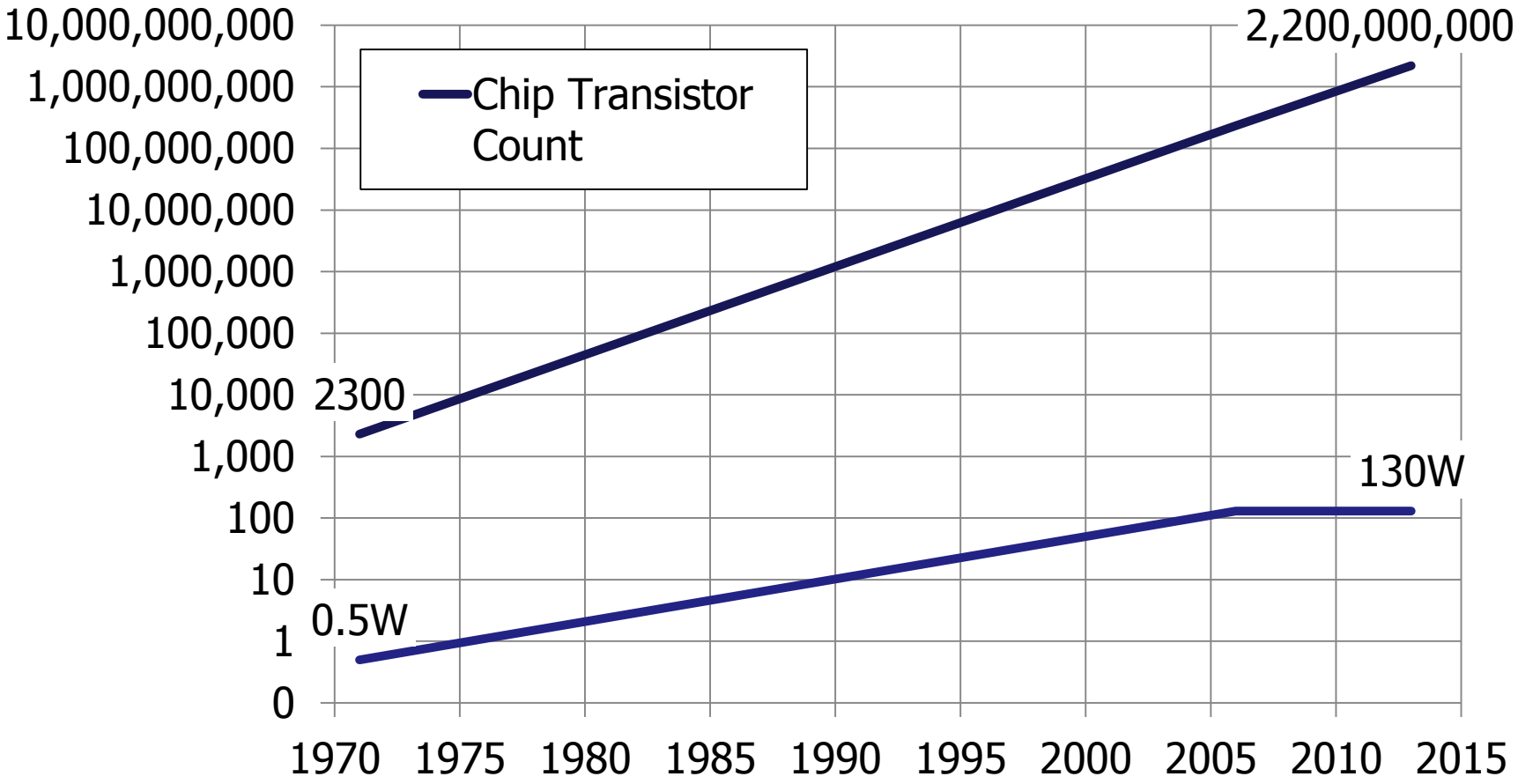
- Every two years:
  - Double the number of transistors
  - Build higher performance general-purpose processors
    - Make the transistors available to the masses
    - Increase performance (1.8×↑)
    - Lower the cost of computing (1.8×↓)
- Sounds great, what's the catch?



Gordon Moore

Motivation: Moore's Law (cont.)

- The "catch" – powering the transistors without melting the chip!

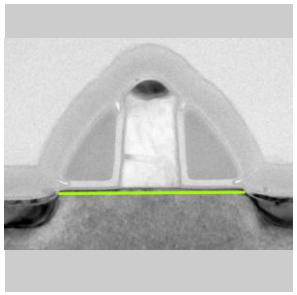


Motivation: Dennard Scaling

- As transistors get smaller their power density stays constant

Robert Dennard

Transistor: 2D Voltage-Controlled Switch

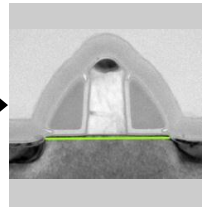


Dimensions

Voltage

Doping Concentrations

×0.7 →



Area ——— **0.5×↓** ———→

Capacitance ——— **0.7×↓** ———→

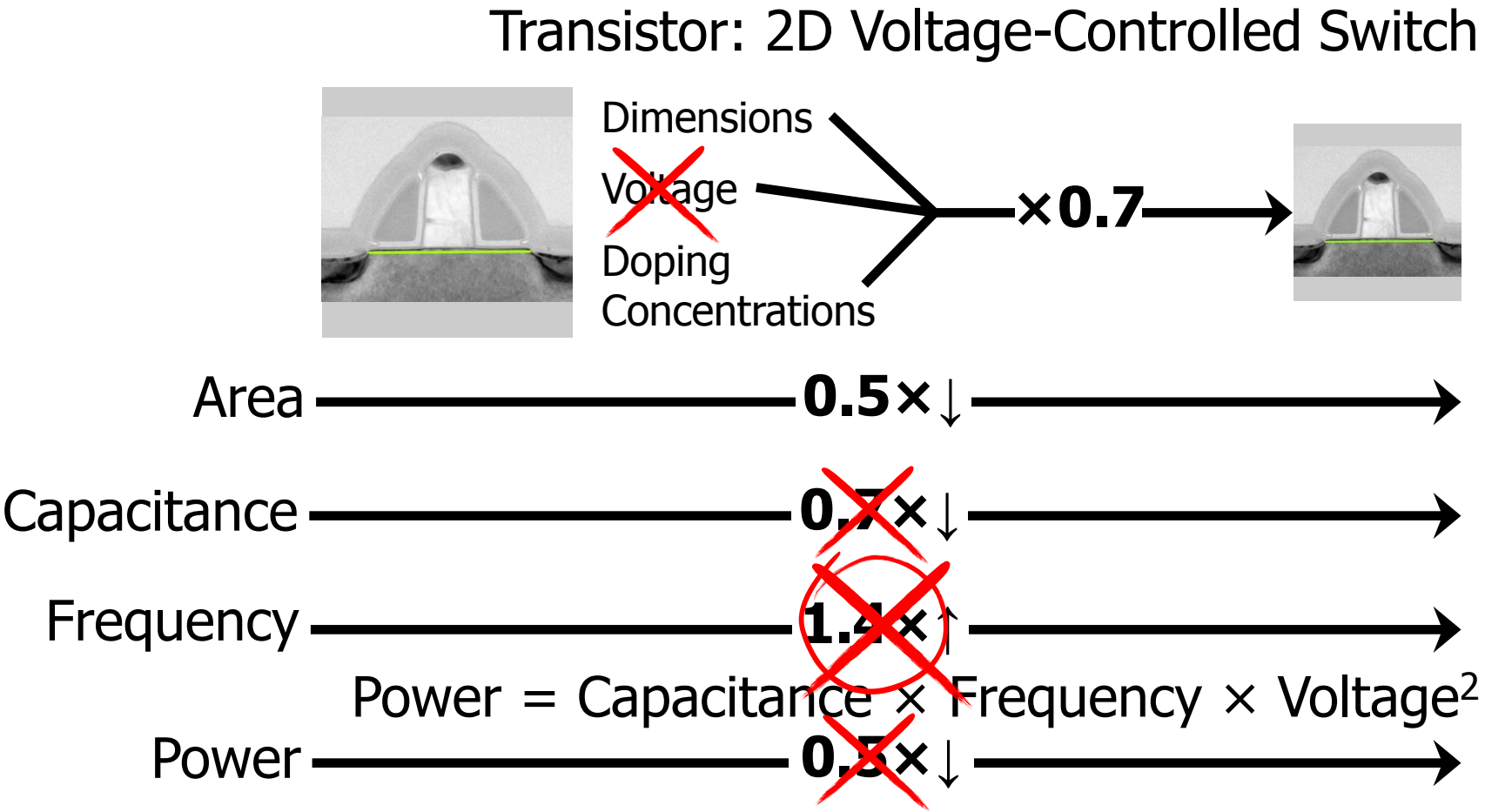
Frequency ——— **1.4×↑** ———→

Power = Capacitance × Frequency × Voltage^2

Power ——— **0.5×↓** ———→

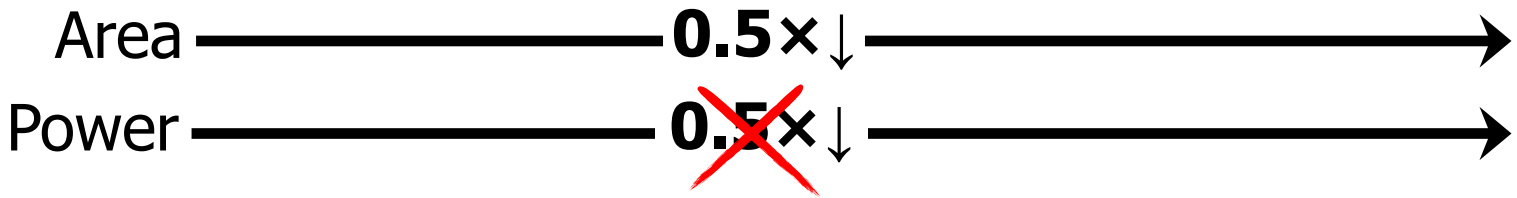
Motivation Dennard Scaling (cont.)

- In mid 2000s, Dennard scaling "broke"

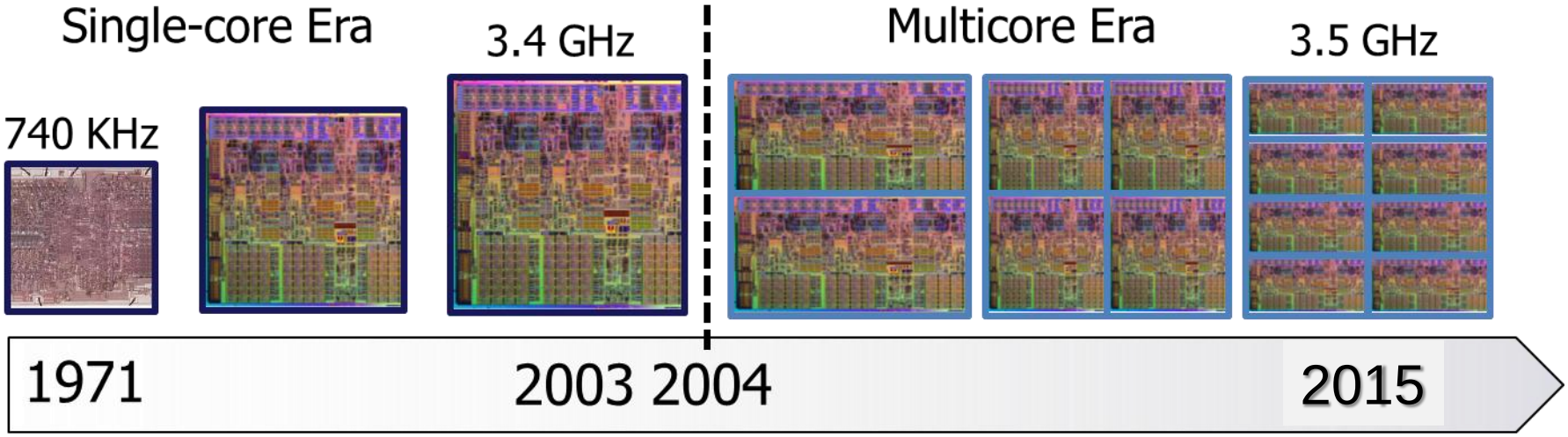


Motivation: Dark Silicon

- **Dark silicon** – the fraction of transistors that need to be powered off at all times (due to power + thermal constraints)



- Evolution of processors strongly motivated by this ending of Dennard scaling
  - Expected continued evolution towards HW specialization / acceleration



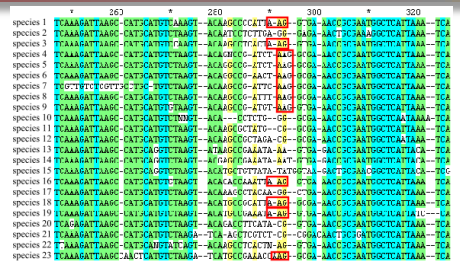
This Week's Topic

- Hardware Acceleration:
  - Performance analysis and overhead
  - Coprocessors vs accelerators
  - Common acceleration techniques
  - Acceleration examples

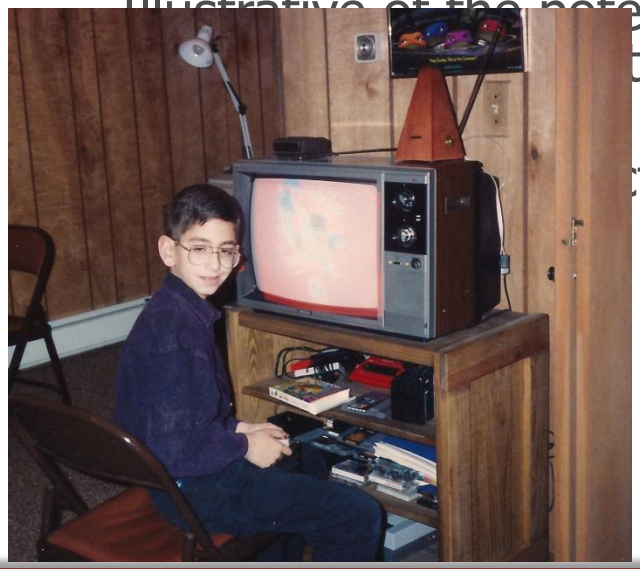
- Reading: Wolf section 10.4-10.5

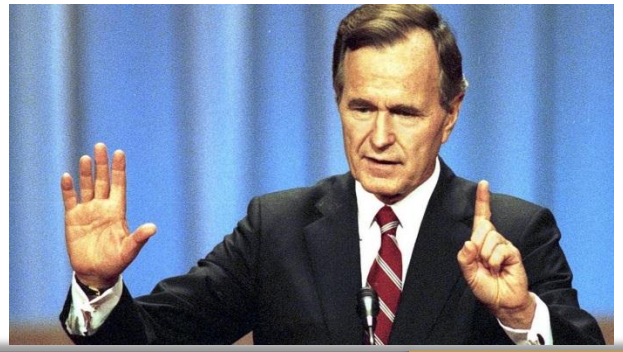
Straight from the Headlines...

- Accelerating a diverse range of applications using reconfigurable logic is a trending area:
  - D. Hoang, D. Lopresti, "FPGA Implementation of Systolic Sequence Alignment"
  - D. Ross, O. Vellacott, M. Turner, "An FPGA-based Hardware Accelerator for Image Processing"
  - J. Lockwood, "Design and Implementation of a Multicast, Input-Buffered ATM Switch for the iPOINT Testbed"



- What these examples have in common:
  - Illustrative of the potential for custom computing
  - tific and engineering
  - t (a bridge from ... a)





And Today?

- Reconfigurable logic supporting the data center, specializing clusters, and tightly integrated on-chip

EXTREMETECH

Computing / Mobile / Internet / Gaming / Electronics / Extreme

ELECTRONICS › MICROSOFT TO ACCELERATE BING SEARCH WITH NEURAL NETWORK

Microsoft to accelerate Bing search with neural network

By John Hewitt on February 25, 2015 at 3:46 pm | 19 Comments

Share This Article 991 105 50

When we search Google's web index, we are only searching around 10 percent of the half-a-trillion or so pages that are potentially available. Much of the content in the larger deep web — not to be confused with the dark web — is buried further down in the sites that make up the visible surface web. The indexes of competitors like Yahoo and Bing (around 15 billion pages each) are still only half as large as Google's. To close this gap, Microsoft has recently pioneered sophisticated new Field-Programmable Gate Array (FPGA) technology to make massive web crawls more efficient, and faster.

Google's engineers have previously estimated that a typical 0.2-second web query reflects a quantity of work spent in indexing and retrieval equal to about 0.0003 kWh of energy per search. With over 100 billion looks per month at their petabyte index, well-executed page ranking has become a formidable proposition. Microsoft's approach with Bing has been to break the ranking portion of search into three parts — feature extraction, free-form expressions, and machine learning scoring:

ENTERPRISETECH
SYSTEMS EDITION

Home | Systems | Software | Datacenter | Cloud | Storage | Networks | Insight | Sectors

IBM Accelerates Power8 Clusters With GPUs, FPGAs, And Flash

October 2, 2014 by Timothy Prickett Morgan

It is perhaps a lucky stroke of timing or perhaps by design that only days after Big Blue sold off its System x X86 server business to Lenovo Group for $2.1 billion that the company is coming out swinging with Power8 servers that are augmenting their performance using a variety of adjunct co-processors and flash storage. But ahead of next week's Enterprise2014 event in Las Vegas, where it will be talking about its increasing focus on Power Systems and System z mainframes, the company is launching a number of systems that are designed to take workloads away from X86 clusters.

As *EnterpriseTech has previously reported*, IBM has been telling customers to expect larger Power8-based machines with more than two sockets as well as systems that would use field programmable gate arrays (FPGAs). IBM has also been hinting that OpenPower partner and GPU coprocessor maker Nvidia would be working together to get a Power8-Tesla hybrid system into the field before the end of the year.

It turns out that IBM is launching three different systems tuned up for three different kinds of workloads that are based on its "scale-out" Power8 systems. By scale-out, IBM means a system is designed with one or two sockets and is intended to be used in clusters that have distributed applications that scale their capacity by adding multiple nodes in a loosely coupled fashion. This is distinct from "scale-up" machines, which more tightly couple server nodes and their main memory together, usually using non-uniform memory access (NUMA) technology, to create what is in essence a single large processor to run fat applications or their databases.

Big Blue is also rolling out scale-up versions of its Power8 systems, which it has also promised would come this year, ahead of the Enterprise2014 event. So don't think the Power8 rollout is only about creating a Power8 alternative to the workhorse, two-socket server based on Intel's Xeon E5-2600 processors. (We will report on these NUMA machines, which are called the Power Enterprise Systems, in a separate story.)

The new Power S824L is a Linux-only version of the existing Power S824 machine that IBM announced back

EXTREMETECH

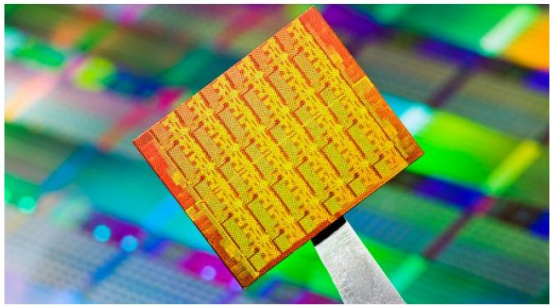
Top Searches: Windows 10 • Autos • Quantum • Intel Trending: Linux • Windows 8 • NASA • Batteries •

Computing / Mobile / Internet / Gaming / Electronics / Extreme

COMPUTING › INTEL UNVEILS NEW XEON CHIP WITH INTEGRATED FPGA, TOUTS 20X PERFORMANCE BOOST

Intel unveils new Xeon chip with integrated FPGA, touts 20x performance boost

By Sebastian Anthony on June 19, 2014 at 1:19 pm | 59 Comments



Share This Article 503 115 142

Late yesterday, Intel quietly announced one of the biggest ever changes to its chip lineup: It will soon offer a new type of Xeon CPU with an integrated FPGA. This new Xeon+FPGA chip will fit in the standard E5 LGA2011 socket, but the integrated FPGA will allow each chip to be customized to specific workloads. This move is almost certainly intended to make Intel-x86 a better all-round platform for a wider variety of workloads in enterprise and data center settings, and to dissuade customers from switching to GPGPU accelerators from the likes of Nvidia.

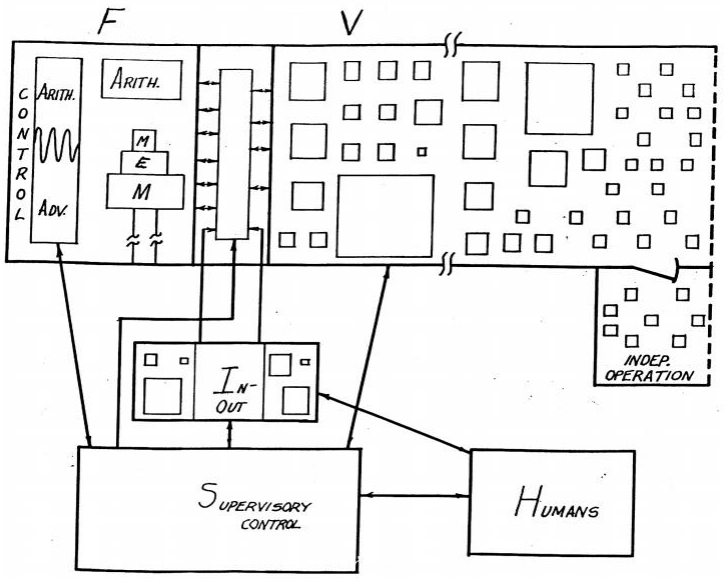
The Xeon+FPGA also raises the question of whether Intel would ever consider integrating an FPGA into its consumer Core line of chips — it's exceedingly unlikely, but it it's hard to deny how awesome it would be if next-gen games and apps had access to an FPGA to speed up core processes. But more on that at the end of the story.

What is an FPGA?

A (Brief) Background

- Earliest reconfigurable computer proposed at UCLA in the 1960s
  - G. Estrin et al., "Parallel Processing in a Restructurable Computer System," *IEEE Trans. Electronic Computers*, pp. 747-755, Dec. 1963.
  - Basic concepts well ahead of the enabling technology – could only prototype a crude approximation

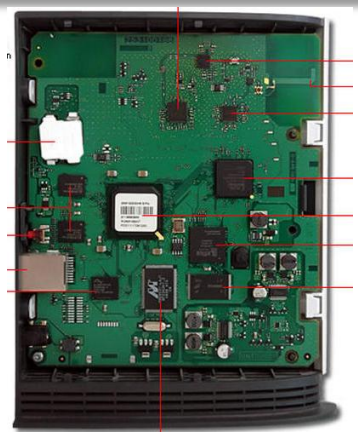
- Current chips – contain memory cells that hold both configuration and state information
  - Only a partial architecture exists before programming
  - After configuration, the device provides an execution environment for a specific application



- Goal is to **adapt** at the **logic-level** to solve **specific** problems

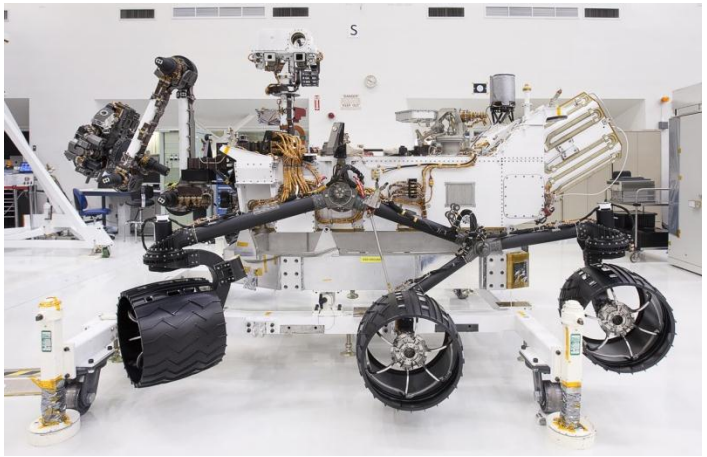
A (Brief) Background

- Today, well known niches for FPGAs:
  - Emulation
  - Telecom
  - Space / defense
- $4.5B market, major device manufacturers:
  - Xilinx (~45% market share)
  - Altera (~40%)
  - Microsemi / Actel (10%)
  - Others (Lattice, Achronix, Tabula)



"Vodafone Sure Signal: Inside a Femtocell" – http://zdnet.com (Xilinx Spartan 3E FPGA used for glue logic)

"Sneak Peak: Inside NVIDIA's Emulation Lab" – http://blogs.nvidia.com (Cadence Palladium cluster used for GPU emulation)



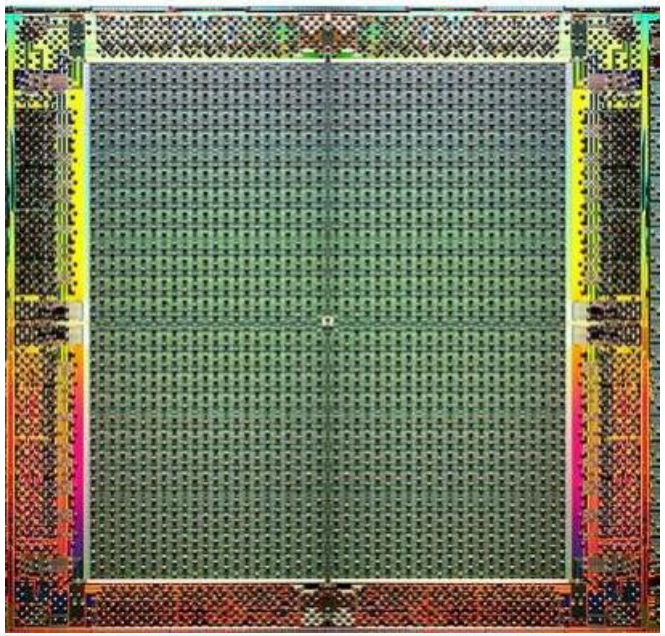
"How Mars Rover Got its 'Dream Mode'" – http://eetimes.com (Microsemi FPGAs reprogrammed for surface exploration)

So Who Cares?

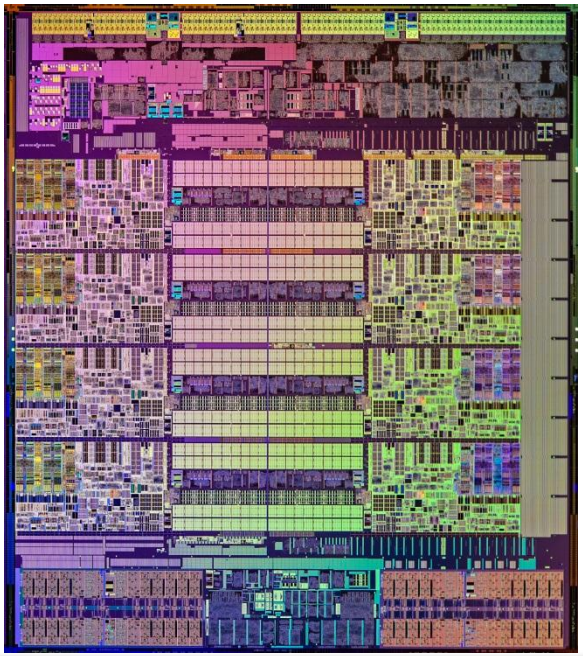
- Claim – reconfigurable processors offer a definite advantage over general-purpose counterparts with regards to functional density
  - A. DeHon. "The Density Advantage of Configurable Computing," *IEEE Computer,* pp. 41-49, Apr. 2000
  - Computations per chip area per cycle time

- Considering general computing trends (Moore's Law, Dennard Scaling, "dark" silicon) – what can I get for my N billion transistors?

Altera Stratix IV EP4S40G5

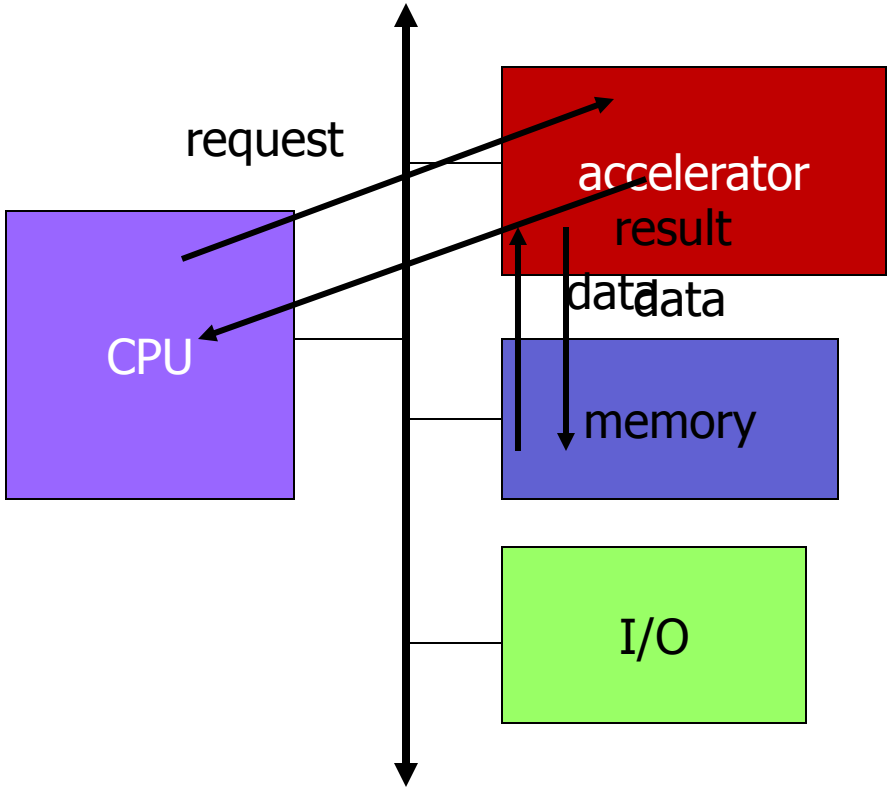


Intel Haswell Core i7-5960X



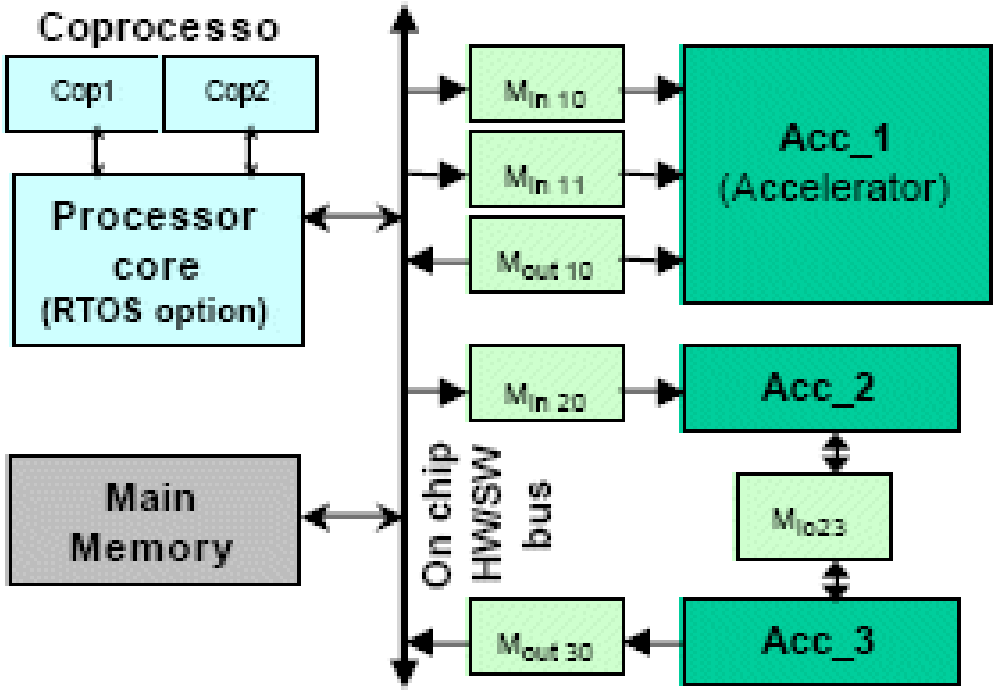
Accelerated Systems

- Use additional computational units dedicated to some functionality
- Hardware/software co-design: joint design of hardware and software architectures



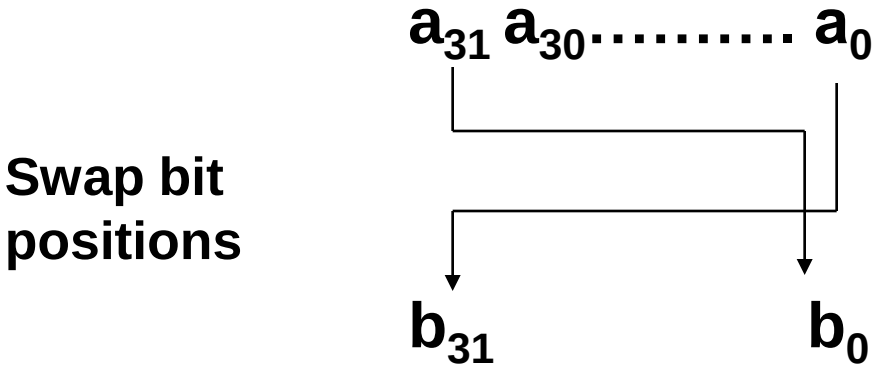
Accelerator vs. Coprocessor

- A coprocessor executes instructions
  - Instructions are dispatched by the CPU
- An accelerator appears as a device on the bus
  - Typically controlled by registers (memory-mapped IO)



EX-08.1: Instruction Augmentation

- Processor can only describe a small number of basic computations in a cycle
  - I bits -> 2^I operations
- Many operations could be performed on 2 W-bit words
- ALU implementations restrict execution of some simple operations
  - e. g. bit reversal



Accelerated System Design

- First, determine that the system really needs to be accelerated
  - How much faster will the accelerator on the core function?
  - How much data transfer overhead? Compute bound vs memory bound vs I/O bound?
- Design accelerator and system interface
- If tighter CPU integration required:
  - Create a functional unit for augmented instructions
  - Compiler techniques to identify/use new functional unit

Amdahl's Law

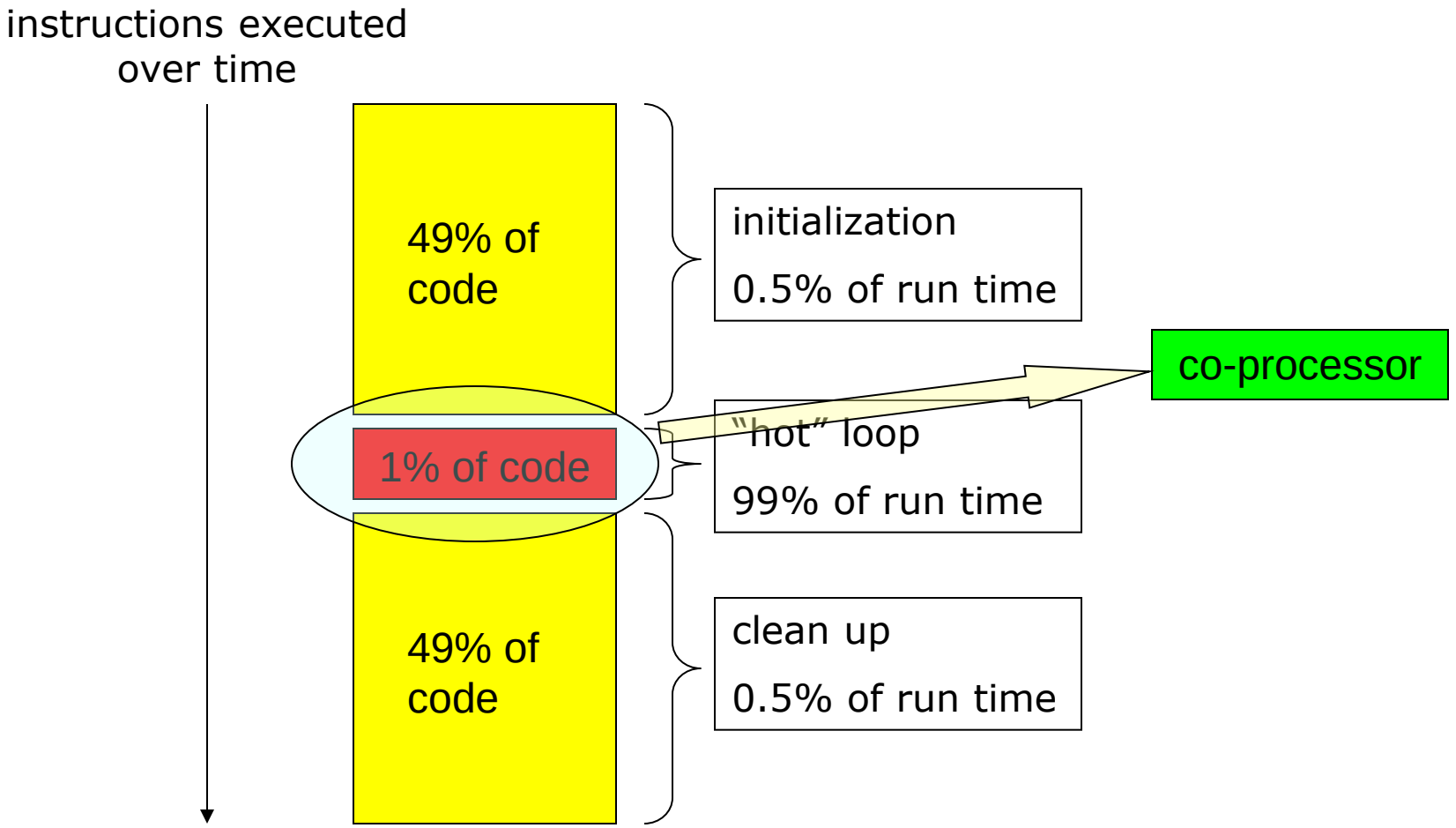
- If an optimization improves a fraction f of execution time by a factor of a

$$Speedup = \frac{Told}{[(1-f)+f/a]\times Told} = \frac{1}{(1-f)+f/a}$$

- This formula is known as Amdahl's Law
- Lessons from
  - If $f \rightarrow 100\%$, then speedup = a
  - If $a \rightarrow \infty$, the speedup = $1/(1-f)$
- Summary
  - Make the common case fast
  - Watch out for the non-optimized component

Gene Amdahl

Heterogeneous Execution Model

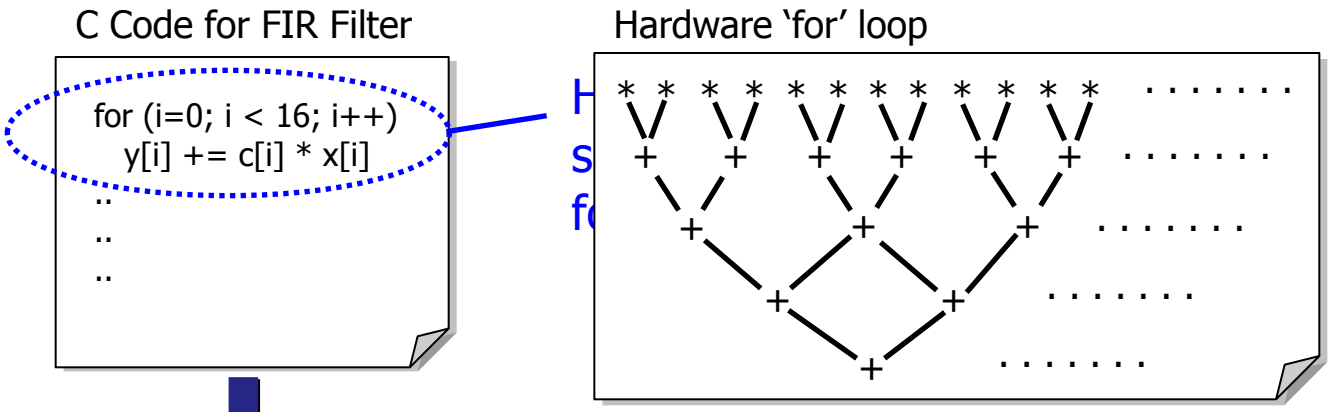


Heterogeneous Computing: Performance

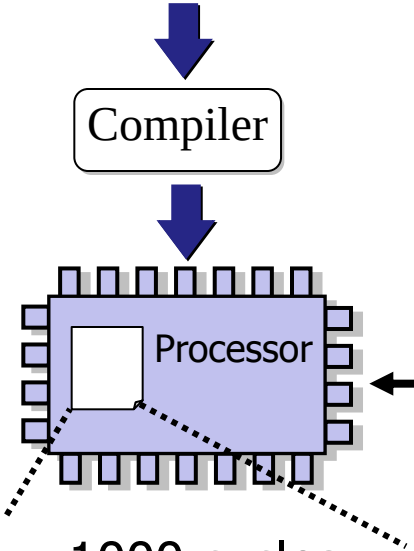
- Move "bottleneck" computations from software to hardware
- Example:
  - Application requires a week of CPU time
  - One computation consumes 99% of execution time

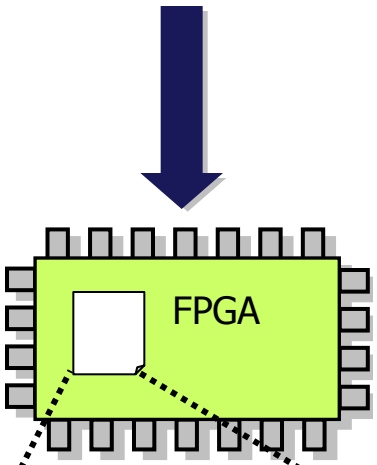
Kernel speedup	Application speedup	Execution time
50	34	5.0 hours
100	50	3.3 hours
200	67	2.5 hours
500	83	2.0 hours
1000	91	1.8 hours

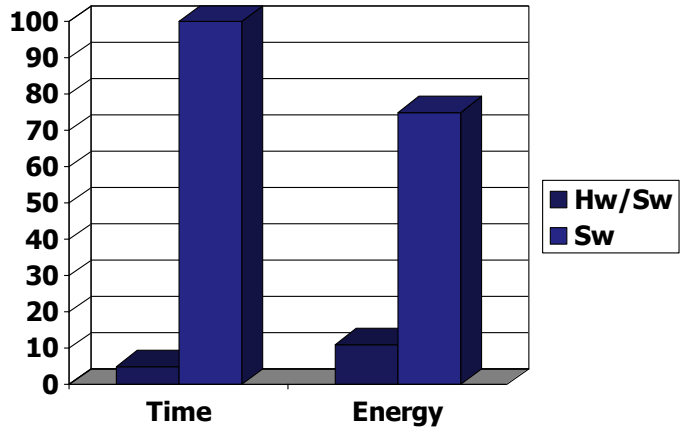
Hardware/Software Partitioning



Designer creates custom accelerator using hardware design methodology

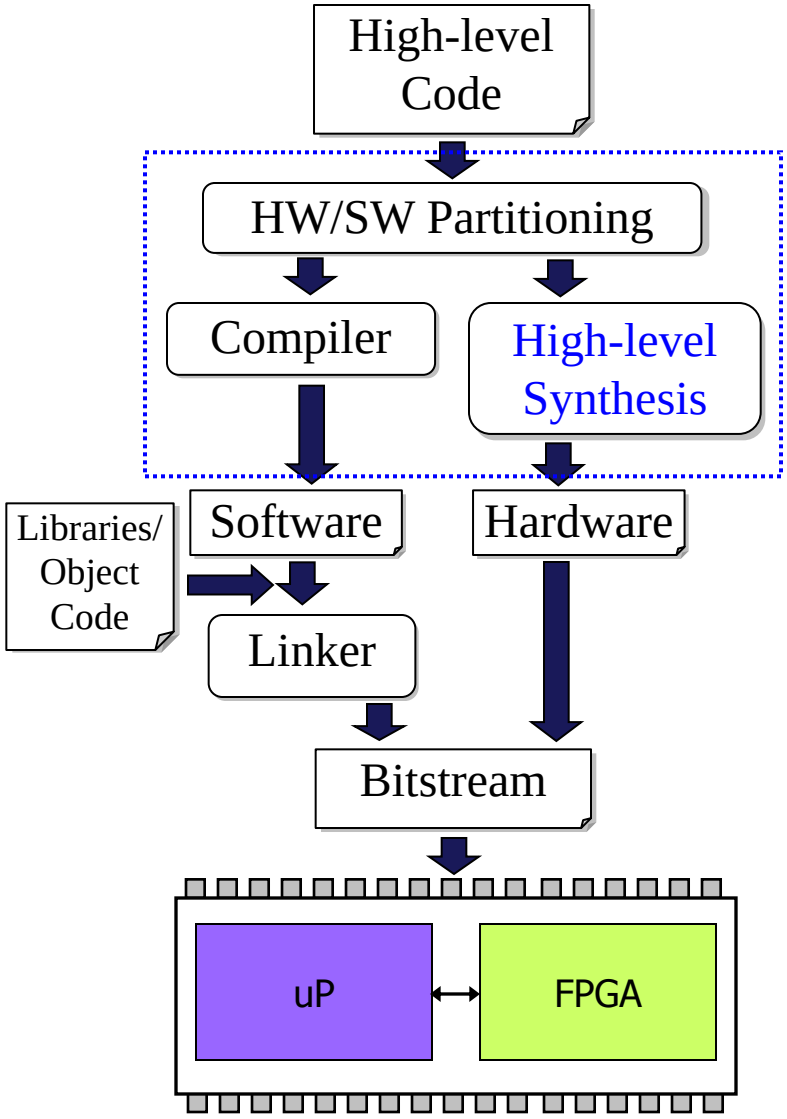






- ~1000 cycles
- ~ 10 cycles
  - Speedup = 1000 cycles/ 10 cycles
    = 100x

High-Level Synthesis



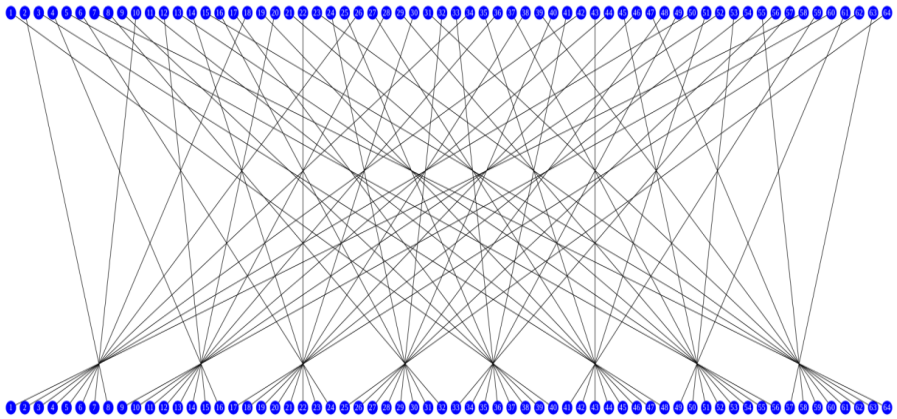
- *Problem:* Describing circuit using HDL is time consuming/difficult
- *Solution:* High-level synthesis
  - Create circuit from high-level code
  - Allows developers to use higher-level specification
  - Potentially, enables synthesis for software developers
- More on this in a bit

Accelerator Design Challenges

- *Debugging* – how to properly test the accelerator separately, and then in conjunction with the rest of the system (hw/sw co-simulation)
- *Coherency* – how to safely share results between CPU and accelerator
  - Impact on cache design, shared memory
  - Solutions looks similar to those for resource sharing in conventional operating systems, but are typically ad-hoc
- *Analysis* – determining the effects of any hardware parallelism on performance
  - Must take into account accelerator execution time, data transfer time, synchronization overhead
  - Heterogeneous multi-threading helps, but complicates design significantly
  - Overlapping I/O and computation (streaming)

A Representative Example

- PRISM-I: early example of an FPGA / CPU coupling:
  - P. Athanas and H. Silverman. "Processor Reconfiguration through Instruction Set Metamorphosis," *IEEE Computer,* pp. 11-18, Mar. 1993
  - An attempt to augment a RISC instruction set with application-specific operators
  - Significant performance speedups for parallel bit-level operations
- Consider a permutation table from the Data Encryption Standard (DES):



DES Initial Permutation Table. Note the equivalent HW realization can be implemented solely via routing resources.

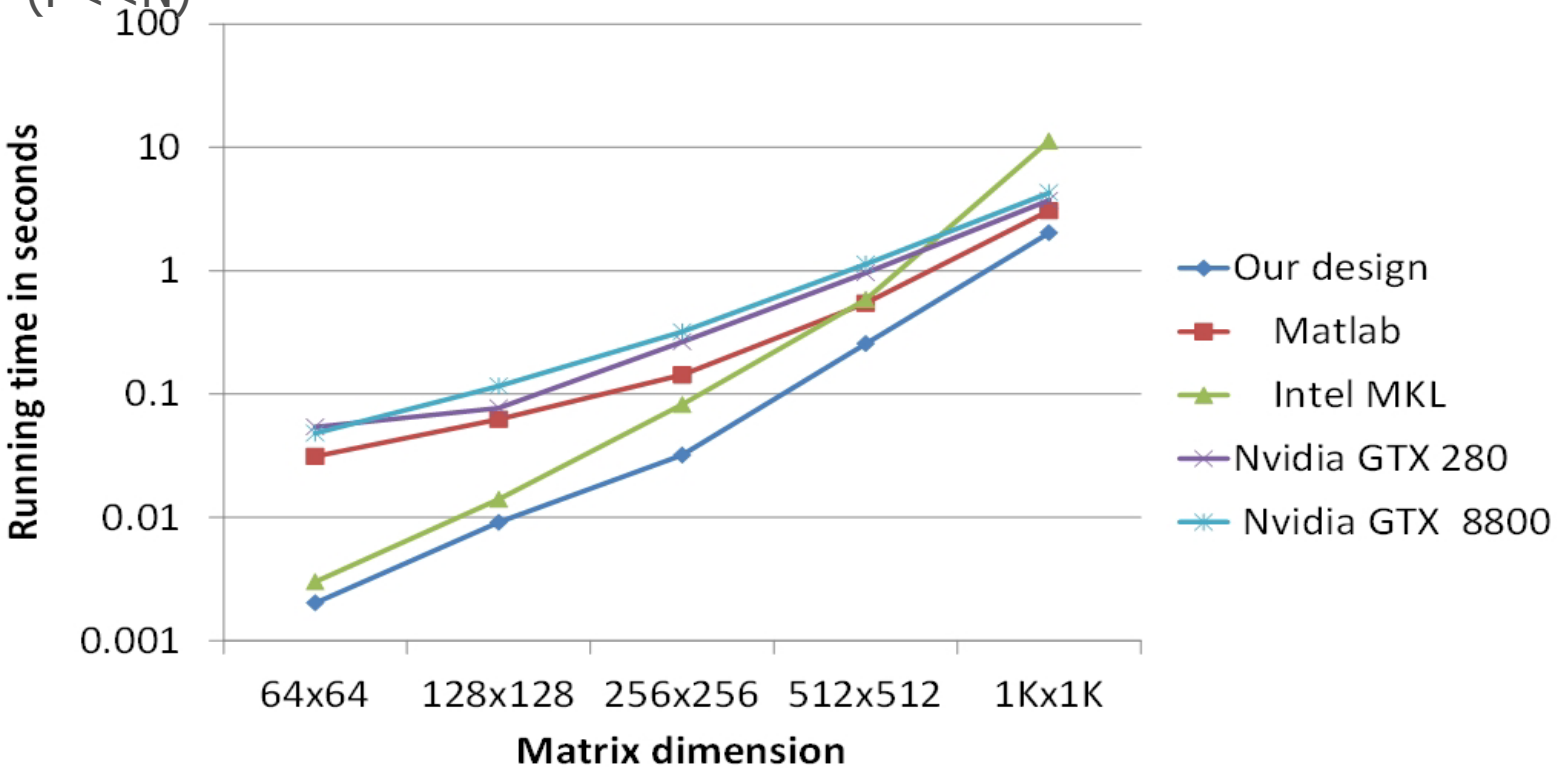
```
#define DO_PERMUTATION(a, temp, b, offset, mask)      \
    temp = ((a>>offset) ^ b) & mask;                  \
    b ^= temp;                                        \
    a ^= temp<<offset;

#define INITIAL_PERMUTATION(left, temp, right)        \
    DO_PERMUTATION(left, temp, right, 4, 0x0f0f0f0f)  \
    DO_PERMUTATION(left, temp, right, 16, 0x0000ffff) \
    DO_PERMUTATION(right, temp, left, 2, 0x33333333)  \
    DO_PERMUTATION(right, temp, left, 8, 0x00ff00ff)  \
    right =  (right << 1) | (right >> 31);            \
    temp  =  (left ^ right) & 0xaaaaaaaa;             \
    right ^= temp;                                    \
    left  ^= temp;                                    \
    left  =  (left << 1) | (left >> 31);
```

Partial libgcrypt implementation from `des.c`. Not well-suited for conventional SW implementation.

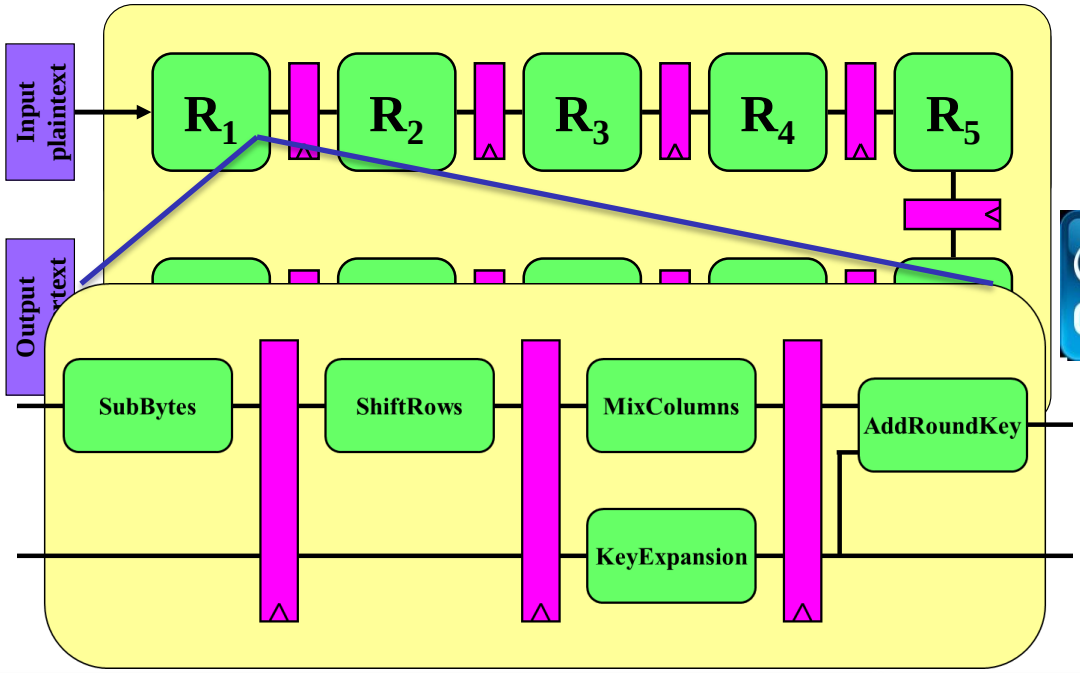
An (Equally) Representative Ex.

- Applicability of FPGAs to a commonly occurring computational kernel in scientific applications
  - X. Wang and J. Zambreno, "An Efficient Architecture for Floating-Point Eigenvalue Decomposition", *Proc. of the Int'l Symposium on Field-Programmable Custom Computing Machines (FCCM)*, May 2014
  - Uses the modified one-sided Jacobi rotation approach, mapped to a 1D systolic array to extract parallelism in EVD computation
  - Fundamental design challenge – tackling $O(N^k)$ complexity with P resources (P<<N)



A Cause for Pessimism?

- Hardware amenability was a design criterion for the Advanced Encryption Standard (AES)
  - Flurry of activity looking at AES on FPGA (literally 1000s of implementations, optimizations, attacks)
  - J. Zambreno, D. Nguyen and A. Choudhary, "Exploring Area/Delay Tradeoffs in an AES FPGA Implementation". *Proc. of the Int'l Conference on Field-Programmable Logic and its Applications (FPL)*, Aug. 2004
    - Main contribution: an early exploration of the design decisions that lead to area/delay tradeoffs in an AES accelerator
    - Significant (at the time) throughput of 23.57 Gbps for AES-128E in ECB mode




```
aesni_encrypt:
                ...
.L000enc1_loop:
        aesenc      %xmm4,%xmm0
        decl        %ecx
        movups      (%edx),%xmm4
        leal        16(%edx),%edx
        jnz         .L000enc1_loop
        aesenclast  %xmm4,%xmm0
        movups      %xmm0,(%eax)
        ret
```

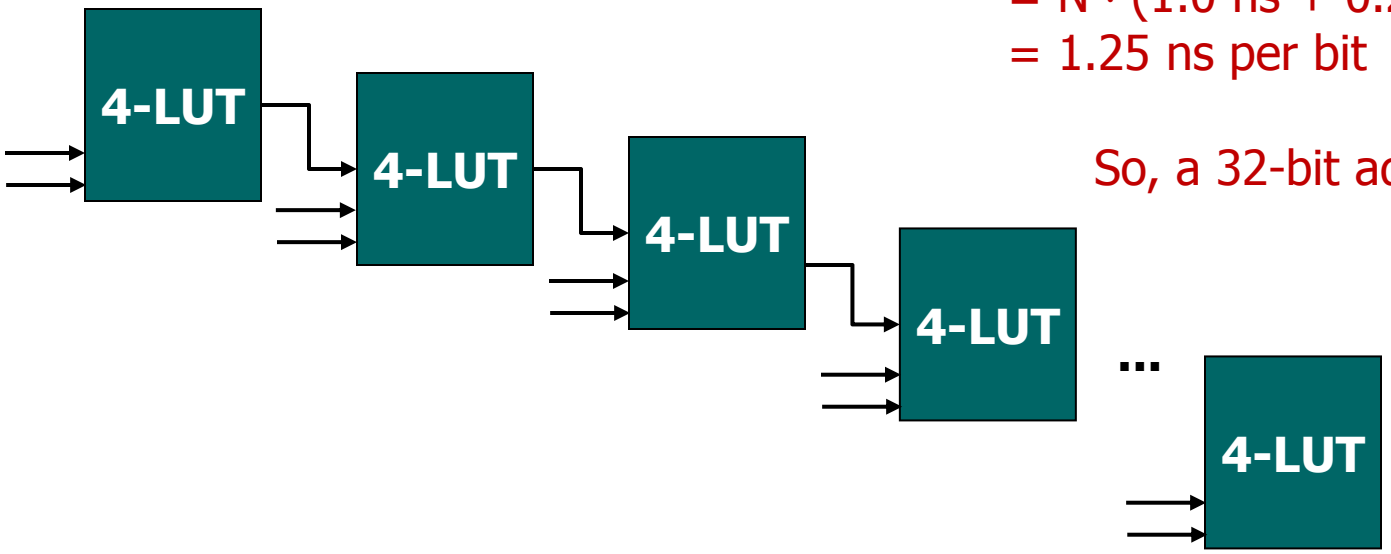
Partial libgcrypt implementation using Intel's AES-NI instruction set. Performance is ~0.75 cycles per byte (40+ Gbps per-thread with relatively easy software patches)

FPGA Evolutionary Pressures

- Base logic element consists of a binary lookup-table (LUT) and a flip-flop
- Inter-element routing dominates both delay and area for non-trivial circuits
- Consider the critical path of a (naïve) ripple-carry adder:

delay = N · (LUT_delay + routing_delay)
= N · (1.0 ns + 0.25 ns)
= 1.25 ns per bit

So, a 32-bit adder at 25 MHz?

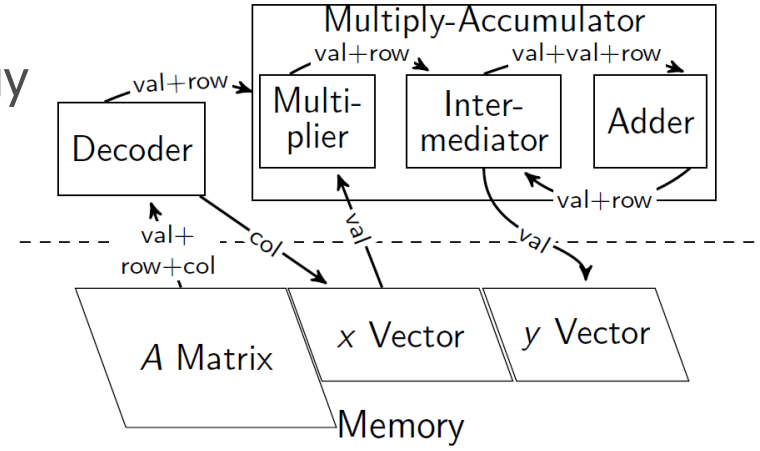


FPGA Evolutionary Pressures (2)

1) Simple arithmetic circuits
   - Dedicated carry logic and routing paths
   - Altera Flex 6000, Xilinx XC4000 series (1991)
2) Memory-intensive tasks
   - LUTs configurable as distributed RAM resources, later built-in SRAM blocks, FIFOs
   - Altera Flex 10K, Xilinx Virtex series (1998)

3) DSP applications
   - Dedicated multipliers, later multiply-accumulate operators
   - Altera Stratix, Xilinx Virtex II series (2001)

- Compelling application:
  - K. Townsend and J. Zambreno, "Reduce, Reuse, Recycle (R^3): a Design Methodology for Sparse Matrix Vector Multiplication on Reconfigurable Platforms", *Proc. of the Int'l Conference on Application-specific Systems, Architectures and Processors (ASAP)*, 2013.
  - 13.7 Gflops for SpMV (matrix dependent)



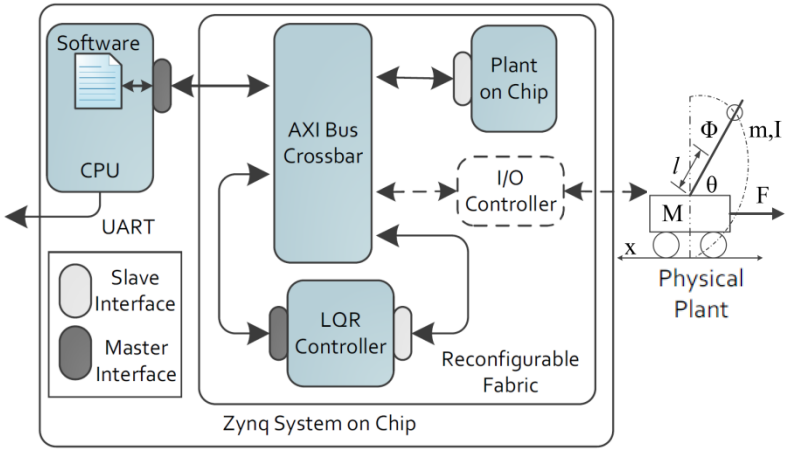
FPGA Evolutionary Pressures (3)

4) Hardware / software codesign
   - Dedicated CPUs
   - Altera Cyclone V, Xilinx Virtex II Pro Series (2002)

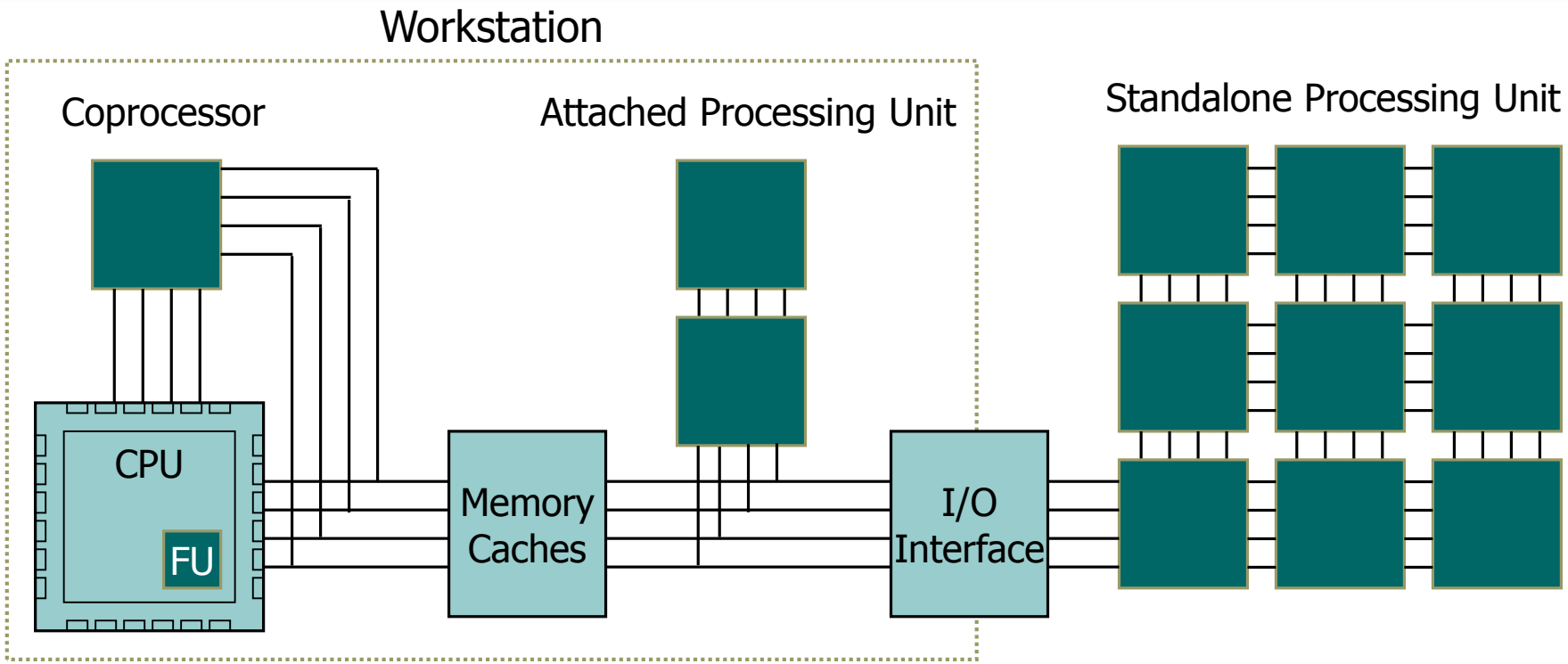
5) Hardware-in-the-loop integration
   - Various transceivers, clock generators, interfaces, ADCs, temperature sensors, etc.
   - Altera Starix, Xilinx Virtex II Pro Series (2002)

6) Scientific / engineering workloads
   - Floating-point cores
   - Altera Stratix 10 series (2014)

- Compelling application:
  - S. Vyas, C. Kumar, J. Zambreno, C. Gill, R. Cytron and P. Jones, "An FPGA-based Plant-on-Chip Platform for Cyber-Physical System Analysis", *IEEE Embedded Systems Letters (ESL)*, vol. 6, no. 1, 2014
  - Close integration with NIOS CPU, interfacing with sensors, actuators



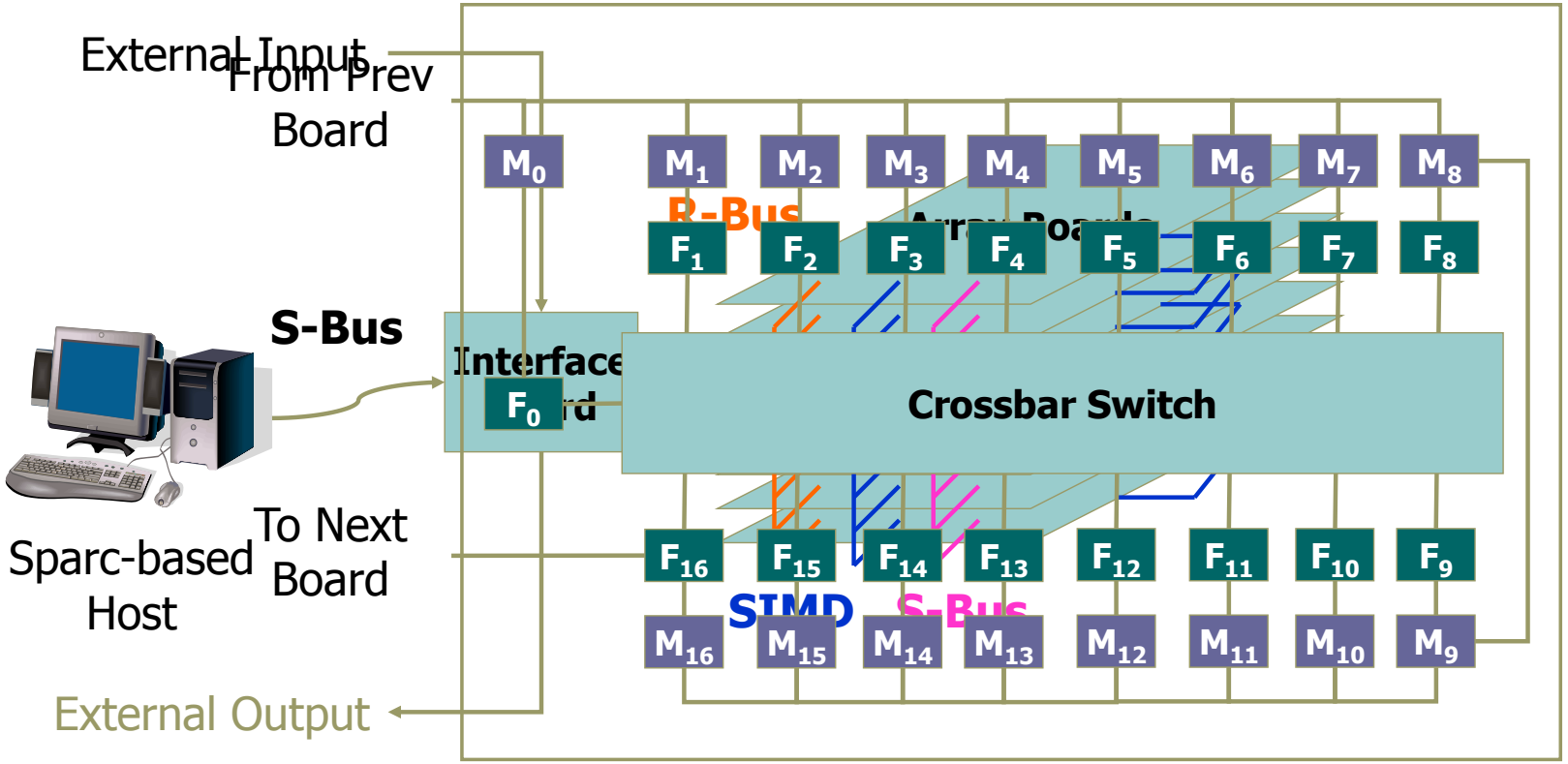
Accelerator Proximity



- Although self-reconfiguration is possible, some software integration with a reconfigurable accelerator is almost always present
- CPU – FPGA proximity has implications for programming model, device capacity, I/O bandwidth

Splash 2 (1993)

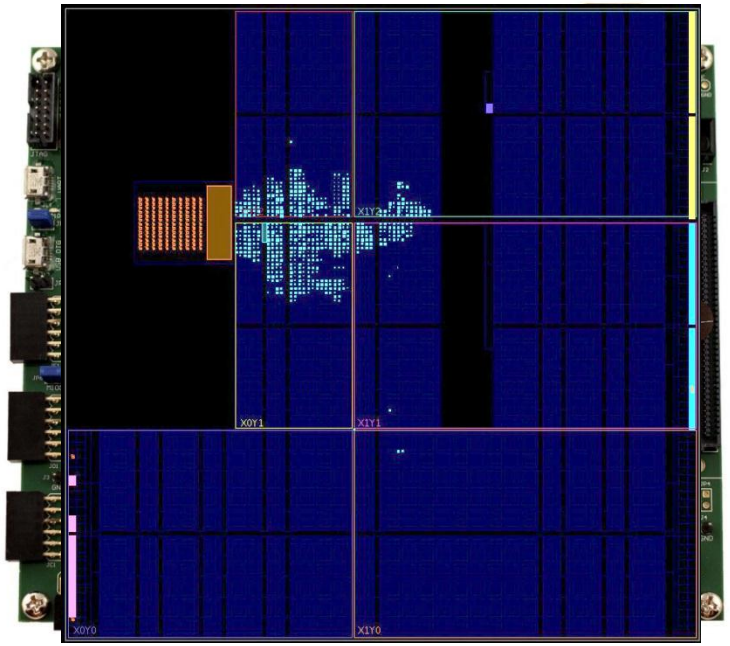
- J. Arnold et al., "The Splash 2 processor and applications," *Proc. Of the Int'l Conference on Computer Design (ICCD)*, Oct. 1993.
  - Attached to a Sparc-2 base, with 17 Xilinx XC4010 FPGAs, 17 512KB SRAMs, and 9 TI SN74ACT8841s



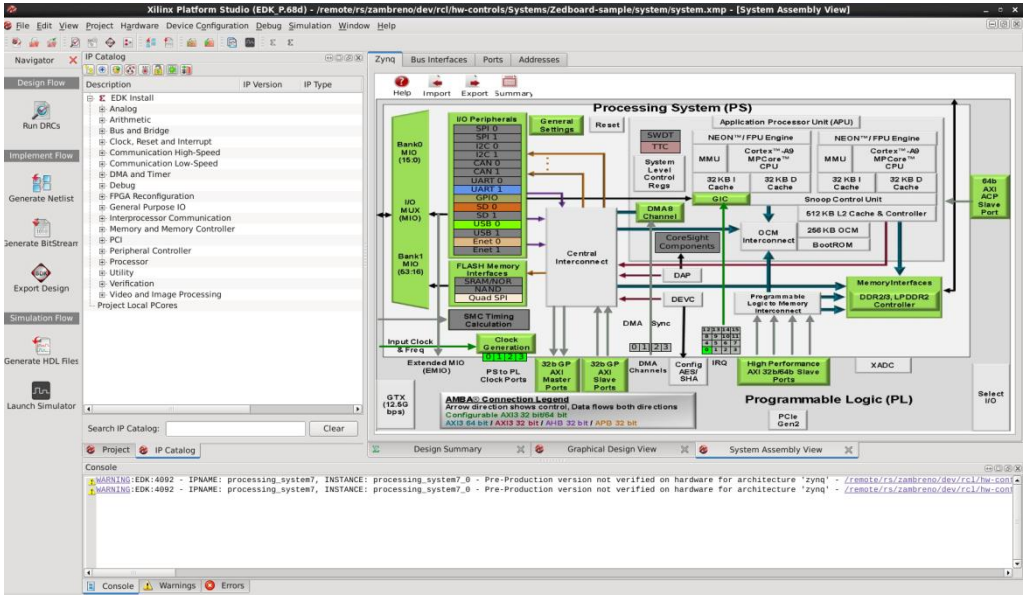
  - Development required up to 5 (!) programs, with code for the host CPU, interface board, F0, F1-F16, and the TI crossbar chips

Xilinx Zynq (2013)

- Coupling of dual-core ARM Cortex-A9 with reconfigurable logic
- "Processing System" is fully integrated and hardwired, and the platform can behave like a typical processor by default
- ARM CPU can control reconfiguration of programmable logic region
- Development environment is processor/IP-centric versus FPGA-centric



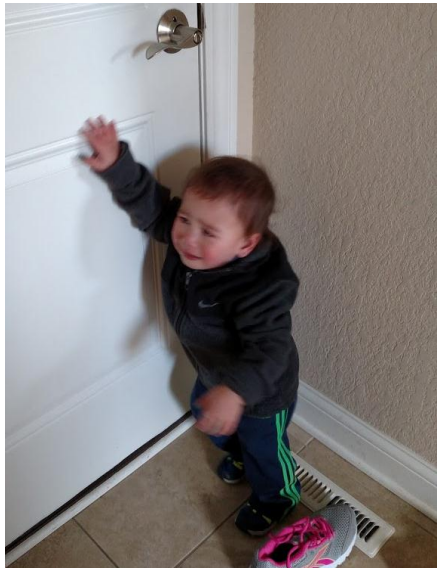
Digilent ZedBoard and Zynq-7000 FPGA



Xilinx Zynq System-level View in XPS

Automation to the Rescue?

- Developer efficiency continues to be a limiting factor
- Numerous approaches to the "behavioral synthesis" problem of generating useful hardware from high-level descriptions
- C-to-HDL variants:
  - Handel-C (Oxford)
  - ROCCC (UC-Riverside)
  - Catapult C (Mentor Graphics)
  - SystemC (Accelera)
  - Cynthesizer C (Cadence)
  - ImpulseC (Impulse)
- Many other comparable approaches:
  - HDL Coder (Mathworks)
  - Vivado High-Level Synthesis (Xilinx)
  - Bluespec, SystemVerilog



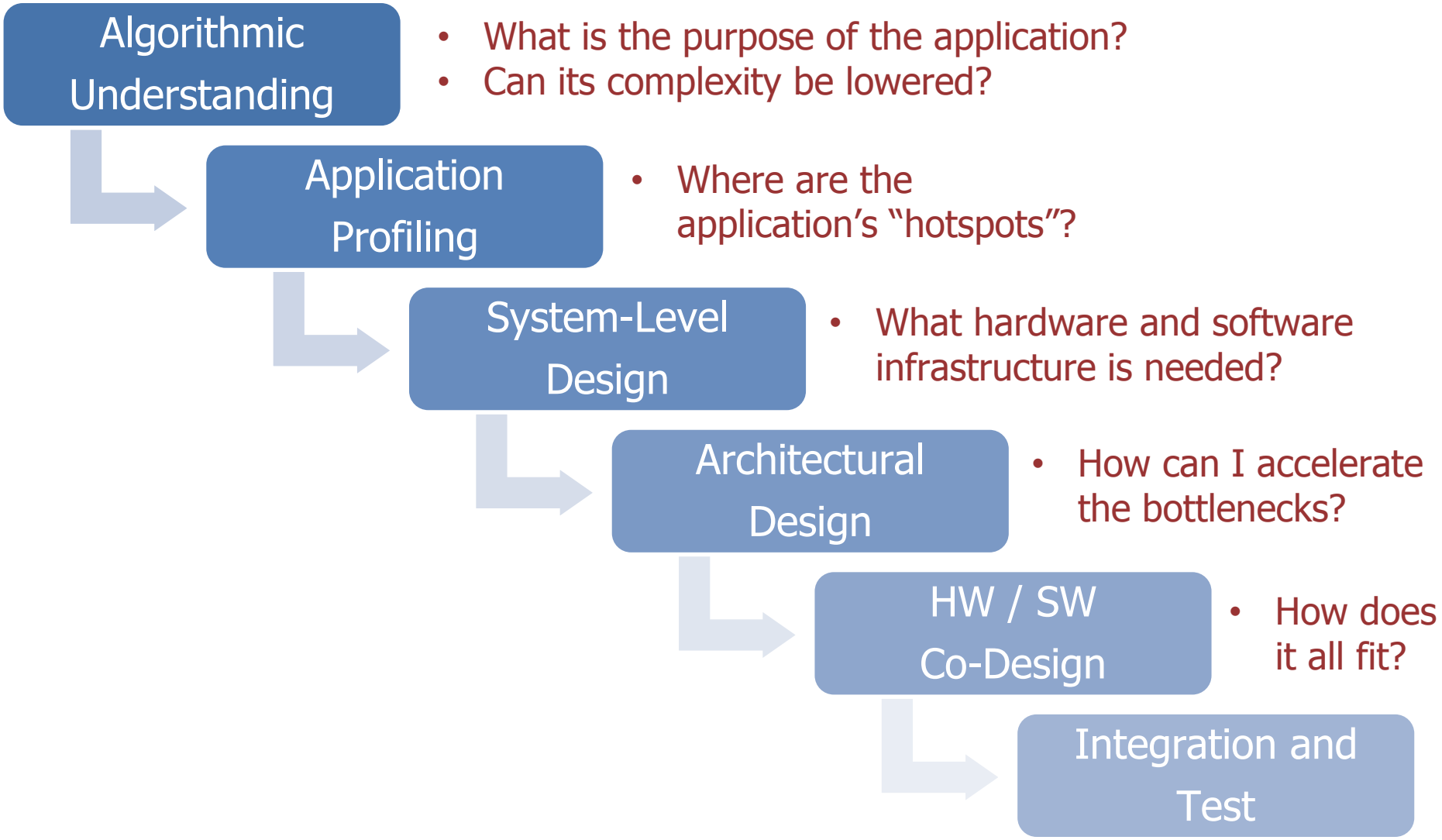
New RCL grad student seen trying to escape the lab

- Opinion: these tools can automate certain classes of logic, BUT:
  - Cannot generate efficient output for "hard problems"
  - Unfamiliar / uncomfortable syntax for both SW and HW engineers
  - Similar algorithmic challenges to auto-parallelizing compilers
  - Sorry students, you're still learning VHDL ☹

The Right Stuff

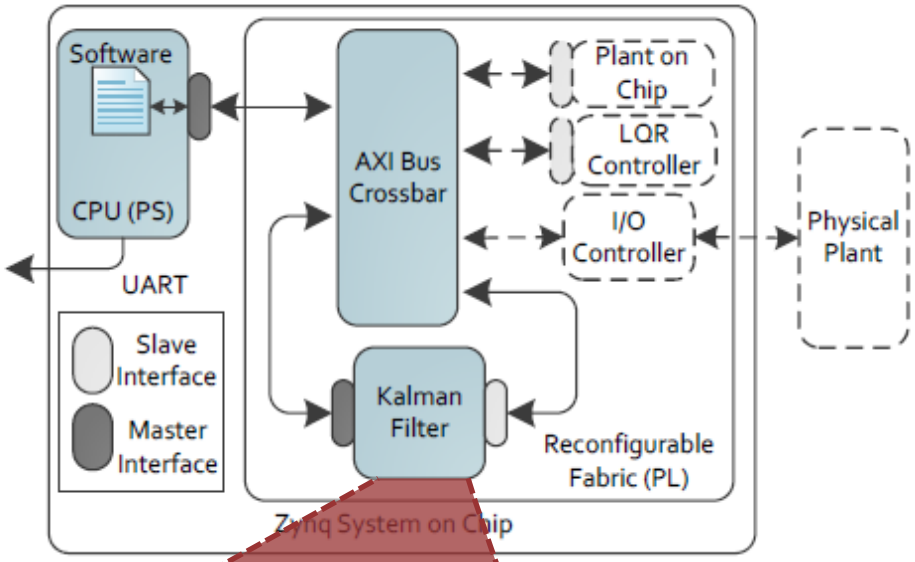
- Applications that map well to FPGA-based acceleration tend to have common characteristics:
  - Significant kernels of computation, significant data
    - Amdahl's law is typically more relevant than Gustafson's law
    - *Profile. Don't Speculate.* – Daniel Bernstein
  - Fixed-point or integer data representation
    - If application is Gflop-constrained, use a GPU
    - Changing (slowly), see Altera Stratix 10-based systems
  - Fine-grained parallelism
    - But if working set fits in cache, will still be hard to beat x86 (MHz for MHz)
    - Systolic model of computation, where FPGA pipeline depth > equivalent CPU depth and number of FPGA PEs >> number of x86 FUs
  - Real-time constraints, system integration
    - HPC workloads should go on HPCs (i.e. accelerators are an orthogonal concern)
    - Current GPUs cannot make useful latency guarantees

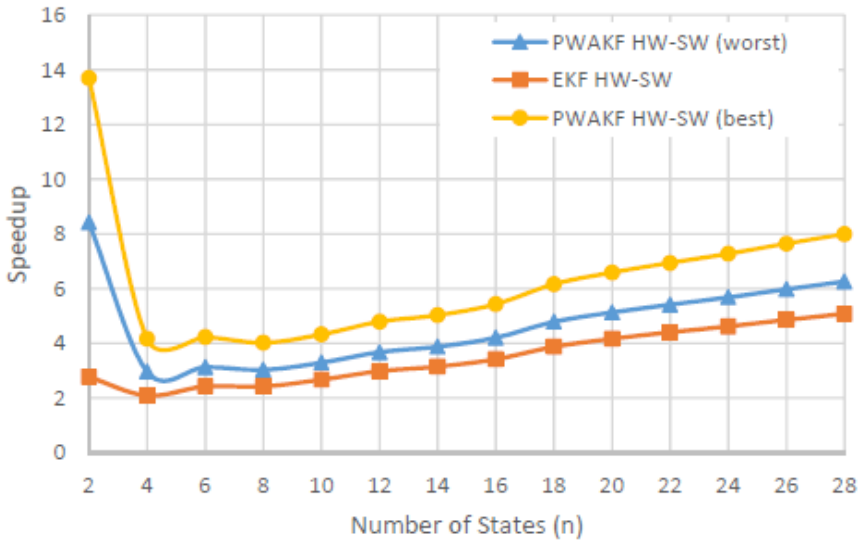
Typical Application Acceleration Methodology



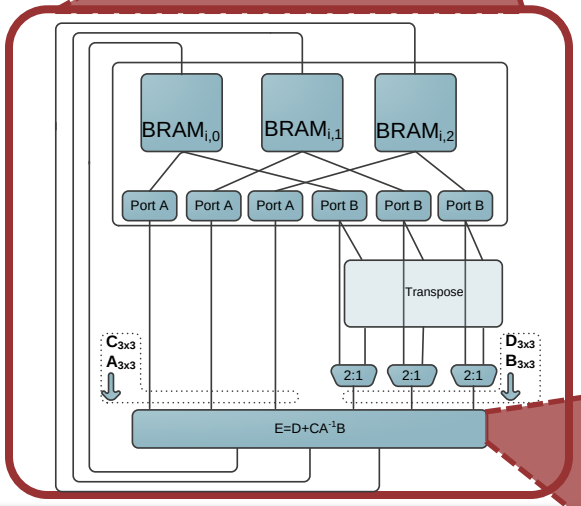
Ex. Project (1) – HW Accelerated Kalman Filtering

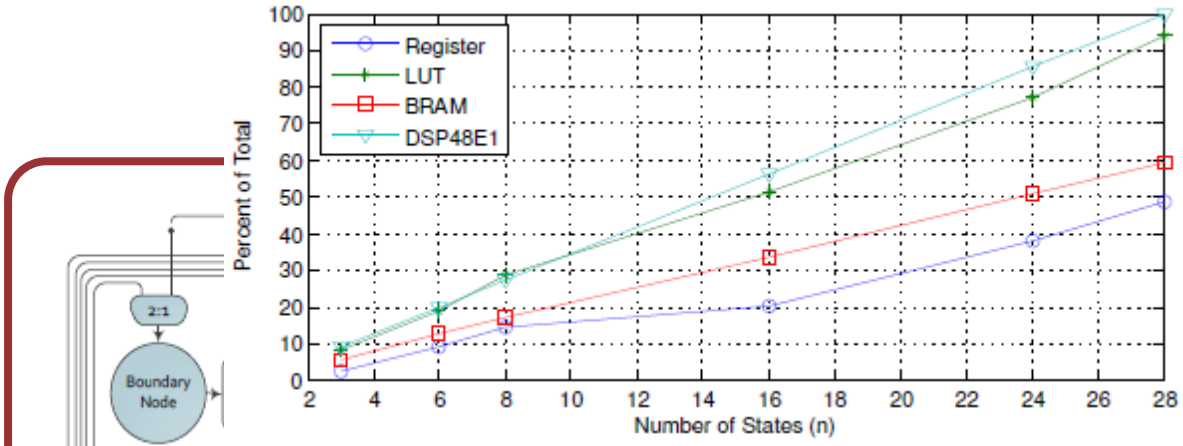
Piecewise-Affine Kalman Filter (PWAKF)





Significant speedups over SW equivalent

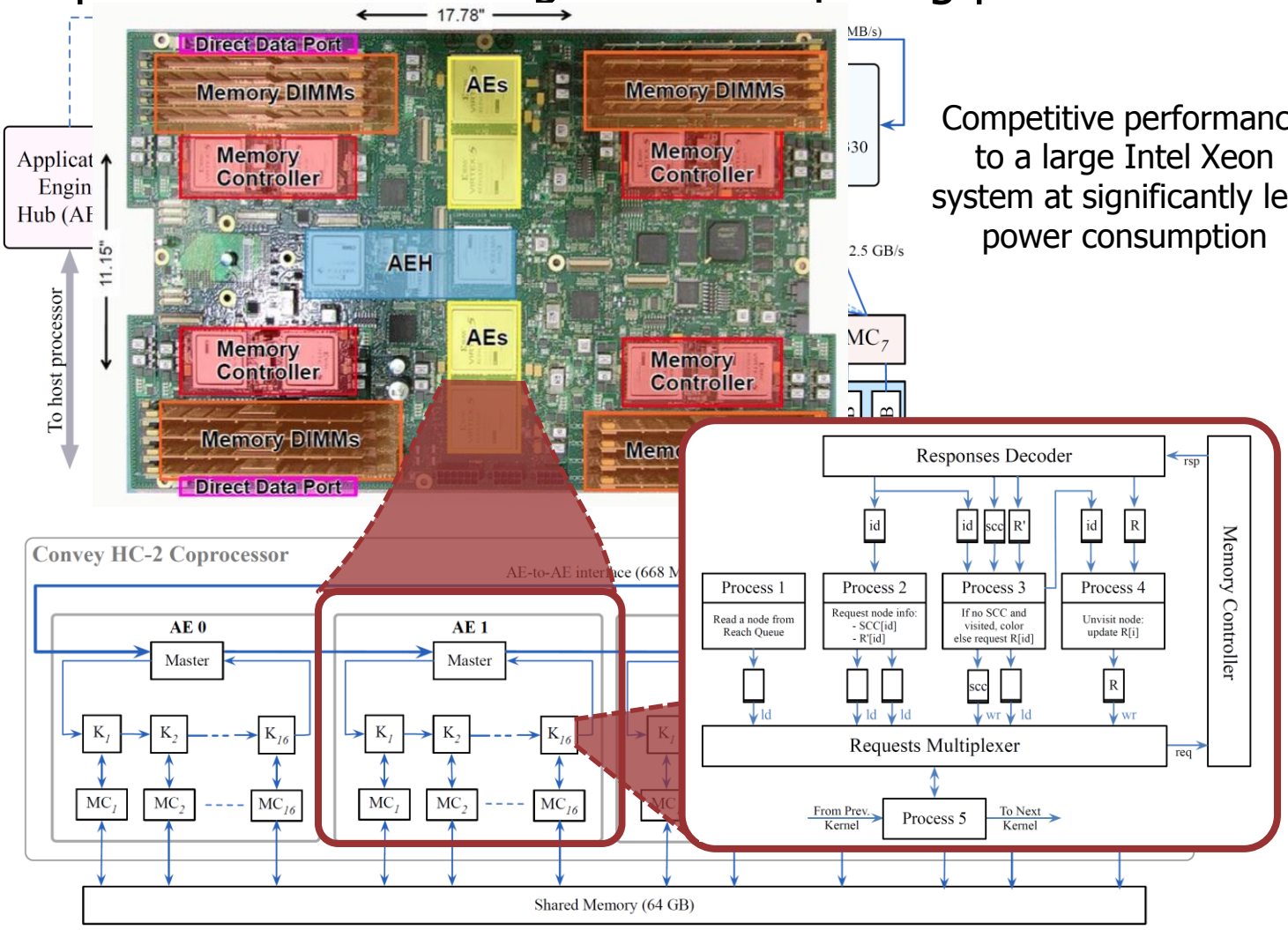




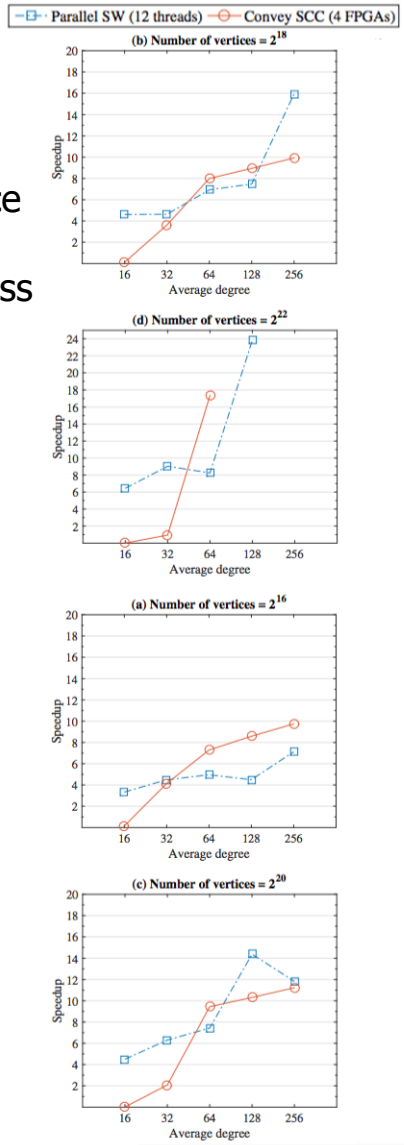
Resource usage grows linearly with model size

Ex. Project (2) – Accelerated Graph Processing

- Implementation of a bulk synchronous parallel model using a high performance reconfigurable computing platform

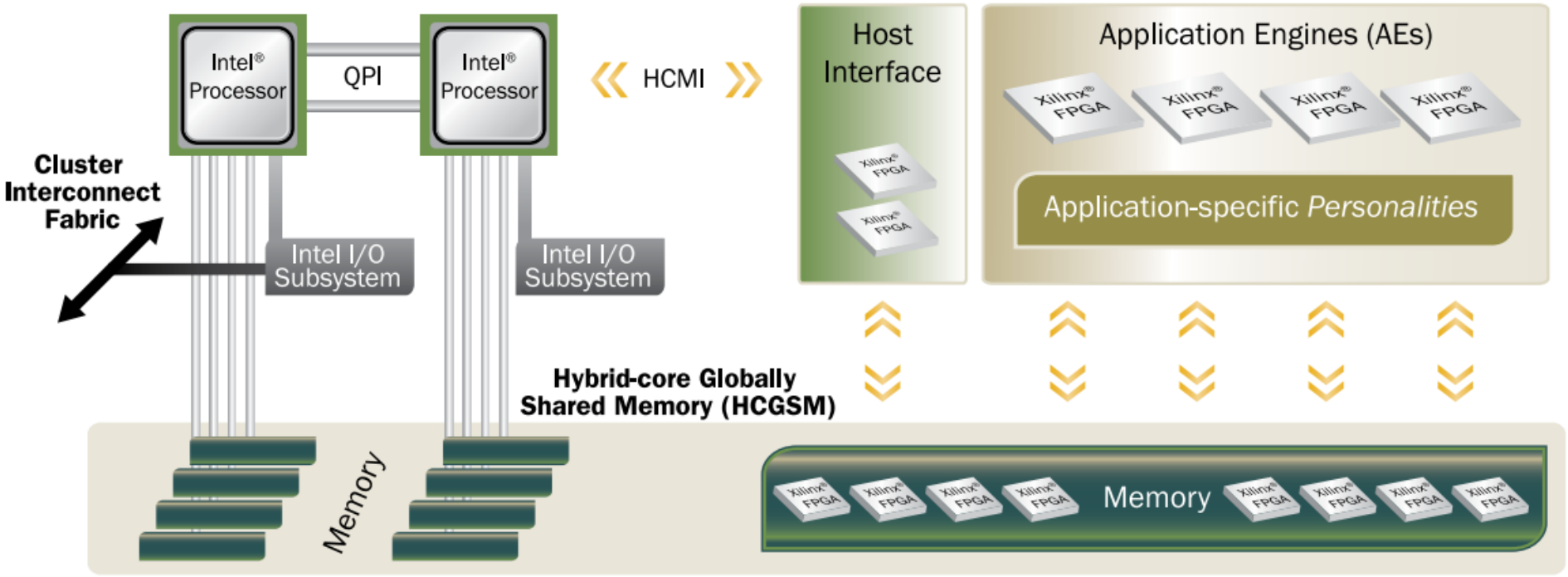


Competitive performance to a large Intel Xeon system at significantly less power consumption

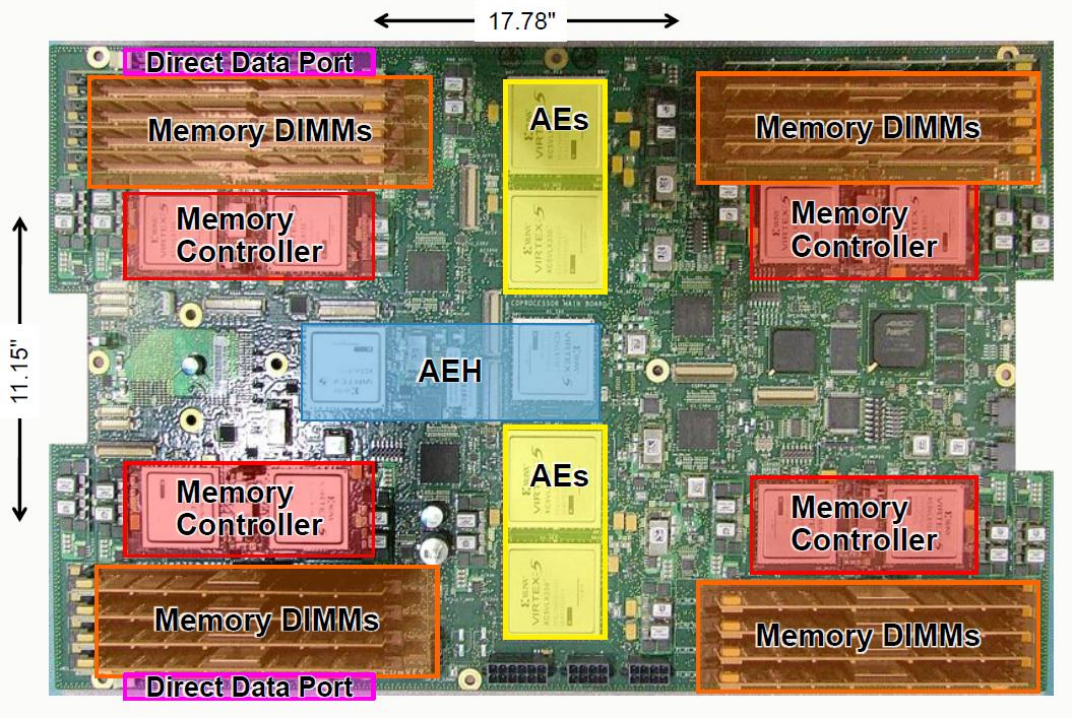


Our Reconfigurable Platform

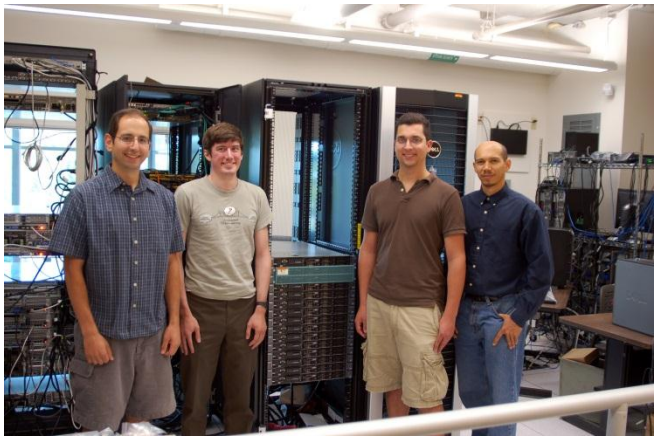
- Convey HC-2ex "hybrid core" computer:
  - Tightly coupled host-coprocessor via HCMI
  - Globally shared, high performance memory (80 GB/s)
  - Four Application Engines (AEs)
- Can be used as a large vector processor, or with custom "personalities" that accelerate arbitrary applications
  - Support for all interfacing logic, hardware/software co-simulation, early prototyping, performance profiling
  - Conventional C/C++ (host) and VHDL/Verilog (coprocessor) for development



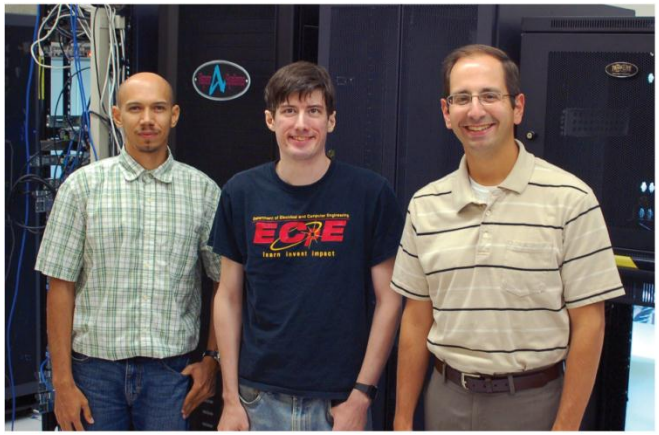
Other Views



"Iowa State University Students Win MemoCODE Design Contest Using Convey Computer HC-1", *MarketWire*, July 2012



"Team from Iowa State Wins 2014 MemoCODE Design Contest", *PRWeb*, Oct. 2014



- 14 FPGAs in total:
  - 4 FPGAs used as the AEs (Xilinx Virtex 5-LX330s operating at 150 MHz)
  - 8 FPGAs are used as Memory Controllers
  - 2 FPGAs are used as the Application Engine Hub to interface with the CPU subsystem

Ex. Project (3) – Real-Time Object Tracking

1. Test a previously developed approach for visually tracking markers mounted on a motor-grader blade
2. Modify as needed, port and accelerate the tracking algorithm to a Zynq-based development board
3. Create modular hardware IP cores for reuse in future projects
4. Demonstrate the real-world accuracy of the tracking
5. Integrate into existing SoC camera-logger-display platform
6. Field testing and evaluation

The Algorithm

"Camera" Model

- Read image
- Undistort image
- Convert to grayscale

Image Segmentation

- Perform adaptive thresholding

Object Pruning

- Mask image using previous location
- Remove too large / small objects

Marker Detection

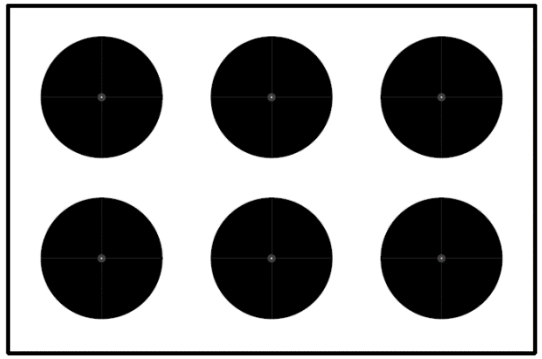
- Find contours
- Fit ellipses to contours
- Find most appropriate ellipses (<=6)

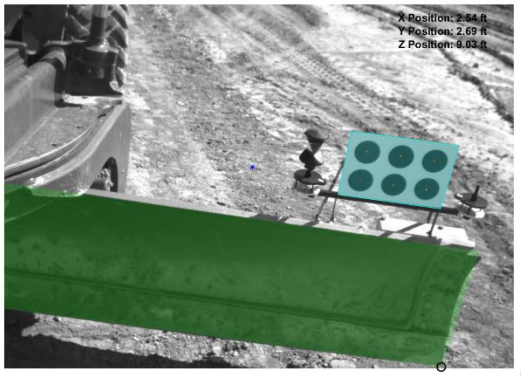
Found 6 centers?

No

Yes

- Reproject centers into image
- Find blade location





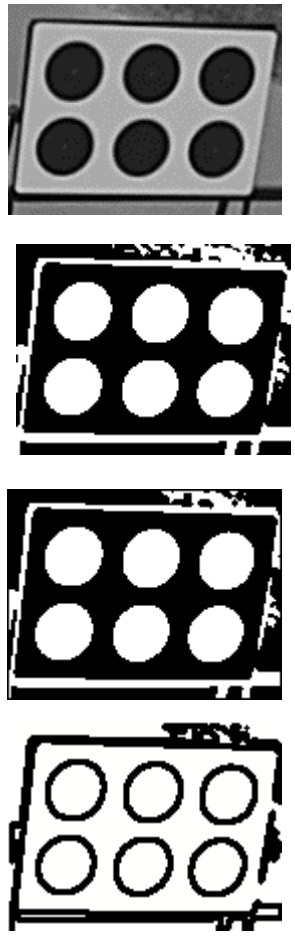
- Sara Beery, "Tracking the Blade of a Motor Grader: A Vision Based Solution". John Deere Intelligent Solutions Machine Automation Group, Sep 2015.

Algorithm Visualization

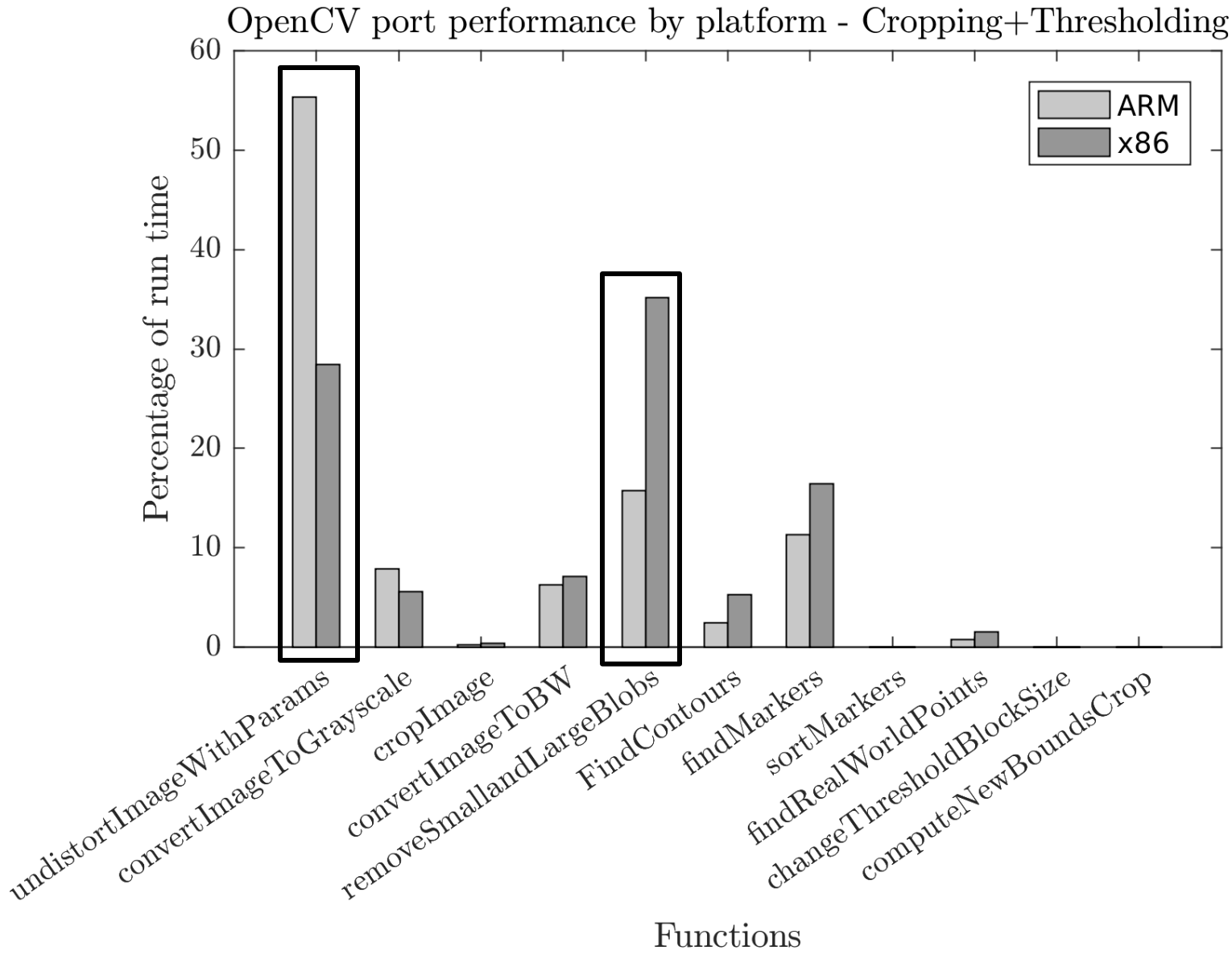
1. Input image (from file)
2. Lens undistortion
3. Grayscale conversion
4. Adaptive thresholding
5. Object removal
6. Contours
7. Detected markers

Our Initial Observations

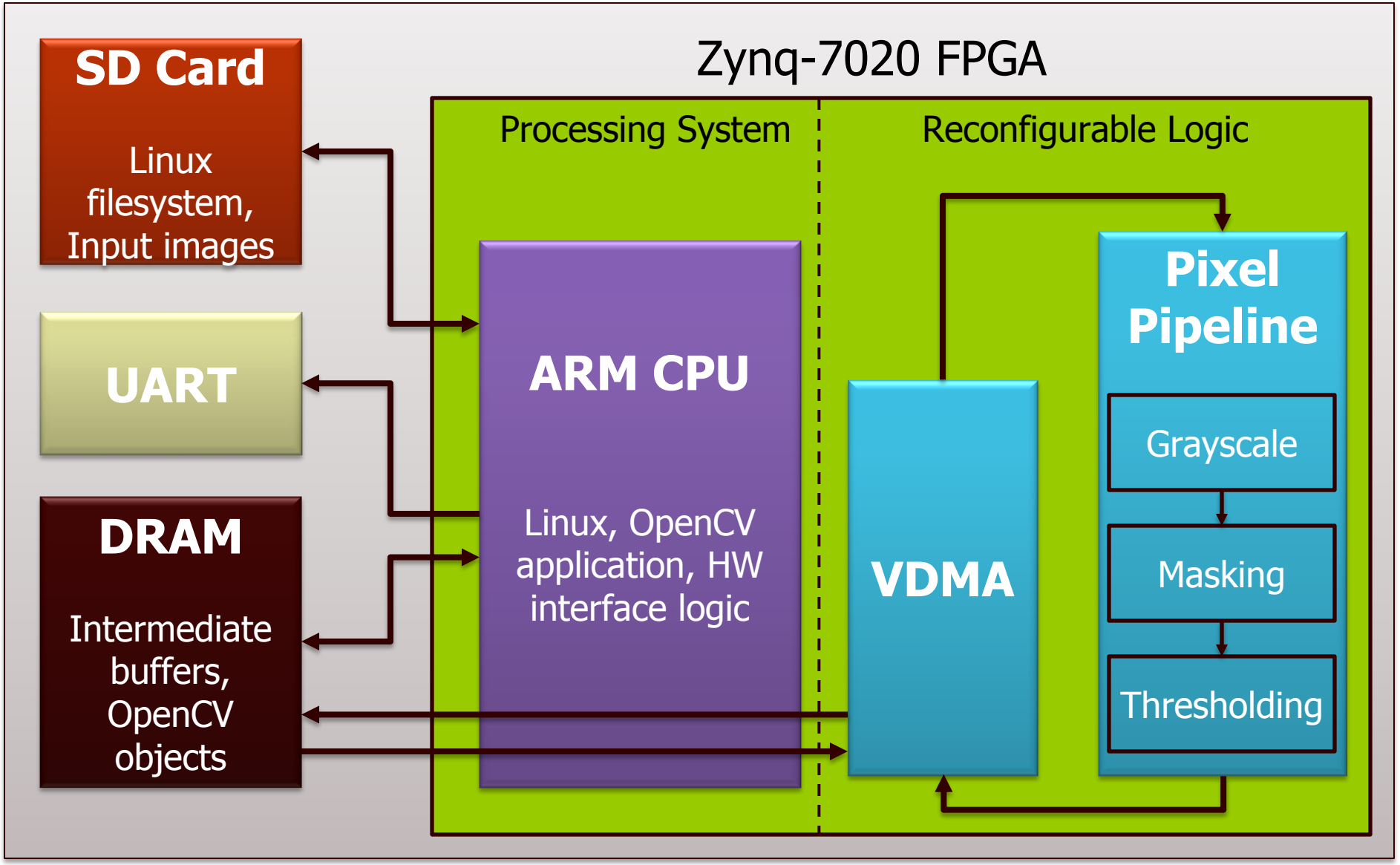
- Matlab prototype made heavy use of the Computer Vision toolbox, so conversion to OpenCV (C/C++) for an embedded platform was relatively straightforward
- Initial performance was quite slow, but with several opportunities for early optimization
  - Image masking / cropping can be done earlier and much more aggressively
  - Simpler adaptive thresholding algorithms can be applied with good accuracy / performance tradeoffs
  - Reading image files from disk a likely bottleneck, but can be generally ignored as project intent was to target SoC camera platform (reading straight from sensor)



Profiling: Version 1.2

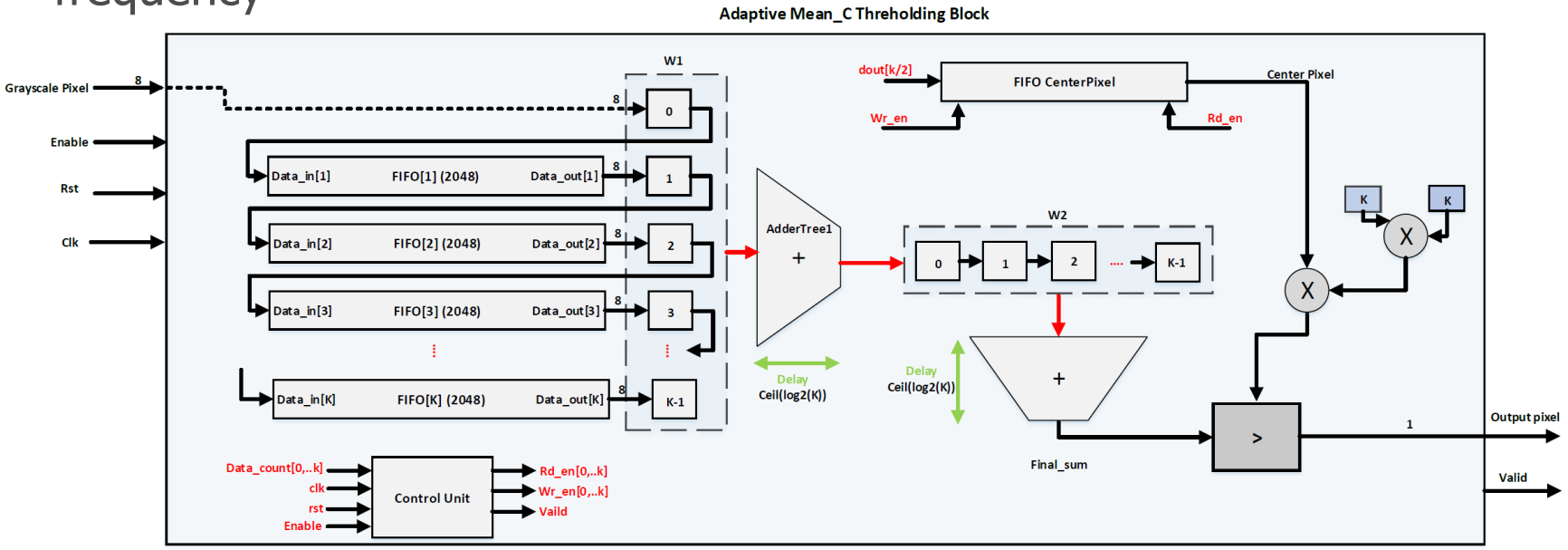


System-Level Design (Zedboard)

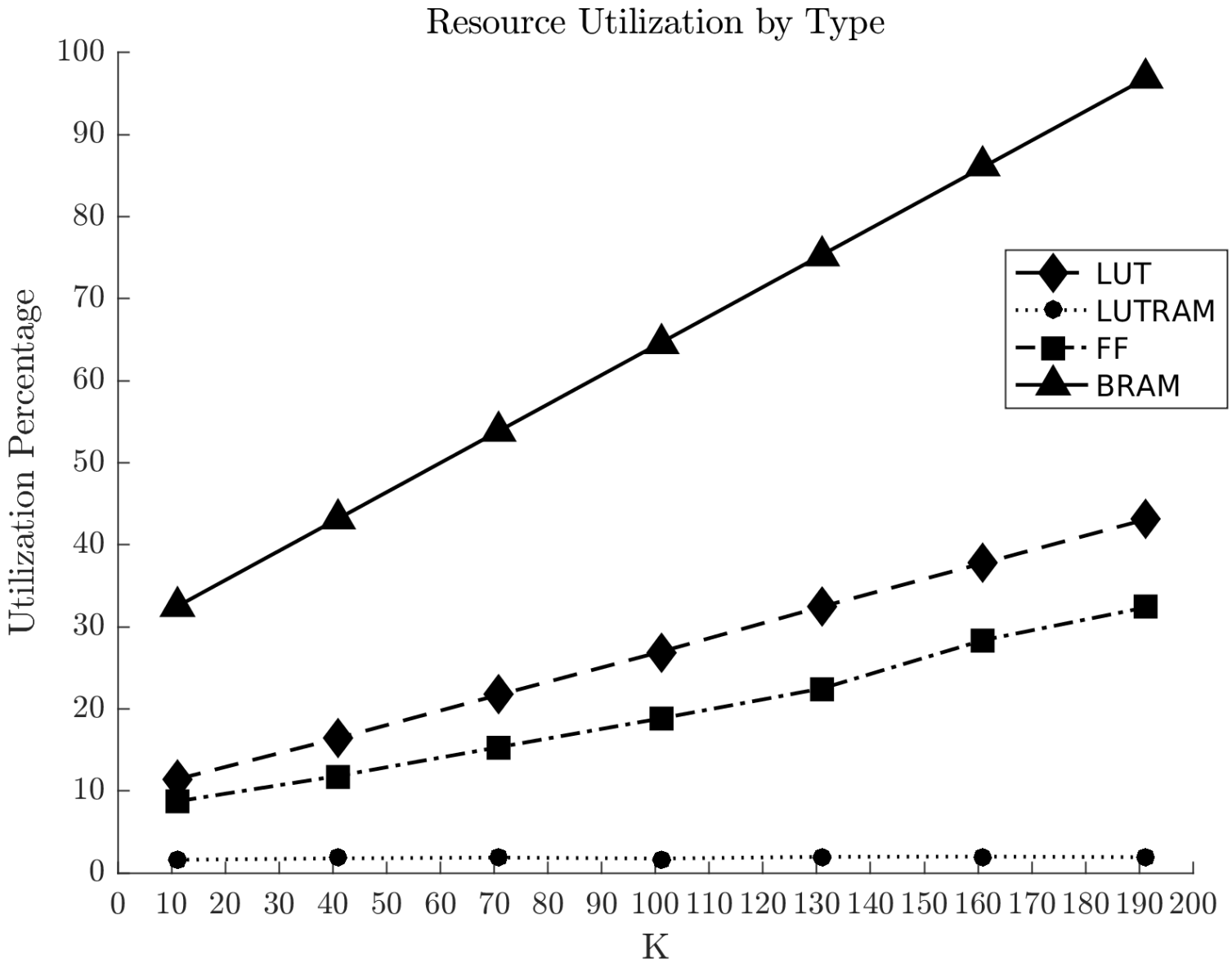


Architectural Design

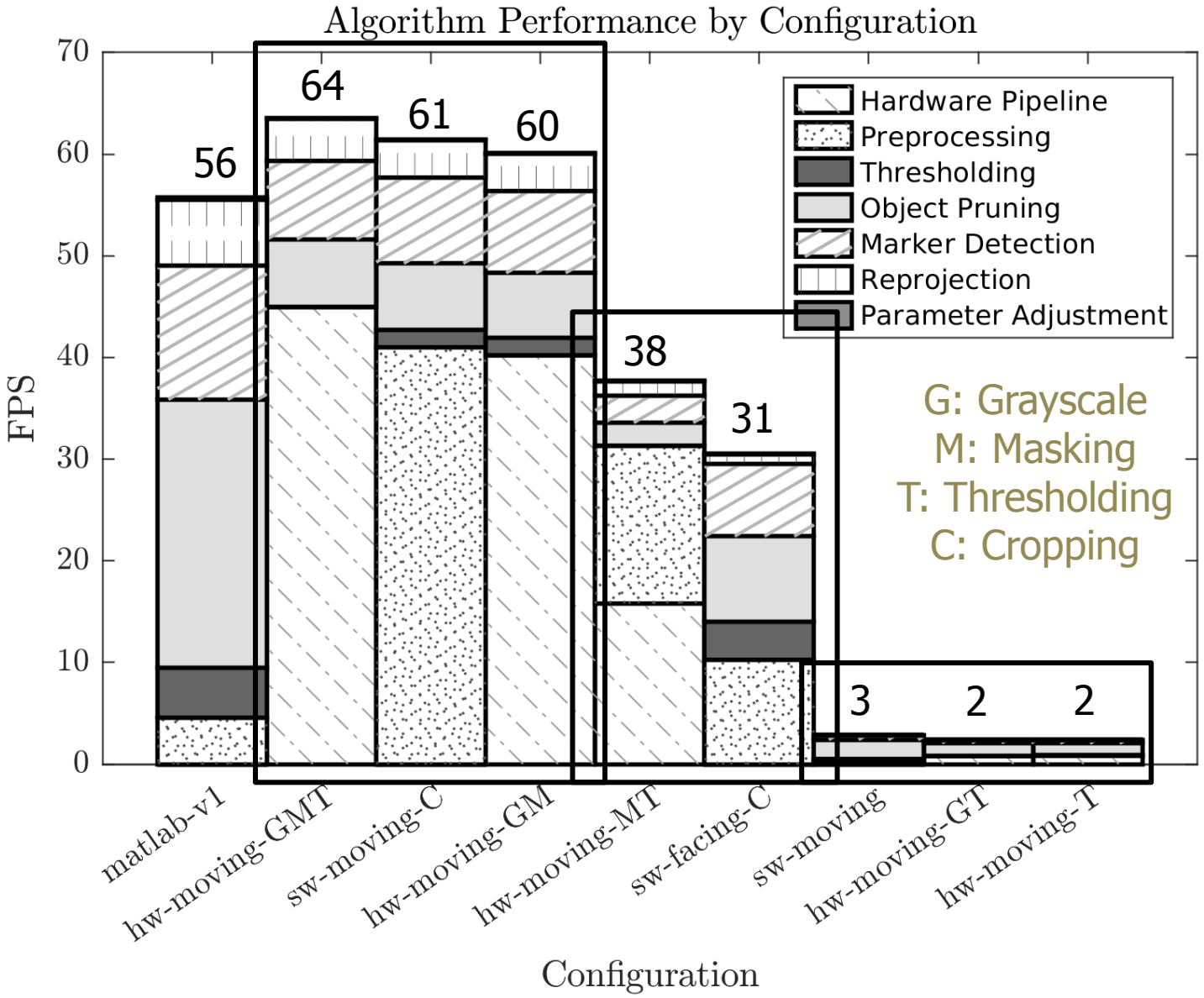
- Grayscale conversion and masking are relatively straightforward
- Sliding window architecture for adaptive thresholding – applicable to many different image processing algorithms
- Advantages:
  - Software configurable for different resolutions, window sizes
  - High performance (1 pixel per clock), heavily pipelined for clock frequency



Resource Utilization

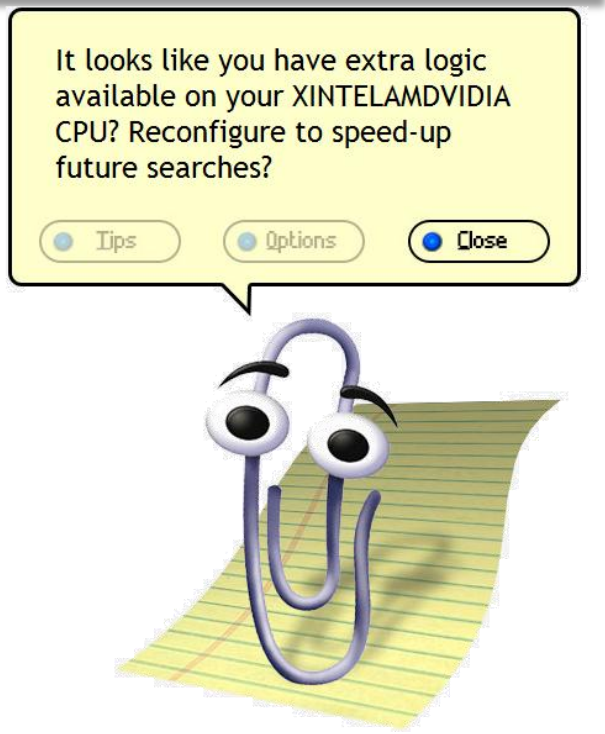


Performance Results



What Does the Future Hold?

- Continued device specialization
  - Blurred lines between FPGAs, GPUs, and CPUs
  - X floating-point units, Y other accelerators, and Z memory controllers with some software-configurable layout and multi-tasking model
- Deeper integration - reconfigurable logic in (more) commodity chips
- Use in embedded applications continues to grow
  - Dynamic power versus performance tradeoffs
- Slow adoption in HPC space
  - Graph 500 (now), Green 500 (soon), Top 500 (eventually)



Hottest Windows 11 feature

Acknowledgments

- These slides are inspired in part by material developed and copyright by:
  - Marilyn Wolf (Georgia Tech)
  - Jason Bakos (University of South Carolina)
  - Greg Stitt (University of Florida)
  - Hadi Esmaeilzadeh (Georgia Tech)