

CprE 488 – Embedded Systems Design

Lecture 8 – Hardware Acceleration

Phillip Jones

Electrical and Computer Engineering

Iowa State University

www.ece.iastate.edu/~phjones

rcl.ece.iastate.edu

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

Motivation: Moore's Law

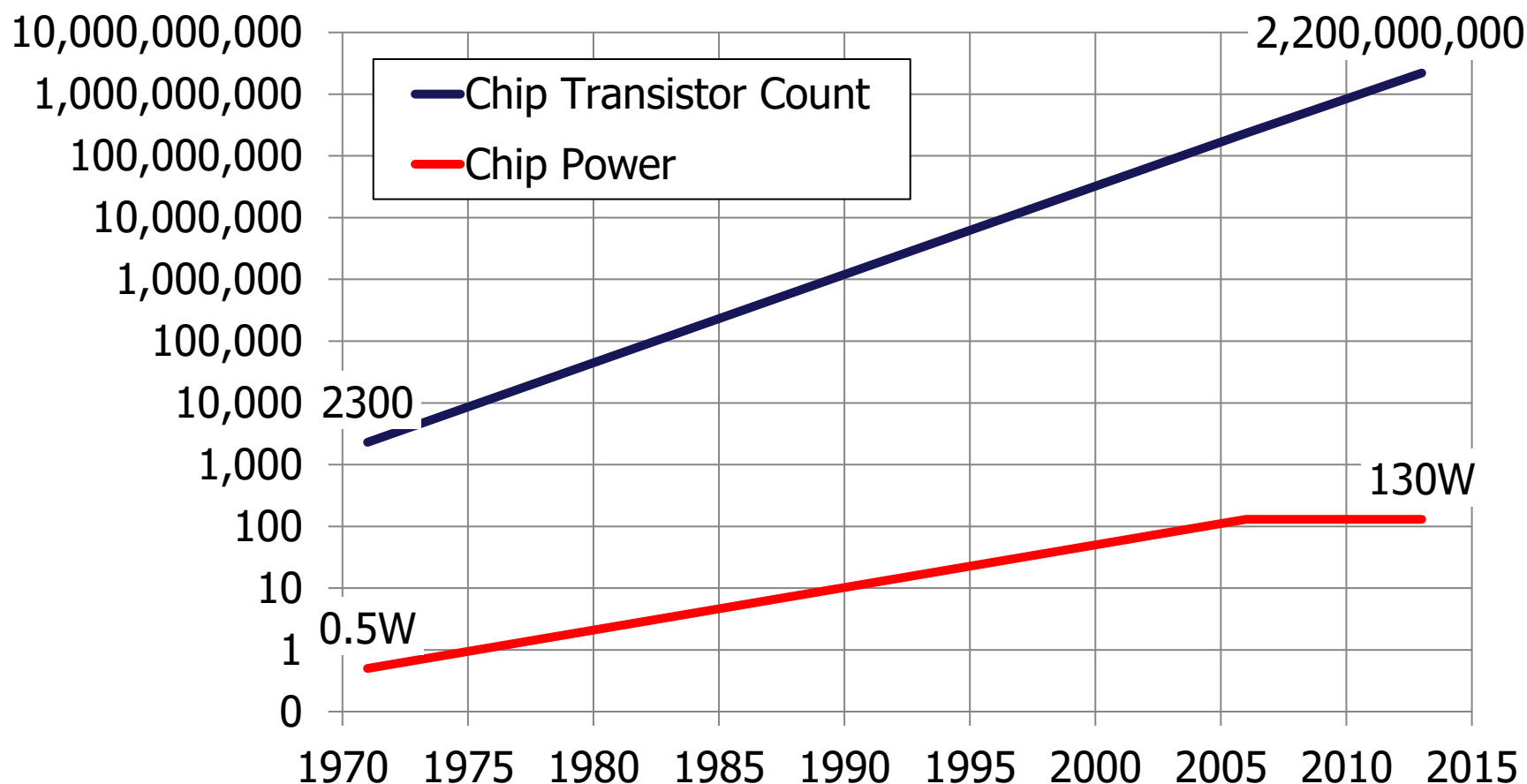
- Every two years:
 - Double the number of transistors
 - Build higher performance general-purpose processors
 - Make the transistors available to the masses
 - Increase performance ($1.8\times\uparrow$)
 - Lower the cost of computing ($1.8\times\downarrow$)
- Sounds great, what's the catch?



Gordon Moore

Motivation: Moore's Law (cont.)

- The “catch” – powering the transistors without melting the chip!
 - See 2003 – 2004 news: <https://wccftech.com/intel-tejas-and-jayhawk-the-story-of-the-abandoned-intel-7-ghz-pentium-5-chips/>



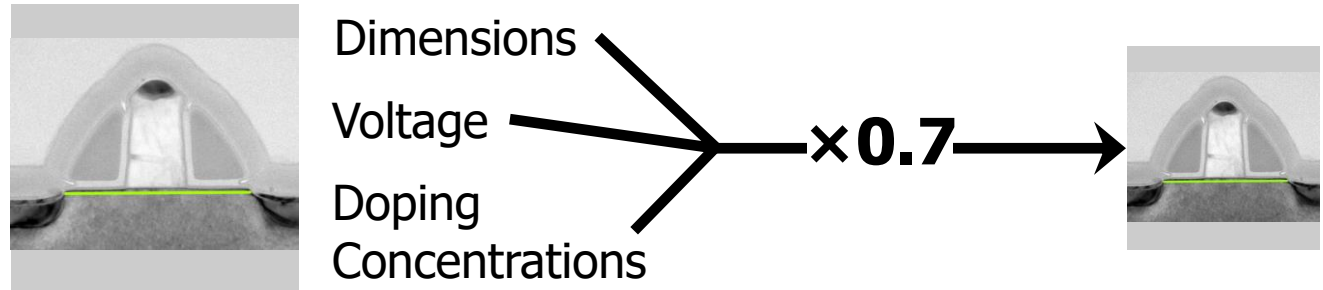
Motivation: Dennard Scaling

- As transistors get smaller their power density stays constant



Robert Dennard

Transistor: 2D Voltage-Controlled Switch



Area $\xrightarrow{\quad\quad\quad} 0.5\times \downarrow \xrightarrow{\quad\quad\quad}$

Capacitance $\xrightarrow{\quad\quad\quad} 0.7\times \downarrow \xrightarrow{\quad\quad\quad}$

Frequency $\xrightarrow{\quad\quad\quad} 1.4\times \uparrow \xrightarrow{\quad\quad\quad}$

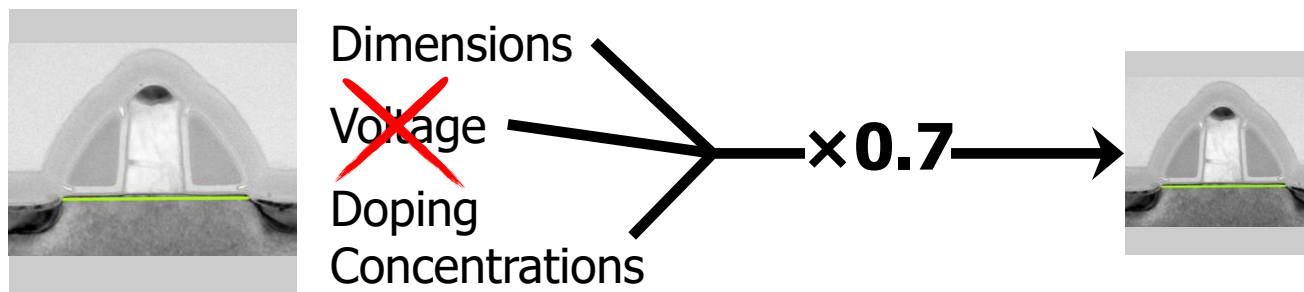
$$\text{Power} = \text{Capacitance} \times \text{Frequency} \times \text{Voltage}^2$$

Power $\xrightarrow{\quad\quad\quad} 0.5\times \downarrow \xrightarrow{\quad\quad\quad}$

Motivation Dennard Scaling (cont.)

- In mid 2000s, Dennard scaling "broke"

Transistor: 2D Voltage-Controlled Switch



Area $\xrightarrow{0.5 \times \downarrow}$

Capacitance $\xrightarrow{\cancel{0.7 \times \downarrow}}$

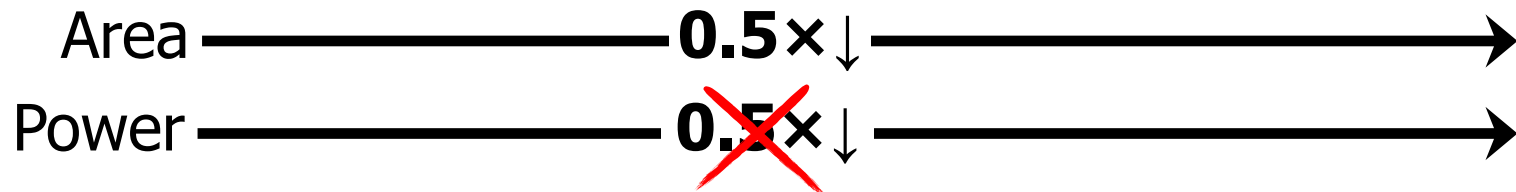
Frequency $\xrightarrow{\cancel{1.4 \times \uparrow}}$

$$\text{Power} = \text{Capacitance} \times \text{Frequency} \times \text{Voltage}^2$$

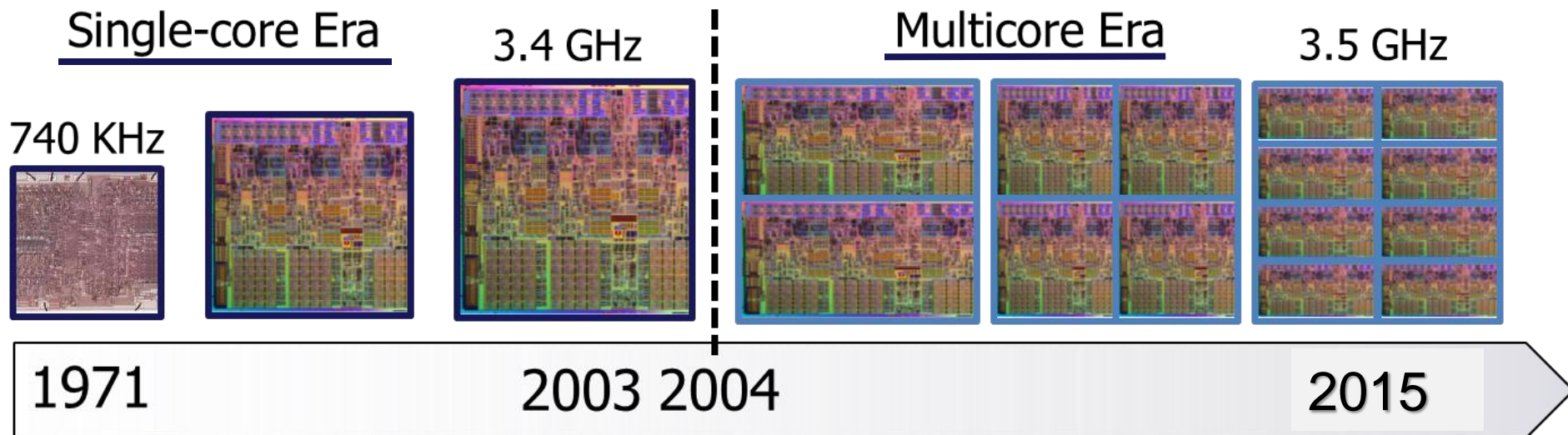
Power $\xrightarrow{\cancel{0.5 \times \downarrow}}$

Motivation: Dark Silicon

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



- Processor evolution strongly motivated by Dennard scaling ending
 - Expected continued evolution towards HW specialization/accel



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

And Today?

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

Microsoft to accelerate Bing search with neural network

By John Hewitt on February 25, 2016 at 3:46 pm | 19 Comments



Share This Article



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:



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



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.

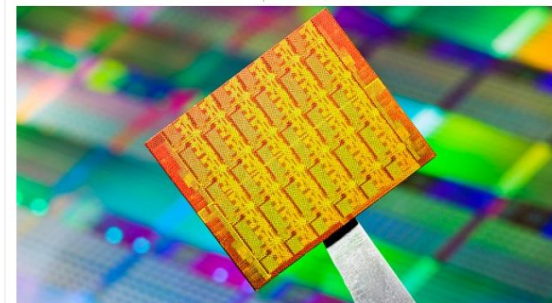
It turns out that IBM is launching three different systems tuned up for three different kinds of workloads that are based on its "scale-out" Power8 systems. By scale-out, IBM means a system is designed with one or two sockets and is intended to be used in clusters that have distributed applications that 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

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



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

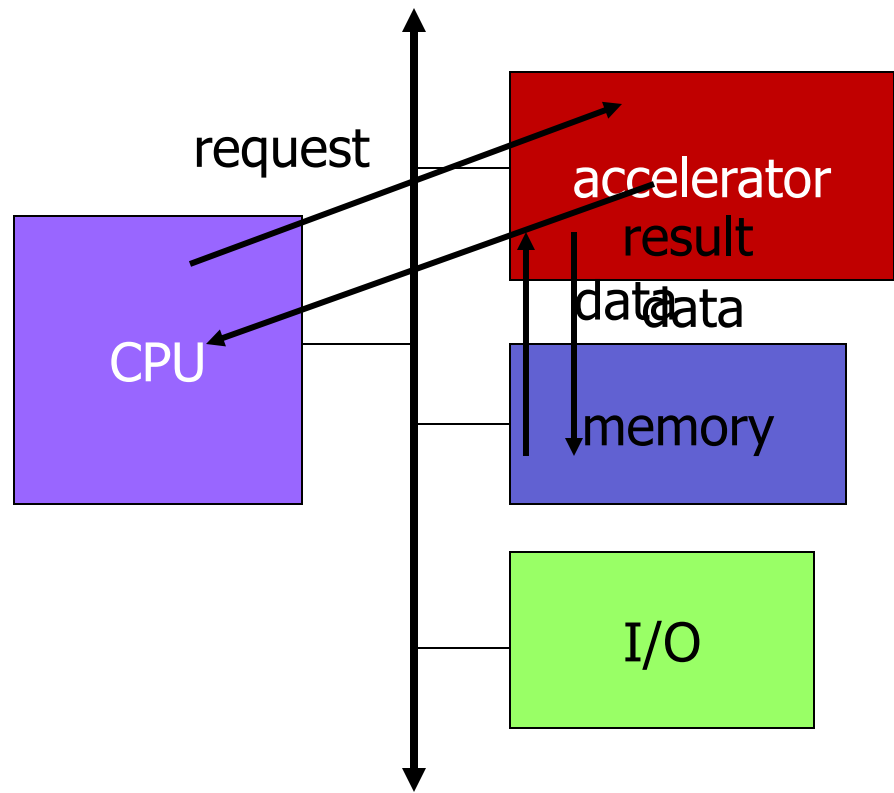
Xeon+FPGA chip will fit in the standard E5 LGA2011 socket, but the integrated FPGA will allow each chip to be customized to specific workloads. This move is almost certainly intended to make Intel-x86 a better all-round platform for a wider variety of workloads in enterprise and data center settings, 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'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?

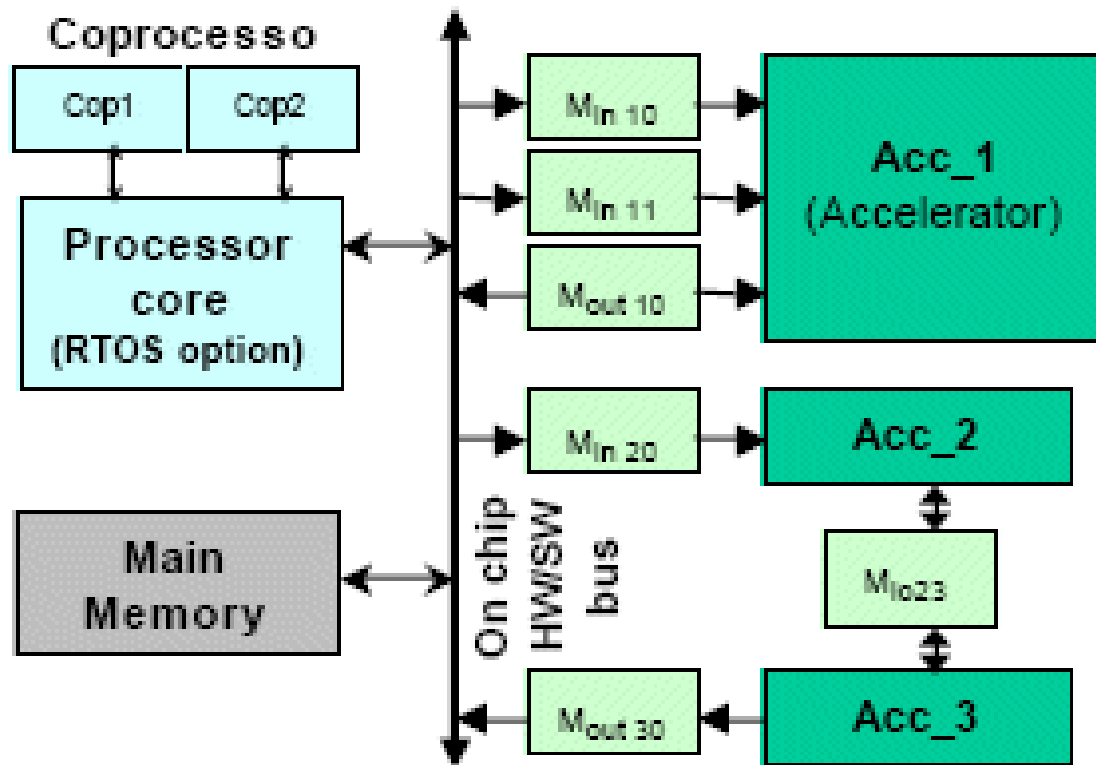
Accelerated Systems

- Additional computational units dedicated to some functionality
- **Hardware/software co-design**: joint design of HW & SW architect

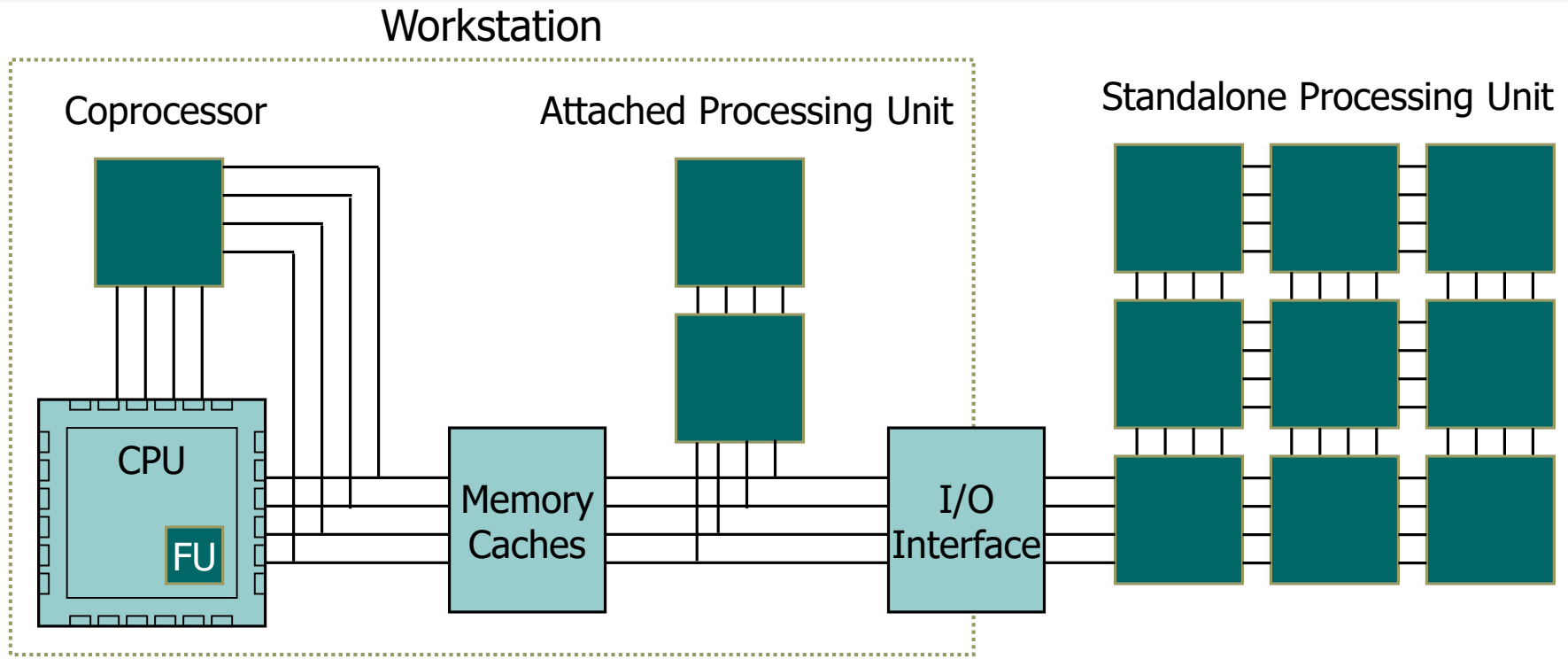


Accelerator vs. Coprocessor

- 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 I/O)



Accelerator Proximity



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

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

Will Hardware Acceleration Help?

Amdahl's Law

- The total application speedup, when an optimization (accelerator) improves a selected fraction (f) of an application's execution time by a factor a is:

$$\textit{Application_Speedup} = \frac{T_{org}}{[(1 - f) + f/a] * T_{org}} = \frac{1}{(1 - f) + f/a}$$



Gene Amdahl

Will Hardware Acceleration Help?

$$Application_Speedup = \frac{T_{org}}{T_{new}} = \frac{T_{org}}{T_{org} - (T_f - T_{fa})}$$

where:

- T_{org} : Application original exec. time
- T_{new} : Application exec. time after accel
- f : fraction of the Application time that a **selected portion** of the App. runs.
- T_f : amount of time the **selected portion** of the App. runs, before accel.
- T_{fa} : amount of time the **selected portion** of the App. runs after accelerated by a
- a : factor by which accelerator speeds up **selected portion** of Application

$$= \frac{T_{org}}{T_{org} - (T_f - T_f/a)}$$

$$= \frac{T_{org}}{T_{org} - ([f * T_{org}] - [f * T_{org}]/a)}$$

$$= \frac{T_{org}}{(1 - ([f] - [f]/a)) * T_{org}}$$

$$= \frac{T_{org}}{(1 + (-[f] + [f]/a)) * T_{org}}$$

$$Application_Speedup = \frac{T_{org}}{[(1 - f) + f/a] * T_{org}} = \frac{1}{(1 - f) + f/a}$$

Amdahl's Law

- The total application speedup, when an optimization (accelerator) improves a selected fraction (f) of an application's execution time by a factor a is:

$$\text{Application_Speedup} = \frac{T_{org}}{[(1 - f) + f/a] * T_{org}} = \frac{1}{(1 - f) + f/a}$$

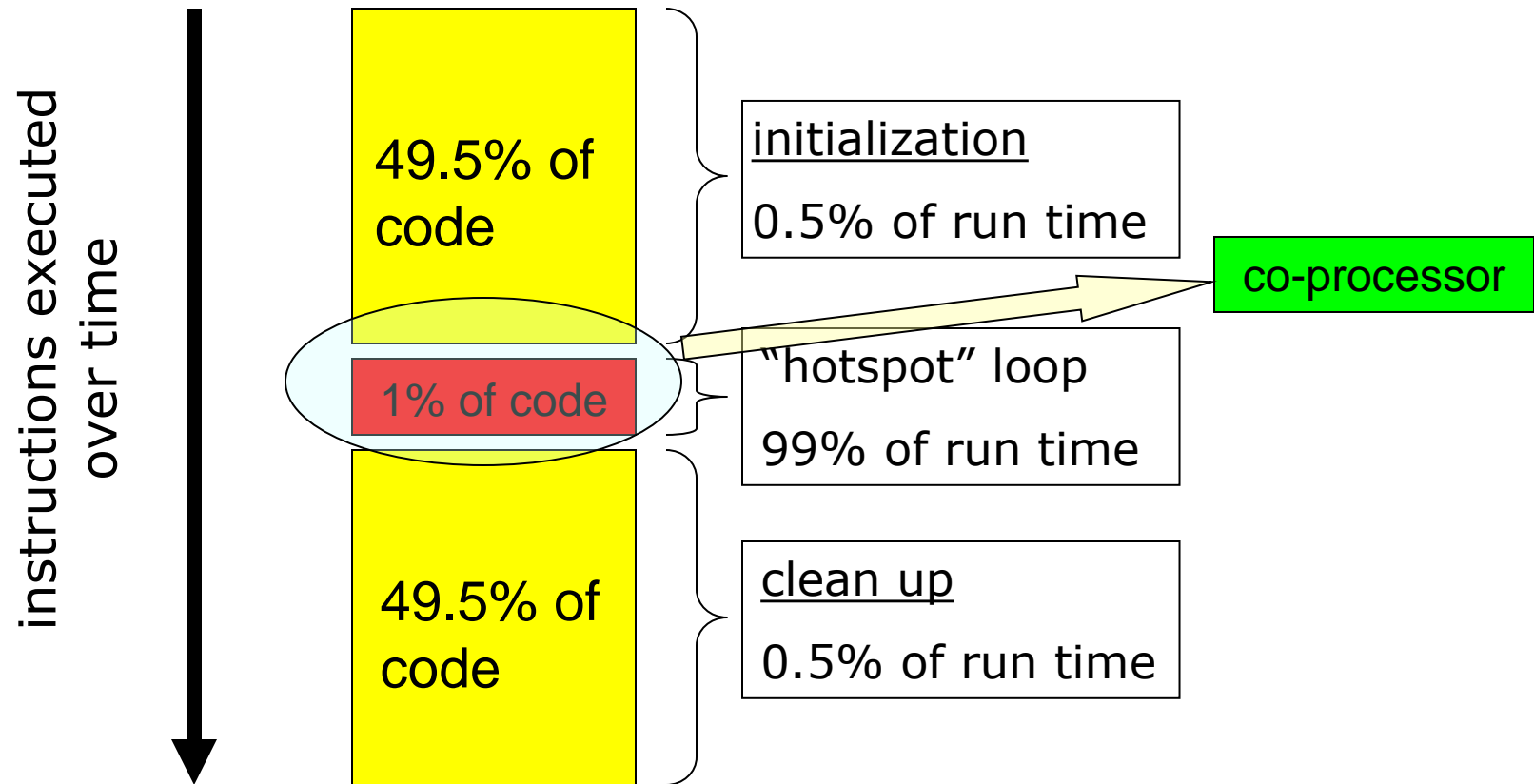
- This formula is known as Amdahl's Law, where:
 - T_{org} is the execution time of the whole Application before any optimization/accelerator (i.e., original execution time)
 - f is the fraction of the Application time that a **selected portion** of the Application takes to run.
 - a is the factor by which an optimization/accelerator speeds up the **selected portion** of the Application
- Lessons from this law:
 - If $f \rightarrow 1$ (i.e., 100%), then $\text{Application_Speedup} = a$
 - If $a \rightarrow \infty$, then $\text{Application Speedup} = 1 / (1 - f)$
- Summary
 - Make the common case fast
 - Watchout for serial parts of an Application (ie, parts that cannot be accelerated)



Gene Amdahl

Heterogeneous Execution Model

- Example: Assume an Application where only 1% of the code runs for 99% of the Application execution time (i.e., $f = .99$). How much will the overall Application speedup, if we create an accelerator to speedup this selected portion (i.e., "hotspot") by a



Heterogeneous Computing: Performance

- Move “**bottleneck/hotspot**” computation from software to hardware
- Example:
 - Application requires a **week** of CPU time (i.e., 168 hours)
 - One “**hotspot**” computation consumes **99%** of execution time

Hotspot Speedup(a)	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

Heterogeneous Computing: Performance

- Move “**bottleneck/hotspot**” computation from software to hardware
- Example:
 - Application requires a **week** of CPU time (i.e., 168 hours)
 - One “**hotspot**” computation consumes **99%** of execution time

Design Time	Hardware Resources	Hotspot Speedup(a)	Application speedup	Execution time
1-week	1-FPGA	50	34	5.0 hours
2-months	2-FPGA	100	50	3.3 hours
6-months	4-FPGA	200	67	2.5 hours
2-years	9-FPGA	500	83	2.0 hours
4-years	20-FPGA	1000	91	1.8 hours

Is the effort and resources needed to create the Hotspot accelerator worth the corresponding Application speedup?

Heterogeneous Computing: Performance

- Move “**bottleneck/hotspot**” computation from software to hardware
- Example:
 - Application requires a **week** of CPU time (i.e., 168 hours)
 - One “**hotspot**” computation consumes **99%** of execution time

Design Time	Hardware Resources	Hotspot Speedup(a)	Application speedup	Execution time
1-week	1-FPGA	50	34	5.0 hours
2-months	2-FPGA	100	50	3.3 hours
6-months	4-FPGA	200	67	2.5 hours
2-years	9-FPGA	500	83	2.0 hours
4-years	20-FPGA	1000	91	1.8 hours

Is the effort and resources needed to create the Hotspot accelerator worth the corresponding Application speedup?

What is the best-case Application speed-up (i.e., App speedup if Hotspot is sped up ∞)?

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)