Connecting Deep Neural Networks

NoCS 203

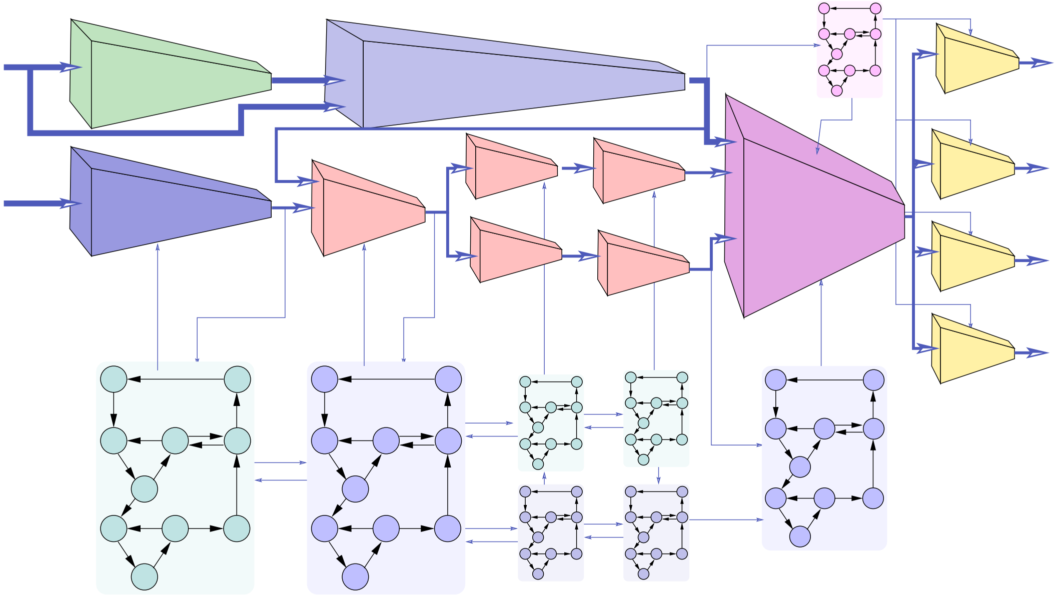
Hamburg, Germany

Axel Jantsch

September 21, 2023

Institut für Computertechnik
Institute of Computer Technology

DNN based Applications



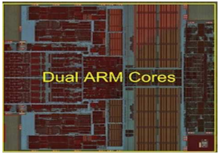
Outline

Trends in Applications, Architecture and Technology

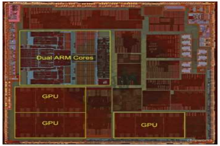
Trends

- Chiplets
- Customization
- DNN based applications

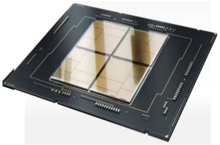
Packaging Hierarchy



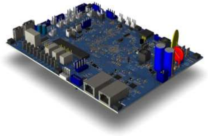
Block, Core



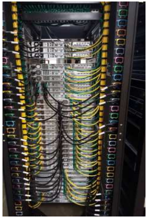
Die



Package

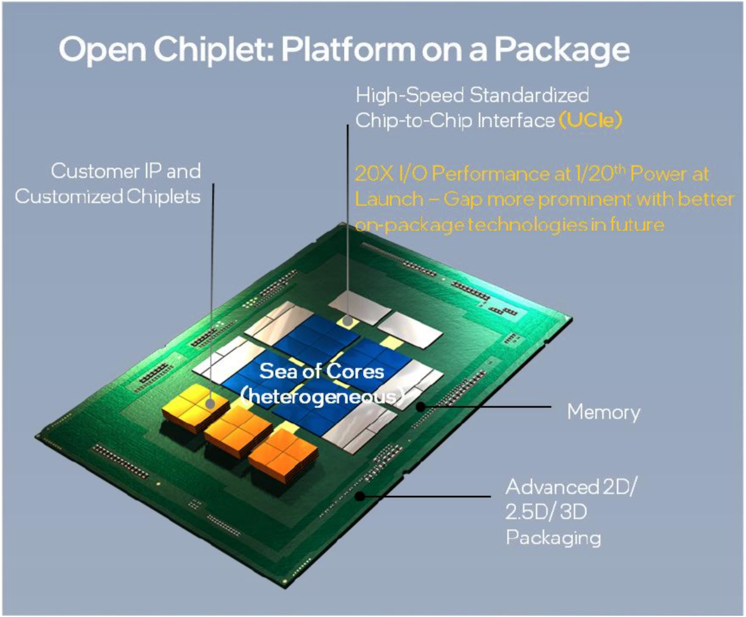


Board



Rack

Chiplets: On-Package Integration



D. Das Sharma. *Universal Chiplet Interconnect express (UCIe): Building an open chiplet ecosystem*. Technical report. White Paper. by UCIe Consortium, 2022

UCIe Performance

	B/W (GB/s/mm)	Energy (pJ/b)	Latency (ns)
PCIe	60	10	15
UCIe	1300	0.25	1
on-chip	50 000	0.01	0.5

Customization

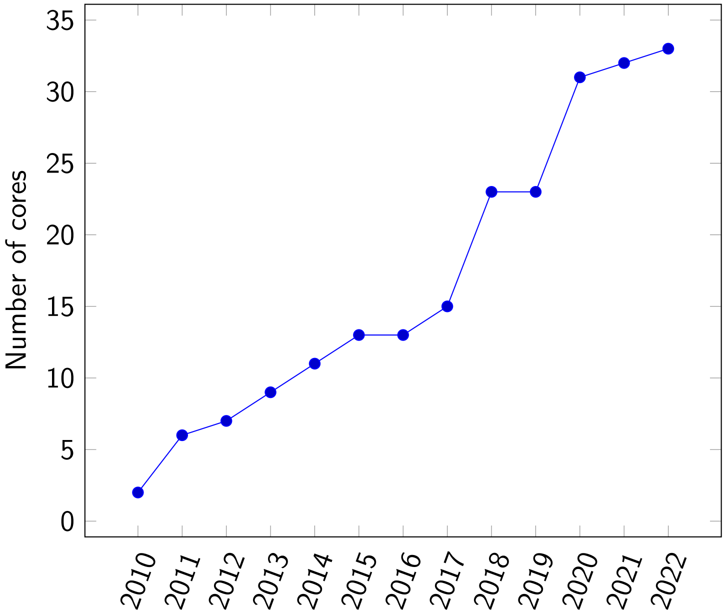
- Custom technology: compute, memory
- Custom architecture: cores, accelerators
- Custom algorithm design: for GPUs, NPUs, ...

Customization of Architecture

Name	Year	Node	CPU	No	GPU	No	ISP	Video Encoder	Audio	Security	Motion Processor	NN	No	Display
A4	2010	45nm	CortexA8	1	PowerVR	1								
A5	2011	45nm	CortexA9	2	PowerVR	2	1		EarSmart					
A6	2012	32nm	ARMv7-A	2	PowerVR	3	1		EarSmart					
A7	2013	28nm	ARMv8-A	2	PowerVR	4	1			Secure Enclave				
A8	2014	20nm	Cyclone	2	PowerVC	4	1	1						
A9	2015	16nm	ArmV8-A	2	PowerVR	6	1	1			M9			
A10	2016	16nm	ArmV8-A	2	PowerVR	6	1	1			M10			
A11	2017	10nm	ArmV8-A	6	GPU	3	1	1			M11	Neural Engine		
A12	2018	7nm	ArmV8.3-A	6	GPU	4		1				NE	8	
A13	2019	7nm	ArmV8.4-A	6	GPU	4		1				NE	8	
A14	2020	5nm	ArmV8.5-A	6	GPU	4		1				NE	16	
A15	2021	5nm	ArmV8	6	GPU	5	1					NE	16	
A16	2022	5nm	ArmV8.6-A	6	GPU	5	1					NE	16	1

Apple Axx SoCs

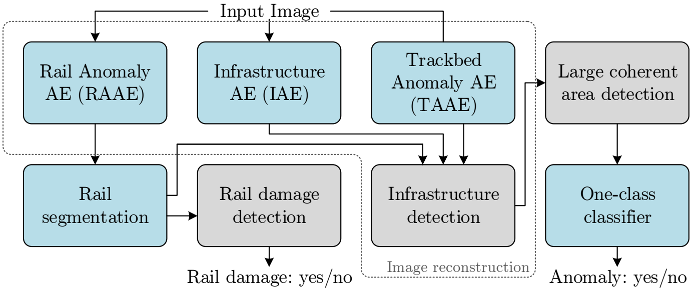
Number of Cores in Axx



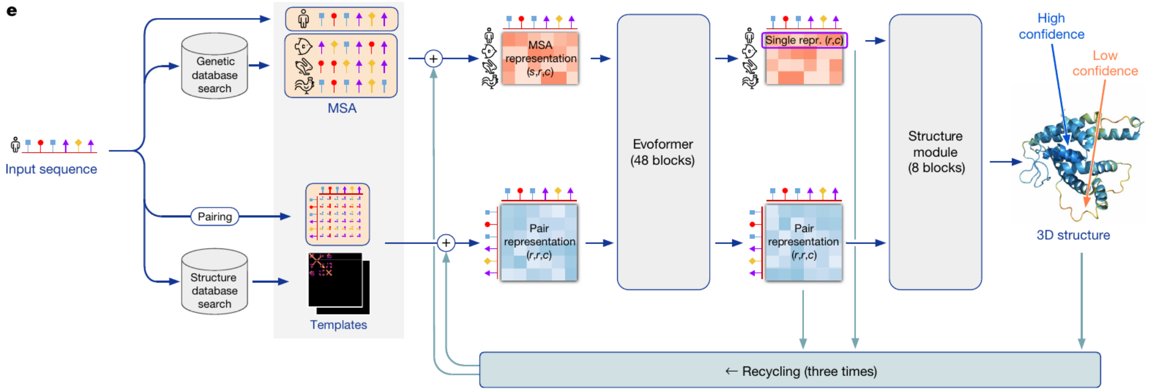
DNN based Applications

- DNN based features appear in many applications
- Many different DNN types, sizes, and architectures

DNN based Application: VADAR



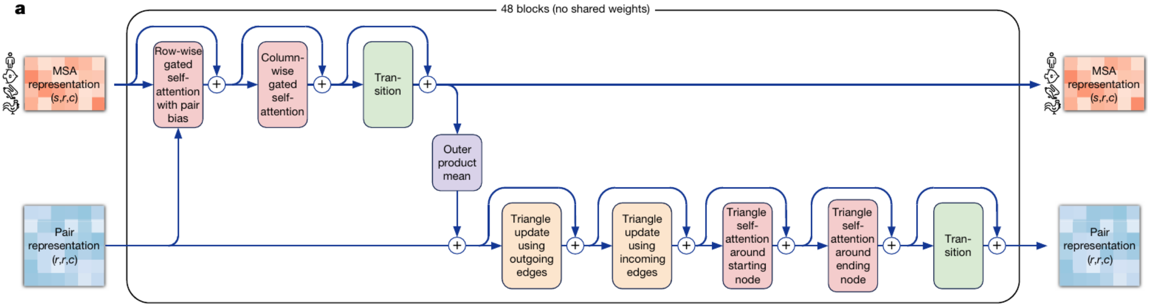
David Breuss, Maximilian Götzinger, Jenny Vuong, Clemens Reisner, and Axel Jantsch. "VADAR: A Vision-based Anomaly Detection Algorithm for Railroads". In: *Proceedings of the 26th Euromicro Conference on Digital System Design (DSD)*. Durres, Albania, Sept. 2023



AlphaFold model architecture

John Jumper et al. “Highly accurate protein structure prediction with AlphaFold”. In: *Nature* 596.7873 (Aug. 2021), pages 583–589

DNN based Application: AlphaFold



Evoformer block

Where will these trends lead?

- Chiplets
- Customization
- DNN based applications

Where will these trends lead?

- Chiplets $\rightarrow$ **Highly flexible super integration**
- Customization
- DNN based applications

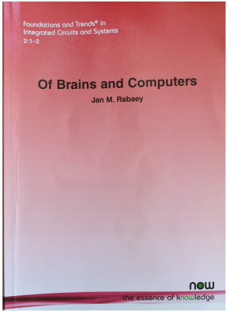
Where will these trends lead?

- Chiplets $\rightarrow$ **Highly flexible super integration**
- Customization $\rightarrow$ **Explosion of the design space**
- DNN based applications

Where will these trends lead?

- Chiplets → **Highly flexible super integration**
- Customization → **Explosion of the design space**
- DNN based applications → **Third digital wave in the digital revolution**

Rabaey's Design Principles

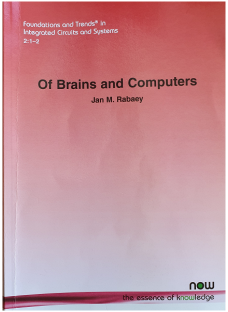


Jan M. Rabaey. "Of Brains and Computers". In: *Foundations and Trends in Integrated Circuits and Systems* 2.1–2 (2022), pages 1–192

Peter Sterling and Simon Laughlin. *Principles of Neural Design*. MIT Press, June 2015

Rabaey's Design Principles

1. Compute with chemistry whenever possible

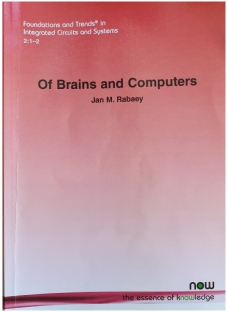


Jan M. Rabaey. "Of Brains and Computers". In: *Foundations and Trends in Integrated Circuits and Systems* 2.1–2 (2022), pages 1–192

Peter Sterling and Simon Laughlin. *Principles of Neural Design*. MIT Press, June 2015

Rabaey's Design Principles

1. Compute with chemistry whenever possible
2. Send only information that is needed; send it as slow as possible

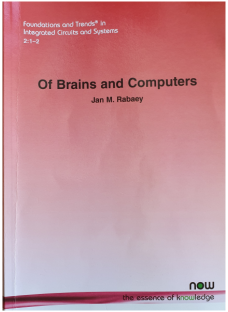


Jan M. Rabaey. "Of Brains and Computers". In: *Foundations and Trends in Integrated Circuits and Systems* 2.1–2 (2022), pages 1–192

Peter Sterling and Simon Laughlin. *Principles of Neural Design*. MIT Press, June 2015

Rabaey's Design Principles

1. Compute with chemistry whenever possible
2. Send only information that is needed; send it as slow as possible
3. Store information at the site where it is processed

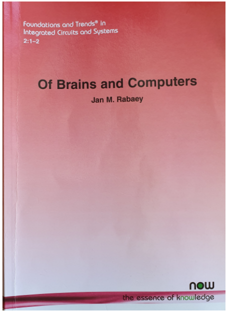


Jan M. Rabaey. "Of Brains and Computers". In: *Foundations and Trends in Integrated Circuits and Systems* 2.1–2 (2022), pages 1–192

Peter Sterling and Simon Laughlin. *Principles of Neural Design*. MIT Press, June 2015

Rabaey's Design Principles

1. Compute with chemistry whenever possible
2. Send only information that is needed; send it as slow as possible
3. Store information at the site where it is processed
4. Customize (or ... complicate)

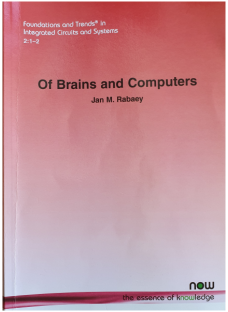


Jan M. Rabaey. "Of Brains and Computers". In: *Foundations and Trends in Integrated Circuits and Systems* 2.1–2 (2022), pages 1–192

Peter Sterling and Simon Laughlin. *Principles of Neural Design*. MIT Press, June 2015

Rabaey's Design Principles

1. Compute with chemistry whenever possible
2. Send only information that is needed; send it as slow as possible
3. Store information at the site where it is processed
4. Customize (or ... complicate)
5. Randomize

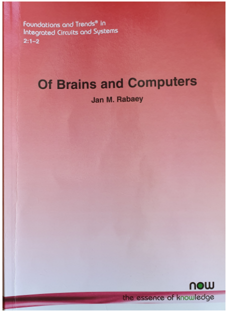


Jan M. Rabaey. "Of Brains and Computers". In: *Foundations and Trends in Integrated Circuits and Systems* 2.1–2 (2022), pages 1–192

Peter Sterling and Simon Laughlin. *Principles of Neural Design*. MIT Press, June 2015

Rabaey's Design Principles

1. Compute with chemistry whenever possible
2. Send only information that is needed; send it as slow as possible
3. Store information at the site where it is processed
4. Customize (or ... complicate)
5. Randomize
6. Self-calibrate, adapt and heal



Jan M. Rabaey. "Of Brains and Computers". In: *Foundations and Trends in Integrated Circuits and Systems* 2.1–2 (2022), pages 1–192

Peter Sterling and Simon Laughlin. *Principles of Neural Design*. MIT Press, June 2015

Outline

CUSTOMIZATION

Customization

1. **Compute with chemistry**
4. **Customize**

- Specific technology:
  - Logic, DRAM, AMS,
  - Resistive memory,
  - Phase change memory
  - Magnetoresistive memory
  - ...
- Algorithm specialization
  - DNN optimization
  - DNN customization
  - Heterogeneous DNNs

- Custom architectures:
  - DNN accelerators
  - Course grain reconfigurable computing
  - video encoders
  - audio processors
  - security engines
  - face recognition
  - language processing
  - goal management
  - ...

John L. Hennessy and David A. Patterson. "A New Golden Age for Computer Architecture". In: *Commun. ACM* 62.2 (Jan. 2019), pages 48–60

Customization

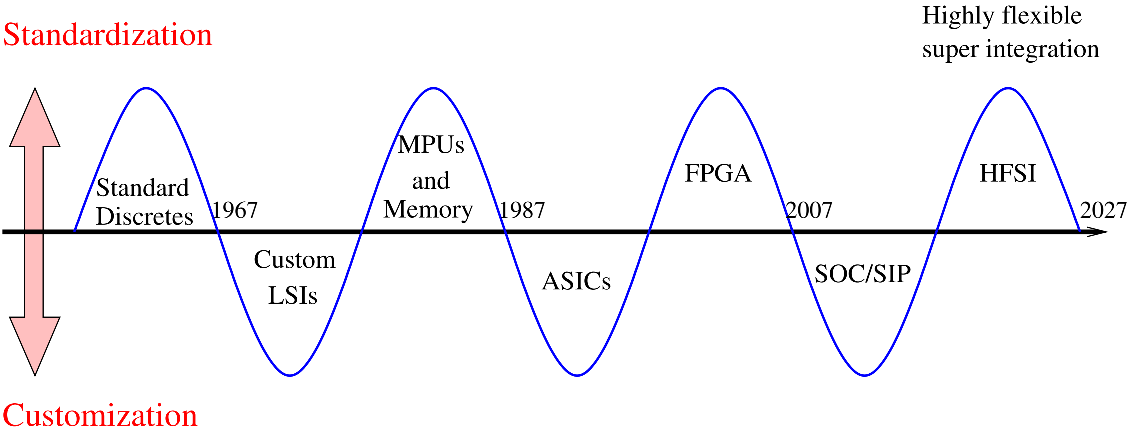
1. **Compute with chemistry**
4. **Customize**

- Specific technology:
  - Logic, DRAM, AMS,
  - Resistive memory,
  - Phase change memory
  - Magnetoresistive memory
  - …
- Algorithm specialization

- Custom architectures:
  - DNN accelerators
  - Course grain reconfigurable computing
  - video encoders
  - audio processors
  - security engines
  - face recognition

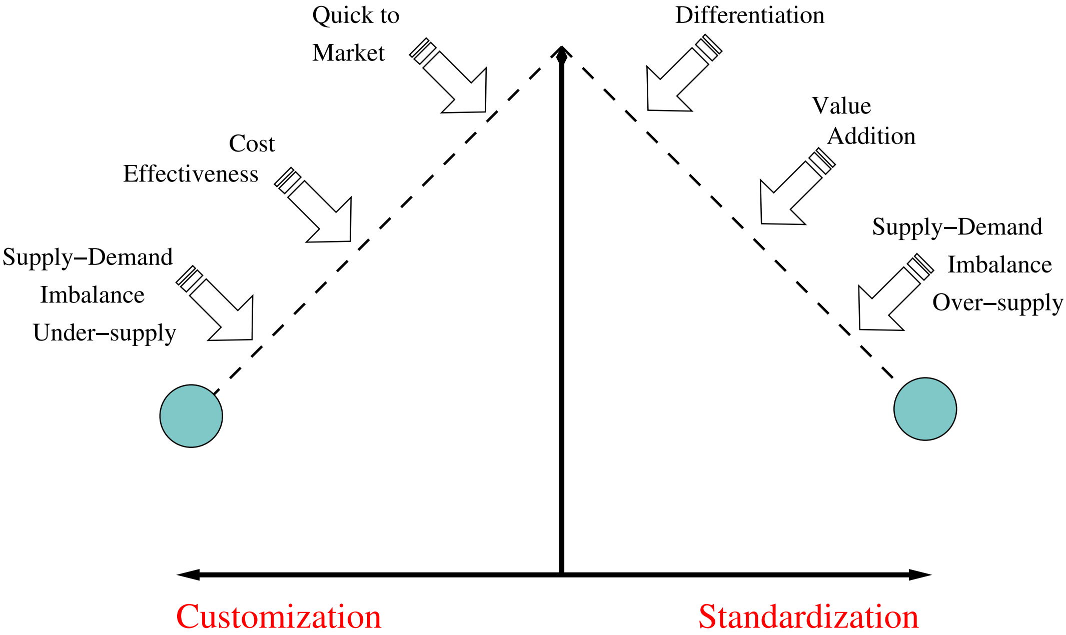
> *The next decade will see a Cambrian explosion of novel computer architectures, meaning exciting times for computer architects in academia and in industry.*

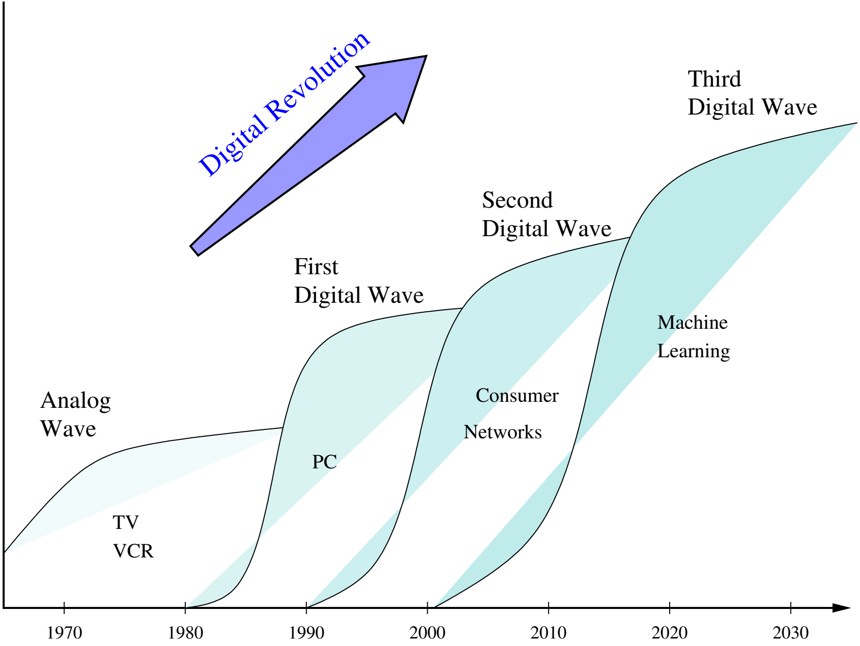
John L. Hennessy and David A. Patterson. "A New Golden Age for Computer Architecture". In: *Commun. ACM* 62.2 (Jan. 2019), pages 48–60



Makimoto's Wave. https://semiengineering.com/knowledge_centers/standards-laws/laws/makimotos-wave/. Accessed: 2023-09-12

Tsugio Makimoto. "The hot decade of field programmable technologies". In: *Proceedings of the IEEE International Conference on Field-Programmable Technology (FPT)*. 2002, pages 3–6


Quick to Market
Differentiation
Value Addition
Cost Effectiveness
Supply−Demand Imbalance Under−supply
Supply−Demand Imbalance Over−supply
Customization
Standardization



Digital Revolution
Third
Digital Wave
Second
Digital Wave
First
Digital Wave
Analog
Wave
Machine
Learning
Consumer
Networks
PC
TV
VCR
1970
1980
1990
2000
2010
2020
2030

Outline

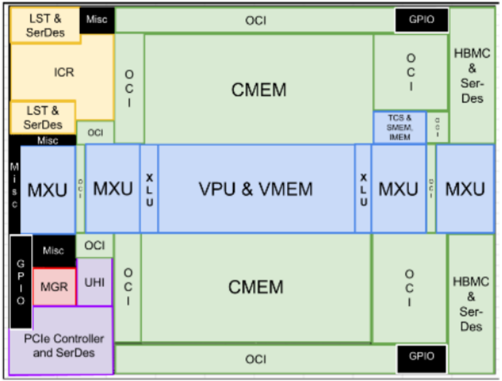
Minimize Communication

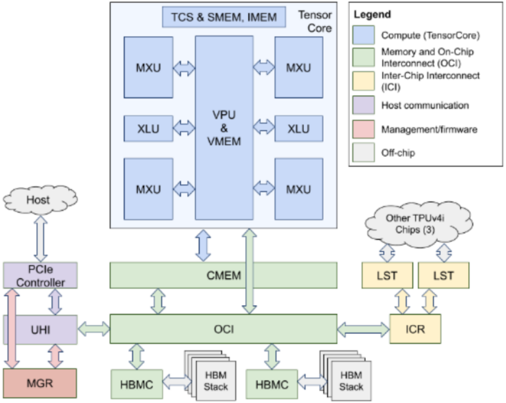
Minimize Communication

❷ Send only information that is needed; send it as slow as possible

- Near memory computing
- In-memory computing
- Lazy communication: transmit on demand only

Near Memory Computing

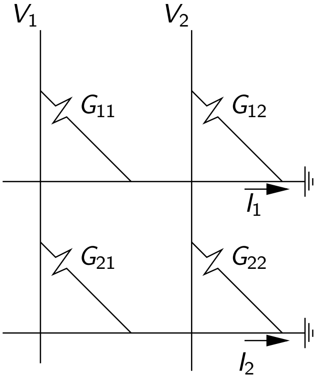




Google's Tensor Processing Unit TPUv4i

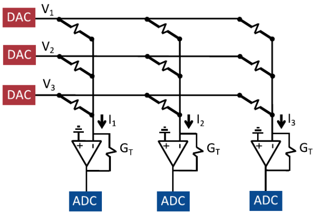
Norman P. Jouppi et al. "Ten Lessons From Three Generations Shaped Google's TPUv4i : Industrial Product". In: *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. 2021, pages 1–14

In Memory Computing

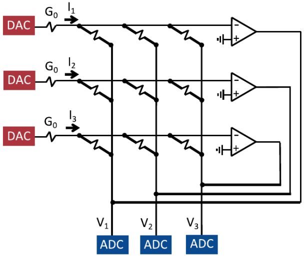


$$I_i = \sum_j G_{ij} V_j$$

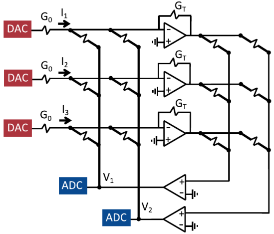
Ohm's law: $I = \frac{V}{R}$
Kirchhoffs law: $I_{\text{out}} = \sum_i I_i$
Conductance $G = \frac{1}{R}$



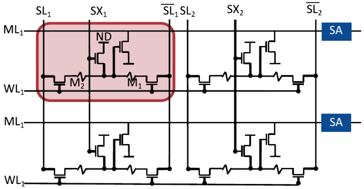
Matrix Vector Multiplication



Linear Equation ($Ax = b$)



Regression solver



Ternary Content Addressable Memory

Daniele Ielmini and Giacomo Pedretti. "Device and Circuit Architectures for In-Memory Computing". In: *Advanced Intelligent Systems* 2.7 (2020), page 2000040. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/aisy.202000040

Lazy Communication

Communicate as slow as possible

$$P \sim [\frac{1}{D}, \frac{1}{D^2}]$$

Lazy Communication

Communicate as slow as possible

$$P \sim [\frac{1}{D}, \frac{1}{D^2}]$$

$\Rightarrow$ doubling delay decreases power by 2x - 4x

Lazy Communication

Communication control

State of the Art

- Global, hard-wired and central control
- Application controlled: data request triggers communication
- Platform controlled: Power management, DVFS
- Assuming AFSP: As Fast and as Soon as Possible

Lazy Communication

Communication control

State of the Art

- Global, hard-wired and central control
- Application controlled: data request triggers communication
- Platform controlled: Power management, DVFS
- Assuming AFSP: As Fast and as Soon as Possible

Lazy Communication

- Default: ASLIP:
  - as Slow,
  - as Late,
  - as Inaccurate as Possible
- Application provides
  - deadline for data reception,
  - level of required approximation
- Network schedules packets based on the ASLIP principle

Outline

Imperfection

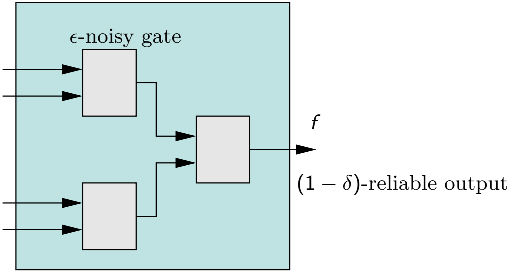
Imperfection

⑤ Randomize

- Perfection is expensive
- Allow for as much imperfection as can be tolerated
- Approximate computing is a broad trend

Cost of Perfection

An (ϵ, δ) circuit is a device that consists of ϵ-noisy gates and computes a function f with $(1 - \delta)$-reliability.

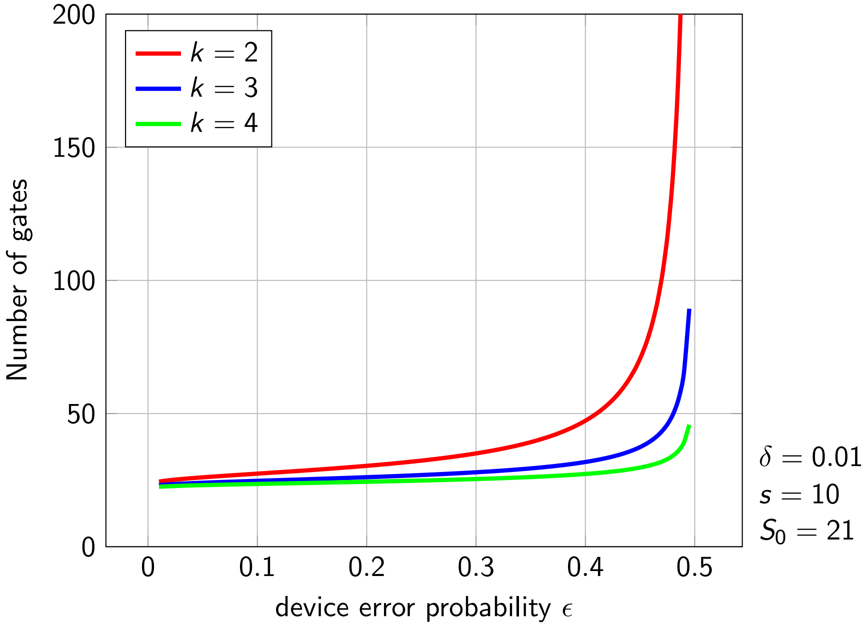


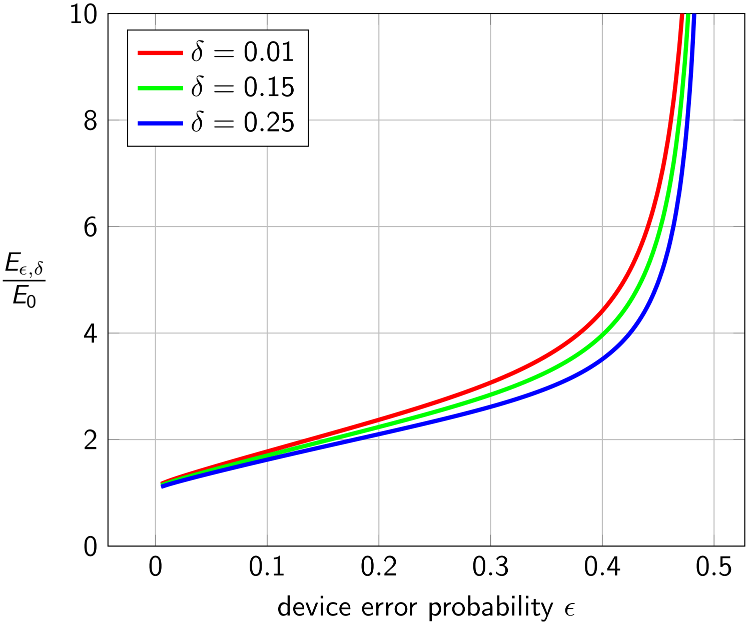
What is a lower bound of the costs, in terms of gates and energy?

Diana Marculescu. "Energy Bounds for Fault-Tolerant Nanoscale Designs". In: *Proceedings oif the of the Design, Automation and Test in Europe Conference and Exhibition (DATE)*. 2005

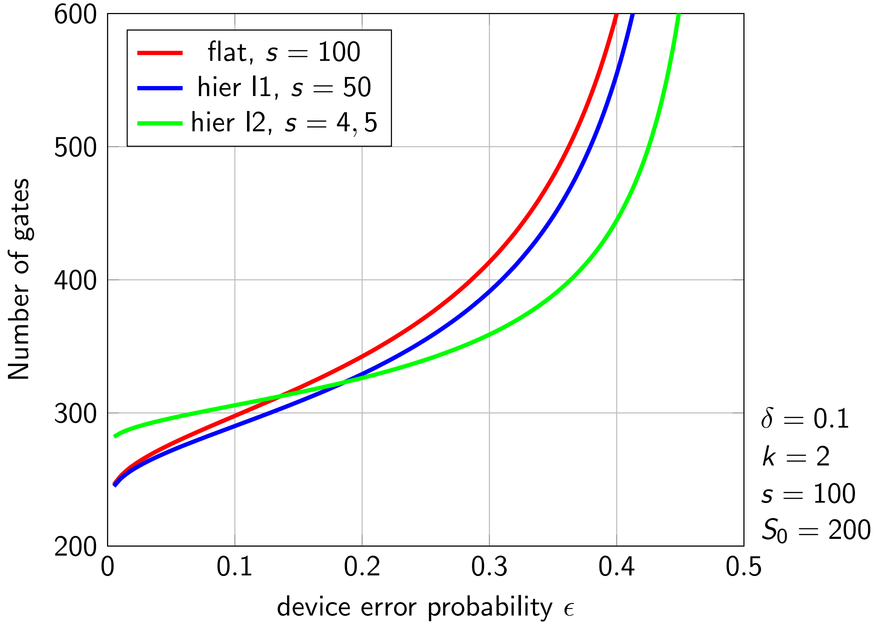
William Evans. "Information Theory and Noisy Computation". PhD thesis. Berkeley, CA, USA: Computer Science Division, University of California at Berkeley, 1994

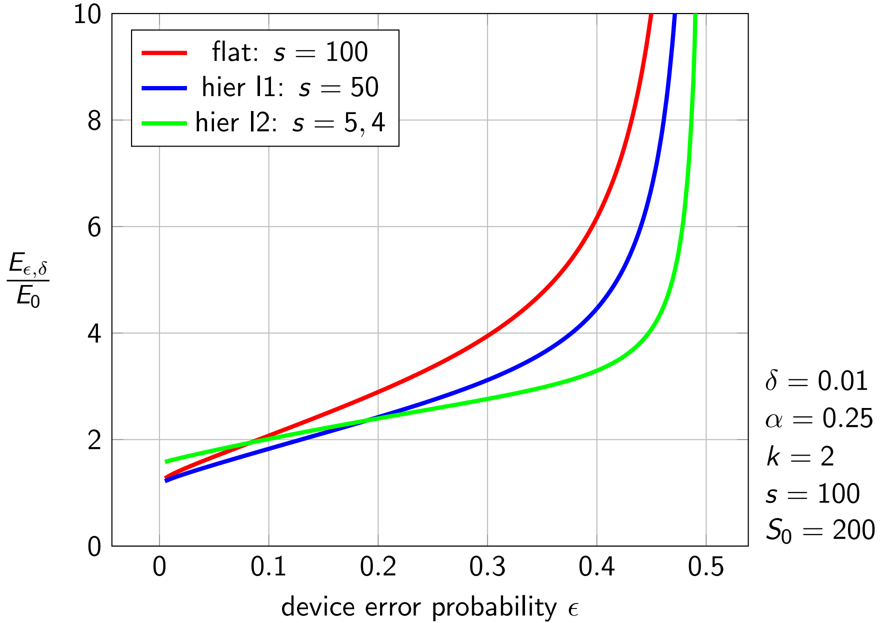
John von Neumann. "Probabilistic logics abd the synthesis of reliable organisms from unreliable components". In: *Automata Studies*. Edited by C. E. Shannon and J. McCarthy. Princeton University, 1996, pages 329–378


k = 2
k = 3
k = 4
Number of gates
200
150
100
50
0
0
0.1
0.2
0.3
0.4
0.5
device error probability ϵ
$\delta = 0.01$
$s = 10$
$S_0 = 21$



$\alpha = 0.25$
$k = 2$
$s = 10$
$S_0 = 21$


flat, $s = 100$
hier l1, $s = 50$
hier l2, $s = 4, 5$
Number of gates
600
500
400
300
200
0
0.1
0.2
0.3
0.4
0.5
device error probability ϵ
$\delta = 0.1$
$k = 2$
$s = 100$
$S_0 = 200$



flat: $s = 100$
hier l1: $s = 50$
hier l2: $s = 5, 4$
$\frac{E_{\epsilon,\delta}}{E_0}$
0
2
4
6
8
10
0
0.1
0.2
0.3
0.4
0.5
device error probability ϵ
$\delta = 0.01$
$\alpha = 0.25$
$k = 2$
$s = 100$
$S_0 = 200$

Imperfection in Networks

State of the Art

- Protocol stack: Fault free communication in all layers except the lowest
- Design of networks is modular: layers, hierarchy, composition
- Each module guarantees error-free communication
- Modular network design facilities network design

Imperfection in Networks

State of the Art

- Protocol stack: Fault free communication in all layers except the lowest
- Design of networks is modular: layers, hierarchy, composition
- Each module guarantees error-free communication
- Modular network design facilities network design

Imperfect Networks

- In each level/module allow for as much imperfection as possible
- Fault tolerance is distributed across the layers and the network
- Level of perfection is application dependent
- Level of perfection varies over time
- Each network layer and component can tune its fault tolerance

Outline

Self-calibrate and Adapt

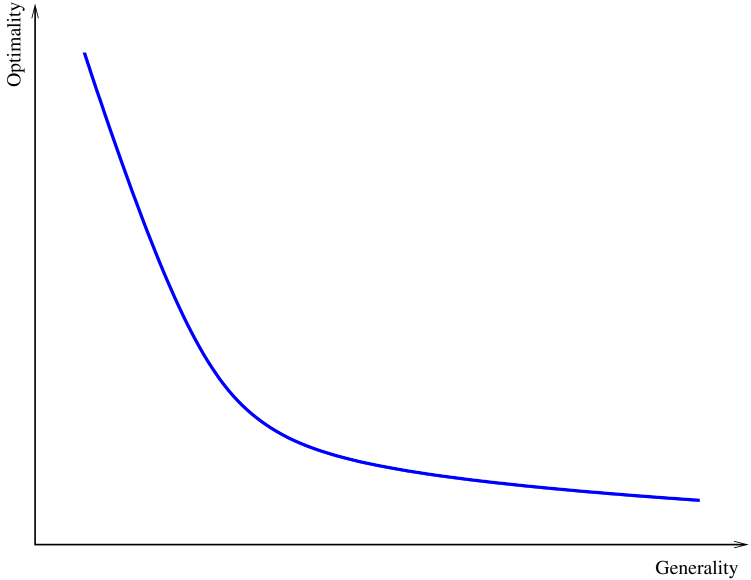
Challenge

❻ Self-calibrate, adapt and heal

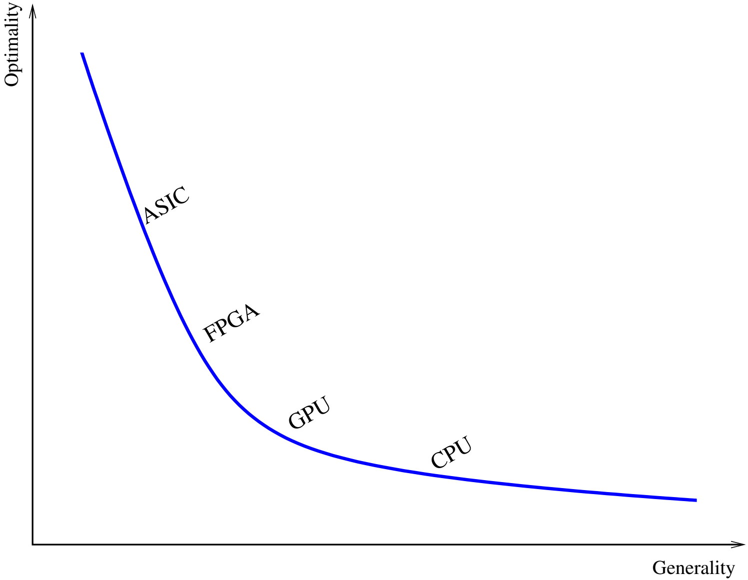
In-field adaptation addresses a fundamental challenge:

- Design and operation is separated in phases
- A system is designed for a wide range of applications and situations
- A well designed system is always too general and not optimal for the given case
- Trade-off between generality and optimality

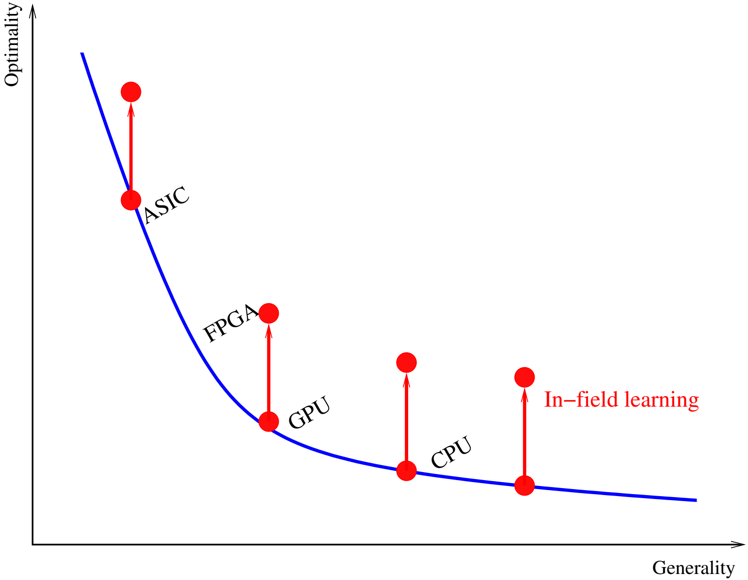
In-Field Learning



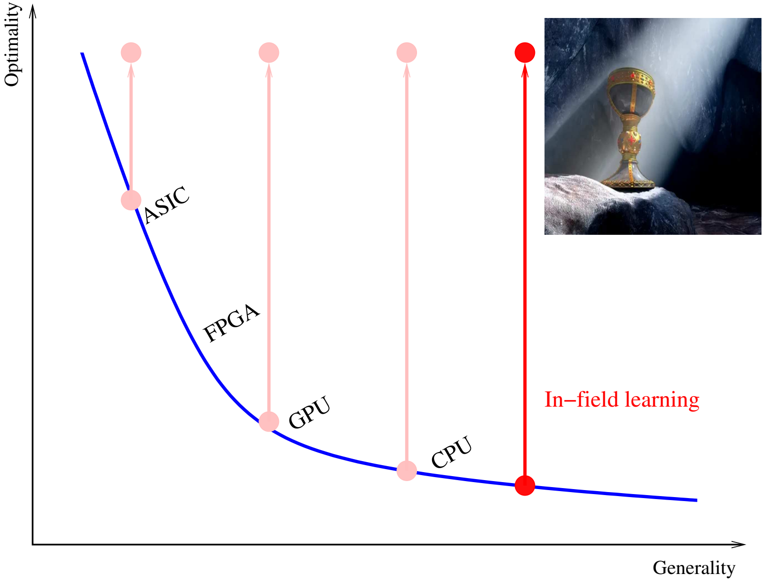
In-Field Learning

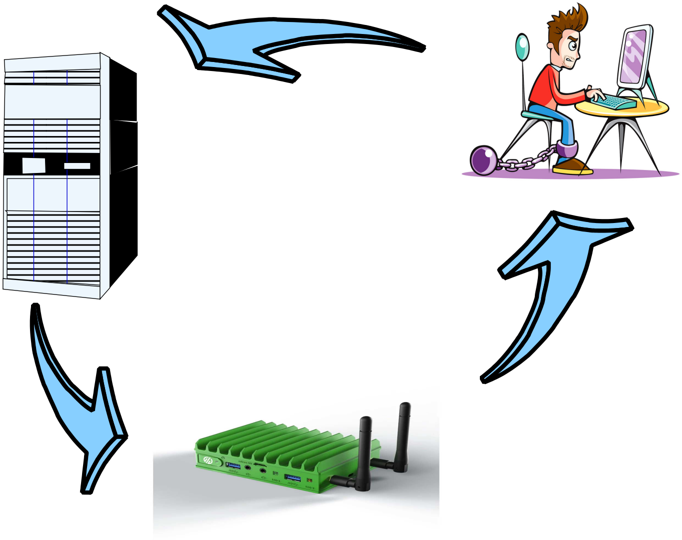


In-Field Learning

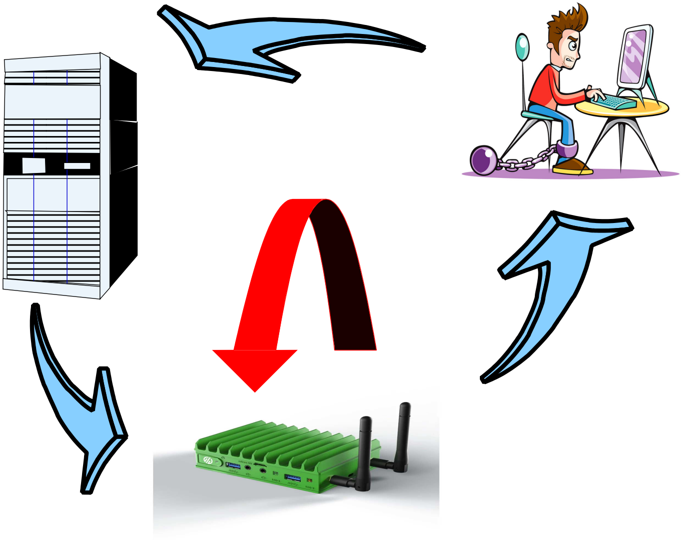


In-Field Learning





Design time learning



Design time learning
+ In-field Learning

In-Field Learning

In-field learning applies to

- Lazy communication 2
- Keep information local 3
- Customization 4
- Adapt to required fault tolerance 5

Adaptive Networks

When to adapt

- adapt to explicit application requests
- short term: adapt to short term demand variations; ns - ms
- medium term: adapt to medium term changes: ms - min
- long term: adapt to new application scenarios and slow changes: min - days

Adaptive Networks

When to adapt

- adapt to explicit application requests
- short term: adapt to short term demand variations; ns - ms
- medium term: adapt to medium term changes: ms - min
- long term: adapt to new application scenarios and slow changes: min - days

What to adapt

- Performance, delay
- When to send
- What to send
- Accuracy
- Fault tolerance

(Performance, delay; When to send; What to send; Accuracy: } Lazy communication)

Adaptive Networks

When to adapt

- adapt to explicit application requests
- short term: adapt to short term demand variations; ns - ms
- medium term: adapt to medium term changes: ms - min
- long term: adapt to new application scenarios and slow changes: min - days

What to adapt

- Performance, delay
- When to send
- What to send
- Accuracy
- Fault tolerance

(Performance, delay; When to send; What to send; Accuracy: } Lazy communication)

How to adapt

- Predefined
- Constrained
- Unconstrained in-field learning

Outline

DATA DRIVEN COMPUTING

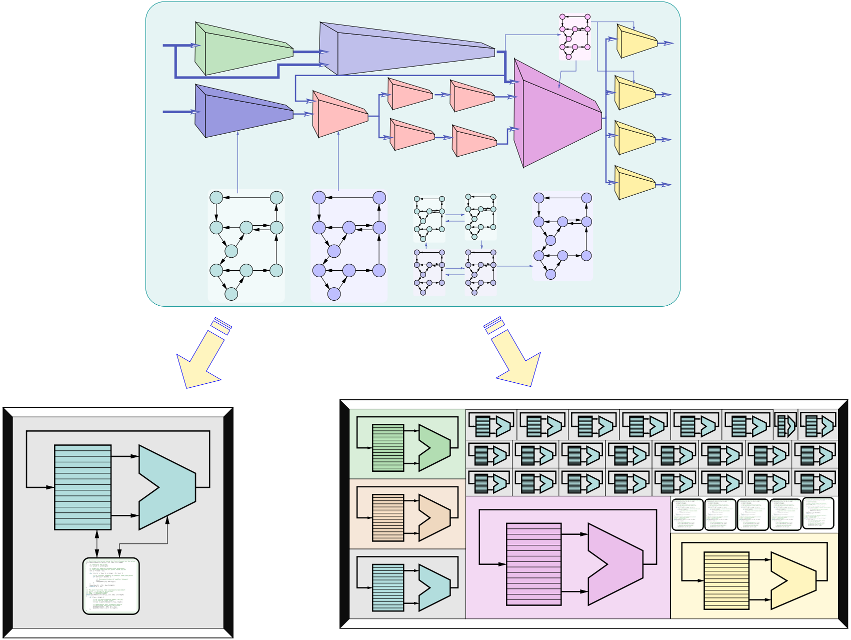
Control vs Data Driven Computing

Control driven

- Classic von Neumann
- Limited, shared resources
- Elaborate control for allocation of resources and scheduling of the execution

Data driven

- No shared resources
- Data flow determines the execution
- Demand-driven or data-driven
- Resources operate as slow as possible

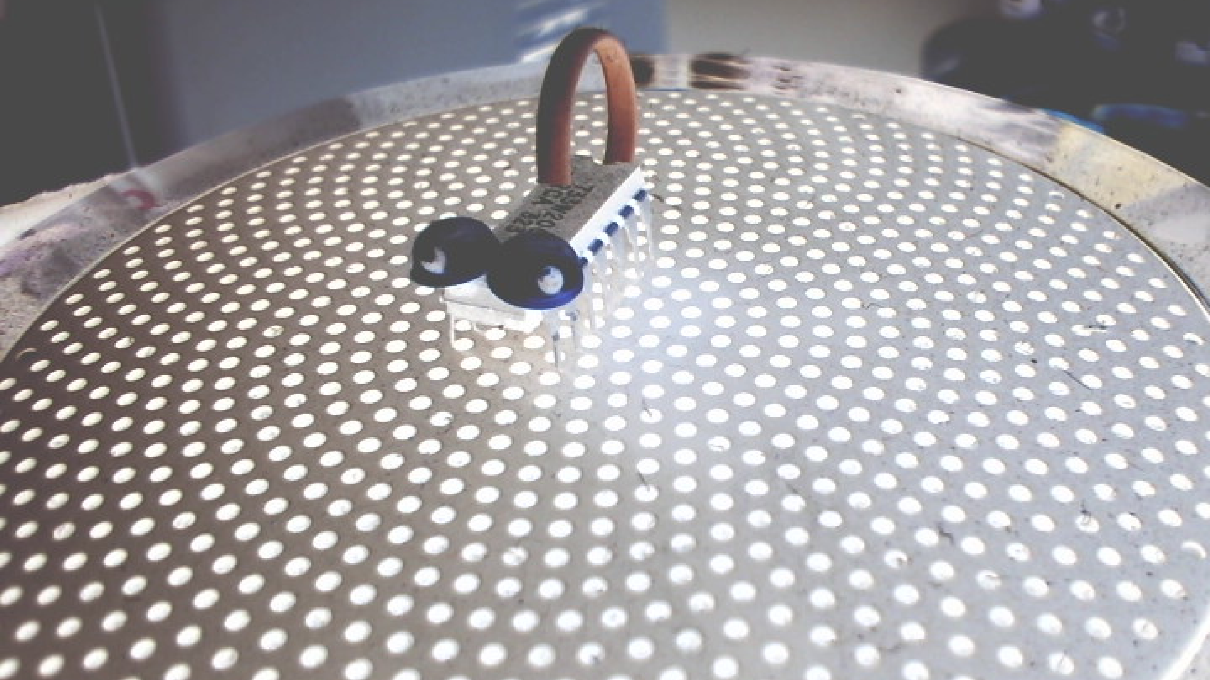


Outline

SUMMARY

Conclusions for NoCs

- Heterogeneity:
    - Hierarchical network to connect cores on chip and chiplets on package
    - Connect nodes with very different requirements
    - Requirements vary over time
    - Addressing by function, not location
- Lazy Communication: Network should communicate
    - as slow,
    - as late, and
    - as inaccurate as possible
- Fault tolerance:
    - Distribute fault tolerance across hierarchy layers
    - Adapt fault-tolerance level
- Continuous in-field learning and adapting




TECHNISCHE HOCHSCHULE