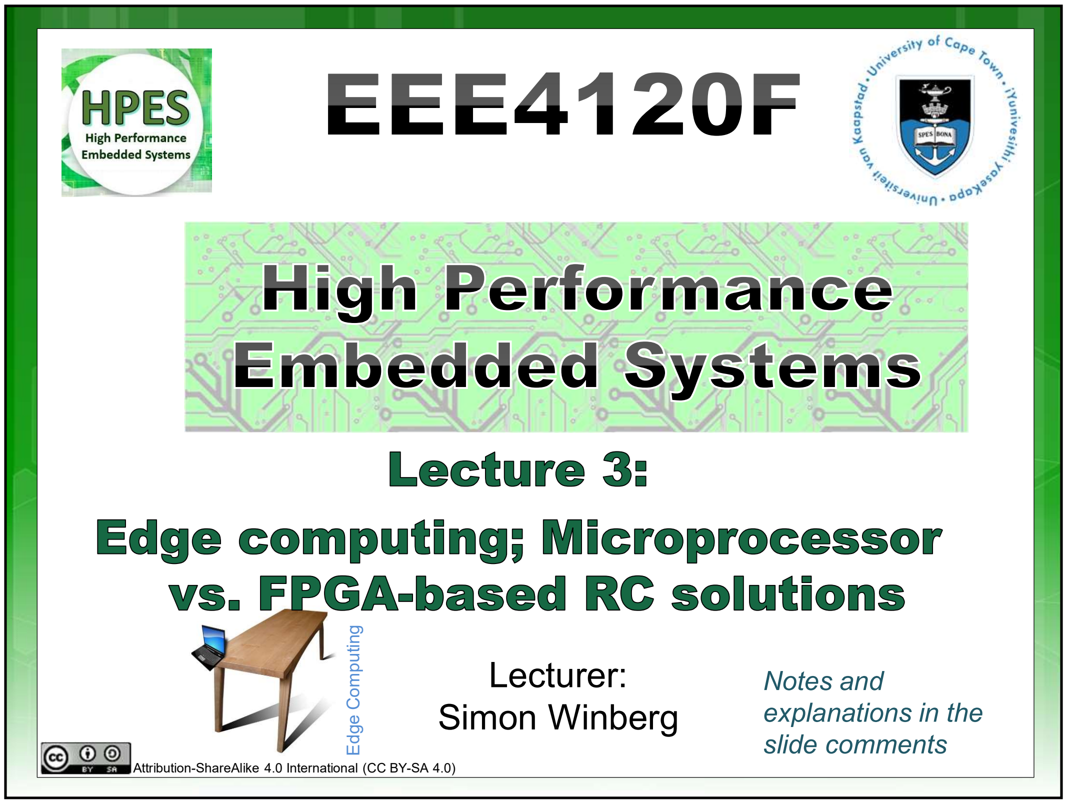


EEE4120F

High Performance Embedded Systems

Lecture 3: Edge computing; Microprocessor vs. FPGA-based RC solutions

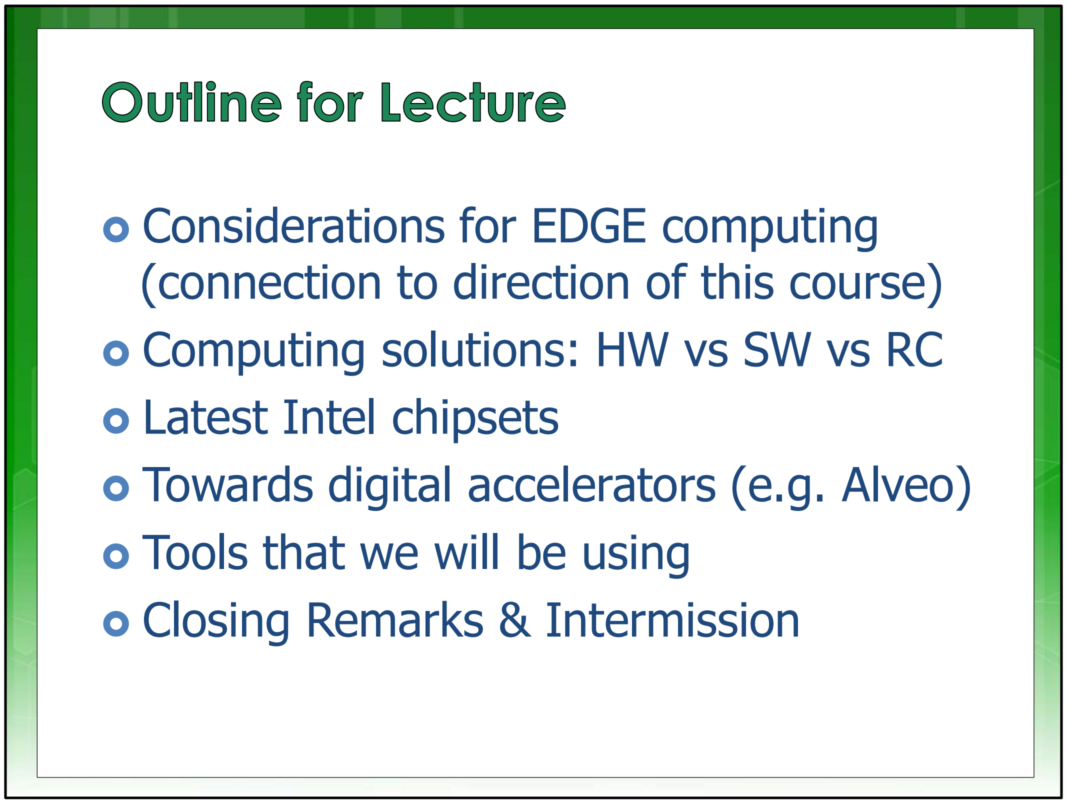
Edge Computing

Lecturer:
Simon Winberg

Notes and explanations in the slide comments

Elaborative text has been added to these slides, these are provided as a means to just supplement some of the information. The lecture sessions will cover the essentials of this slideshow, and provide an opportunity for questions and discussion of the topics.

In this lecture, we're going to dip into the concept of edge computing. This is highly relevant to HPES design, as nowadays there is increasingly more demand for a significant amount of computing to be done 'at the edge' close to the environment that the sensors and/or actuators are in, without having to send the data to some remote processing centre.



Outline for Lecture

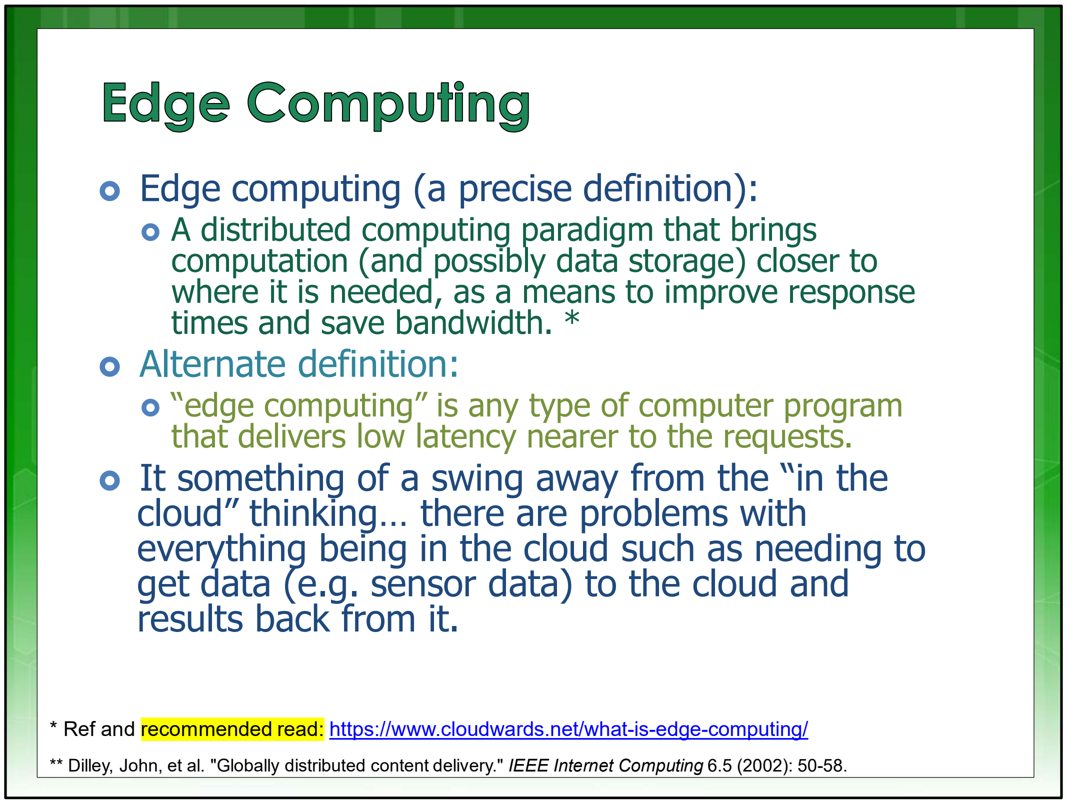
- Considerations for EDGE computing (connection to direction of this course)
- Computing solutions: HW vs SW vs RC
- Latest Intel chipsets
- Towards digital accelerators (e.g. Alveo)
- Tools that we will be using
- Closing Remarks & Intermission

Here’s the overview of this topic.

Edge Computing

EEE4120F HPES

Edge Computing. You have probably heard the term by now. This course isn’t just on the edge. It connects with edge computing.



Edge Computing

- Edge computing (a precise definition):
  - A distributed computing paradigm that brings computation (and possibly data storage) closer to where it is needed, as a means to improve response times and save bandwidth. *
- Alternate definition:
  - "edge computing" is any type of computer program that delivers low latency nearer to the requests.
- It something of a swing away from the "in the cloud" thinking… there are problems with everything being in the cloud such as needing to get data (e.g. sensor data) to the cloud and results back from it.

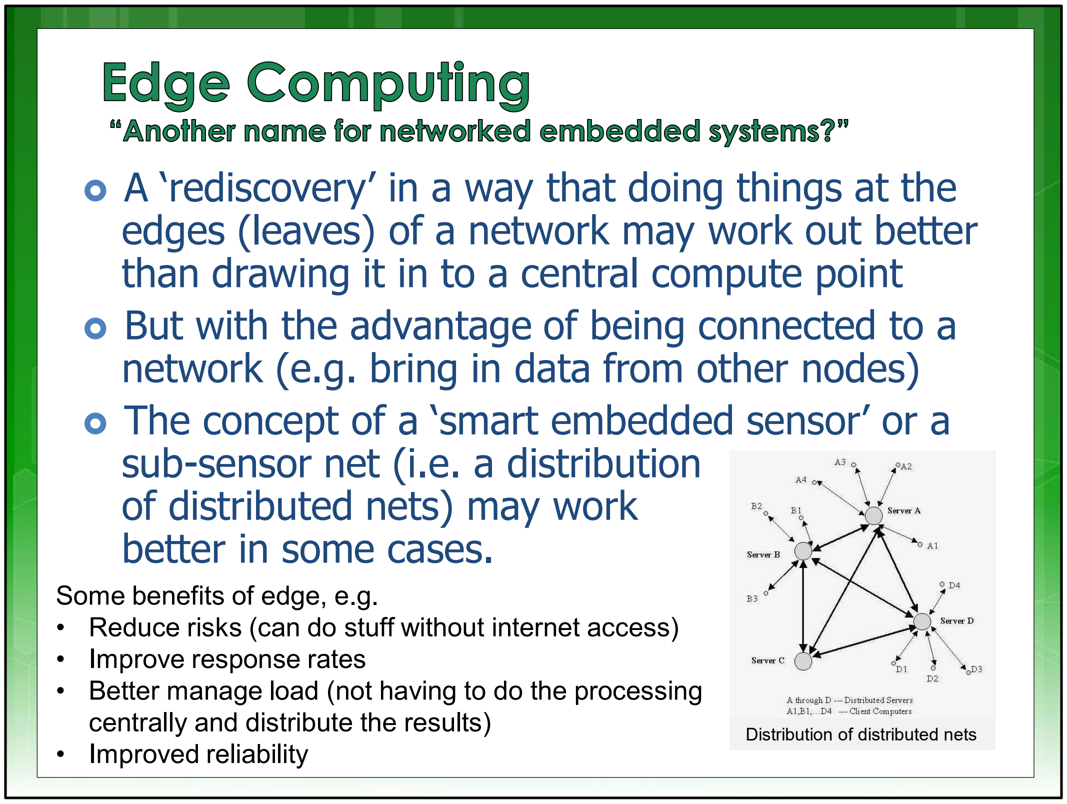
* Ref and recommended read: https://www.cloudwards.net/what-is-edge-computing/

** Dilley, John, et al. "Globally distributed content delivery." *IEEE Internet Computing* 6.5 (2002): 50-58.

This is Edge Computing more precisely defined…

Edge computing is a distributed computing paradigm that brings computation closer to where it is needed. It is an approach to improve response times and to save bandwidth.

If you read the Landscape of Parallel computing, it one of the many pendulum swing in computing. The old school computing (old-CW) was where the user is at the computer where stuff is processed. Then it became the user sitting at a dump terminal away from the mainframe. Then it swung to desktops … kind of a swing back to sitting next to where the processing is happening. Then along come the explosion of CPS and IoT and centralizing data. Then. Or now. Came along computing At The Edge. It was a method that stopped sending all the data for central processing, and to get it done at the edges instead. Obviously this could only be done effectively once the things at the edges had sufficiently beefy computing resources to do useful processing. Like me, you might have further ramblings and daydreams on the subject… such as when quantum computing fully comes on the scene we will we be able to just predict what the edges will tell us… and not actually need anything at the edge … like Schrödinger's cat we could consider the result is bother there and not there until we actually take a look (that might have just been a joke to get out implementing something ;-)).

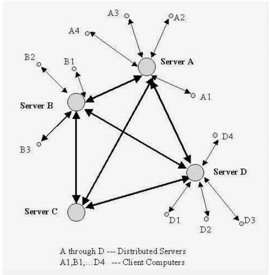


Edge Computing

"Another name for networked embedded systems?"

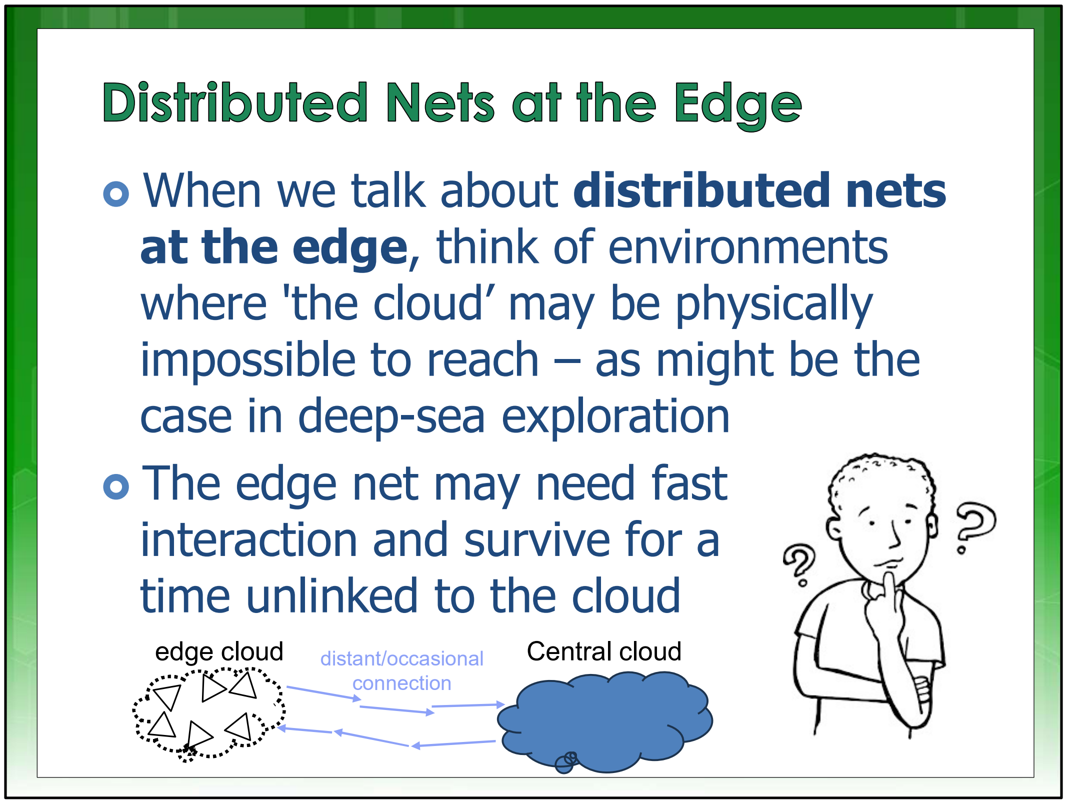
- A 'rediscovery' in a way that doing things at the edges (leaves) of a network may work out better than drawing it in to a central compute point
- But with the advantage of being connected to a network (e.g. bring in data from other nodes)
- The concept of a 'smart embedded sensor' or a sub-sensor net (i.e. a distribution of distributed nets) may work better in some cases.

Some benefits of edge, e.g.
- Reduce risks (can do stuff without internet access)
- Improve response rates
- Better manage load (not having to do the processing centrally and distribute the results)
- Improved reliability


A3
A2
A4
B2
B1
Server A
A1
Server B
D4
B3
Server D
Server C
D1
D2
D3
A through D --- Distributed Servers
A1,B1,...D4 --- Client Computers

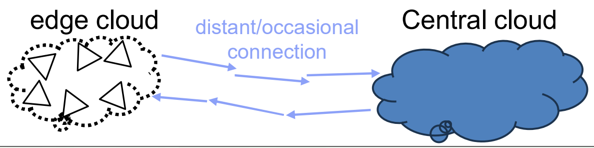
Distribution of distributed nets

Edge computing can also be referred to as networked embedded systems. Where computing takes place on the edges or the leaves of a network. For some cases this could be a more beneficial setup in comparison to having a central server. Furthermore, it is sometimes useful to have a distribution of distributed networks where you have multiple servers closer to their edge, which can be referred to as sub-sensor networks. This provides numerous benefits such as improved response time, better load management and improved reliability as you are not dependent on one central server.

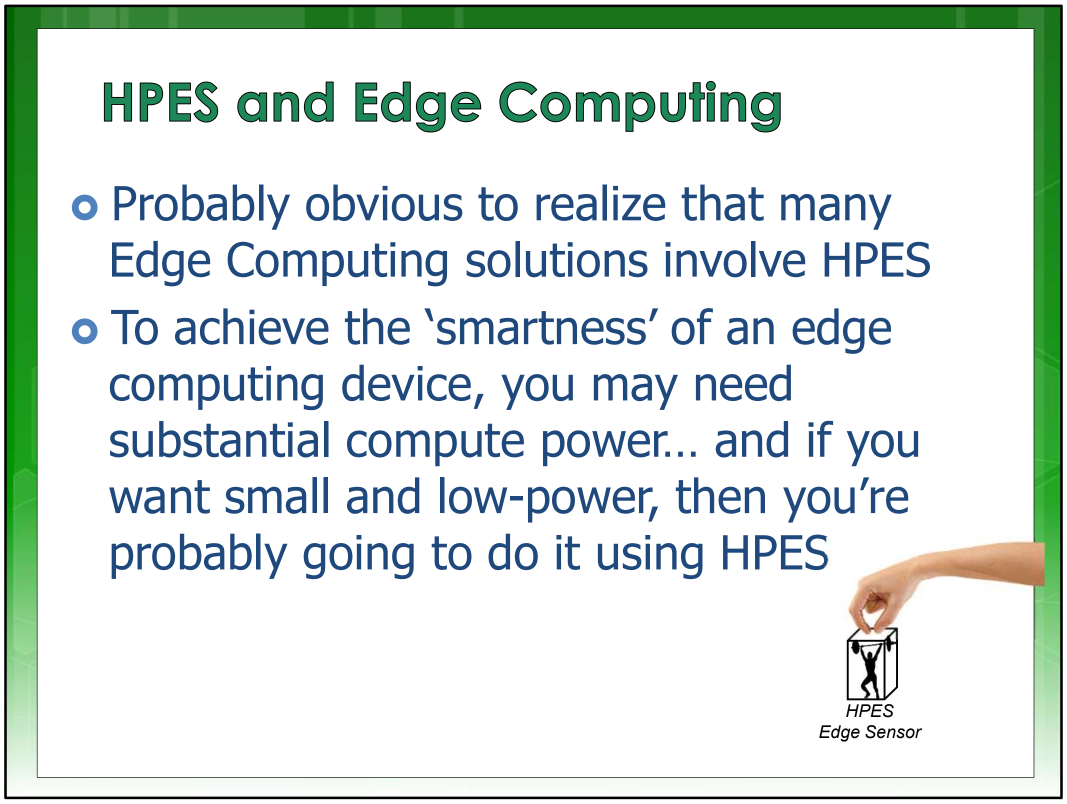


Distributed Nets at the Edge

- When we talk about **distributed nets at the edge**, think of environments where 'the cloud' may be physically impossible to reach – as might be the case in deep-sea exploration
- The edge net may need fast interaction and survive for a time unlinked to the cloud

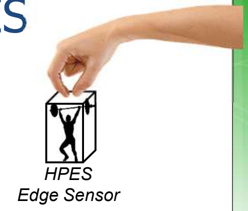


You may be encountering this issue shortly ;-)

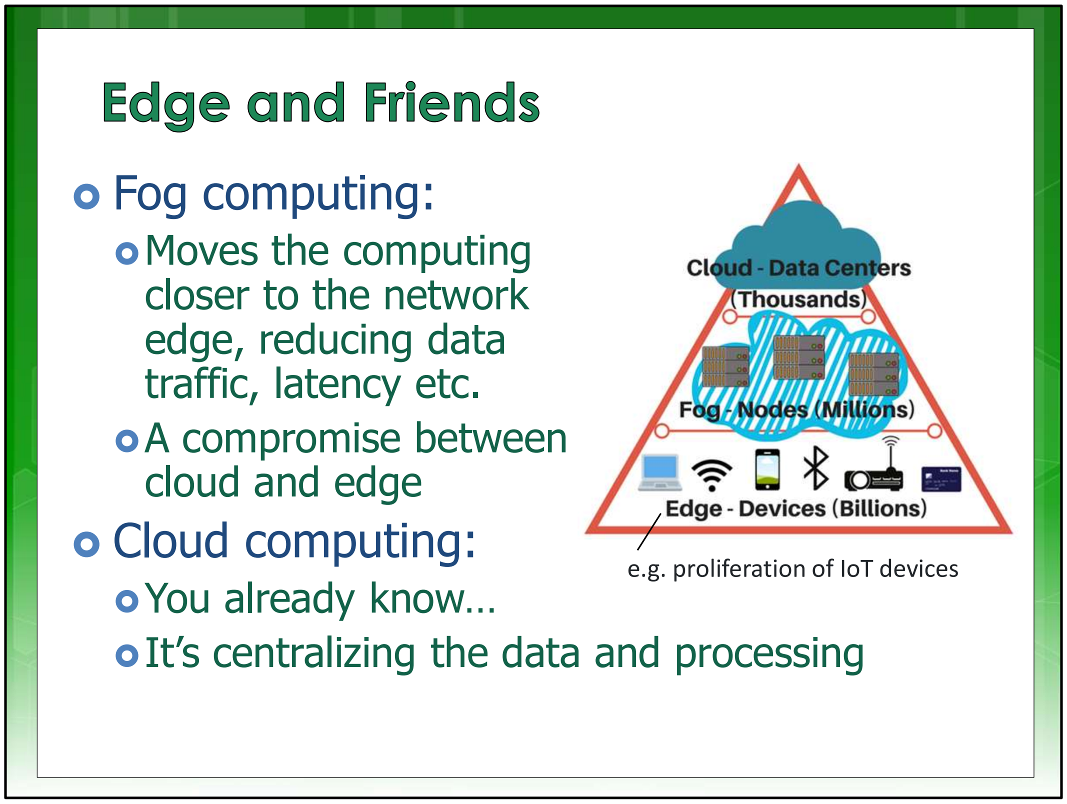


HPES and Edge Computing

- Probably obvious to realize that many Edge Computing solutions involve HPES
- To achieve the 'smartness' of an edge computing device, you may need substantial compute power… and if you want small and low-power, then you're probably going to do it using HPES

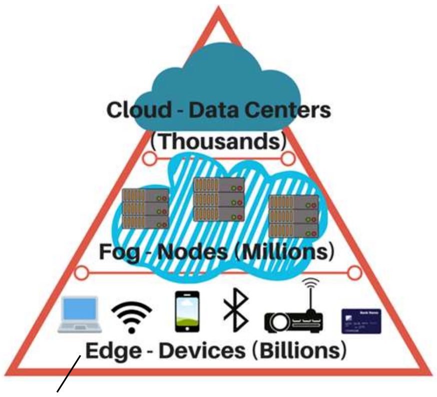


So, a more concise motivation of why HPES is becoming favoured for edge computing: For edge computing you need small, low power devices with substantial computing power, making high performance embedded systems an optimal existing solution for this need.



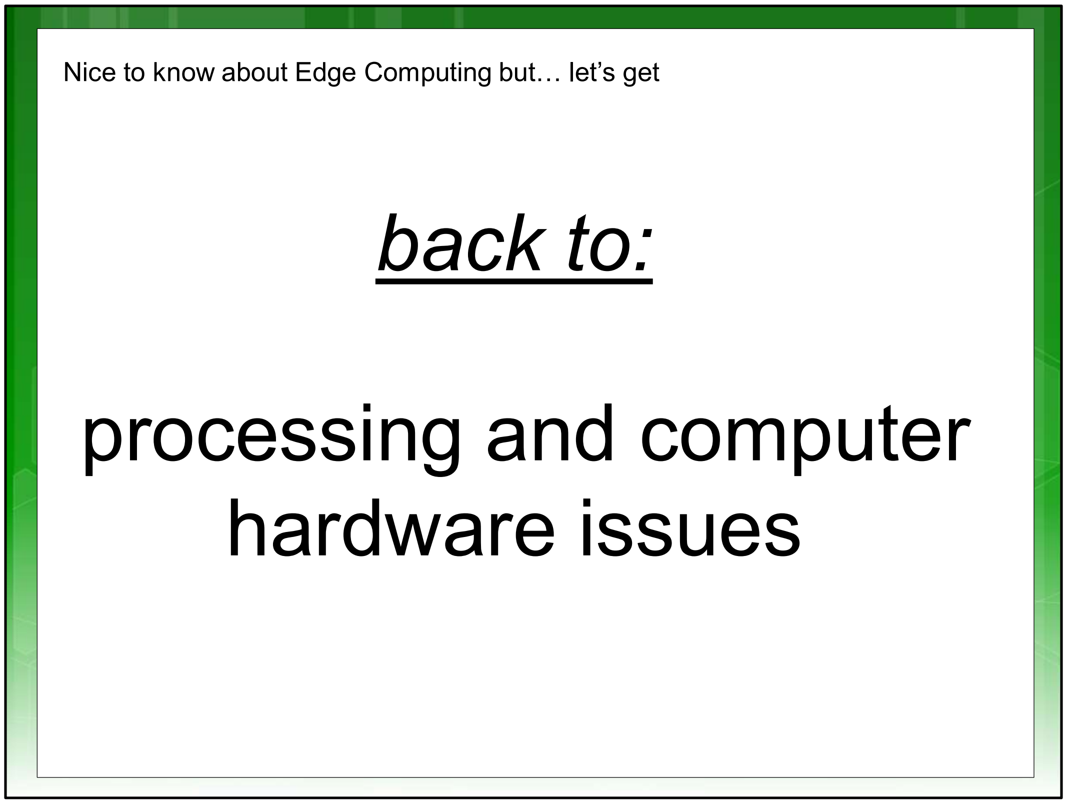
Edge and Friends

- Fog computing:
  - Moves the computing closer to the network edge, reducing data traffic, latency etc.
  - A compromise between cloud and edge
- Cloud computing:
  - You already know…
  - It's centralizing the data and processing



e.g. proliferation of IoT devices

Fog computing is where there are numerous servers on different local networks that perform most of the computing. Fog computing is bringing the computing one step closer to the edge, but not necessarily all the way to the edge. Fog computing, compared to cloud computing, is a compromise as it reduces data traffic and improving latency. The difference between edge and fog computing is that edge computing usually takes place on the edge device … for computing might be attached to some edges (i.e. run on, or hosted, on those platforms) or it may be located in separate hubs or nodes in the system. Lastly, cloud computing is the more standard approach where most of the processing takes place on a centralized device.

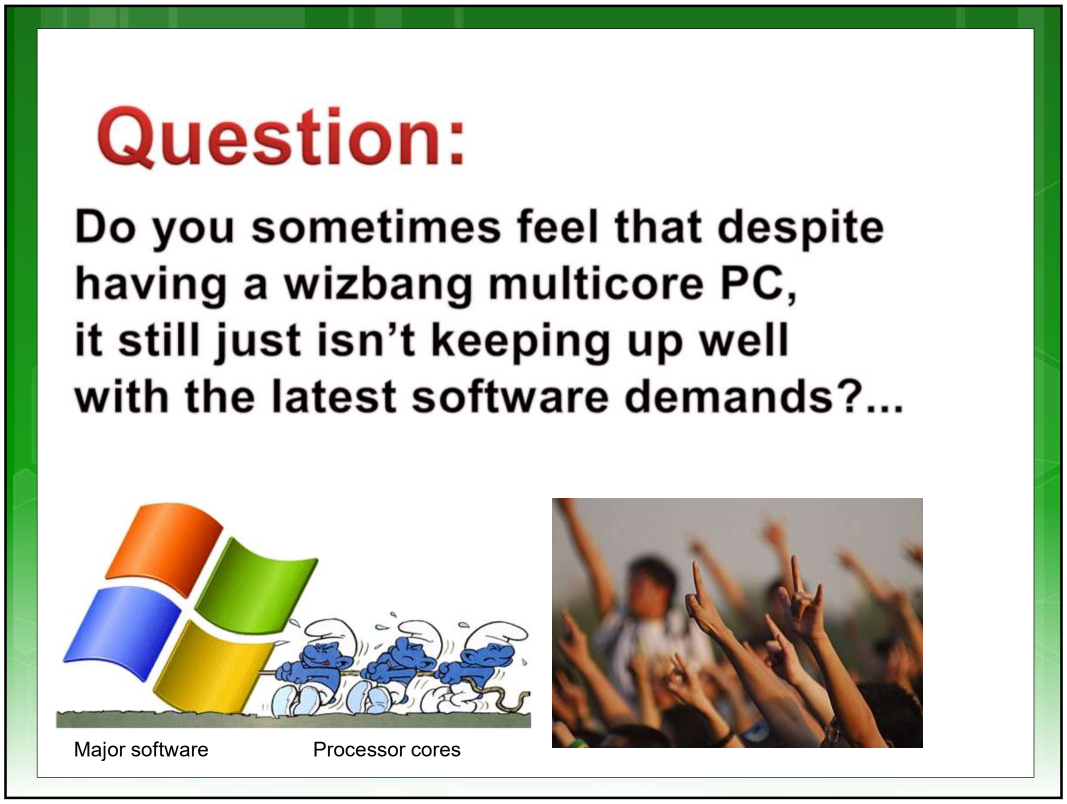


Nice to know about Edge Computing but… let's get

back to:

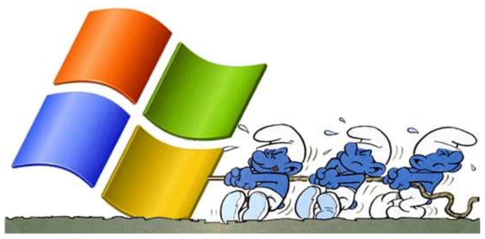
processing and computer hardware issues

We've discussed cloud, fog, and edge computing concepts very briefly; the point was that these brief terms convey much meaning in regards to the direction of a distributed system design; for which the device you are building may be an element within that broader networked system.



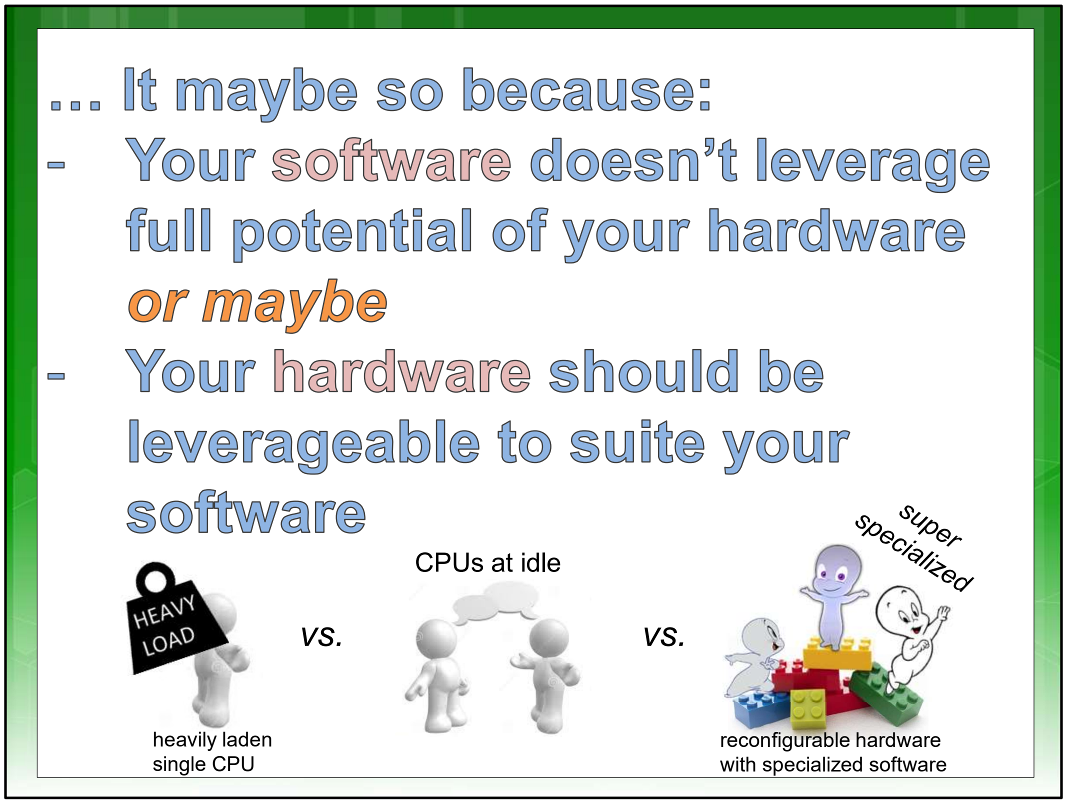
Question:

Do you sometimes feel that despite having a wizbang multicore PC, it still just isn't keeping up well with the latest software demands?...



The main goal of a computer system, from a user-perspective, is that it starts up quick (you don't have to wait for it to churn away for minutes, before being ready to use) and when you ask it to do stuff it does so quickly and gives accurate (enough) results. A highly parallel manycore system is not always the best way to achieve that objective.

But, might a slow PC be just a hardware problem? Let's think a bit on that…

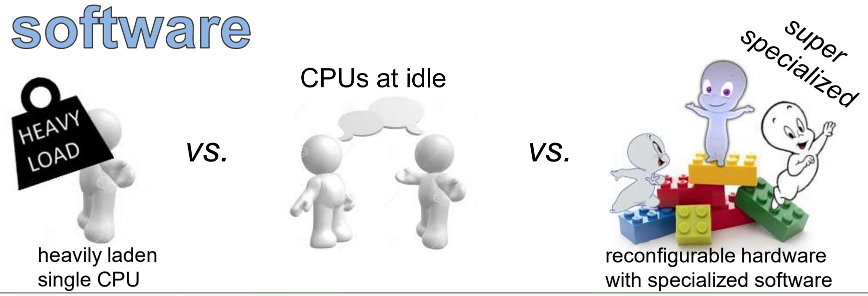


… It maybe so because:

- Your software doesn't leverage full potential of your hardware

  or maybe

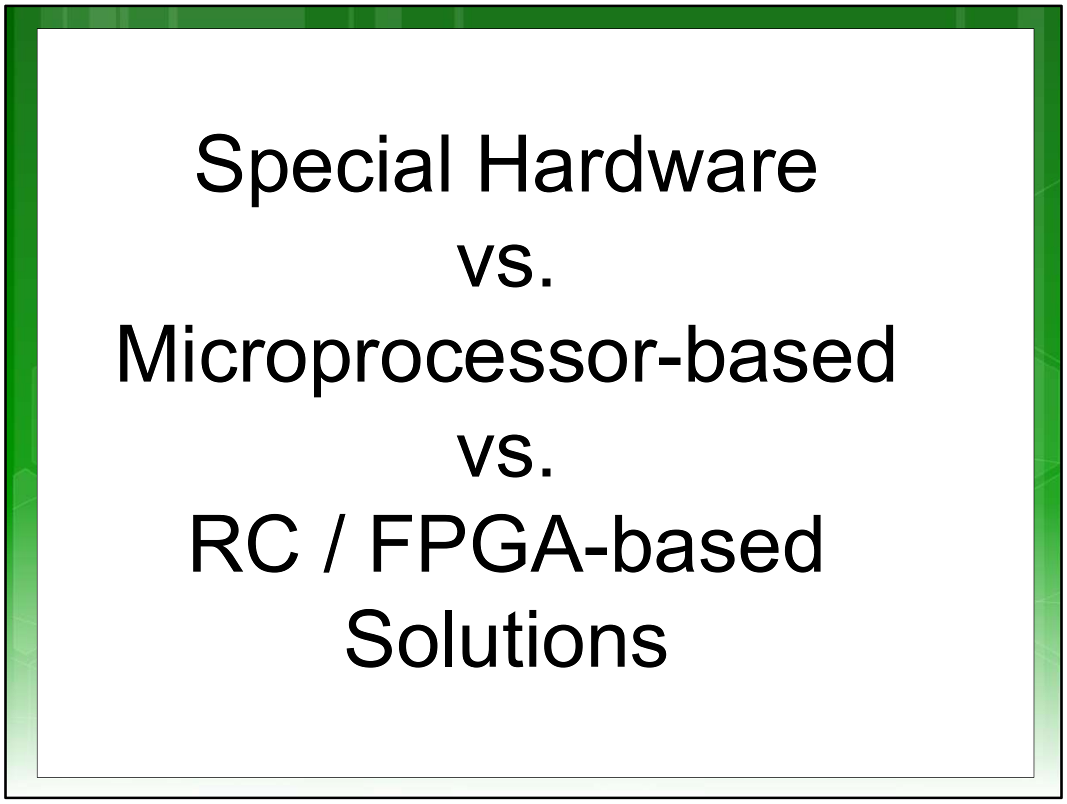
- Your hardware should be leverageable to suite your software



Certainly, a slow PC could be a hardware problem… *but* (and it might sound like I'm blaming software developers) a slow PC might be slow because it doesn't leverage the full potential of the hardware available (e.g. using a whole lot of legacy code that implements long division in fixed point, instead of utilizing the floating-point instructions that may be available on the CPU used).

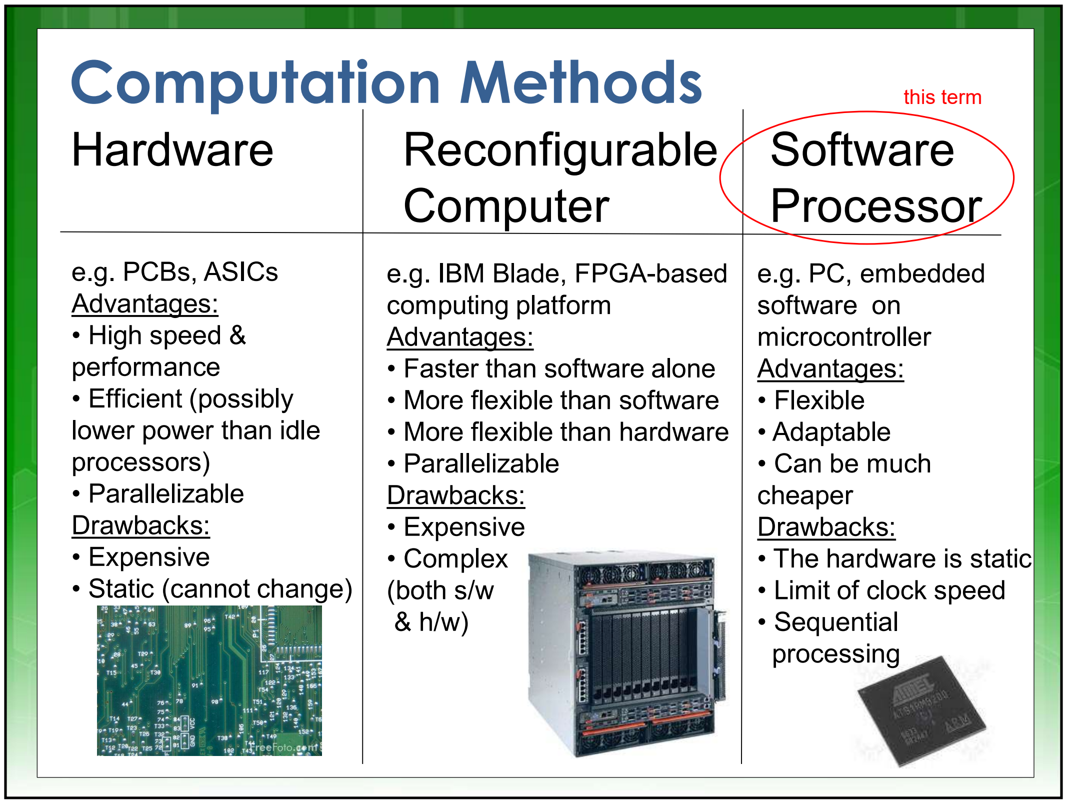
And, this might sounds mistaken to some but makes sense to FPGA developers, the software might not be leveraging the flexibility of the hardware. (Usually one thinks software is the flexible thing and it should change… but with reconfigurable computing it can also be the other way round, where, with the help of specialized tools, the hardware might be dynamically adjusted to what the software needs).

So, as the pictures at the bottom on the slide aims to suggest, it's about trying to find a suitable design choice… it might be more efficient long term (engineering costs etc.) to leave the program as dong all the work and no worries about parallelism. Might be nice just leaving other available CPUs as idle, maybe helping occasionally with comms. Or a super-specialized solution that leverages multiple cores, as well as dynamically reconfigurable hardware, and developing specialized software that runs on the custom platform.



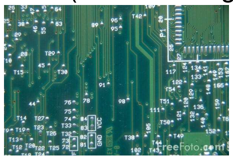
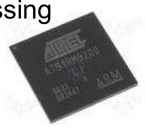
Special Hardware vs. Microprocessor-based vs. RC / FPGA-based Solutions

This slide is a reminder that processing isn’t always just done on a CPU that runs a specific instruction set.



Computation Methods

this term

Hardware	Reconfigurable Computer	Software Processor
e.g. PCBs, ASICs	e.g. IBM Blade, FPGA-based computing platform	e.g. PC, embedded software on microcontroller
Advantages:	Advantages:	Advantages:
• High speed & performance	• Faster than software alone	• Flexible
• Efficient (possibly lower power than idle processors)	• More flexible than software	• Adaptable
• Parallelizable	• More flexible than hardware	• Can be much cheaper
	• Parallelizable	
Drawbacks:	Drawbacks:	Drawbacks:
• Expensive	• Expensive	• The hardware is static
• Static (cannot change)	• Complex (both s/w & h/w)	• Limit of clock speed
		• Sequential processing
		

You could consider computation methods as being somewhat on a continuum, from a fully hardware-based computing solution, through various reconfigurable computing solutions (e.g. allowing different ways for cores to connect, and reconfigurable digital hardware that can be interfaced to CPUs), and all the way to an entirely software-based solutions that focus on the instructions to be done (nevertheless these still needing an underlying machine on which to executed the instructions on, but issues such as the way in which the instructions are implemented can be ignored).

Supplementary reading

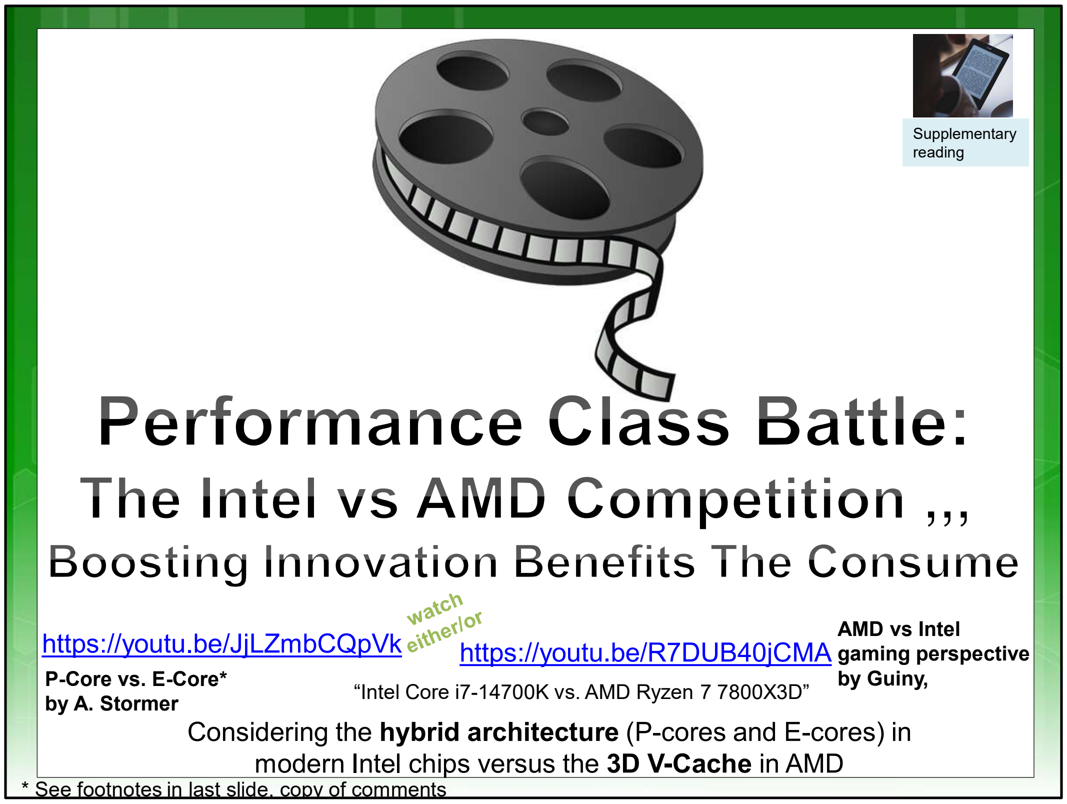
Useful video: Latest Intel Chipsets

https://youtu.be/TK7eOrgOISc

"9th Gen Intel CPU Comparison — i5 vs i7 vs i9 Benchmarks"

Core i9: "Intel Core i9 Explained"
https://youtu.be/suQnh1TvGHw?list=TLPQMjIwMjIwMjAkrwJfYkr9qA

As a respite from reading notes, and to be more familiar with the latest reconfigurable computing solutions available, do take a look at any or all of videos that are linked in the three next slides.



Supplementary reading

Performance Class Battle:
The Intel vs AMD Competition ,,,
Boosting Innovation Benefits The Consume

https://youtu.be/JjLZmbCQpVk
P-Core vs. E-Core*
by A. Stormer

watch either/or

https://youtu.be/R7DUB40jCMA
“Intel Core i7-14700K vs. AMD Ryzen 7 7800X3D”

AMD vs Intel gaming perspective by Guiny,

Considering the **hybrid architecture** (P-cores and E-cores) in modern Intel chips versus the **3D V-Cache** in AMD

* See footnotes in last slide, copy of comments

As a respite from reading notes, and to be more familiar with the latest reconfigurable computing solutions available, do take a look at any or all of videos that are linked in the three next slides.

Points re P-Cores vs E-Cores

P-cores = Performance-cores

E-cores = Efficient-cores

These part of Intel's hybrid architecture, designed around balancing high-speed, heavy-duty processing with power efficiency.

P-cores -> maximum speed and high Instructions Per Clock (IPC) ratings

E-cores -> optimized for throughput, handling background tasks. If using both P- and E-cores in chip could plan around cases where no urgent/high priority jobs put P-cores to sleep and just use E-cores to save power, if urgent tasks then P-cores wake up and do bulk of processing with assistance from E-cores.

P-cores support Hyper-Threading (two threads per core). E-cores do not (one thread per core). So basically can plan around P-cores having two threads simultaneously not needing to do as much timesharing.

Multiple E-cores can fit in space of one P-core, so can mean less space needed and lower cost.

Comment on calculations used in P- vs E-cores:

The performance calculation that the author uses is called “Theoretical Aggregate Throughput”, it works as follows:

Total_Performance = (Np x Tp x Fp) + (Ne x Te x Fe)

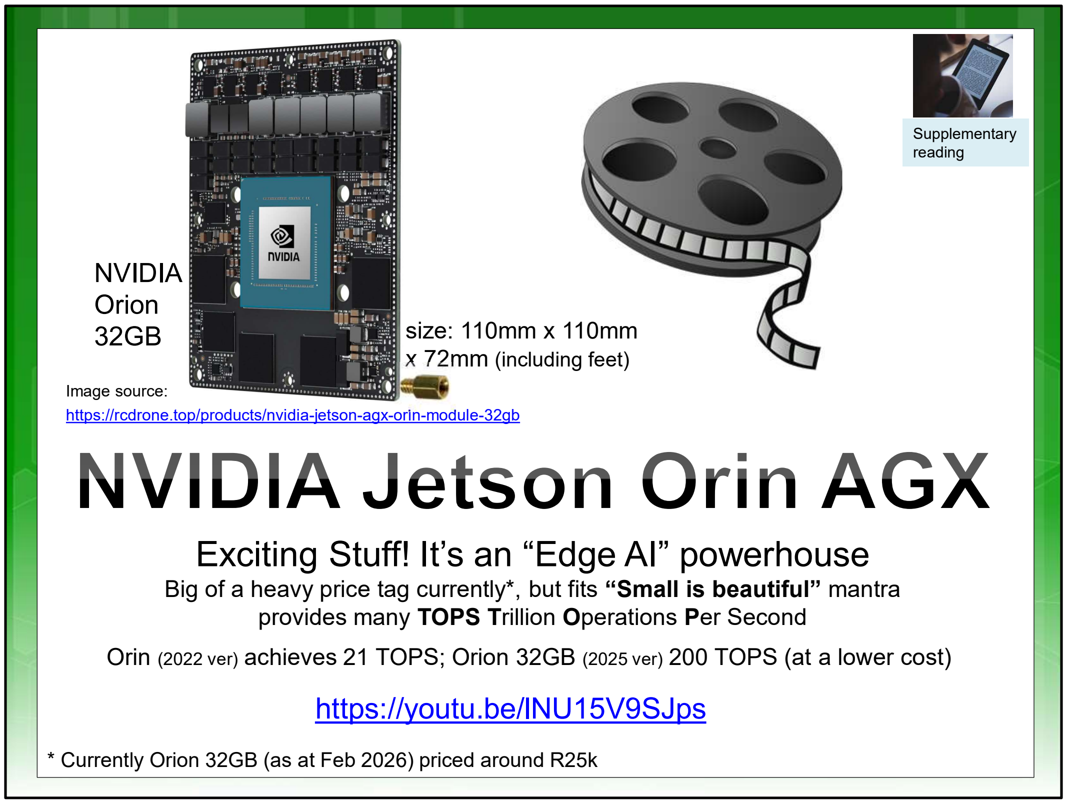
Np = num p-cores processor has Tp = number threads p-core has Fp = Frequency of p-core clock

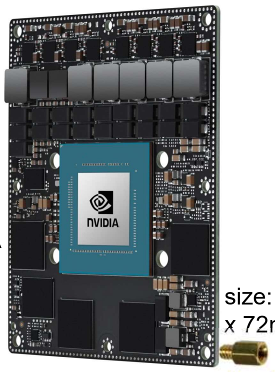
Ne = num e-cores processor has Te = threads per e-core Fe = Frequency of e-core clock

Adding the two together because the processor has a number of both core types

It’s kind of a optimistic aggregate throughput in the number of instructions that gets through (it doesn’t cater for things like data dependencies that may cause NOPs or other blocking that slows down throughput)

In gaming, oftentime more cores, not necessarily faster cores, are better (how difficult is it to make decisions on moves compared to drawing the displays and deciding what textures to use or generate and scenery to load).





NVIDIA Orion 32GB

size: 110mm x 110mm x 72mm (including feet)

Image source: https://rcdrone.top/products/nvidia-jetson-agx-orin-module-32gb

Supplementary reading

NVIDIA Jetson Orin AGX

Exciting Stuff! It's an "Edge AI" powerhouse

Big of a heavy price tag currently*, but fits **"Small is beautiful"** mantra provides many **TOPS T**rillion **O**perations **P**er Second

Orin (2022 ver) achieves 21 TOPS; Orion 32GB (2025 ver) 200 TOPS (at a lower cost)

https://youtu.be/lNU15V9SJps

* Currently Orion 32GB (as at Feb 2026) priced around R25k

Note as to why we're dipping more into GPU and small is beautiful …

While these GPU/many core platforms often use CUDA/TensorRT, the underlying need for specialized hardware acceleration is why the course now prioritizes **Verilog HDL** as opposed to GPU programming (the UCT CS department covers GPU programming thoroughly)

Digital Accelerators: New Practice

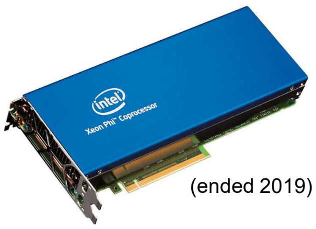
Digital accelerator cards (including GPGPUs) are ever increasing in popularity, including use in data centres.

(launched end 2018)

Xilinx Alevo: Adaptable Accelerator Cards for Data Centres
https://www.xilinx.com/products/boards-and-kits/alveo.html

Program with OpenCL or Xilinx's owns accelerator design suite.

And… some of this it may involve designing specialized compute architectures for the need (using a combination of languages and tools e.g. OpenCL / Verilog, C, R, etc.)

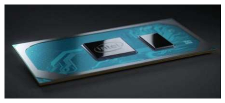


(ended 2019)

HP Intel 5110P Xeon Phi Coprocessor Kit
Intel Xeon Phi
Where and why it's no longer being made, replaced by Xeon Scalable Platform

Replaced by (from 2020)

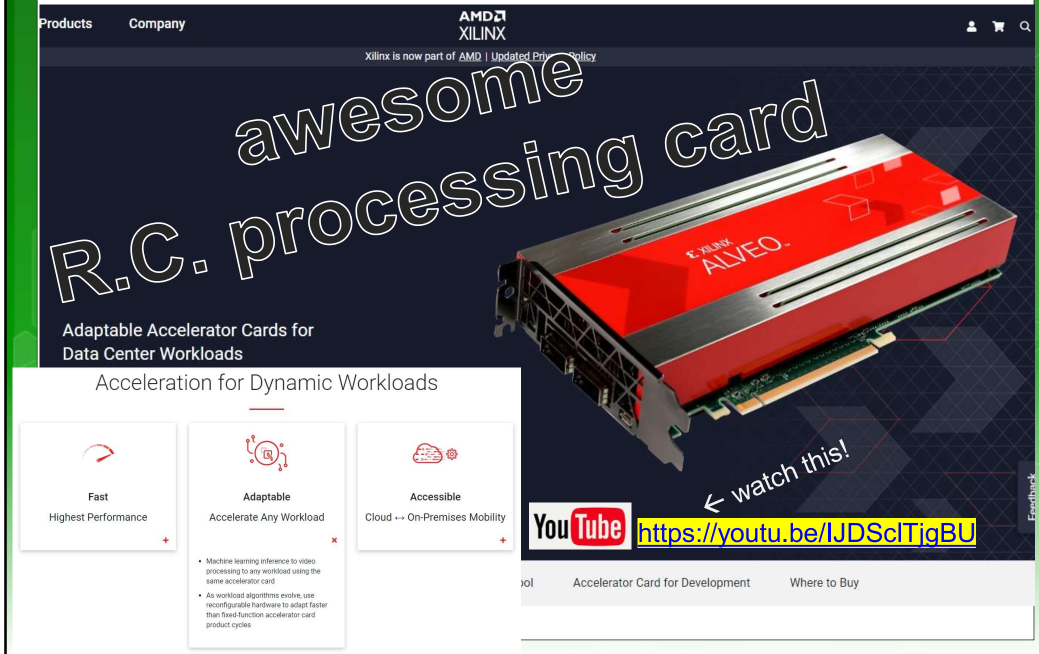
Intel Ice Lake (10 nm process) 10th generation core, successor for Xeon Phis.



Core i7 1068G7 (2020 debut) 4 CPU cores + 64 Iris+ GPCPU cores

* What was it said in the "Berkeley Landscape" … "Small is beautiful" and "manycore is the future of Computing".

AMD Xilinx Alveo



Products
Company
AMD
XILINX
Xilinx is now part of AMD | Updated Privacy Policy
XILINX ALVEO
Feedback
You Tube
ool
Accelerator Card for Development
Where to Buy

awesome

R.C. processing card

Adaptable Accelerator Cards for Data Center Workloads

Acceleration for Dynamic Workloads

Fast
Highest Performance

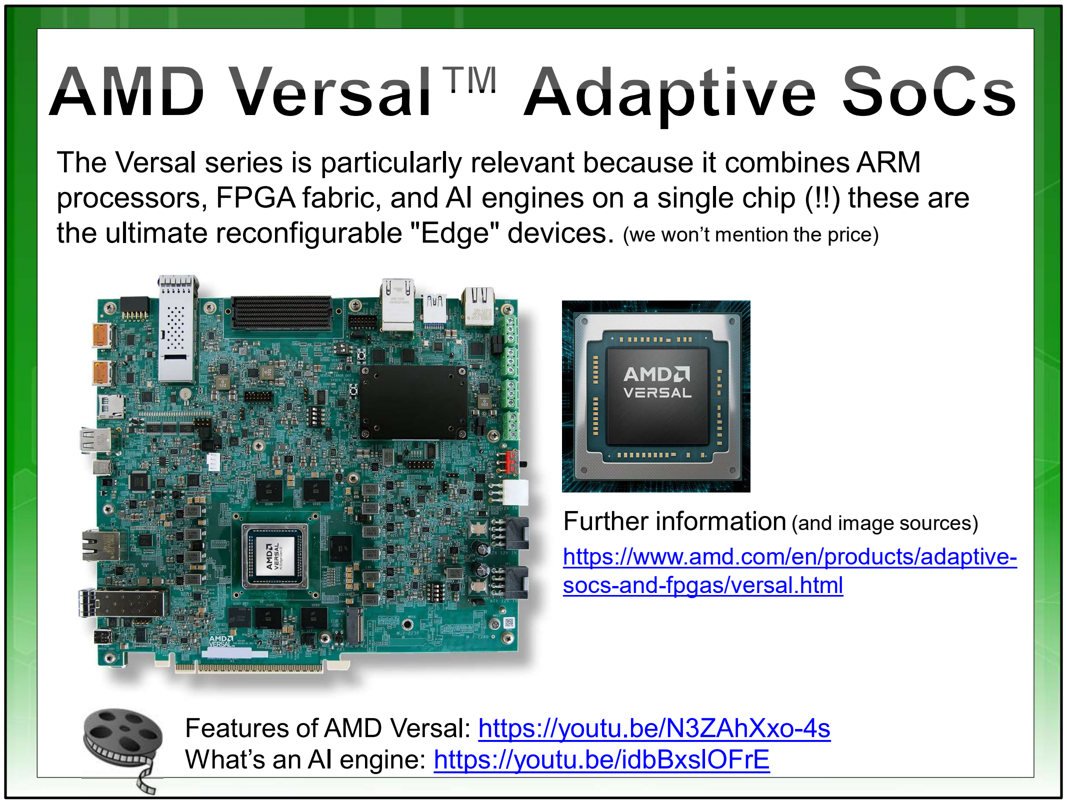
Adaptable
Accelerate Any Workload

- Machine learning inference to video processing to any workload using the same accelerator card
- As workload algorithms evolve, use reconfigurable hardware to adapt faster than fixed-function accelerator card product cycles

Accessible
Cloud ↔ On-Premises Mobility

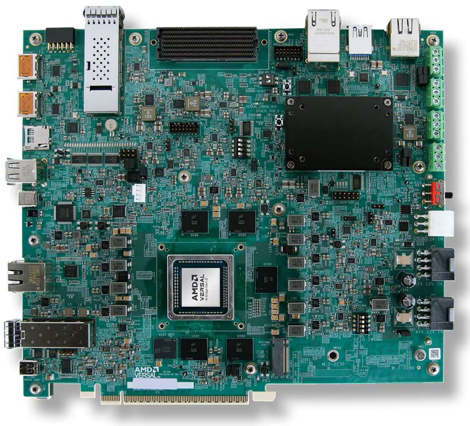
← watch this!

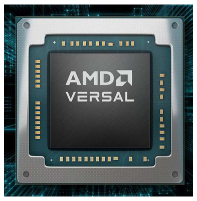
https://youtu.be/lJDSclTjgBU



AMD Versal™ Adaptive SoCs

The Versal series is particularly relevant because it combines ARM processors, FPGA fabric, and AI engines on a single chip (!!) these are the ultimate reconfigurable "Edge" devices. (we won't mention the price)





Further information (and image sources)
https://www.amd.com/en/products/adaptive-socs-and-fpgas/versal.html

Features of AMD Versal: https://youtu.be/N3ZAhXxo-4s
What's an AI engine: https://youtu.be/idbBxslOFrE

This is the latest greatest type of development device for complex edge devices.

If you desperately need to know the cost for the evaluation kit, it's around 7k USD (around 112k ZAR)

but now back to Earth …

Tools that we will be using

- MATLAB / OCTAVE ✓ (and MATLAB PCT)
- C / C++ (for host and coordination)
- MPICH (distributed computing)
- Verilog HDL (i.e. to design our own special-purpose processor or digital accelerator)

Hope you're suitable ready!

About Pracs & Prac 2

- Status of Prac1 completion?
- Prac2: MATLAB Parallel Computing Toolkit

  (getting deeper into parallel programming)
- Recommend attempting to install nVidia or Intel or AMD GPU drivers on your own if available, MATLAB can make use of it
  - For coding suggest
    Code::Blocks https://www.codeblocks.org/

    OR:

    DevC++ http://dev-cpp.com/
- RedLab machines run Windows. If you want try Ubuntu Linux if can boot from USB!? *

* It might be a bit slow to boot, but should be fine for the pracs as none of them use hectic compiling, program processing or large amounts of data/

closing remarks & reminders…

End of Lecture 3

ADDENDUM: notes regarding slide 15

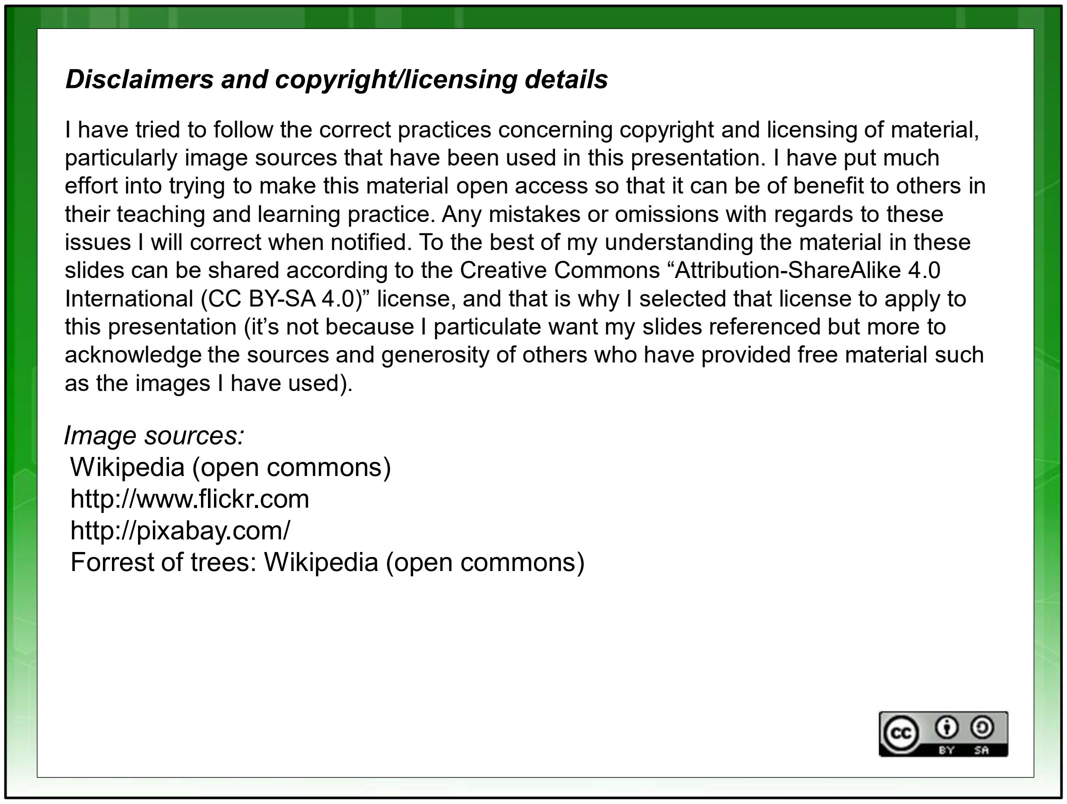
As a respite from reading notes, and to be more familiar with the latest reconfigurable computing solutions available, do take a look at any or all of videos that are linked in the three next slides.

Points re P-Cores vs E-Cores
P-cores = Performance-cores
E-cores = Efficient-cores
These part of Intel's hybrid architecture, designed around balancing high-speed, heavy-duty processing with power efficiency.
  P-cores -> maximum speed and high Instructions Per Clock (IPC) ratings
  E-cores -> optimized for throughput, handling background tasks. If using both P- and E-cores in chip could plan around cases where no urgent/high priority jobs put P-cores to sleep and just use E-cores to save power, if urgent tasks then P-cores wake up and do bulk of processing with assistance from E-cores.

  P-cores support Hyper-Threading (two threads per core). E-cores do not (one thread per core). So basically can plan around P-cores having two threads simultaneously not needing to do as much timesharing.
  Multiple E-cores can fit in space of one P-core, so can mean less space needed and lower cost.

Comment on calculations used in P- vs E-cores:
 The performance calculation that the author uses is called "Theoretical Aggregate Throughput", it works as follows:
 Total_Performance = (Np x Tp x Fp) + (Ne x Te x Fe)
  Np = num p-cores processor has Tp = number threads p-core has Fp = Frequency of p-core clock
  Ne = num e-cores processor has Te = threads per e-core Fe = Frequency of e-core clock
  Adding the two together because the processor has a number of both core types
 It's kind of a optimistic aggregate throughput in the number of instructions that gets through (it doesn't cater for things like data dependencies that may cause NOPs or other blocking that slows down throughput)
 In gaming, oftentime more cores, not necessarily faster cores, are better (how difficult is it to make decisions on moves compared to drawing the displays and deciding what textures to use or generate and scenery to load).

Not included in exam syllabus



Disclaimers and copyright/licensing details

Image sources:
Wikipedia (open commons)
http://www.flickr.com
http://pixabay.com/
Forrest of trees: Wikipedia (open commons)