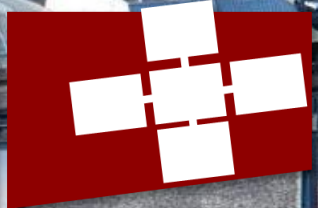


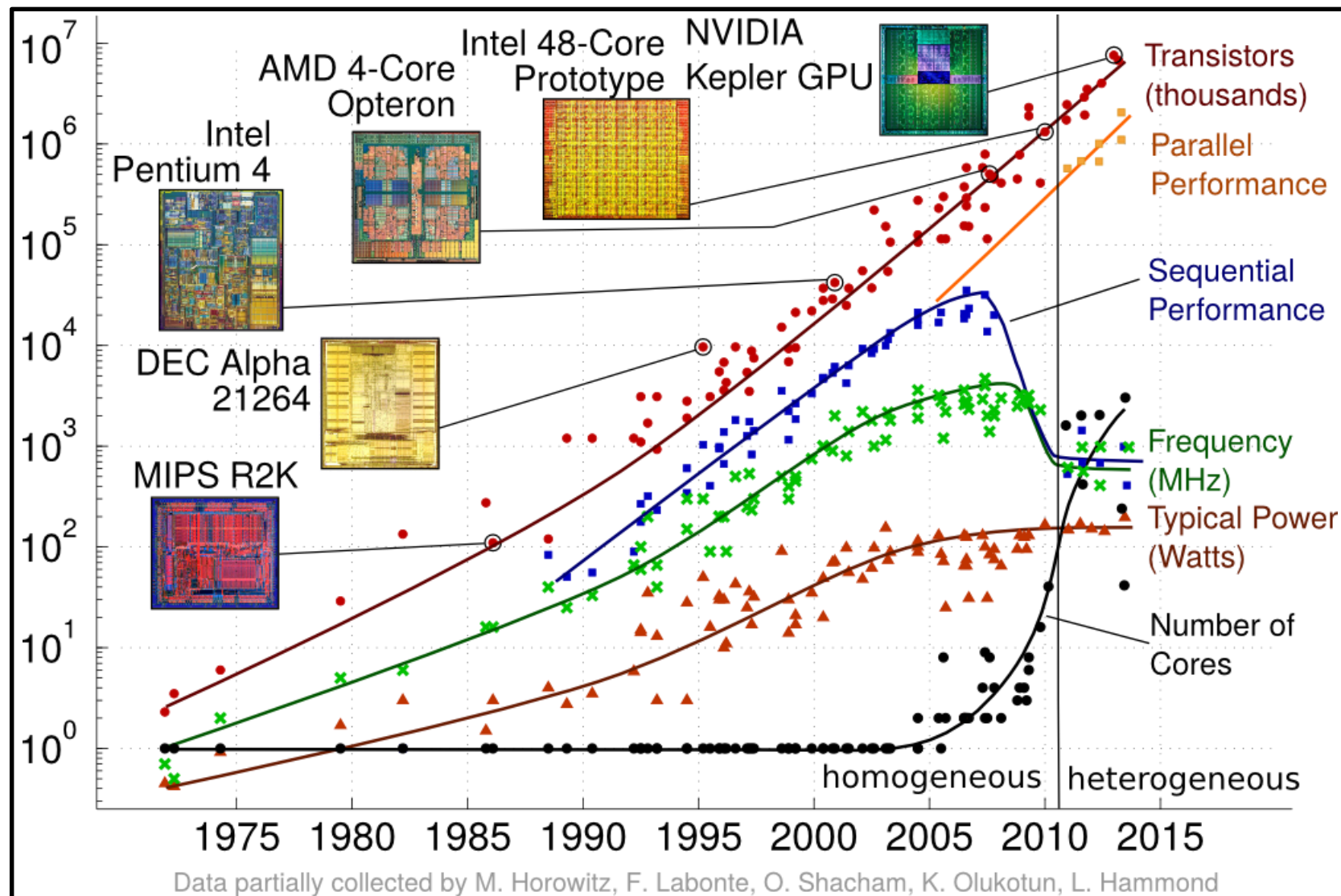
J. DE FINE LICHT AND T. HOEFLER

Productive parallel programming on FPGA using High-level Synthesis

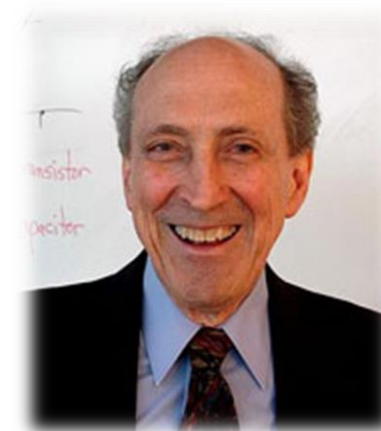
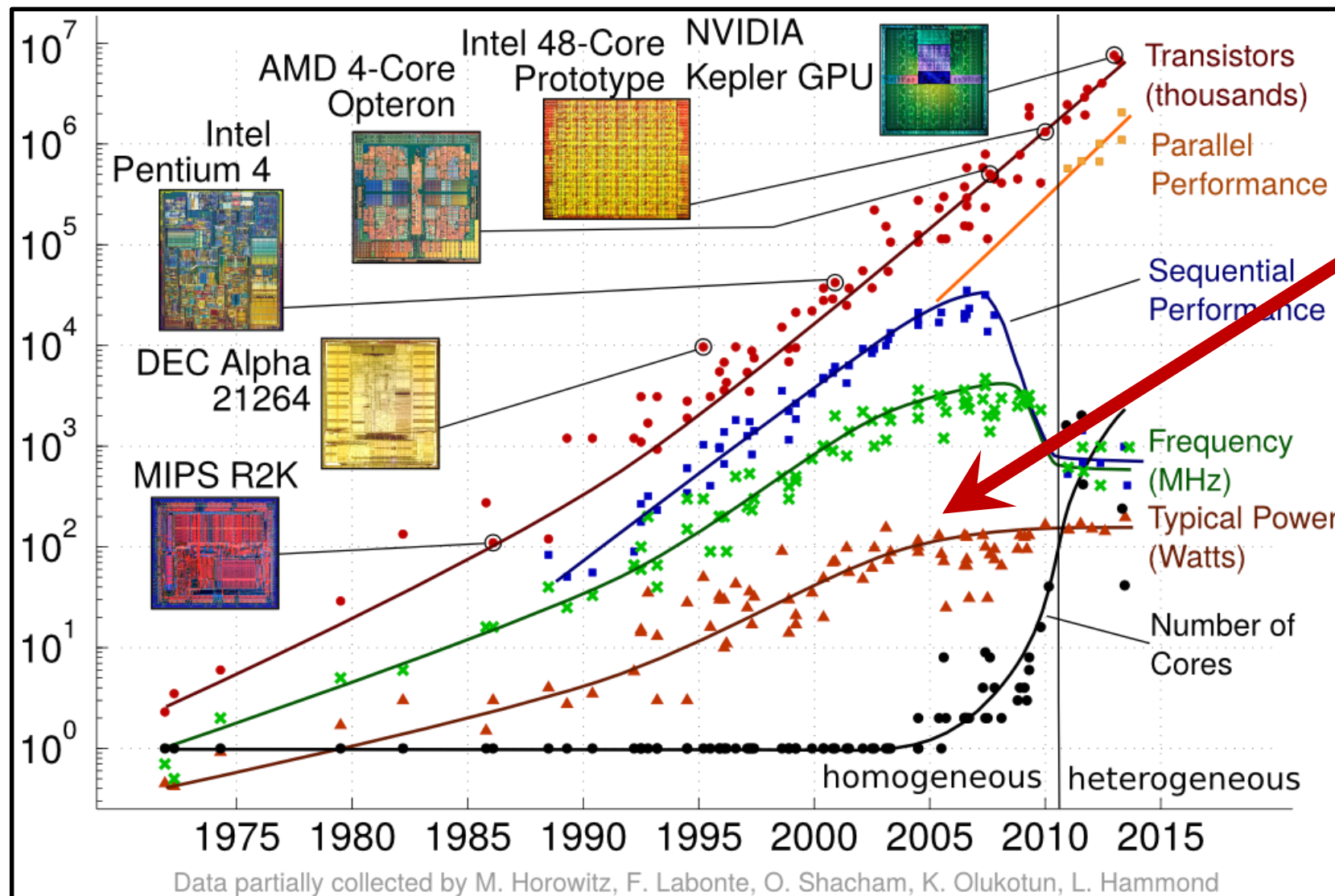


Changing hardware constraints and the physics of computing

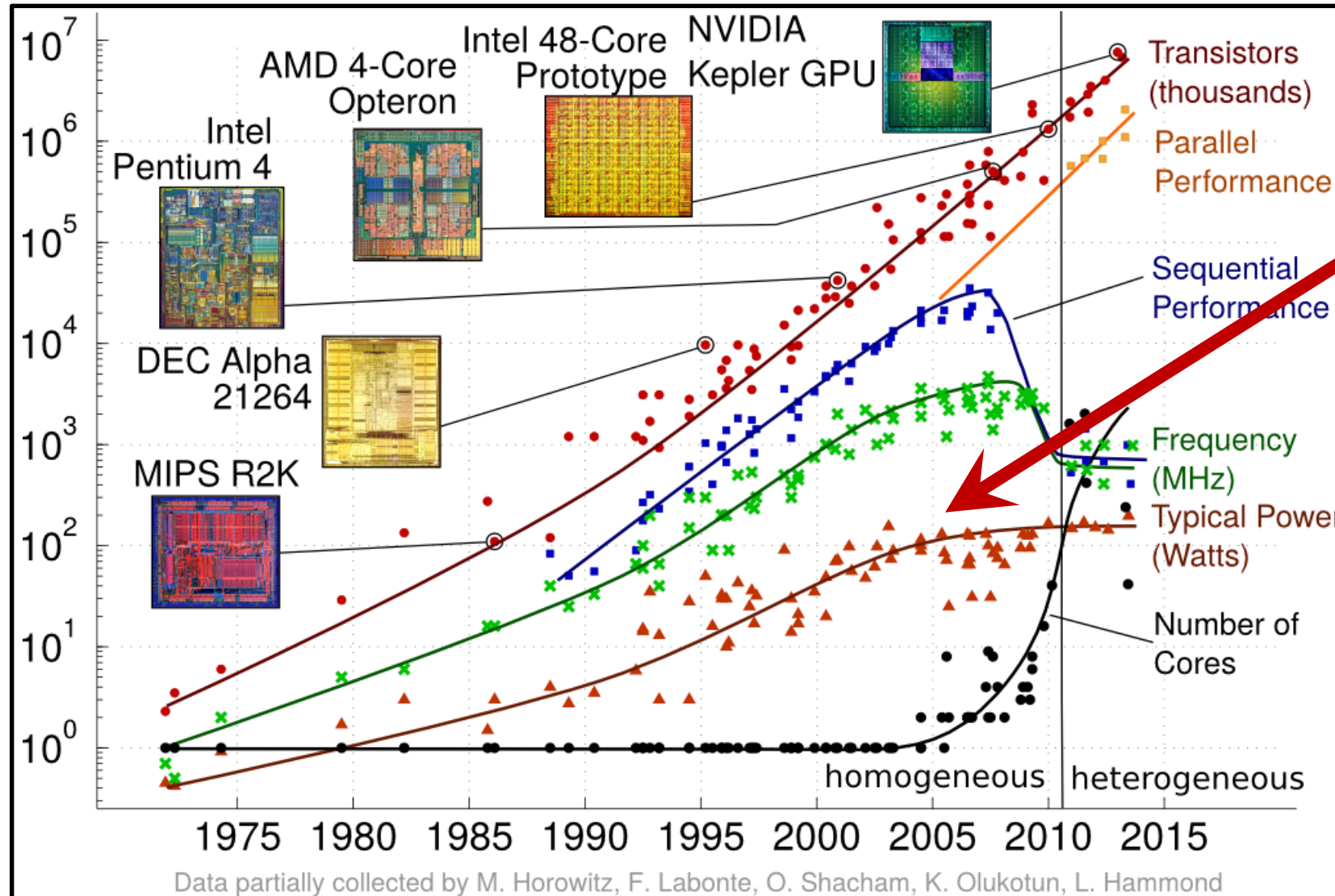
Changing hardware constraints and the physics of computing



Changing hardware constraints and the physics of computing



Changing hardware constraints and the physics of computing



Moore's law really is dead this time

The chip industry is no longer going to treat Gordon Moore's law as the target to aim for.

PETER BRIGHT - 2/11/2016, 2:22 AM

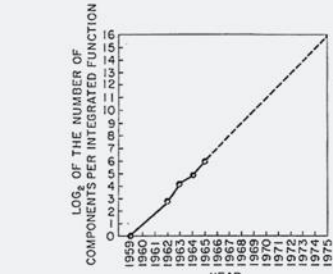


Fig. 2 Number of components per integrated function for minimum cost per component extrapolated vs time.

Gordon Moore's original graph, showing projected transistor counts, long before the term "Moore's law" was coined. Moore's original observation was that transistor density doubled every year; in 1975, this was expected to double every two years.

Moore's law has died at the age of 51 after an extended illness.

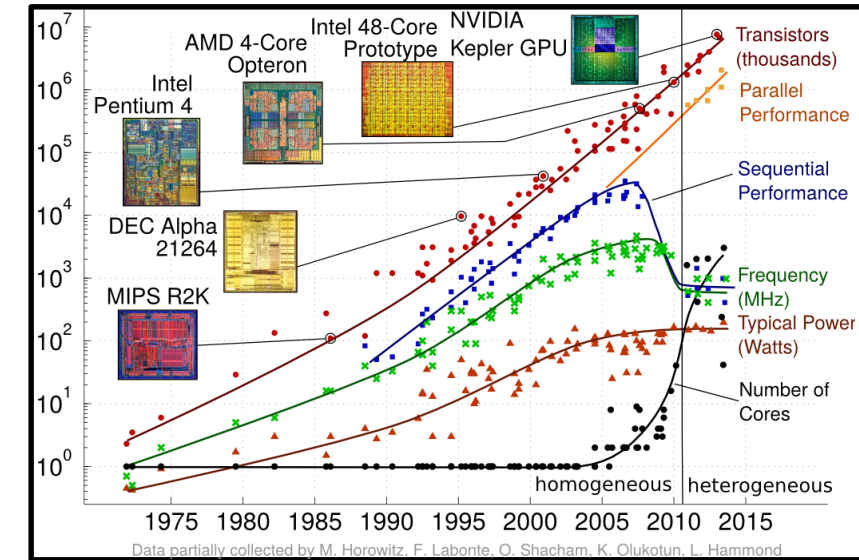
In 1965, Intel co-founder Gordon Moore made an observation that the number of components in integrated circuits was doubling every 12 months or so. Moreover, as this site wrote extensively about in 2003, that the number of transistors per chip that resulted in the lowest price per transistor was doubling every 12 months. In 1965, this meant that 50 transistors per chip offered the lowest per-transistor cost; Moore predicted that by 1970, this would rise to 1,000 components per chip, and that the price per transistor would drop by 90 percent.

With a little more data and some simplification, this observation became "Moore's law": the number of transistors per chip would double every 12 months.

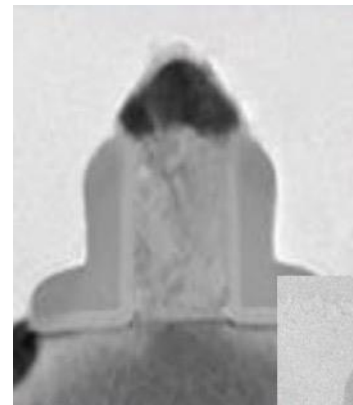
Gordon Moore's observation was not driven by any particular scientific or engineering necessity. It was a reflection on just how things happened to turn out. The silicon chip industry took note and started using it not merely as a descriptive, predictive observation, but as a prescriptive, positive law: a target that the entire industry should hit.

Moore's original

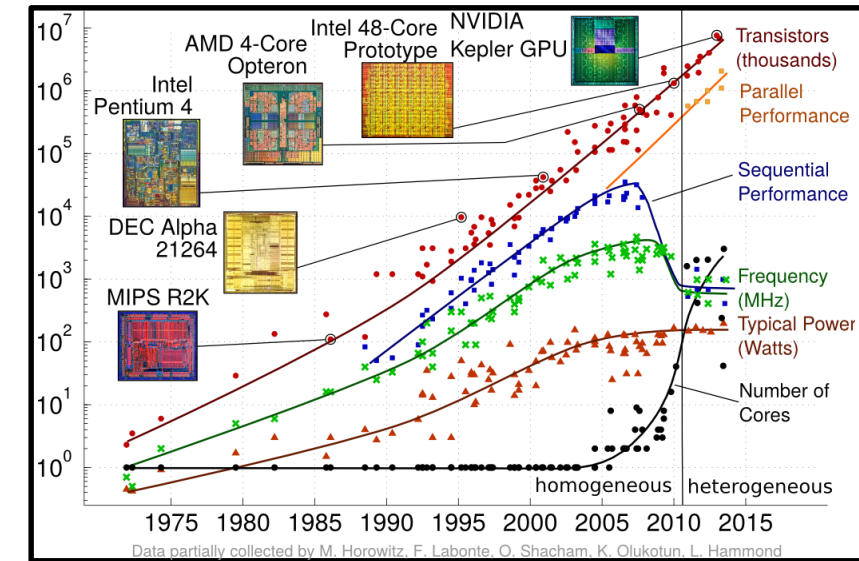
Changing hardware constraints and the physics of computing



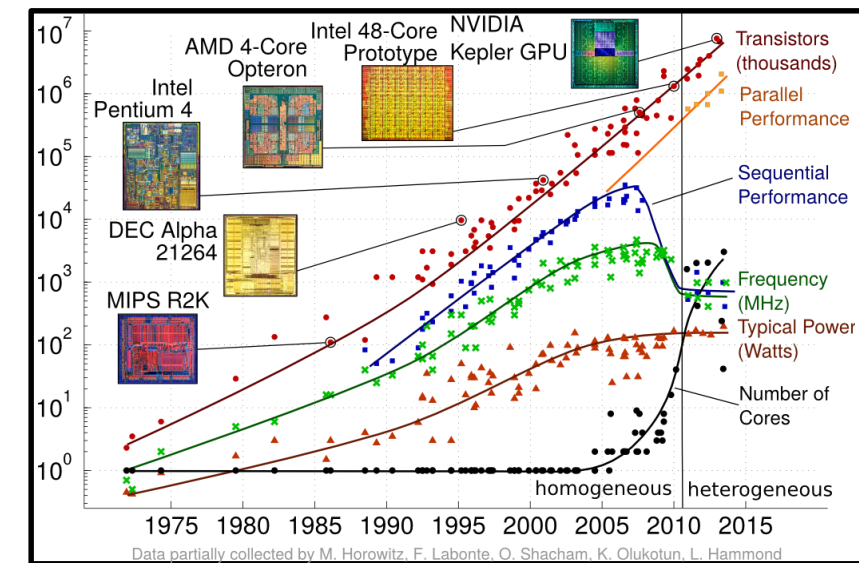
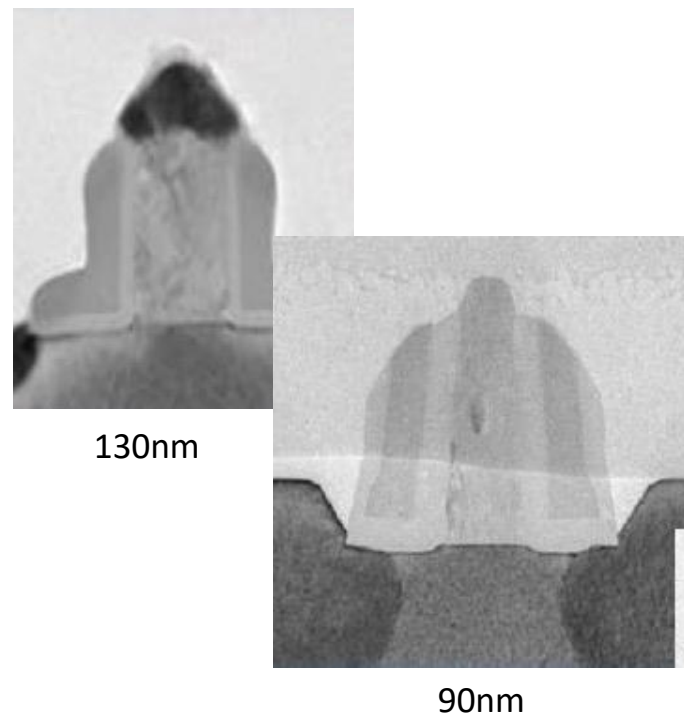
Changing hardware constraints and the physics of computing



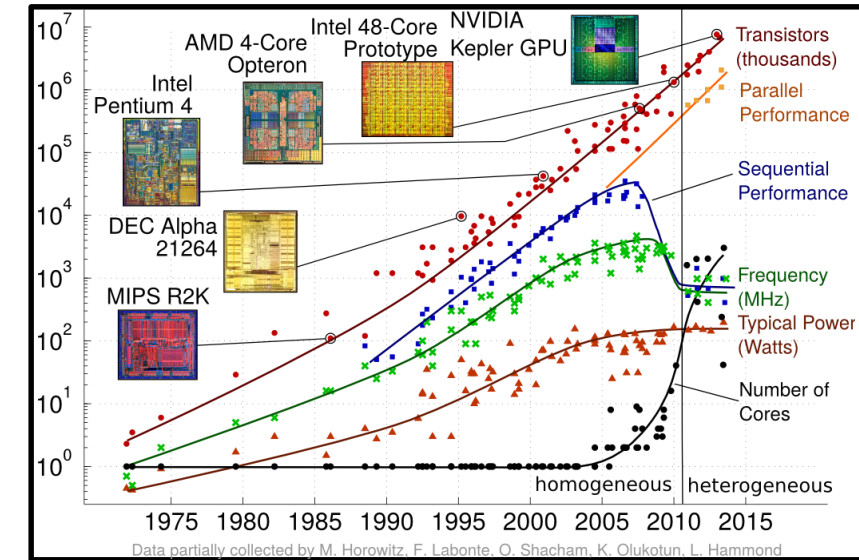
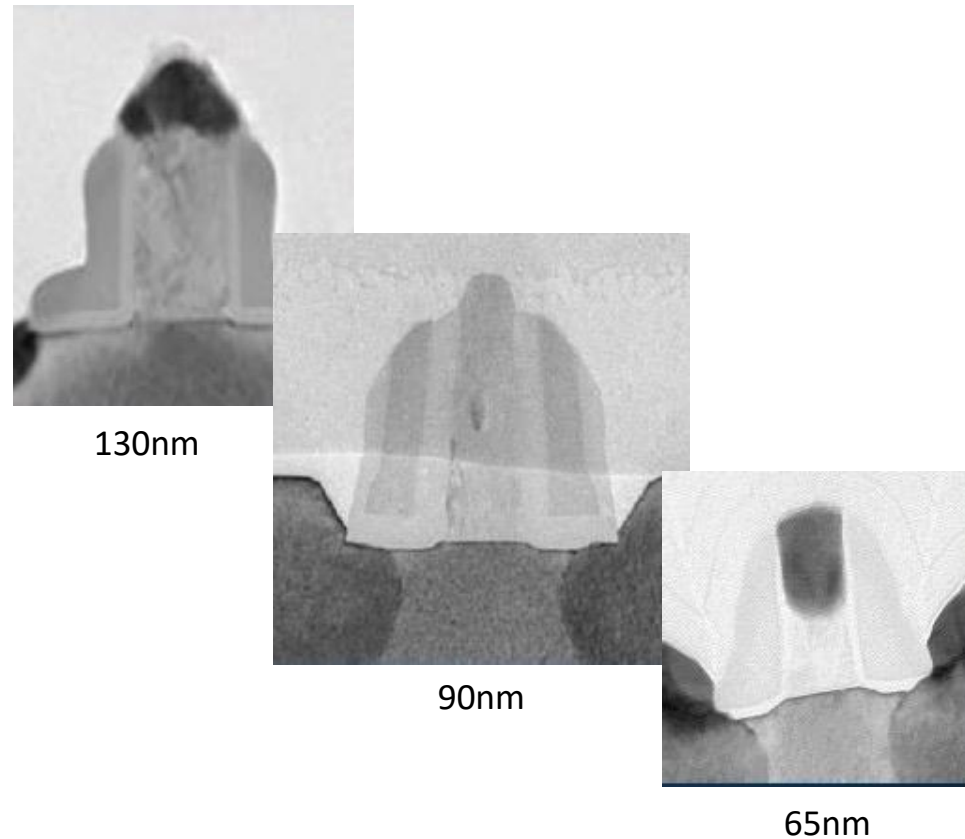
130nm



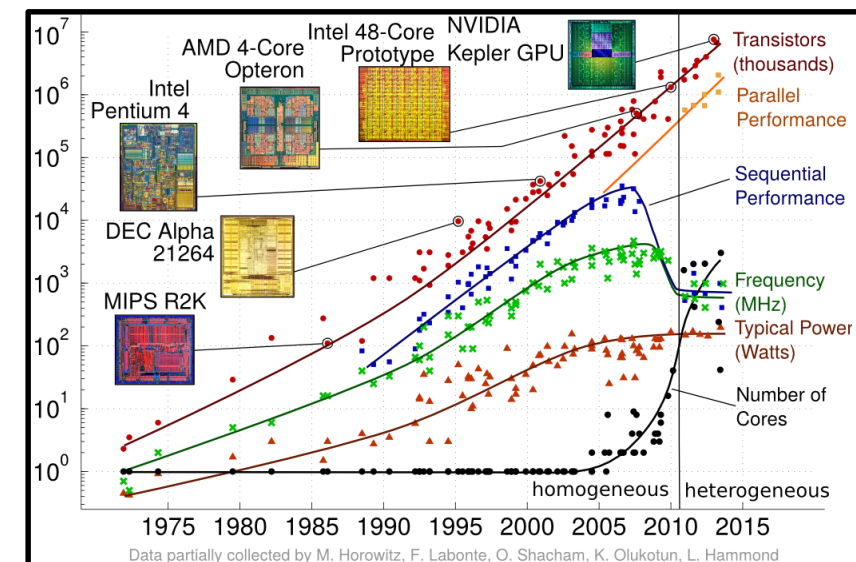
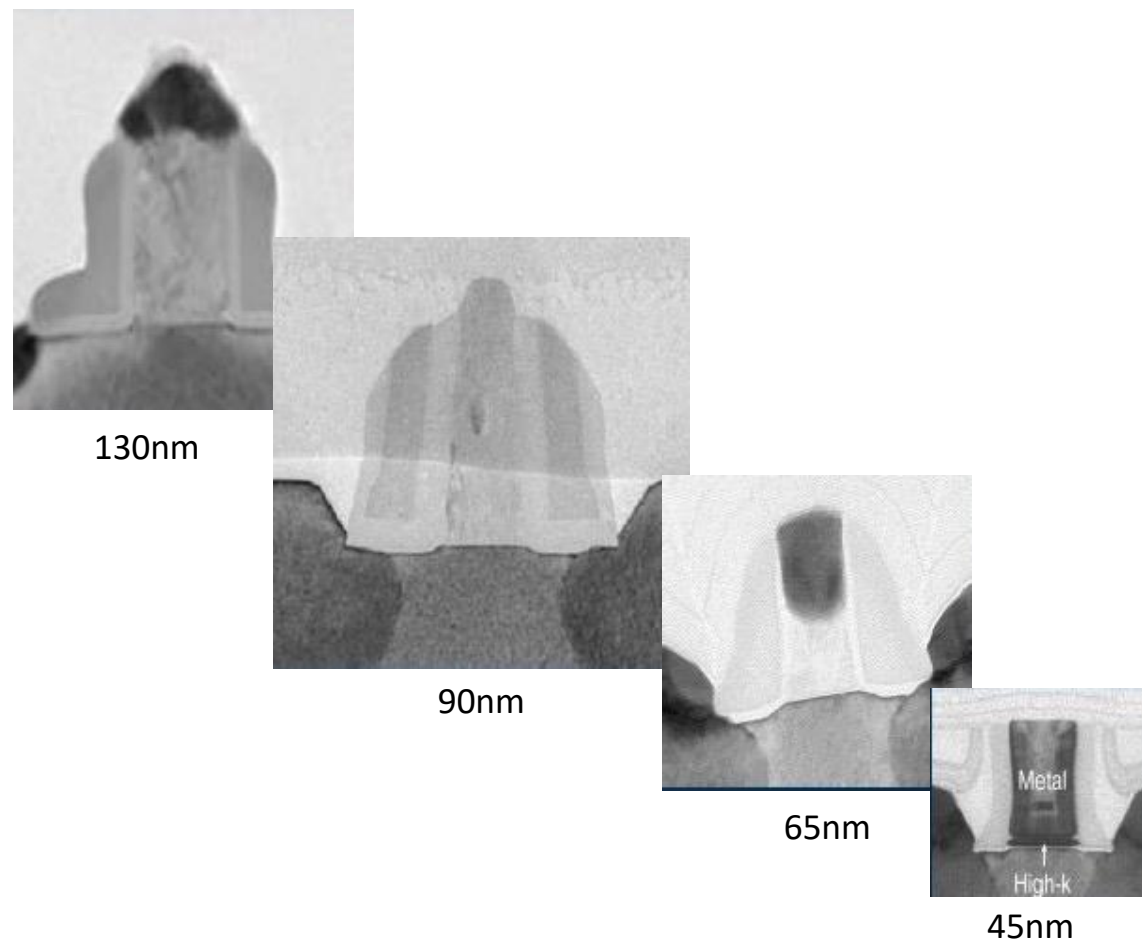
Changing hardware constraints and the physics of computing



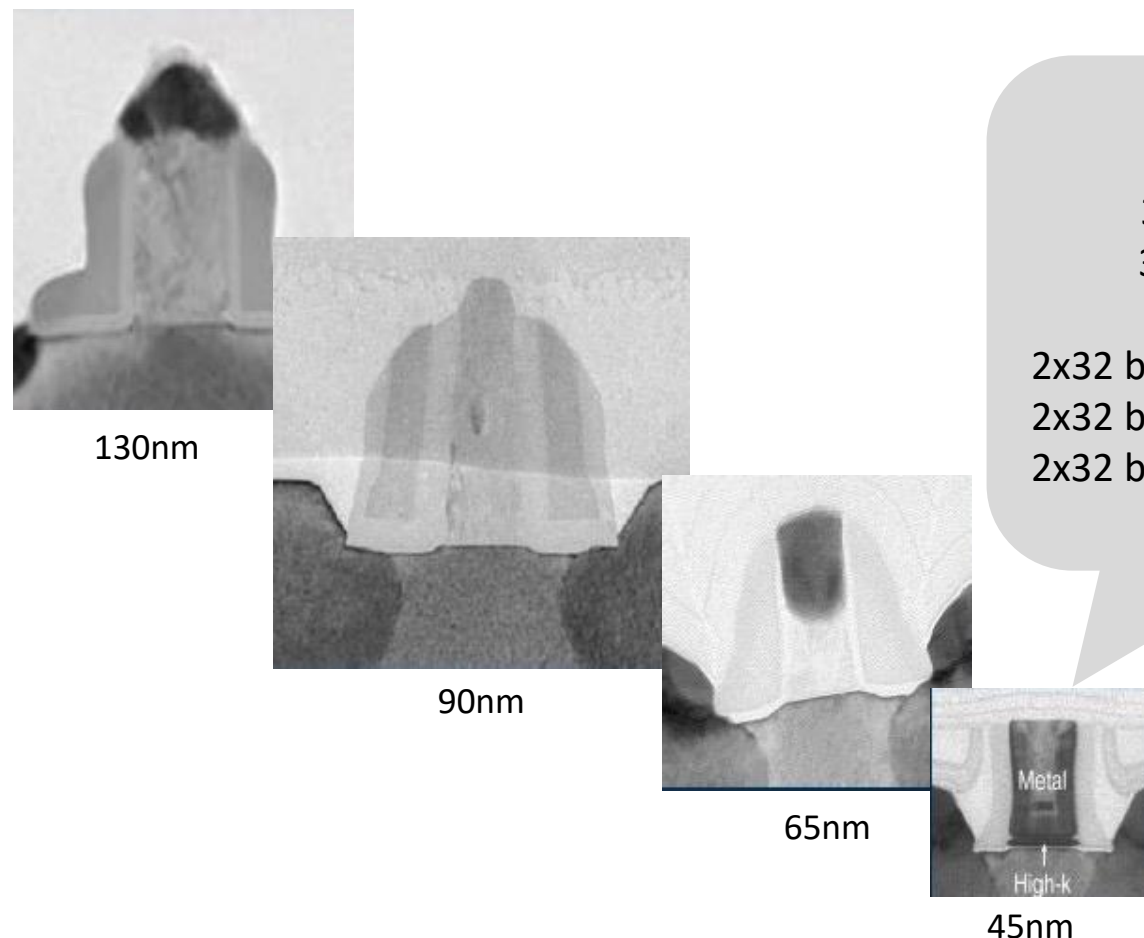
Changing hardware constraints and the physics of computing



Changing hardware constraints and the physics of computing



Changing hardware constraints and the physics of computing



0.9 V [1]

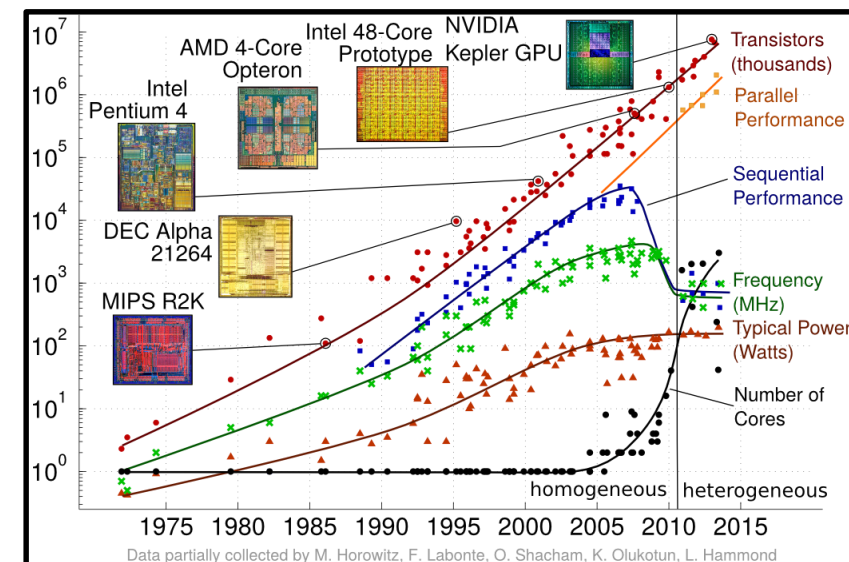
32-bit FP ADD: 0.9 pJ

32-bit FP MUL: 3.2 pJ

2x32 bit from L1 (8 kiB): 10 pJ

2x32 bit from L2 (1 MiB): 100 pJ

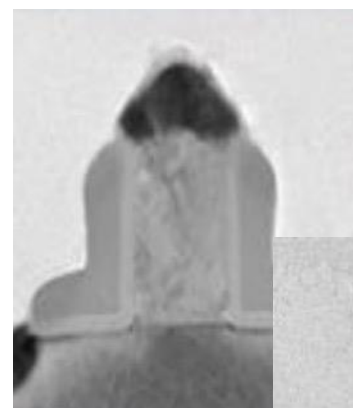
2x32 bit from DRAM: 1.3 nJ



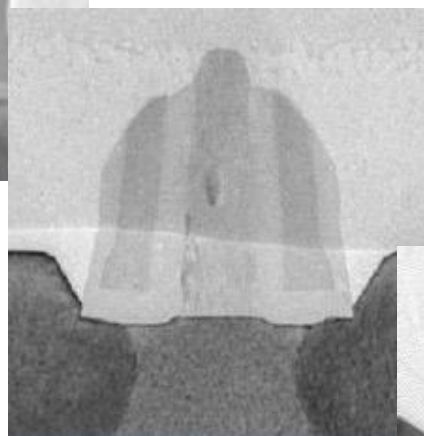
[1]: Marc Horowitz, Computing's Energy Problem (and what we can do about it), ISSC 2014, plenary

[2]: Moore: Landauer Limit Demonstrated, IEEE Spectrum 2012

Changing hardware constraints and the physics of computing



130nm



90nm



65nm



45nm



32nm

0.9 V [1]

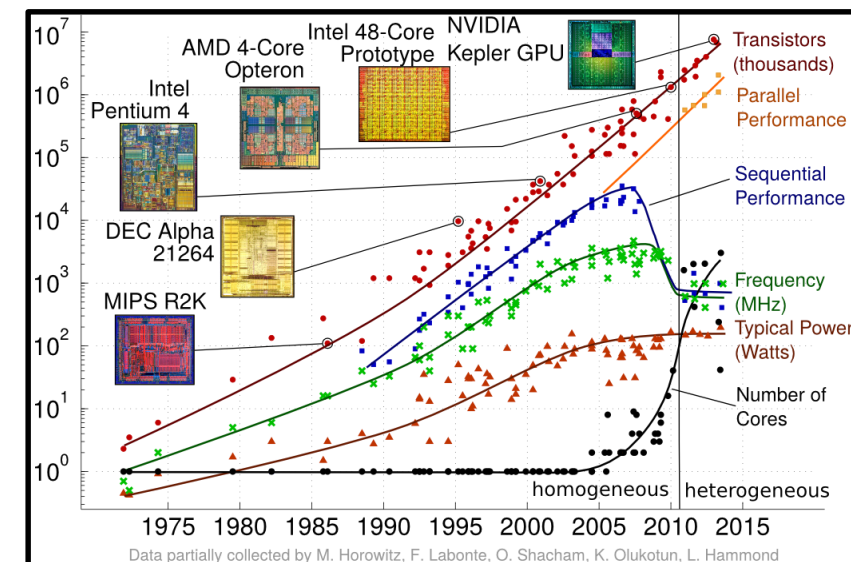
32-bit FP ADD: 0.9 pJ

32-bit FP MUL: 3.2 pJ

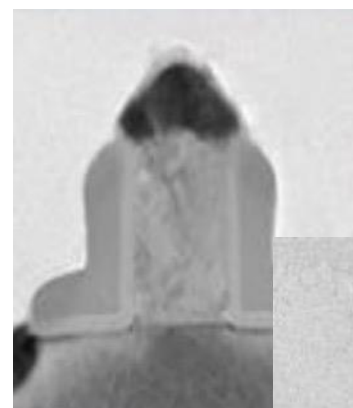
2x32 bit from L1 (8 kiB): 10 pJ

2x32 bit from L2 (1 MiB): 100 pJ

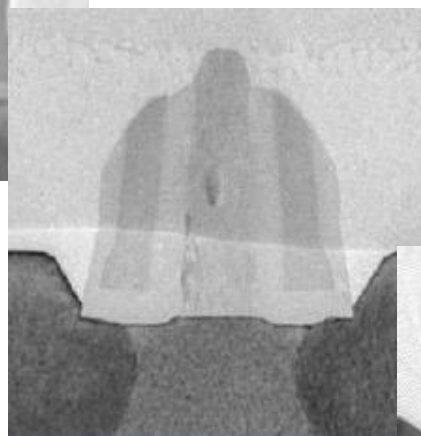
2x32 bit from DRAM: 1.3 nJ



Changing hardware constraints and the physics of computing



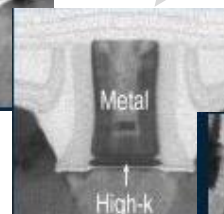
130nm



90nm



65nm



45nm



32nm



22nm

0.9 V [1]

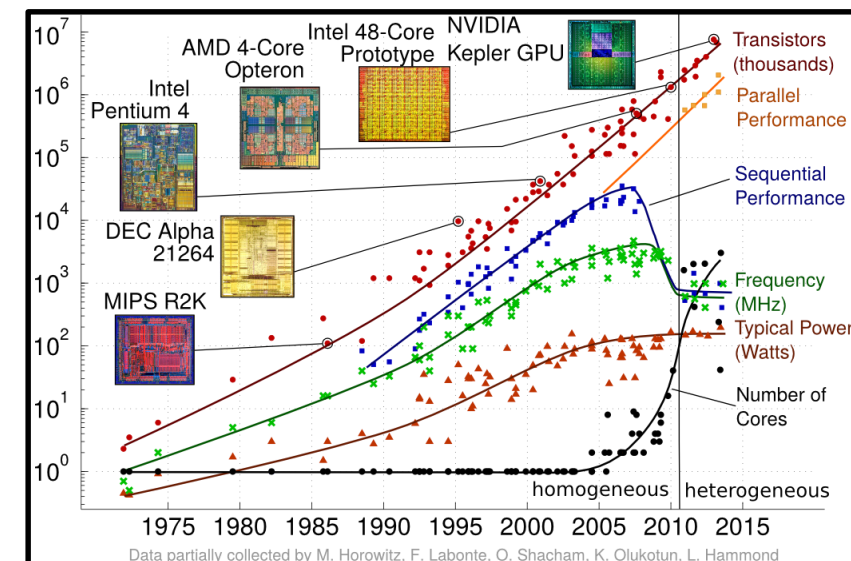
32-bit FP ADD: 0.9 pJ

32-bit FP MUL: 3.2 pJ

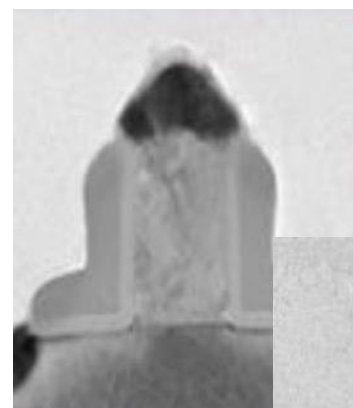
2x32 bit from L1 (8 kiB): 10 pJ

2x32 bit from L2 (1 MiB): 100 pJ

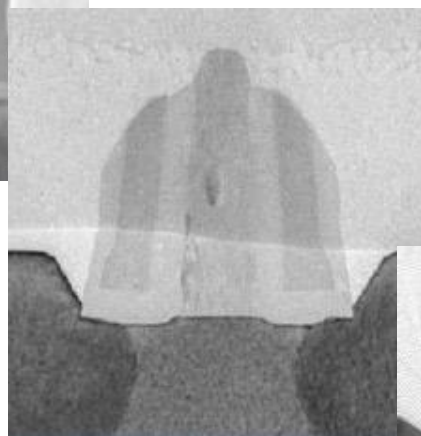
2x32 bit from DRAM: 1.3 nJ



Changing hardware constraints and the physics of computing



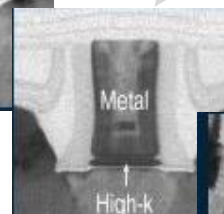
130nm



90nm



65nm



45nm



32nm



22nm



14nm

0.9 V [1]

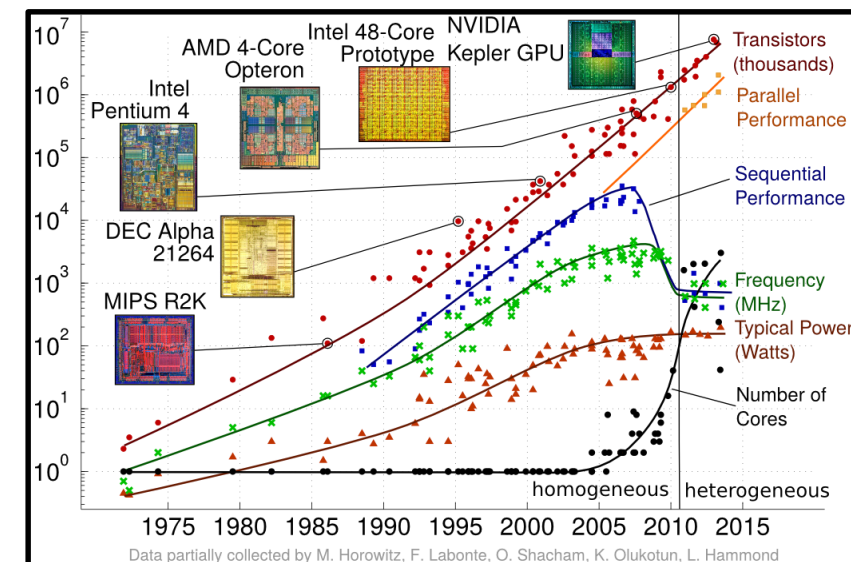
32-bit FP ADD: 0.9 pJ

32-bit FP MUL: 3.2 pJ

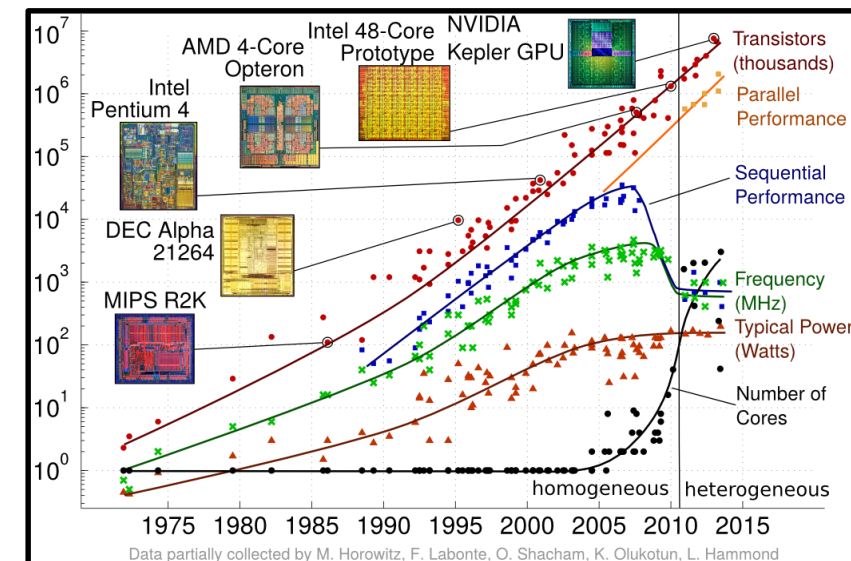
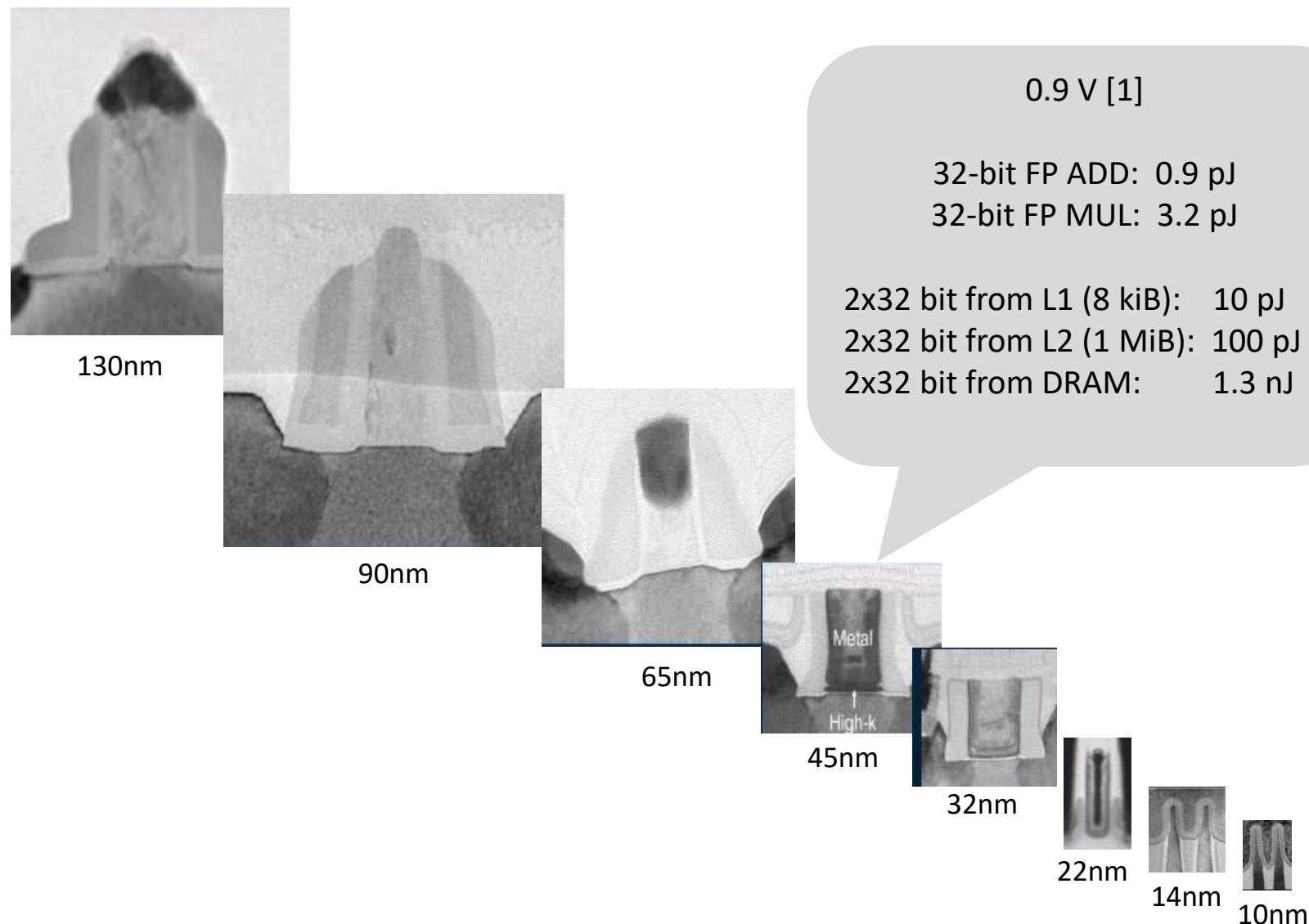
2x32 bit from L1 (8 kiB): 10 pJ

2x32 bit from L2 (1 MiB): 100 pJ

2x32 bit from DRAM: 1.3 nJ



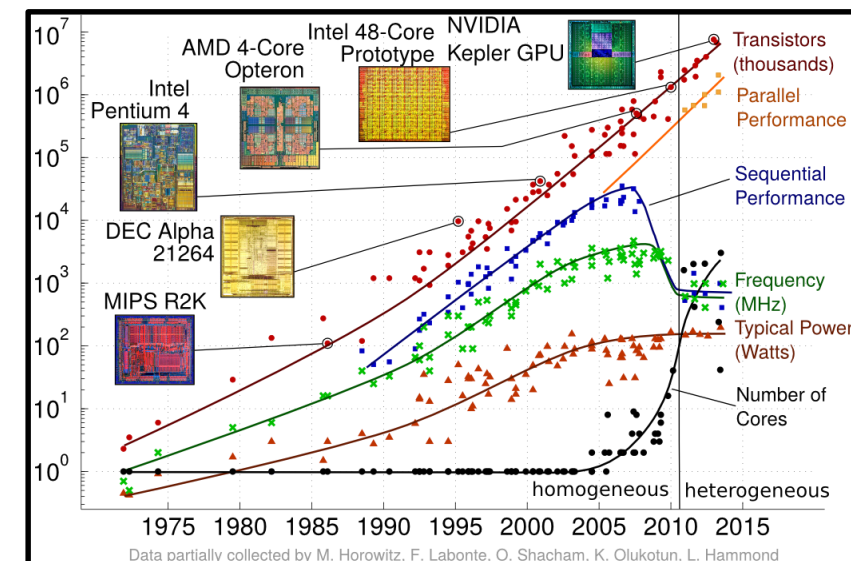
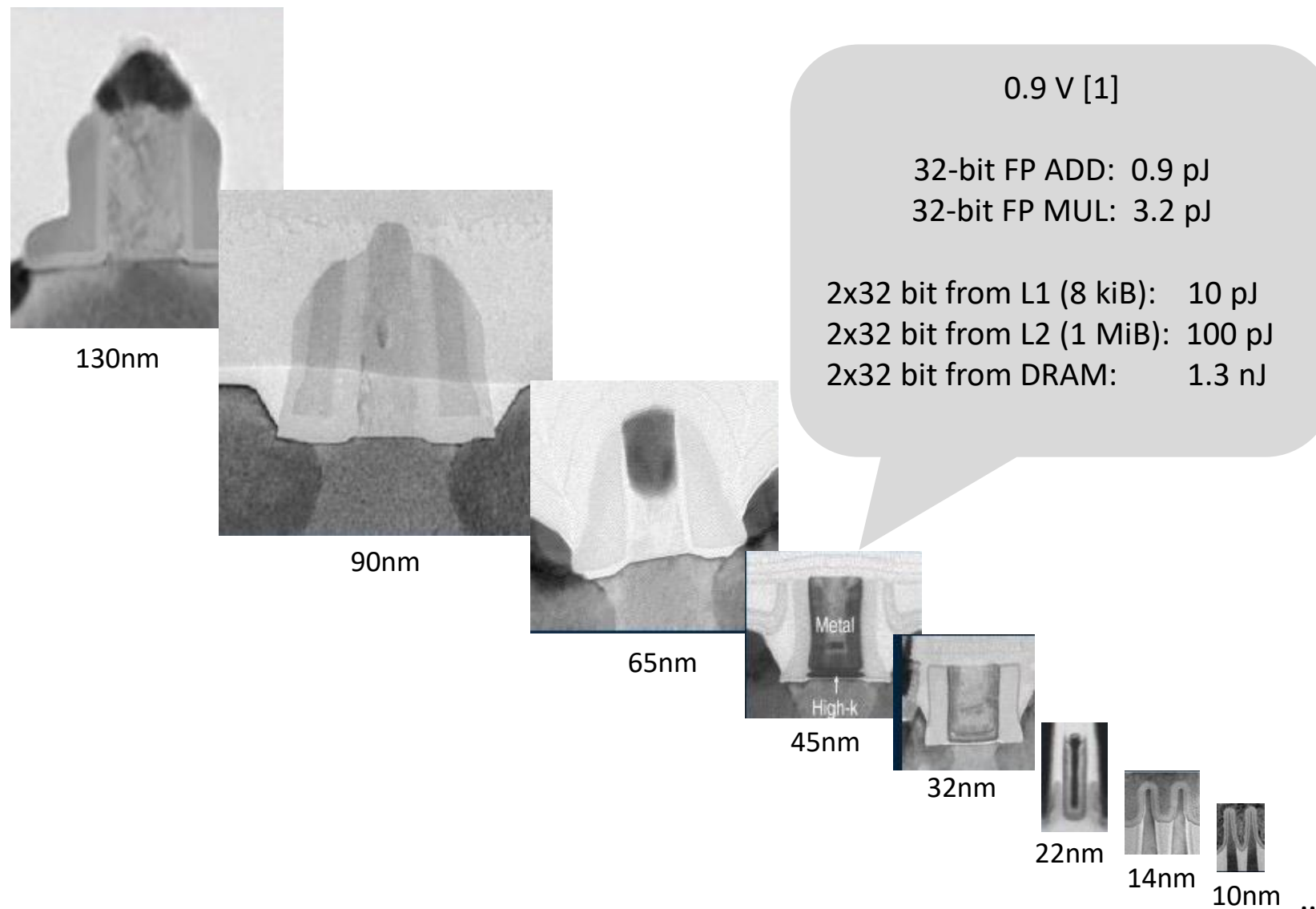
Changing hardware constraints and the physics of computing



[1]: Marc Horowitz, Computing's Energy Problem (and what we can do about it), ISSC 2014, plenary

[2]: Moore: Landauer Limit Demonstrated, IEEE Spectrum 2012

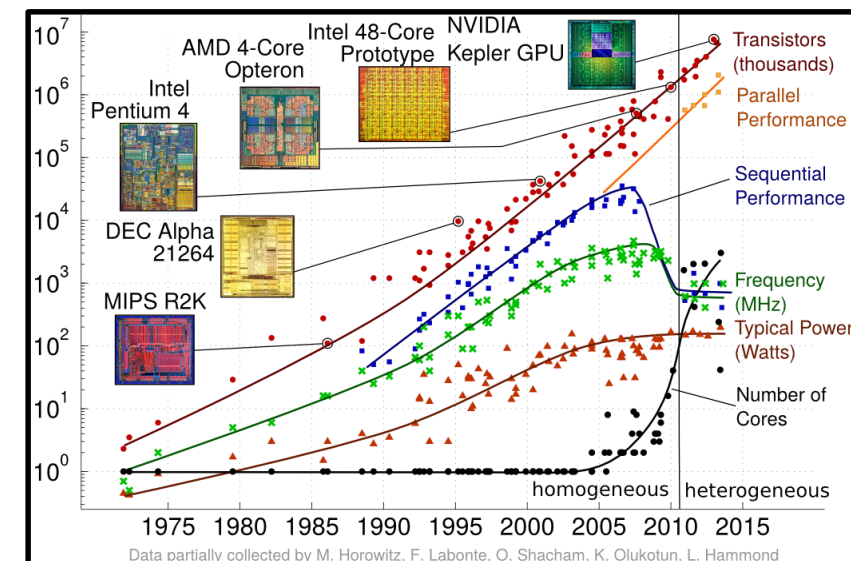
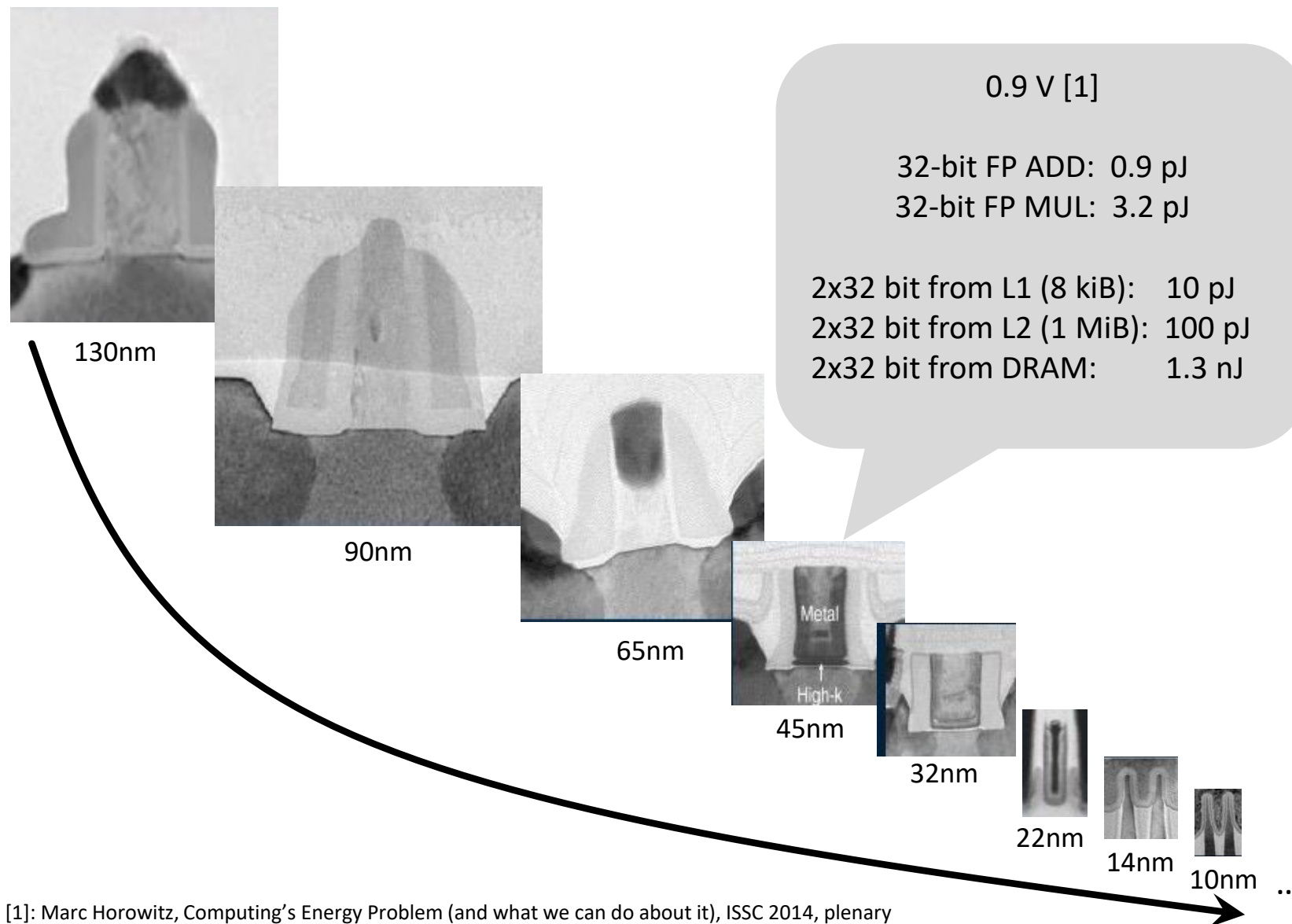
Changing hardware constraints and the physics of computing



[1]: Marc Horowitz, Computing's Energy Problem (and what we can do about it), ISSC 2014, plenary

[2]: Moore: Landauer Limit Demonstrated, IEEE Spectrum 2012

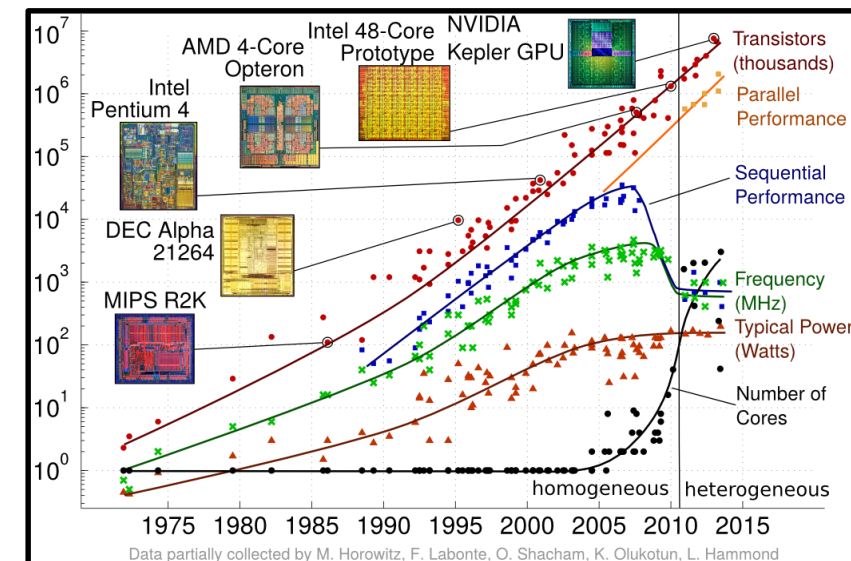
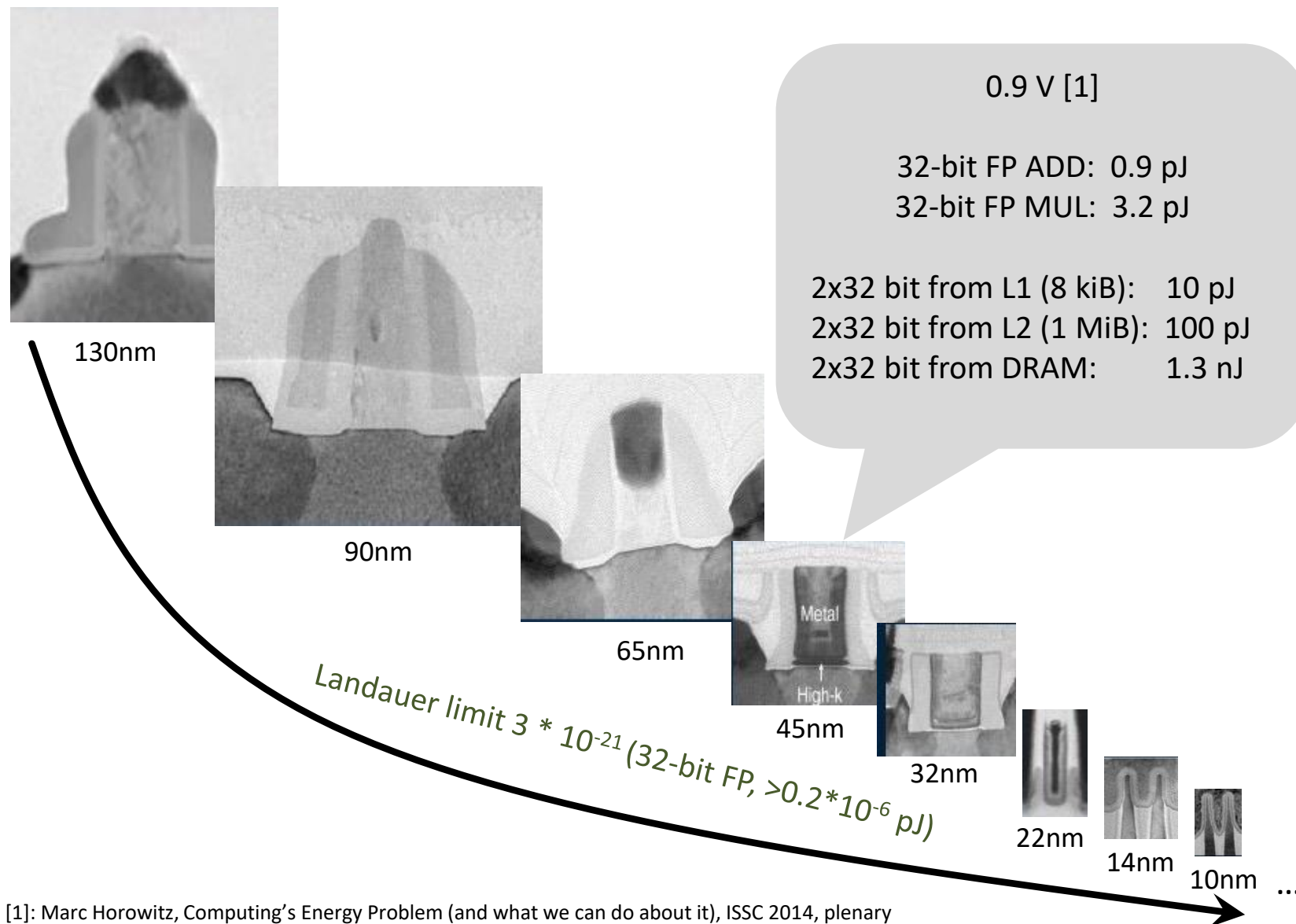
Changing hardware constraints and the physics of computing



[1]: Marc Horowitz, Computing's Energy Problem (and what we can do about it), ISSC 2014, plenary

[2]: Moore: Landauer Limit Demonstrated, IEEE Spectrum 2012

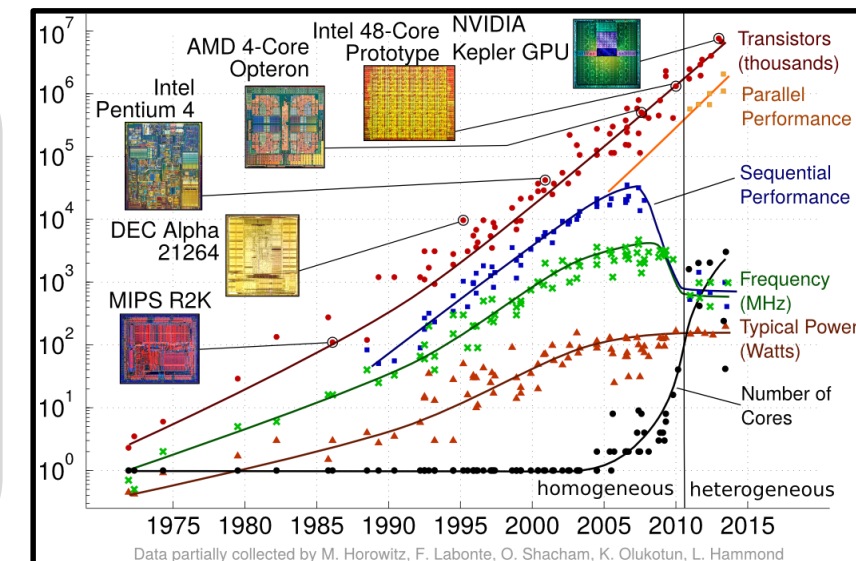
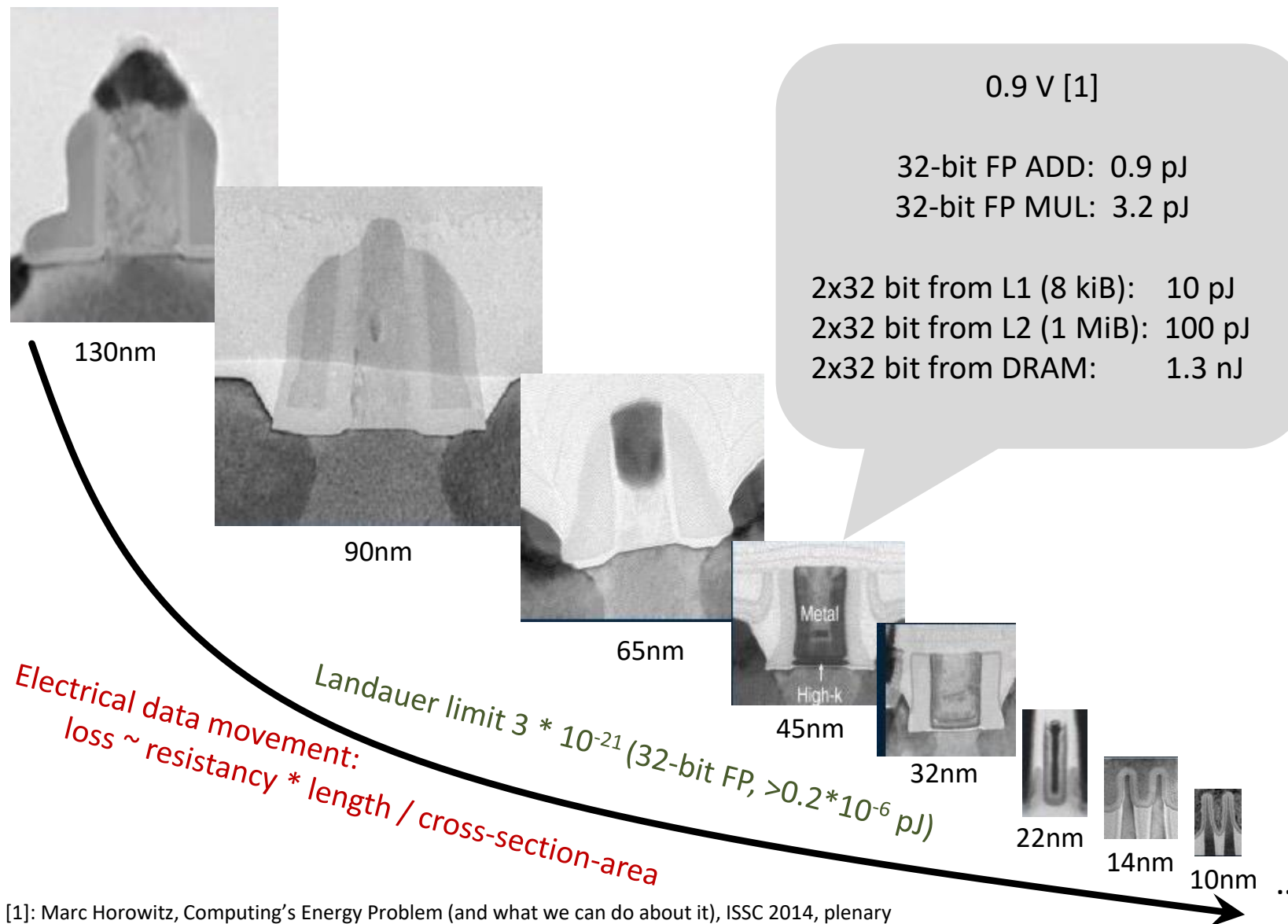
Changing hardware constraints and the physics of computing



[1]: Marc Horowitz, Computing's Energy Problem (and what we can do about it), ISSC 2014, plenary

[2]: Moore: Landauer Limit Demonstrated, IEEE Spectrum 2012

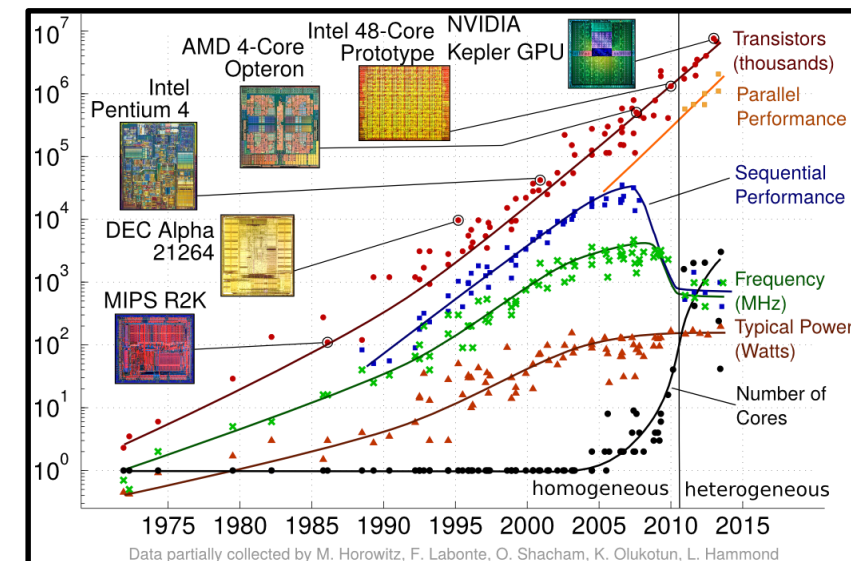
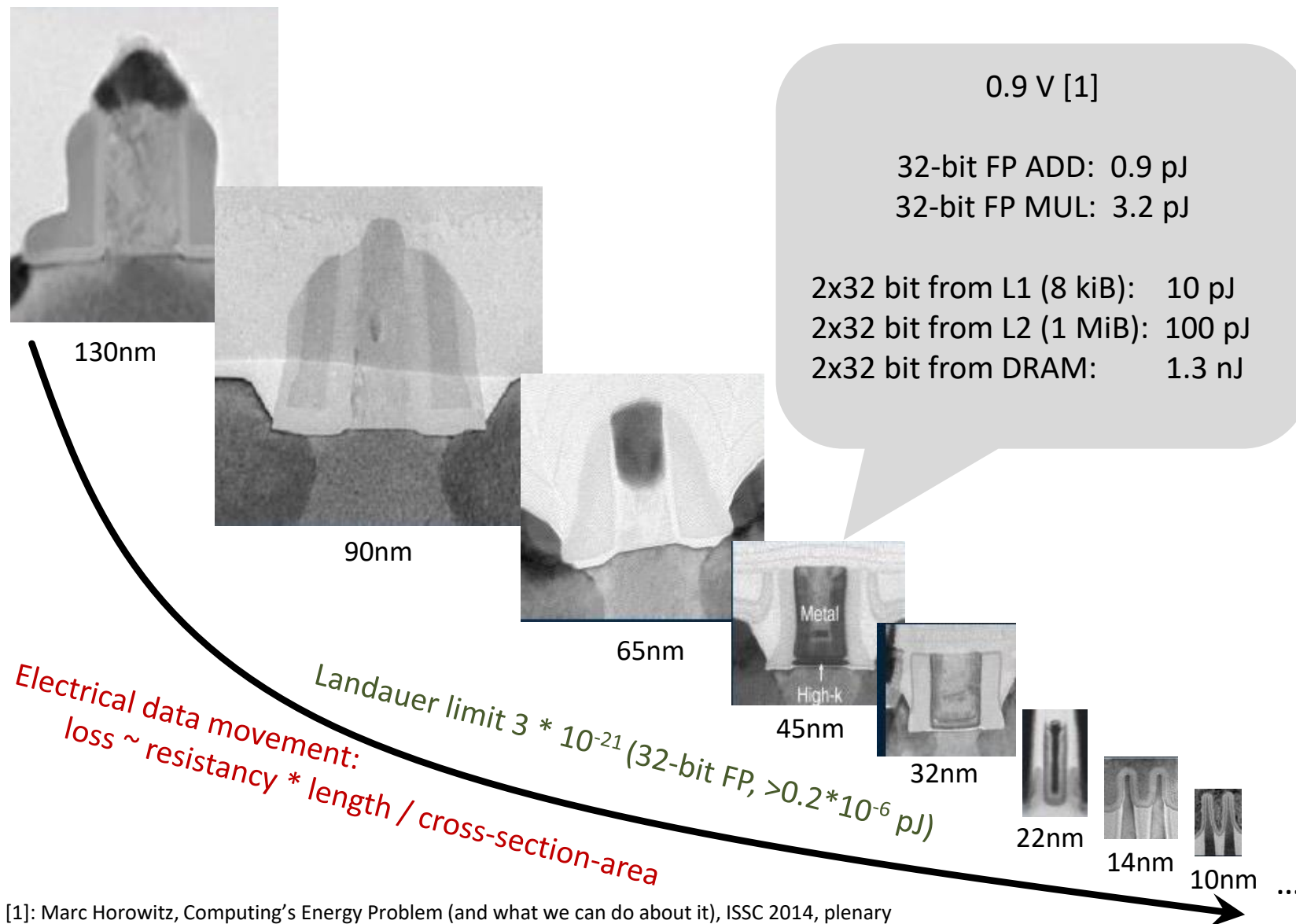
Changing hardware constraints and the physics of computing



[1]: Marc Horowitz, Computing's Energy Problem (and what we can do about it), ISSC 2014, plenary

[2]: Moore: Landauer Limit Demonstrated, IEEE Spectrum 2012

Changing hardware constraints and the physics of computing

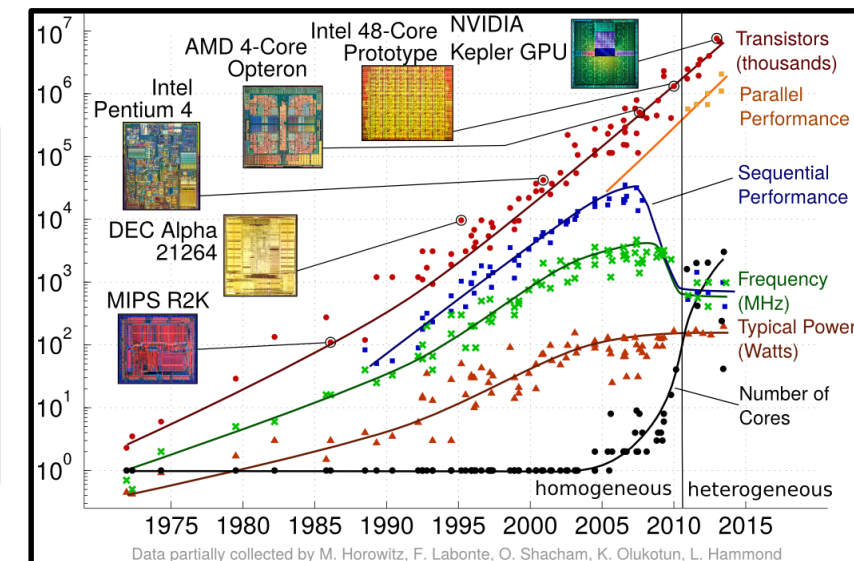
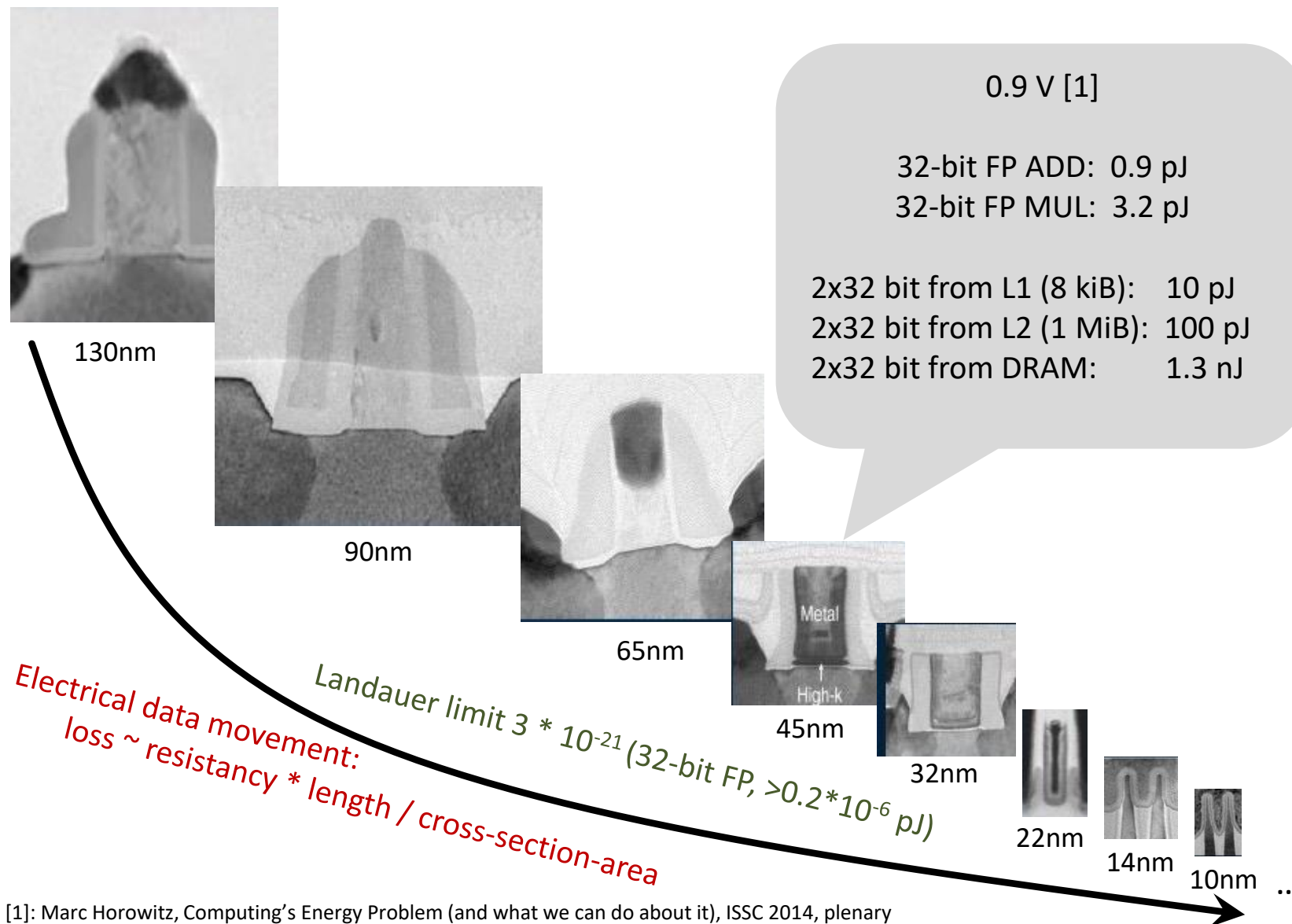


Three Ls of modern computing:

[1]: Marc Horowitz, Computing's Energy Problem (and what we can do about it), ISSC 2014, plenary

[2]: Moore: Landauer Limit Demonstrated, IEEE Spectrum 2012

Changing hardware constraints and the physics of computing



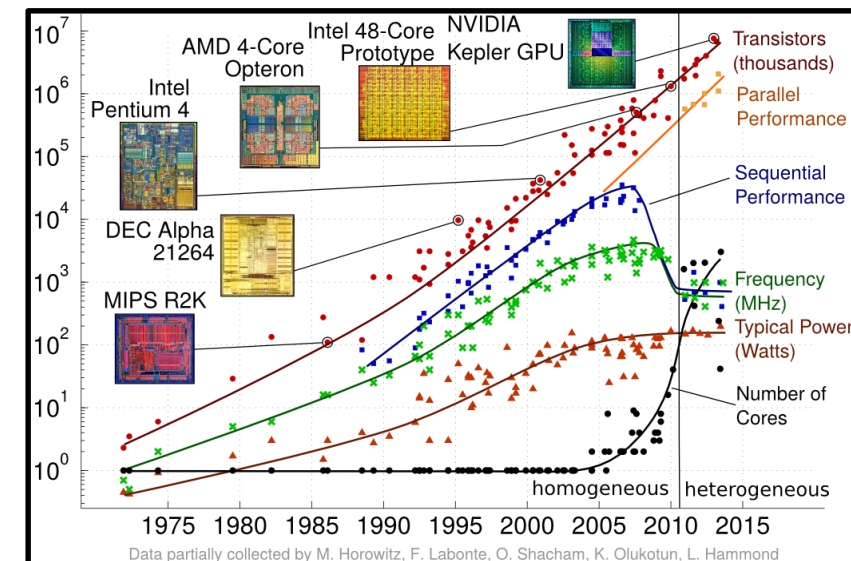
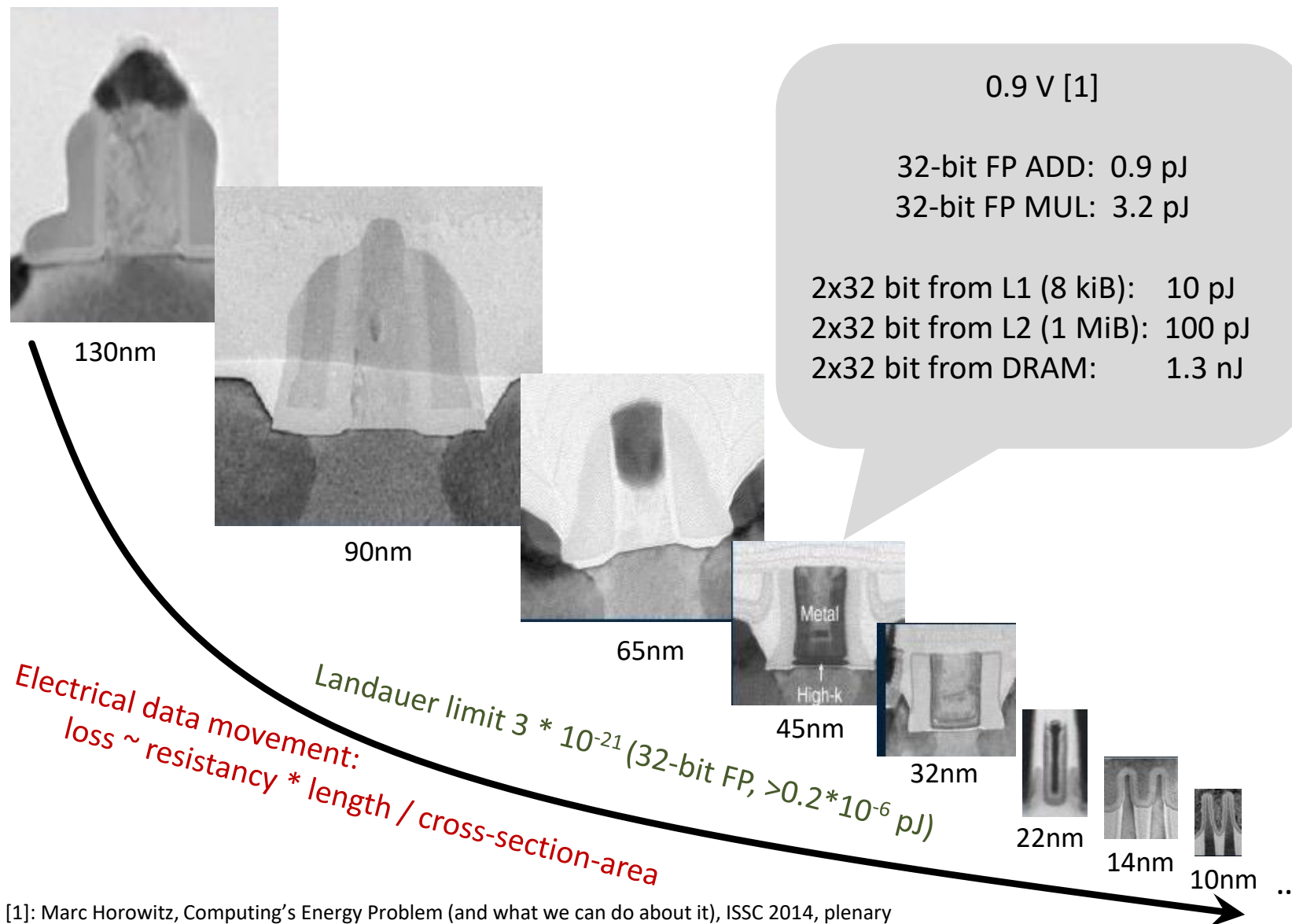
Three Ls of modern computing:

Spatial Locality

[1]: Marc Horowitz, Computing's Energy Problem (and what we can do about it), ISSC 2014, plenary

[2]: Moore: Landauer Limit Demonstrated, IEEE Spectrum 2012

Changing hardware constraints and the physics of computing



Three Ls of modern computing:

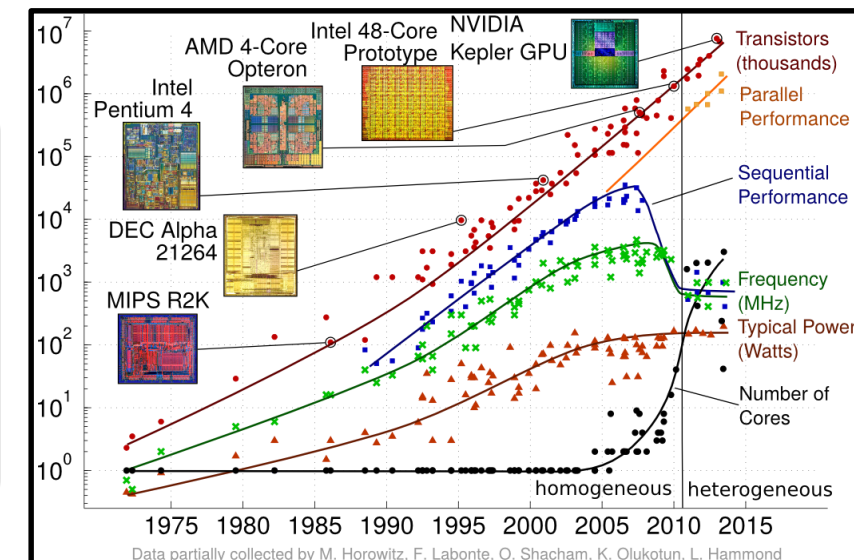
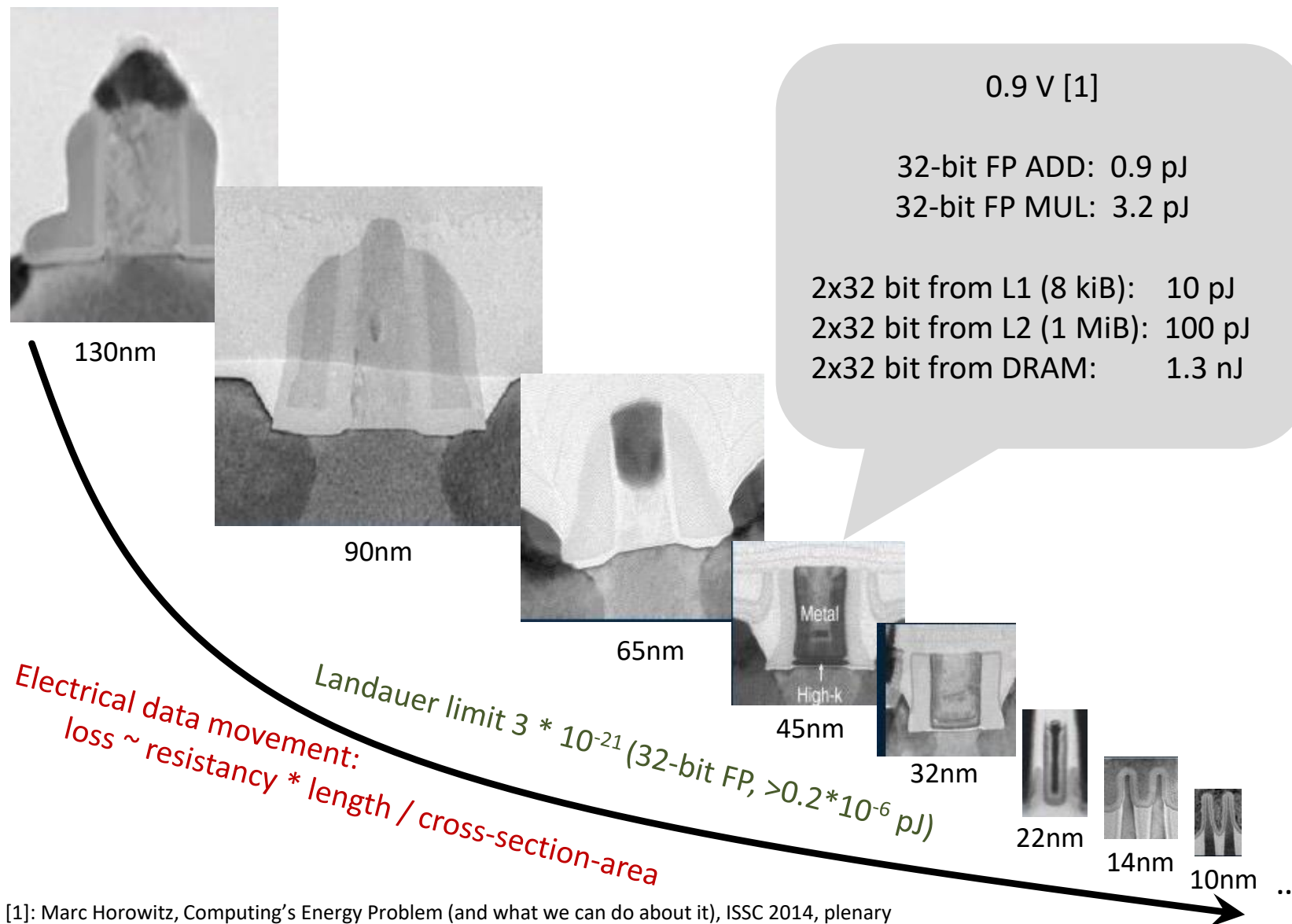
Spatial Locality

Temporal Locality

[1]: Marc Horowitz, Computing's Energy Problem (and what we can do about it), ISSC 2014, plenary

[2]: Moore: Landauer Limit Demonstrated, IEEE Spectrum 2012

Changing hardware constraints and the physics of computing



Three Ls of modern computing:

Spatial Locality

Temporal Locality

Control Locality

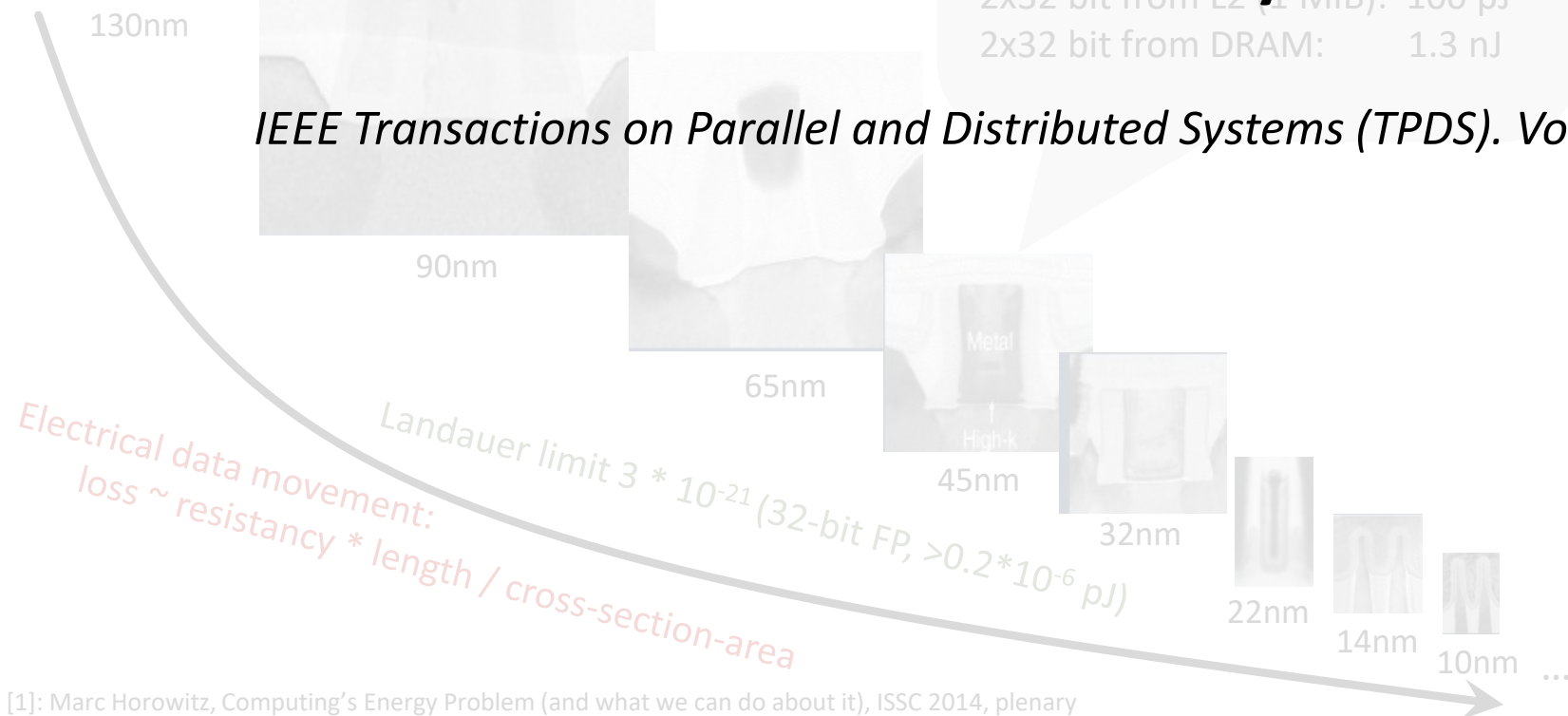
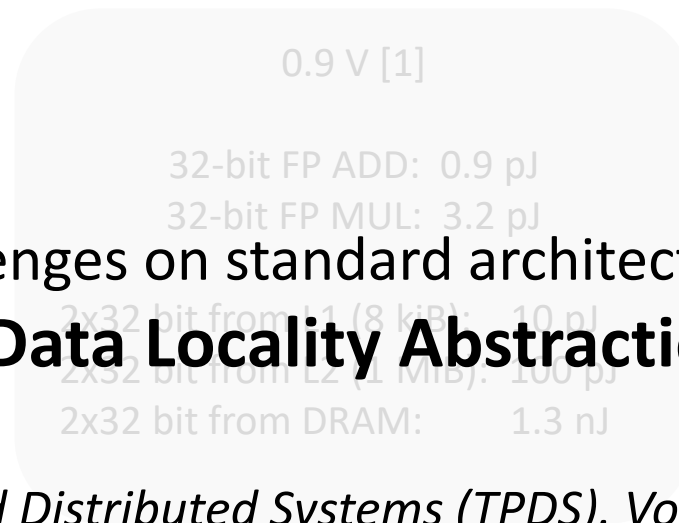
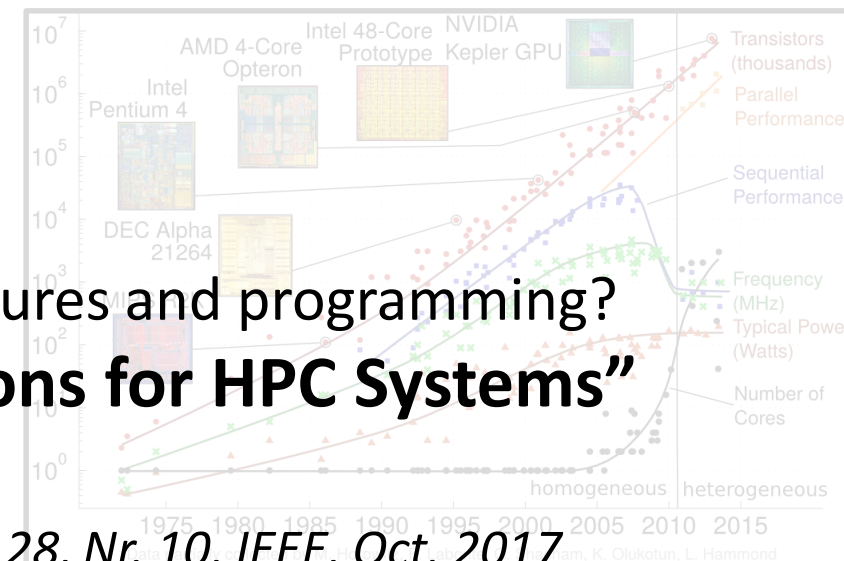
[1]: Marc Horowitz, Computing's Energy Problem (and what we can do about it), ISSC 2014, plenary

[2]: Moore: Landauer Limit Demonstrated, IEEE Spectrum 2012

Changing hardware constraints and the physics of computing

How to address locality challenges on standard architectures and programming?
D. Unat et al.: **“Trends in Data Locality Abstractions for HPC Systems”**

IEEE Transactions on Parallel and Distributed Systems (TPDS). Vol 28, Nr. 10, IEEE, Oct. 2017



Three Ls of modern computing:
Spatial Locality
Temporal Locality
Control Locality

[1]: Marc Horowitz, Computing's Energy Problem (and what we can do about it), ISSC 2014, plenary

[2]: Moore: Landauer Limit Demonstrated, IEEE Spectrum 2012

Load-store vs. Dataflow architectures

Load-store (“von Neumann”)

$x = a + b$



Static Dataflow (“non von Neumann”)

$y = (a + b) * (c + d)$

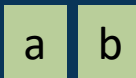
Load-store vs. Dataflow architectures

Load-store (“von Neumann”)

$$x=a+b$$



Memory



Static Dataflow (“non von Neumann”)

$$y=(a+b)*(c+d)$$

Load-store vs. Dataflow architectures

Load-store (“von Neumann”)

$x = a + b$



ALU

Memory

a b

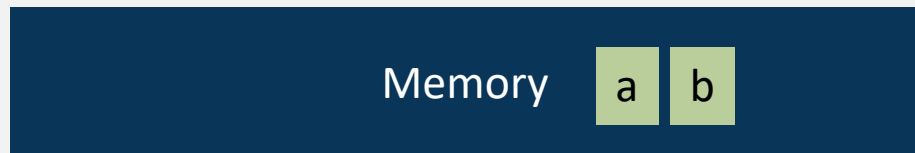
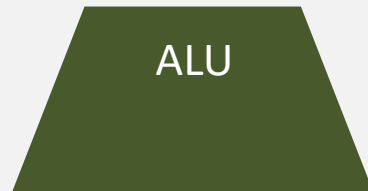
Static Dataflow (“non von Neumann”)

$y = (a + b) * (c + d)$

Load-store vs. Dataflow architectures

Load-store (“von Neumann”)

$$x=a+b$$



Static Dataflow (“non von Neumann”)

$$y=(a+b)*(c+d)$$

Load-store vs. Dataflow architectures

Load-store (“von Neumann”)

$$x = a + b$$



ALU

Control

Registers

Memory

a

b

Static Dataflow (“non von Neumann”)

$$y = (a + b) * (c + d)$$

Load-store vs. Dataflow architectures

Load-store (“von Neumann”)

$$x=a+b$$



ALU

Control

Registers

Cache

Memory

a

b

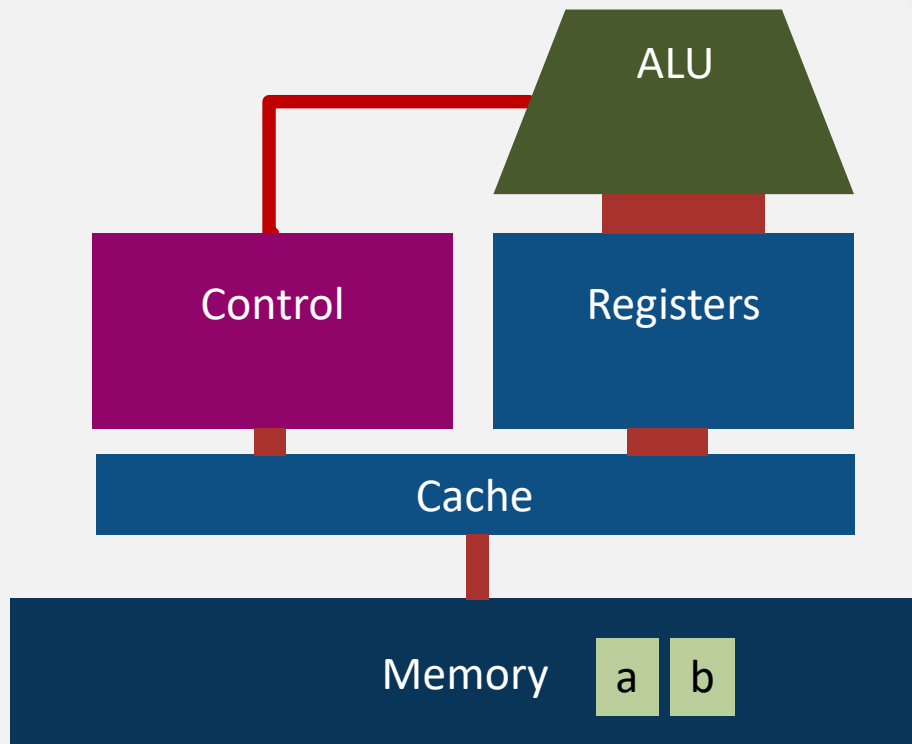
Static Dataflow (“non von Neumann”)

$$y=(a+b)*(c+d)$$

Load-store vs. Dataflow architectures

Load-store (“von Neumann”)

$$x = a + b$$



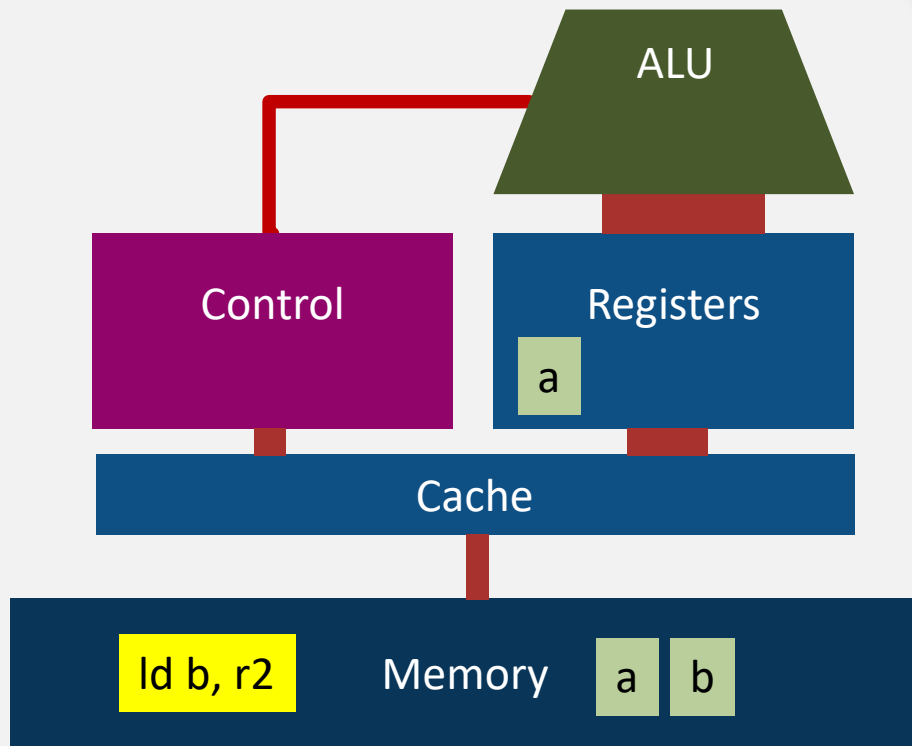
Static Dataflow (“non von Neumann”)

$$y = (a + b) * (c + d)$$

Load-store vs. Dataflow architectures

Load-store (“von Neumann”)

$$x = a + b$$



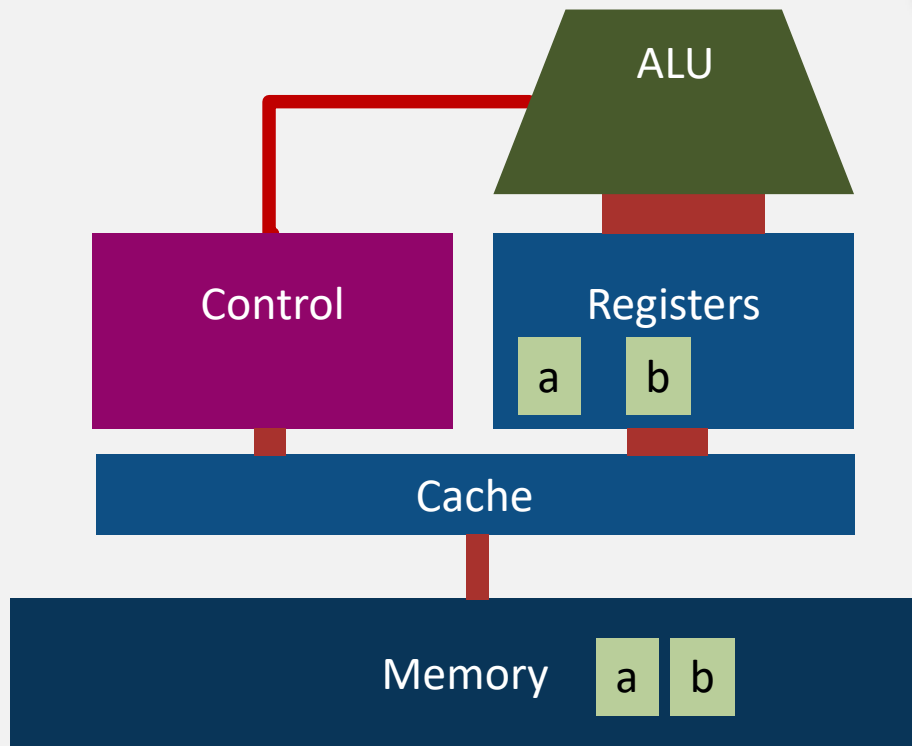
Static Dataflow (“non von Neumann”)

$$y = (a + b) * (c + d)$$

Load-store vs. Dataflow architectures

Load-store (“von Neumann”)

$$x = a + b$$



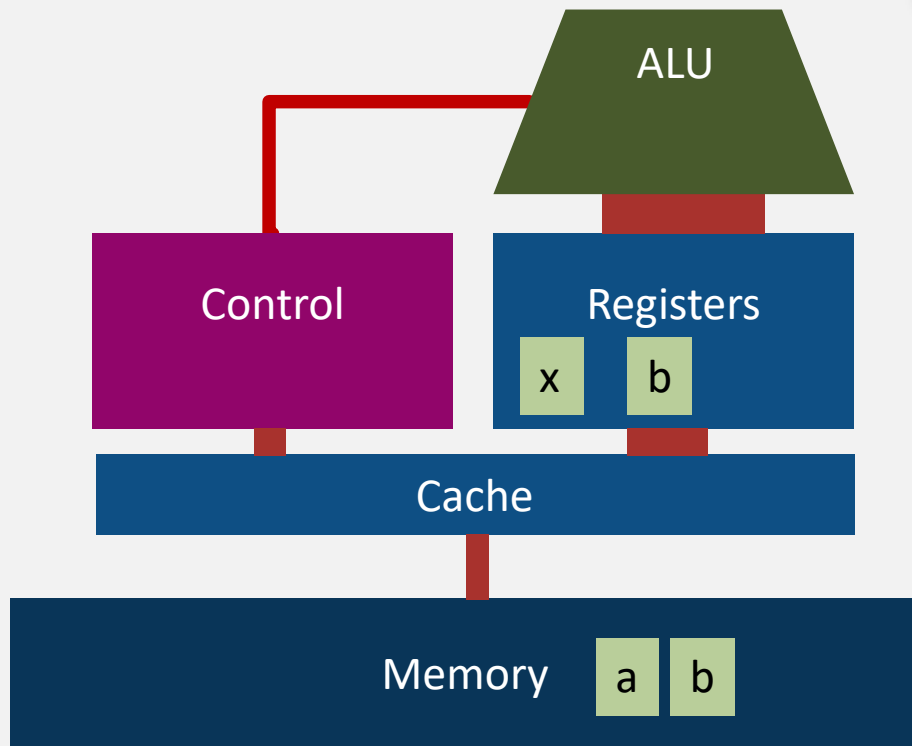
Static Dataflow (“non von Neumann”)

$$y = (a + b) * (c + d)$$

Load-store vs. Dataflow architectures

Load-store (“von Neumann”)

$$x = a + b$$



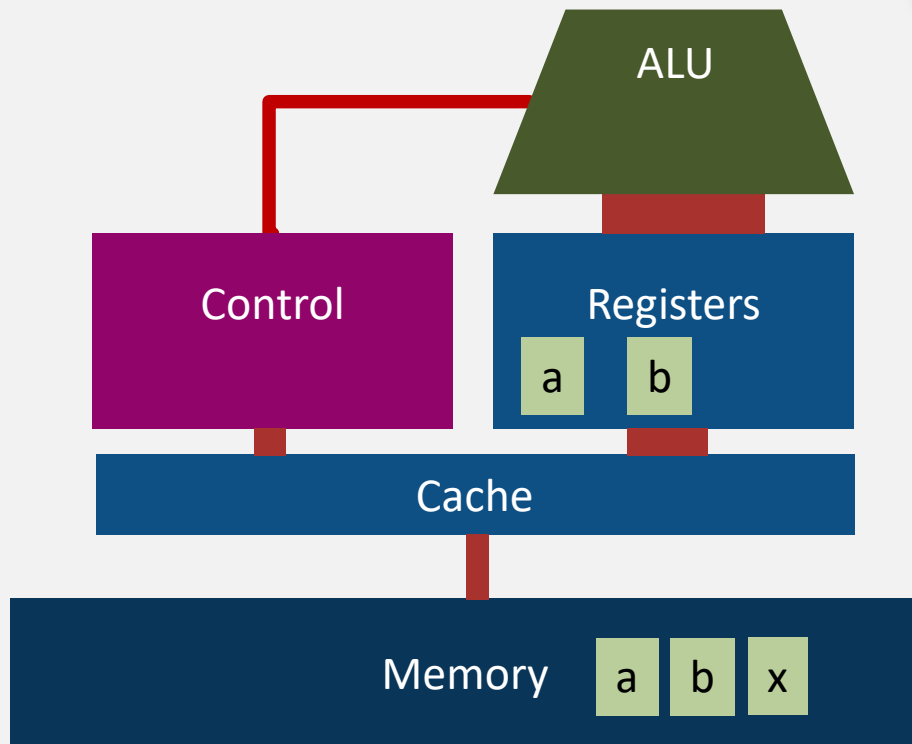
Static Dataflow (“non von Neumann”)

$$y = (a + b) * (c + d)$$

Load-store vs. Dataflow architectures

Load-store (“von Neumann”)

$$x=a+b$$



Static Dataflow (“non von Neumann”)

$$y=(a+b)*(c+d)$$

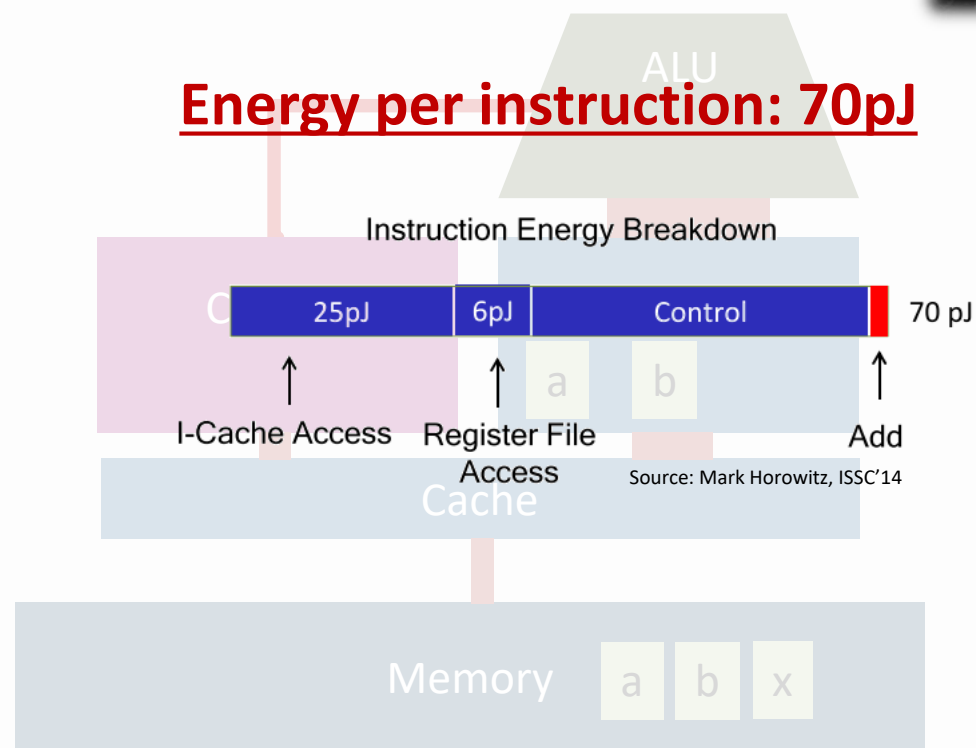
Load-store vs. Dataflow architectures

Load-store (“von Neumann”)

$$x=a+b$$



Energy per instruction: 70pJ



Static Dataflow (“non von Neumann”)

$$y=(a+b)*(c+d)$$

Load-store vs. Dataflow architectures

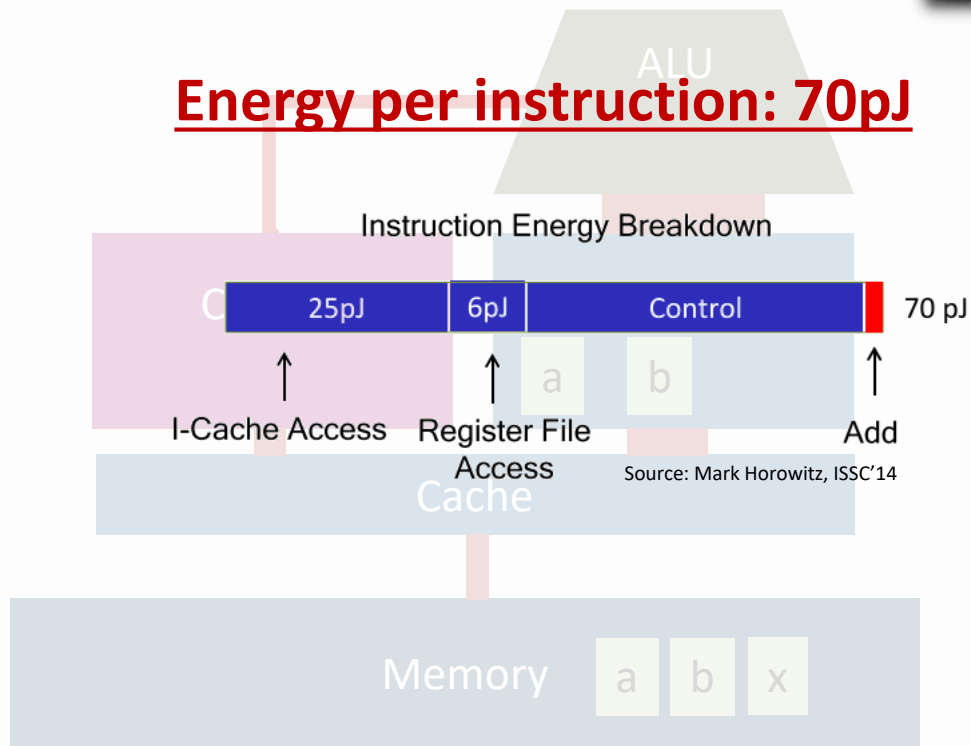
Turing Award 1977 (Backus): "Surely there must be a less primitive way of making big changes in the store than pushing vast numbers of words back and forth through the von Neumann bottleneck."

Load-store ("von Neumann")



$$x = a + b$$

Energy per instruction: 70pJ



Static Dataflow ("non von Neumann")



$$y = (a + b) * (c + d)$$

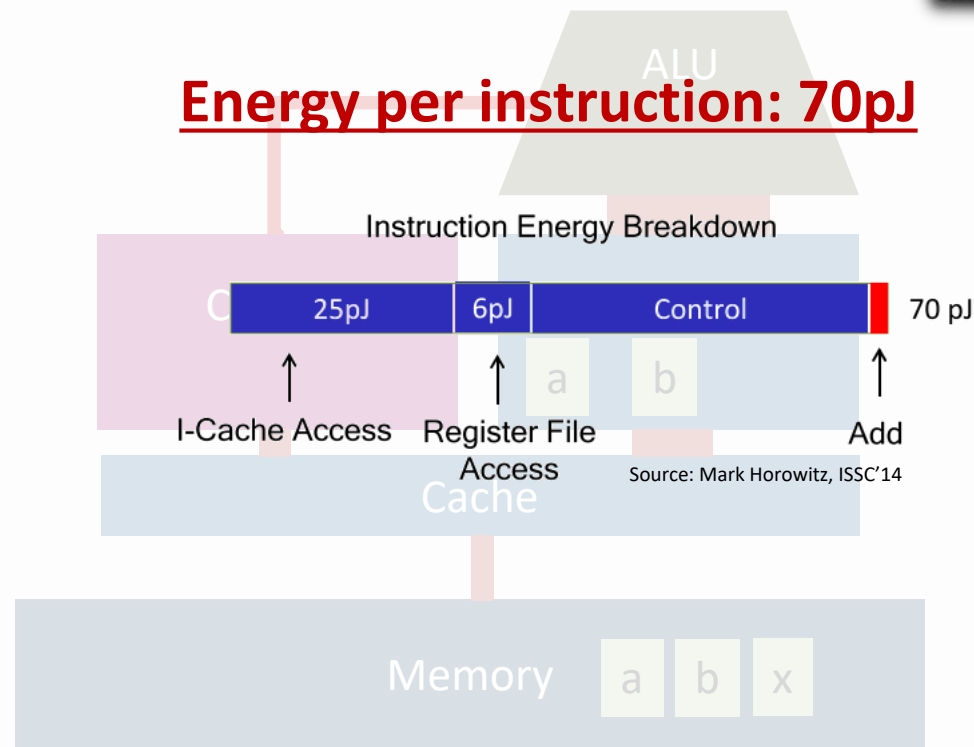
Load-store vs. Dataflow architectures

Load-store (“von Neumann”)

$$x=a+b$$



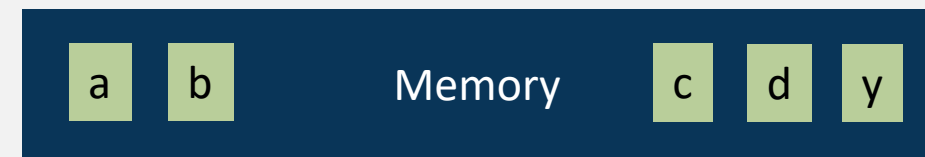
Energy per instruction: 70pJ



Turing Award 1977 (Backus): "Surely there must be a less primitive way of making big changes in the store than pushing vast numbers of words back and forth through the von Neumann bottleneck."

Static Dataflow (“non von Neumann”)

$$y=(a+b)*(c+d)$$



Load-store vs. Dataflow architectures

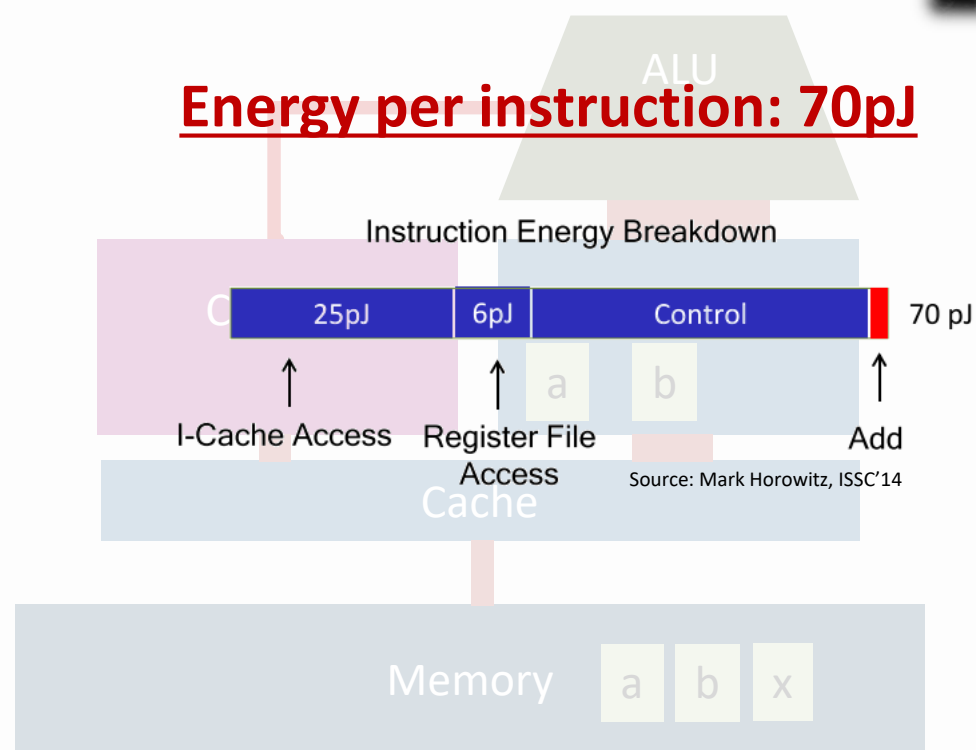
Turing Award 1977 (Backus): "Surely there must be a less primitive way of making big changes in the store than pushing vast numbers of words back and forth through the von Neumann bottleneck."

Load-store ("von Neumann")



$$x = a + b$$

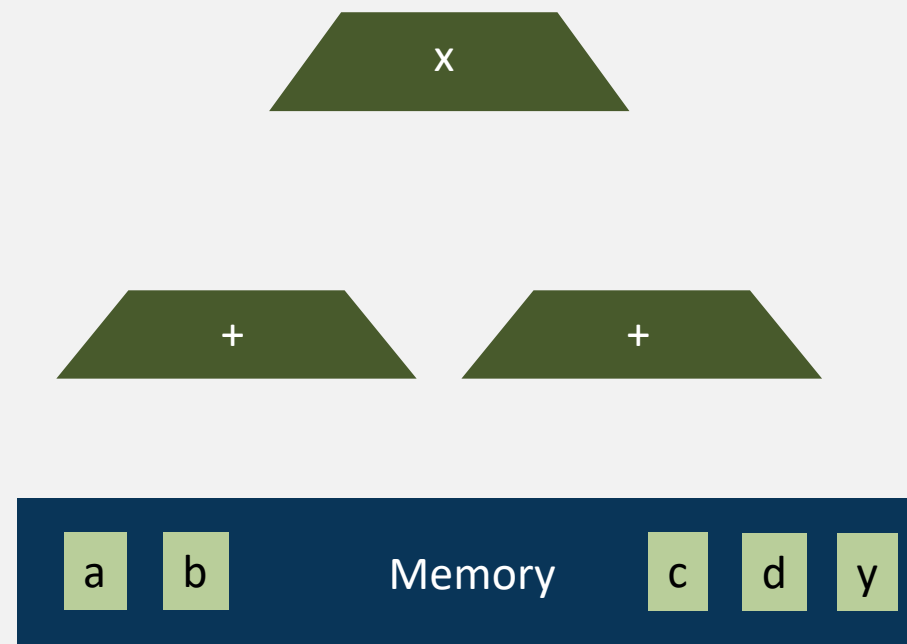
Energy per instruction: 70pJ



Static Dataflow ("non von Neumann")



$$y = (a + b) * (c + d)$$



Load-store vs. Dataflow architectures

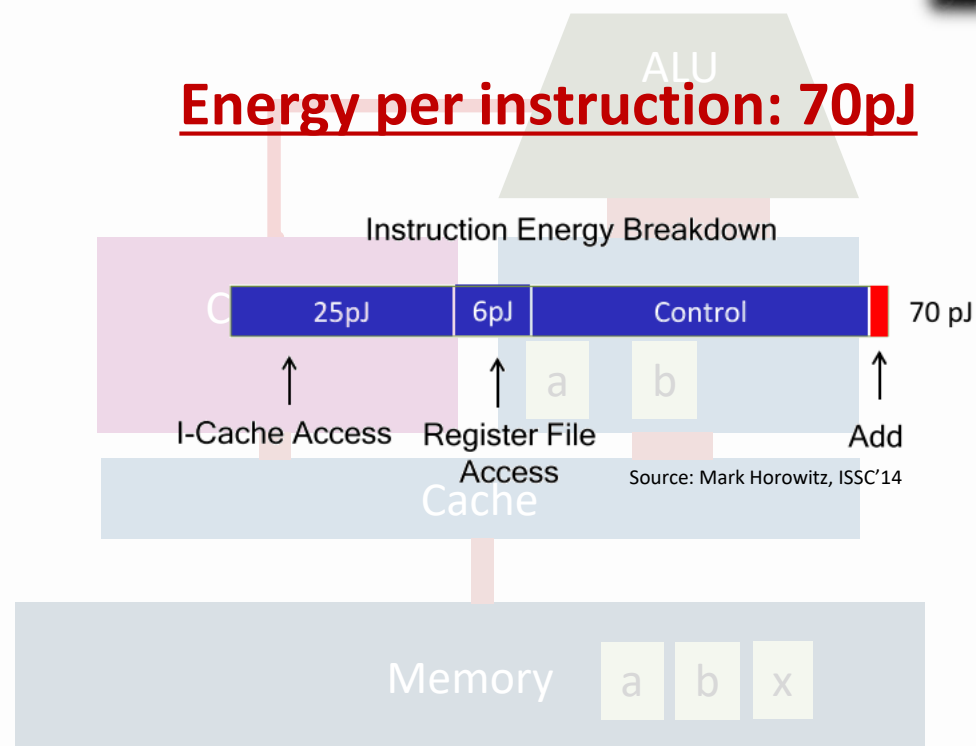
Turing Award 1977 (Backus): "Surely there must be a less primitive way of making big changes in the store than pushing vast numbers of words back and forth through the von Neumann bottleneck."

Load-store ("von Neumann")



$$x = a + b$$

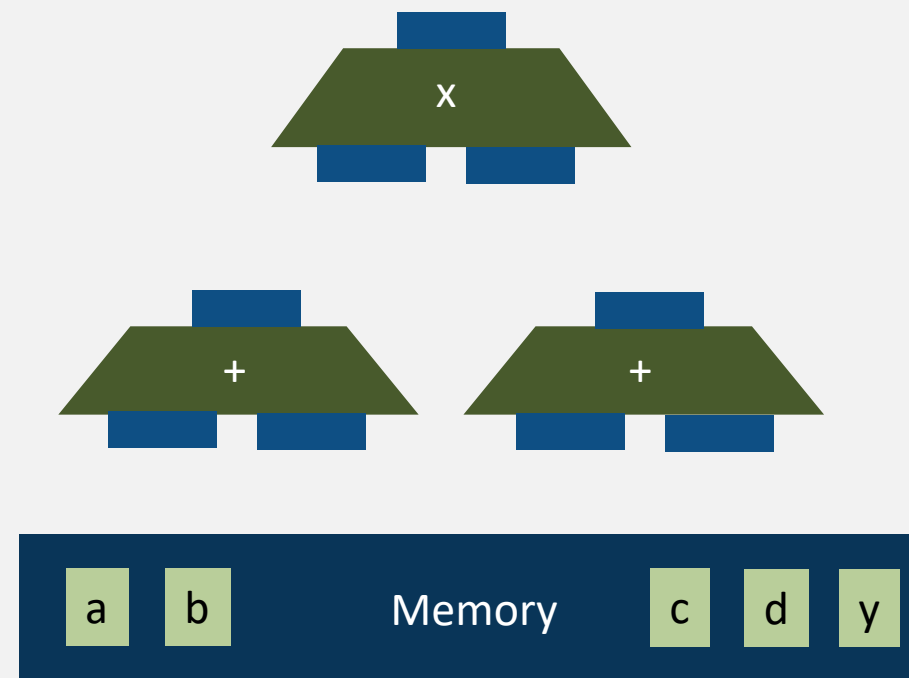
Energy per instruction: 70pJ



Static Dataflow ("non von Neumann")



$$y = (a + b) * (c + d)$$



Load-store vs. Dataflow architectures

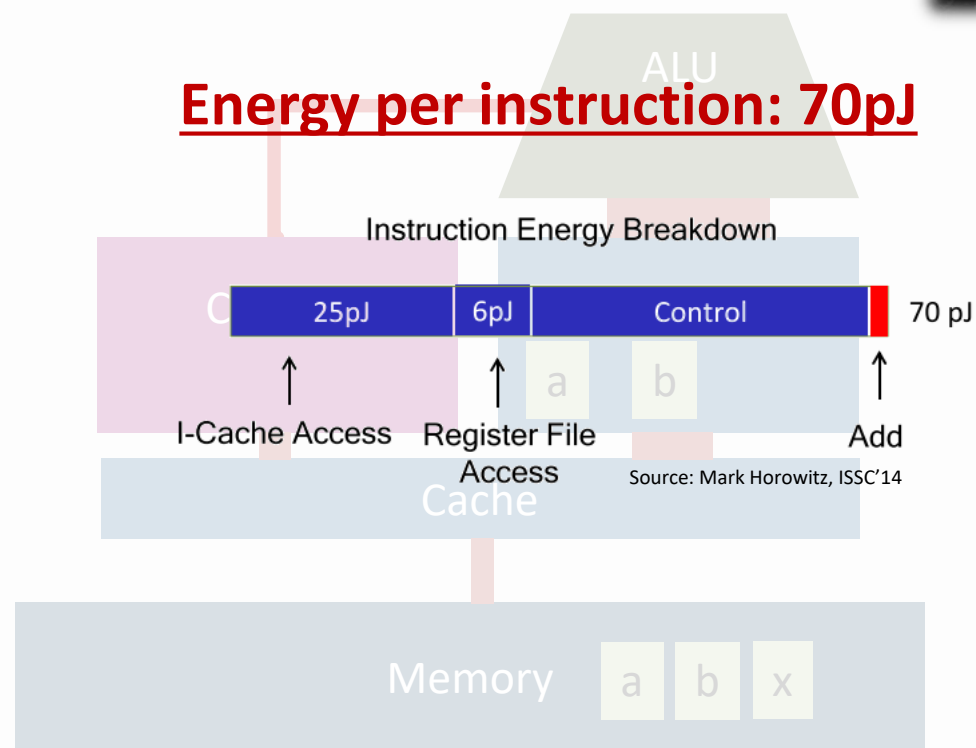
Turing Award 1977 (Backus): "Surely there must be a less primitive way of making big changes in the store than pushing vast numbers of words back and forth through the von Neumann bottleneck."

Load-store ("von Neumann")



$$x = a + b$$

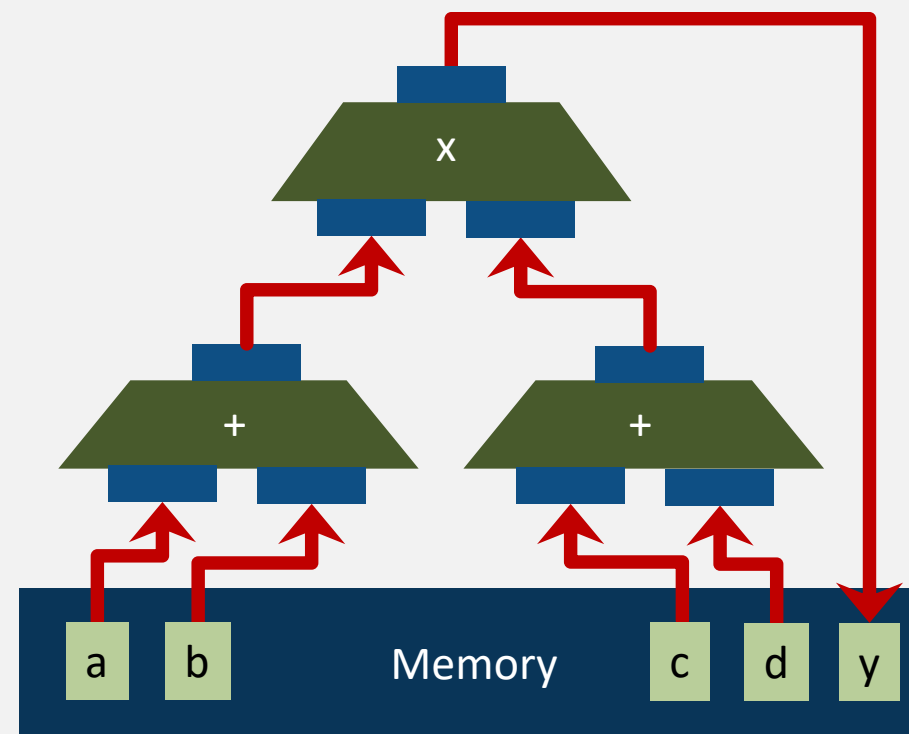
Energy per instruction: 70pJ



Static Dataflow ("non von Neumann")



$$y = (a + b) * (c + d)$$



Load-store vs. Dataflow architectures

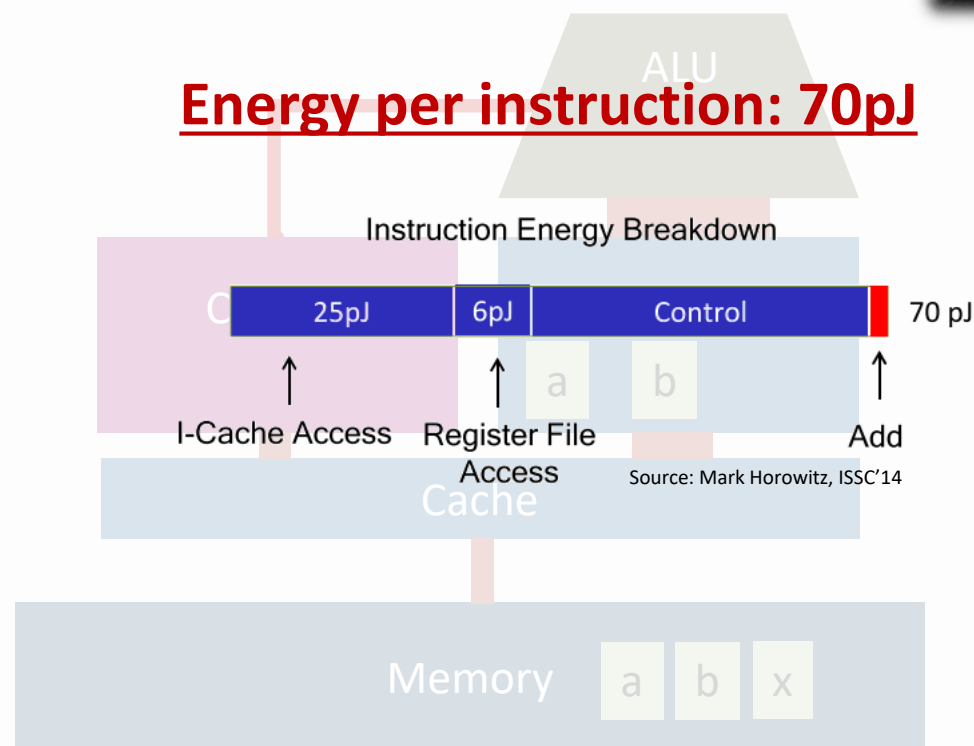
Turing Award 1977 (Backus): "Surely there must be a less primitive way of making big changes in the store than pushing vast numbers of words back and forth through the von Neumann bottleneck."

Load-store ("von Neumann")



$$x = a + b$$

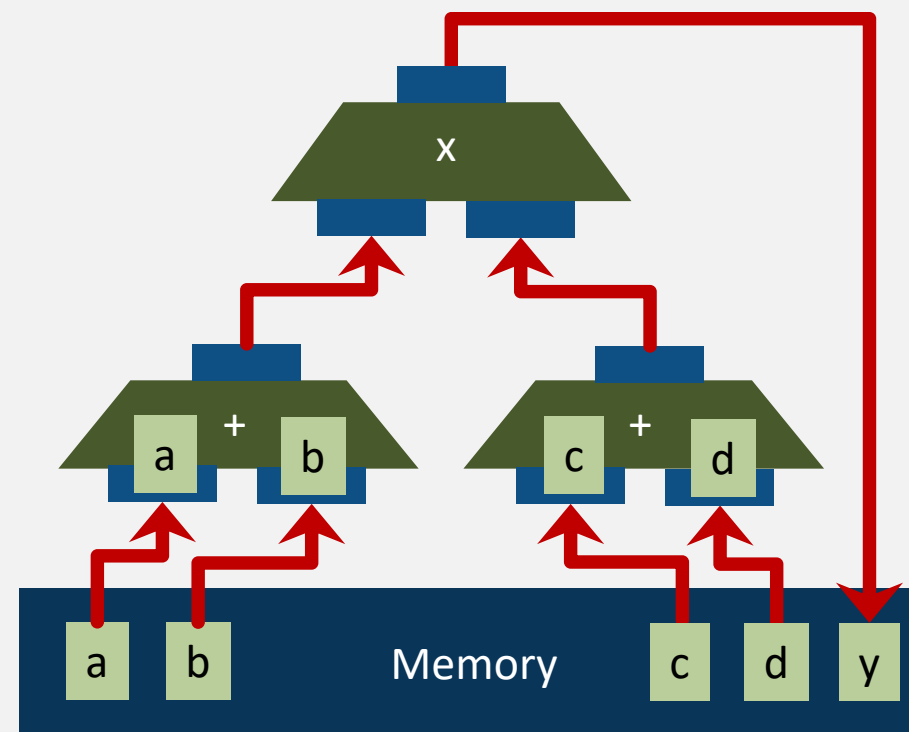
Energy per instruction: 70pJ



Static Dataflow ("non von Neumann")



$$y = (a + b) * (c + d)$$



Load-store vs. Dataflow architectures

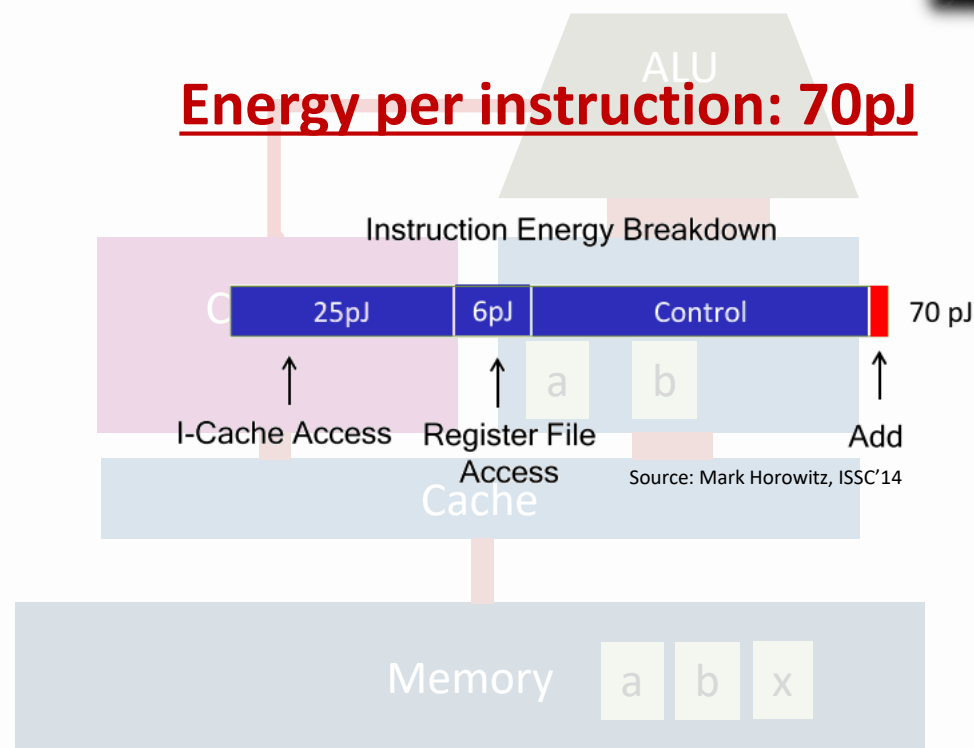
Turing Award 1977 (Backus): "Surely there must be a less primitive way of making big changes in the store than pushing vast numbers of words back and forth through the von Neumann bottleneck."

Load-store ("von Neumann")



$$x = a + b$$

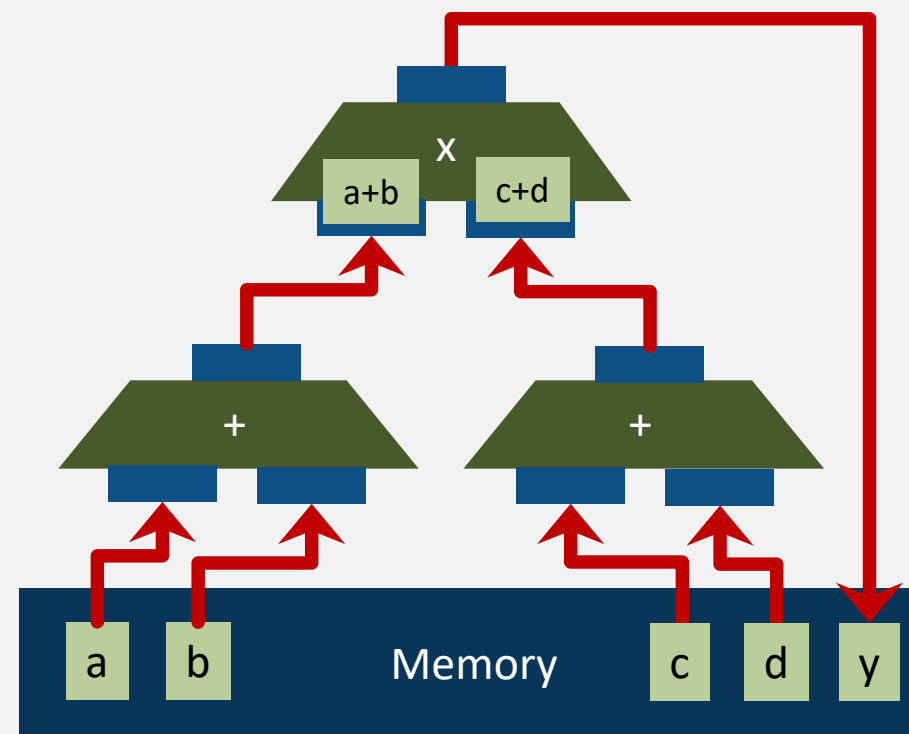
Energy per instruction: 70pJ



Static Dataflow ("non von Neumann")



$$y = (a + b) * (c + d)$$



Load-store vs. Dataflow architectures

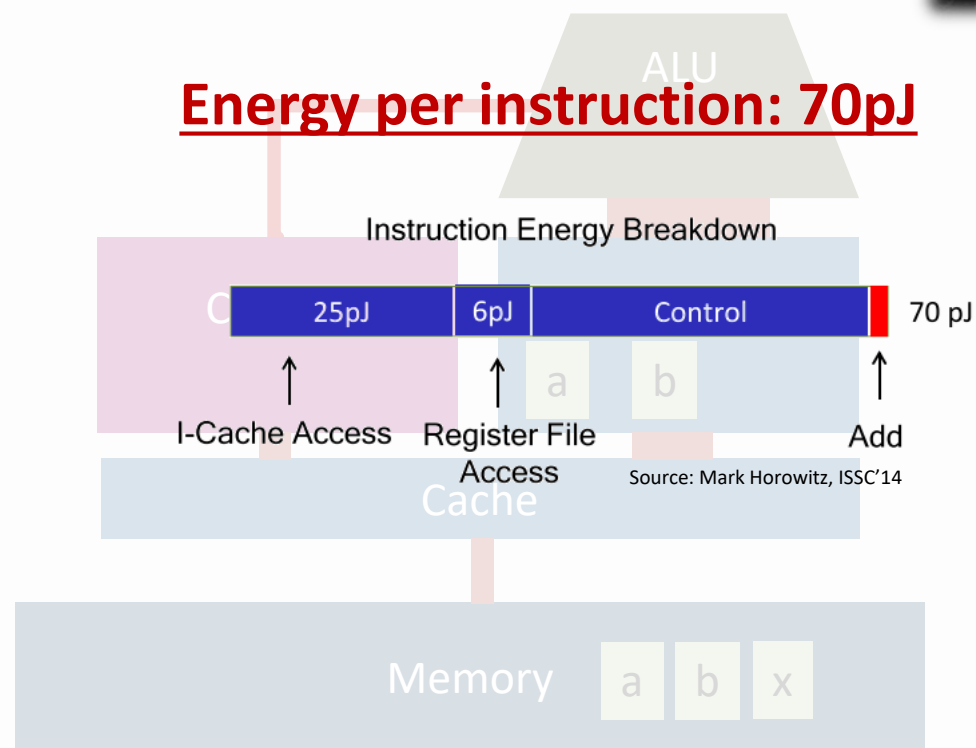
Turing Award 1977 (Backus): "Surely there must be a less primitive way of making big changes in the store than pushing vast numbers of words back and forth through the von Neumann bottleneck."

Load-store ("von Neumann")



$$x = a + b$$

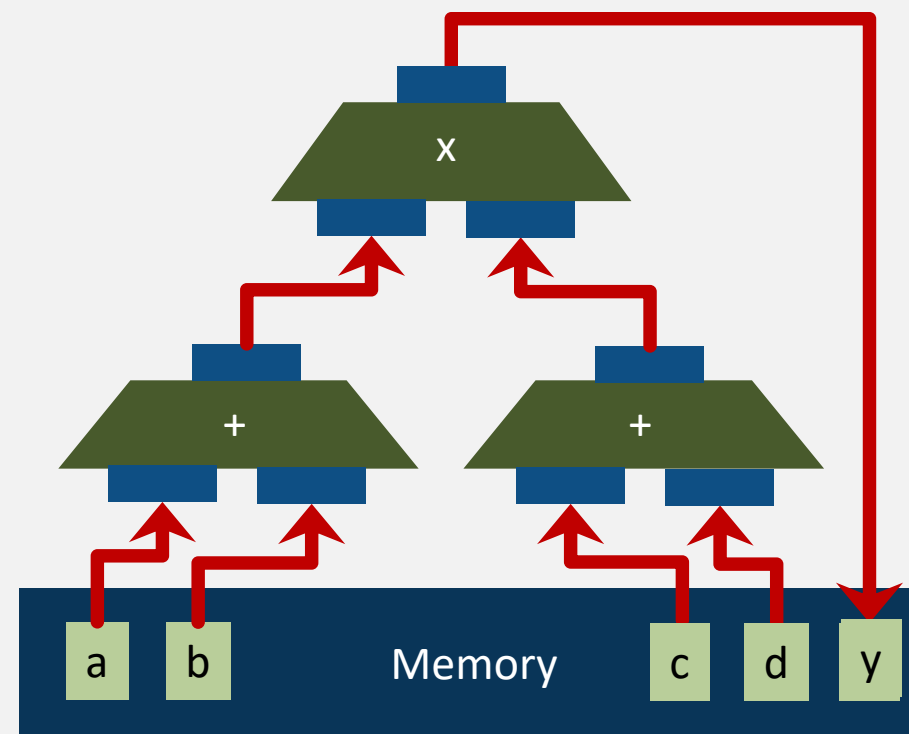
Energy per instruction: 70pJ



Static Dataflow ("non von Neumann")



$$y = (a + b) * (c + d)$$



Load-store vs. Dataflow architectures

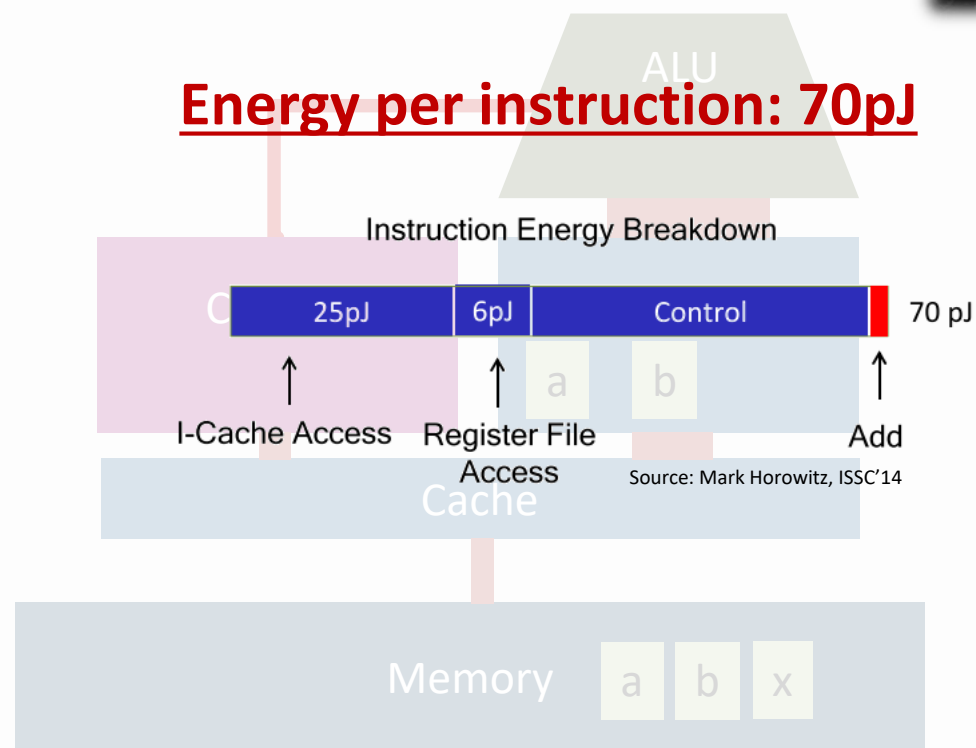
Turing Award 1977 (Backus): "Surely there must be a less primitive way of making big changes in the store than pushing vast numbers of words back and forth through the von Neumann bottleneck."

Load-store ("von Neumann")



$$x = a + b$$

Energy per instruction: 70pJ

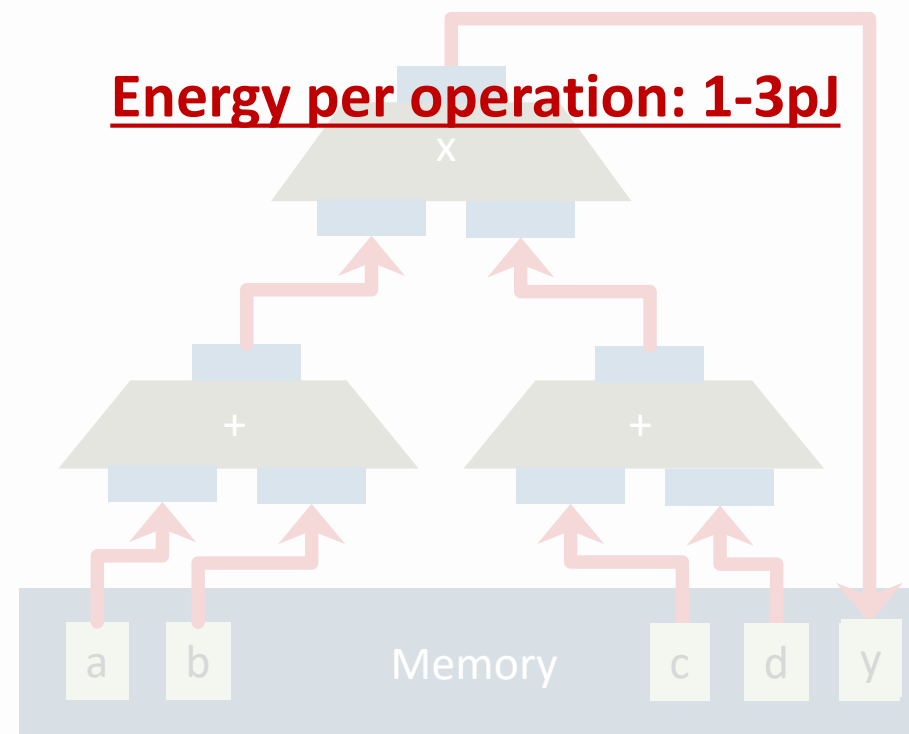


Static Dataflow ("non von Neumann")



$$y = (a + b) * (c + d)$$

Energy per operation: 1-3pJ



Load-store vs. Dataflow architectures

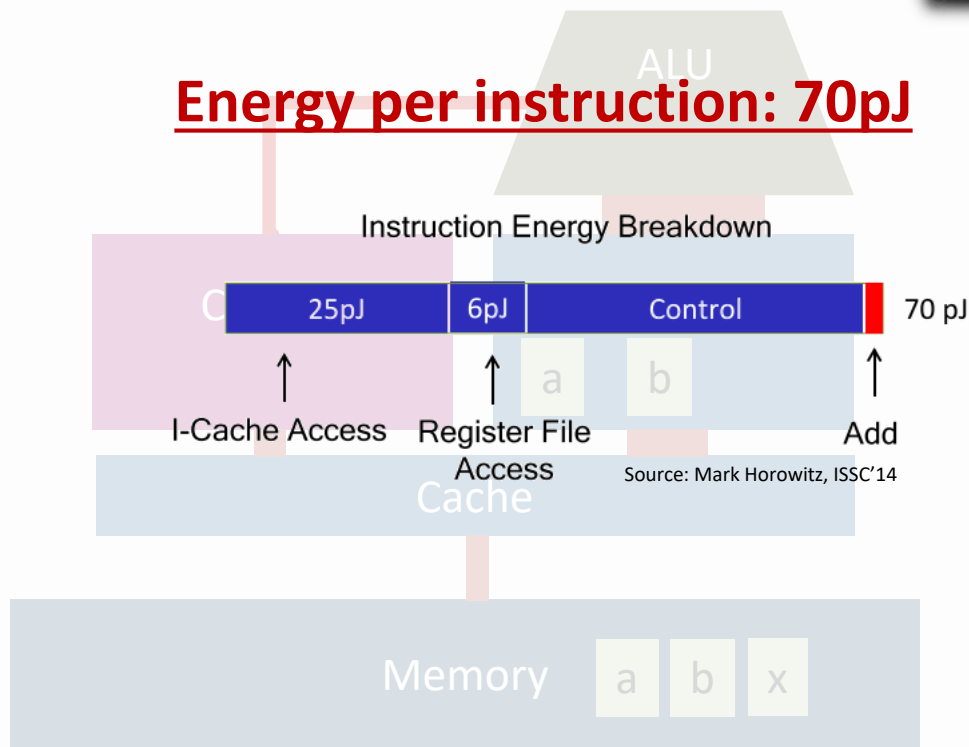
Turing Award 1977 (Backus): "Surely there must be a less primitive way of making big changes in the store than pushing vast numbers of words back and forth through the von Neumann bottleneck."

Load-store ("von Neumann")



$$x = a + b$$

Energy per instruction: 70pJ

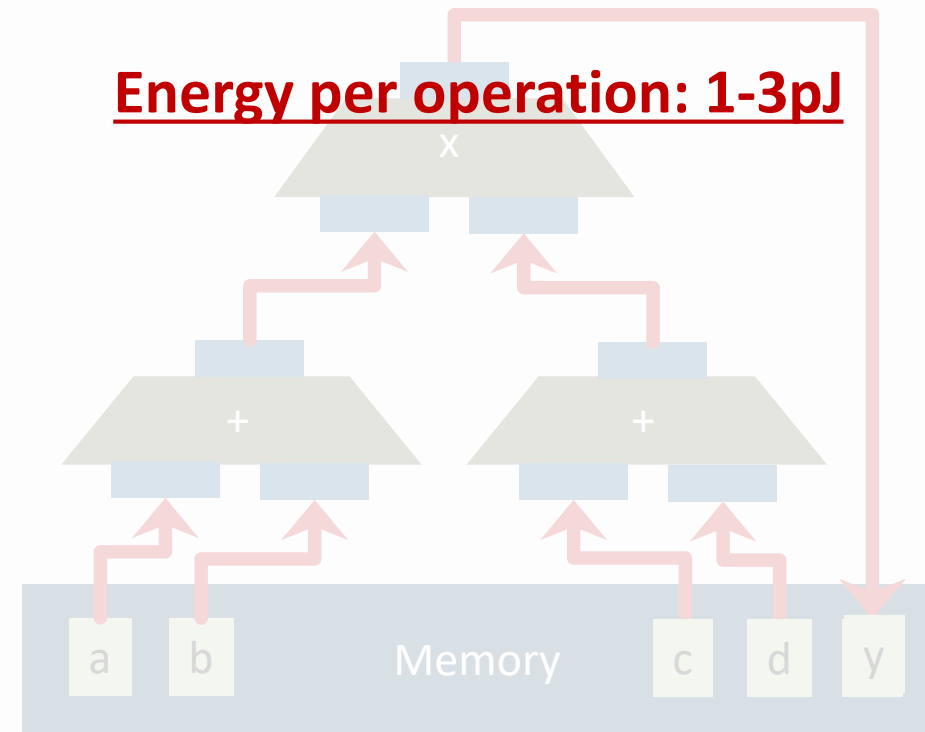


Static Dataflow ("non von Neumann")



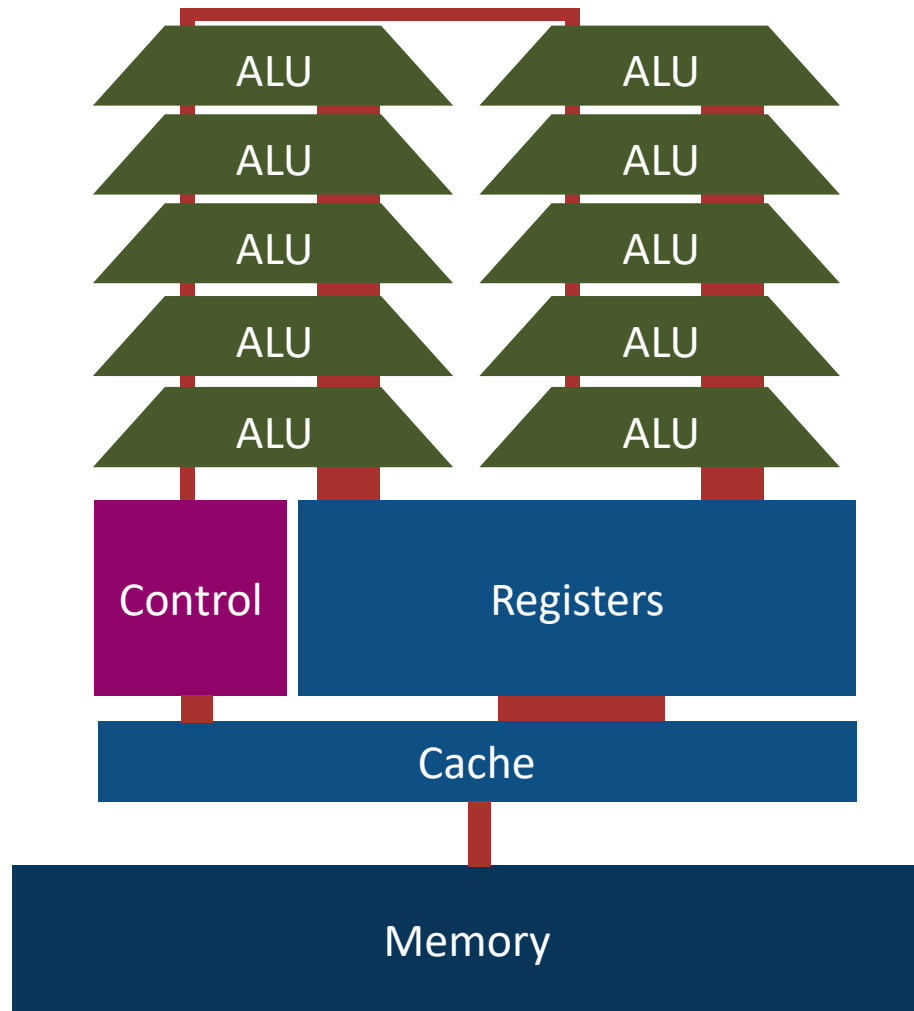
$$y = (a + b) * (c + d)$$

Energy per operation: 1-3pJ

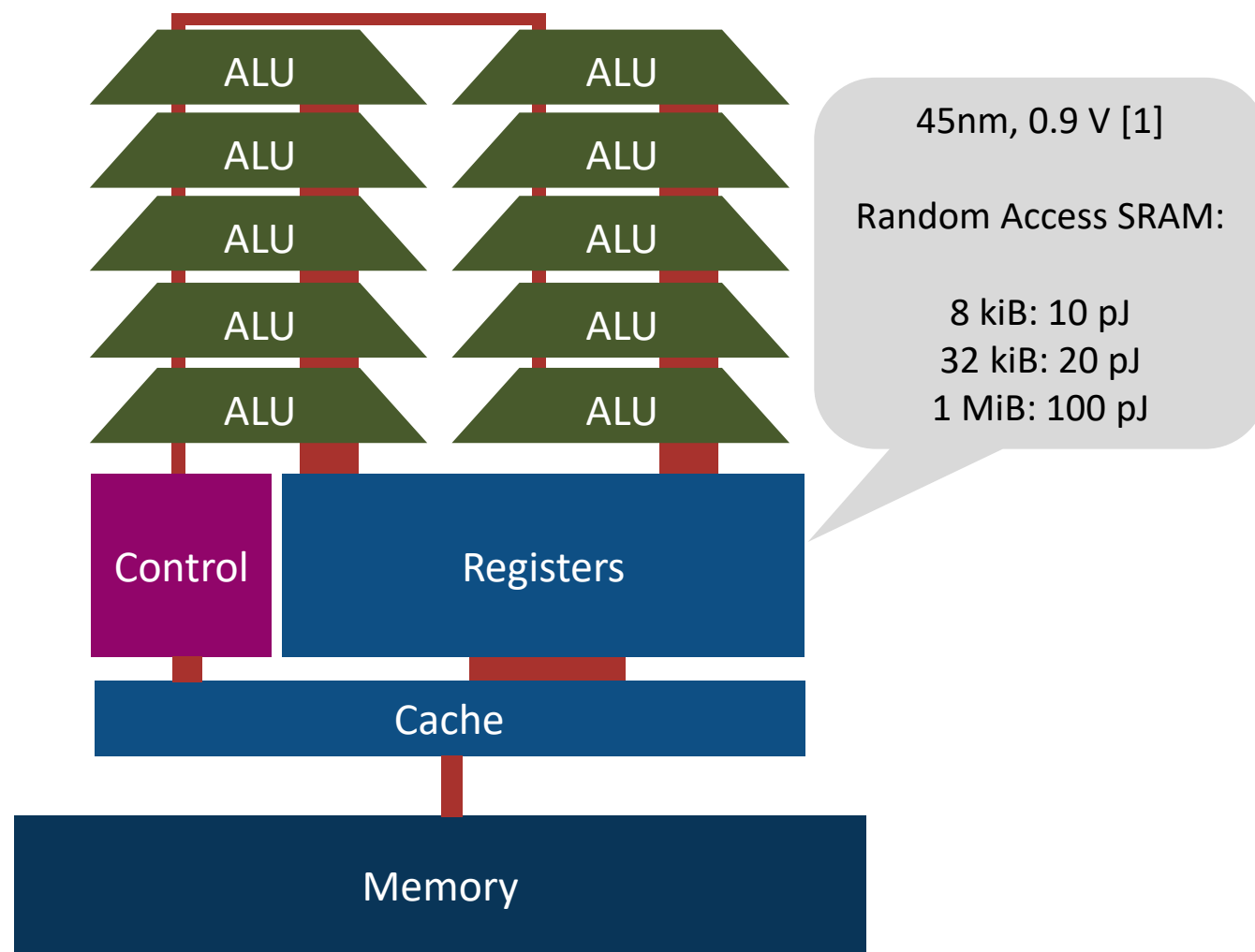


Control Locality

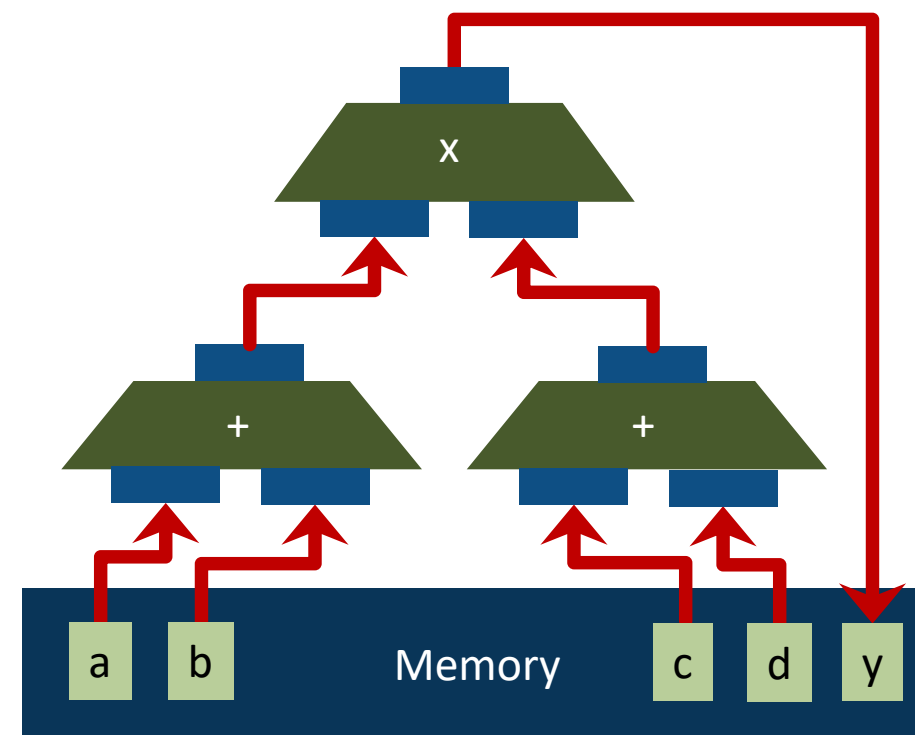
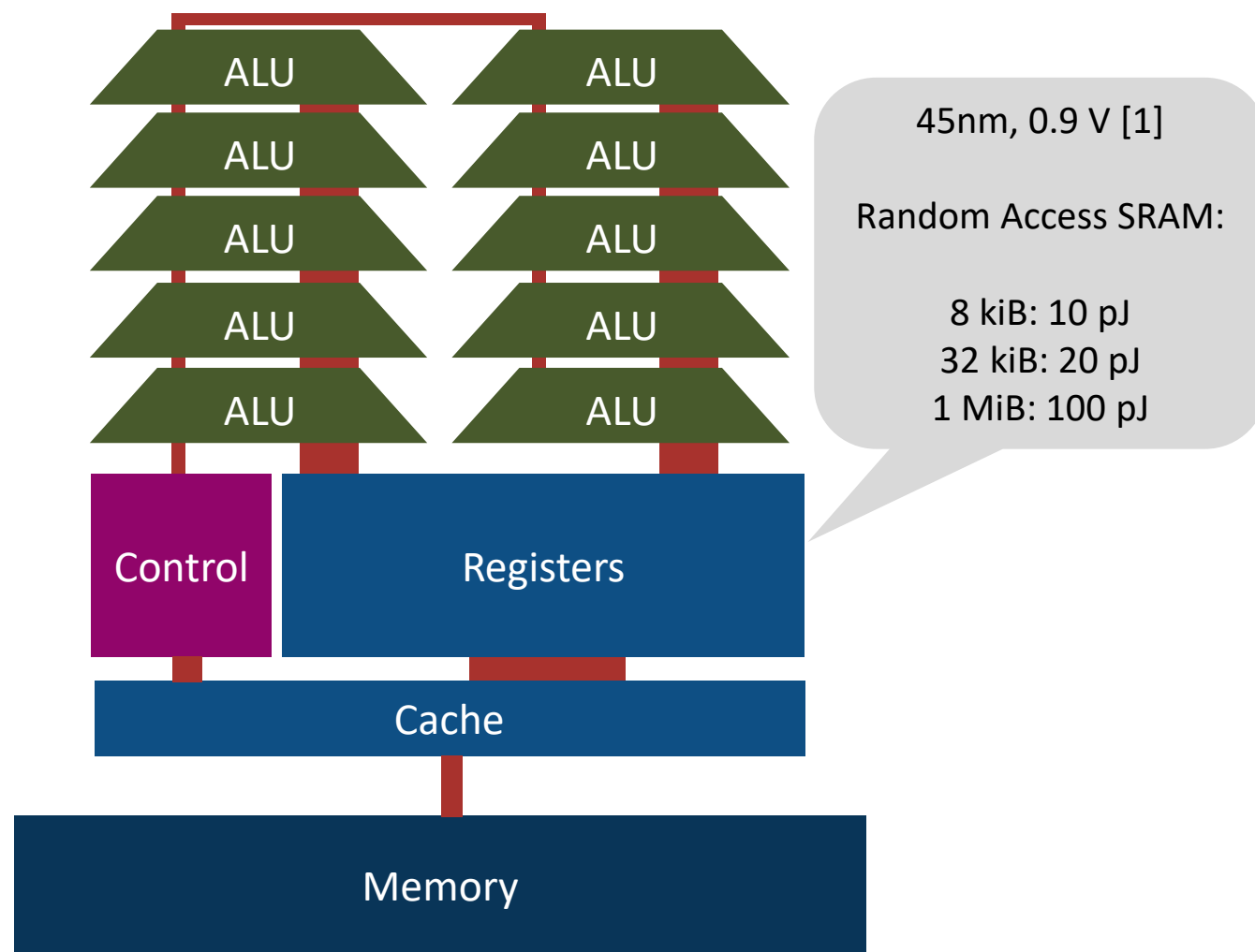
Single Instruction Multiple Data/Threads (SIMD - Vector CPU, SIMT - GPU)



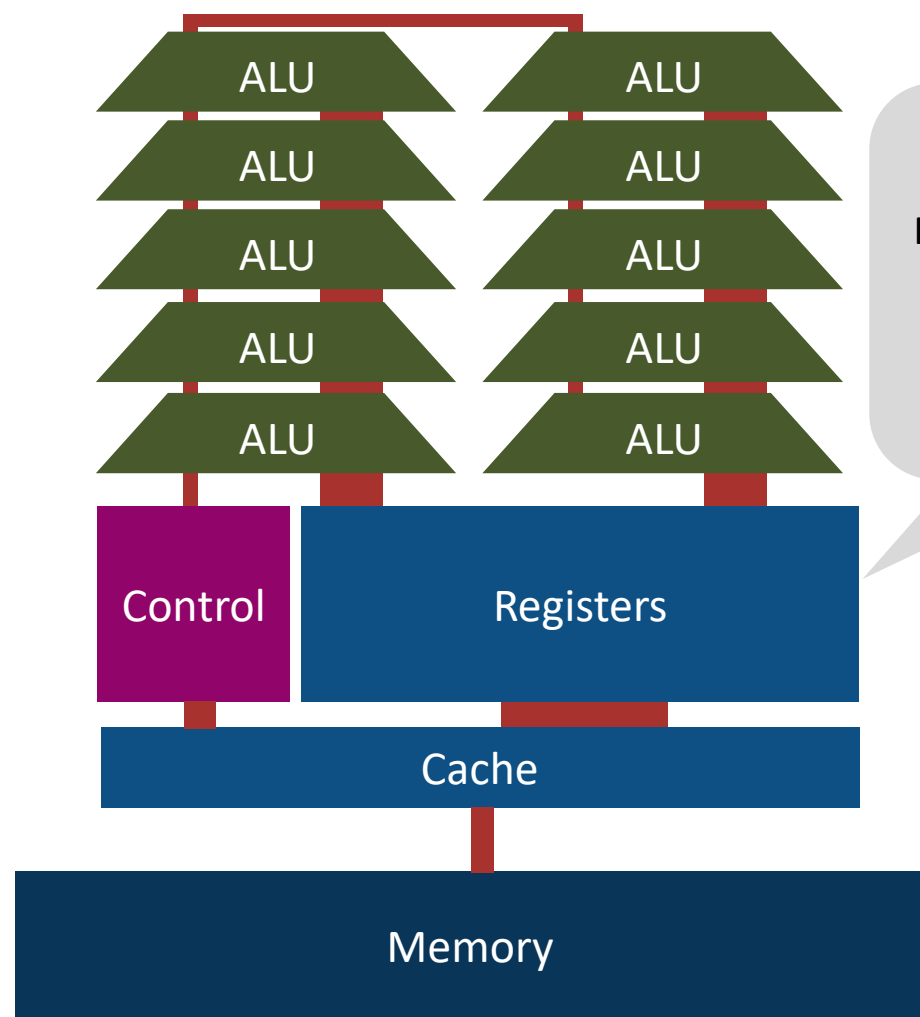
Single Instruction Multiple Data/Threads (SIMD - Vector CPU, SIMT - GPU)



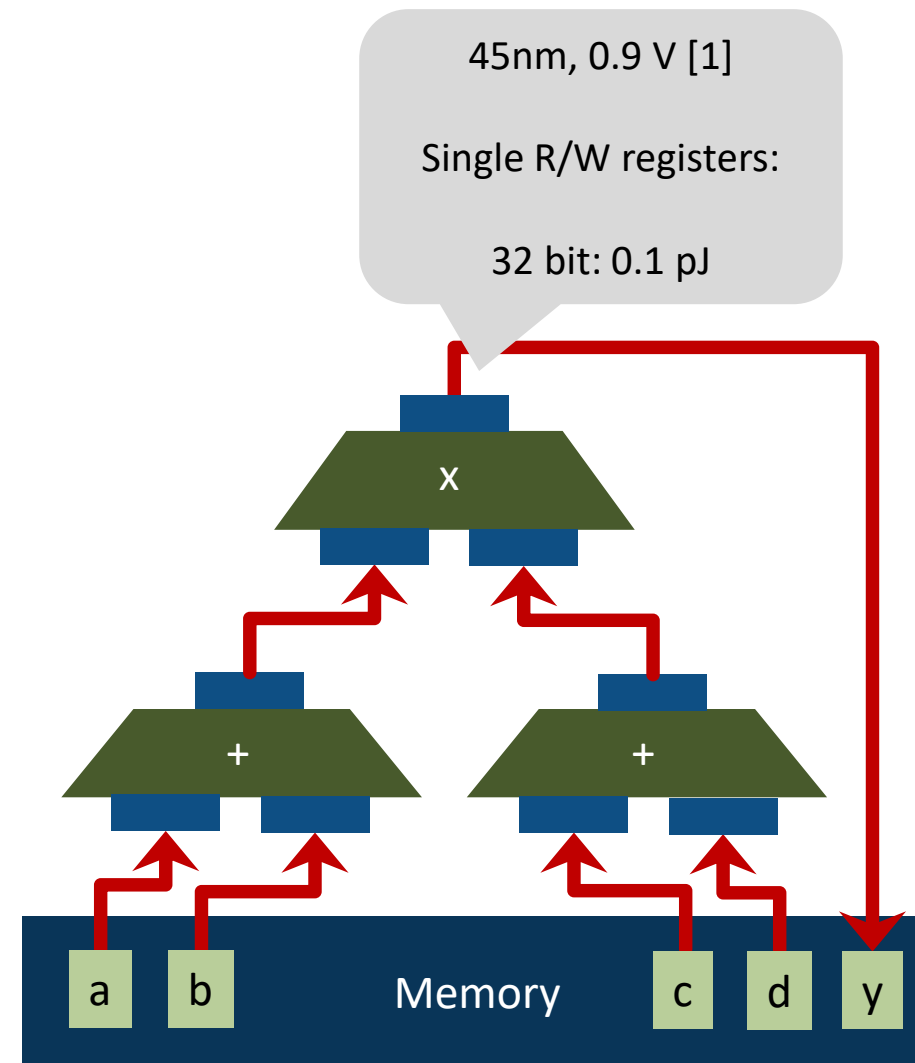
Single Instruction Multiple Data/Threads (SIMD - Vector CPU, SIMT - GPU)



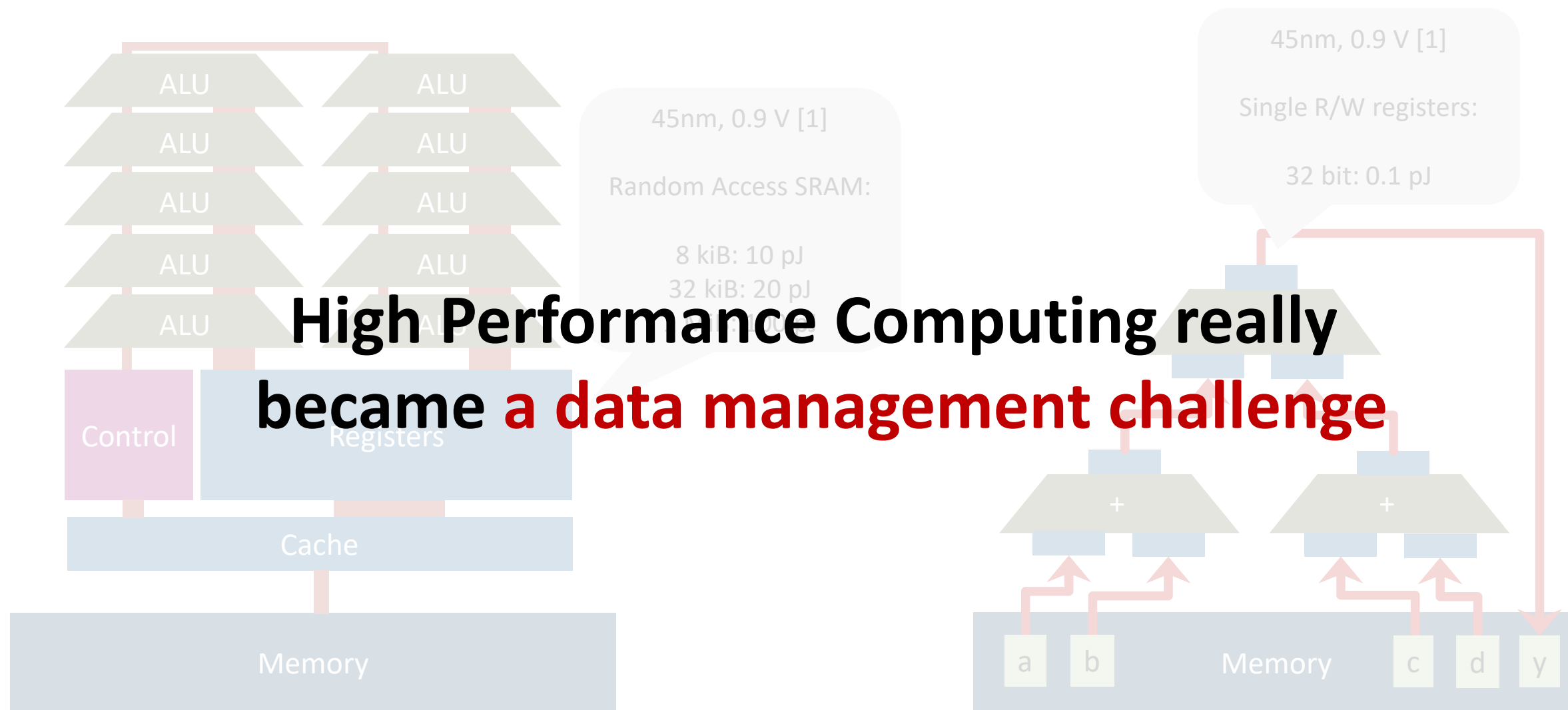
Single Instruction Multiple Data/Threads (SIMD - Vector CPU, SIMT - GPU)



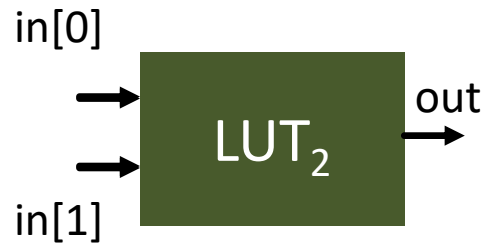
45nm, 0.9 V [1]
 Random Access SRAM:
 8 kiB: 10 pJ
 32 kiB: 20 pJ
 1 MiB: 100 pJ



Single Instruction Multiple Data/Threads (SIMD - Vector CPU, SIMT - GPU)

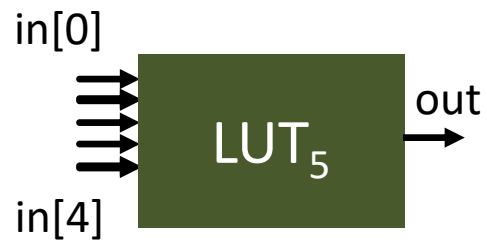


Field Programmable Gate Arrays (FPGA)



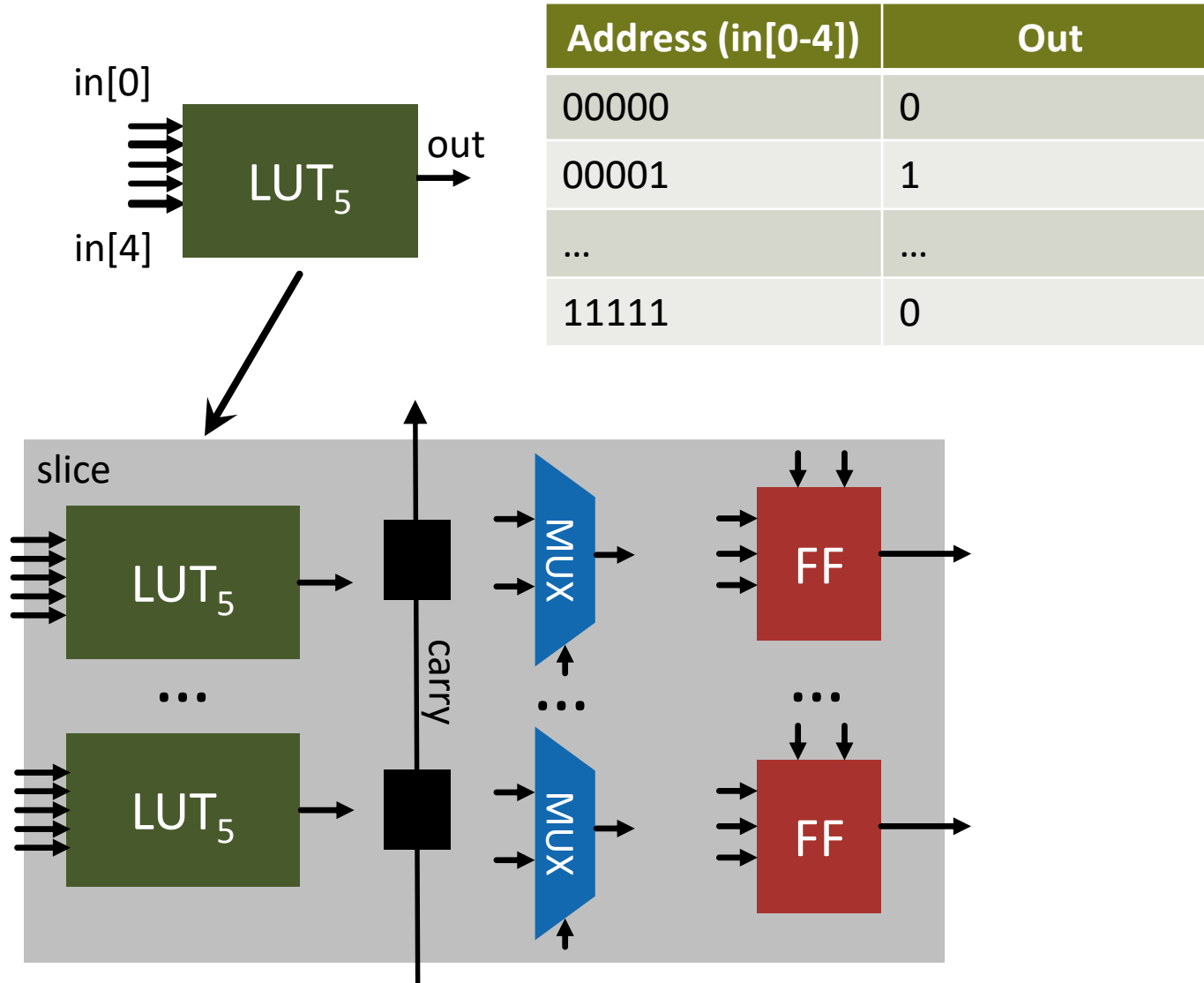
Address (in[0-1])	Out
00	0
01	1
10	1
11	0

Field Programmable Gate Arrays (FPGA)

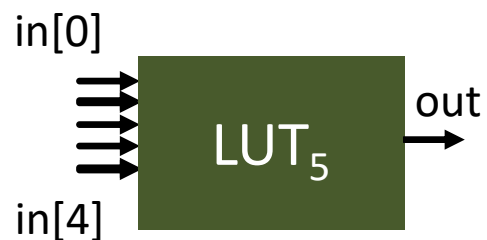


Address (in[0-4])	Out
00000	0
00001	1
...	...
11111	0

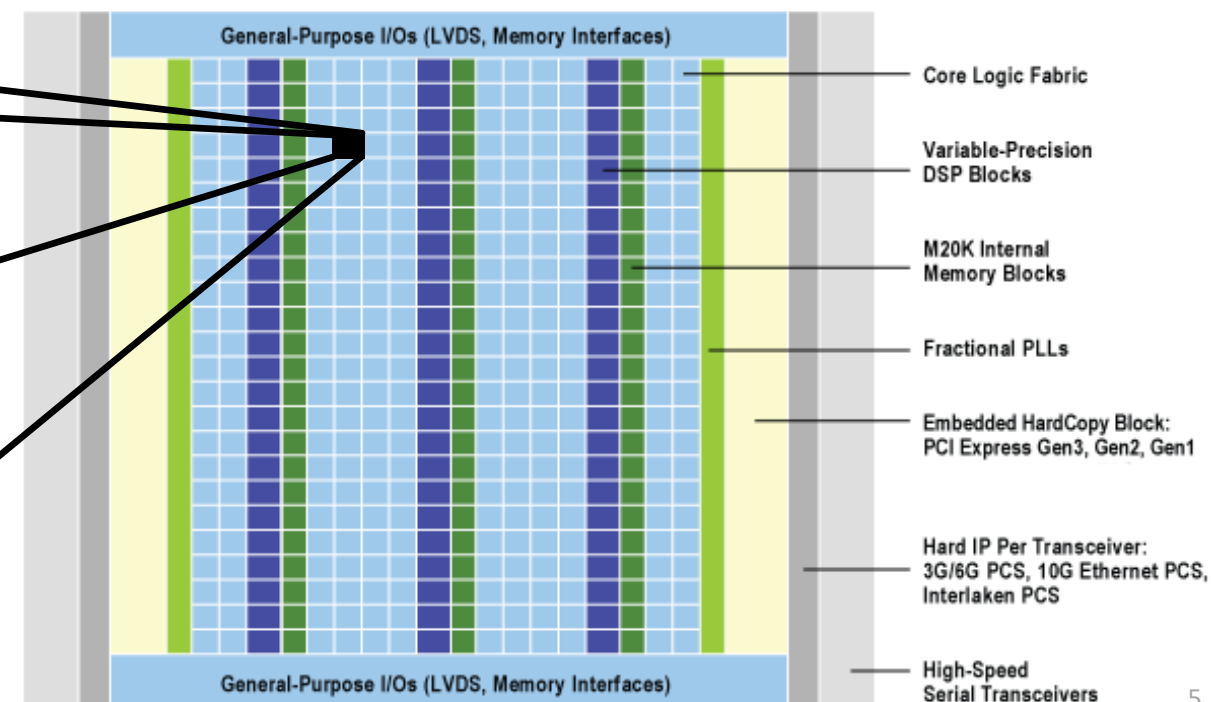
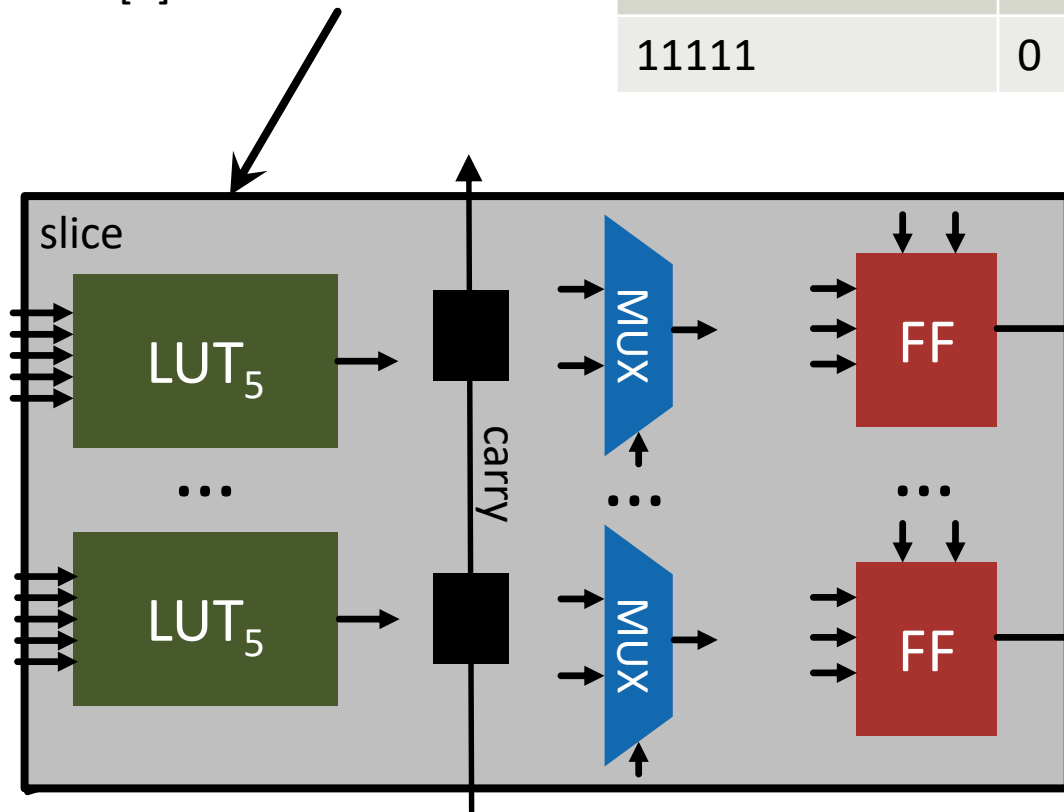
Field Programmable Gate Arrays (FPGA)



Field Programmable Gate Arrays (FPGA)



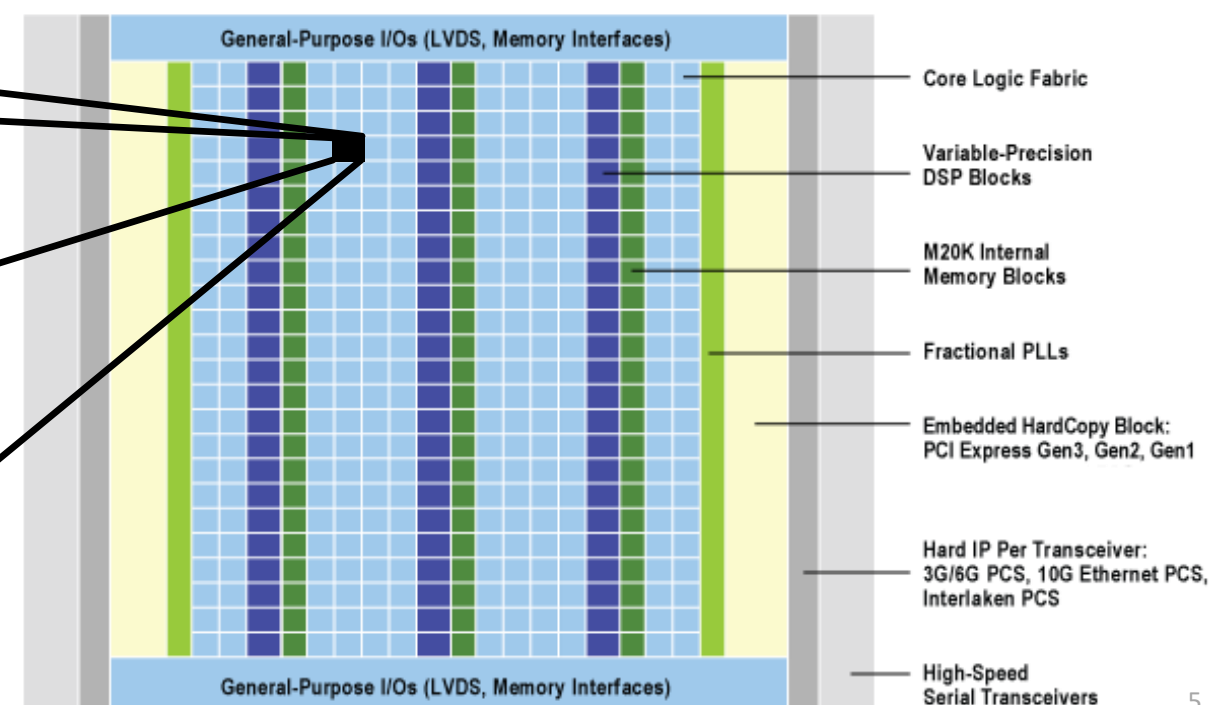
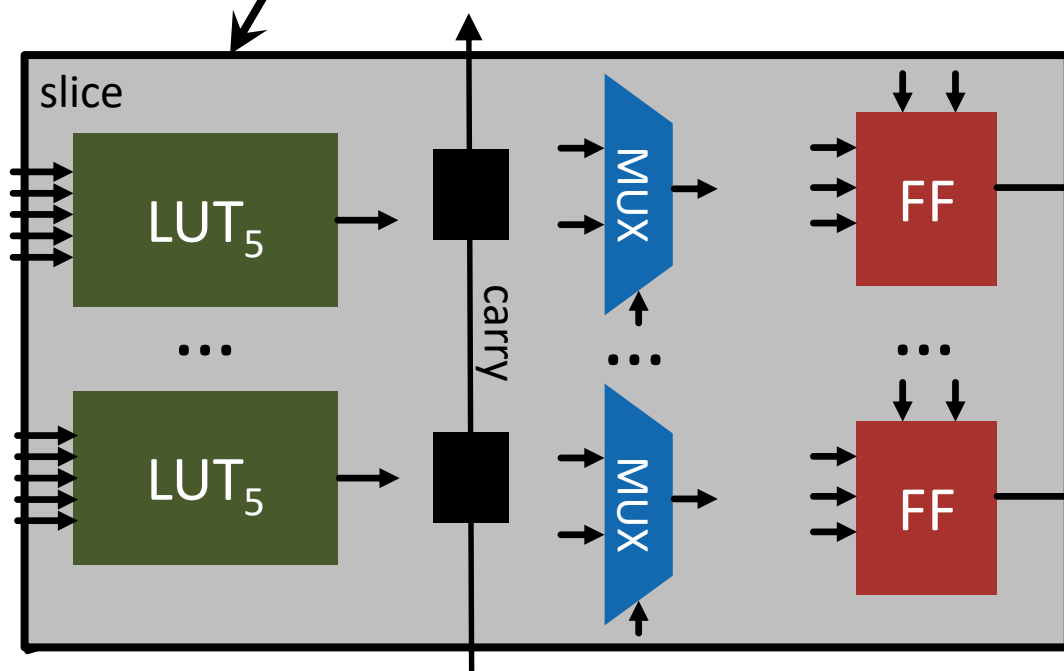
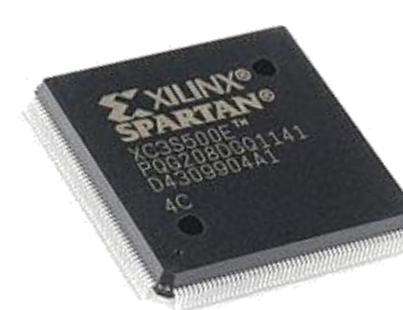
Address (in[0-4])	Out
00000	0
00001	1
...	...
11111	0



Field Programmable Gate Arrays (FPGA)



Address (in[0-4])	Out
00000	0
00001	1
...	...
11111	0



High-Performance FPGA Hardware Overview



- **14 nm Intel Tri-Gate**
 - 1 GHz
- **10 TF single precision**
- **5.5M Logic Elements**
 - 4-input LUT, register, carry, etc.
- **Block RAM: 28.6 MiB**
- **Hardened DRAM controller DDR 4**
 - Various options for memory
- **Hyper Flex Interconnect with Regs.**
- **TDP: 125W (estimated)**

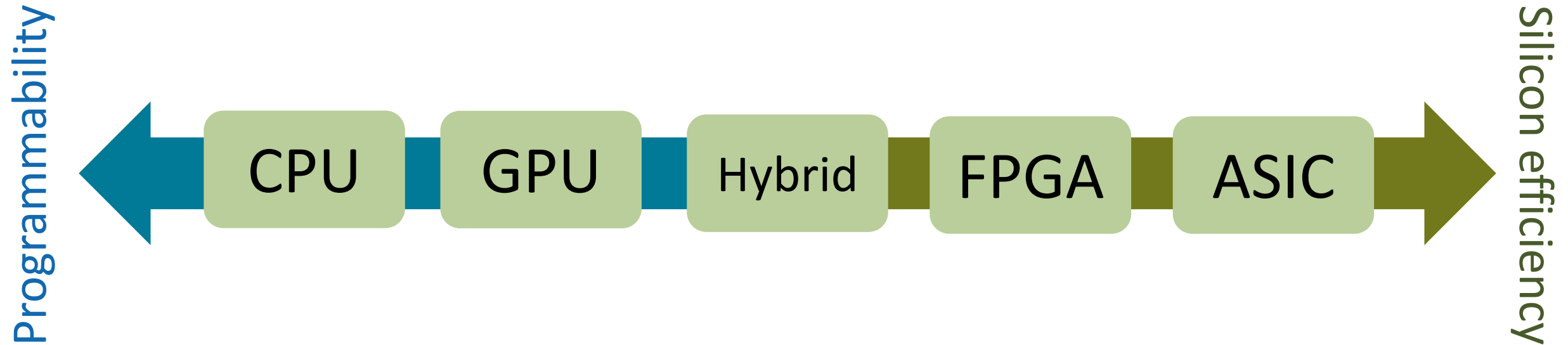


- **16 nm FinFET**
 - ~600 MHz
- **6,840 DSPs (3.1 TF single prec.)**
- **2.5M Logic Elements**
 - 1,182,000 5-input LUTs
 - 2,364,000 FFs
- **Block RAM: 9.1 MiB**
- **TDP: 95 W (Amazon F1 power limit)**

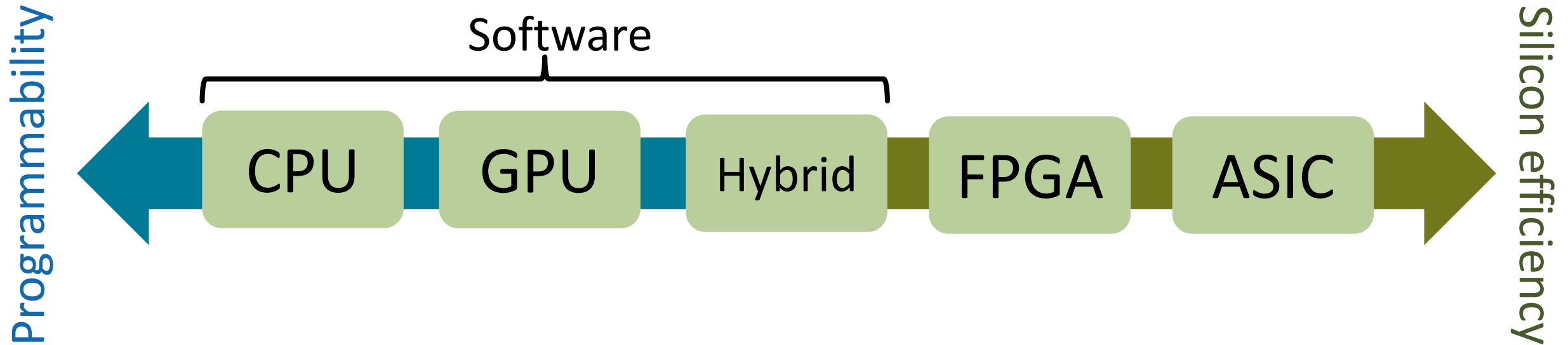


- **12 nm**
 - 1455 MHz
- **5,120 cores (15.7 TF single prec.)**
 - CUDA programming
- **On-chip memory:**
 - Registers: 20.8 MiB
 - L1/SM: 7.7 MiB
 - L2 Cache: 6.1 MiB
- **TDP: 300W**

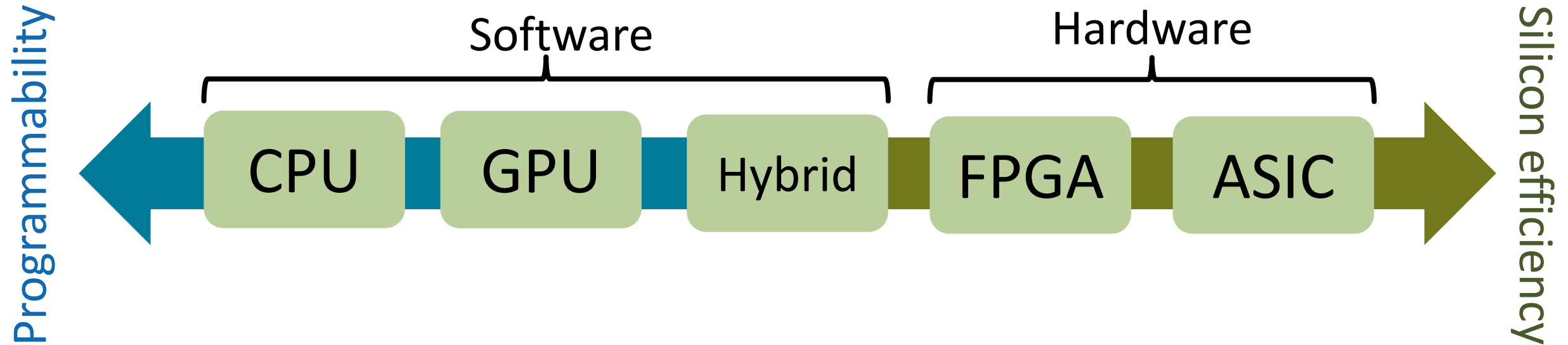
Ease of programmability vs. efficiency



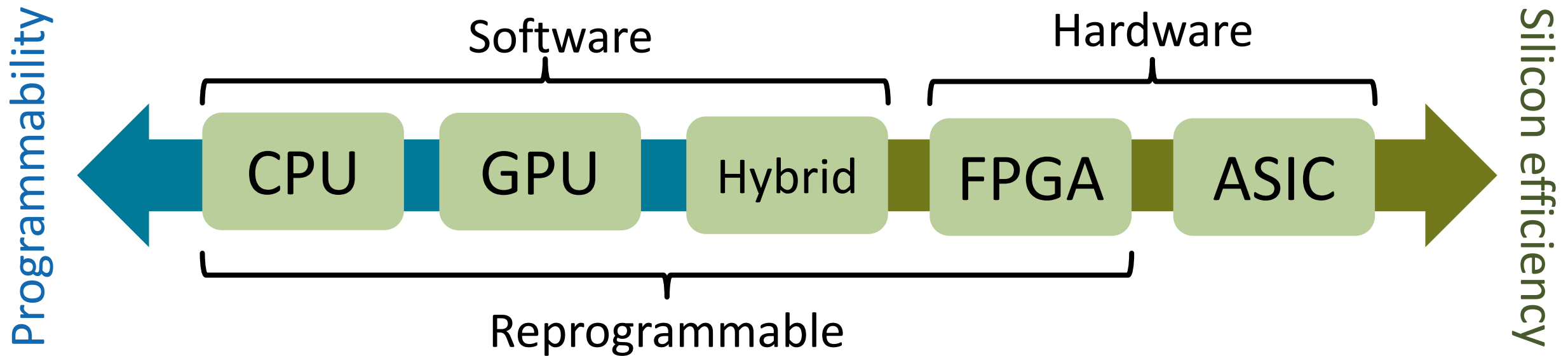
Ease of programmability vs. efficiency



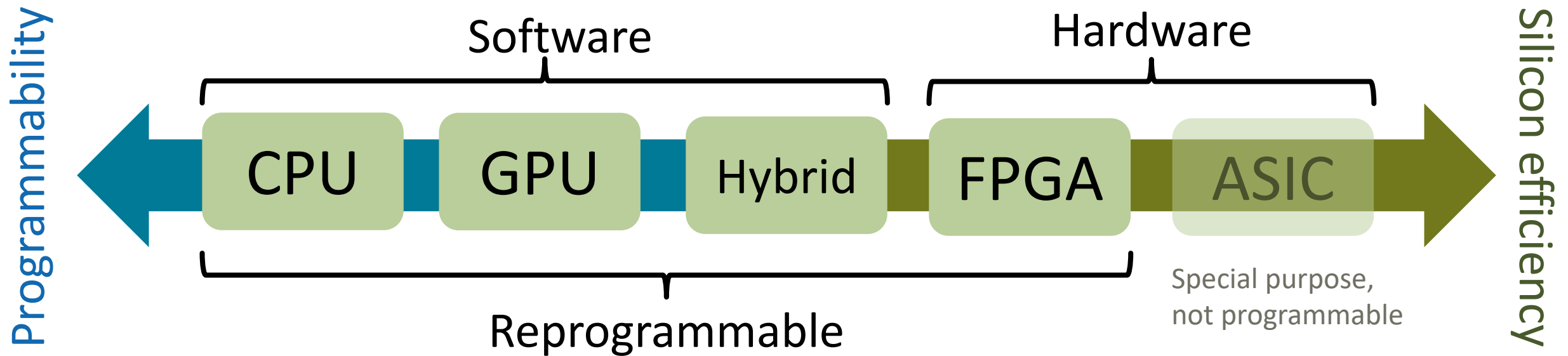
Ease of programmability vs. efficiency



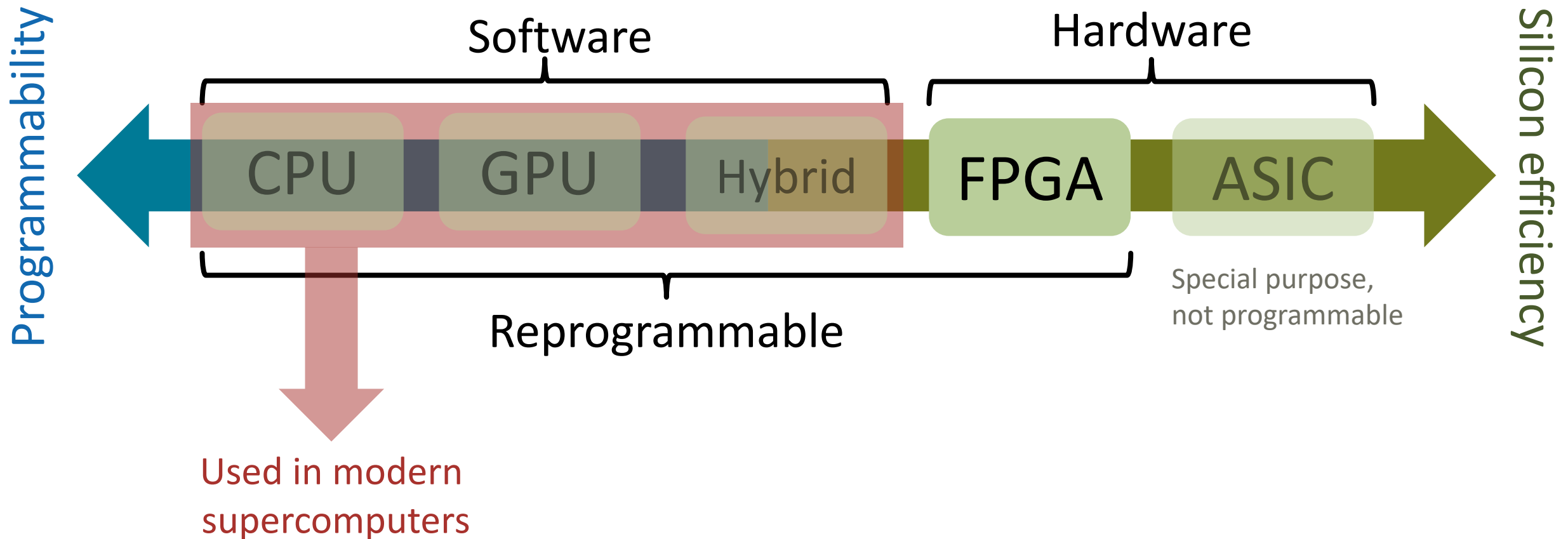
Ease of programmability vs. efficiency



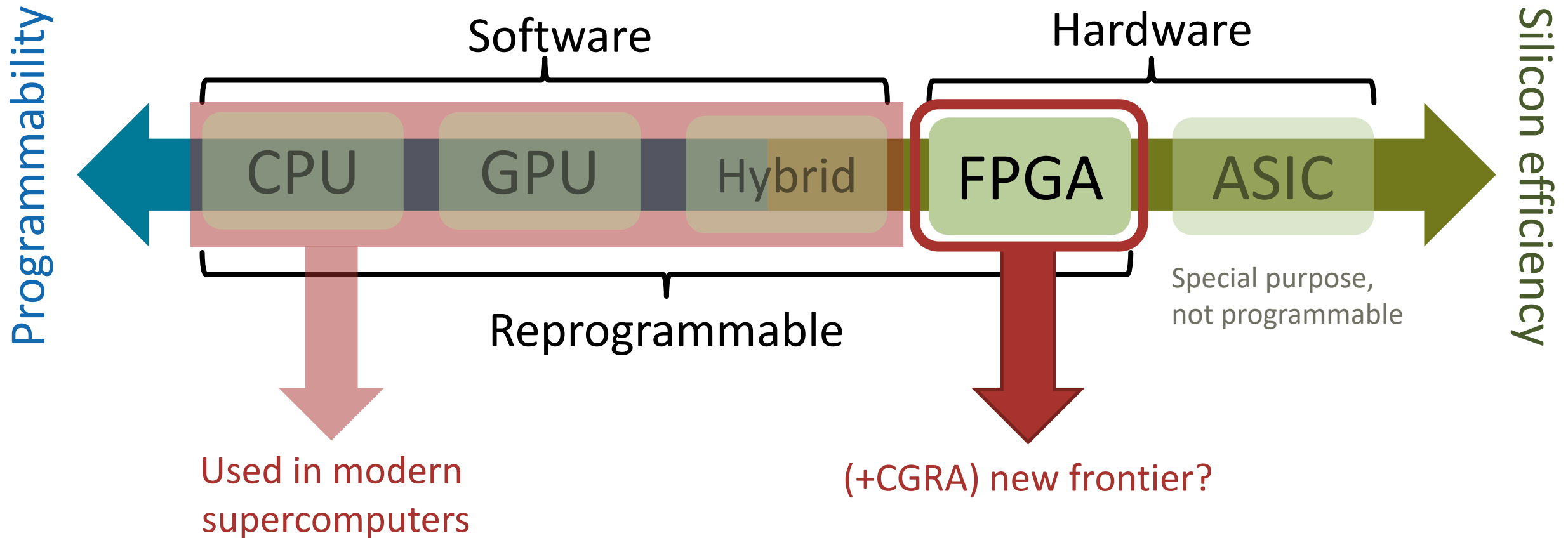
Ease of programmability vs. efficiency



Ease of programmability vs. efficiency



Ease of programmability vs. efficiency



FPGAs are already in the cloud since a while

Microsoft Goes All in for FPGAs to Build Out AI Cloud

Michael Feldman | September 27, 2016 08:42 CEST



Software giant bets the [server] farm on reconfigurable computing

Microsoft has revealed that Altera FPGAs have been installed across every Azure cloud server, creating what the company is calling "the world's first AI supercomputer." The deployment spans 15 countries and represents an aggregate performance of more than one exa-op. The announcement was made by Microsoft CEO Satya Nadella and engineer Doug Burger during the opening keynote at the Ignite Conference in Atlanta.

The FPGA build-out was the culmination of more than five years of work at Microsoft to find a way to accelerate machine learning and other throughput-demanding applications and services in its Azure cloud. The effort began in earnest in 2011, when the company launched Project Catapult, the R&D initiative to design an acceleration fabric for AI services and applications. The rationale was that CPU evolution, a la Moore's Law, was woefully inadequate in keeping up with the demands of these new hyperscale applications. Just as in traditional high performance computing, multicore CPUs weren't keeping up with demand.



Doug Burger with Microsoft-designed FPGA card

Amazon Adds FPGA Instance to Public Cloud

Michael Feldman | December 7, 2016 07:17 CET



In a blog post published last week, Amazon Web Services Chief Evangelist Jeff Barr announced a new Elastic Compute Cloud (EC2) instance, known as the F1, which incorporates Xilinx FPGAs. The web giant says users will not only be able to build FPGA-accelerated applications for their own purposes with the new instance, but also resell them in the AWS Marketplace to third parties.



Barr makes the conventional pitch for FPGA acceleration, noting its inherent advantage in implementing parallel computation much more efficiently than general-purpose chips by being able to implement an application's dataflow at the level of the logic elements. Writes Barr:

"This highly parallelized model is ideal for building custom accelerators to process compute-intensive problems. Properly programmed, an FPGA has the potential to provide a 30x speedup to many types of genomics, seismic analysis, financial risk analysis, big data search, and encryption algorithms and applications."

The F1 instance is equipped with Xilinx's Virtex Ultrascale+ VU9P, a 16nm device that contains about 2.5 million logic elements and over 6,800 DSP engines. The F1's host CPU is Intel's 18-core Broadwell E5 2686 v4 processor, which runs at 2.3 GHz. The instance may be configured with up to 8 FPGAs, 967 GiB of memory and 4 TB of NVMe flash storage. The PCIe fabric allows multiple FPGAs to communicate with one another directly, as well as share the same memory space.

Application developers will have access to the Xilinx Vivado Design Suite free of charge, but that will require that the application code be implemented in VHDL or Verilog. Third-party tools can also be used, including higher level frameworks like OpenCL, but that means users will be responsible for their own development environments.

At this point, FPGA application development is geared toward a rather niche audience, so this will hardly be a volume business for Amazon in the near-term. HPC cloud specialist Nimbix recently added Xilinx FPGAs to its own infrastructure in the hopes of building a customer base of FPGA computing enthusiasts. Much of the initial customer base will be Xilinx engineers themselves, who will be developing and testing software on their own product.

Vendors trying to push FPGAs into HPC (remember the Cray XD1, 2005)

Intel Launches FPGA Accelerator Aimed at HPC and HPDA Applications

Michael Feldman | December 20, 2017 12:44 CET



E-mail



Tweet



Like



+1



Share

4

Intel has released the Stratix 10 MX, the company's first FPGA family that includes integrated High Bandwidth Memory [HBM2].



HBM2 is comprised of a stack of DRAM chips linked to one another in parallel using thru silicon via (TSV) interconnects. The 3D design enables much faster data communication rates compared to normal 2D setups, which are limited by the number of pins that can be connected to the edge of the DRAM chips. According to Intel, HBM2 will deliver memory speeds as much as 10 times faster – up to 256 GB/second per HBM2 stack – than what would be possible with a conventional DDR4 solution. The technology also has the advantage of delivering about 35 percent better performance per watt.

Depending on the specific model, and Stratix 10 MX devices will be equipped with 3.25, 8, or 16 gigabytes of HBM2 memory.

But HPC uses GPUs – why?

- **Productivity:** CUDA enabled GPGPU without hacking the graphics pipeline
- **Hardware support:** Tesla line with ECC, double/half precision

```
template <typename T>
__global__ void integrateBodies(typename vec4<T>::Type *__restrict__ newPos,
                                typename vec4<T>::Type *__restrict__ oldPos,
                                typename vec4<T>::Type *vel,
                                unsigned int deviceOffset,
                                unsigned int deviceNumBodies,
                                float damping, int numTiles)
{
    int index = blockIdx.x * blockDim.x + threadIdx.x;

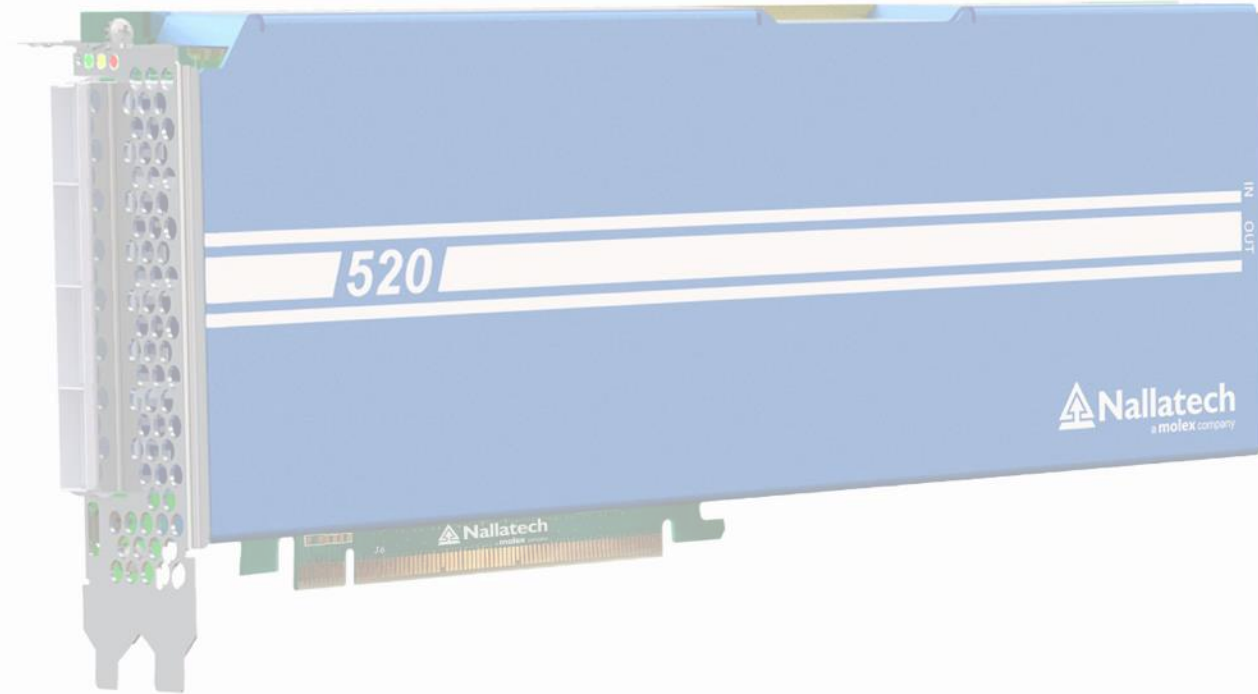
    if (index >= deviceNumBodies) {
        return;
    }

    typename vec4<T>::Type position = oldPos[deviceOffset + index];
```



FPGA are not used in HPC - why?

- **Productivity:** steep learning curve of hardware design, unpolished tools
- **Hardware support:** low bandwidth, no native floating point units

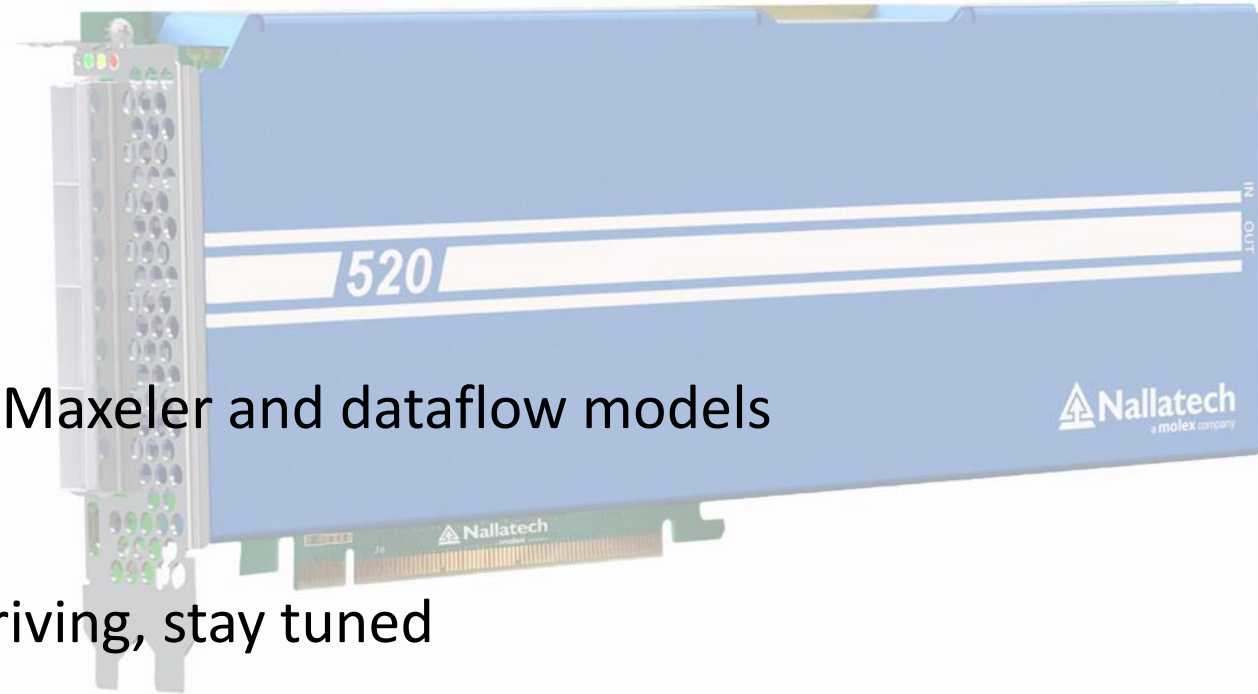


FPGA are not used in HPC - why?

- **Productivity:** steep learning curve of hardware design, unpolished tools
- **Hardware support:** low bandwidth, no native floating point units

Recent developments

- **Productivity:** OpenCL, HLS, (not so recent) Maxeler and dataflow models
 - Major focus on portability now!
- **Hardware support:** Stratix 10, HBM just arriving, stay tuned



High performance on FPGA

- **Typically low clock rates**
 - 250 MHz is considered good, 800 MHz somewhat of a record
 - For performance we rely on (spatially) massively parallel computation

High performance on FPGA

- **Typically low clock rates**
 - 250 MHz is considered good, 800 MHz somewhat of a record
 - For performance we rely on (spatially) massively parallel computation
- **Parallelism depends on *depth* and width of computations performed on the input data**
 - More complex optimization space
 - Tradeoff between logic, buffering, and time

Many more opportunities than von-Neumann architectures

High performance on FPGA

- **Typically low clock rates**
 - 250 MHz is considered good, 800 MHz somewhat of a record
 - For performance we rely on (spatially) massively parallel computation
- **Parallelism depends on *depth* and width of computations performed on the input data**
 - More complex optimization space
 - Tradeoff between logic, buffering, and time

Many more opportunities than von-Neumann architectures
- **Performance is mostly an exercise in data movement**
 - On-chip data movement can be “hard-wired” (data flow)
 - No instructions, just flow and state machines
- **Peak performance?**
 - Hard to define, fill chip with functional units?

Spatial Register Transfer Logic (RTL) Programming

- **Traditionally done in hardware description languages**
 - Register Transfer Logic (RTL)
VHDL, Verilog, System C, SystemVerilog
 - These are very verbose and very low level: every clock cycle is accounted for
Tedious programming at the bit-level
- **HDLs are still an abstraction: arbitrarily wide wires, registers and integer operations**
 - Allows some library-based operations
 - Still tedious if you just want to write “for(i=0; i<10; i++) x=a+b*c/x”
 - Chisel for Scala and MyHDL for Python are more productive approaches to hardware design
- **Place and route**
 - Map RTL “program” to 2D chip
 - Complex (NP hard) problem

High-level synthesis

- **Both major vendors, as well as third parties, now offer HLS tools**
 - Input C/C++/OpenCL is transformed to the spatial paradigm
 - Lift programming from the bit level to the word/datatype level
- **Xilinx and Intel both offer a C/C++ and an OpenCL tool**
 - Xilinx focused on the former, and Altera on the latter
 - Many other HLS tools ...



Bambu

LegUp High-Level Synthesis



bluespec



Microsoft Catapult

... many many more (cf. Nane et al.: “A Survey and Evaluation of FPGA High-Level Synthesis Tools”, Oct. 2016)