TU WIEN Informatics

Advanced Computer Architecture

Block A: Motivation, Content & Organization

Daniel Mueller-Gritschneder

Agenda

- Motivation: Why learn about Computer Architecture?
- Course Content and Organization
- Outlook: Other courses in the HW domain

A1 Motivation

A1.0. Chip Production (1)

- Sand -> silicon ingot

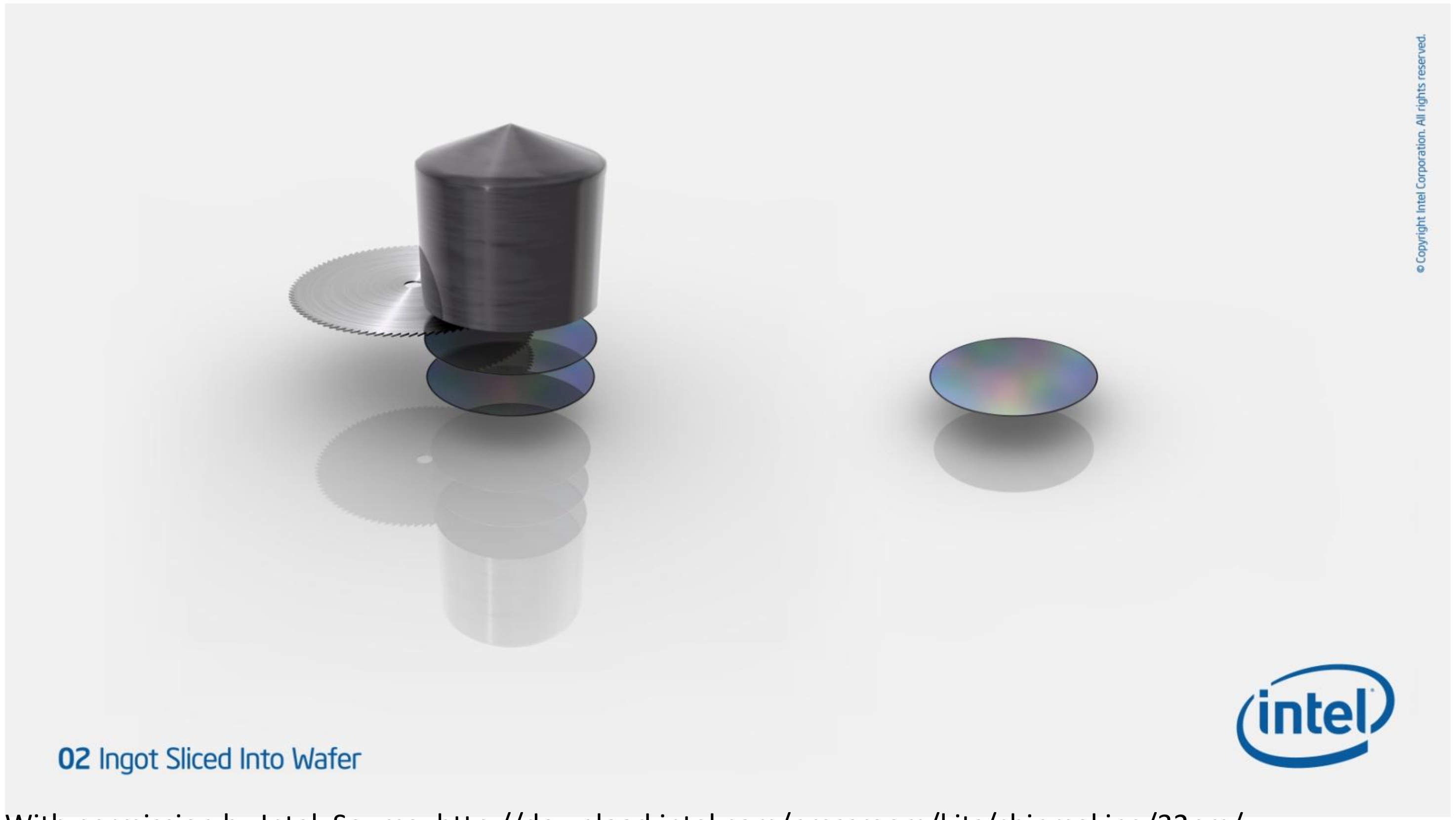
01 Sand Melted Into Ingot
© Copyright Intel Corporation. All rights reserved.

With permission by Intel, Source:
http://download.intel.com/pressroom/kits/chipmaking/32nm/

Deutsches Museum
Ausstellung Mikroelektronik

A1.0. Chip Production (2)

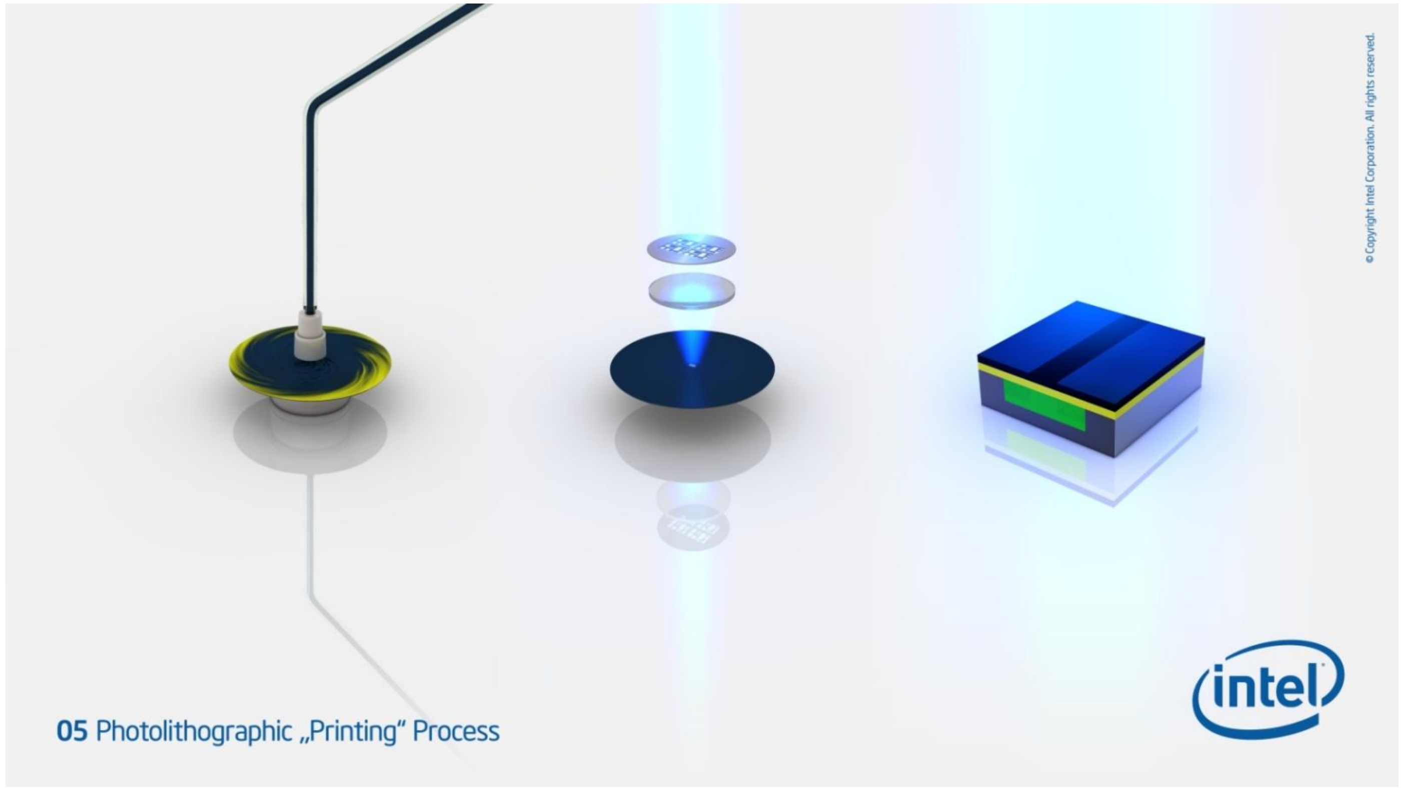
- Silicon Ingot -> Wafer


© Copyright Intel Corporation. All rights reserved.
02 Ingot Sliced Into Wafer
intel

With permission by Intel, Source: http://download.intel.com/pressroom/kits/chipmaking/32nm/

A1.0. Chip Production (3)

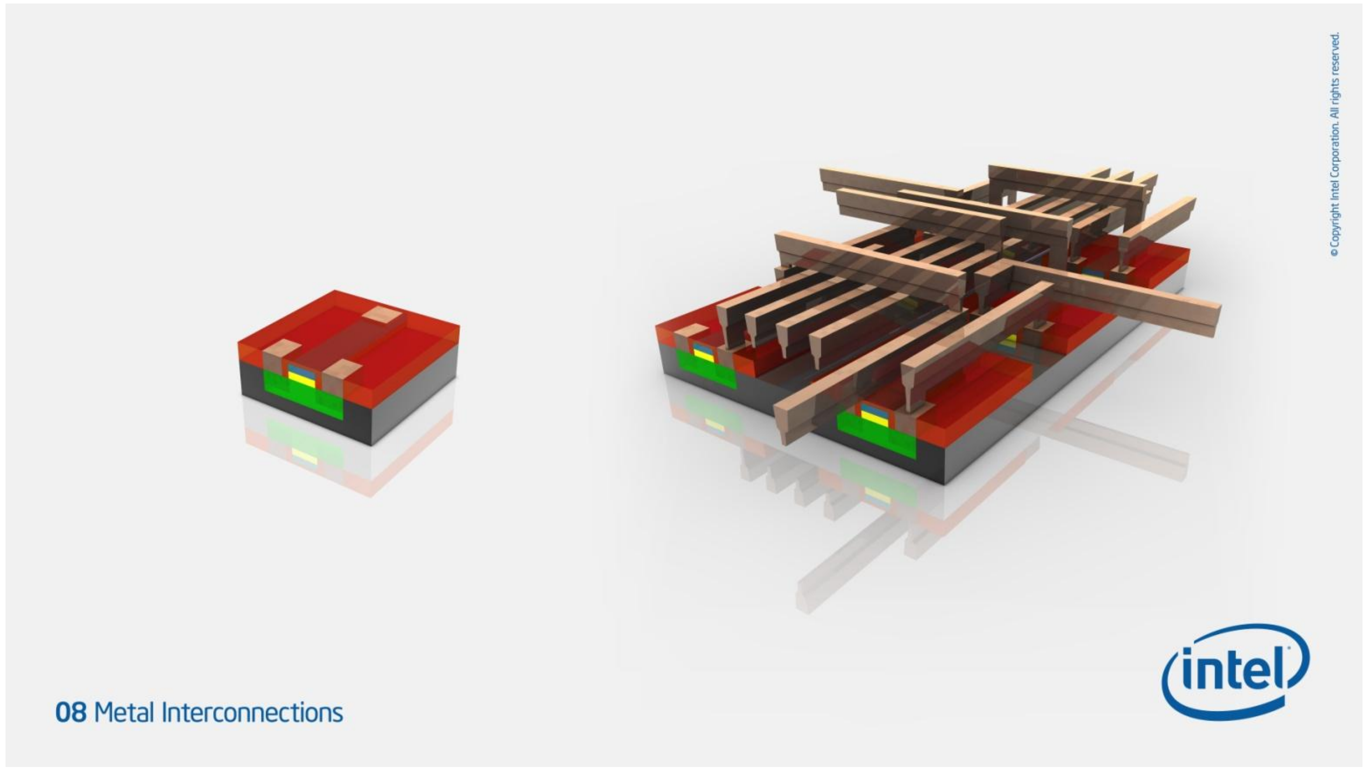
- One of many processing steps: lithography


© Copyright Intel Corporation. All rights reserved.
intel
05 Photolithographic „Printing“ Process

With permission by Intel, Source: http://download.intel.com/pressroom/kits/chipmaking/32nm/

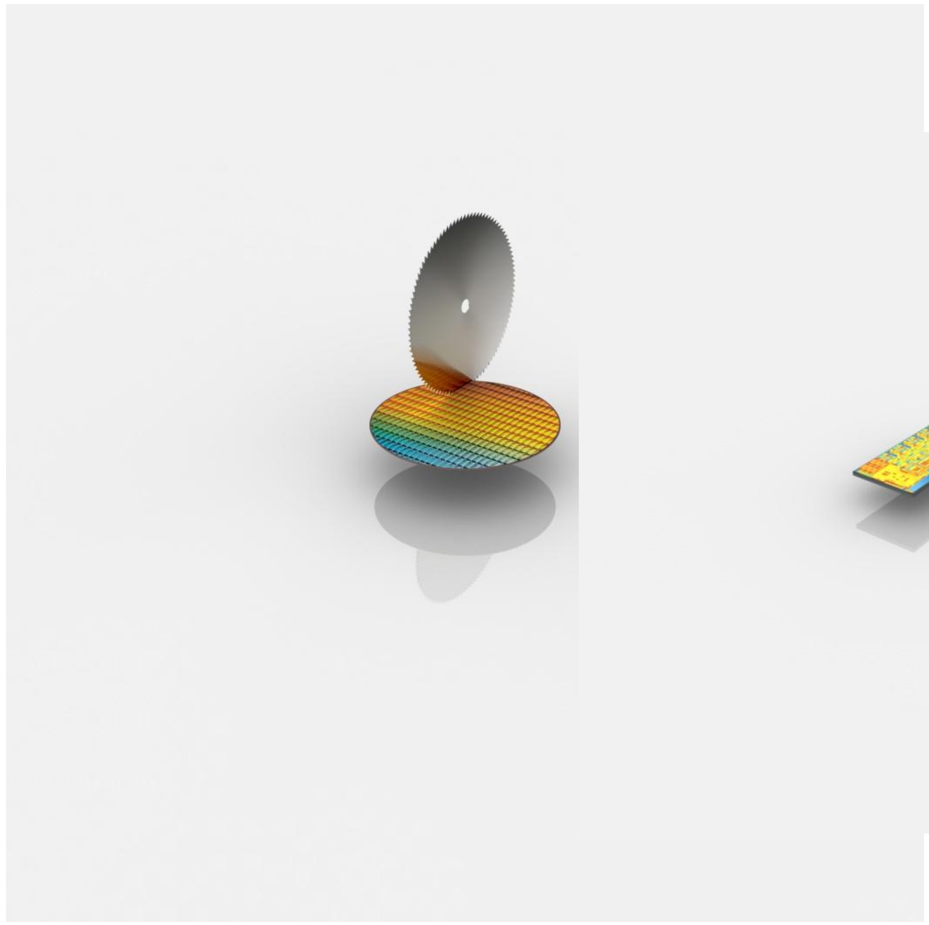
A1.0. Chip Production (4)

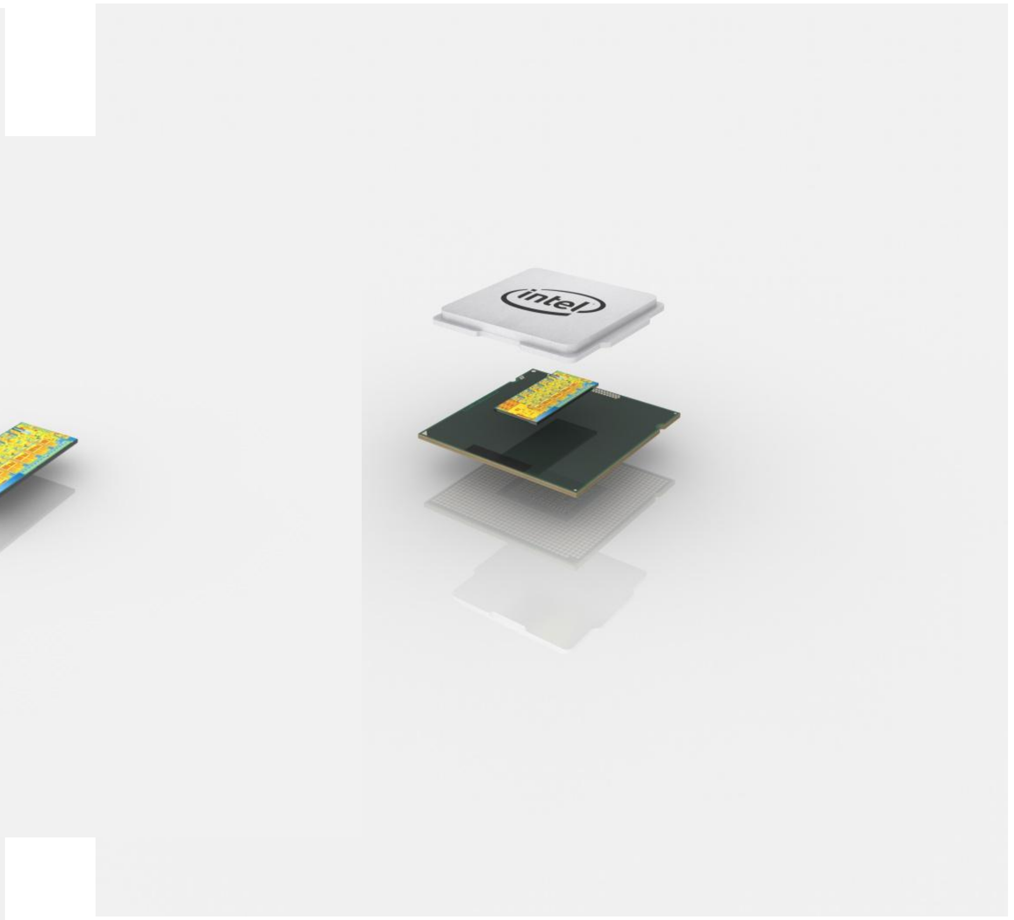
- Wires between transistors


08 Metal Interconnections
© Copyright Intel Corporation. All rights reserved.
intel

With permission by Intel, Source: http://download.intel.com/pressroom/kits/chipmaking/32nm/

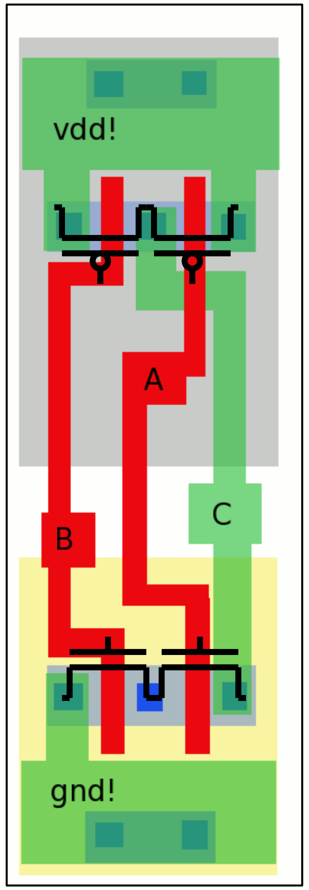
A1.0. Chip Production (5)

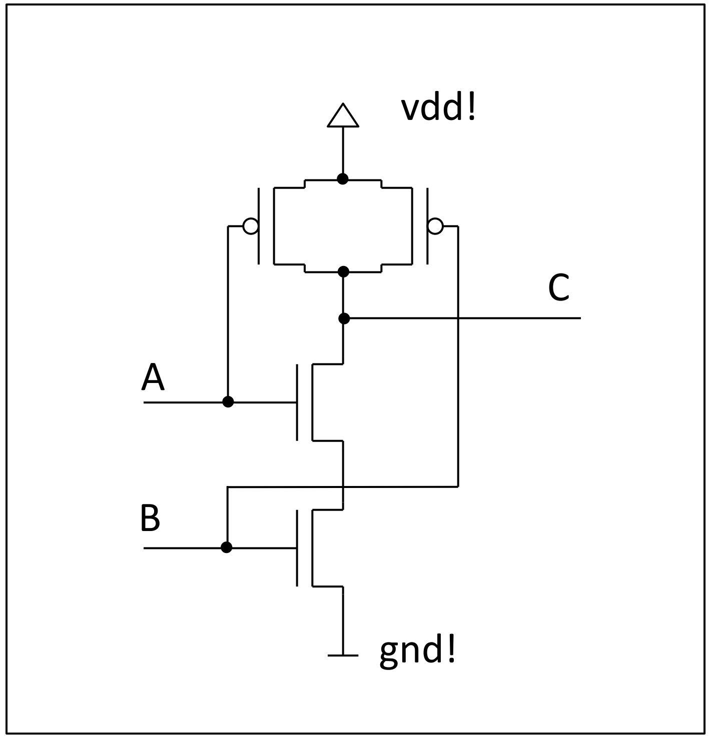




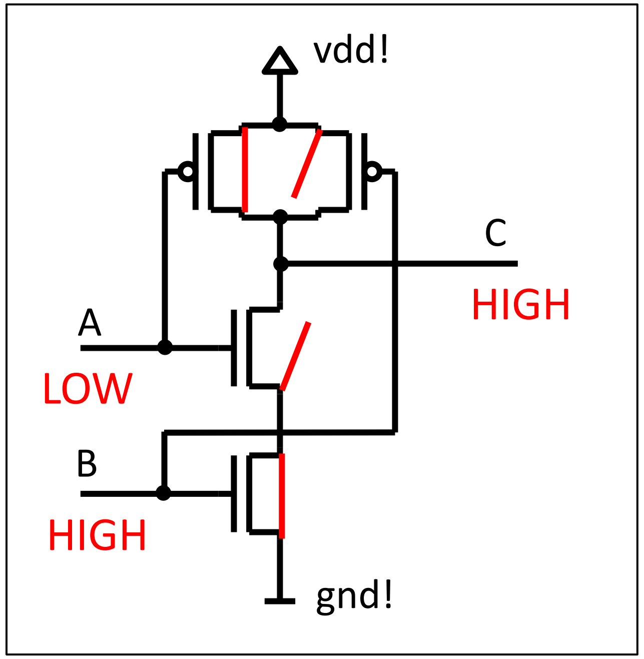
With permission by Intel, Source: http://download.intel.com/pressroom/kits/chipmaking/32nm/

Layout of a standard cell





Logic levels



A	B	C
LOW	LOW	HIGH
LOW	HIGH	HIGH
HIGH	LOW	HIGH
HIGH	HIGH	LOW

HIGH: High voltage near to VDD
LOW: Low voltage near VSS (gnd)

Positive / Negative Logic

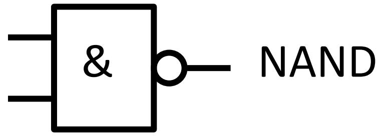
- **Positive logic**: the logic level HIGH represents the logic value 1 and the logic level LOW the logic value 0.

Logic levels

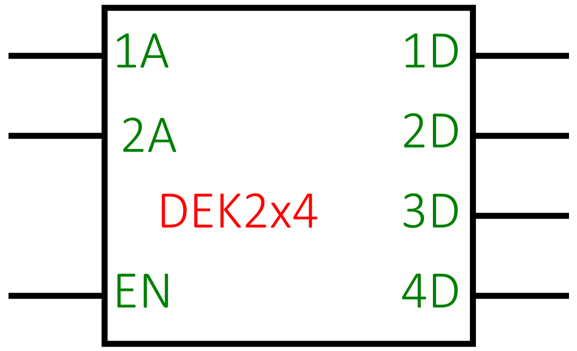
A	B	C
LOW	LOW	HIGH
LOW	HIGH	HIGH
HIGH	LOW	HIGH
HIGH	HIGH	LOW

Positive logic

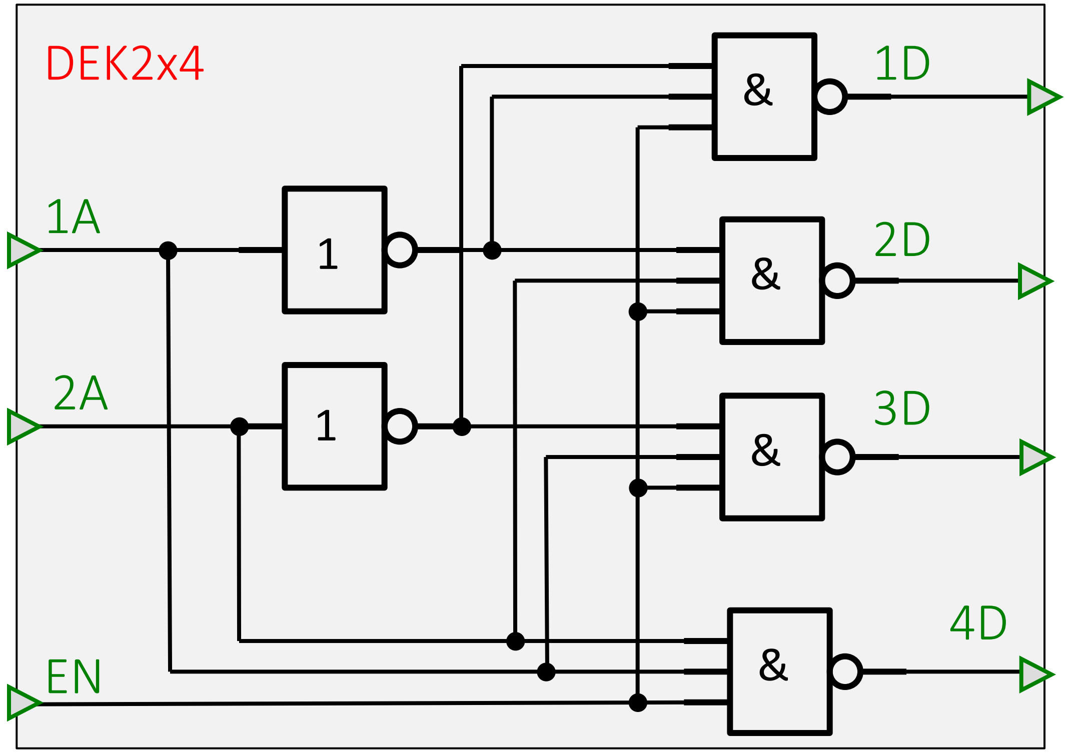
a	b	c
0	0	1
0	1	1
1	0	1
1	1	0



Example: 2x4-Decoder with Enable

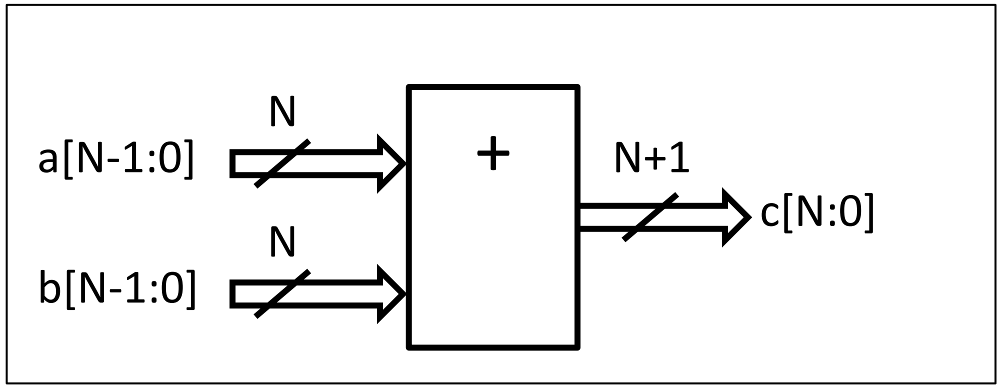


EN	1A	2A	1D	2D	3D	4D
0	X	X	1	1	1	1
1	0	0	**0**	1	1	1
1	0	1	1	**0**	1	1
1	1	0	1	1	**0**	1
1	1	1	1	1	1	**0**

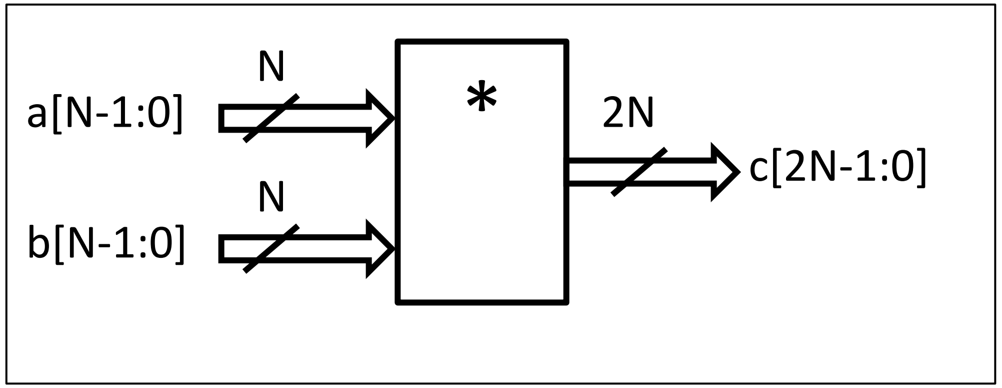


Adders and Multipliers

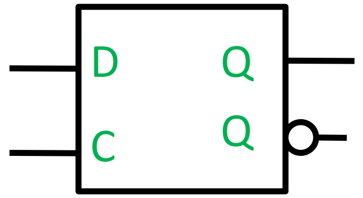
- Adder
  - Carry Ripple
  - Carry Look ahead



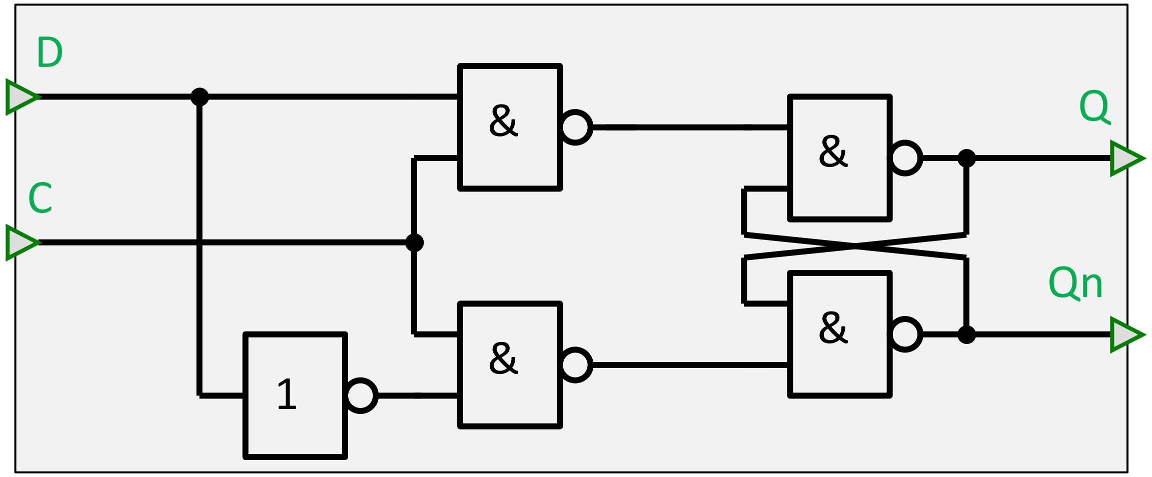
- Multiplier



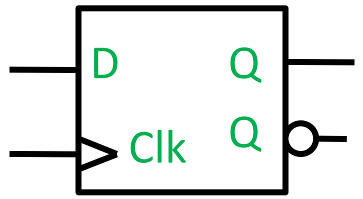
D-Latch



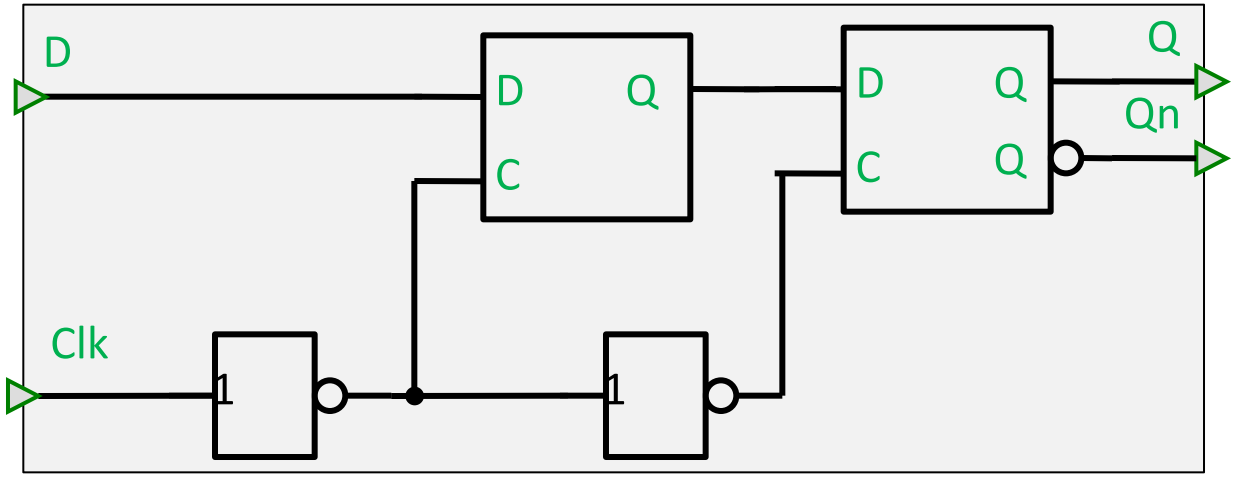
C	D	Q	Qn
1	0	0	1
1	1	1	0
0	X	Last Q	Last Qn



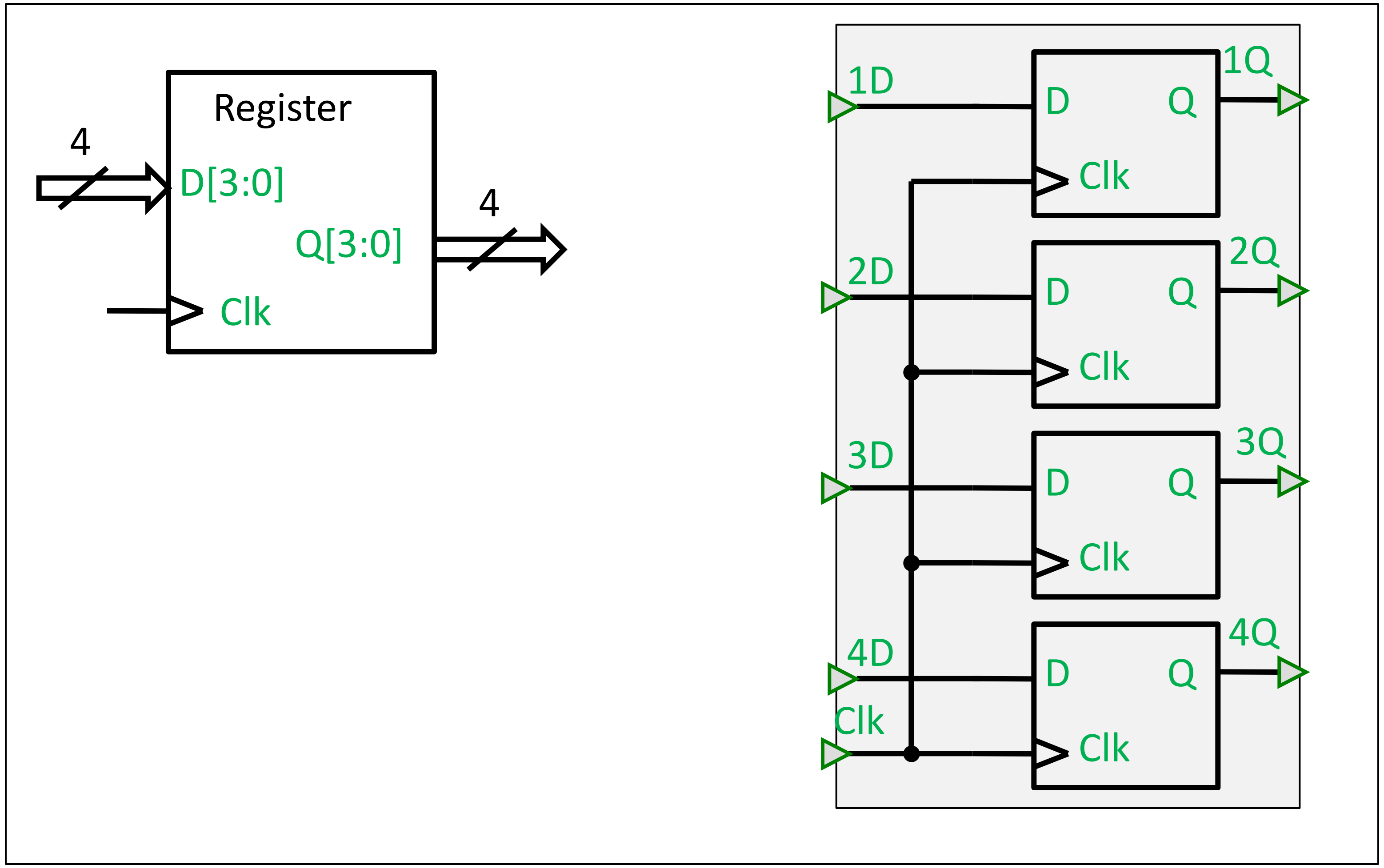
Edge-driven D Flip Flop



Clk	D	Q	Qn
↑	0	0	1
↑	1	1	0
0	X	Last Q	Last Qn
1	X	Last Q	Last Qn

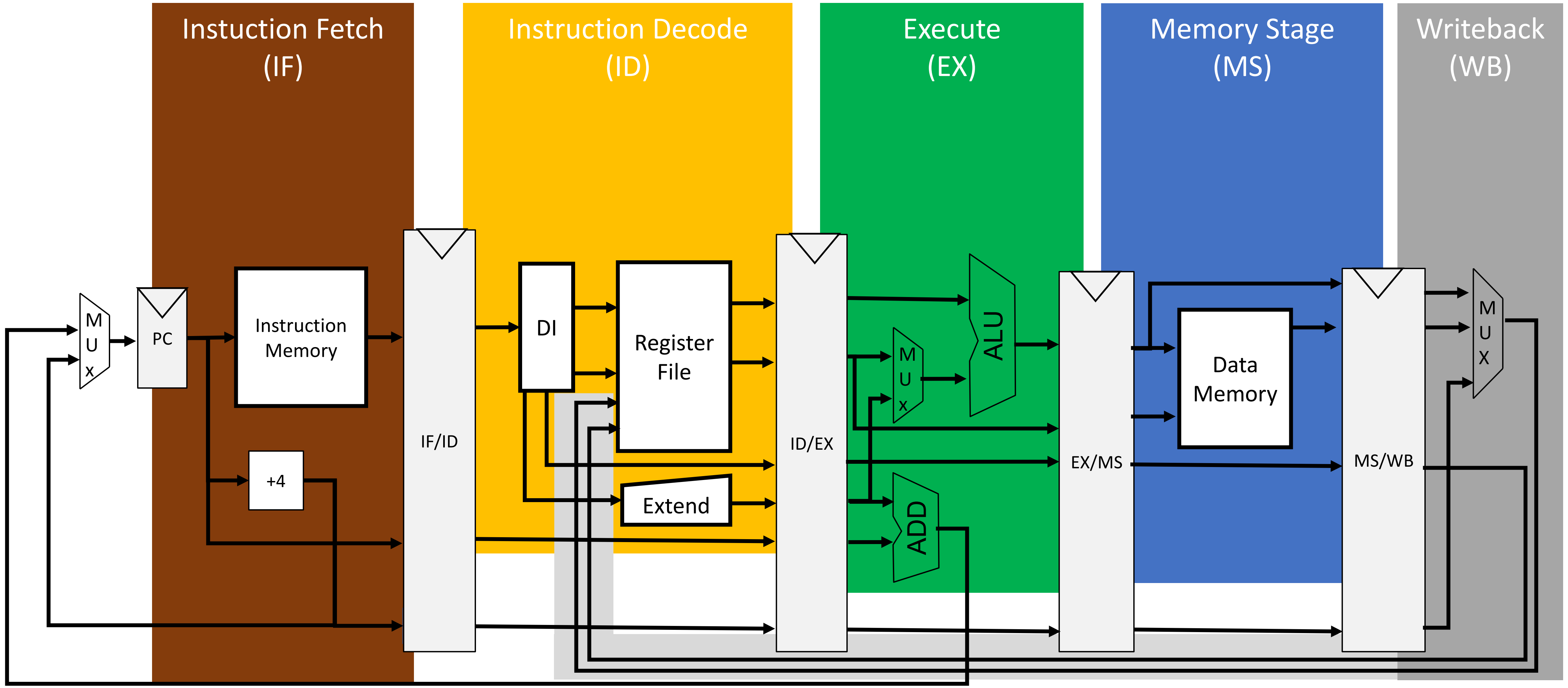


Example: 4-bit Register



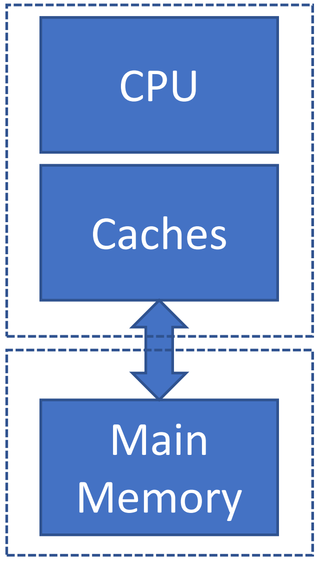
Five-stage Pipeline - Stages

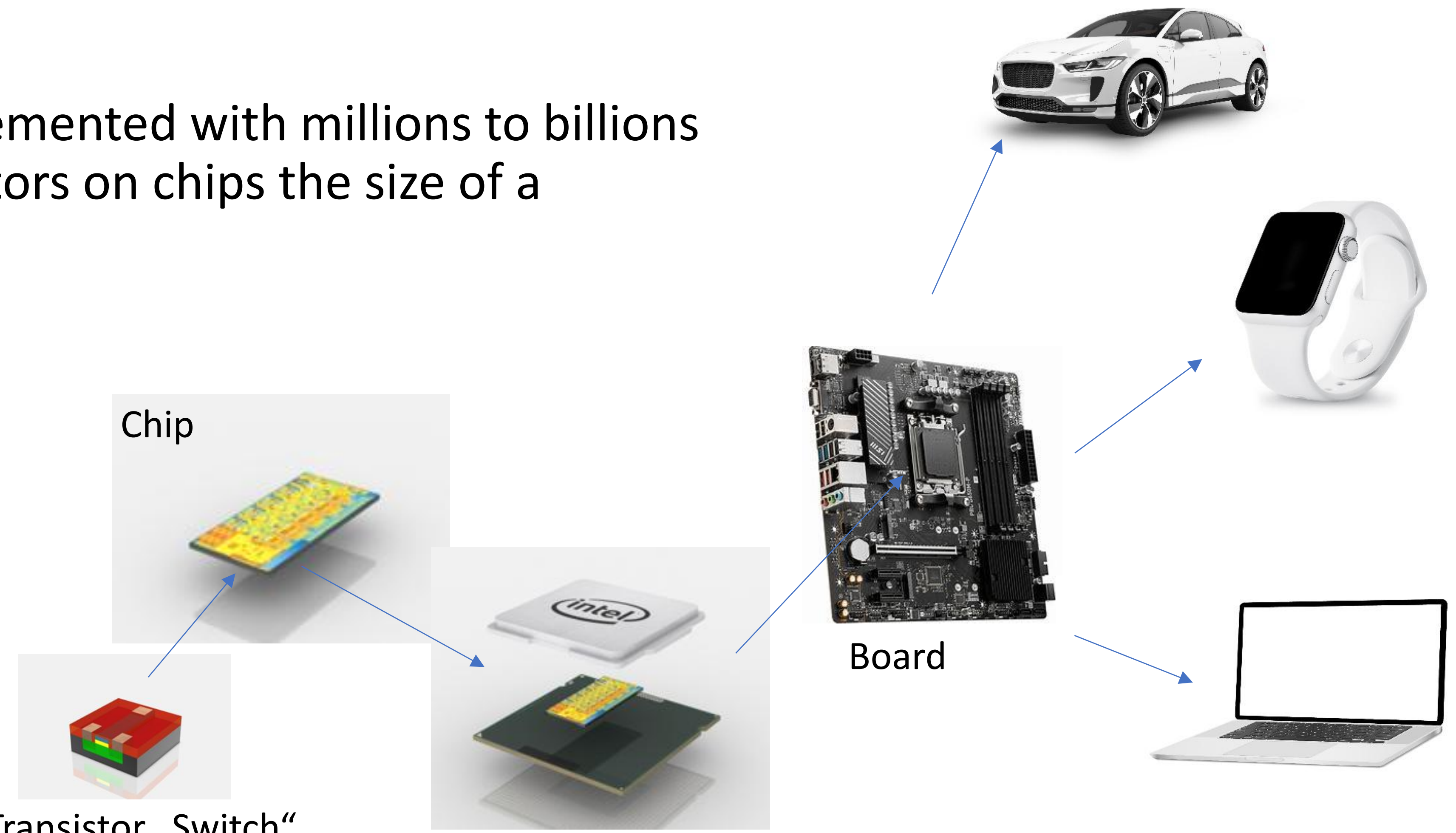
- Stages:



Semiconductor Chips

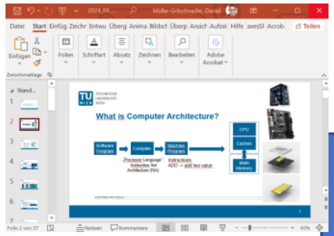
- Processors implemented with millions to billions of CMOS transistors on chips the size of a thumbnail

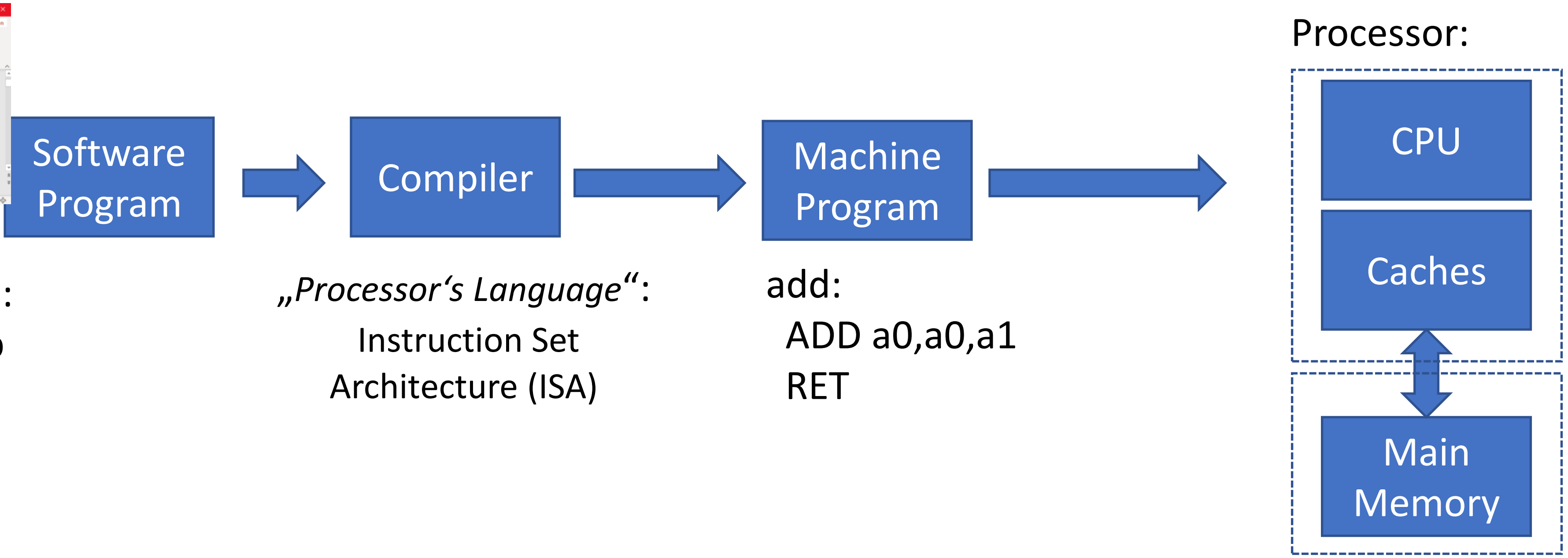




- http://download.intel.com/pressroom/kits/chipmaking/32nm/
- Adobe Stock

What is Computer Architecture?





```
def add(a,b):
  return a+b
```

„Processor's Language":
Instruction Set
Architecture (ISA)

```
add:
  ADD a0,a0,a1
  RET
```

- **Central Processing Unit (CPU)**: Executes the program (instructions)
- **Main Memory**: Stores your program's instructions and data
- **Caches**: Keep local copies because reading and writing to main memory takes long

➢ **Computer Architecture: How to design the processor such that it can execute the software fast and energy-efficient while considering the costs.**

A new Golden Age for Computer Architecture

- Turing Lecture June 2018
  John Hennessy & David Patterson

- Ongoing shift how processors are designed

➢ Driven by major challenges in semiconductor development

turing lecture

DOI:10.1145/3282307

Innovations like domain-specific hardware, enhanced security, open instruction sets, and agile chip development will lead the way.

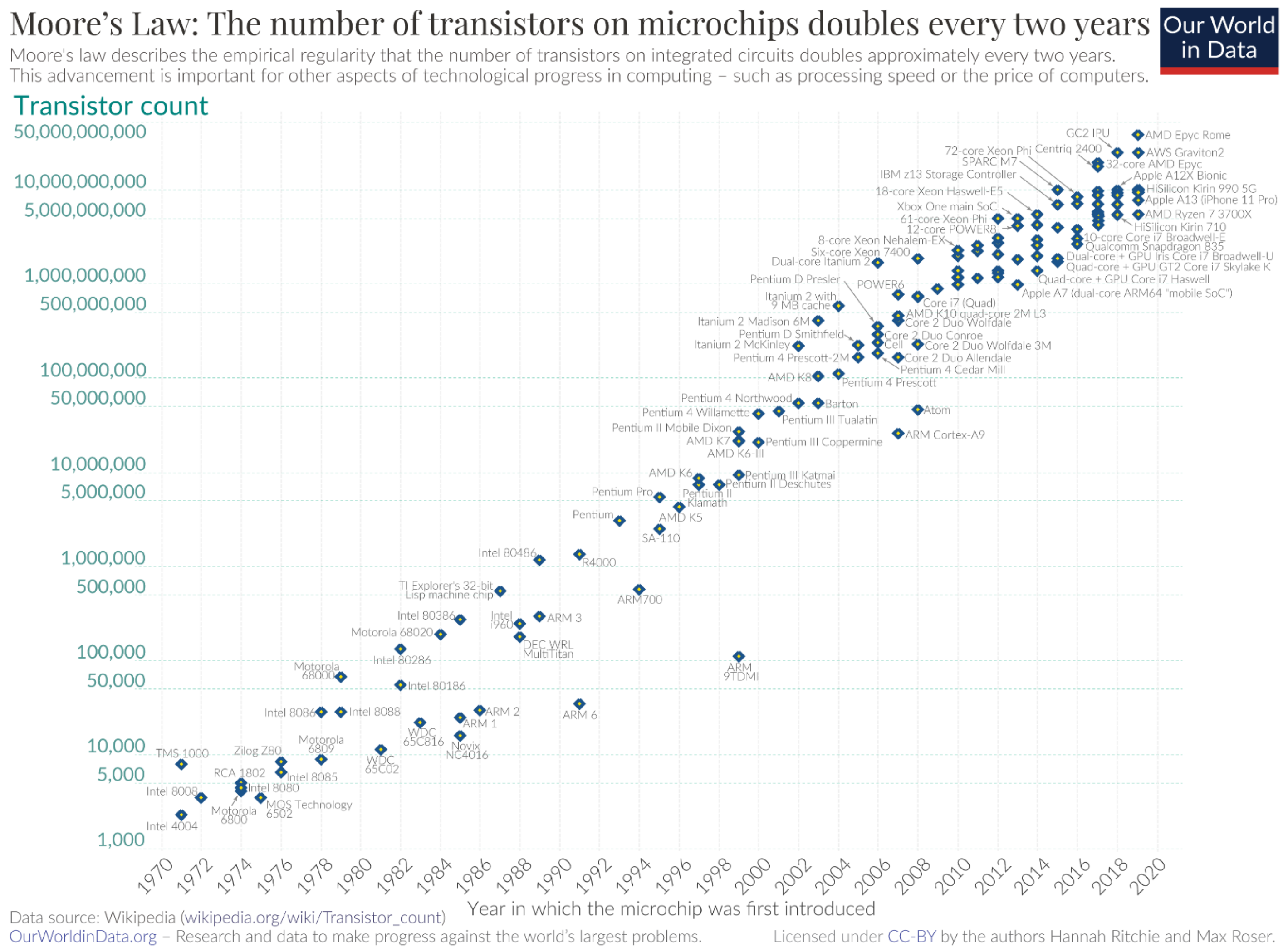
BY JOHN L. HENNESSY AND DAVID A. PATTERSON

A New Golden Age for Computer Architecture

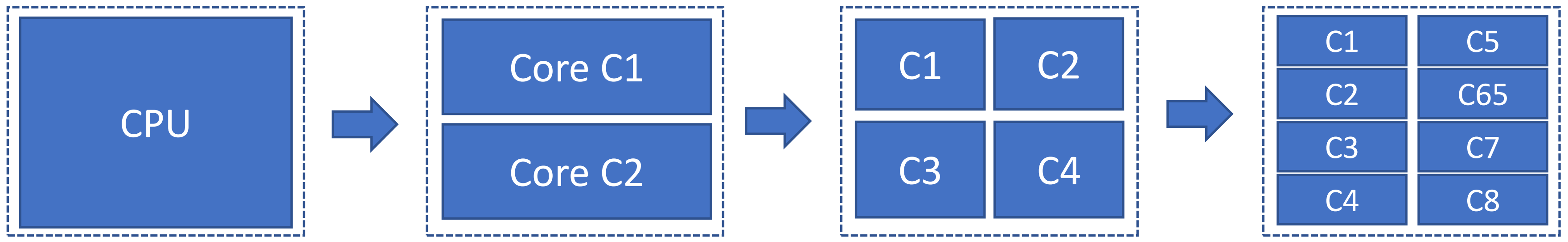
- https://dl.acm.org/doi/pdf/10.1145/3282307

Moore‘s Law

- Transistor size is shrinking with advancing semiconductor technology
- The number of transistors on chips doubles every two years
- ~Double the compute on same area



Dual Core CPU



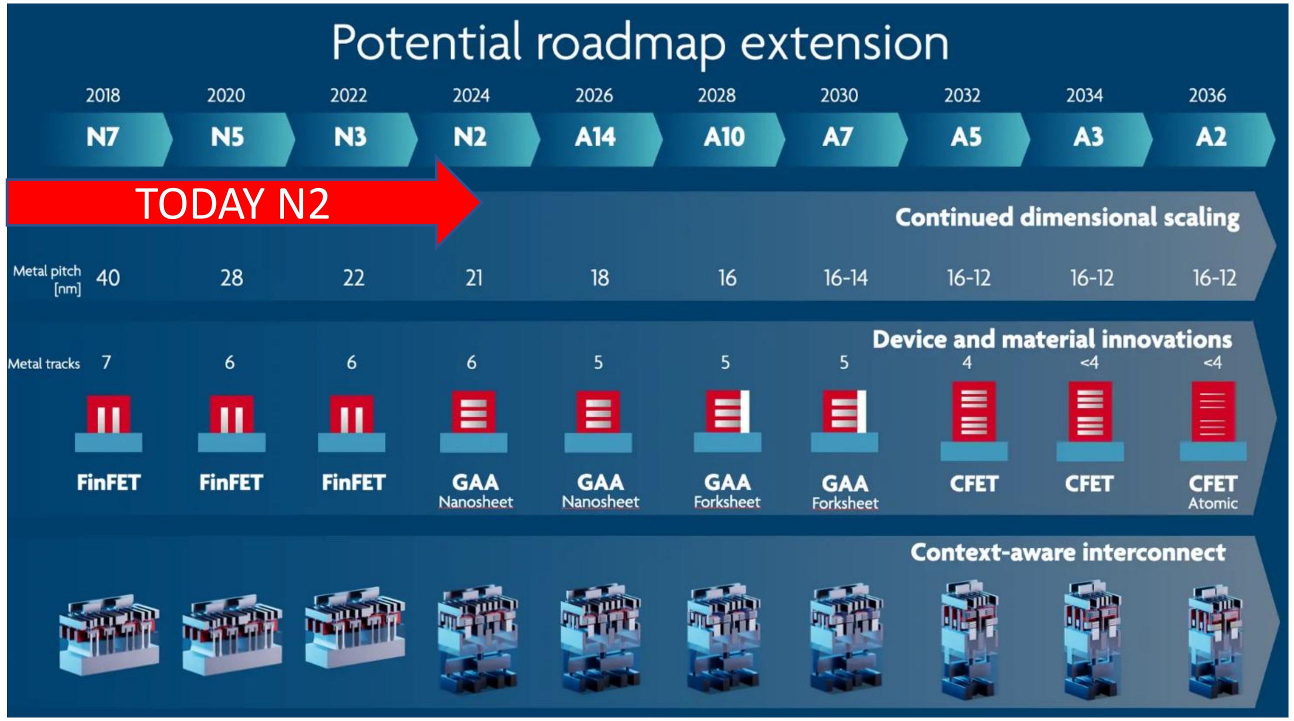
Semiconductor Challenges

- **Scaling**: Transistor density is coming close to the size of atoms, quantum effects start interfering with the operation

Wave length of light: 380nm-760nm

N2: 2 nm = 2 * 10^{-9} m
A14: 14 ångström = 1,4nm

Atomic radius of silver = 1,72 ångström



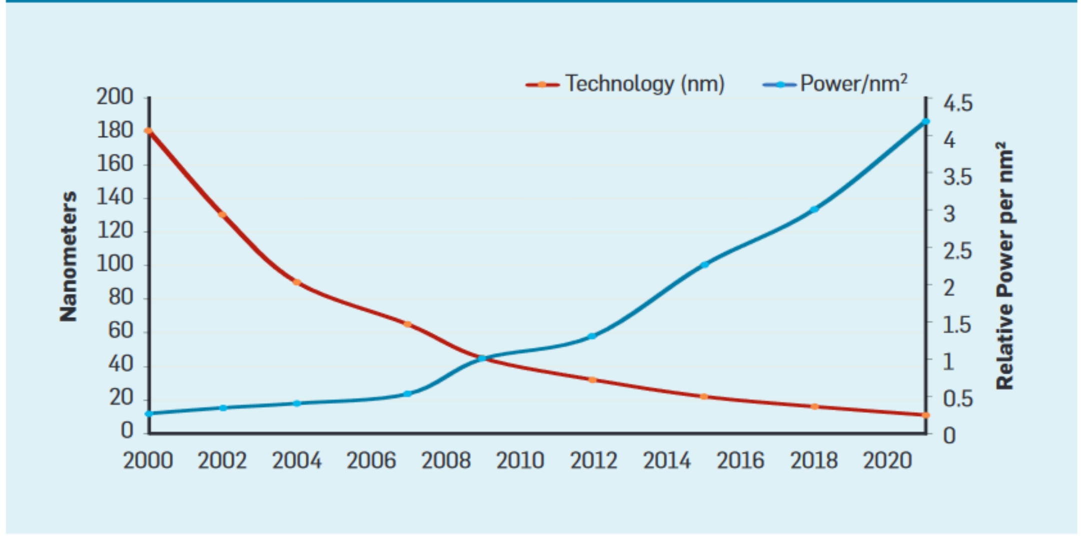
Potential roadmap extension
2018 N7
2020 N5
2022 N3
2024 N2
2026 A14
2028 A10
2030 A7
2032 A5
2034 A3
2036 A2
TODAY N2
Continued dimensional scaling
Metal pitch [nm] 40 28 22 21 18 16 16-14 16-12 16-12 16-12
Device and material innovations
Metal tracks 7 6 6 6 5 5 5 4 <4 <4
FinFET FinFET FinFET
GAA Nanosheet
GAA Nanosheet
GAA Forksheet
GAA Forksheet
CFET CFET
CFET Atomic
Context-aware interconnect

https://periodictableguide.com/silver-ag-element-periodic-table/
https://www.tomshardware.com/news/imec-reveals-sub-1nm-transistor-roadmap-3d-stacked-cmos-20-plans

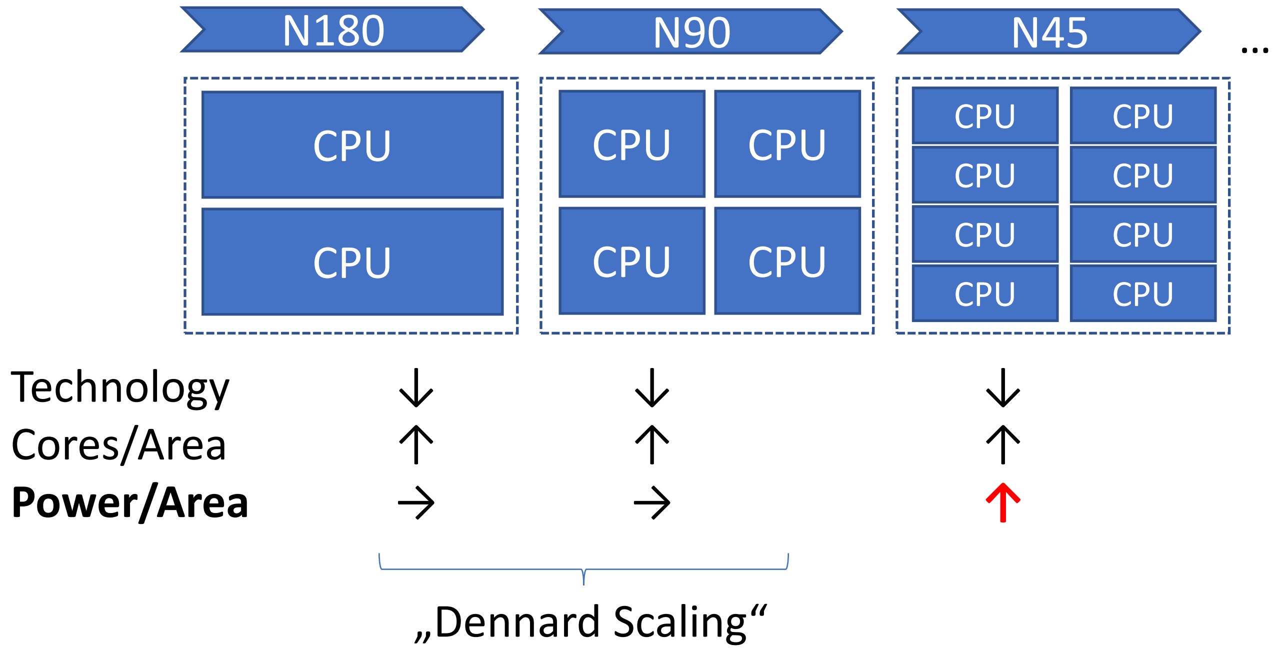
Semiconductor Challenges

- **Scaling**: Transistor structures are coming close to the size of atoms, quantum effects start interfering with the operation
- **Power**: „*Dennard Scaling*“ ended - Hard to bring power to and heat from the chips.

Figure 3. Transistors per chip and power per mm².



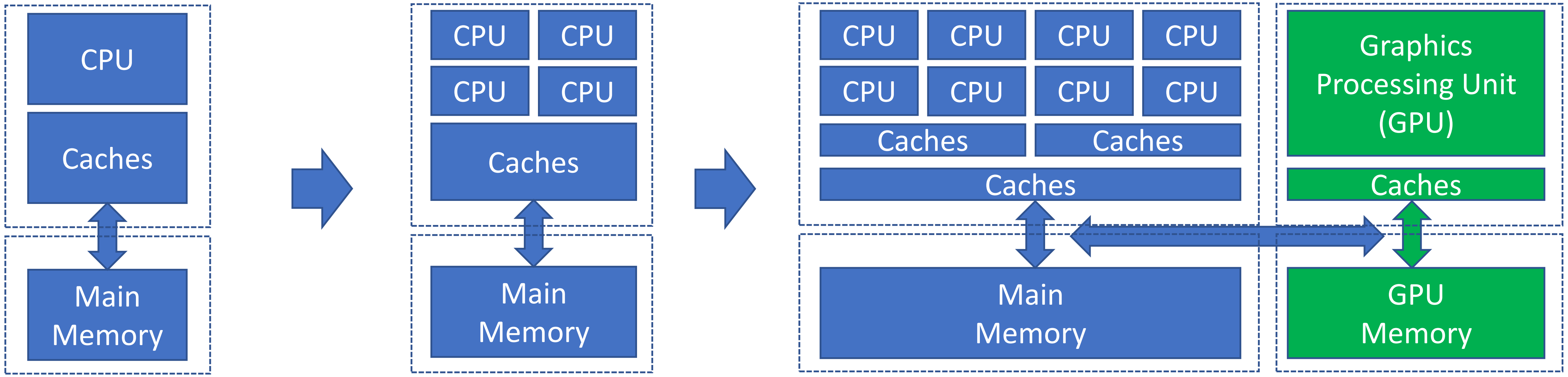
https://dl.acm.org/doi/pdf/10.1145/3282307



	N180	N90	N45
Technology	↓	↓	↓
Cores/Area	↑	↑	↑
Power/Area	→	→	↑

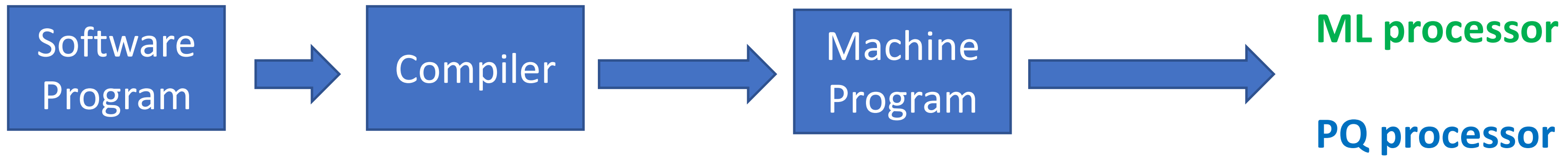
Semiconductor Challenges

- **Scaling**: Transistor structures are coming close to the size of atoms, quantum effects start interfering with the operation
- **Power**: „*Dennard Scaling*“ ended, Hard to bring power to and heat from the chips.
- **Memory**: Memory bandwidth cannot keep up with processor performance



How to further scale system performance?

- Shift to domain-specific architectures (DSAs) that are specialized for a certain application (workload)



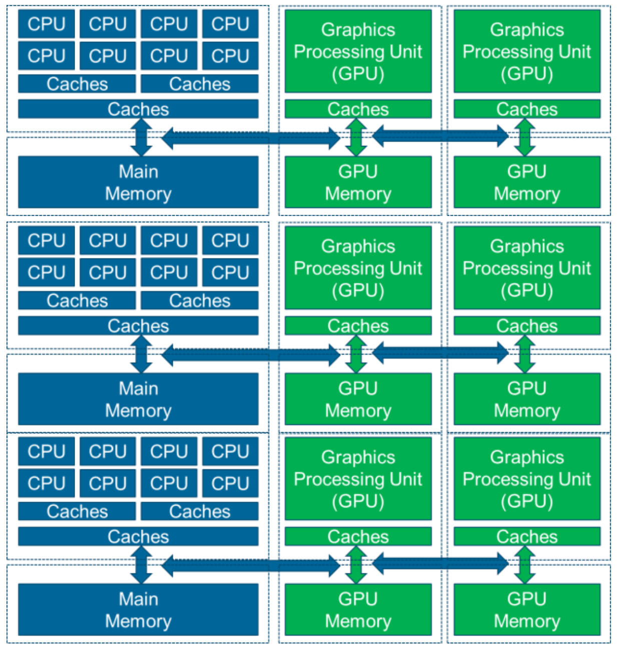
Machine Learning (ML)
Postquantum Security (PQ)

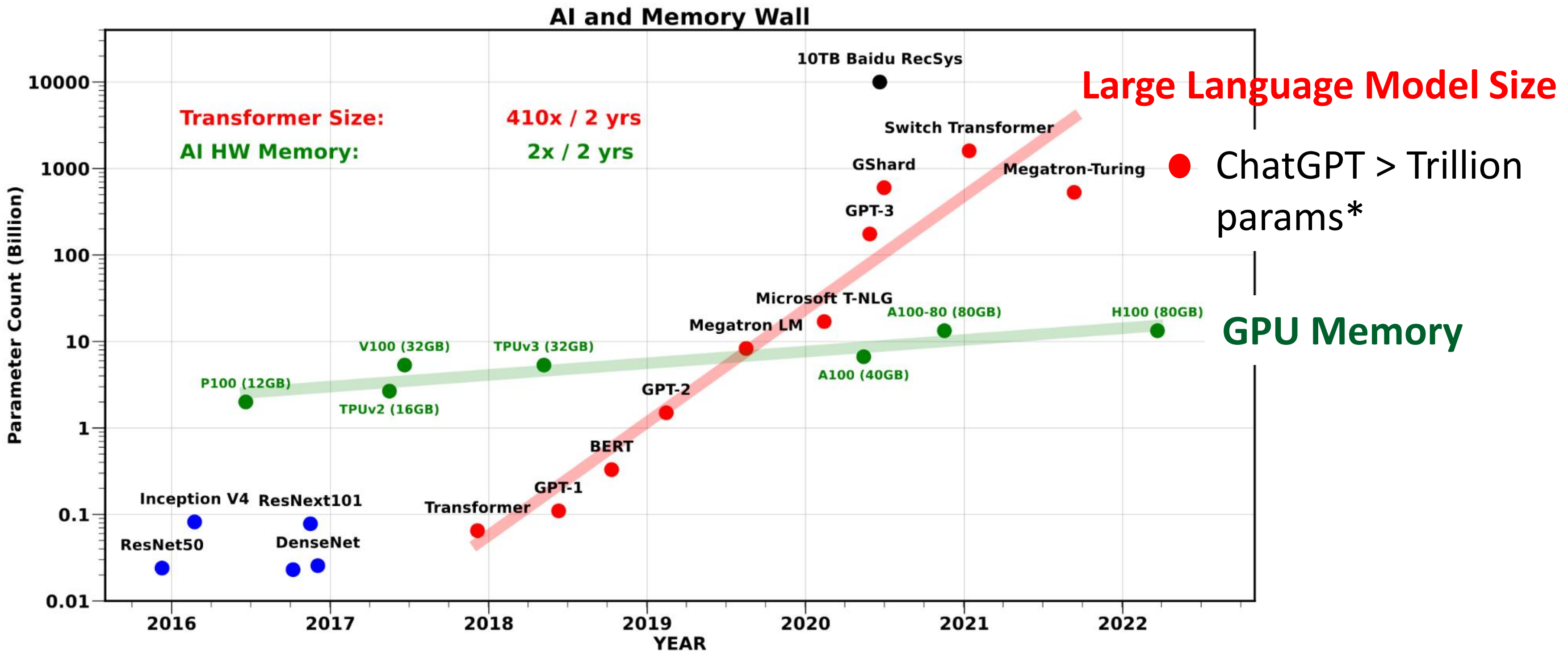
„Processor's Language":
Instruction Set Architecture (ISA)

- Post-CMOS technologies

Domain: Large Language Models (LLM)

- General Purpose GP-GPU clusters for LLMs such as ChatGPT



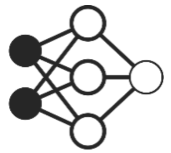


● ChatGPT > Trillion params*

Domain: Automotive

Advanced driver assistance functions (ADAS)

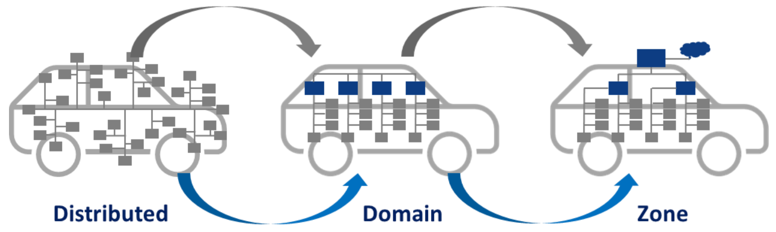
- Rising number of machine-learning(ML)-based workloads.

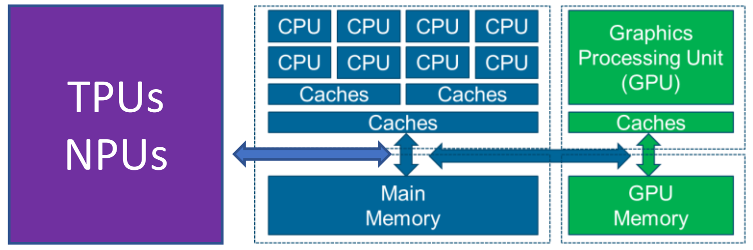


Convolutional Neural Networks, Transformers

Hardware view:

- Migration from Distributed, Domain to Zone.
- Powerful central compute platforms.
- Processors tailored for ML:
  - General-purpose GPUs (GP-GPUs)
  - Tensor Processing Units (TPUs)
  - Neural Processing Units (NPUs)

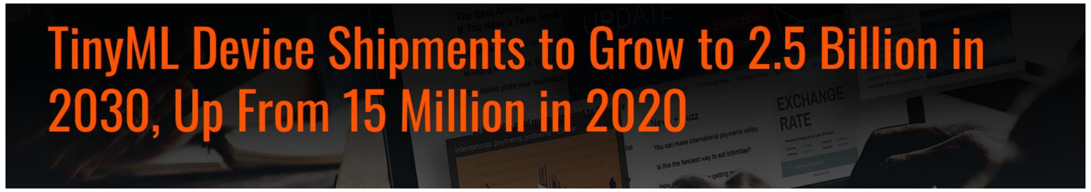




Safety-critical real-time system

- Real-time constraints / deadlines.
- Functional safe and secure.

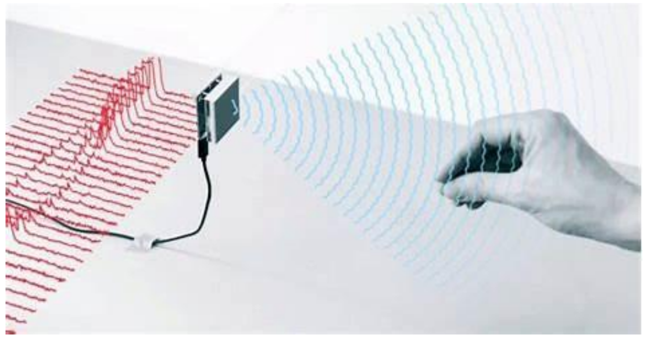
Domain: tinyML

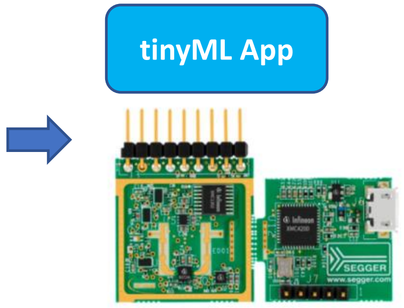

TinyML Device Shipments to Grow to 2.5 Billion in 2030, Up From 15 Million in 2020

https://www.abiresearch.com/press/tinyml-device-shipments-grow-25-billion-2030-15-million-2020/

Running small-scale ML applications on low-power micro-controllers / IoT Devices

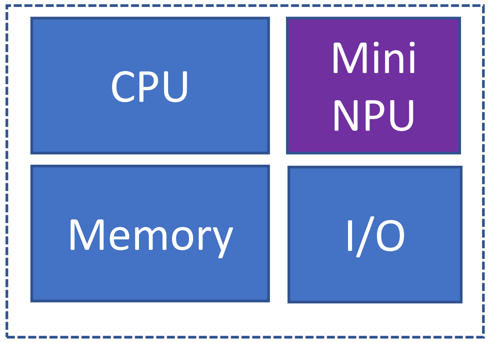
- Example: Radar Gesture Recognition for Touchless User Interfaces
- Example: ML-enhanced Motor Control (PENTA ECOMAI project: ecomai.eu)


Source: Infineon



Platform: Infineon XMC1302
32 MHz Micro-controller CPU
32 kB Flash, 16 kB RAM

ECOMAI HW platform:



So why learn about Advanced Computer Architecture

- Hardware-alone does not give you the performance anymore.
- Any class of system (from embedded systems to servers) now integrates a wide range of dedicated processing:
  - General-purpose: Single/Multi-cores processors (with special instructions: RISC-V)
  - GPUs
  - Accelerators (e.g., so called NPUs/TPUs for machine learning)
  - Even FGPAs (programmable hardware)
- Software must exploit dedicated processing to meet performance/energy targets

Developers need a basic understanding from the low-level digital hardware up to computer architectures as well as of the memory system and interconnect organization.

A2 Pre-Knowledge

Pre-Knowledge

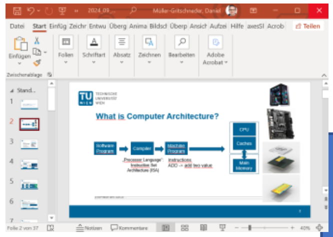
- Digital HW Design (Sequential Digital Circuits)
- Basic of Processors (5 stage pipeline)
- Assembly Programming (ARM, RISC-V,…)
- C/C++ Programming (Functions, Pointers, Arrays, …)
- Basic Linux / Shell Usage

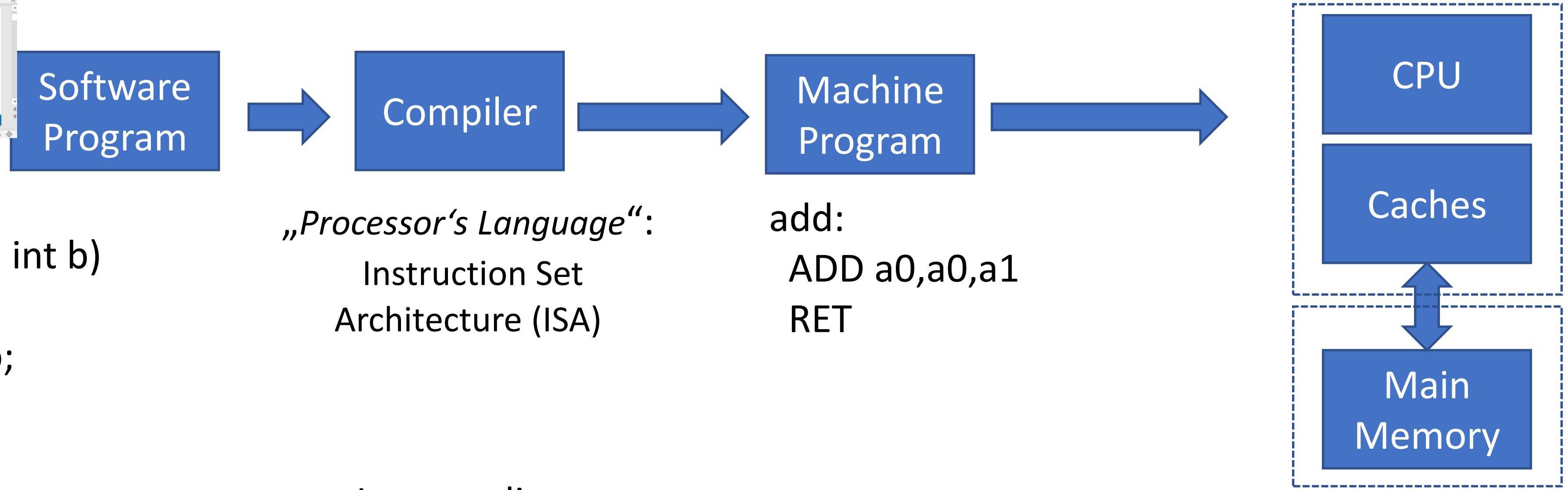
A3 Course Content

Course Content Overview

- Block A: Introduction
- Block B: RISC-V ISA + Compiler Basics
- Block C: Processor Pipelines and Vector Processors
- Lab 1: Vector Processor Architectural Exploration
- Block D: Memory and Multi-Core Processors
- Lab 2: Multi-Core Programming Basics
- Block E: High Level Synthesis
- Lab 3: High Level Synthesis
- Block F: Interconnects
- Block G: Heterogeneous SoCs

Block B: RISC-V ISA and Compilers





```
int add(int a, int b)
{
    return a+b;
}
```

„Processor's Language":
Instruction Set Architecture (ISA)

```
add:
  ADD a0,a0,a1
  RET
```

Intermediate Representation

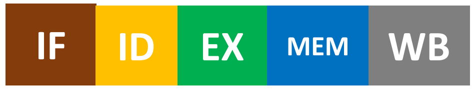
Compiler Optimizations

RISC-V

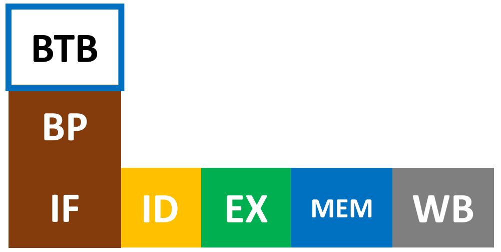
RISC-V Vector

Block C: Processor Pipelines

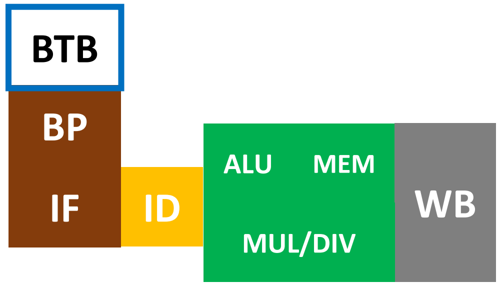
- In-order pipeline
- Five Stages
- Scalar pipeline: CPI >= 1



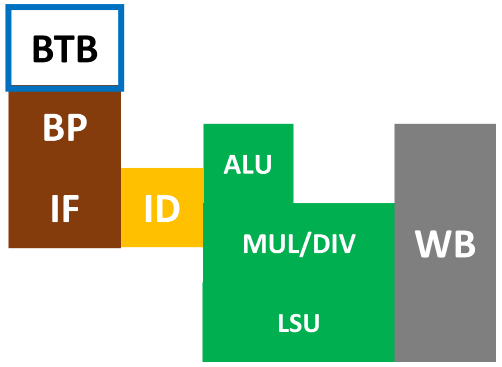
- Branch Predictor (BP)
- Branch Target buffer (BTB)



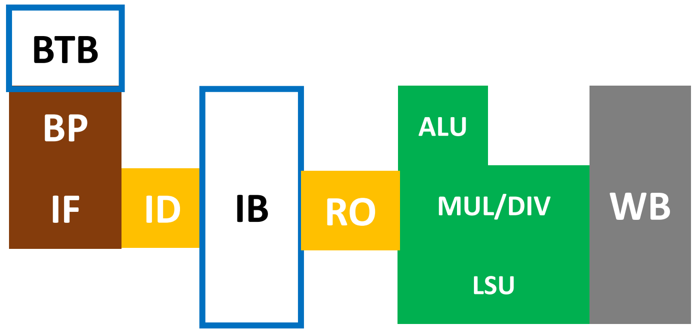
- Multi-cycle
- 4-stage



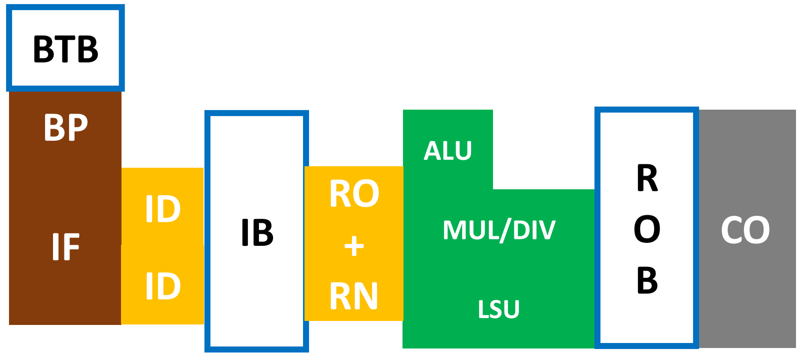
- Multi-cycle
- 4-stage
- Load-Store Unit (LSU)



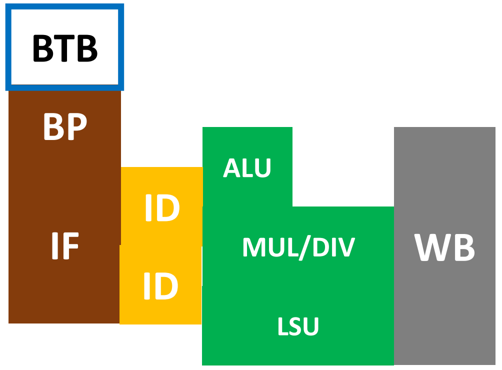
- Instruction Issue Buffer (IB)
- Out-of-order (OoO)



- Superscalar , Reorder-Buffer (ROB)
- Register Renaming

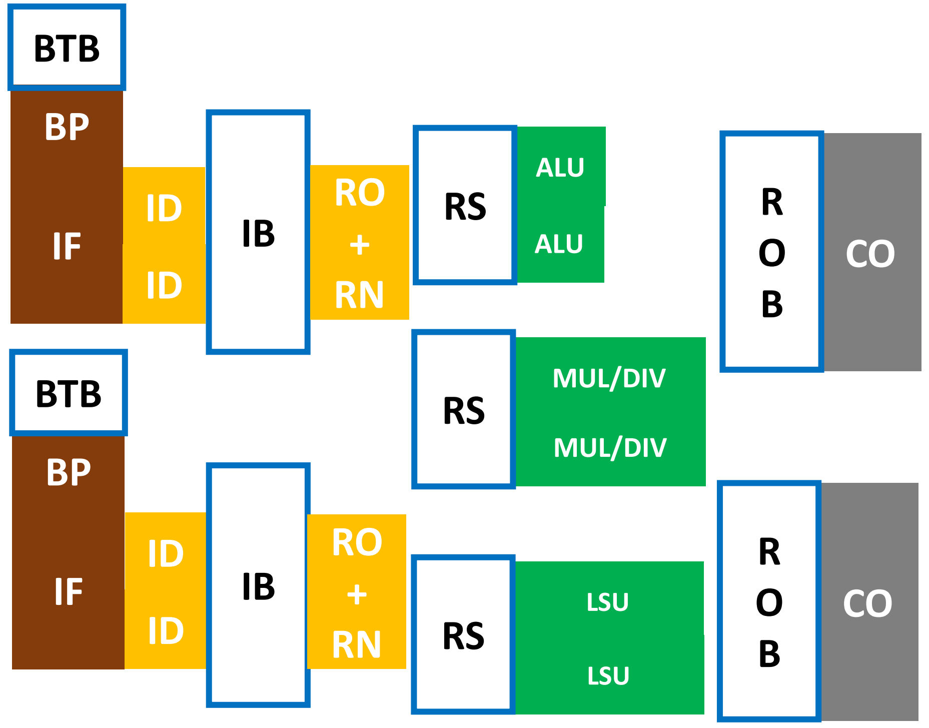


- Very Large Instruction Word (VLIW)

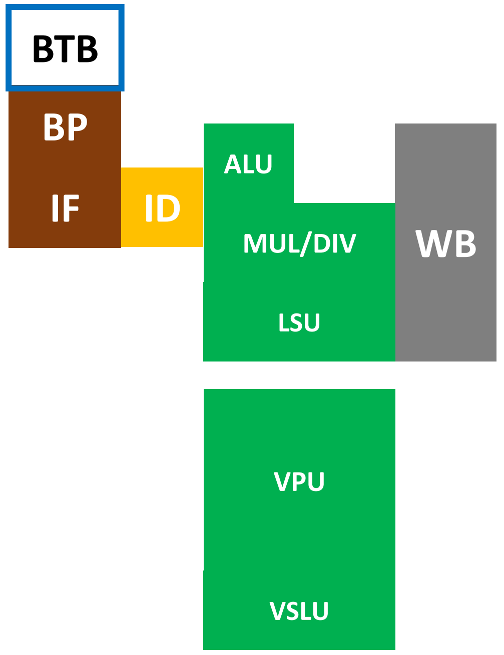


Block C: Multi-threading and VPUs

- Multi Threading

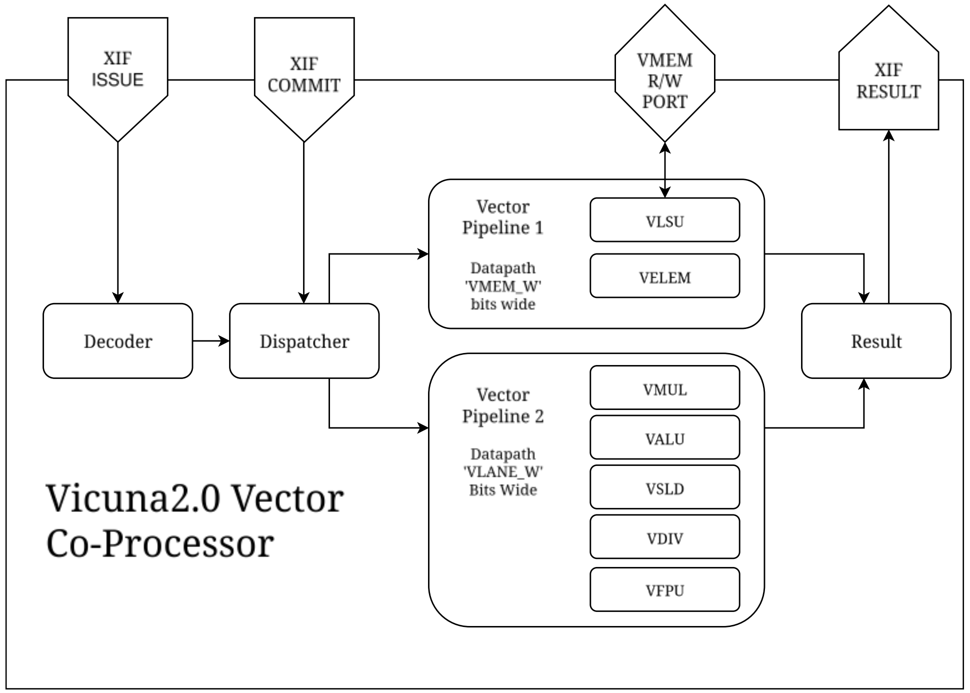


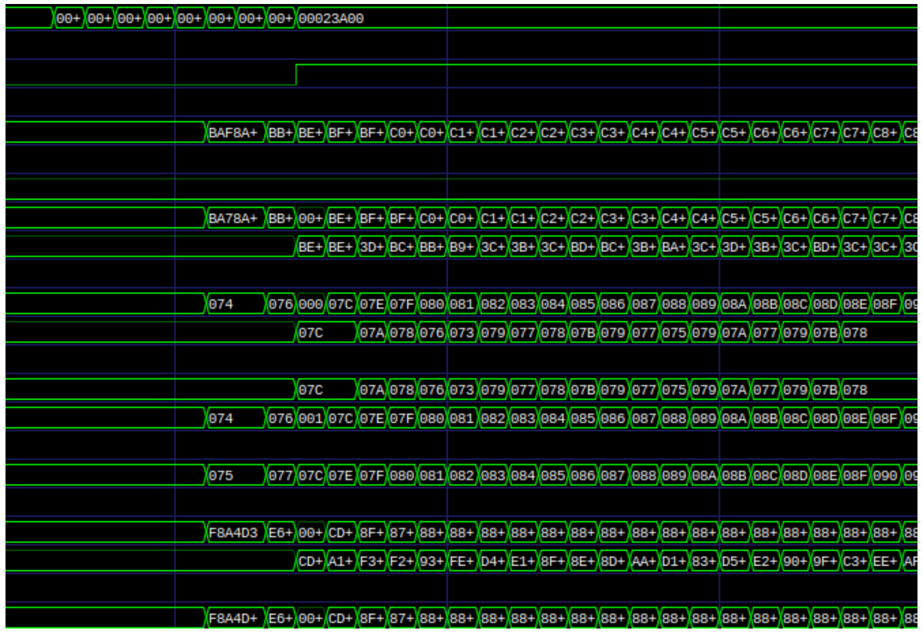
- Vector Processing Unit (VPU)



Lab 1: Vector Processor Architectural Exploration

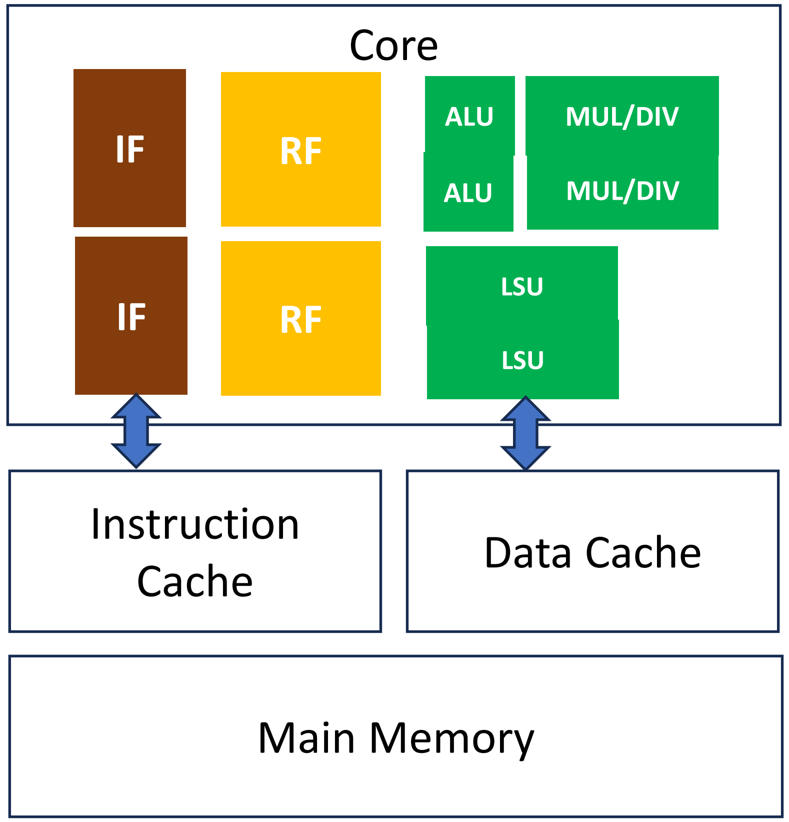
- Customize a vector processor for specific workloads
- Write code optimized for your system
- Attempt to beat performance and resource utilization targets



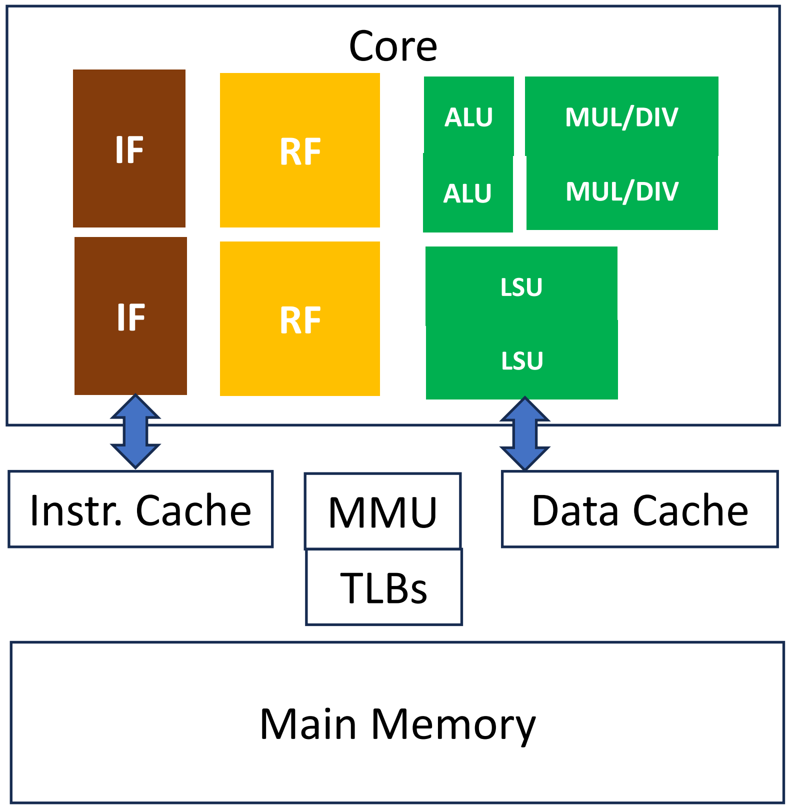


Block D: Memory

- Caches

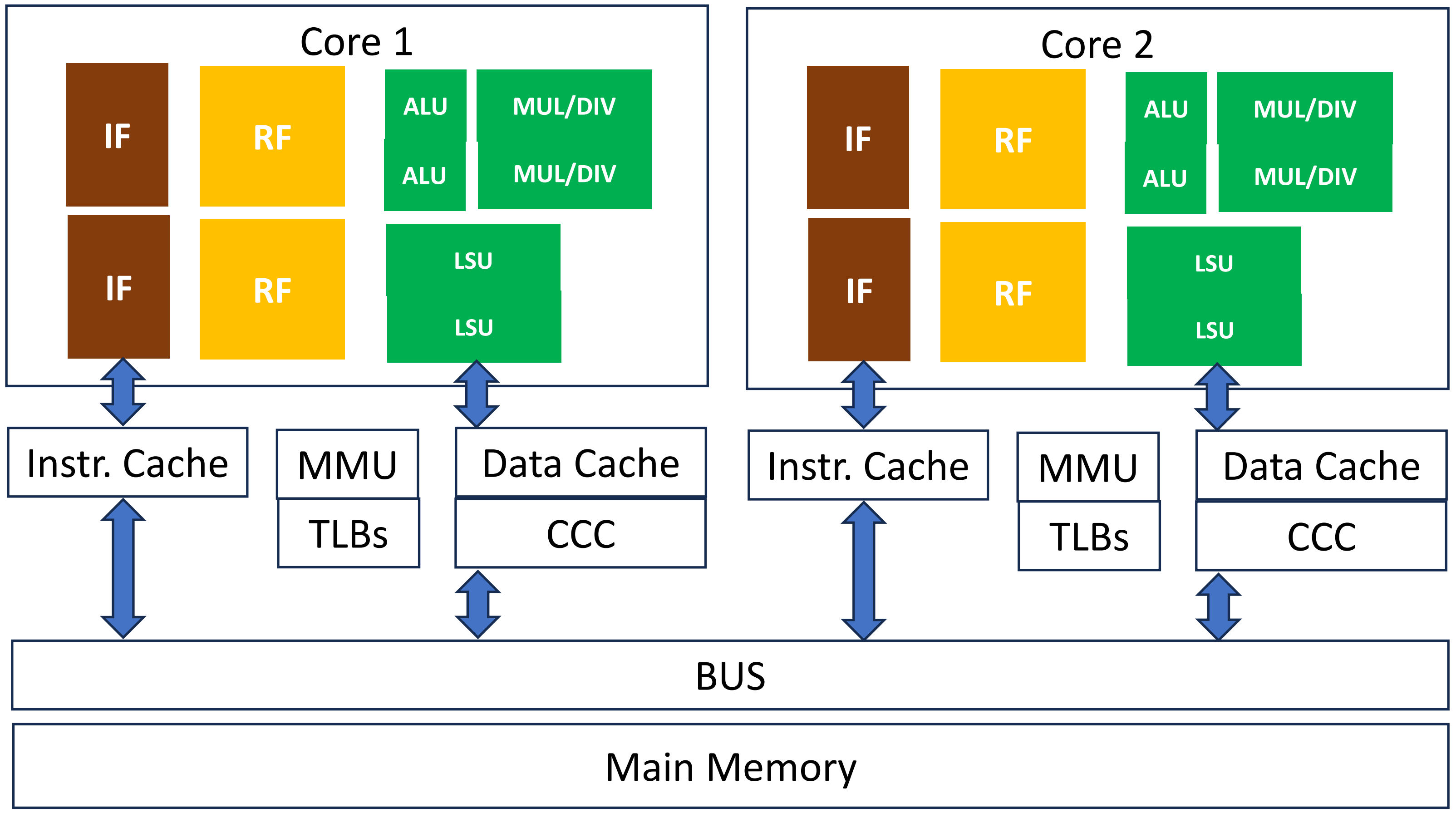


- Virtual Memory



• Block D: Multi-Core

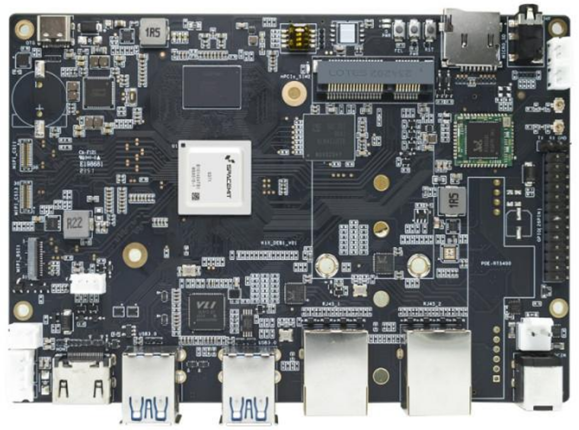
- Multi-Core
- Cache Coherency Controller (CCC)
- On-Chip Interconnect Buses



Lab 2: Multi-Core Programming Basics

- Single Board PC with RISC-V Multi-Core and VPU

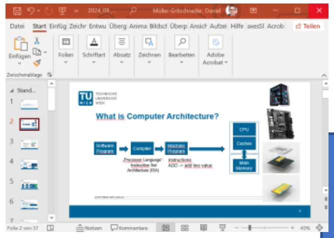
- Target: Use OpenMP and/or pthreads to implement simple multi-threaded program

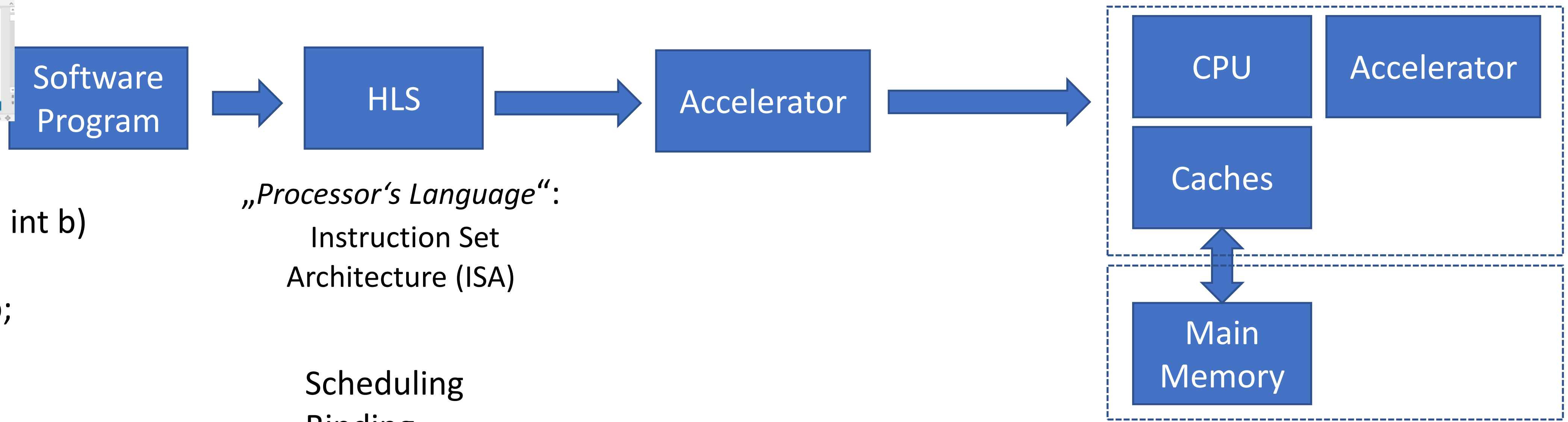


Banana PI F3
SpacemiT K1 Octa-core CPU RISC-V with Vector Unit Linux-capable

Source: Golem.de

Block E: High Level Synthesis (HLS)





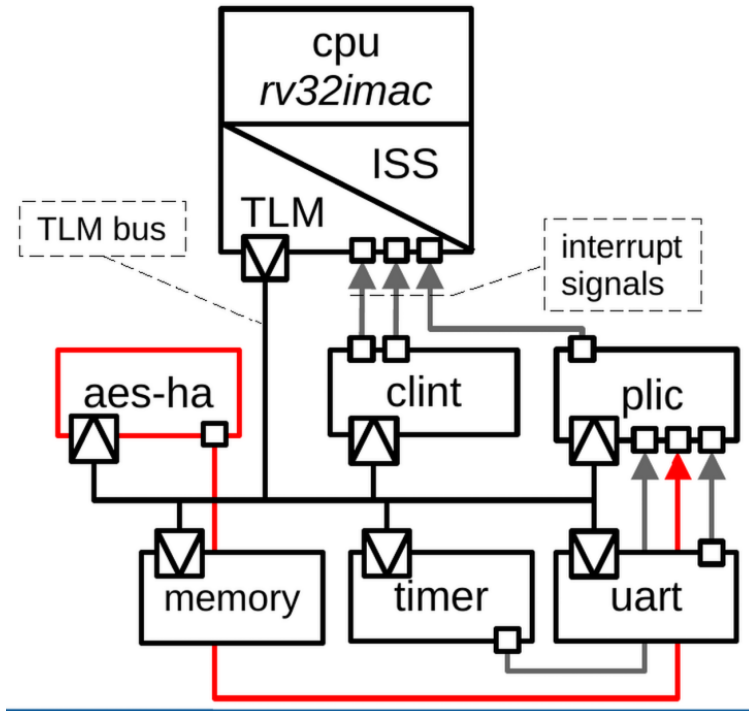
int add(int a, int b)
{
    return a+b;
}

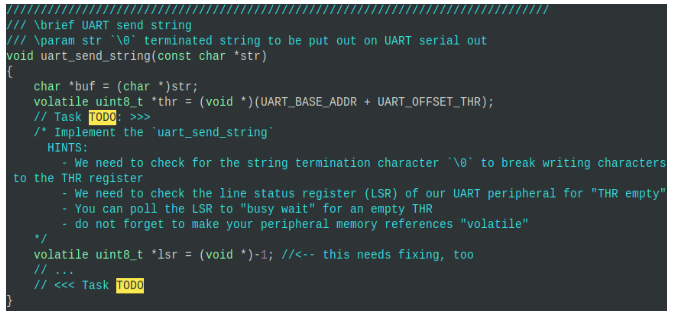
„Processor's Language":
Instruction Set
Architecture (ISA)

Scheduling
Binding
Allocation

Lab 3: High Level Synthesis

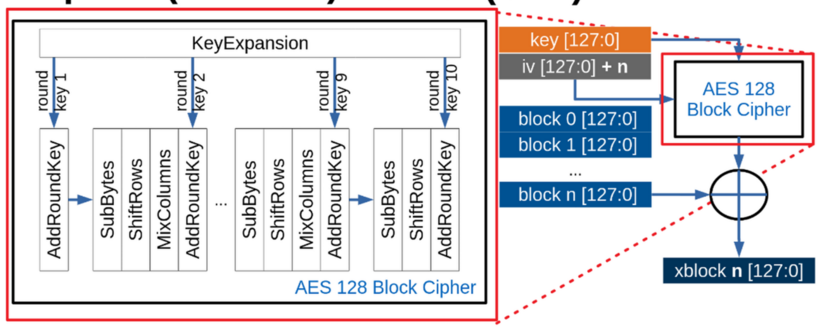
- Virtual Prototype RISC-V Processor to develop SW on a x86 host machine
- Develop small interrupt based drivers
- Use HLS productively for a memory-mapped Cryptography Algorithm (AES)





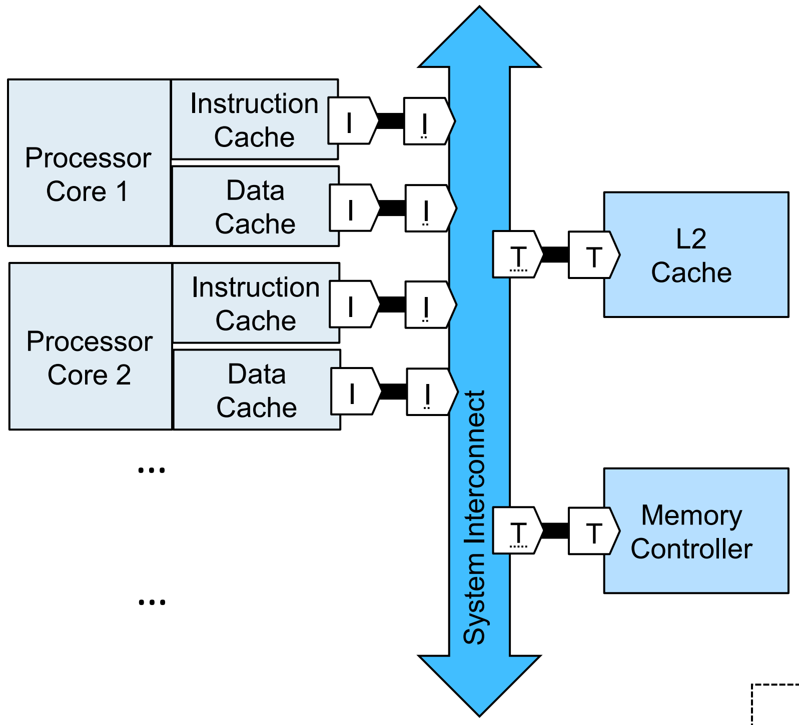
```
////////////////////////////////////////////////////////////////////////////////
/// \brief UART send string
/// \param str `\0` terminated string to be put out on UART serial out
void uart_send_string(const char *str)
{
    char *buf = (char *)str;
    volatile uint8_t *thr = (void *)(UART_BASE_ADDR + UART_OFFSET_THR);
    // Task TODO: >>>
    /* Implement the `uart_send_string`
      HINTS:
        - We need to check for the string termination character `\0` to break writing characters
 to the THR register
        - We need to check the line status register (LSR) of our UART peripheral for "THR empty"
        - You can poll the LSR to "busy wait" for an empty THR
        - do not forget to make your peripheral memory references "volatile"
    */
    volatile uint8_t *lsr = (void *)-1; //<-- this needs fixing, too
    // ...
    // <<< Task TODO
}
```

Cipher: (AES-128) BCMO: (CTR)

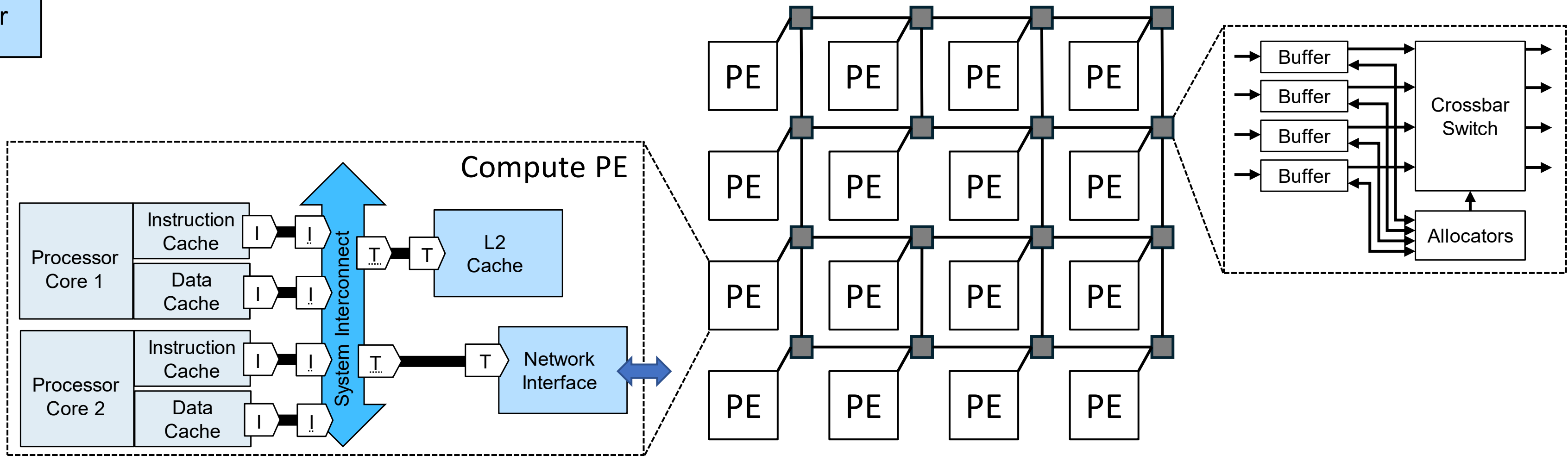


Block F: Interconnects

On-Chip Buses

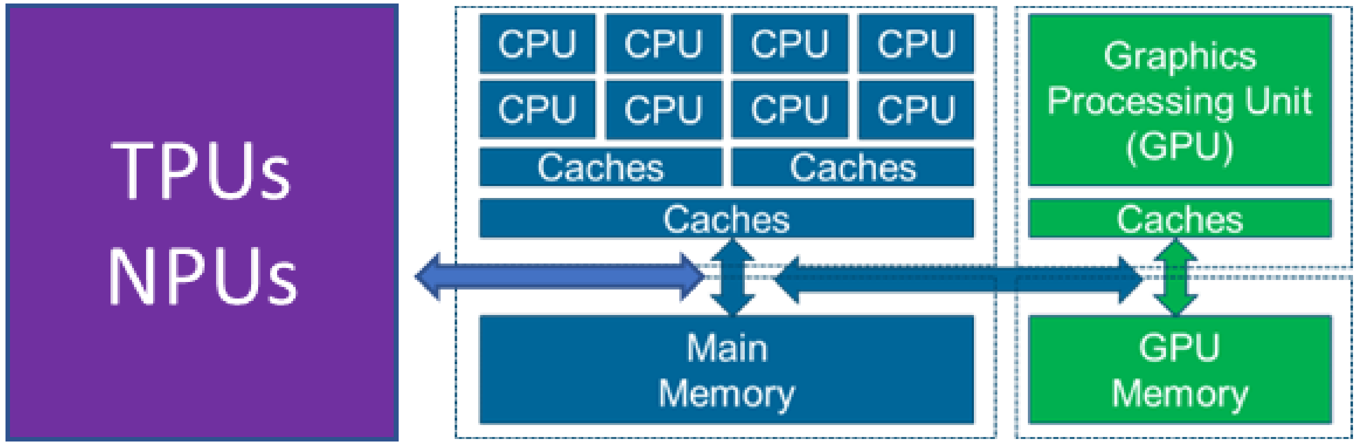


Network on Chip



Block G: Heterogeneous SoCs

- Accelerators (Systolic Arrays)
- GPUs



- Neuromorphic Computing

A4 Course Organization

The Team

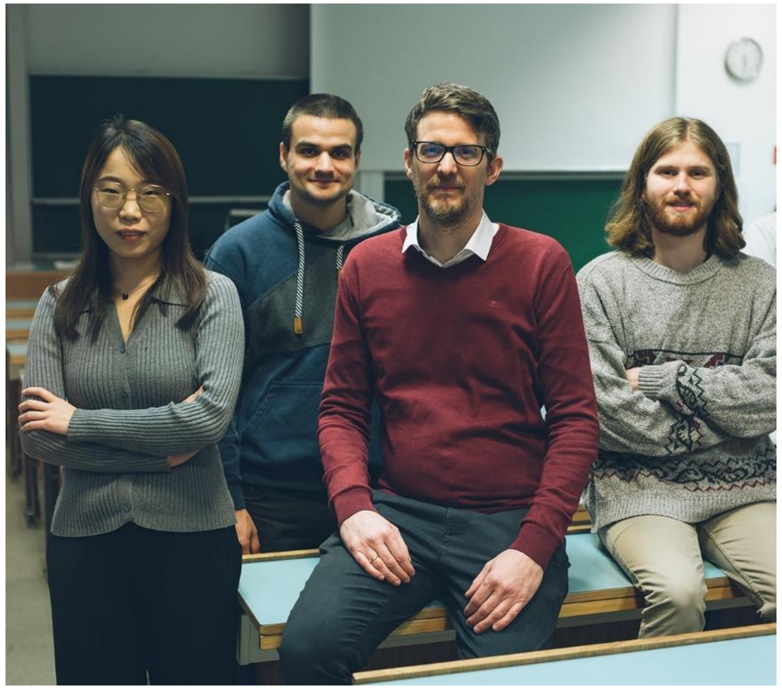
Daniel Müller-Gritschneder

Yang Liu

Parker Jones

Johannes Kappes

Embedded Computing Systems
Treitlstr. 3, 2nd floor



Course Registration

- Registration
  - Start of registration: **already open**
  - End of registration: **see TISS, 29.10.2025**
  - End of de-registration: **see TISS, 13.11.2025**

- Limited to 70 spots, preference to students with mandatory course

Materials

- Provided via TUWEL (tuwel.tuwien.ac.at)
- Lecture Part
    - Lecture slides
    - Links to lecture recordings
    - Exercise sheets
    - Exercise solution sheets
    - Further readings / material
- Lab Material

Organization

- Lectures (attendance not required, but appreciated)
  - Tuesday 10:15 – 11:45, EI 11
  - Thursday 10:15 – 11:45, EI 11
  - Lecture exam (Exercises and Content from the Lecture)

- Lab
  - Introduction Session in the lecture slot (EI 11)
  - Material to work on task either at home, remote or local at TIlab (Treitlstr. 3, 1st floor)
  - Lab exam: Show skill on a new task (individual work in TIlab)

Lab + Exam Schedule

- Lab 1: Vector Processors
  - Introduction: Thu, 30.10.2025
  - Lab exam: Fri, 14.11.2025
- Lab 2: Multi-Core Programming Basics
  - Introduction: Thu, 13.11.2025
  - Lab Exam: Fri, 05.12.2025
- Lab 3: High Level Synthesis
  - Introduction: Thu, 04.12.2025
  - Lab exam: Fri, 19.12.2025
- Lecture Exam:
  - Exam prep session: Thu, 20.01.2026
  - Final: Fri, 23.01.2026
  - Repetition: Fri, 06.03.2026

Assessment

- Lab part:
  - One lab test per lab (TIlab)
  - Each lab test: 15 points
  - Max 40 points for all labs
  - Lab test points = min(40,(Lab1+Lab2+Lab3))
- Lecture part:
  - Final lecture exam: 60 points
  - Repetition exam (if you missed or failed final exam)
- Overall = Lab test points + Lecture exam points
  - 100 points max.
  - 50 points to pass the course

Questions

- Please use the TUWEL forums.

- Individual questions - Mailing list: aca@ecs.tuwien.ac.at

Enjoy the Semester!

We wish you a successful semester!