COMP1001

Computer Systems

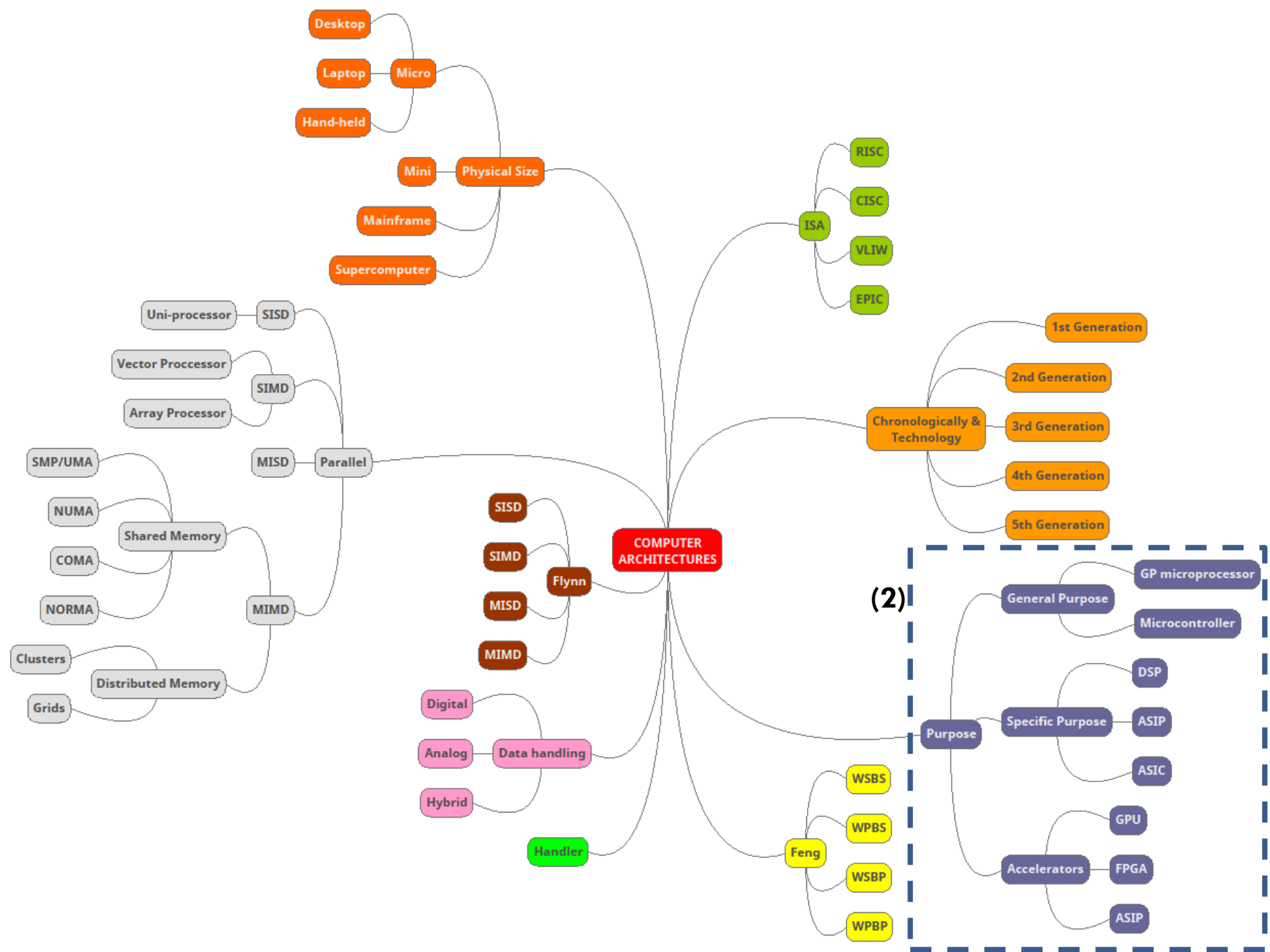
Dr. Vasilios Kelefouras

Email: v.kelefouras@plymouth.ac.uk
Website:
https://www.plymouth.ac.uk/staff/vasilios-kelefouras

Introduction

Outline

- Different computer architectures – classified regarding purpose
- General Purpose Processors
- Application Specific Processors
- Coprocessors / accelerators
- Multi-core processors
- Many-core processors
- Simultaneous Multithreading
- Single Instruction Multiple Data
- Heterogeneous Systems


COMPUTER ARCHITECTURES
Physical Size
Micro
Desktop
Laptop
Hand-held
Mini
Mainframe
Supercomputer
ISA
RISC
CISC
VLIW
EPIC
Chronologically & Technology
1st Generation
2nd Generation
3rd Generation
4th Generation
5th Generation
Parallel
SISD
Uni-processor
SIMD
Vector Proccessor
Array Processor
MISD
MIMD
Shared Memory
SMP/UMA
NUMA
COMA
NORMA
Distributed Memory
Clusters
Grids
Flynn
SISD
SIMD
MISD
MIMD
Data handling
Digital
Analog
Hybrid
Handler
Feng
WSBS
WPBS
WSBP
WPBP
(2)
Purpose
General Purpose
GP microprocessor
Microcontroller
Specific Purpose
DSP
ASIP
ASIC
Accelerators
GPU
FPGA
ASIP

Computer architectures – classified regarding purpose (1)

Fig.1. CPU market analysis

Computer architectures – classified regarding purpose (2)

1. General Purpose Processors
2. Specific Purpose Processors
3. Accelerators, also called co-processors

General Purpose Processors (GPP)

- They are classified into:
  1. **General purpose microprocessors** - general purpose computers, e.g., desktop PCs, laptops
     - Very powerful CPUs, e.g., Intel, AMD
     - Superscalar and Out of Order, big cache memories
  2. **Microcontrollers** - Embedded systems
     - Less powerful CPUs, e.g., ARM, Texas Instruments
     - They are usually designed for specific tasks in embedded systems
     - They usually have control oriented peripherals
     - They have on chip CPU, fixed amount of RAM, ROM, I/O ports
     - Lower cost, lower performance, lower power consumption, smaller than microprocessors
     - Appropriate for applications in which cost, power consumption and chip area are critical

GPP - General Purpose Microprocessor

- General Purpose Microprocessor – general purpose computers
  - They are designed for general purpose computers such as PCs, workstations, Laptops, notepads etc
  - Higher frequency than microcontrollers
  - Higher cost than microcontrollers
  - Higher performance than microcontrollers
  - Higher power consumption than microcontrollers
  - General purpose processors are designed to execute multiple applications and perform multiple tasks

GPP – Microcontrollers

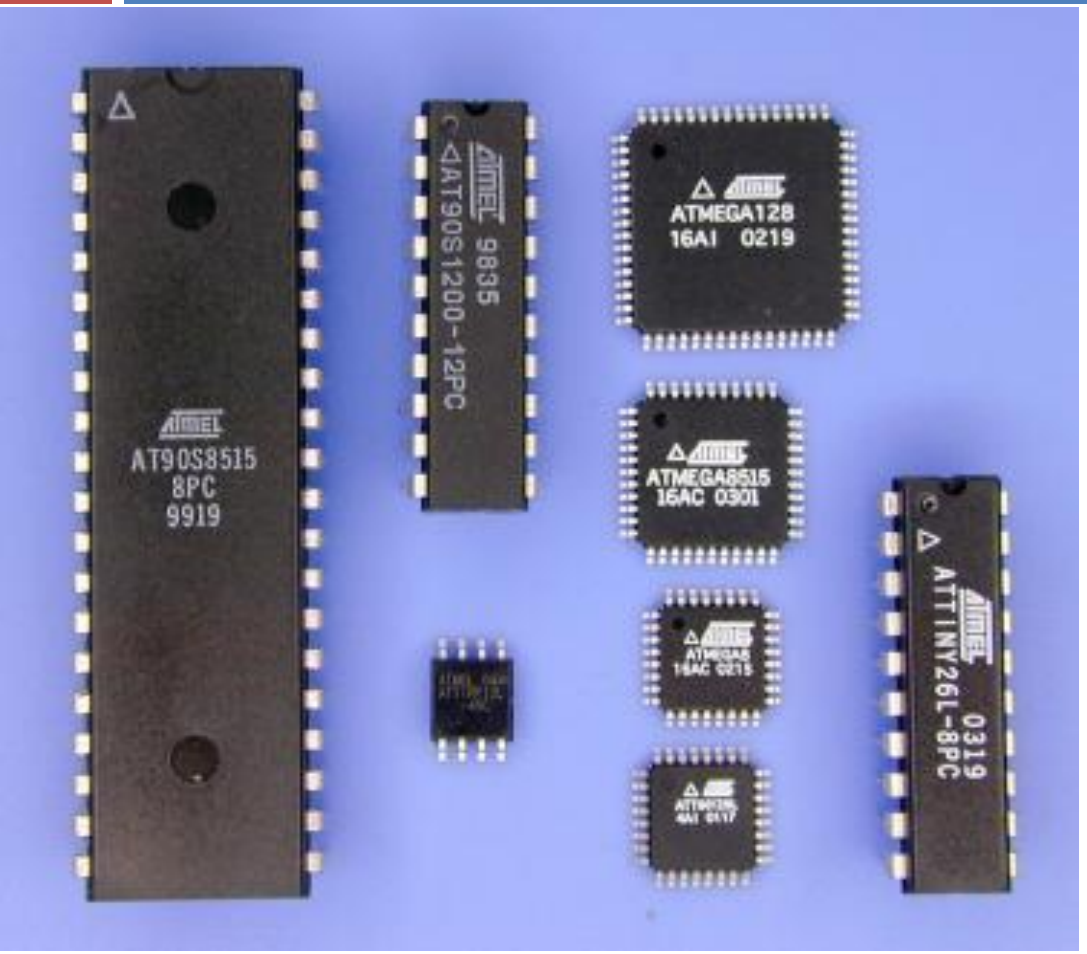


Fig.2. Microcontrollers

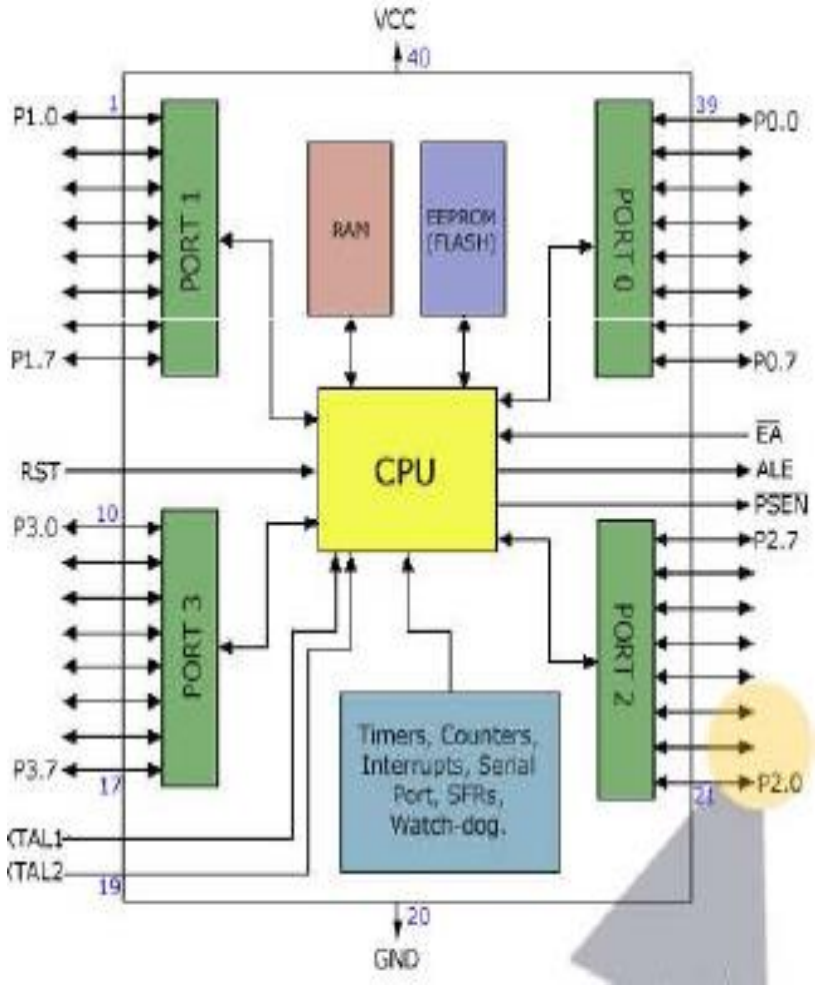
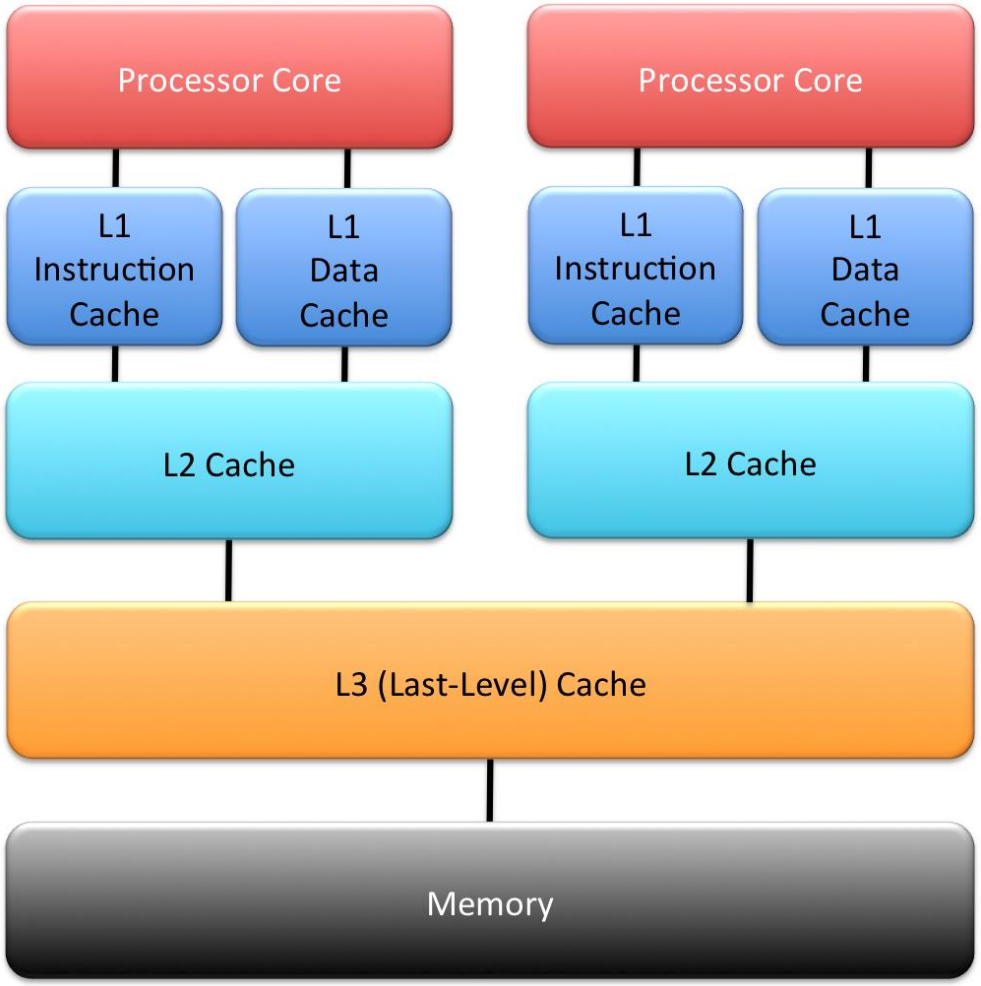


Fig.3. Components of the Microcontroller

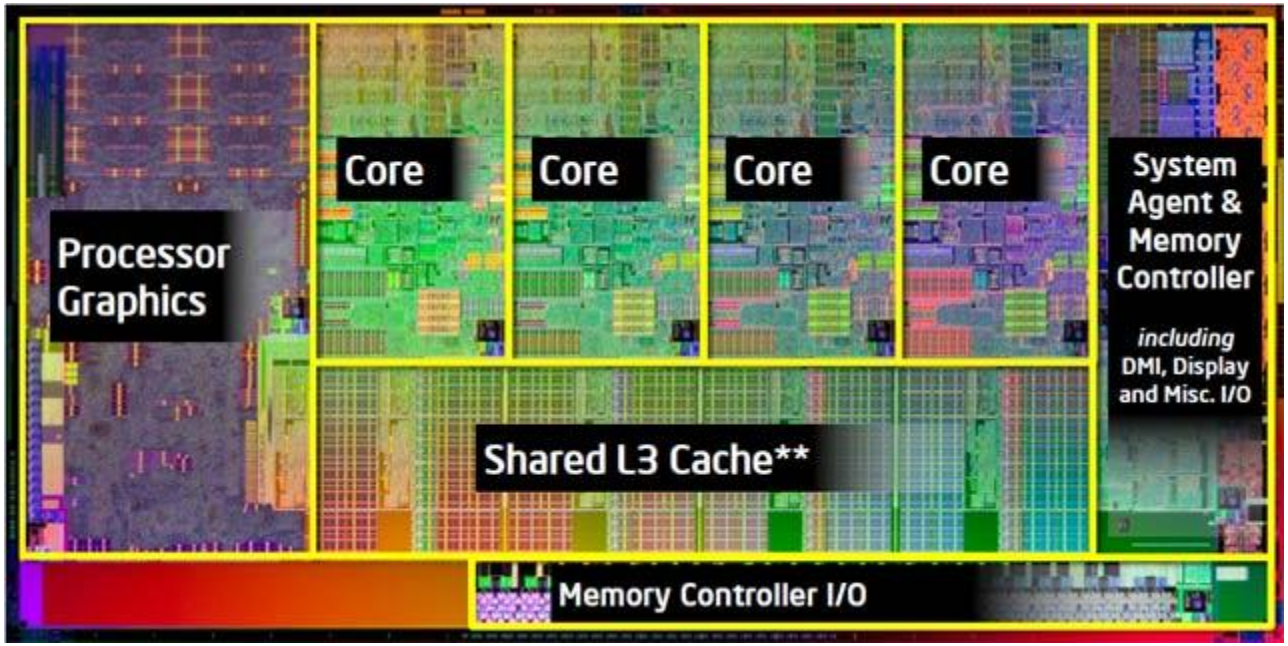
GPPs - Multi-core CPUs

- Multiple cores on the same chip using a shared cache
- Typically from 2-8 cores
- Both cores compete for the same hardware resources
- Both cores are identical
- Every core is a superscalar out of order CPU

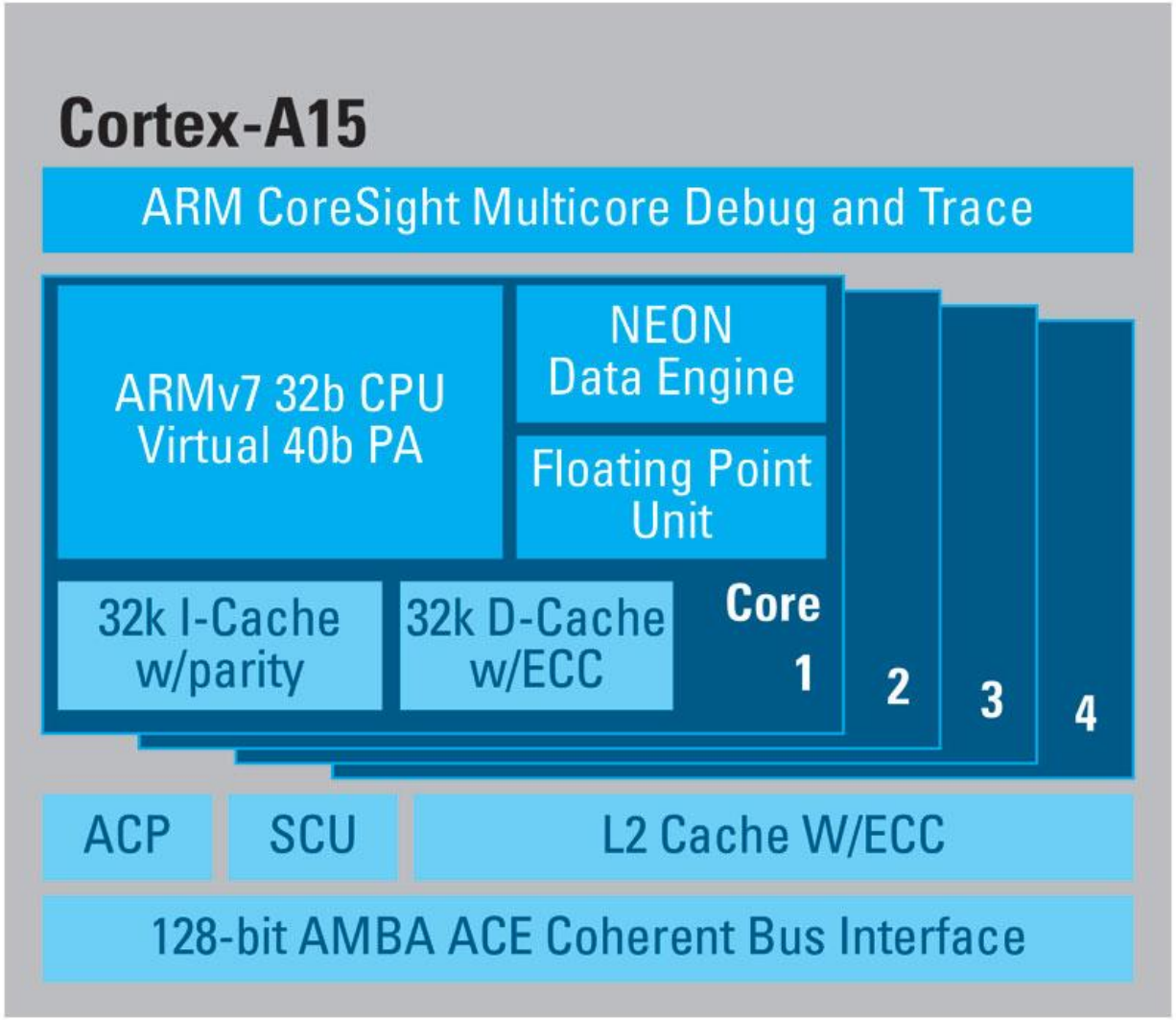


GPPs - Multi-core CPUs - Intel i7 architecture

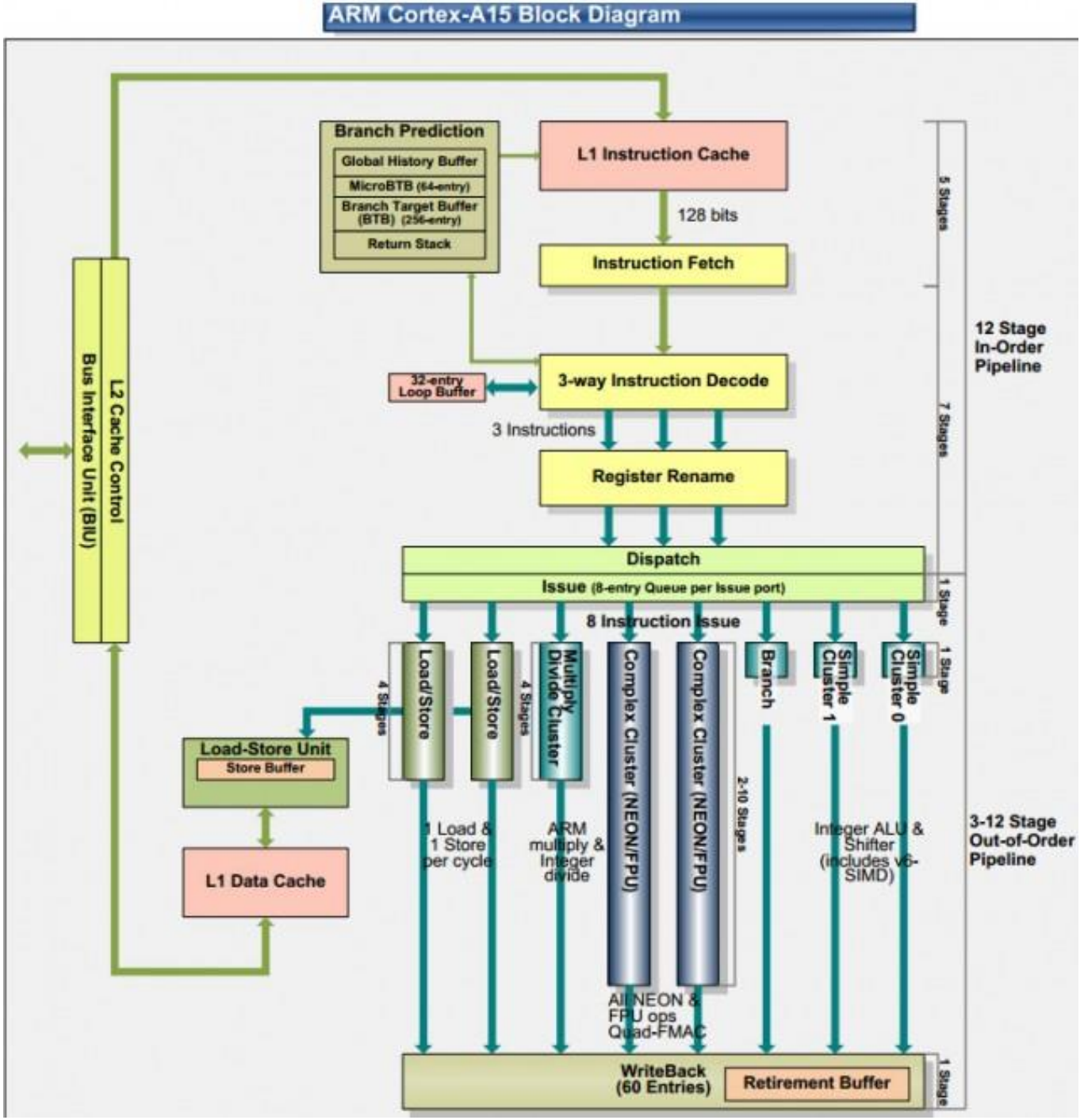
- In the figure below there is the Intel i7 CPU, where four CPU cores and the GPU reside in the same chip



GPPs - Multi-core CPUs – ARM Cortex-A15

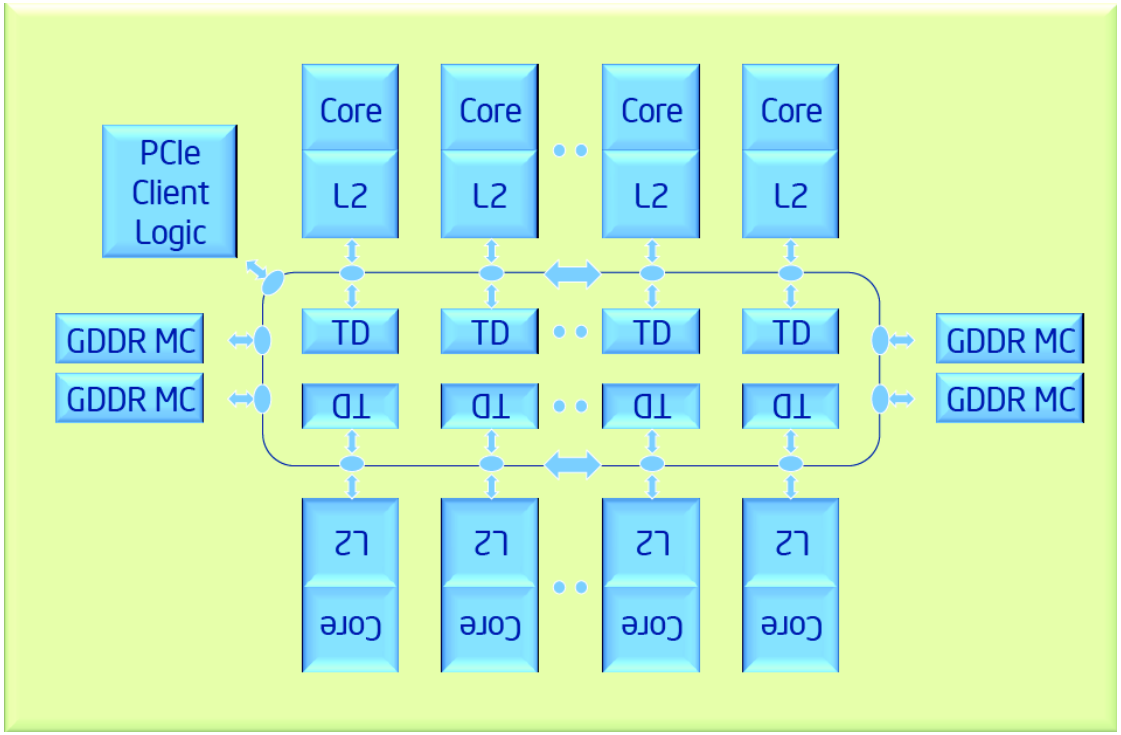


ARM Cortex-A15



Many core Processors – Intel Xeon Phi

- They are intended for use in supercomputers, servers, and high-end workstations
- 57-61 in-order simpler than i7 cores
- 1-1.7 Ghz
- 512bit vector instructions
- each core is connected to a ring interconnect via the Core Ring Interface



Application Specific Processors (1)

- General purpose processors offer good performance for all different applications but specific purpose processors offer better for a specific task
- **Application specific processors emerged as a solution for**
  - **higher performance**
  - **lower power consumption**
  - **Lower cost**
- Application specific processors have become a part of our life and can be found almost in every device we use on a daily basis
- Devices such as TVs, mobile phones and GPSs they all have application specific processors
- **They are classified into**
  1. **Digital Signal Processor (DSPs)**
  2. **Application Specific Instruction Set Processors (ASIPs)**
  3. **Application Specific Integrated Circuit (ASICs)**

Digital Signal Processors (DSPs)

1. **DSP**: Programmable microprocessor for extensive real-time mathematical computations
   - specialized microprocessor with its architecture optimized for the operational needs of digital signal processing
   - DSP processors are designed specifically to perform large numbers of complex arithmetic calculations and as quickly as possible
   - DSPs tend to have a different arithmetic Unit architecture;
     - specialized hardware units, such bit reversal, Multiply-accumulate units etc
     - Normally DSPs have a small instruction cache but no data cache memory

Application Specific Instruction set Processor (ASIP)

2. **ASIP:** Programmable microprocessor where hardware and instruction set are designed together for one special application
   - Instruction set, micro architecture and/or memory system are customised for an application or family of applications
   - Usually, they are divided into two parts: static logic which defines a minimum ISA and configurable logic which can be used to design new instructions
   - The configurable logic can be programmed and extend the instruction set similar to FPGAs
   - better performance, lower cost, and lower power consumption than GPP

Application Specific Integrated Circuit (ASIC)

3. **ASIC:** Algorithm completely implemented in hardware

- An Integrated Circuit (IC) designed for a specific line of a company – full custom
- It cannot modified – it is produced as a single, specific product for a particular application only
- Proprietary by nature and not available to the general public
- ASICs are full custom therefore they require very high development costs
- ASIC is just built for one and only one customer
- ASIC is used only in one product line
- Only volume production of ASICs for one product can make sense which means low unit cost for high volume products, otherwise the cost is not efficient
- **There is a lot of effort to implement an ASIC – there are specific languages such as VHDL and Verilog**

Building an application specific system on an embedded system (1)

Consider that we want to build and application specific system. We can choose:

1. GPP

- Functionality of the system is exclusively build on the software level
- it is not efficient in term of performance, power consumption, cost, chip area and heat dissipation

2. ASIC:

- No flexibility and extensibility

3. ASIP:

- a compromise between the two extremes
- used in embedded and system-on-chip solutions

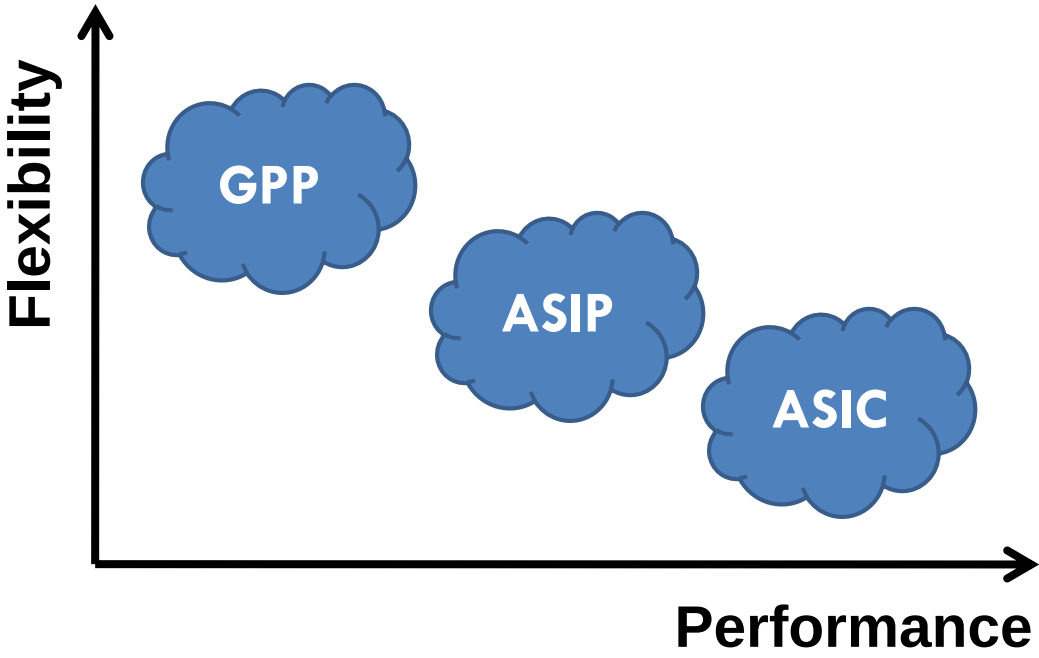


Fig.4. Comparison between Performance and flexibility

Building an application specific system on an embedded system (2)

Table 1. Comparison between different approaches for Building Embedded Systems [1]

	GPP	ASIP	ASIC
Performance	Low	High	Very High
Flexibility	Excellent	Good	Poor
HW design	None	Large	Very large
SW design	Small	Large	None
Power	Large	Medium	Small
reuse	Excellent	Good	Pure
market	Very large	Relatively large	Small
Cost	High	Medium	Volume sensitive

Accelerators - coprocessors

- Accelerators / co-processors are used to perform some functions more efficiently than the CPU

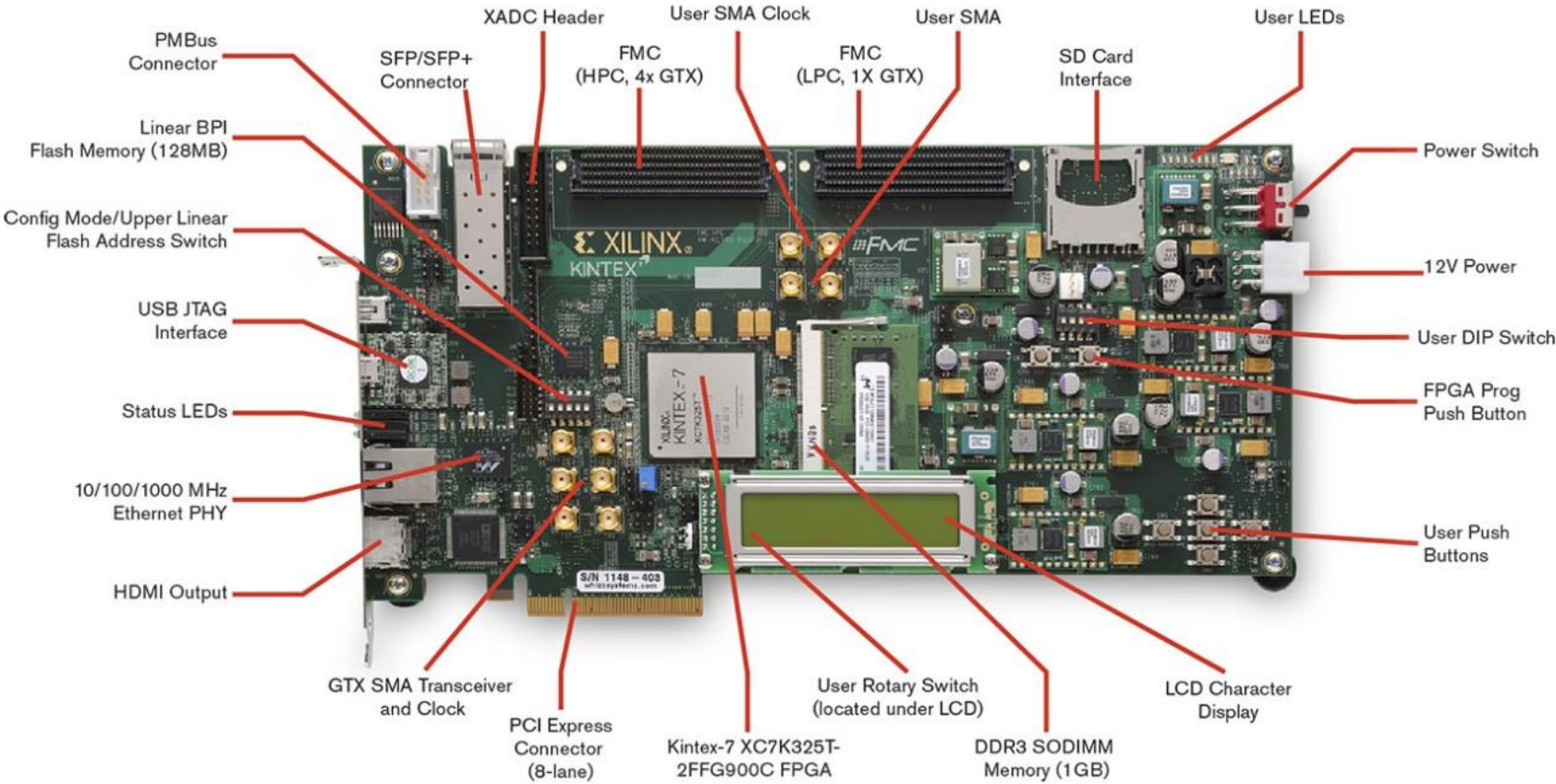
- They offer
  - Higher performance
  - Lower power consumption
  - But they are harder to program

Field Programmable Gate Arrays (FPGAs)

- FPGAs are reprogrammable devices that allow us to create our own digital circuits
- An FPGA (Field Programmable Gate Array) is an array of logic gates that can be hardware-programmed to fulfill user-specified tasks
  - **FPGAs contain programmable logic components called "logic blocks", and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together"**
  - An application can be implemented entirely in HW
  - The FPGA configuration is generally specified using a hardware description language (HDL) like VHDL and Verilog **– hard to program**
  - High Level Synthesis (HLS) provides a solution to this problem. Engineers write C/C++ code instead, but it is not that efficient yet

FPGAs (2)

- FPGAs come on a board. This board is connected to a PC and programmed. Then, it can work as a standalone component

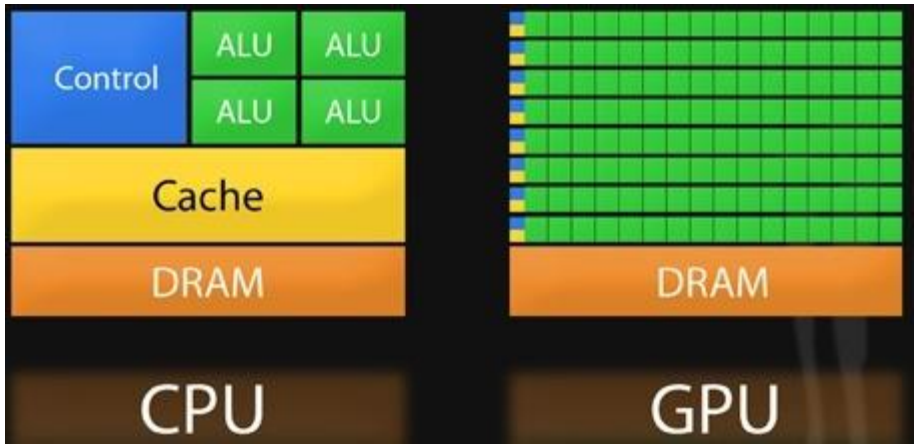


FPGAs (3)

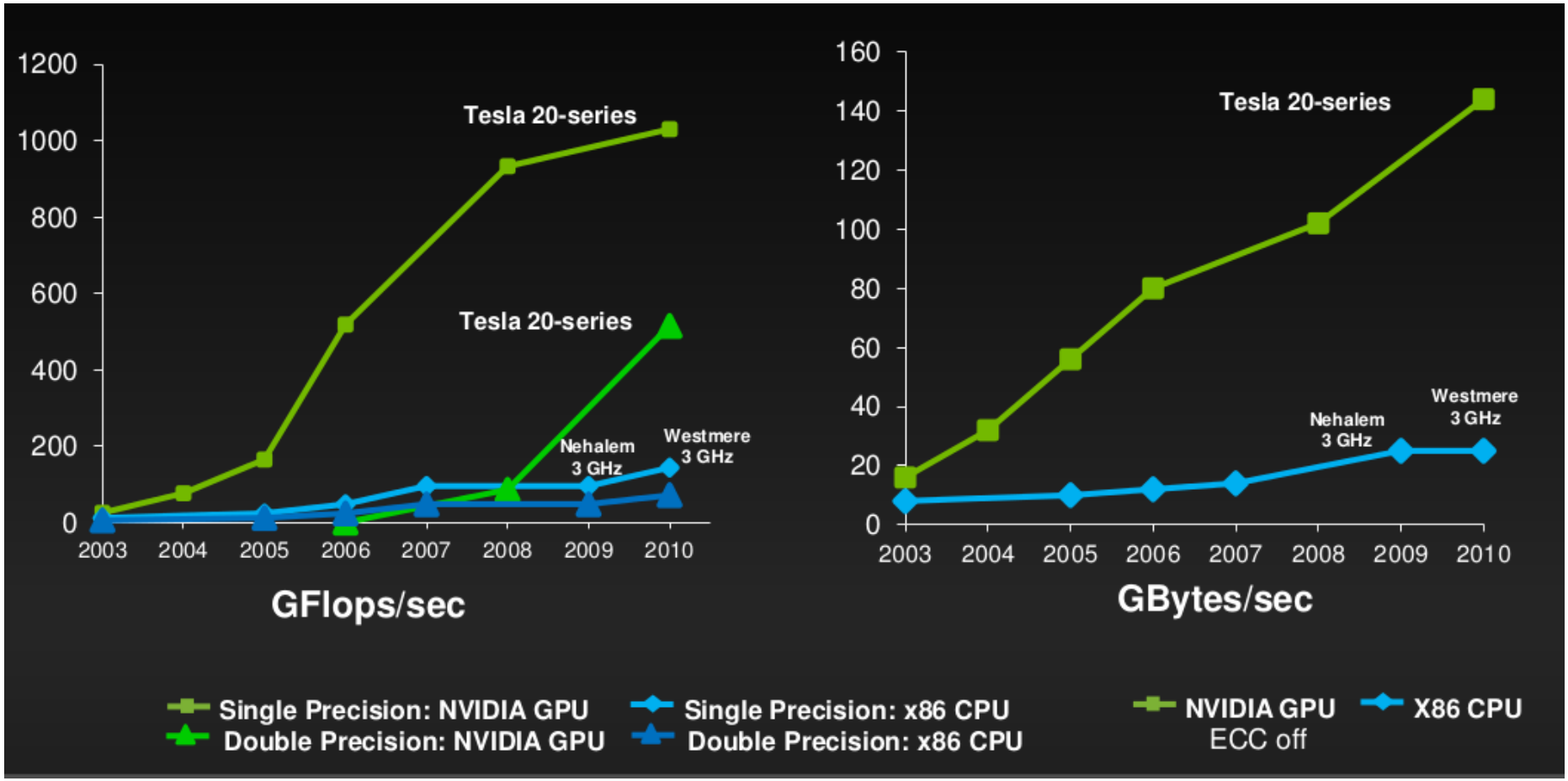
- **Unlike an ASIC the circuit design is not set and you can reconfigure an FPGA as many times as you like!**
  - Creating an ASIC also costs potentially millions of dollars and takes weeks or months to create.
  - However, the recurring cost is lower than the cost of the FPGA (no silicon area is wasted in ASICs).
  - ASICs are cheaper only when the production number is very high
- Intel supports hybrid CPU-FPGA chips

GPUs (1)

- Graphics Processing Unit (GPU)
  - The GPU's advanced capabilities were originally used primarily for 3D game graphics. But now those capabilities are being harnessed more broadly to accelerate computational workloads in other areas too
  - GPUs are very efficient for
    - Data parallel applications
    - Throughput intensive applications - the algorithm is going to process lots of data elements



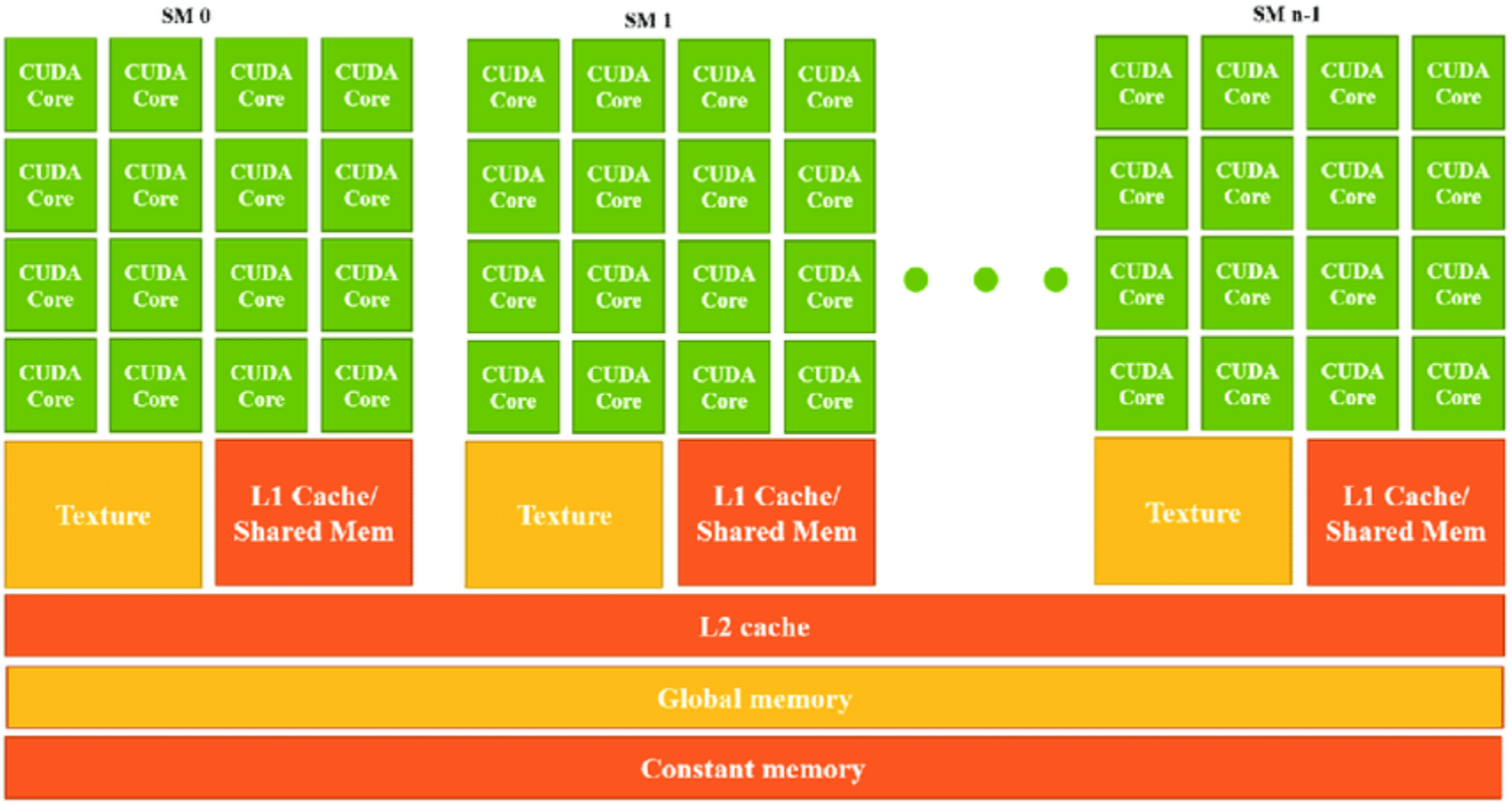
GPUs (2) – why do we need GPUs?



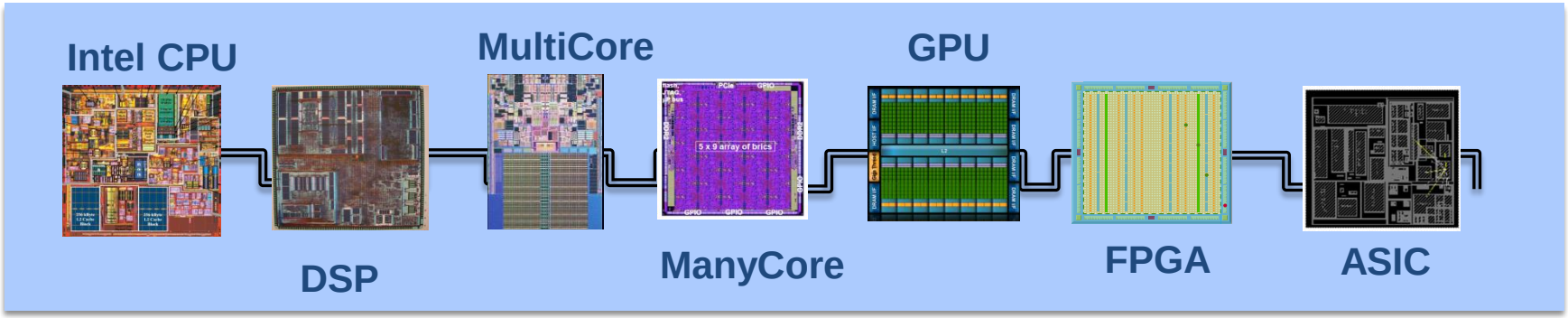
GPUs (3)

- A GPU is always connected to a CPU – GPUs are coprocessors
- GPUs work in lower frequencies than CPUs
- GPUs have many processing elements (thousands of cores)
- GPUs have smaller and faster cache memories
- OpenCL is the dominant open general-purpose GPU computing language, and is an open standard
- The dominant proprietary framework is Nvidia CUDA
- **GPU coding is based on vectorization**

Schematic of Nvidia GPU architecture



Comparison



Flexibility, Programming Abstraction

Performance, Area and Power Efficiency

CPU:
- Market-agnostic
- Accessible to many programmers (Python, C++)
- Flexible, portable

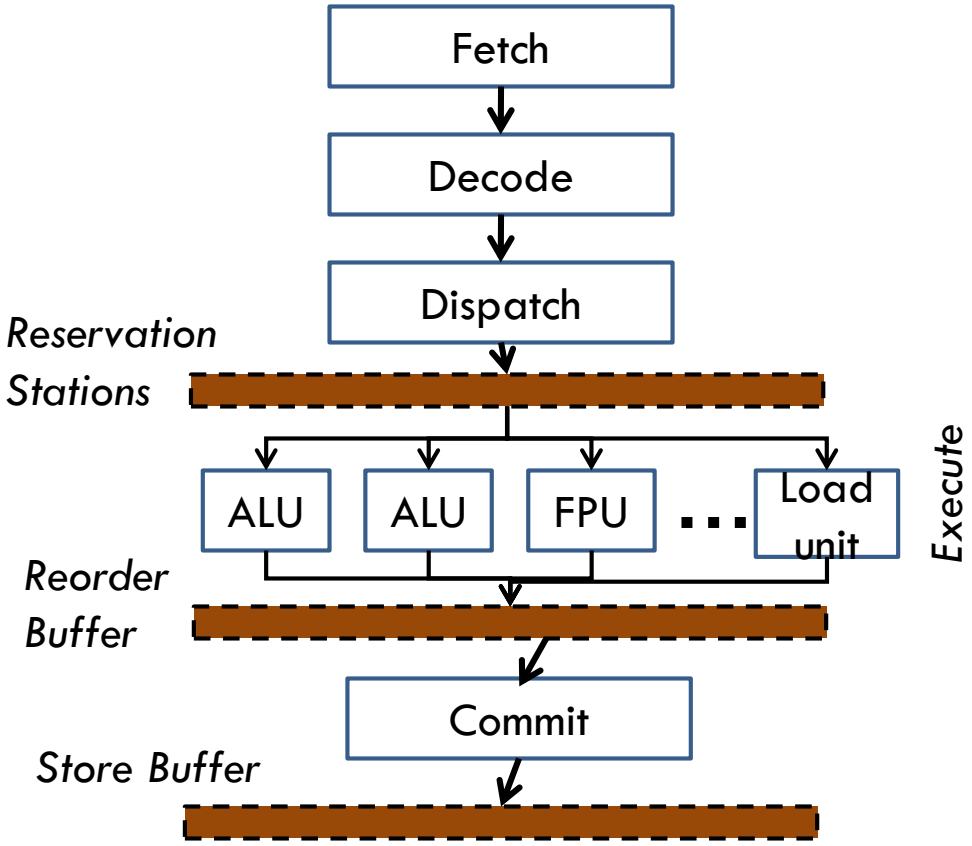
FPGA:
- Somewhat Restricted Market
- Harder to Program (VHDL, Verilog)
- More efficient than SW
- More expensive than ASIC

ASIC
- Market-specific
- Fewer programmers
- Rigid, less programmable
- Hard to build (physical)

Superscalar and Out of Order is not enough (1)

- The approach of exploiting ILP through superscalar execution is seriously weakened by the fact that **normally programs don't have a lot of fine-grained parallelism in them**

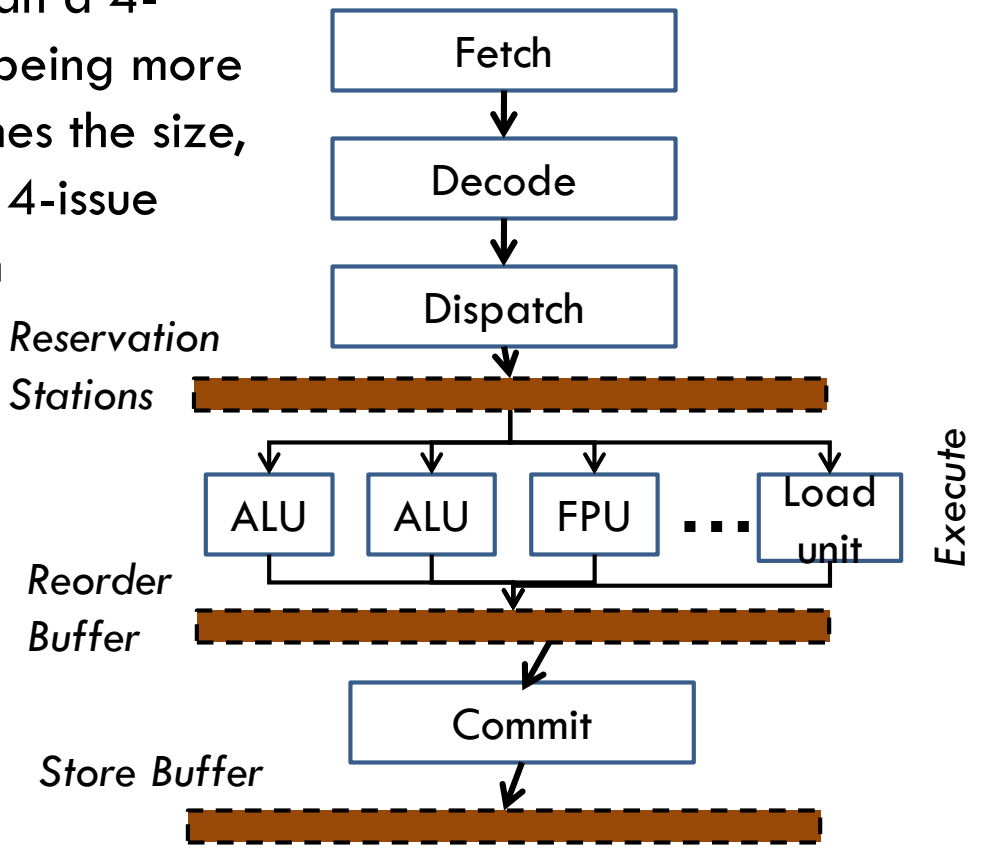
- Because of this, **CPUs normally don't exceed more than 3 instructions per cycle** when running most mainstream, real-world software, due to a combination of load latencies, cache misses, branching and dependencies between instructions



Superscalar and Out of Order is not enough (2)

- Issuing many instructions in the same cycle only ever happens for short bursts
- Moreover, the dispatch logic of a 5-issue processor is more than 50% larger than a 4-issue design (chip area), with 6-issue being more than twice as large, 7-issue over 3 times the size, 8-issue more than 4 times larger than 4-issue (for only 2 times the width), and so on

- **Exploiting instruction level parallelism is expensive**



Superscalar and Out of Order is not enough (3)

Very important features that further improve the performance of CPUs are:

- **Simultaneous multi threading (SMT) or Hyper Threading in Intel processors**
- **Single Instruction Multiple Data (SIMD) - vectorization**

Simultaneous multi-threading (SMT) as a solution to improve CPU's performance (1)

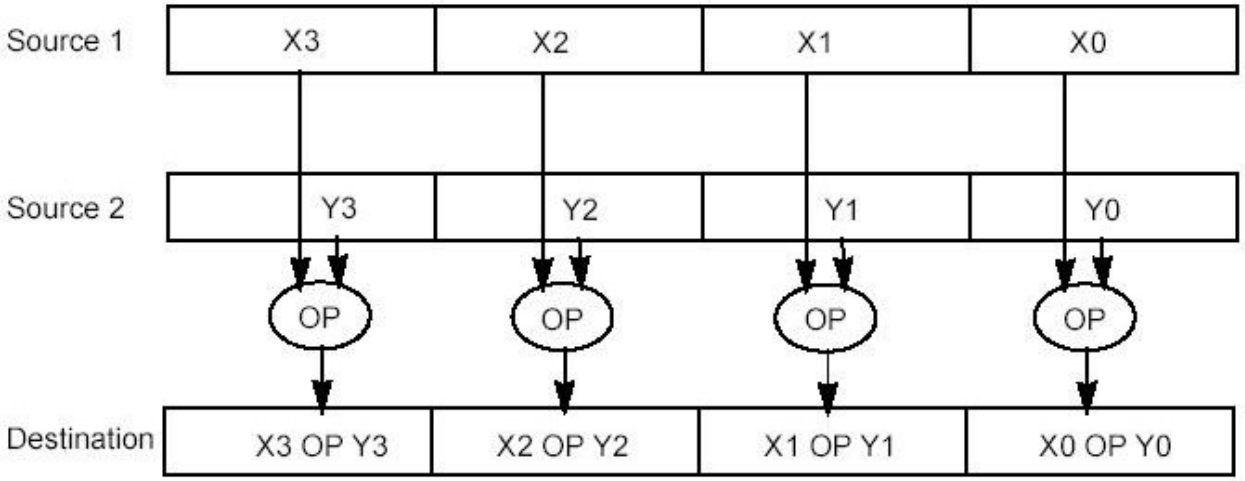
- **SMT is the process of a CPU splitting each of its physical cores into virtual cores**
- Normally 2 threads are executed in one physical CPU core
- If additional independent instructions aren't available within the program being executed, there is another potential source of independent instructions – other running programs, or other threads within the same program
- **Simultaneous multi-threading (SMT) is a processor design technique which exploits exactly this type of thread-level parallelism**
- Fill the empty bubbles in the pipelines with useful instructions, but this time rather than using instructions from further down in the same code, **the instructions come from multiple threads running at the same time, all on the one processor core**
- So, an SMT processor appears to the rest of the system as if it were multiple independent processors, just like a true multi-processor system

Simultaneous multi-threading (SMT) as a solution to improve CPU's performance (2)

- **From a hardware point of view, implementing SMT requires duplicating all of the parts of the processor** which store the "execution state" of each thread
  - These parts only constitute a tiny fraction of the overall processor's hardware
  - The really large and complex parts, such as the decoders and dispatch logic, the functional units, and the caches, are all shared between the threads
- On top of this, the fact that the threads in an SMT design are all sharing just one processor core and just one set of caches, has major performance downsides compared to a true multi-processor (or multi-core)
- **SMT performance can actually be worse than single-thread performance**
- Speedups from SMT on the Pentium 4 ranged from around -10% to +30% depending on the application(s)

Single Instruction Multiple Data (SIMD) – Vectorization (1)

- **In addition to instruction-level parallelism, there is yet another source of parallelism – data parallelism**
- Rather than looking for ways to execute groups of instructions in parallel, the idea is to look for ways to make one instruction apply to a group of data values in parallel
- **This is sometimes called SIMD parallelism (single instruction multiple data). More often, it's called vector processing**

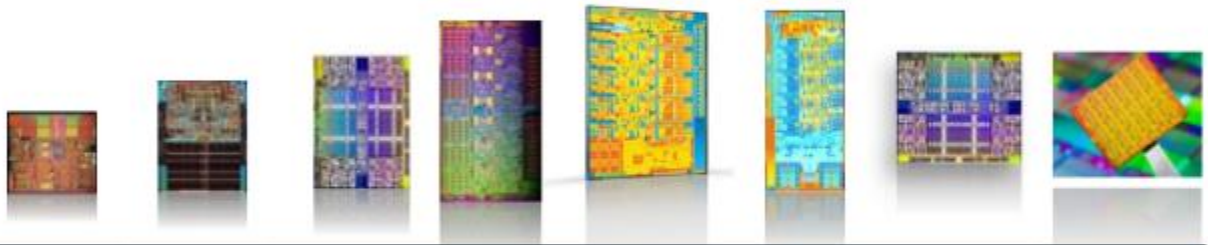


Single Instruction Multiple Data (SIMD) – Vectorization (2)


```
for(i = 0; i <= MAX;i++)
    c[i] = a[i] + b[i];
```

Current trend

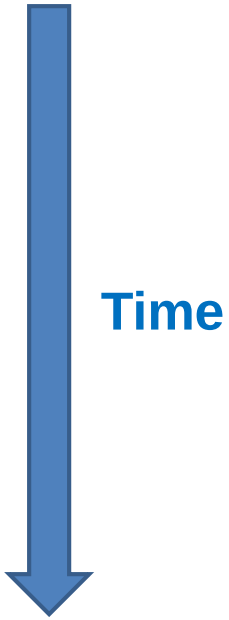
More cores . More Threads . Wider vectors



	Intel® Xeon® processor 64-bit	Intel® Xeon® processor 5100 series	Intel® Xeon® processor 5500 series	Intel® Xeon® processor 5600 series	Intel® Xeon® processor code-named Sandy Bridge EP	Intel® Xeon® processor code-named Ivy Bridge EP	Intel® Xeon® processor code-named Haswell EP	Future Xeon	Intel® Xeon Phi™ coprocessor Knights Corner	Intel® Xeon Phi™ processor & coprocessor Knights Landing[1]
Core(s)	1	2	4	6	8	12	18	>18	61	70+
Threads	2	2	8	12	16	24	36	>36	244	280+
SIMD Width	128	128	128	128	256	256	256	512	512	512

Hardware Evolution

- Scalar Processors
- Pipelined Processors
- Superscalar and VLIW Processors
- Out of order Processors
- Vectorization
- Hyper-Threading
- Multicore Processors
- Manycore Processors
- Heterogeneous systems



Heterogeneous computing (1)

Single core Era -> Multi-core Era -> Heterogeneous Systems Era

- **Heterogeneous computing refers to systems that use more than one kind of processors or cores**
  - These systems gain performance or energy efficiency not just by adding the same type of processors, but by adding dissimilar (co)-processors, usually incorporating specialized processing capabilities to handle particular tasks
  - Systems with General Purpose Processors (GPPs), GPUs, DSPs, ASIPs etc.

- **Heterogeneous systems offer the opportunity to significantly increase system performance and reduce system power consumption**

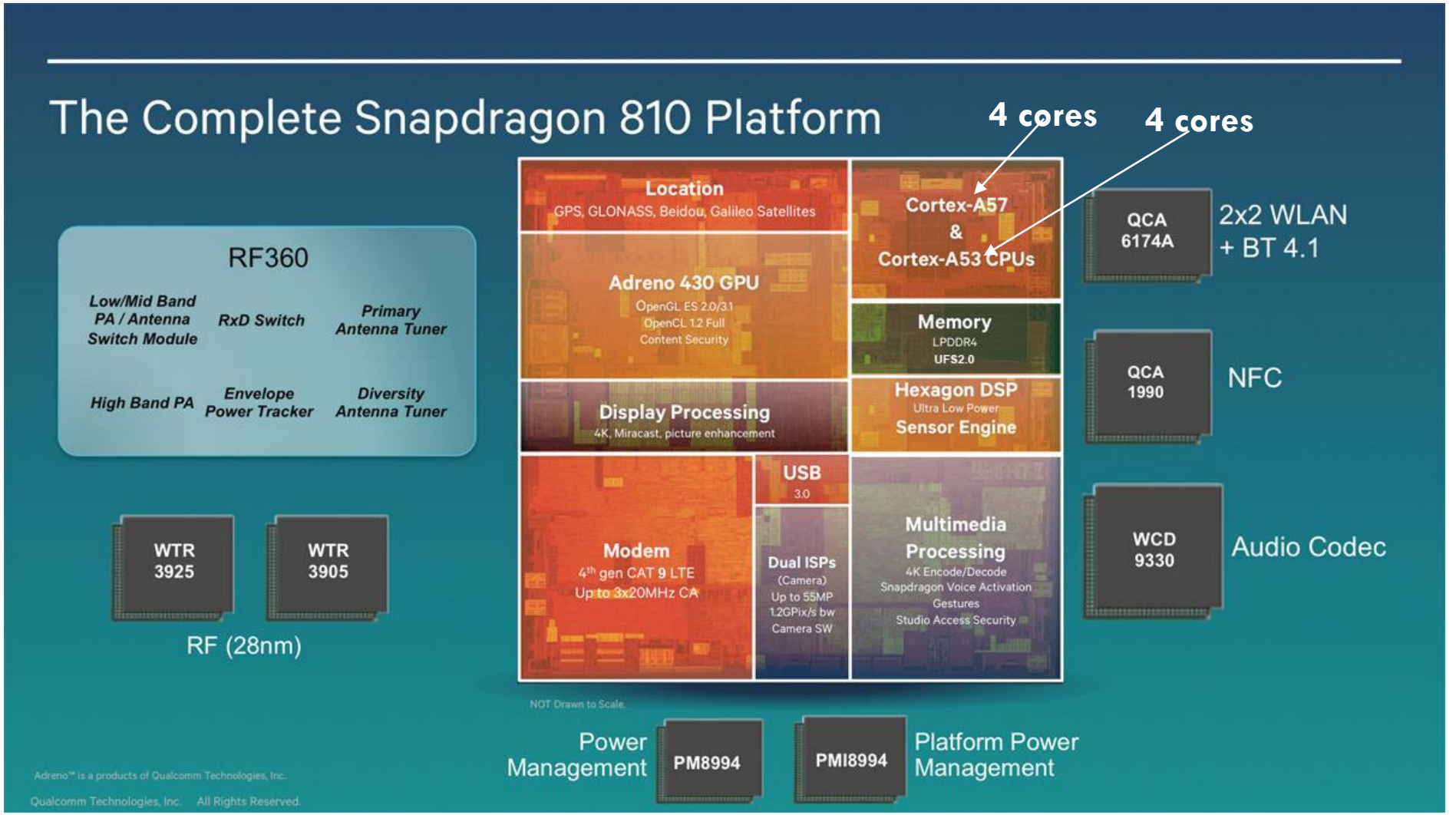
Heterogeneous computing (2)

- Software issues:
  - Offloading
  - Programmability – think about CPU code (C code), GPU code (CUDA), FPGA code (VHDL)
  - Portability - What happens if your code runs on a machine with an FPGA instead of a GPU

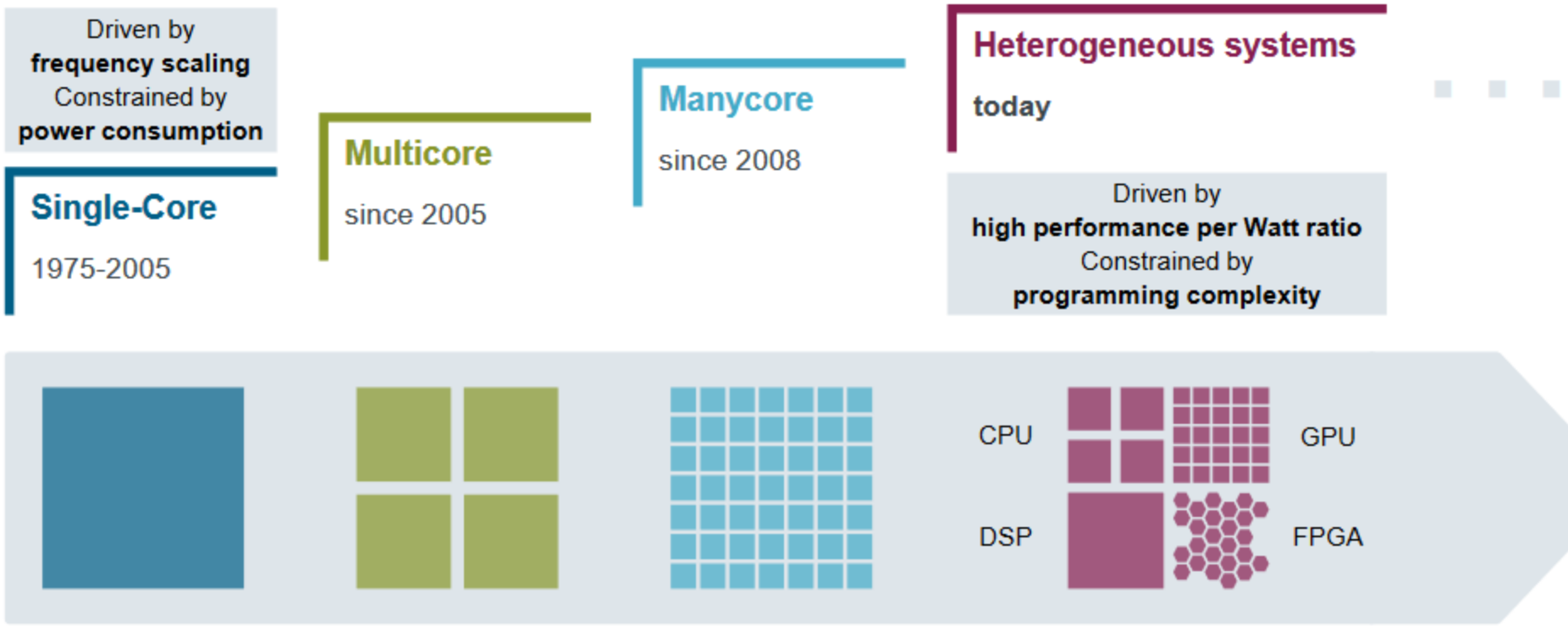
Comparisons between Homogeneous and Heterogeneous Computing

Symmetric, Same cores (Usually CPUs)	Assymmetric, Different cores (CPUs, GPUs, DSPs and accelerators)
operation is guaranteed to be same at each core	operation cannot be supposed to be same at each core
easy to off load tasks	more complicated to off load tasks
good compatibility	less compatibility **specialized for specific tasks**

Heterogeneous computing (3) – A mobile phone system



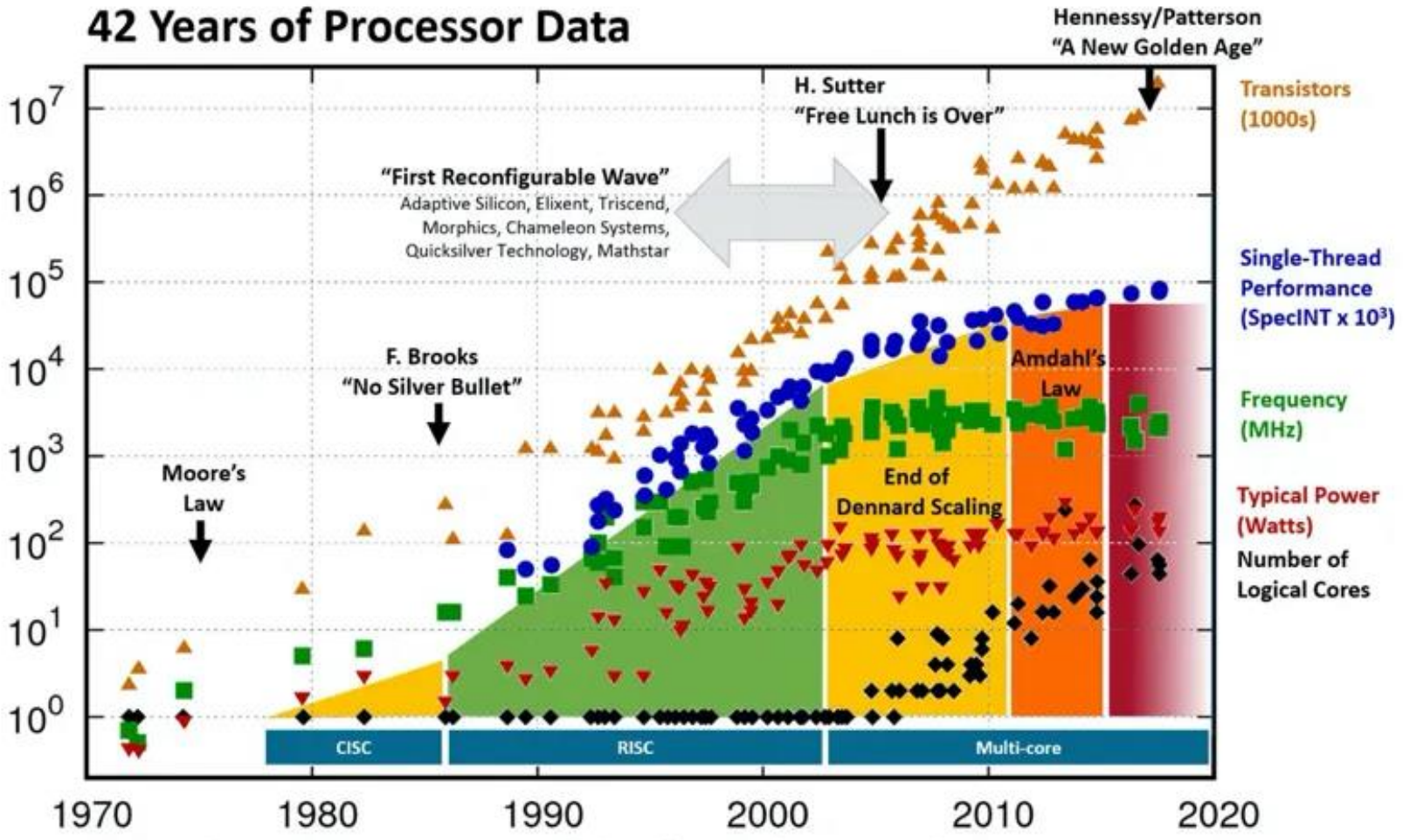
Hardware Trends
From single core processors to heterogeneous systems on a chip



H. Esmaeilzadeh et al., "Dark silicon and the end of multicore scaling", International Symposium on Computer Architecture (ISCA). ACM, 2011.
M. Zahran, "Heterogeneous Computing Here to Stay". ACM Queue, Nov/Dev 2016.

Taken from https://embb.io/downloads/MTAPI_EMBB.pdf

Hardware Architecture Trends



Hennessy and Patterson, Turing Lecture 2018, overlaid over "42 Years of Processors Data"
https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/;"First Wave" added by Les Wilson, Frank Schirrmeister
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 2010-2017 by K. Rupp

Think-Pair-Share Exercise

- What is in your opinion the most appropriate computer architecture for a smart phone and why?
  a. 1 microcontroller
  b. 1 normal speed GPP, e.g., Pentium II
  c. 1 quad-core Intel i7
  d. A heterogeneous computer architecture with 1 normal speed GPP, 1 DSP, 1 GPU and a few Microcontrollers

Conclusions

- Modern Computer Systems include Parallel Heterogeneous Computer Architectures
- Heterogeneous systems offer the opportunity to significantly
  - increase performance
  - reduce power consumption
  - reduce cost
- Issues:
  - Programmability
  - Portability
  - Design good Compilers – optimize the code

References and Further Reading

[1] Nohl, A, Schirrmeister, F & Taussig, D. "Application specific processor design: Architectures, design methods and tools" Computer-Aided Design (ICCAD), 2010 IEEE/ACM International Conference on Nov. 2010.

[2] Tom Spyrou Challenges in the Static Timing Analysis of FPGA's, ALTERA, TAU 2015 3/2015

[3] Modern Microprocessors A 90-Minute Guide!. Lighterra, http://www.lighterra.com/papers/modernmicroprocessors/

[4] Introduction to GPU computing, available at http://www.int.washington.edu/PROGRAMS/12-2c/week3/clark_01.pdf

[5] Yousef Qasim, P Radyumna Janga, Sharath Kumar, Hani Alesaimi, APPLICATION SPECIFIC PROCESSORS, ECE/CS 570 PROJECT FINAL REPORT, available at http://web.engr.oregonstate.edu/~qassimy/index_files/Final_ECE570_ASP_2012_Project_Report.pdf

[6] William Stallings, Computer Organization & Architecture. Designing for Performance, Seventh Edition

[7] Andrew S. Tanenbaum, Todd Austin, Structured Computer Organization. Sixth Edition, PEARSON

Thank you