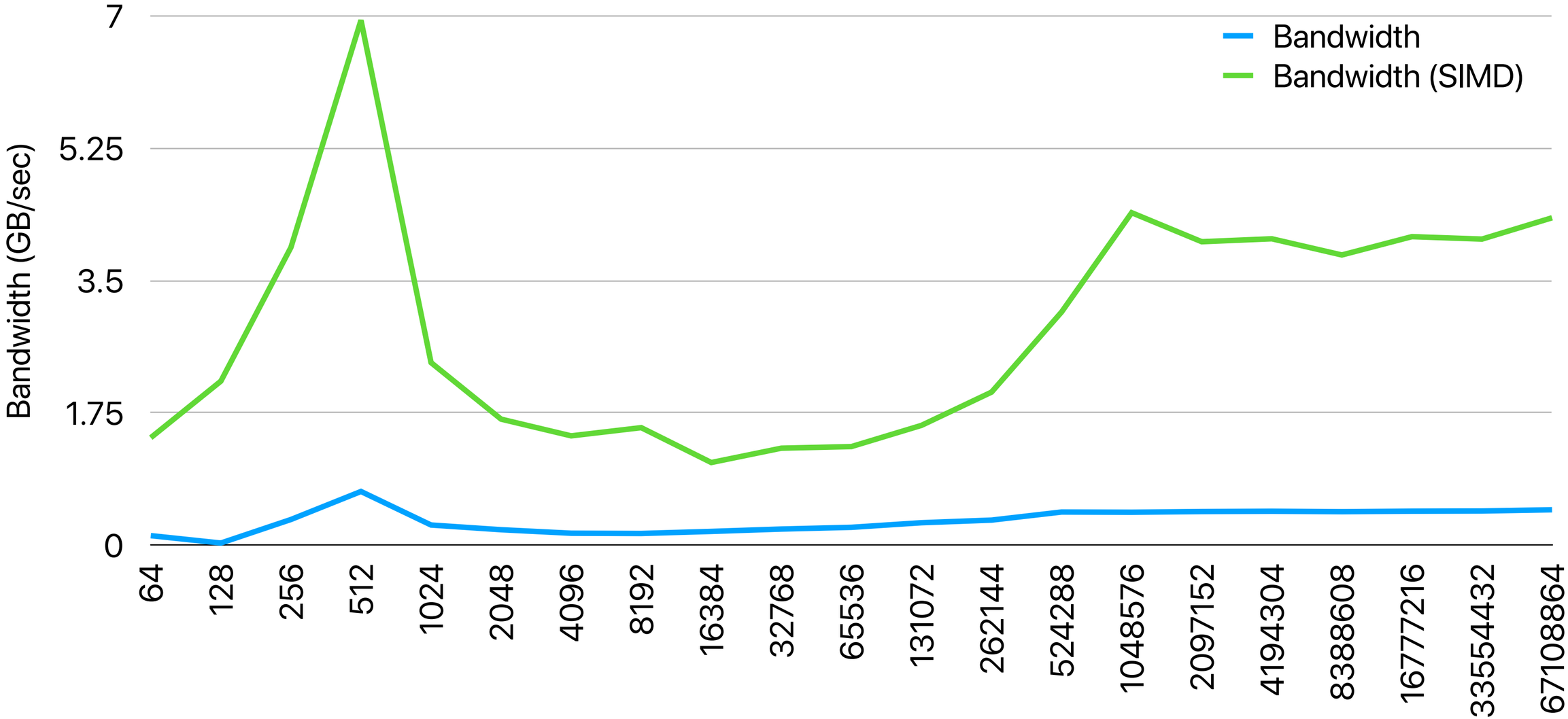
In/Near Memory Processing (2) & Basics on System Peripherals

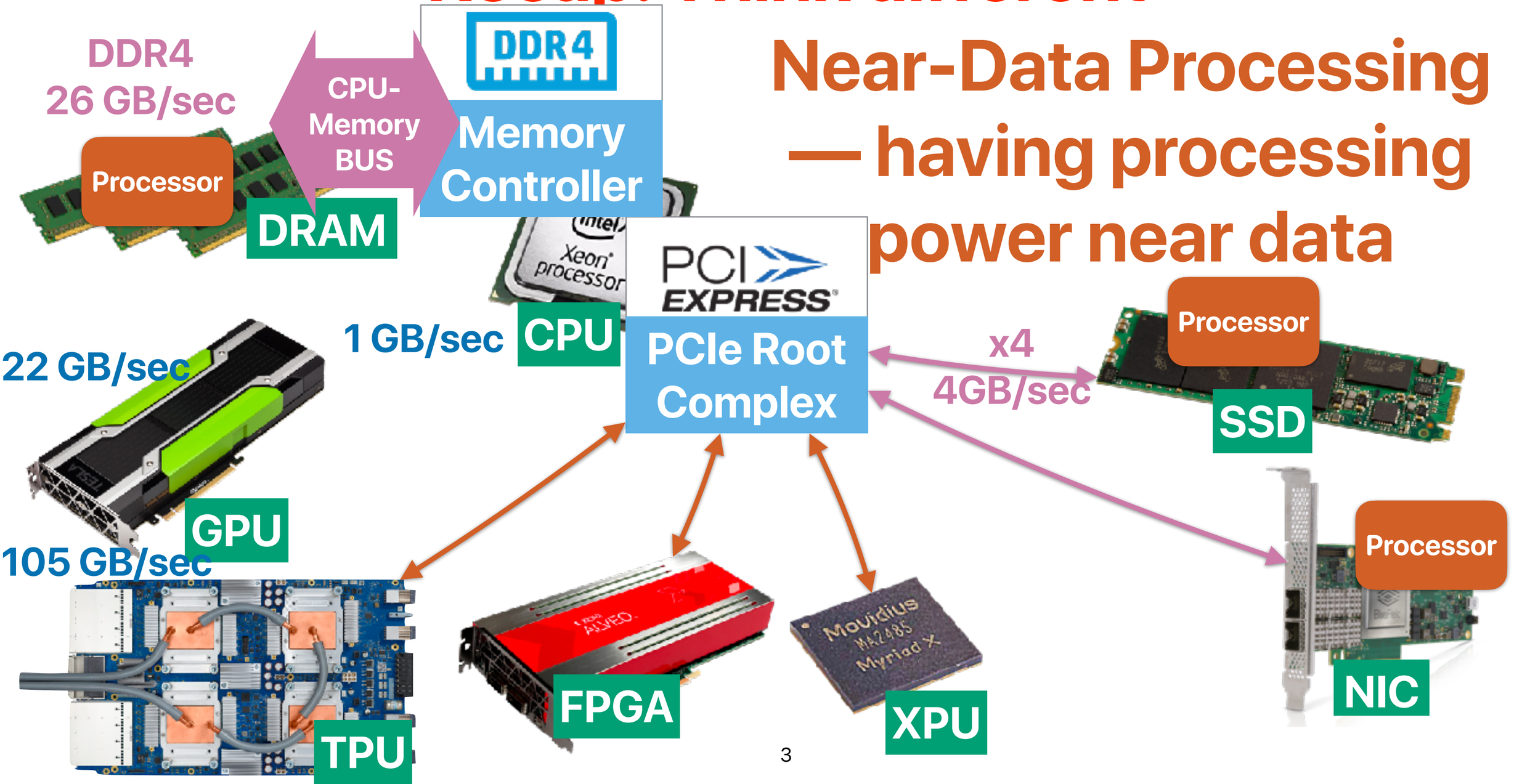
Hung-Wei Tseng

Recap: Memory copy bandwidth using SIMD

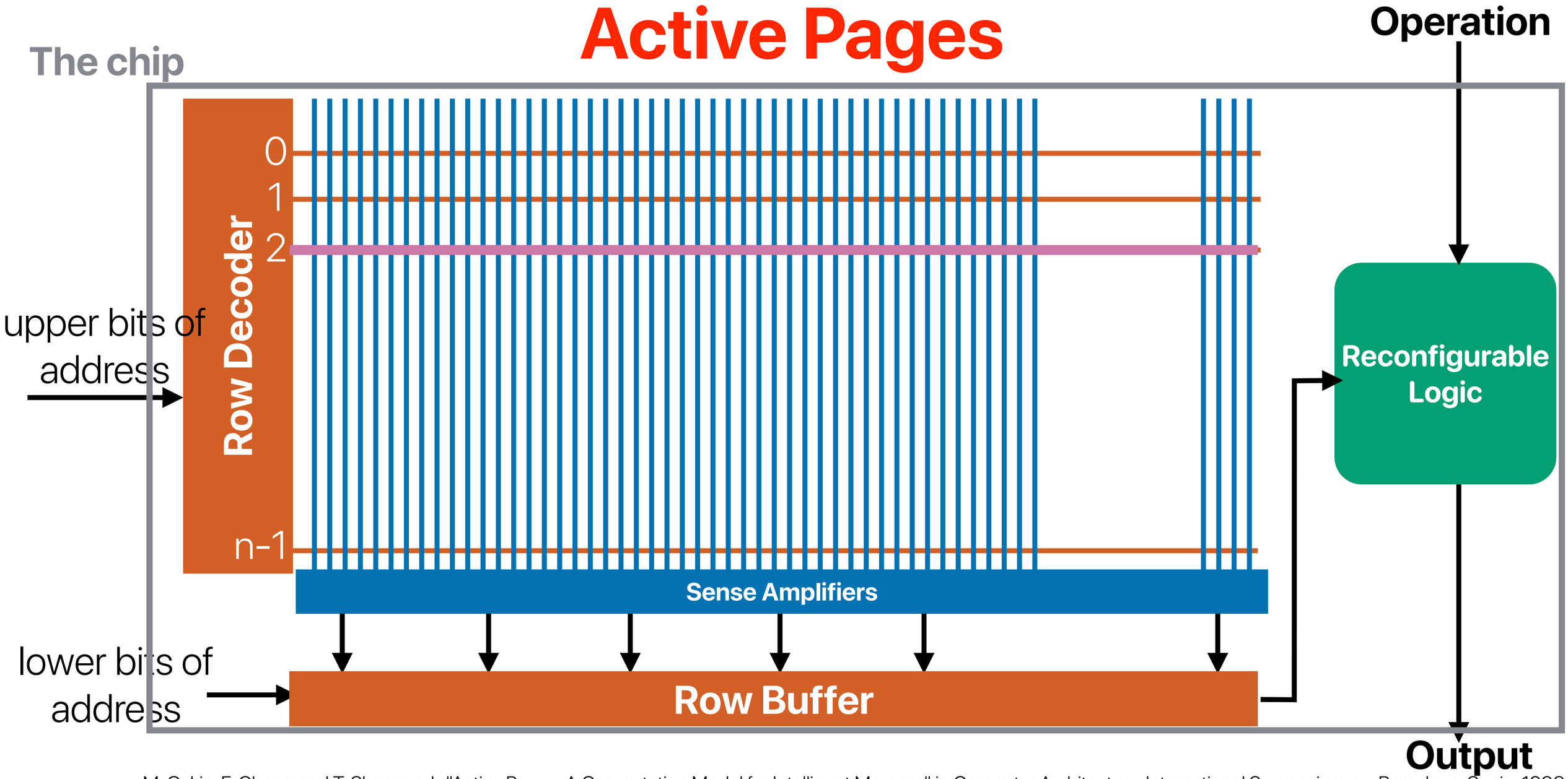


Recap: Think different

Near-Data Processing — having processing power near data

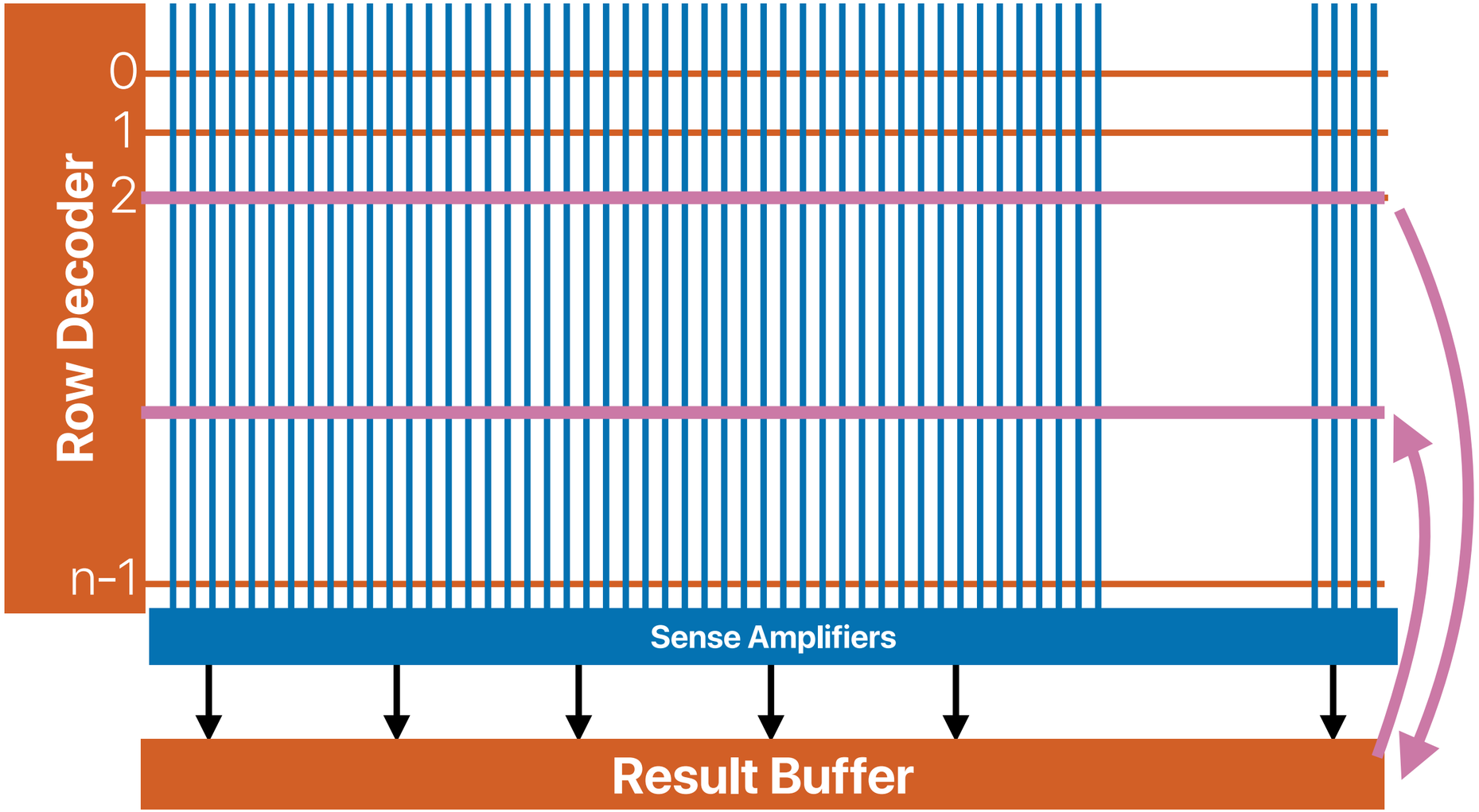


Active Pages



M. Oskin, F. Chong and T. Sherwood, "Active Pages: A Computation Model for Intelligent Memory," in Computer Architecture, International Symposium on, Barcelona, Spain, 1998

Recap: Or we just clone/move?

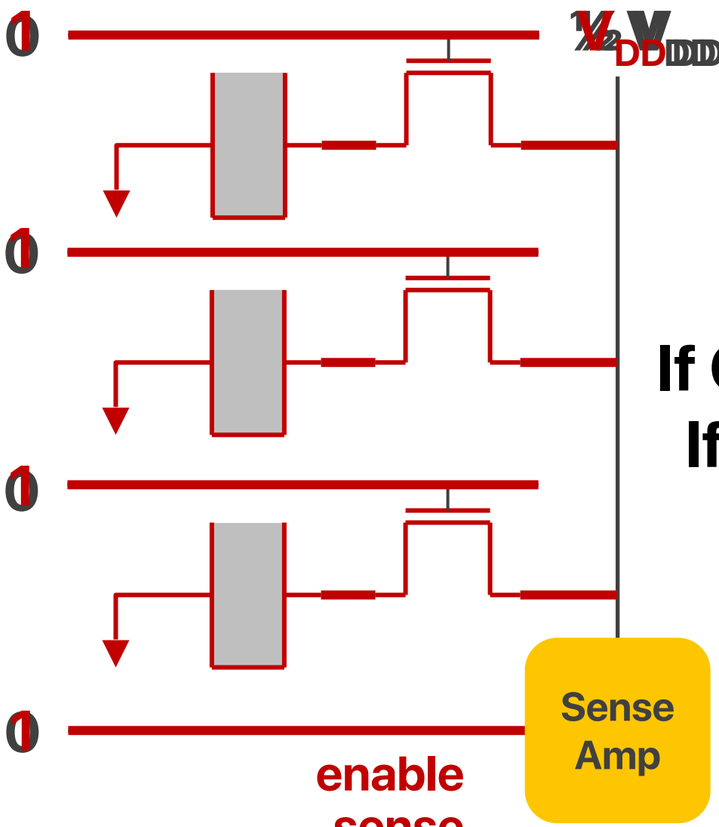


memmove & memcpy: 5% cycles in Google's datacenter

Svilen Kanev, Juan Pablo Darago, Kim Hazelwood, Parthasarathy Ranganathan, Tipp Moseley, Gu-Yeon Wei, and David Brooks.Profiling a warehouse-scale computer ISCA '15

Vivek Seshadri, Yoongu Kim, Chris Fallin, Donghyuk Lee, Rachata Ausavarungnirun, Gennady Pekhimenko, Yixin Luo, Onur Mutlu, Phillip B. Gibbons, Michael A. Kozuch, and Todd C. Mowry. RowClone: fast and energy-efficient in-DRAM bulk data copy and initialization. In MICRO-46.

Recap: Activate Three Rows



AB + BC + AC =
C(A+B) + C′AB

If C is 0, we can compute AND
If C is 1, we can compute OR

Input			Output
A	B	C	
0	0	0	**0**
0	1	0	**0**
1	0	0	**0**
1	1	0	**1**
0	0	1	**0**
0	1	1	**1**
1	0	1	**1**
1	1	1	**1**

		A′B′	A′B	AB	AB′
	Out(A, B)	0,0	0,1	1,1	1,0
C′	0	0	0	1	0
C	1	0	1	1	1

BC AB AC

Recap: Ambit

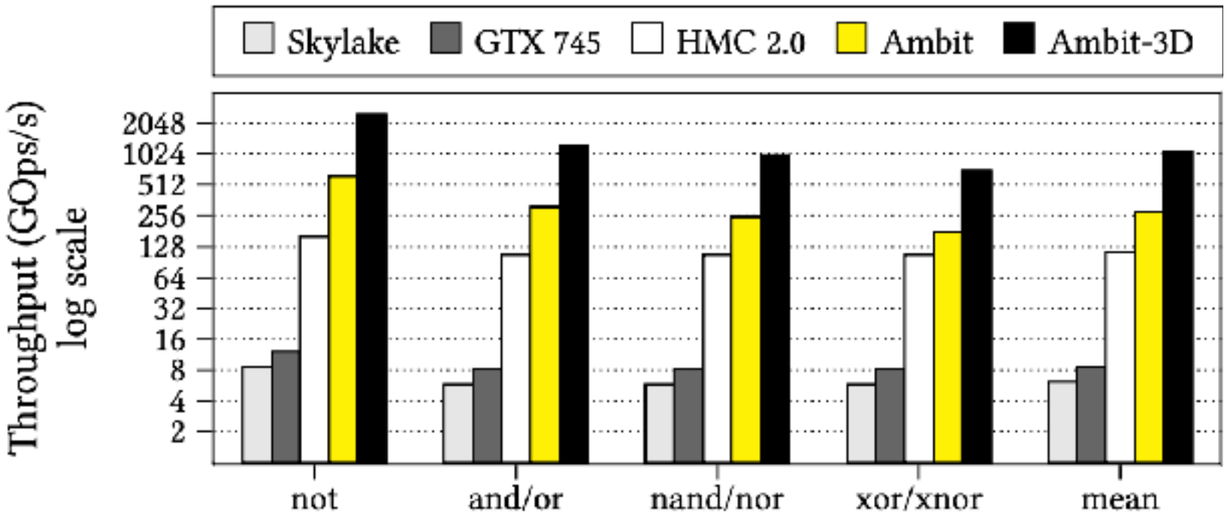


Figure 9: Throughput of bulk bitwise operations.

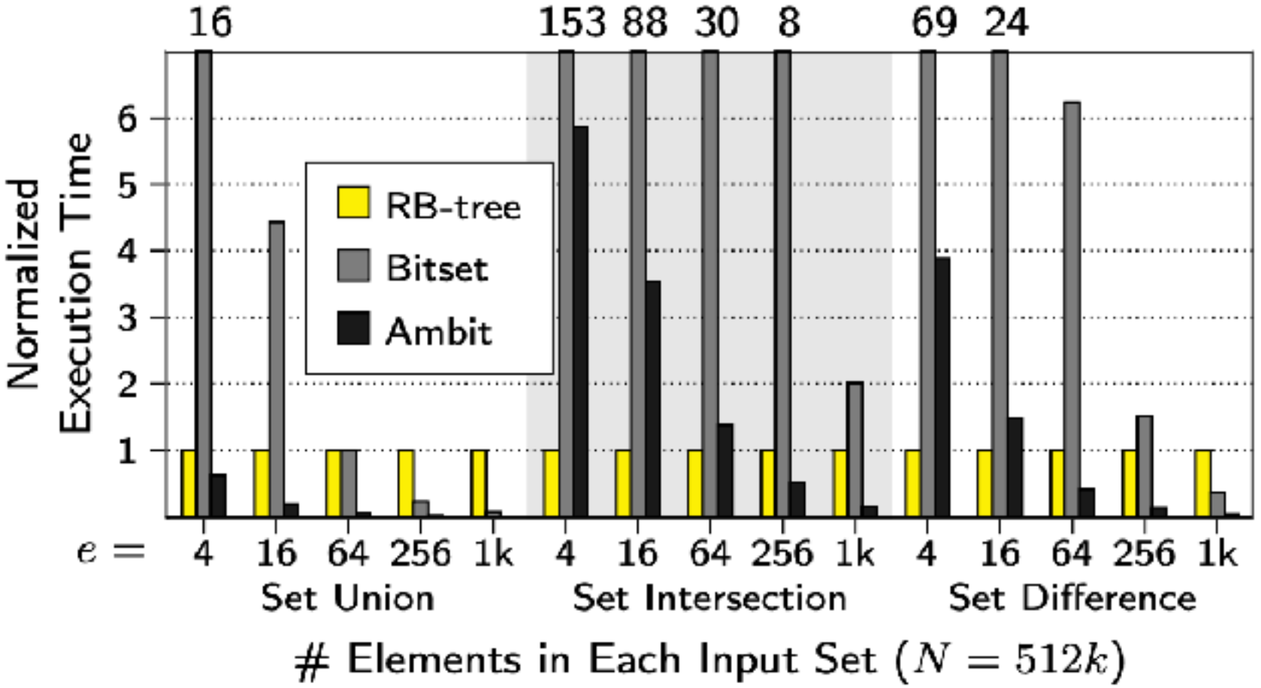


Figure 12: Performance of set operations

Vivek Seshadri, Donghyuk Lee, Thomas Mullins, Hasan Hassan, Amirali Boroumand, Jeremie Kim, Michael A. Kozuch, Onur Mutlu, Phillip B. Gibbons, and Todd C. Mowry. Ambit: in-memory accelerator for bulk bitwise operations using commodity DRAM technology. MICRO-50.

Limitations of in-DRAM processing

- Programming
- Data mapping — it's effective only if data happen to be within the same array/module
- Hardware design — not that many operations are available

Processing Using Memory

Charge-based v.s. resistive memory

- Charge-based memory (e.g., SRAM, DRAM, Flash)
  - Write data by capturing the charge
    - DRAM: capacitor — leakage
    - Flash: floating-gate — wear-out quickly
  - Read data by measuring the voltage level
- Resistive memory (e.g., PCM, MRAM, RRAM)
  - Write data by change the "material" properties
    - PCM: change material phase
    - STT: change magnet polarity
    - RRAM: change atomic structure/atom distance
  - Read data by measuring the resistance

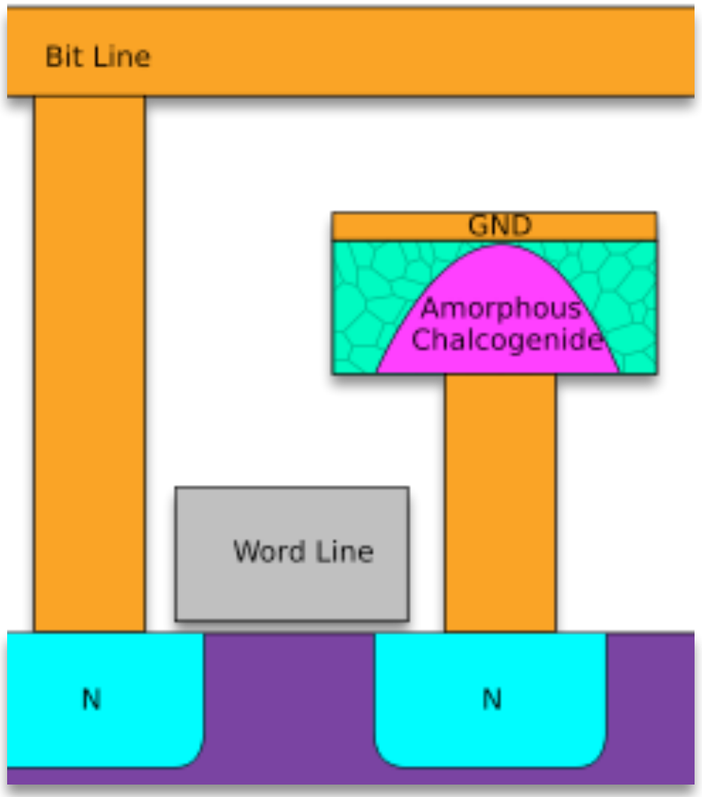
$$V_c = V_s \times e^{\frac{-t}{RC}}$$

Ohm's law

$$V = I \times R$$

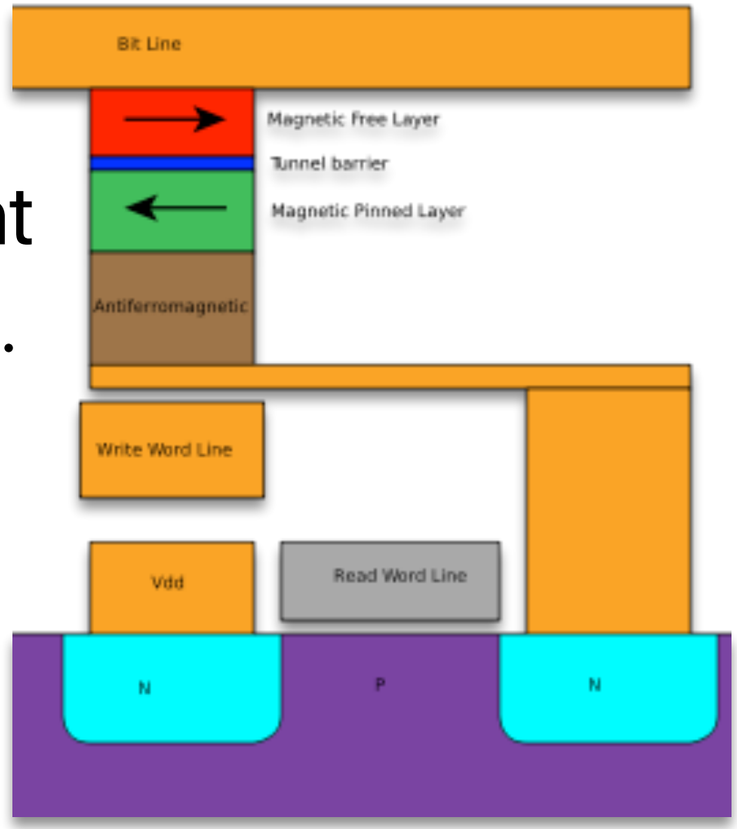
Phase change memory

- The bit is stored in the crystal structure of a tiny spec of metal.
- To write, it melts the metal (650C)
  - let it cool quickly or slowly to set the value
  - Crystaline and amorphous states have different resistance
    - Amorphous: Low optical reflexivity and high electrical resistivity
    - Crystalline: High optical reflexivity and low electrical resistivity



Spin-torque transfer

- Bits stored as magnetic orientation of a thin film
- Change the state using polarized electrons (!)
- Depending on polarization, resistance differs
- More complex cell structure
- Great promise — potential DRAM replacement
  - Roughly the same speed, power, and bandwidth.
  - But it's durable!



“In”-NVM processing

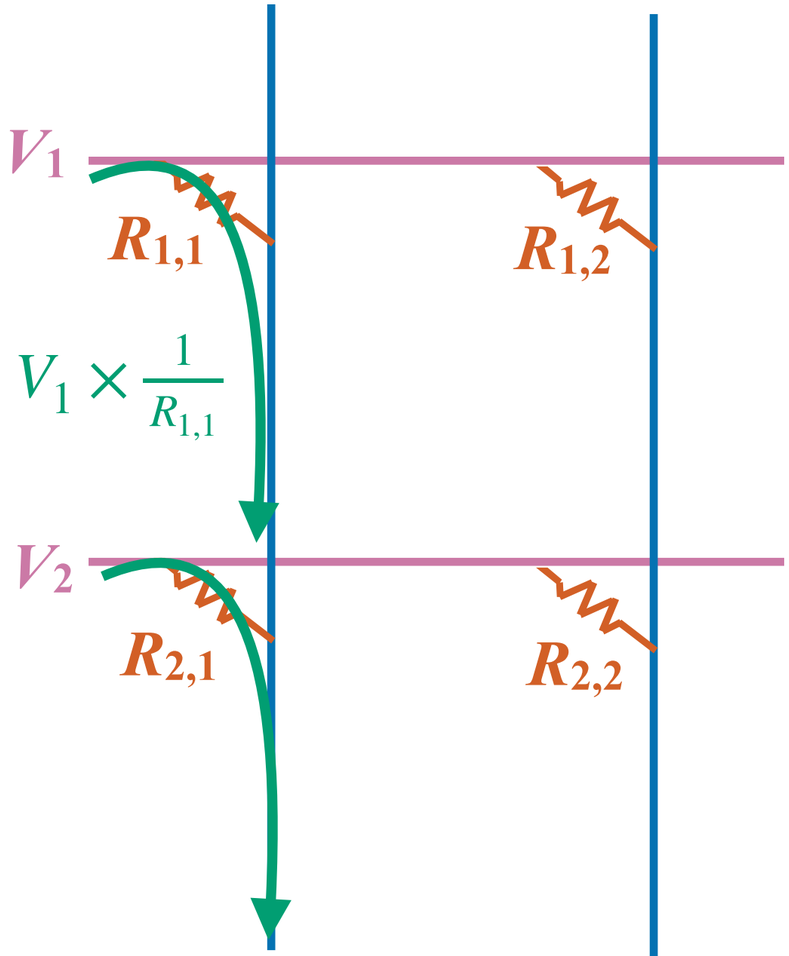
Performance of ReRAM

TABLE III. EFFECTIVE READ LATENCY FOR DIFFERENT R_{sense}.

R_{sense}	T_{sense}	P(retry)	Effective T_{rd} (incl T_{PRE}, T_{BUS})
10KΩ	69 ns	0.0%	80 ns
8KΩ	55 ns	0.0%	66 ns
7.5KΩ	51 ns	0.0%	62 ns
7KΩ	**48 ns**	**0.5%**	**60 ns**
6.5KΩ	44 ns	40%	90 ns

P. J. Nair, C. Chou, B. Rajendran and M. K. Qureshi. Reducing read latency of phase change memory via early read and Turbo Read. 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)

ReRAM



$$I_{1,1} = V_1 \times \frac{1}{R_{1,1}}$$

$$I_1 = V_1 \times \frac{1}{R_{1,1}} + V_2 \times \frac{1}{R_{2,1}} \qquad I_2 = V_1 \times \frac{1}{R_{2,1}} + V_2 \times \frac{1}{R_{2,2}}$$

$$\begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix} \times \begin{bmatrix} R_{1,1}^{-1} & R_{1,2}^{-1} & R_{1,3}^{-1} & R_{1,4}^{-1} \\ R_{2,1}^{-1} & R_{2,2}^{-1} & R_{2,3}^{-1} & R_{2,4}^{-1} \\ R_{3,1}^{-1} & R_{3,2}^{-1} & R_{3,3}^{-1} & R_{3,4}^{-1} \\ R_{4,1}^{-1} & R_{4,2}^{-1} & R_{4,3}^{-1} & R_{4,4}^{-1} \end{bmatrix}$$

3:00

How fast is a ReRAM based accelerator?

How fast is a ReRAM accelerator?

Performance of ReRAM

TABLE III. EFFECTIVE READ LATENCY FOR DIFFERENT R_{sense}.

R_{sense}	T_{sense}	P(retry)	Effective T_{rd} (incl T_{PRE}, T_{BUS})
10KΩ	69 ns	0.0%	80 ns
8KΩ	55 ns	0.0%	66 ns
7.5KΩ	51 ns	0.0%	62 ns
7KΩ	**48 ns**	**0.5%**	**60 ns**
6.5KΩ	44 ns	40%	90 ns

If each ReRAM's array dimension is 4K — the total OPs the ReRAM can perform per cycle is $4096 \times 4096 \times 2 = 8380416$

The throughput is $\frac{8380416}{60ns} = 139TOPS$

P. J. Nair, C. Chou, B. Rajendran and M. K. Qureshi. Reducing read latency of phase change memory via early read and Turbo Read. 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)

3:00

What's the limitation of ReRAM-based matrix multiplier? What applications can tolerate these limitations?

Limitations of ReRAM-based accelerator

Limitations of ReRAM-based accelerator

- Limited Precision
- A/D and D/A Conversion
  - Area and power increases exponentially with ADC resolution and frequency
  - Large area, Power hungry e.g., 98% of the total area and 89% of the total power
- Array Size and Routing
  - Wire dominates energy for array size of 1k × 1k
  - IR drop along wire can degrade read accuracy
- Write/programming energy
  - Multiple pulses can be costly
- Variations & Yield
  - Device-to-device, cycle-to-cycle
  - Non-linear conductance across range

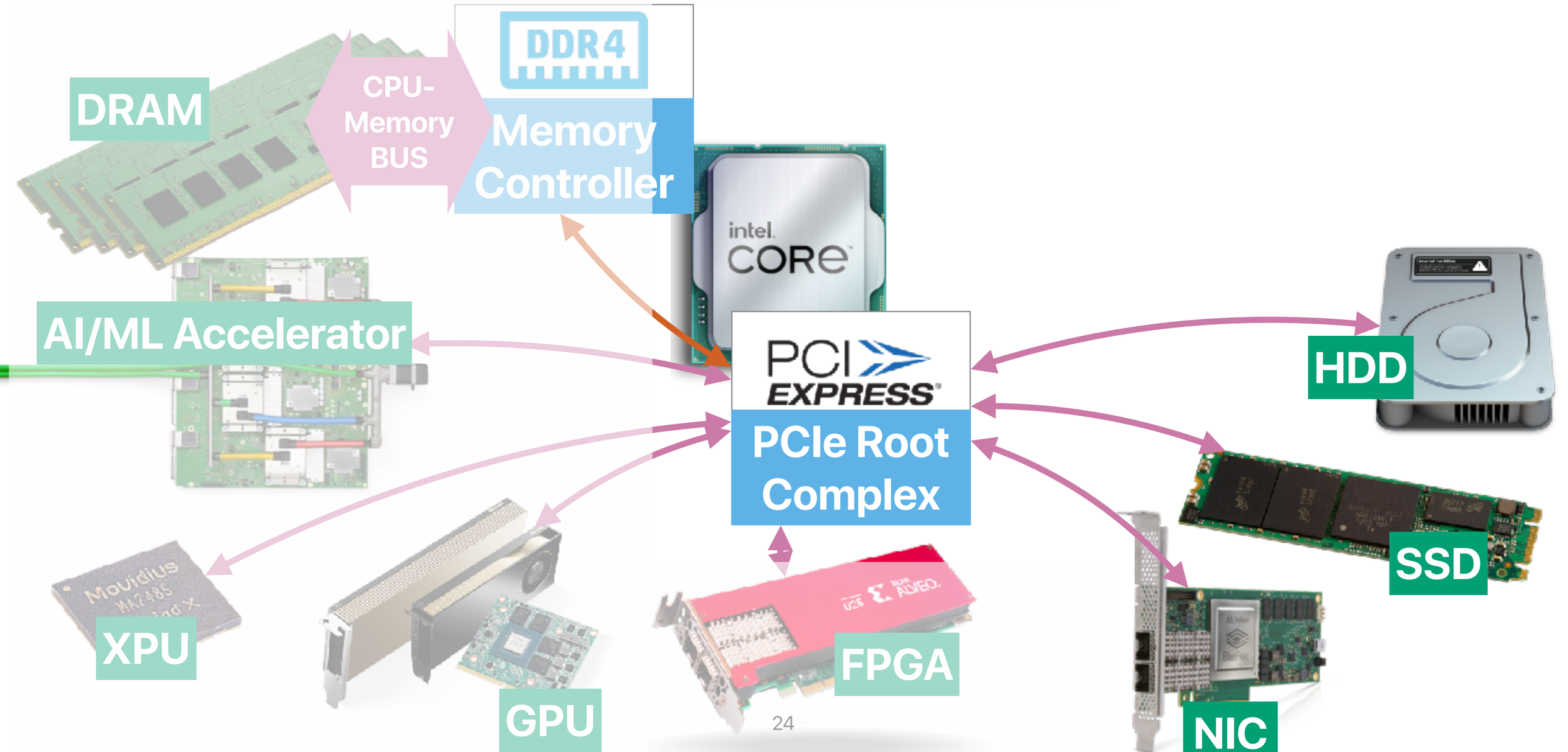
References

- ADC/DAC optimizations
  - H. Yun, H. Shin, M. Kang and L. -S. Kim, "Optimizing ADC Utilization through Value-Aware Bypass in ReRAM-based DNN Accelerator," 2021 58th ACM/IEEE Design Automation Conference (DAC), 2021, pp. 1087-1092, doi: 10.1109/DAC18074.2021.9586140.
  - Qilin Zheng, Zongwei Wang, Zishun Feng, Bonan Yan, Yimao Cai, Ru Huang, Yiran Chen, Chia-Lin Yang, and Hai (Helen) Li. 2020. Lattice: an ADC/DAC-less ReRAM-based processing-in-memory architecture for accelerating deep convolution neural networks. In Proceedings of the 57th ACM/EDAC/IEEE Design Automation Conference (DAC '20). IEEE Press, Article 190, 1–6.
- Digital PIM
  - Mohsen Imani, Saransh Gupta, Yeseong Kim, and Tajana Rosing. 2019. FloatPIM: in-memory acceleration of deep neural network training with high precision. In Proceedings of the 46th International Symposium on Computer Architecture (ISCA '19). Association for Computing Machinery, New York, NY, USA, 802–815. https://doi.org/10.1145/3307650.3322237

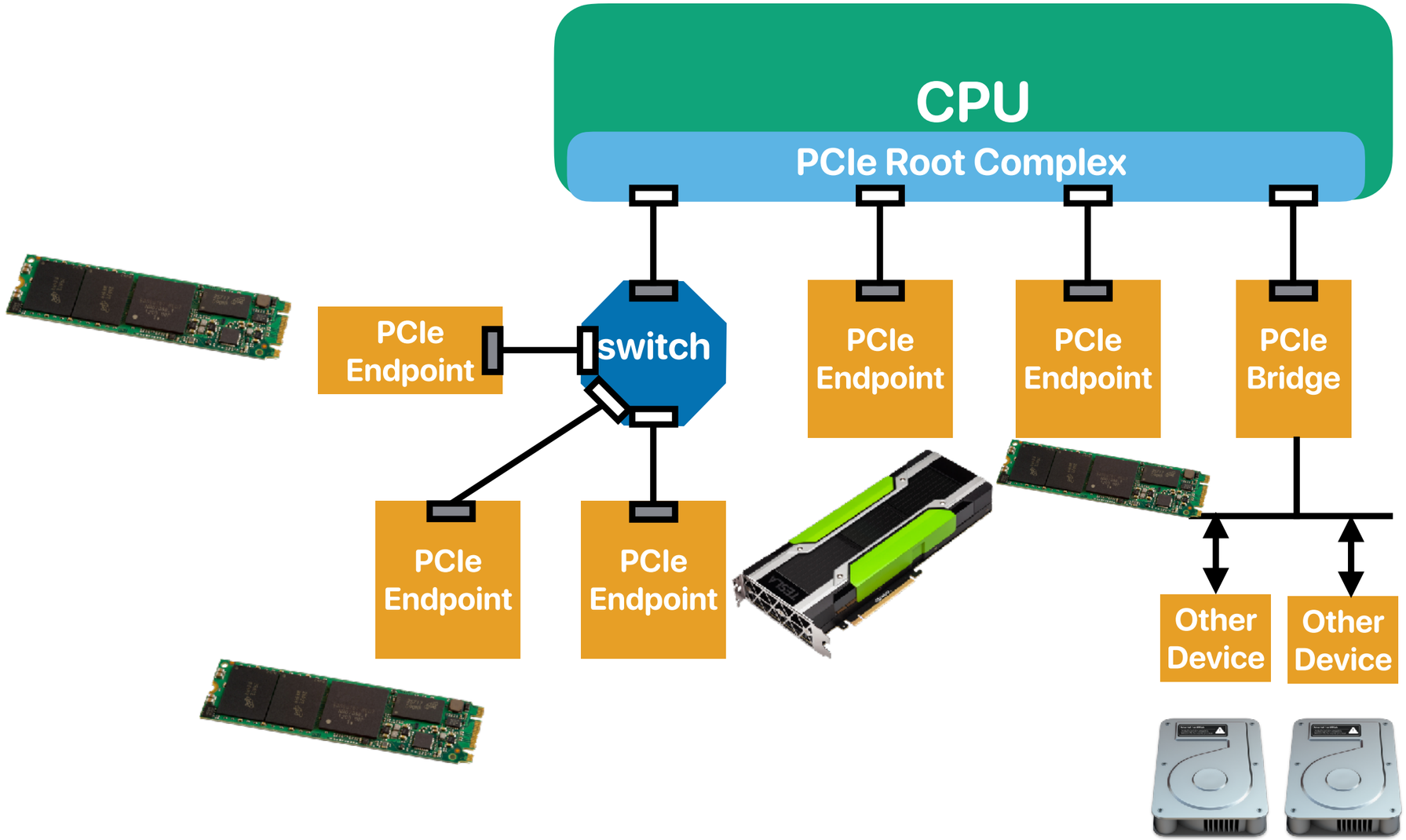
Applications of ReRAM-based accelerator

- NN
  - Ali Shafiee, Anirban Nag, Naveen Muralimanohar, Rajeev Balasubramonian, John Paul Strachan, Miao Hu, R. Stanley Williams, and Vivek Srikumar. 2016. ISAAC: a convolutional neural network accelerator with in-situ analog arithmetic in crossbars. In ISCA '16. 2016
  - Ping Chi, Shuangchen Li, Cong Xu, Tao Zhang, Jishen Zhao, Yongpan Liu, Yu Wang, and Yuan Xie. PRIME: a novel processing-in-memory architecture for neural network computation in ReRAM-based main memory. In ISCA '16. 2016
  - Song, Linghao, Xuehai Qian, Hai Li, and Yiran Chen. Pipelayer: A pipelined reram-based accelerator for deep learning. In HPCA 2017.
  - Mohsen Imani, Saransh Gupta, Yeseong Kim, and Tajana Rosing. 2019. FloatPIM: in-memory acceleration of deep neural network training with high precision. ISCA. 2019.

Recap: The landscape of modern computers



PCIe "Interconnect"

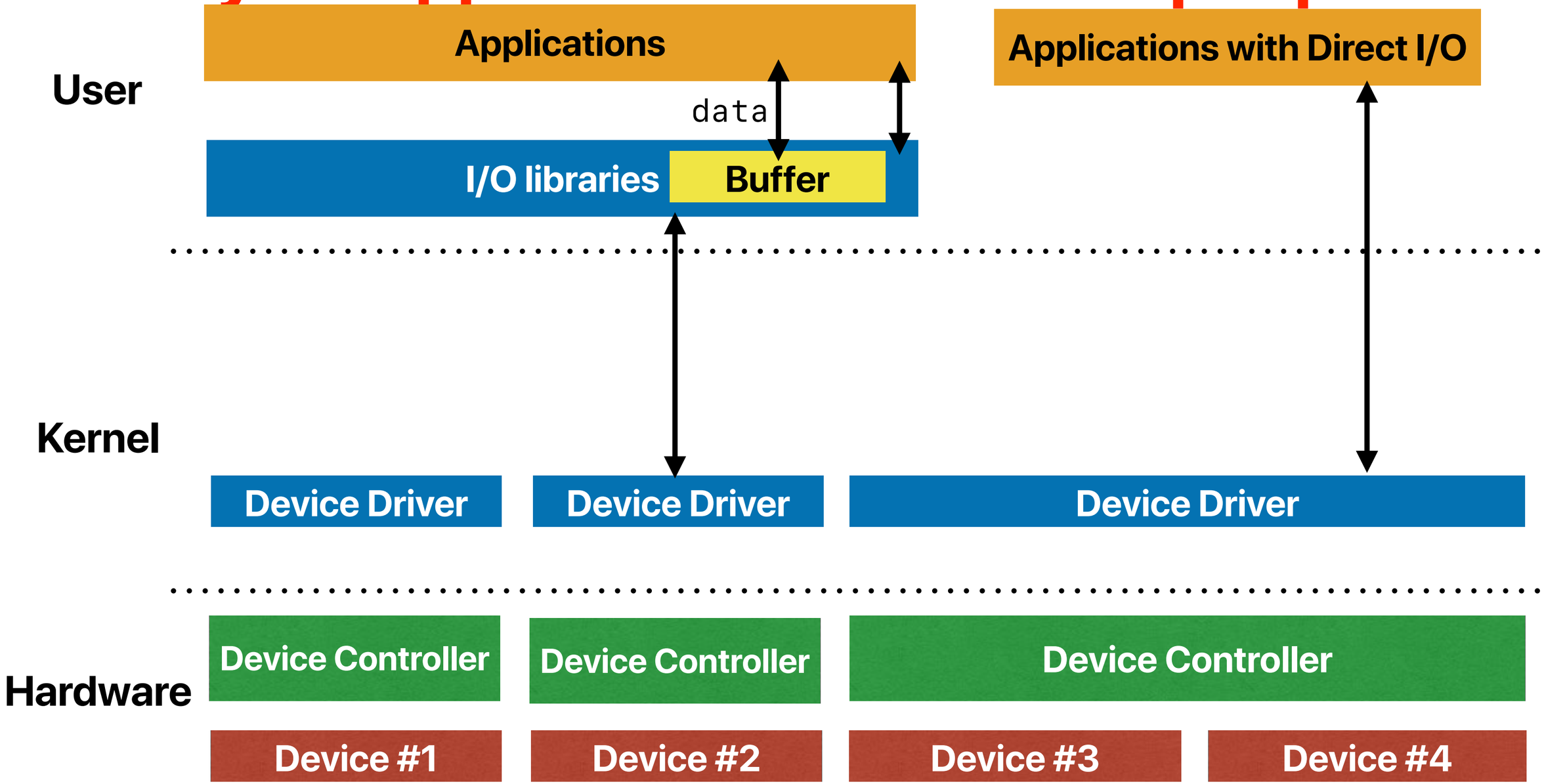


3:00

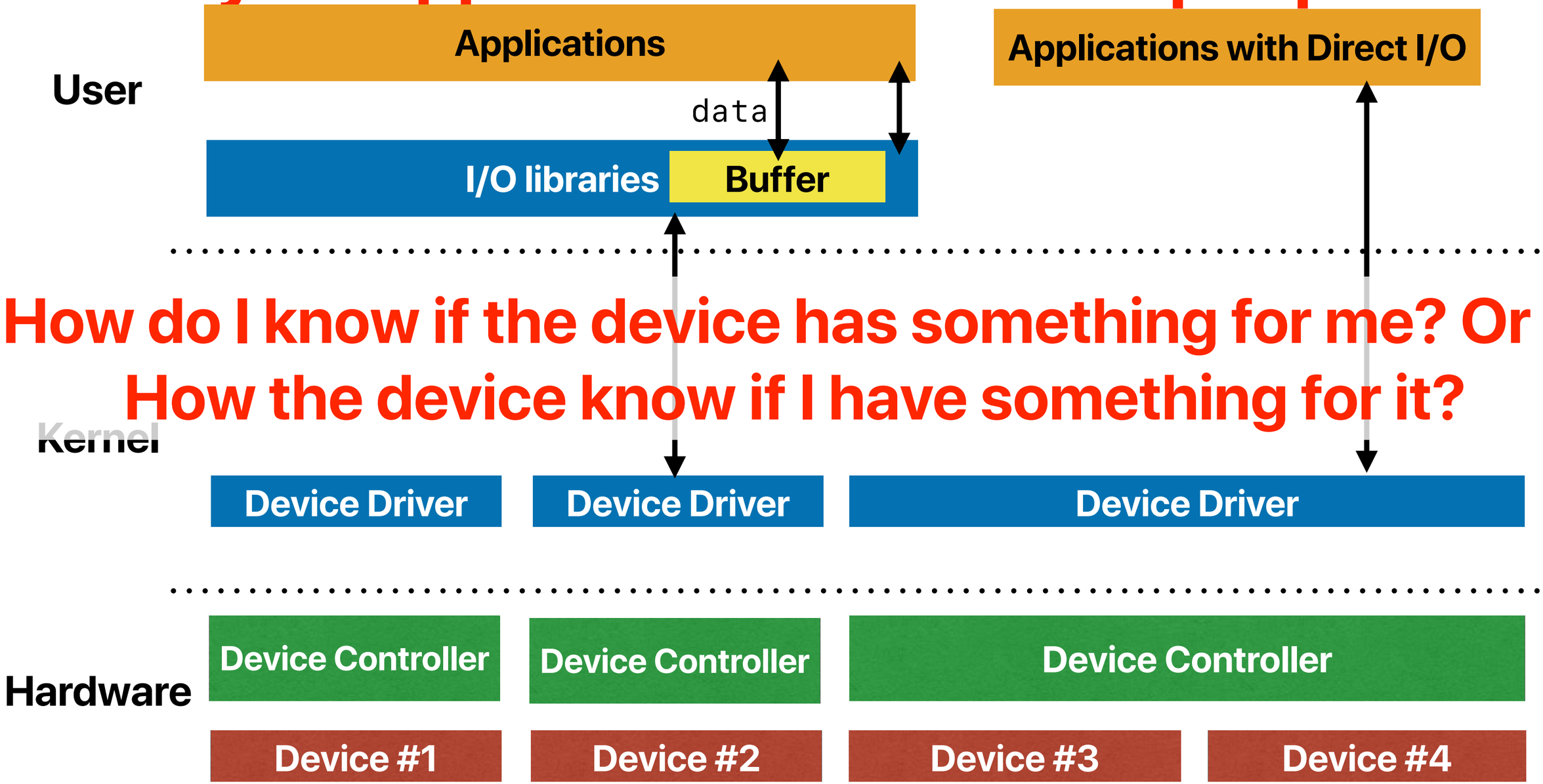
How does your program talk to SSDs, HDDs & NICs?

How does your program talk to devices?

How your application interact with peripherals



How your application interact with peripherals

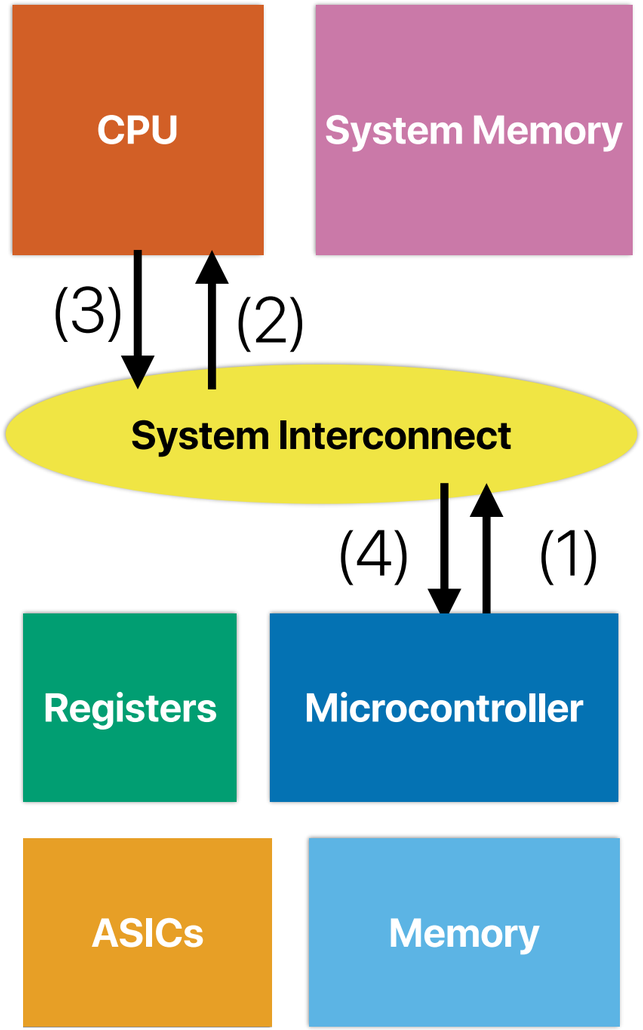


How your CPU/OS interact with I/O devices

- The mechanism of knowing if there is an event
  - Interrupt
  - Polling
- The mechanism of handling the request
  - PIO
  - DMA

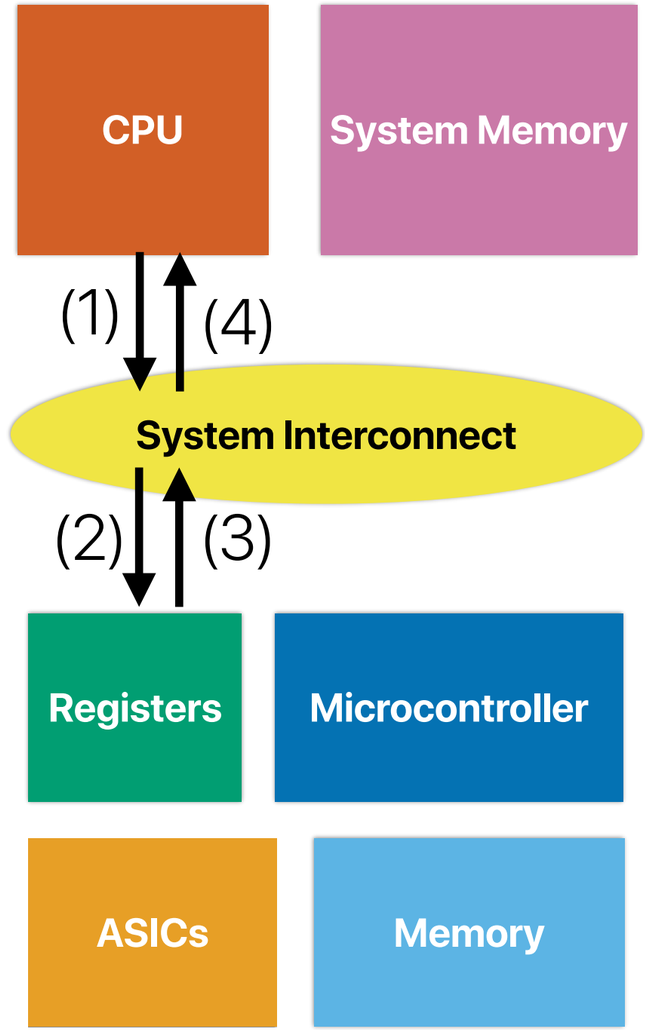
Interrupt

- The device signals the processor only when the device requires the processor/OS handle some tasks/data
- The processor only signals the device when necessary



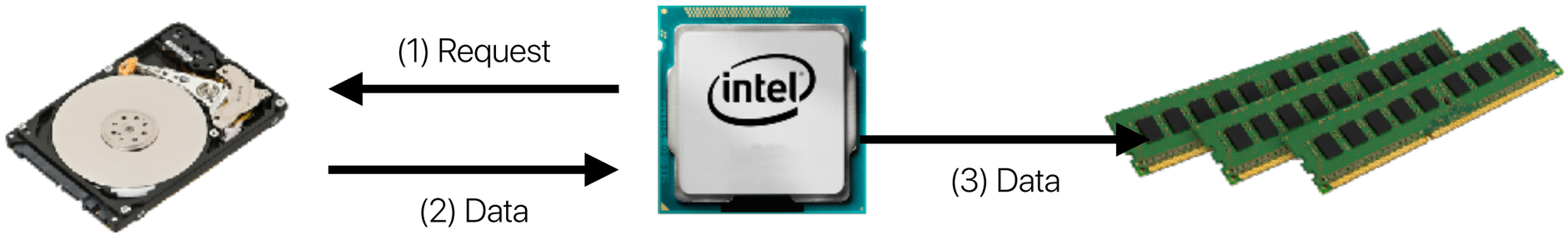
Polling

- The processor/OS constantly asks if the device (e.g. examine the status register of the device) is ready to or requires the processor/OS handle some tasks/data
- The OS/processor executes corresponding handler if the device can handle demand tasks/data or has tasks/data ready



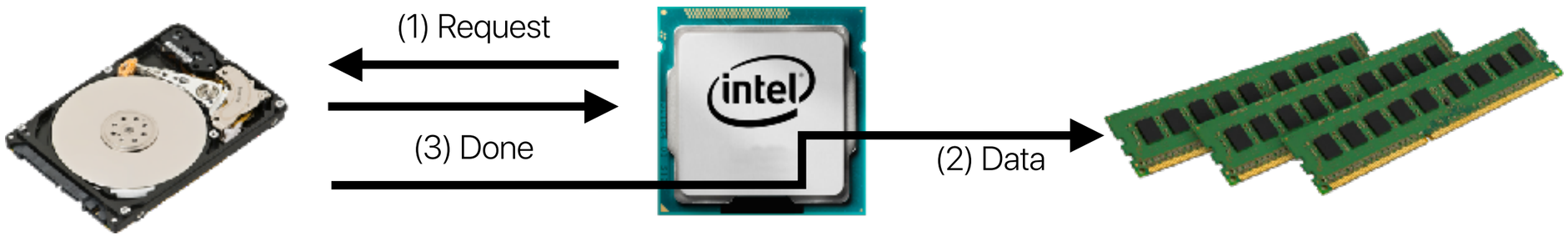
Programmed I/O (PIO)

- The processor/OS executes code and issues commands to handle the data movements between the peripheral and the main memory
  - For example, the in/out instruction in x86



Direct Memory Access (DMA)

- The OS allocates a buffer to exchange data with the peripheral
- The peripheral device can directly access the allocated buffer



3:00

Which model is the best?

Electrical
Computer Science
Engineering

277

つづく