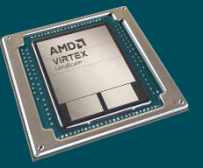
UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TECHNISCHE INFORMATIK

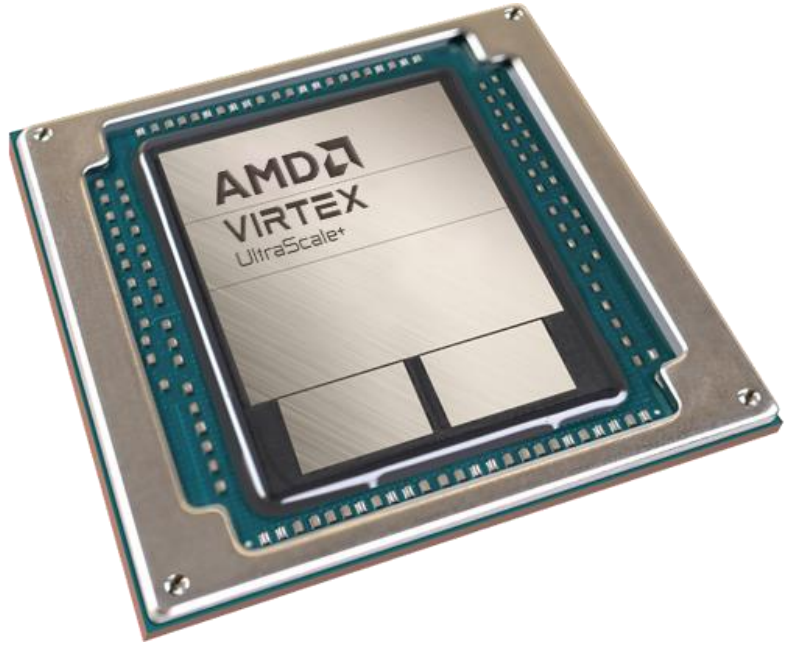
FPGA Architecture and Design-Flow

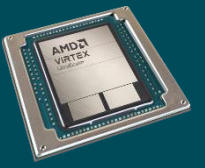
From AI model to circuit

Christopher Blochwitz, M. Sc.
Institute of Computer Engineering
University of Lübeck

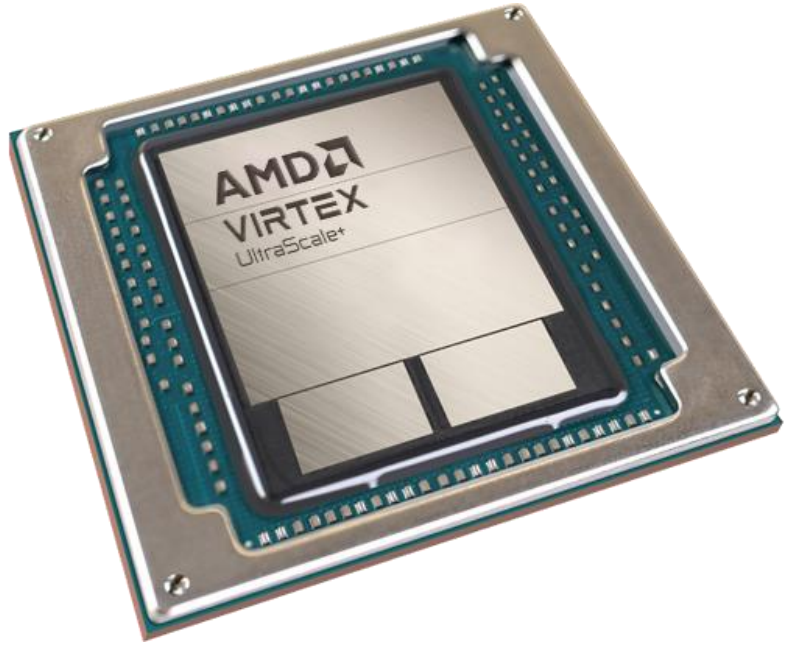


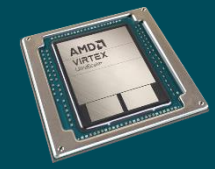
Content

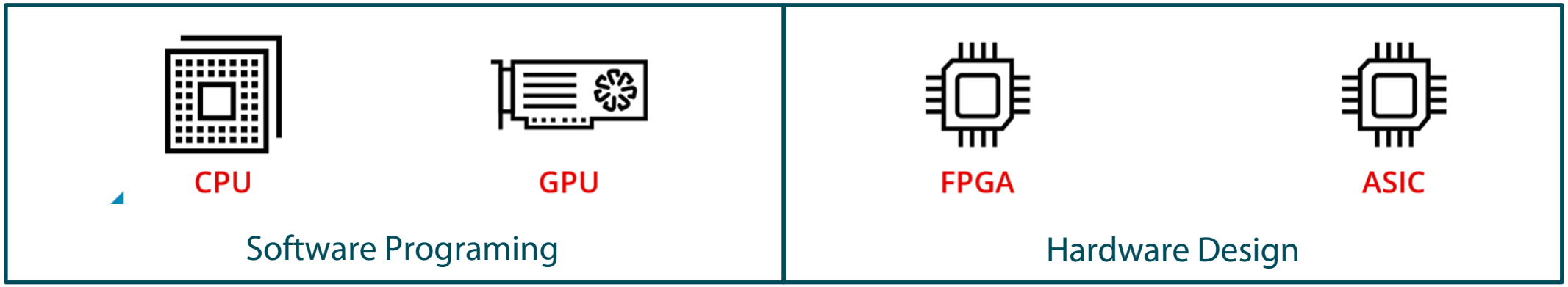


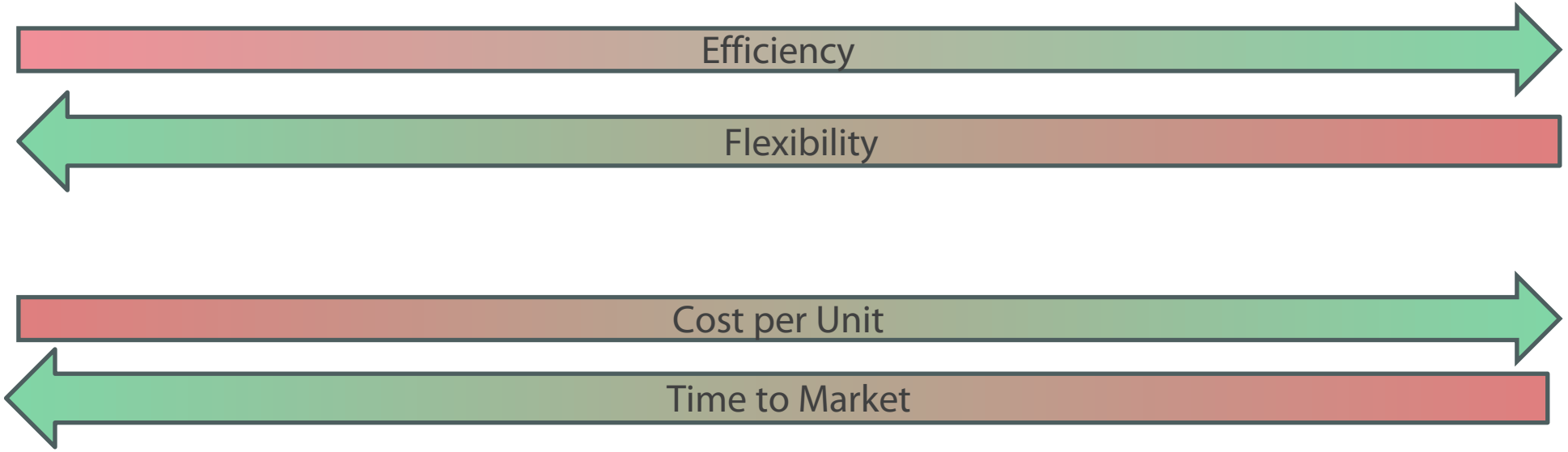


Content






CPU
GPU
FPGA
ASIC
Software Programing
Hardware Design


Efficiency
Flexibility
Cost per Unit
Time to Market

IoT, Edge, Datacenter…

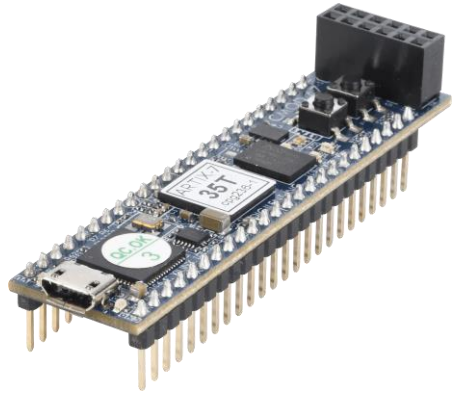
Wide range of Applications

Wide range of FPGA families and parameters:

- Performance
- Low Power
- High and Low-Capacity
- Peripheral
- Technology: 7nm, 16nm …

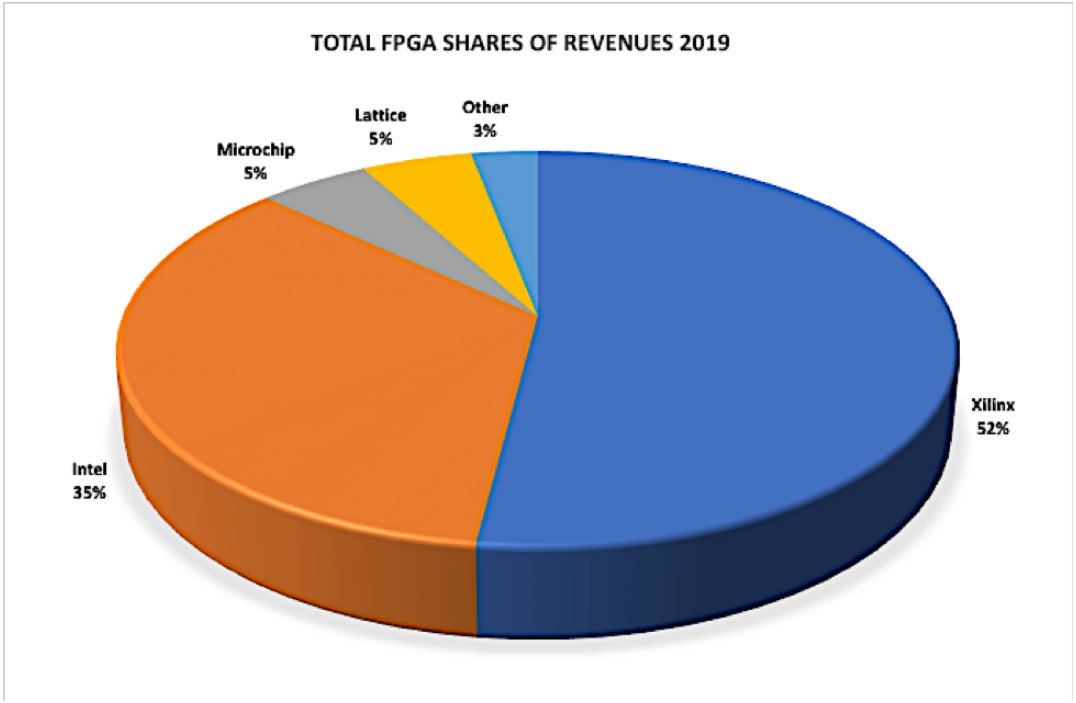
Edge – Kira SOM

Datacenter – Alveo



IoT – Cmod

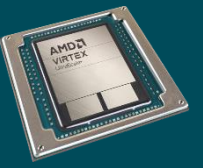
FPGA Market



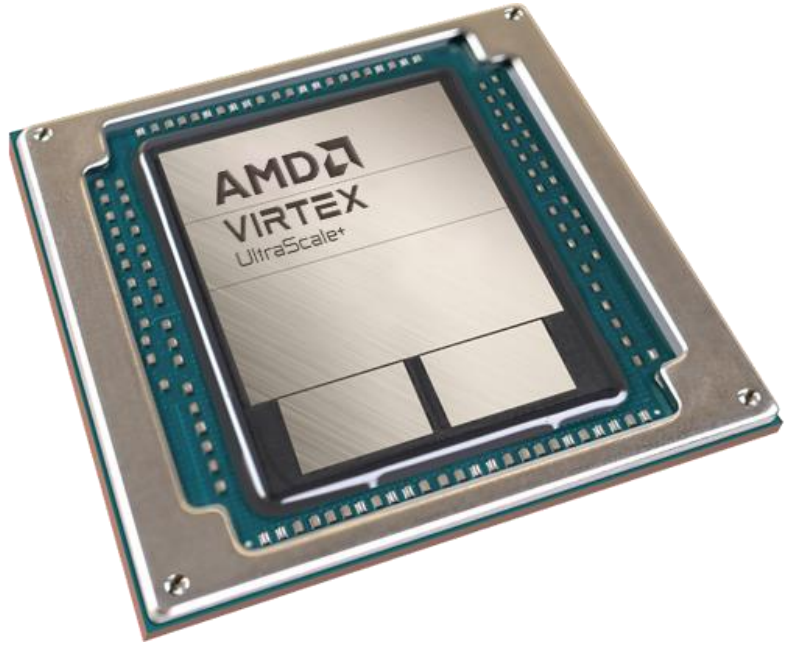
→ Focus on Xilinx (especially in research)

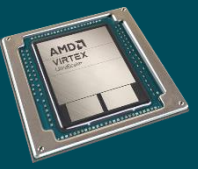
AMD acquired Xilinx for $49B in 2022

(Source: Achronix / https://www.embedded.com/ai-accelerator-ip-rolls-for-fpgas/, https://seekingalpha.com/article/4378735-amd-and-xilinx-prize-is-versal-acap-not-fpgas)



Content

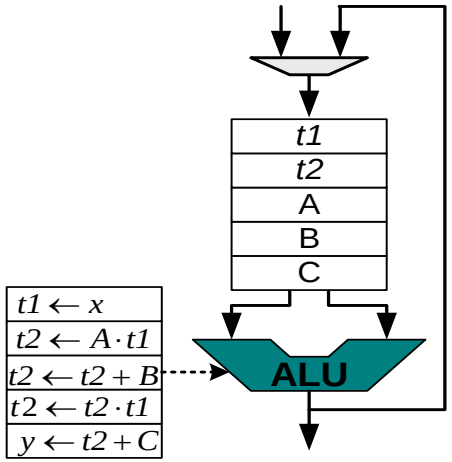




Field Programmable Gate Arrays (FPGA)

Arithmetic principles, for example:

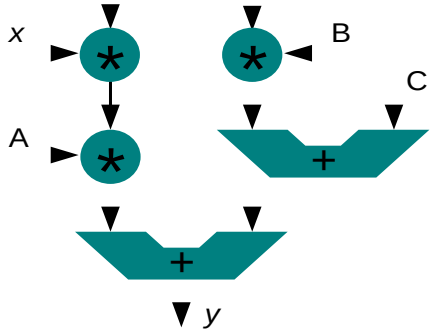
Sequential Computing

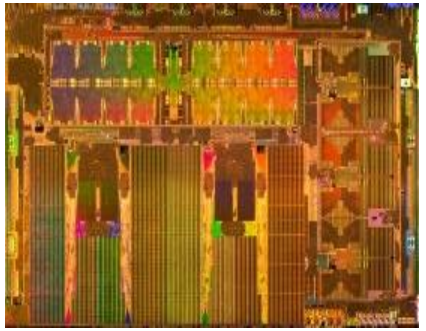


$$y = A \cdot x^2 + B \cdot x + C$$

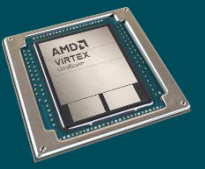
CPU / DSP

Distributed Computing





ASIC



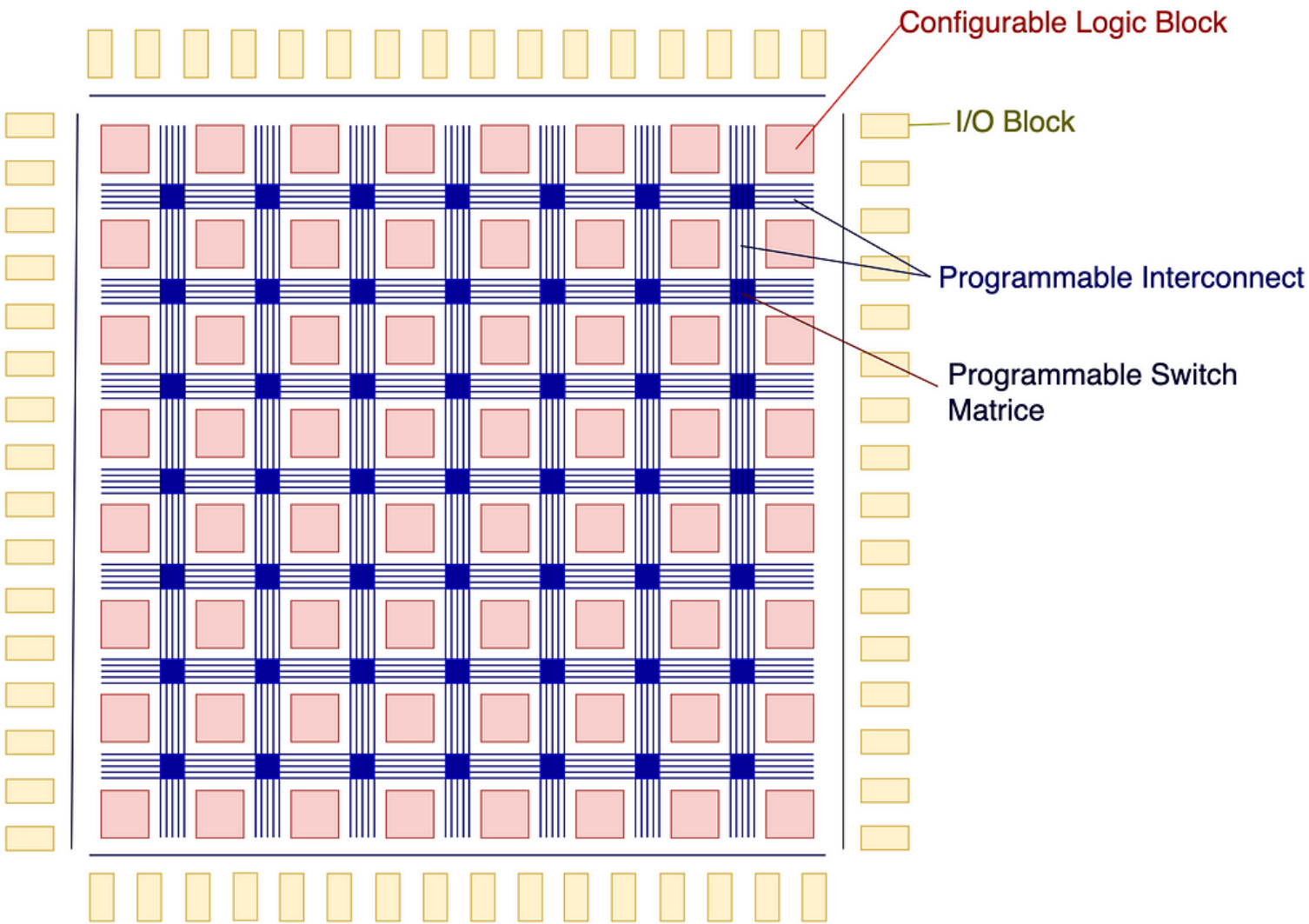
FPGA

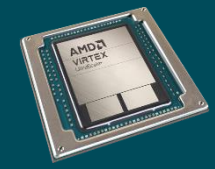
Field Programmable Gate Array

- the End-User programs circuit

Field Programmable Gate Array

- Array of Logic Gates
- Routing Resources for Wire-Connection

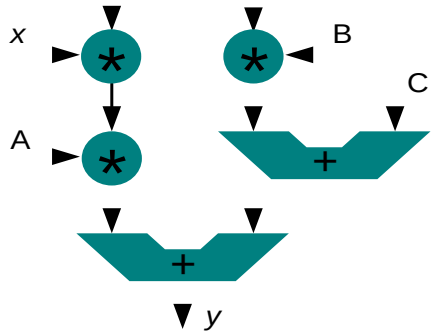


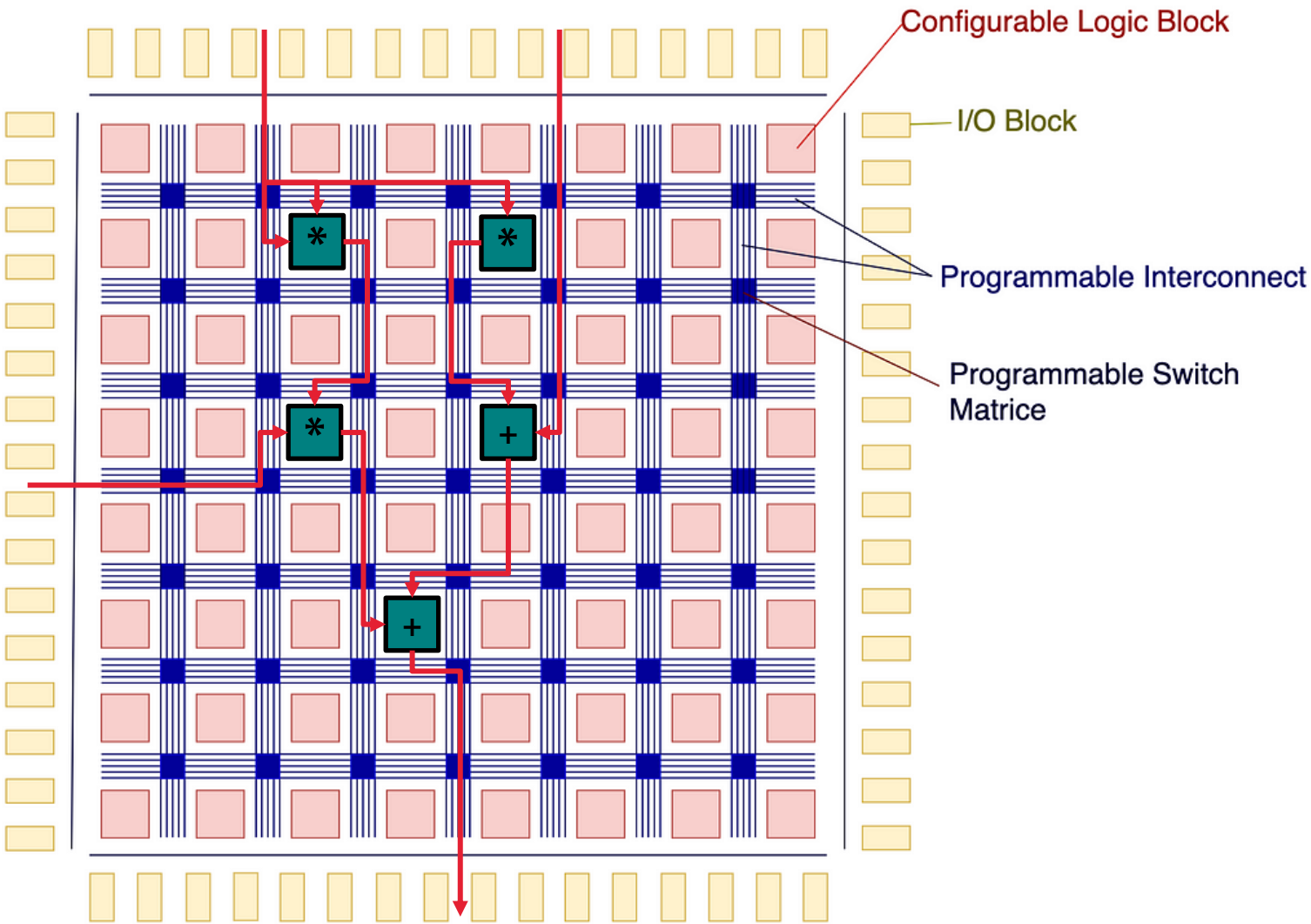


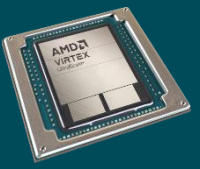
FPGA

Field Programmable Gate Array

Hardware structure similar to ASIC







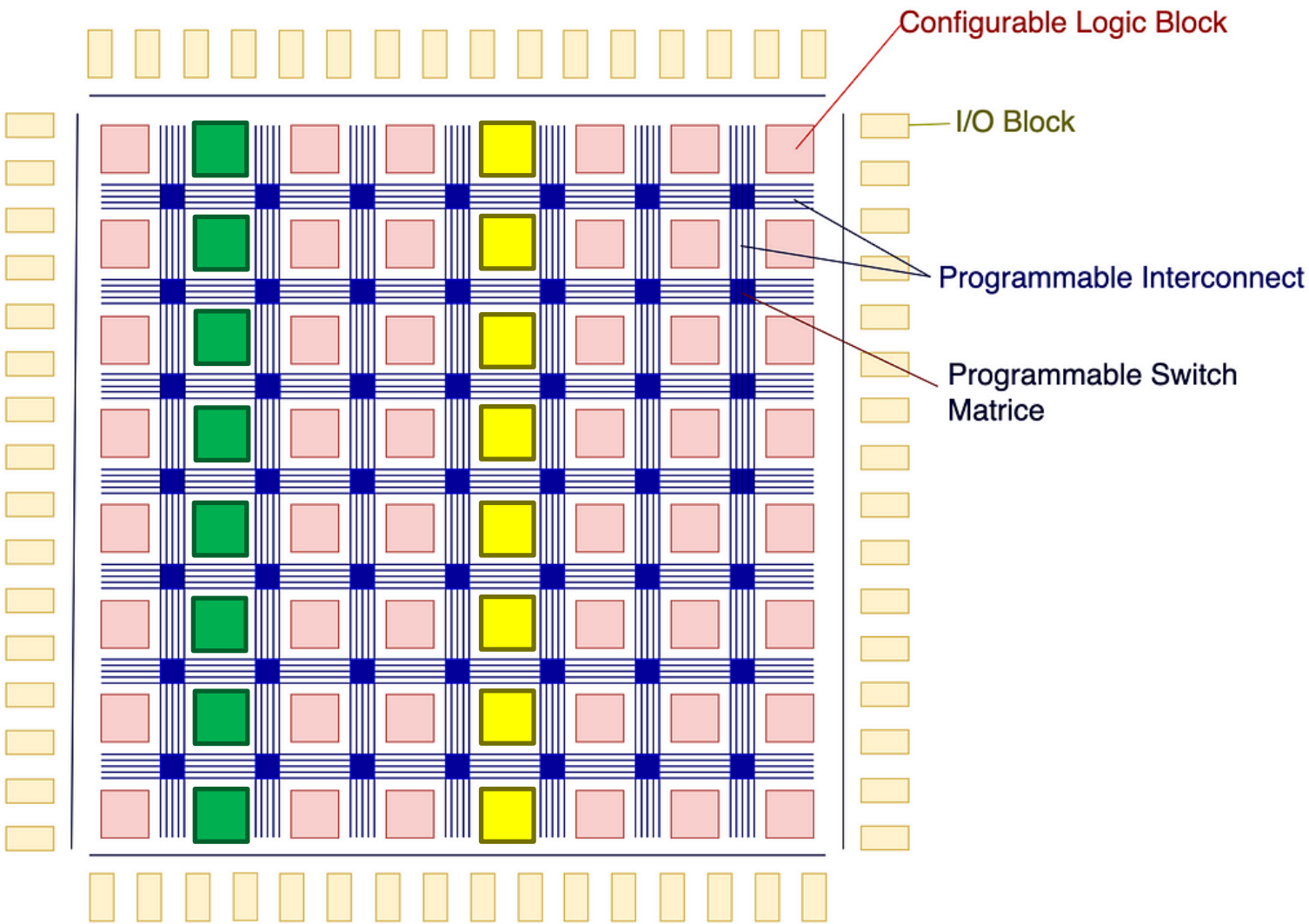
FPGA

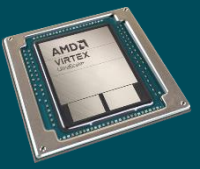
Heterogeneous Design

- Digital Signal Processor
- Embedded Block RAM

→

higher flexibility to
Application spcific Requirments





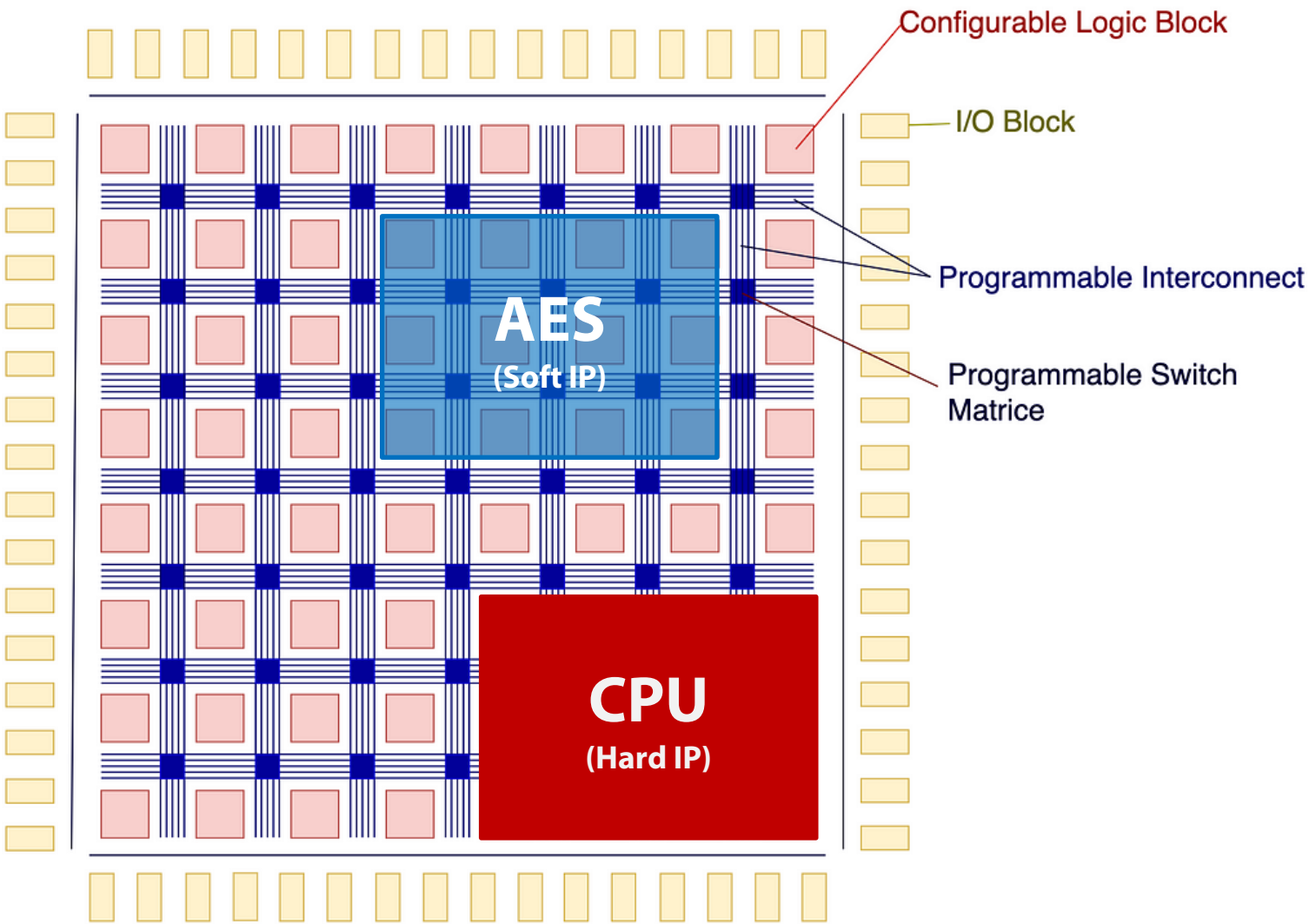
FPGA

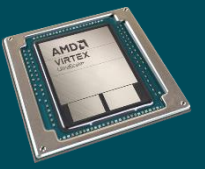
Soft IP (Intellectual Properties)

- Function Block as part of the FPGA Design

Hard IP

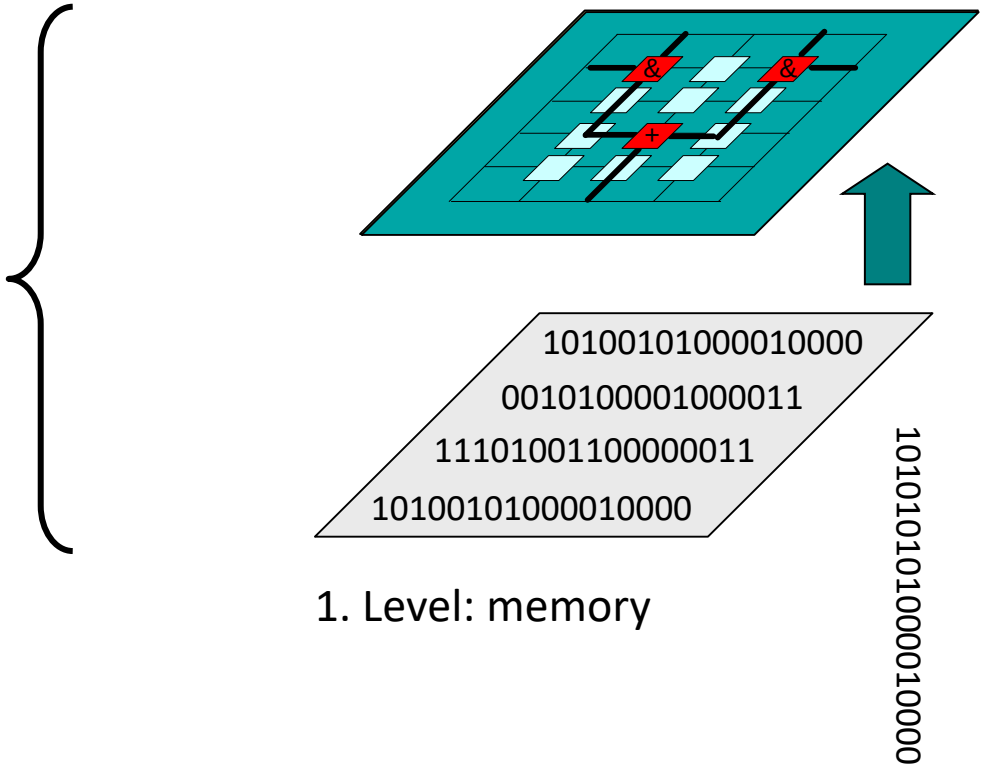
- Function Block integrated in silicon
- e.g., CPUs, Memory Controller, PCIe
- Better performance and efficiency
- reducing complexity during synthesis

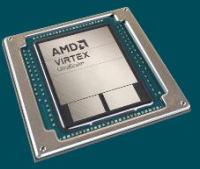




Bitstream to Circuit

Inside FPGA

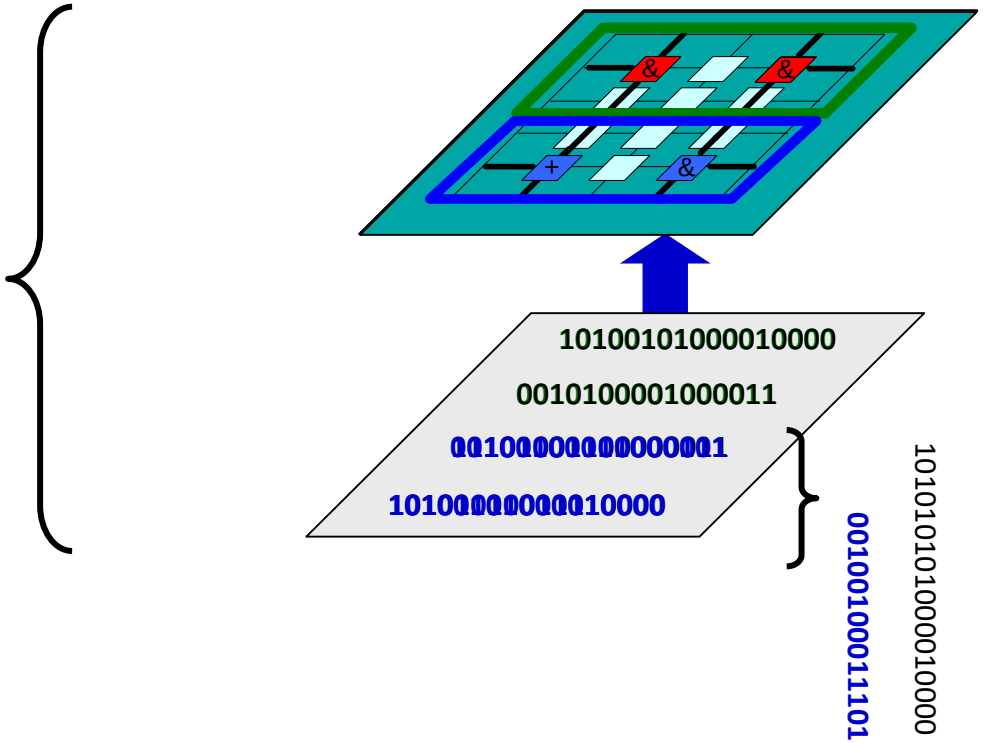




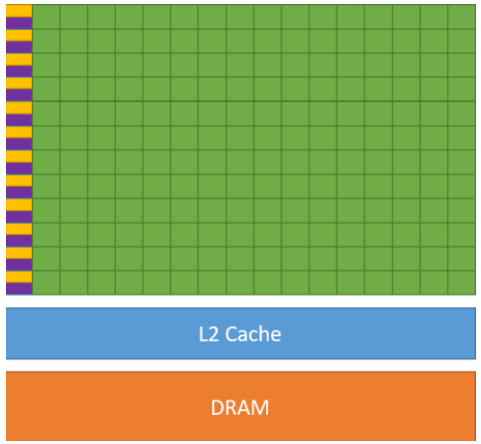
Bitstream to Circuit

Partial Reconfiguration

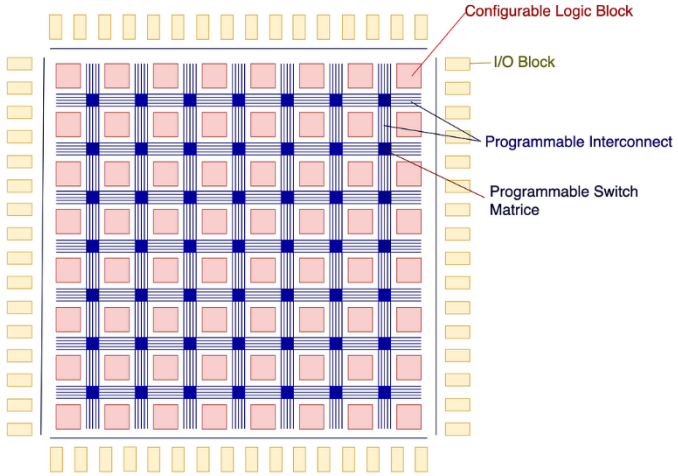
Exchange of configuration data at runtime ➔ Partial dynamic reconfiguration



GPU vs FPGA vs SoC vs ACAP



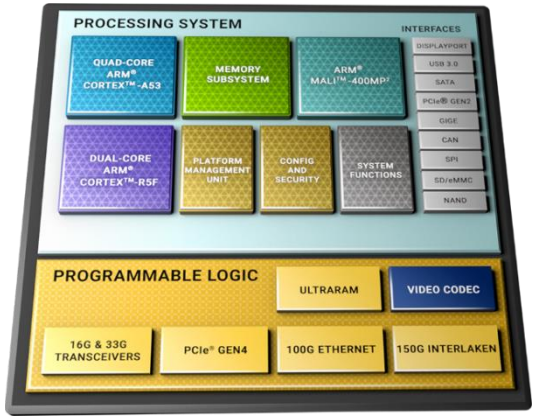
GPU



FPGA

Families:

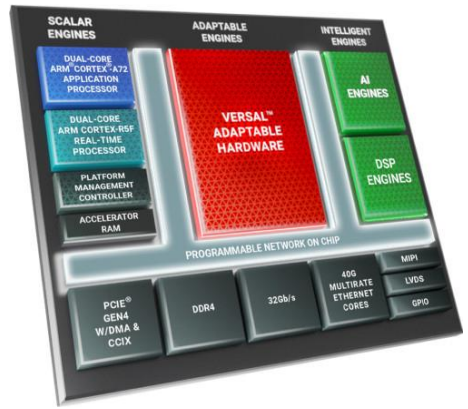
Spartan
Artix
Kintex
Virtex



SoC - System on Chip

Family:

Zynq



ACAP - Adaptive Compute Acceleration Platforms

Families:

Versal Prime
Versal Premium
Versal AI

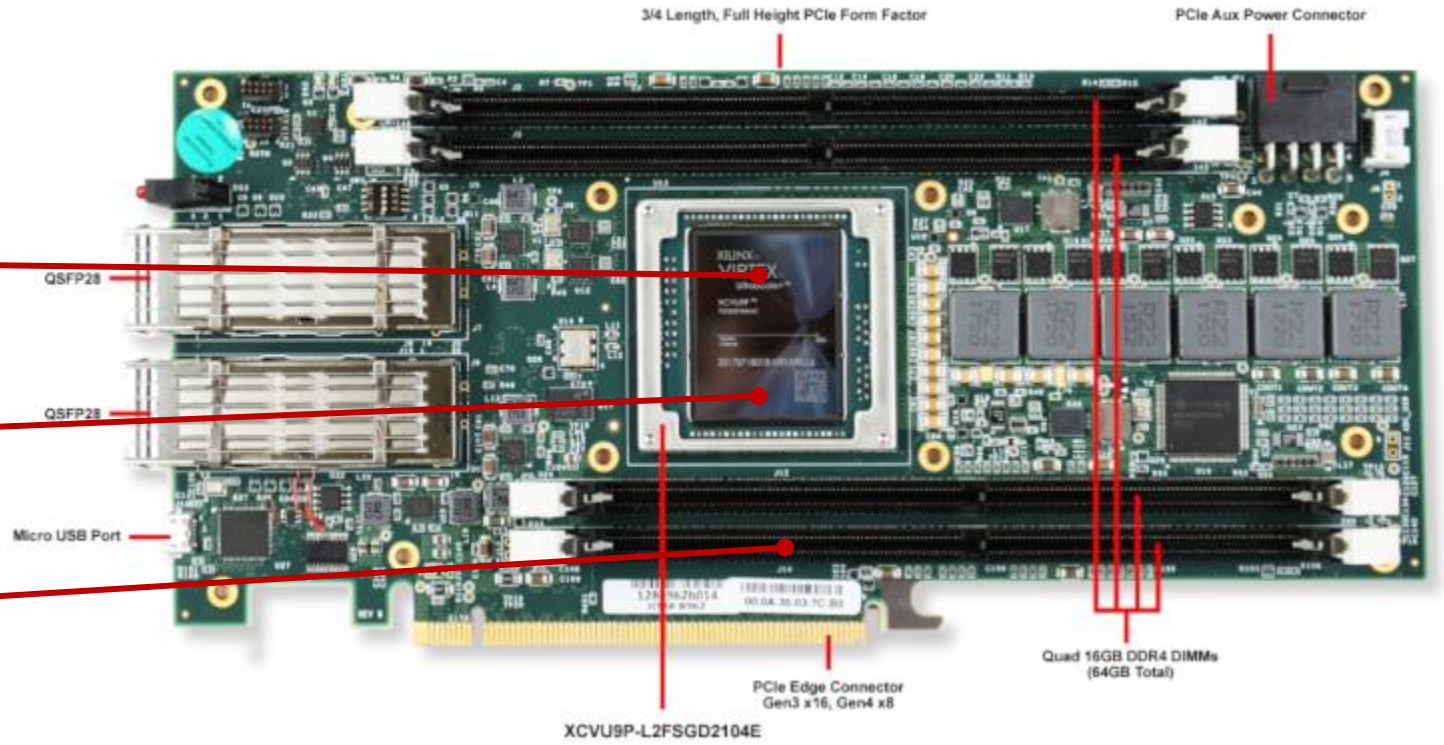
Memory Hierachy

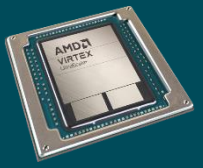
On-chip memory for enhance AI performance

Embedded Block RAM [MB] (BRAM)

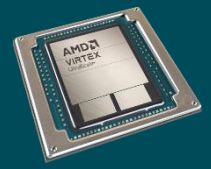
High Bandwidth Memory [GB] (HBM)

DDR Memory [GB]



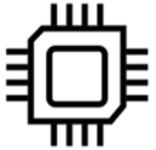


… so what about AI?

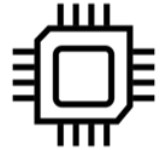


CPU

GPU



FPGA

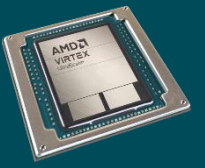


ASIC

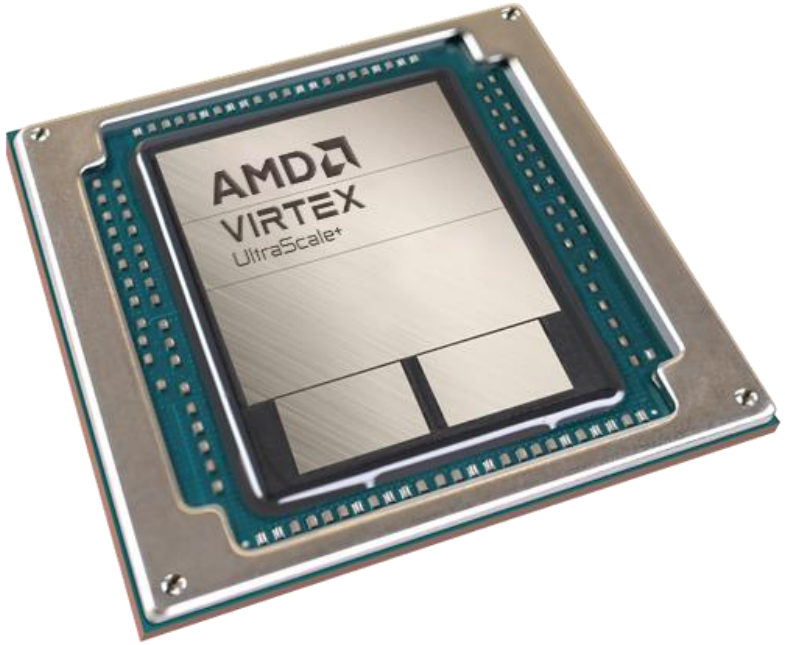
	CPU	GPU	FPGA	ASIC
Training	Poor	Production Ready	Not efficient	Potentially best, but not available
Inference	Poor	Average	Best	Not inference focused

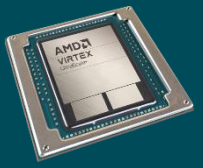
Training:

- mainly Matrix multiplication
- GPUs are already optimized for this Task



Content





FPGAs and Relevance to AI

1.Parallel Processing Capability:

FPGAs are inherently parallel devices aligns well with the parallel nature of many AI algorithms.

2.Customizable Hardware Acceleration, Adaptability and Upgradability:

AI tasks specific hardware accelerators, better performance compared to general-purpose processors.

3.Low Latency, High Throughput, Real-Time Capability:

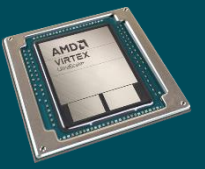
Well-suited for low-latency and real-time applications - predictable latency.

4.Energy Efficiency:

Ability to implement only the necessary logic for a specific task reduces power consumption → edge computing.

5.Scalability:

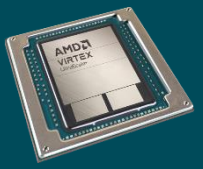
FPGAs offer scalability in terms of parallelism and resources.



FPGA Design Flows for AI

1. **AI Model Definition and Training:**
   - Choose a deep learning framework such as TensorFlow, PyTorch, or Keras.
   - Design and train the neural network architecture for your specific application.
2. **Convert the Model to FPGA-Compatible Format:**
   - Convert the trained model into a format suitable for deployment on FPGAs.
   - Common formats include ONNX (Open Neural Network Exchange) or Xilinx-specific formats like N2Cube.
3. **FPGA Accelerator Design:**
   - Design and implement custom AI accelerator blocks using HLS (High-Level Synthesis) tools or RTL (Register-Transfer Level) coding.
   - Utilize Xilinx AI IP blocks for common AI operations.
4. **FPGA Synthesis and Optimization**

Die Accelerator Design is the hard Part!



Accelerator Design

Approach 1:

Predefined Hardware Design

Configuration of Processing Unit

- Soft IP - CPU/DPU
- Connectivity

→ Running Software / Instructions

Approach 2:

High Level Synthesis (HLS)/ Bare FPGA Design

AI Model to Hardware Compilation

- HLS Synthesis to RTL/Logic Level
- Logic Synthesis

→ Dataflow

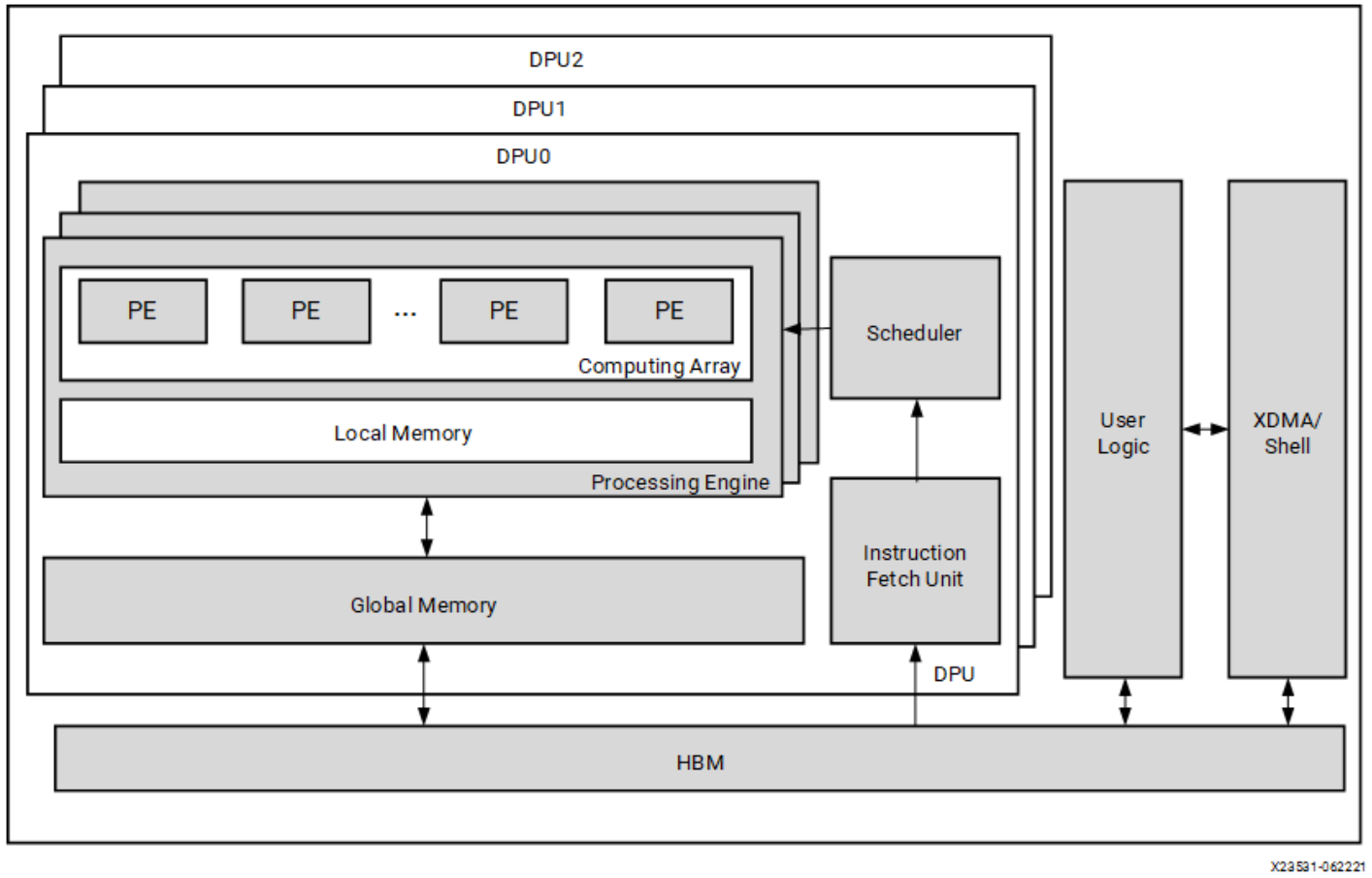
Predefined Hardware Design

Examples:

- **RISC-V**
  + Instruction extension

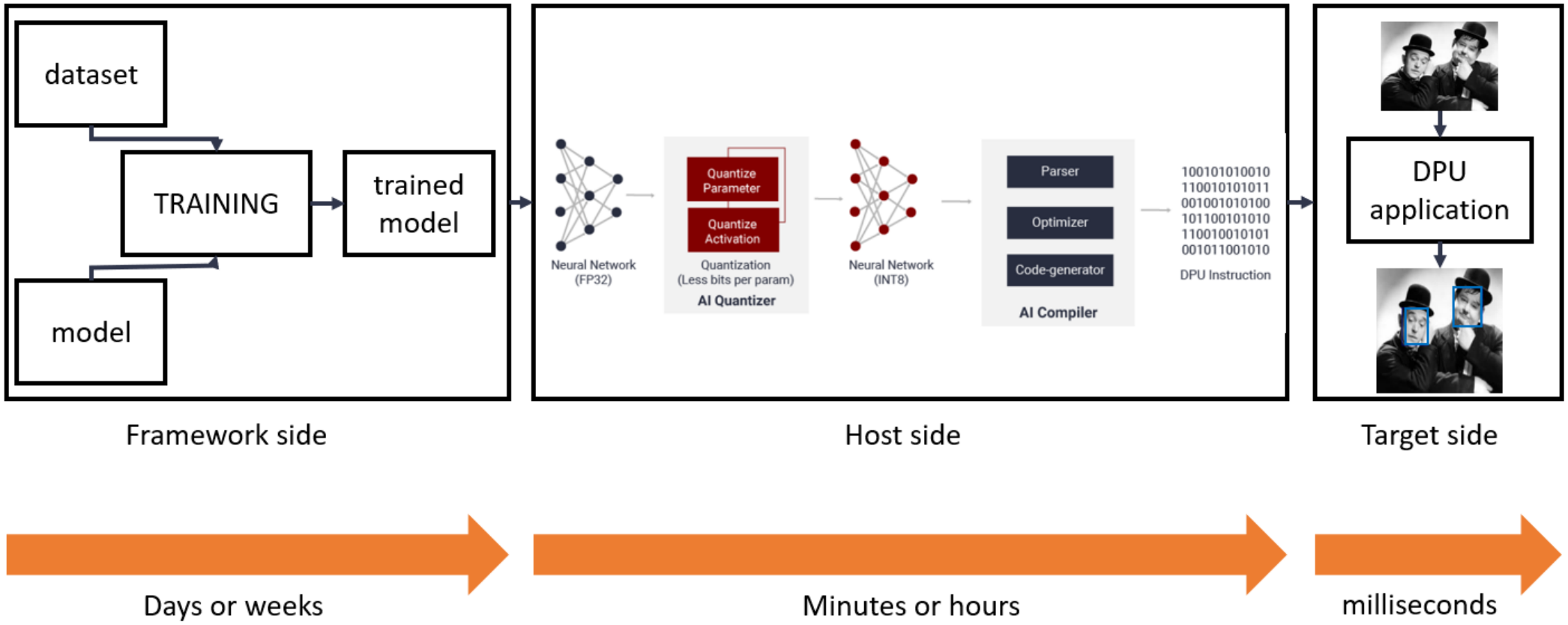
- **RISC-V**
  + embedded Accelerator

- **DPU Design**
  → Xilinx Vitis AI



Predefined Hardware Design

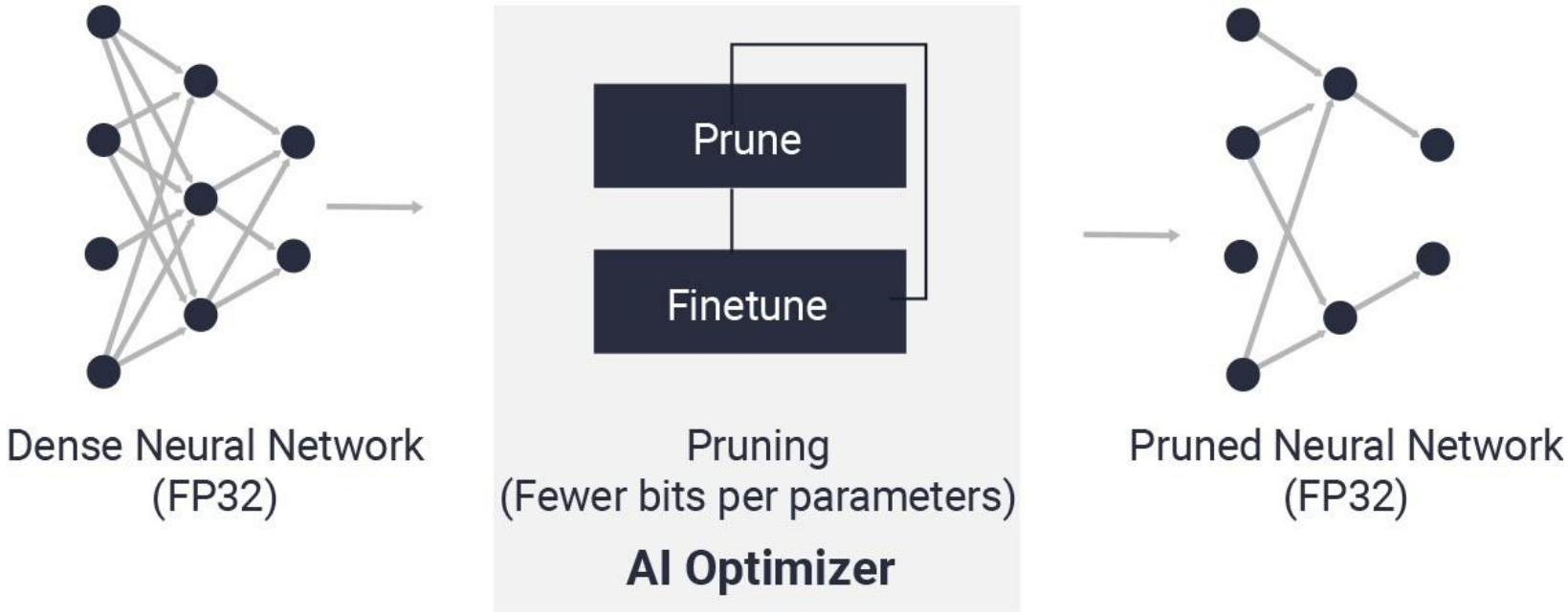
Design Flow for Xilinx Vitis AI



Predefined Hardware Design

Design Flow for Xilinx Vitis AI (Tool)

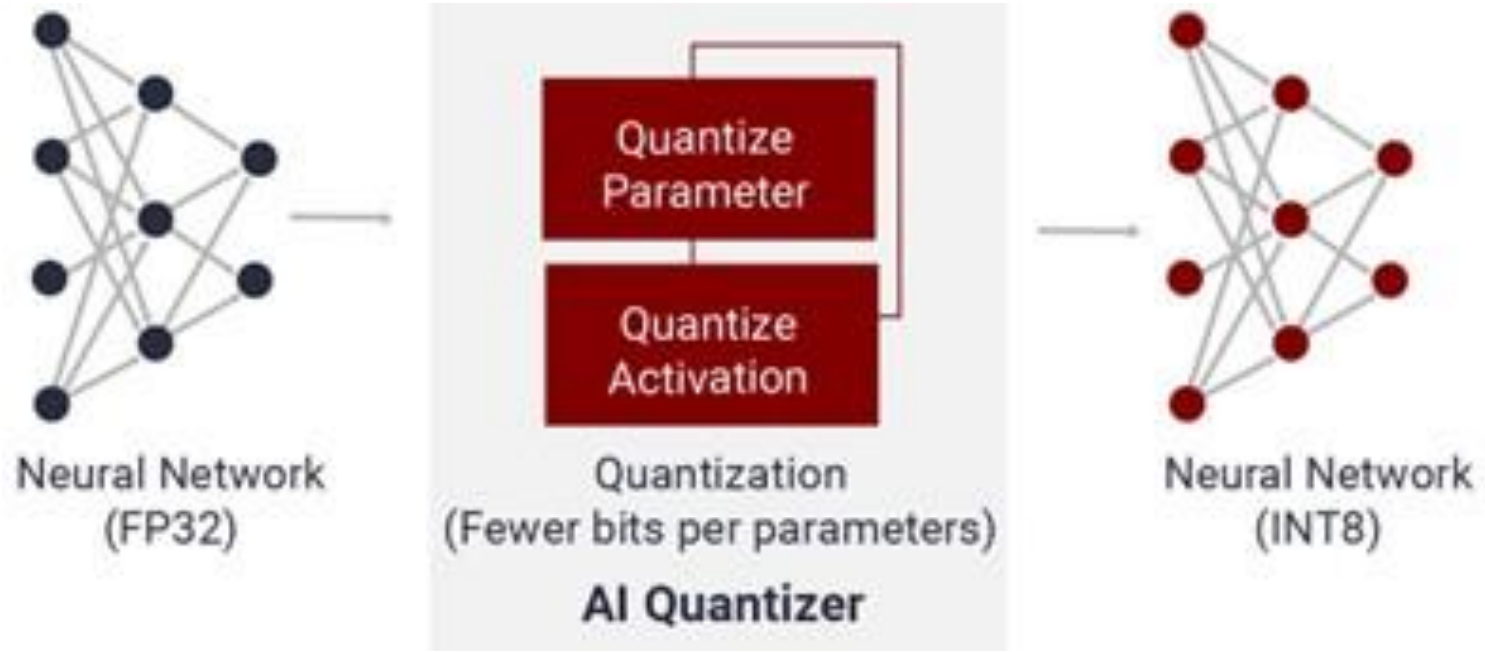
- **AI Optimizer**

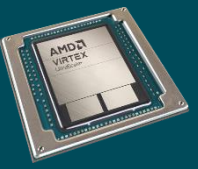


Predefined Hardware Design

Design Flow for Xilinx Vitis AI (Tool)

- **AI Optimizer**
- **AI Quantizer**

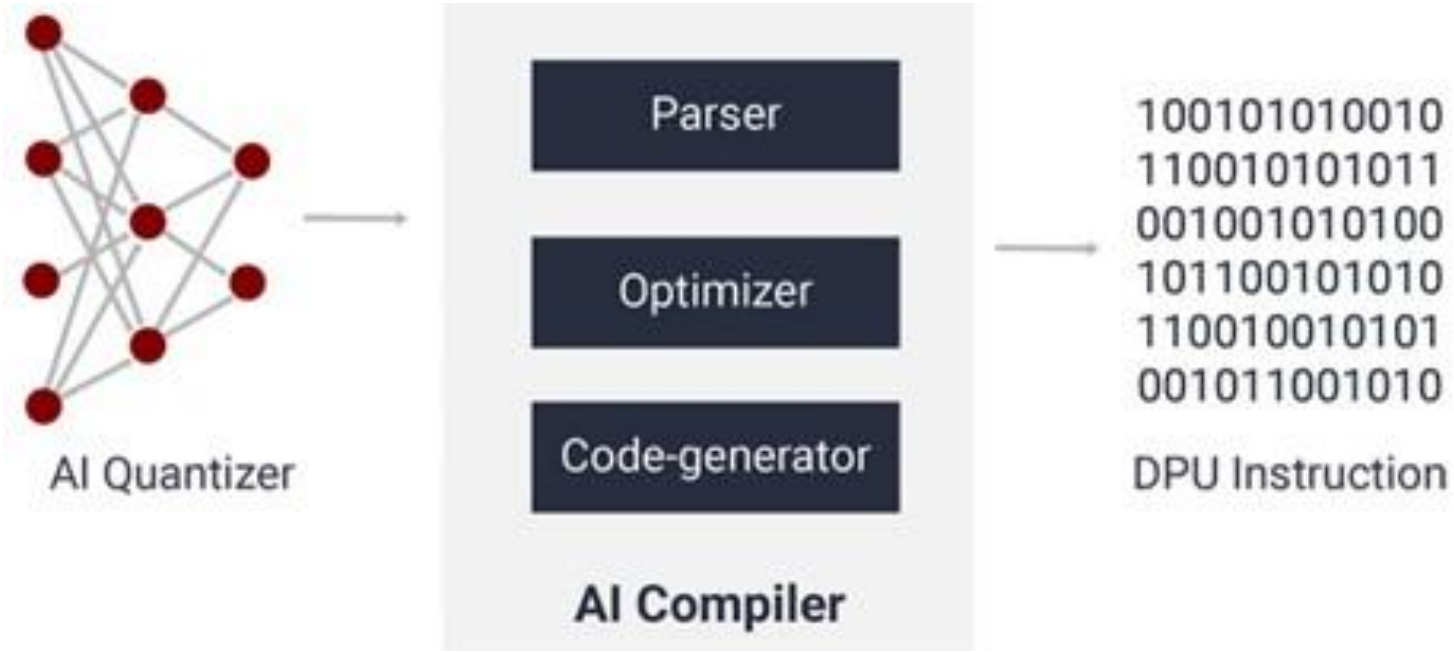




Predefined Hardware Design

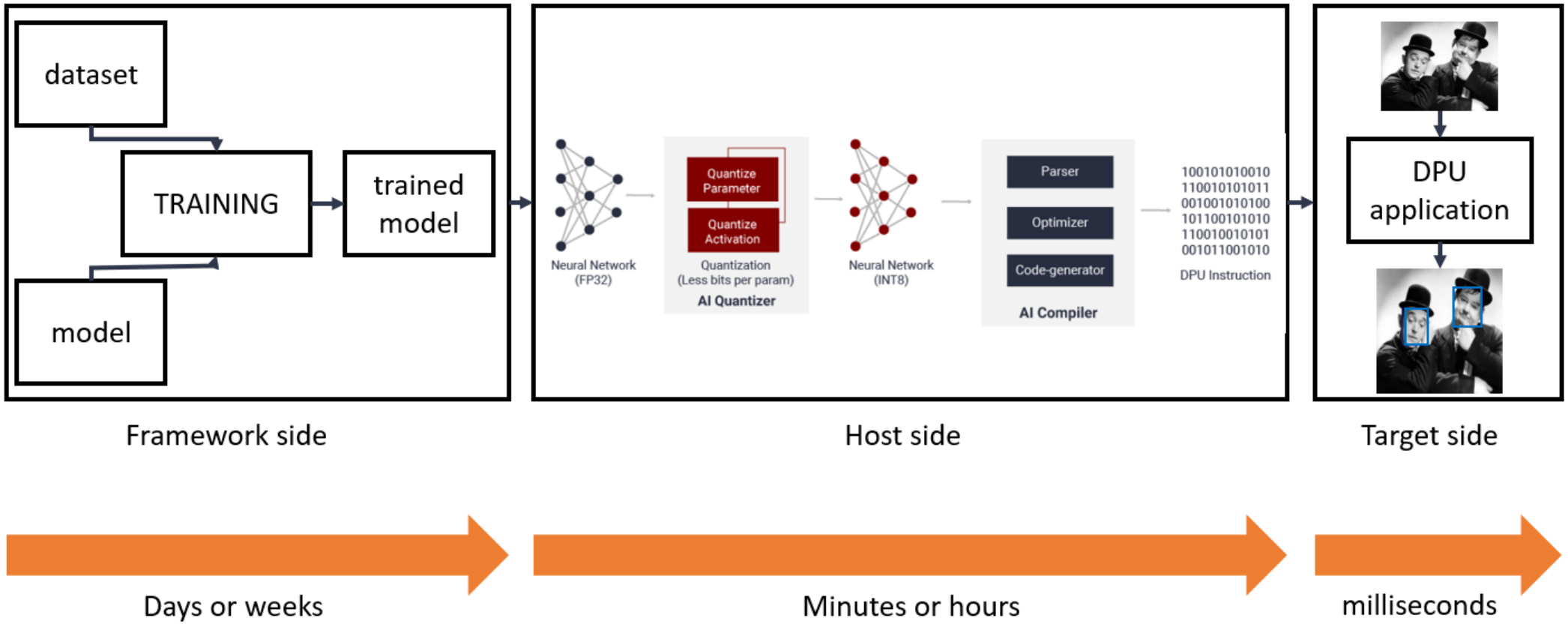
Design Flow for Xilinx Vitis AI (Tool)

- **AI Optimizer**
- **AI Quantizer**
- **AI Compiler**



Predefined Hardware Design

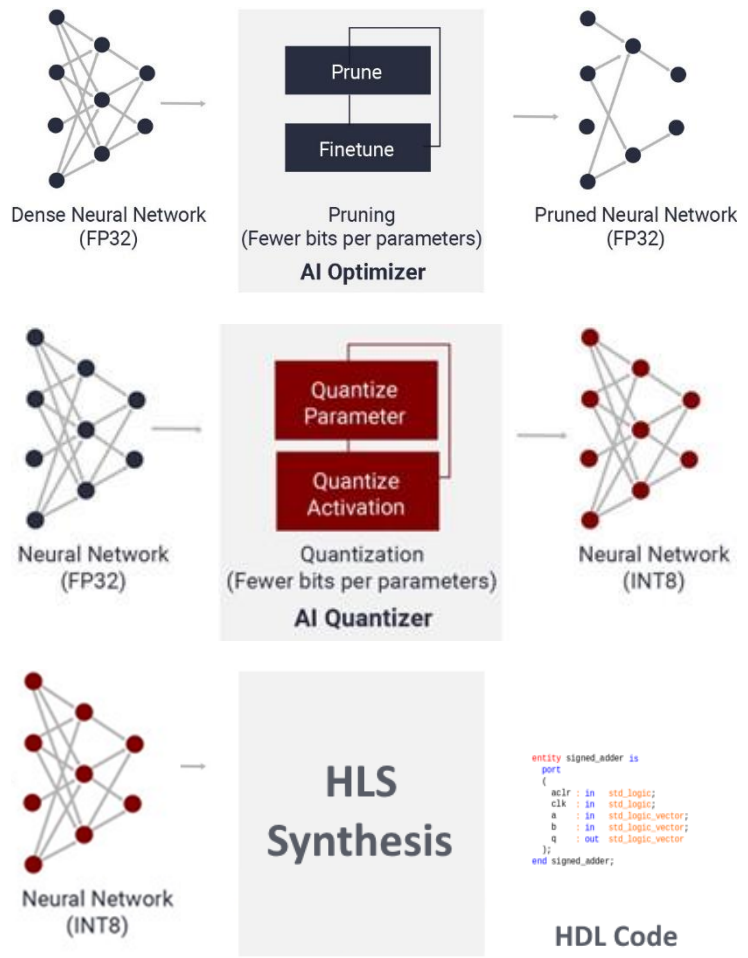
Design Flow for Xilinx Vitis AI



HLS Synthesis / Bare FPGA Design

HLS for AI

- e.g., hls4ml
- Synthesis
  to HDL Language or C/C++



In any Case:
Developer must take care of whole **System Design** (PCIe, Memory)

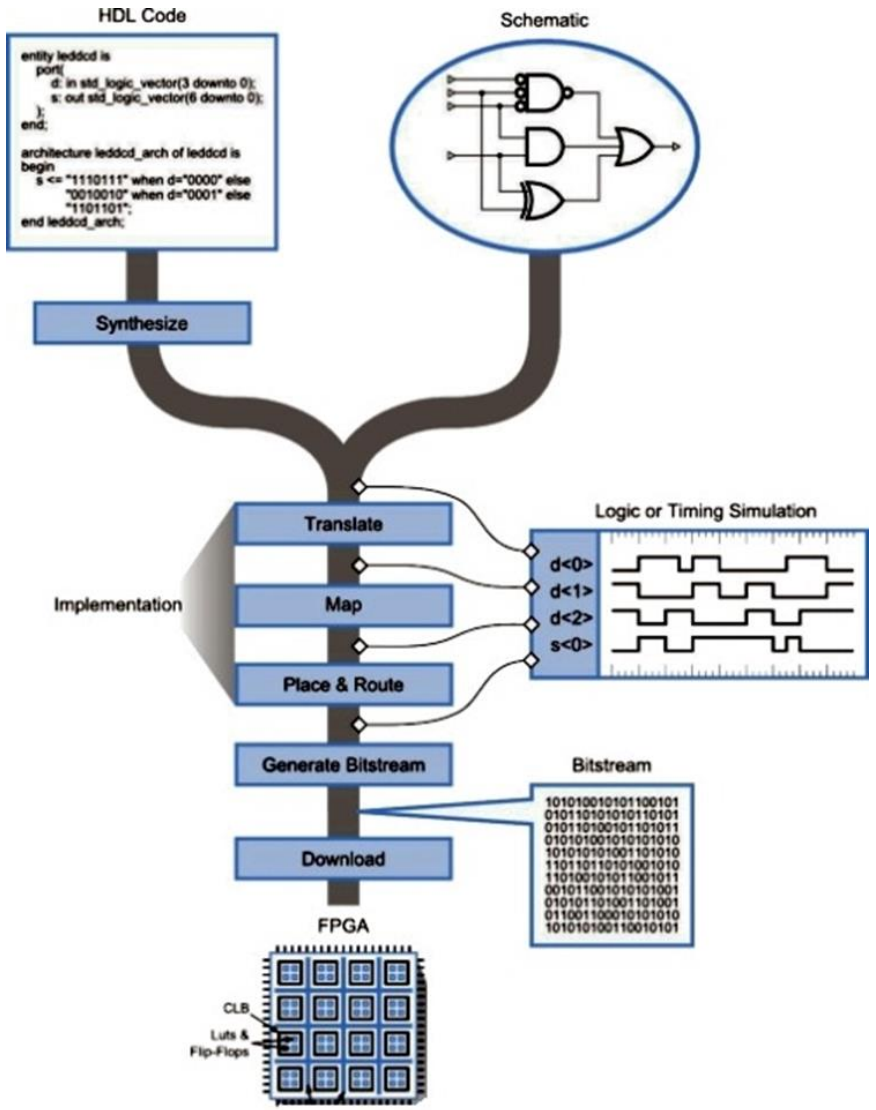
HLS Synthesis / Bare FPGA Design

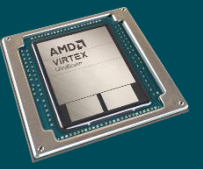
HLS for C/C++

- reduced language scope
- Synthesis to HDL Language

Low-Level HDL Design

- VHDL, Verilog, IPs





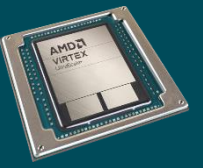
AI Accelerator Design

Predefined Hardware Design + Vitis AI

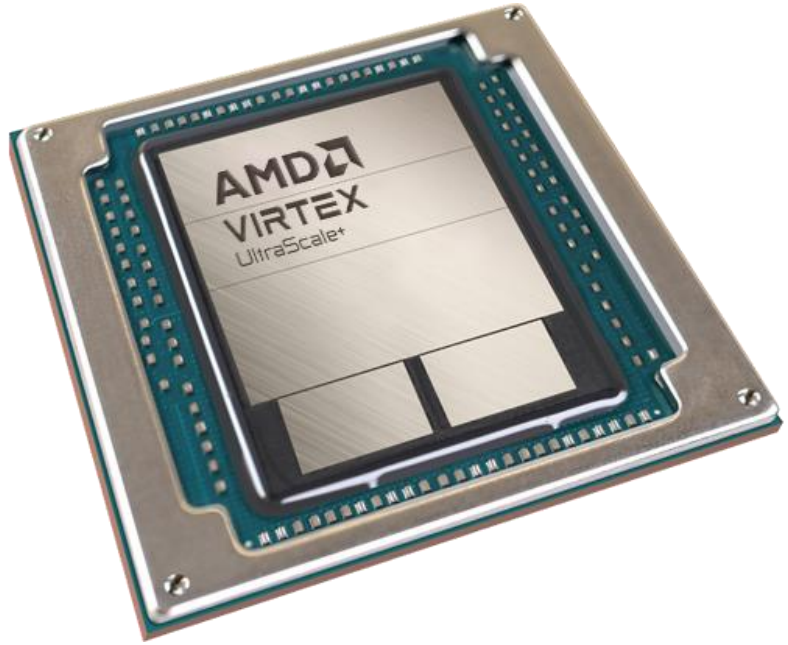
- Addressing AI Engineers + Software Developer
- Paradigma: Sequential Comutation
- Optimized Instructions → better **performance** (compared to CPU)
- Optimized Datastructure / Bitwidth → less **power consumtion** (compared to CPU)

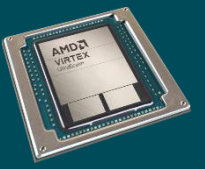
HLS Synthesis / Bare FPGA Design

- Much more Low-Level, addressing Hardware Developer
- Paradigma: Distributed Computation → higher degree of **parallelism**
- Optimized Dataflow → **better Latency, Throughput**



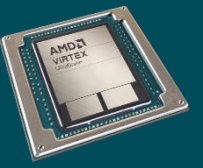
Content



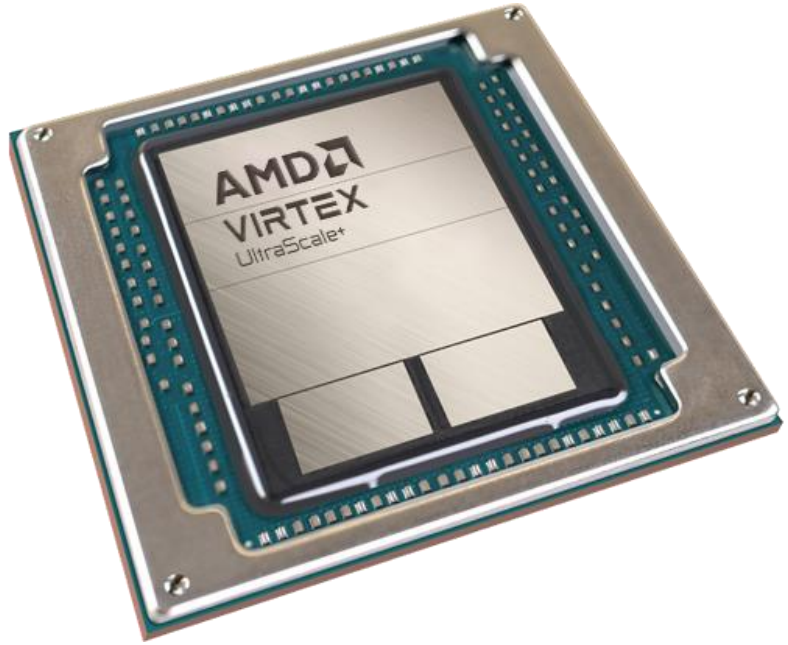


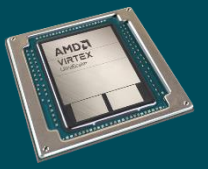
AI-specific Challenges for FPGAs

- Limited On-Chip Memory
- Optimizing for Precision
- Customization Overhead / System Design Integration
- Software and Toolchain Maturity
- Limited Resources for Large Models
- Data Movement and Bandwidth



Content



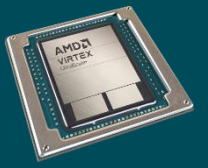


Conclusion

- FPGA Basics

- AI specific Toolflows & Accelerator Approaches
  - Predifined Hardware (DPU) + AI Compiler
  - HLS & Low-Level Synthesis

Only a brief Overview of FPGAs and Toolflows



Questions?