UNIVERSITÄT BONN

Gas Det Bonn

Physikalisches Institut

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

IMPLEMENTATION OF TRACKING ALGORITHMS FOR LIVE RECONSTRUCTION USING AI PROCESSORS

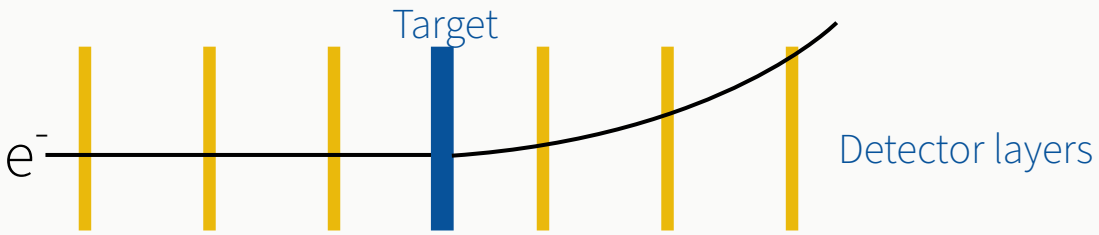
Klaus Desch
Jochen Kaminski
Michael Lupberger
Stephen Neuendorffer
Patrick Schwäbig

2022 DPG Spring Meeting Heidelberg, 3/23/2022

UNIVERSITÄT BONN

MOTIVATION

- Dark photon experiment, electron on target
- To be built at the ELSA accelerator at University of Bonn



- Low signal expected, want to measure 4×10^{14} electrons
- For more information see **T 64.8, directly after this talk, but different sessions**
- Want sophisticated, fast trigger

  → Use hardware accelerated algorithms for the trigger → live tracking
- This talk: pattern recognition with hardware-accelerated neural network

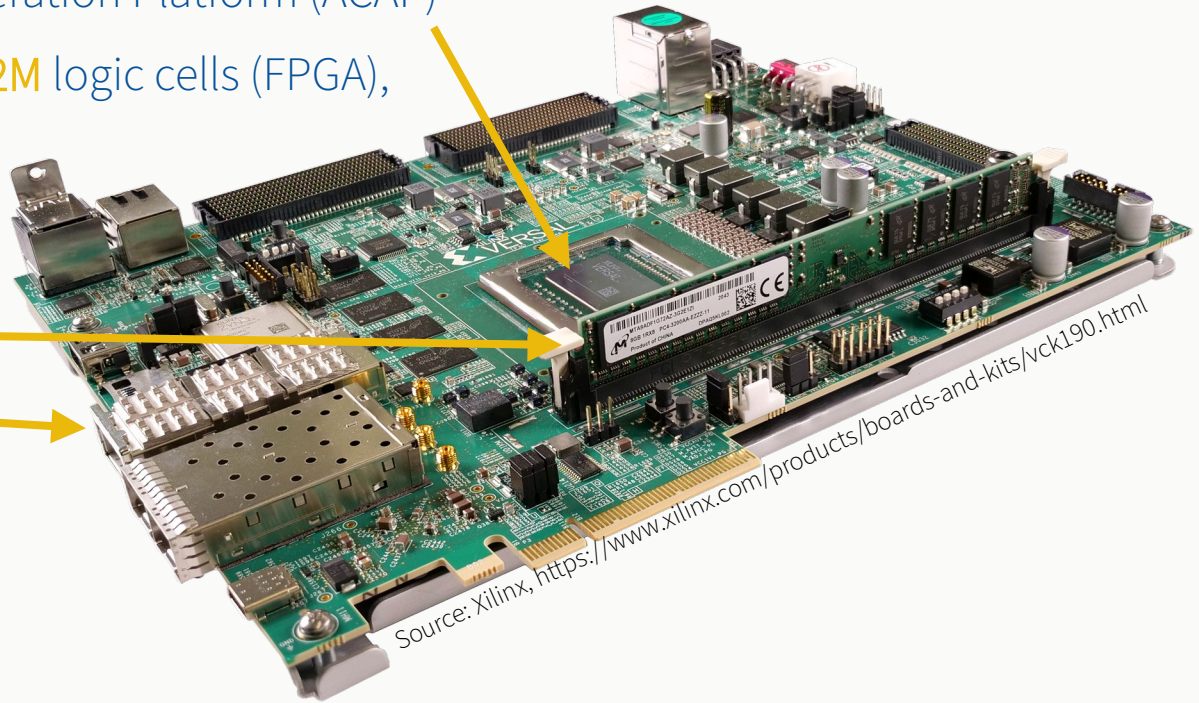
UNIVERSITÄT BONN

HARDWARE: XILINX VERSAL / VCK190

- Evaluation board VCK190
- Versal VC1902 Adaptive Compute Acceleration Platform (ACAP)
- 400 AI processors (“AI engines”), nearly 2M logic cells (FPGA),

  2k DSPs, Arm CPU, Arm RPU

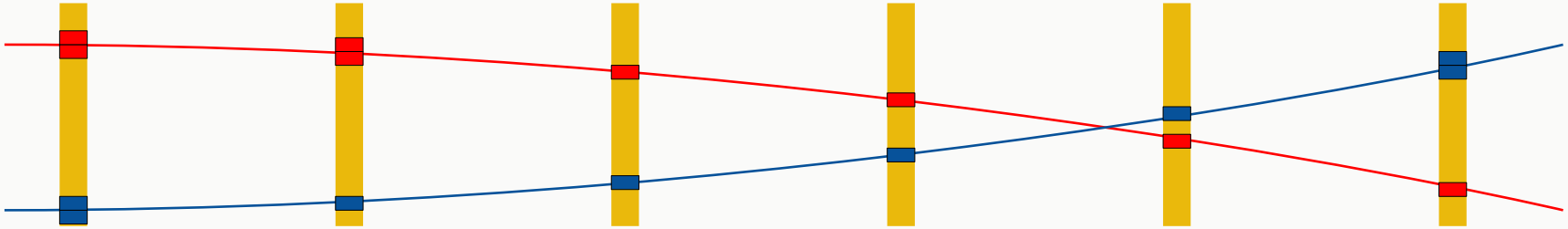
  Board with
  - 8 GB DDR4 RAM
  - QSFP28 for 100 Gbit Ethernet



Source: Xilinx, https://www.xilinx.com/products/boards-and-kits/vck190.html

UNIVERSITÄT BONN

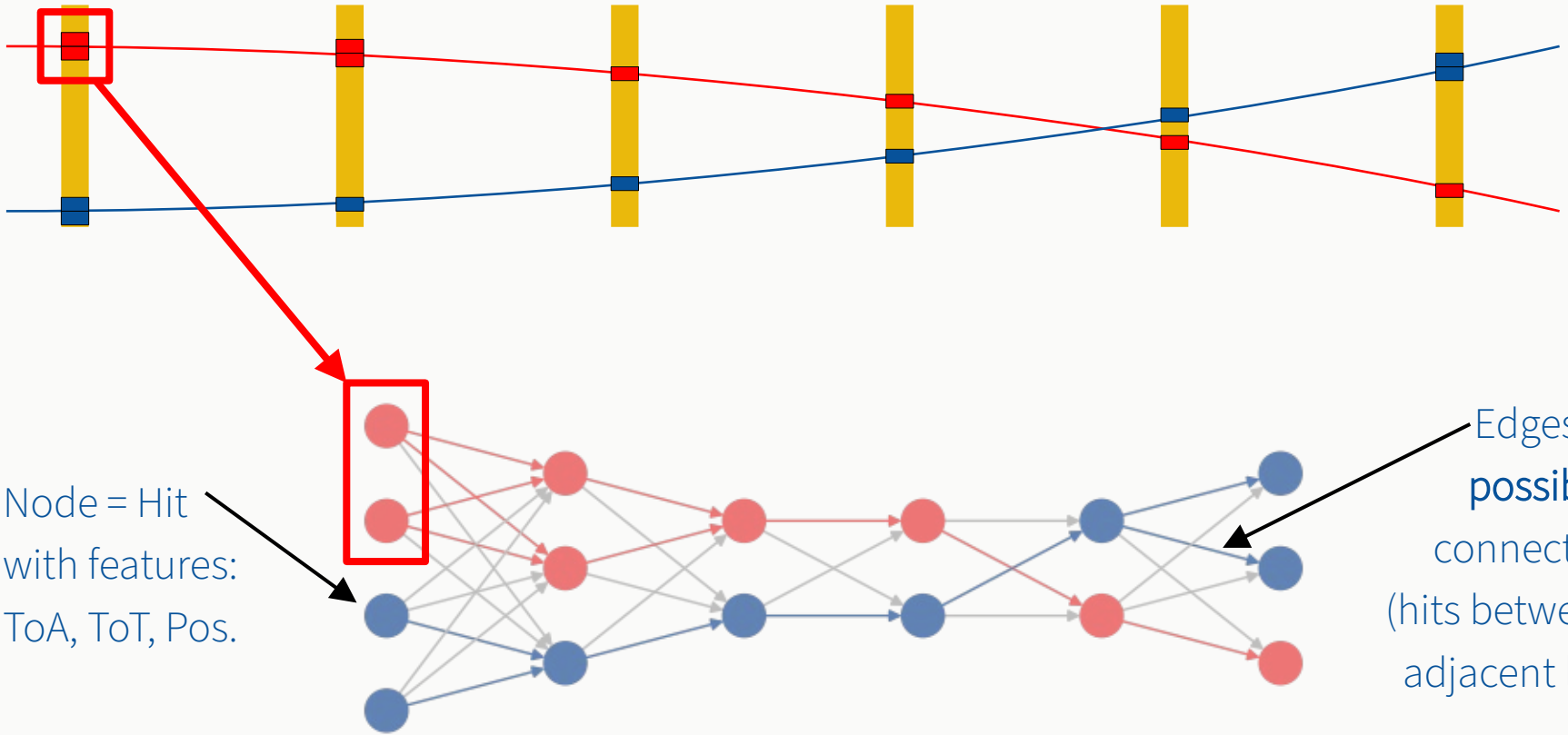
BUILDING NEURAL NETWORK INPUT



- Using Timepix3 chip → ToT, ToA, Position for each hit → read out with ACAP
- Build a graph…

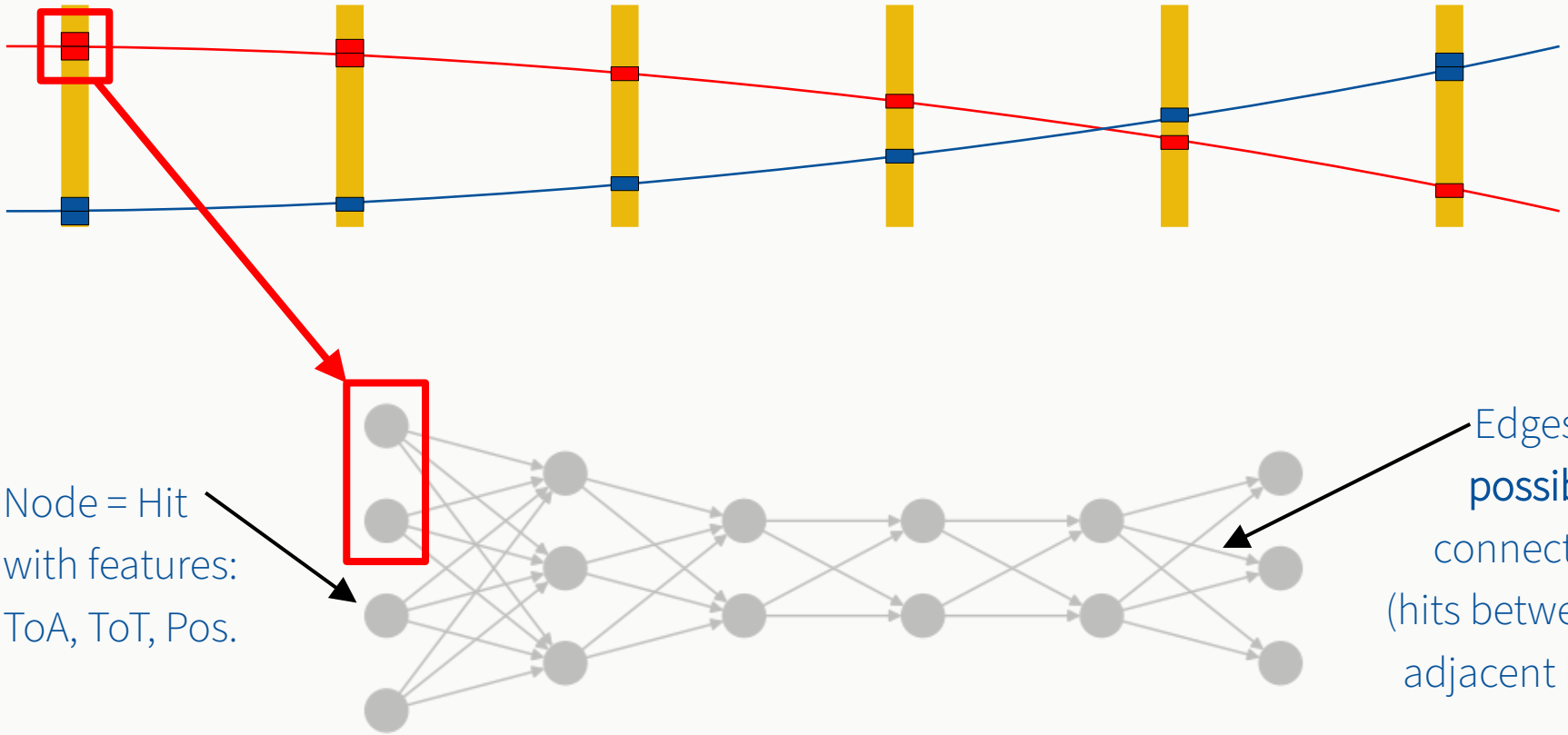
UNIVERSITÄT BONN

BUILDING NEURAL NETWORK INPUT



UNIVERSITÄT BONN

BUILDING NEURAL NETWORK INPUT

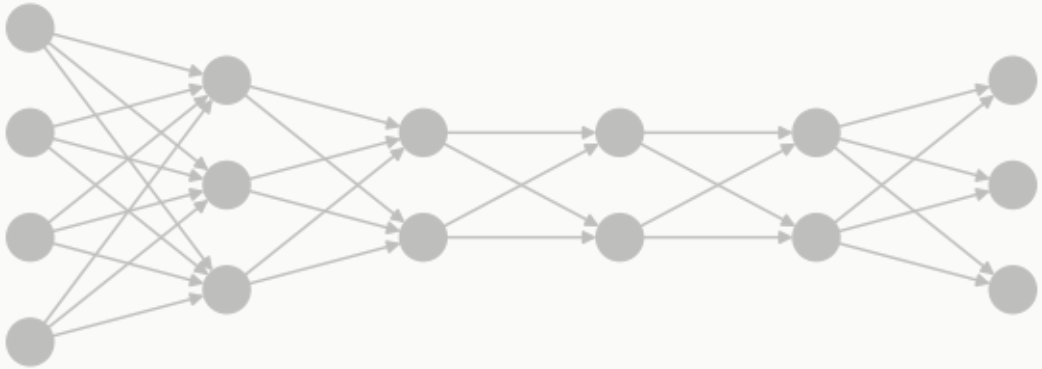


Node = Hit with features: ToA, ToT, Pos.

Edges = **possible** connections (hits between two adjacent layers)

UNIVERSITÄT BONN

GNN FOR PATTERN RECOGNITION



- Classify edges: true edge or false edge
  → Retrieve original track (pattern recognition)
- Use Interaction Network* which is a type of Graph Neural Network (GNN)
- Uses graph (hits and possible connections) as input
- Input graph **is not** the neural network (which can also be shown as a graph)
- GNNs generalization of other NNs: data is not 2D or 1D anymore

* described in DeZoort et al. (2021)

UNIVERSITÄT BONN

STRUCTURE OF THE GNN

Detector/Timepix3 → Readout & build graph (already on ACAP)

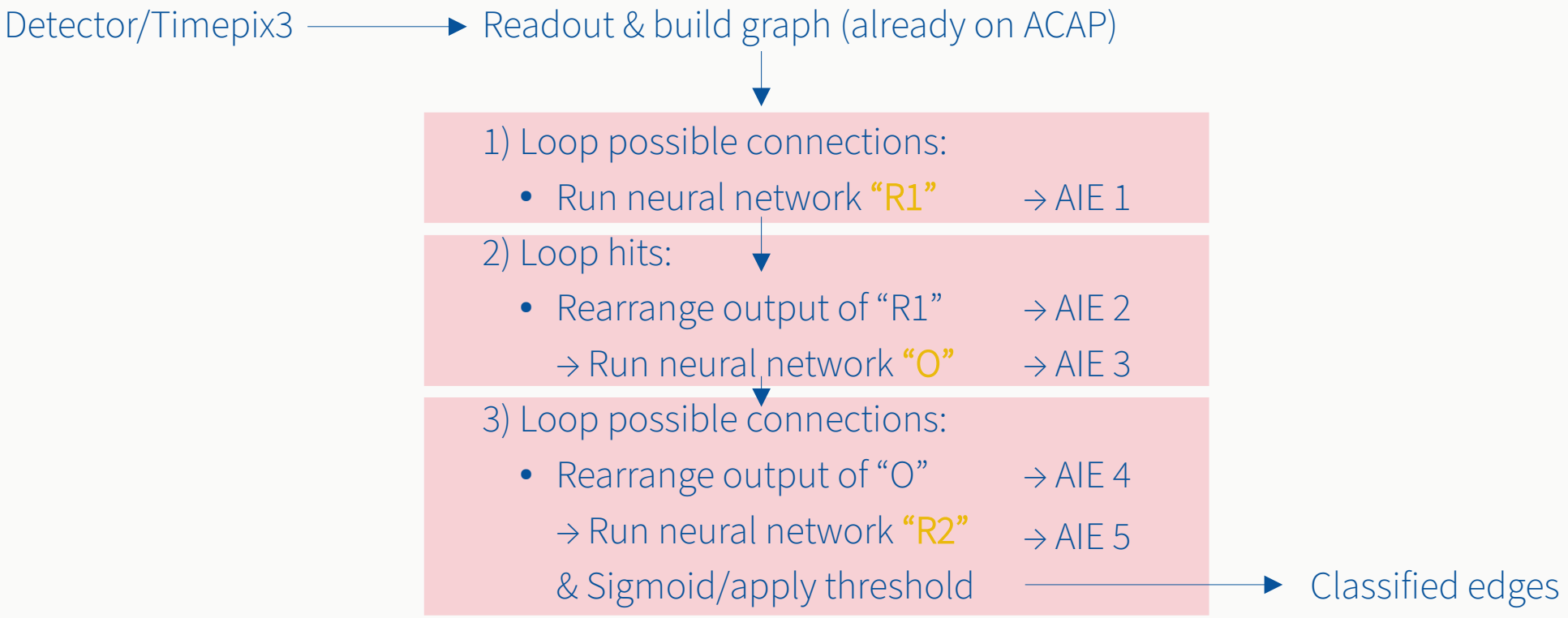
Classically:
Executed sequentially

1) Loop possible connections:
   - Run neural network “R1”
2) Loop hits:
   - Rearrange output of “R1”
     → Run neural network “O”
3) Loop possible connections:
   - Rearrange output of “O”
     → Run neural network “R2”
     & Sigmoid/apply threshold

→ Classified edges

UNIVERSITÄT BONN

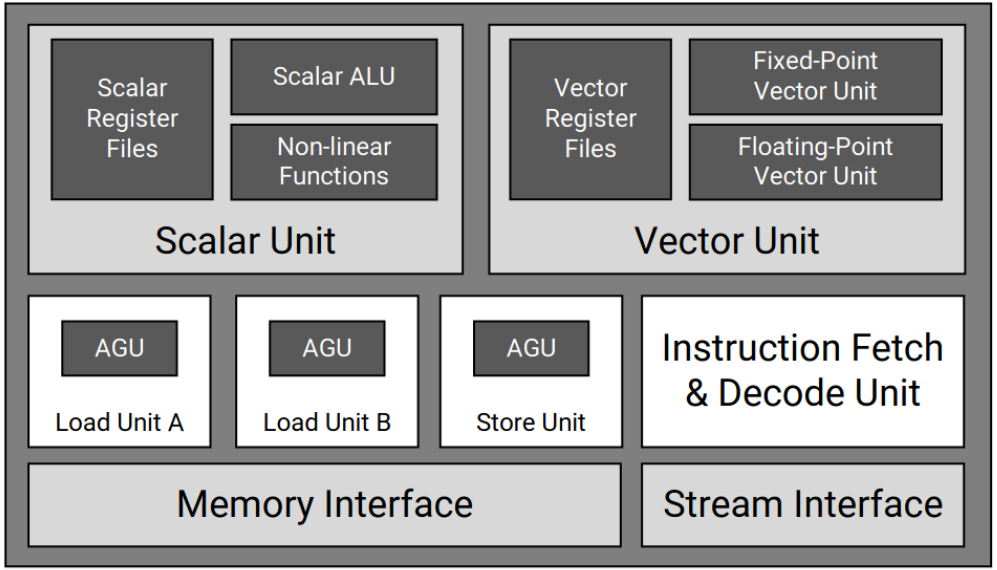
IMPLEMENTATION FOR ACAP — PIPELINING



IMPLEMENTATION FOR ACAP — PARALLEL EXECUTION

AI Engines of VCK190:

- Highly parallelized: SIMD VLIW



Source: Xilinx UG1079

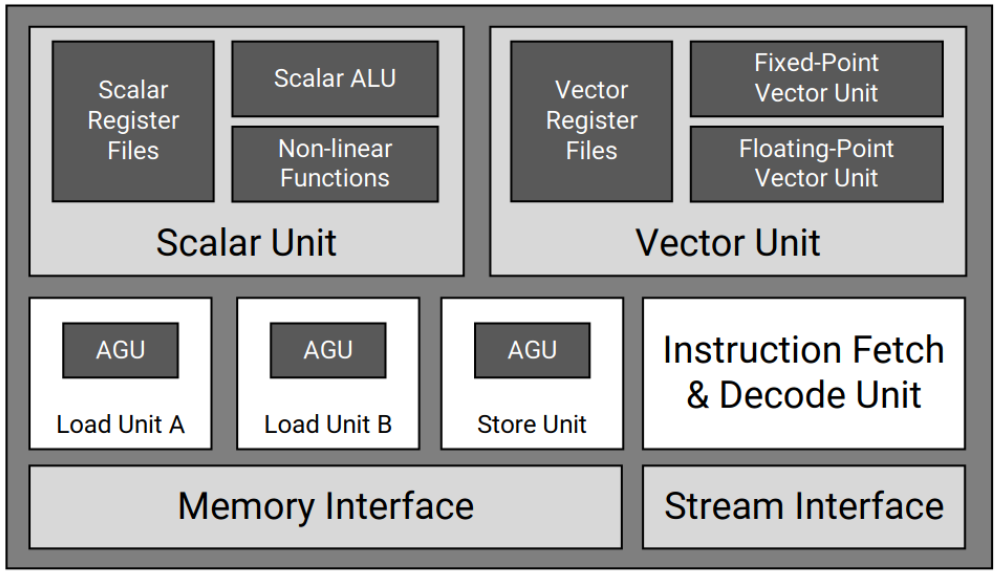
IMPLEMENTATION FOR ACAP — PARALLEL EXECUTION

Single instruction, multiple data

AI Engines of VCK190:

- Highly parallelized: SIMD VLIW

Very long instruction word

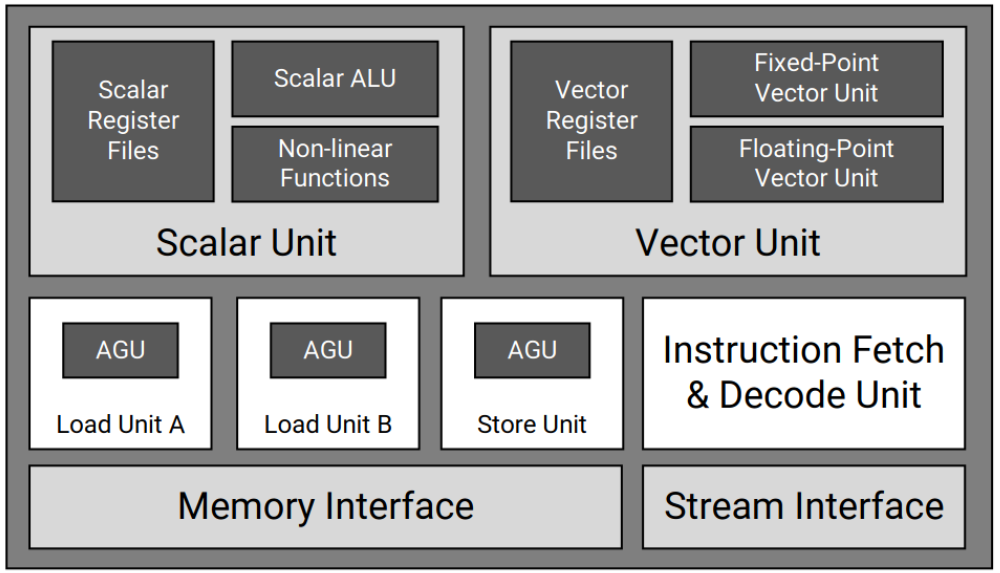


Source: Xilinx UG1079

IMPLEMENTATION FOR ACAP — PARALLEL EXECUTION

AI Engines of VCK190:

- Highly parallelized: SIMD VLIW
- VLIW → Simultaneous execution of:
  - Loading data
  - Storing data
  - Computations (scalar and **vector → SIMD**)
- Running at 1 GHz
- 8 Accumulators for 32b floating-point operations



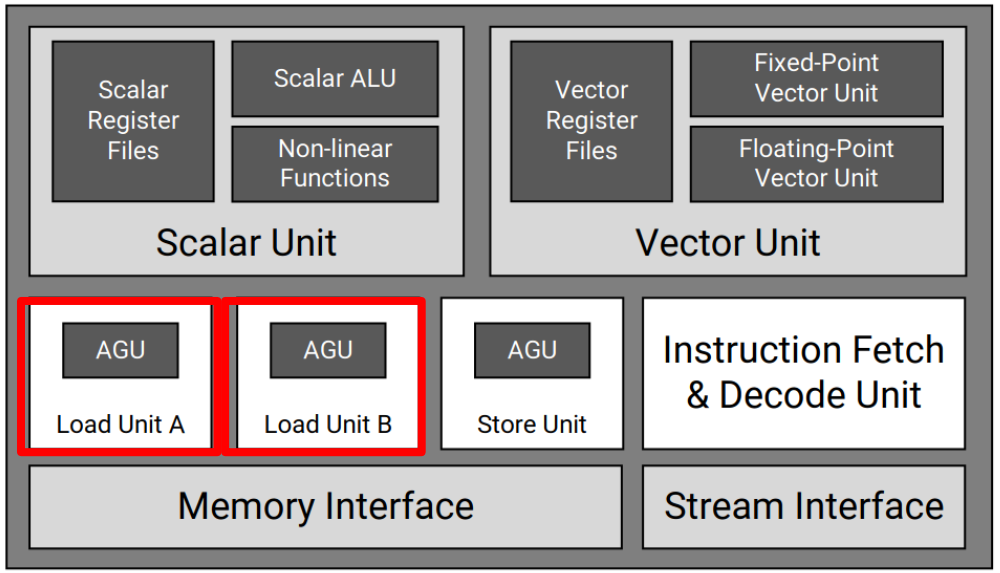
Source: Xilinx UG1079

→ Work on 8 possible connections simultaneously
while loading data, moving data & incrementing pointers…

IMPLEMENTATION FOR ACAP — PARALLEL EXECUTION

AI Engines of VCK190:

- Highly parallelized: SIMD VLIW
- VLIW → Simultaneous execution of:
  - Loading data
  - Storing data
  - Computations (scalar and **vector → SIMD**)
- Running at 1 GHz
- 8 Accumulators for 32b floating-point operations



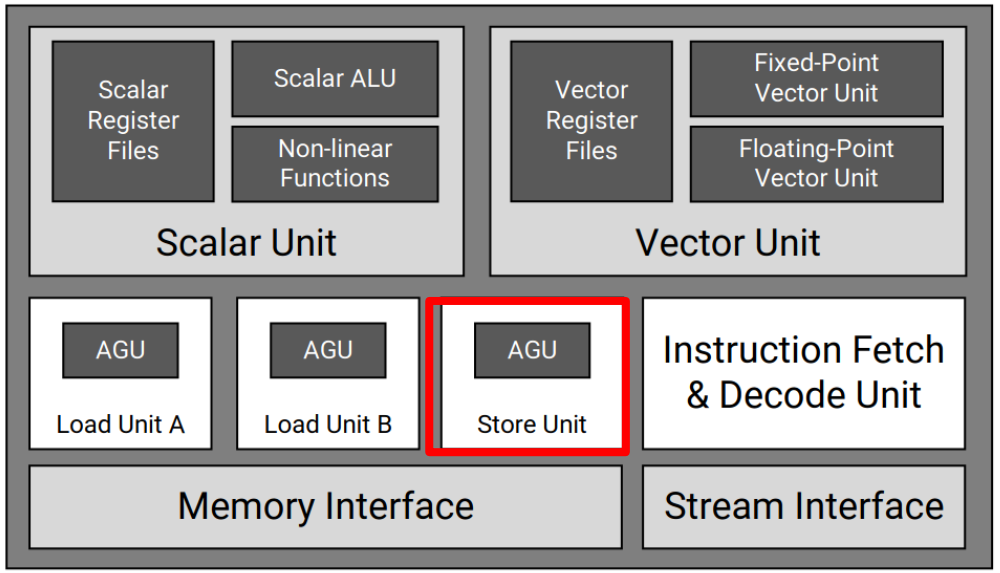
Source: Xilinx UG1079

→ Work on 8 possible connections simultaneously
while loading data, moving data & incrementing pointers…

IMPLEMENTATION FOR ACAP — PARALLEL EXECUTION

AI Engines of VCK190:

- Highly parallelized: SIMD VLIW
- VLIW → Simultaneous execution of:
  - Loading data
  - Storing data
  - Computations (scalar and **vector → SIMD**)
- Running at 1 GHz
- 8 Accumulators for 32b floating-point operations



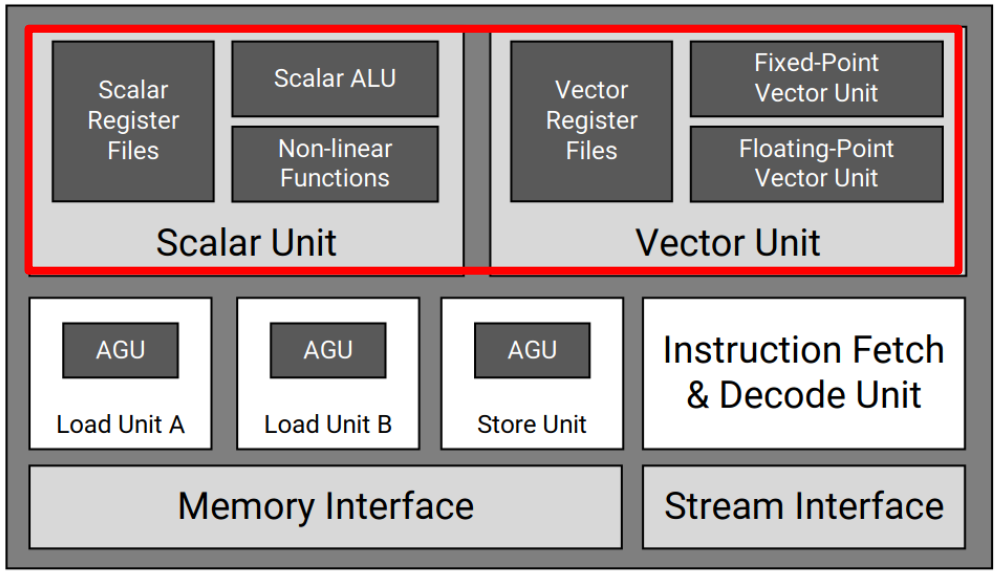
Source: Xilinx UG1079

→ Work on 8 possible connections simultaneously
while loading data, moving data & incrementing pointers…

IMPLEMENTATION FOR ACAP — PARALLEL EXECUTION

AI Engines of VCK190:

- Highly parallelized: SIMD VLIW
- VLIW → Simultaneous execution of:
  - Loading data
  - Storing data
  - Computations (scalar and **vector → SIMD**)
- Running at 1 GHz
- 8 Accumulators for 32b floating-point operations

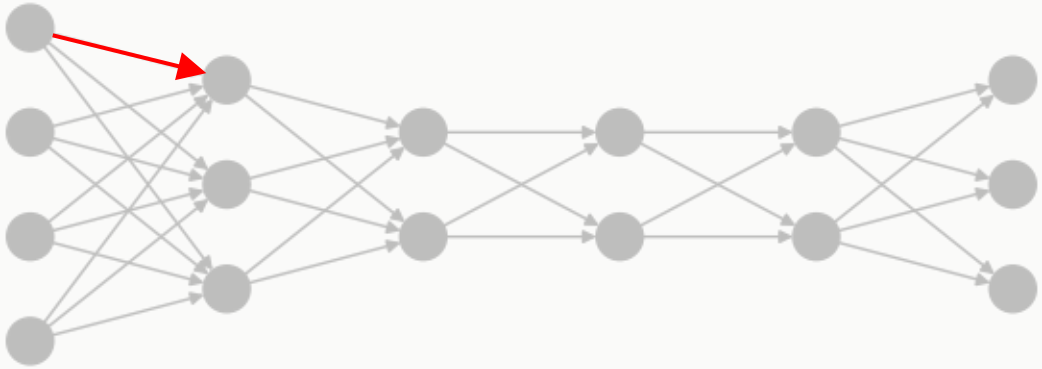


Source: Xilinx UG1079

→ Work on 8 possible connections simultaneously
while loading data, moving data & incrementing pointers…

UNIVERSITÄT BONN

GNN FOR PATTERN RECOGNITION

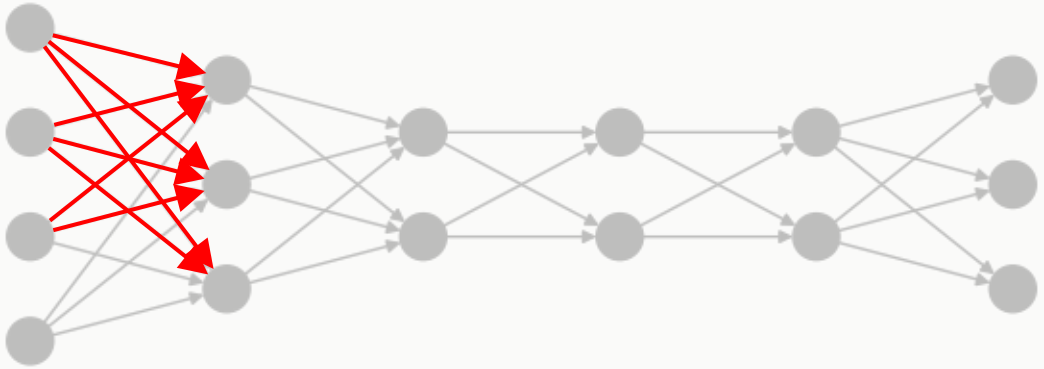


1) Loop possible connections:
   - Run neural network "R1" → AIE 1

Con. 1:

UNIVERSITÄT BONN

GNN FOR PATTERN RECOGNITION



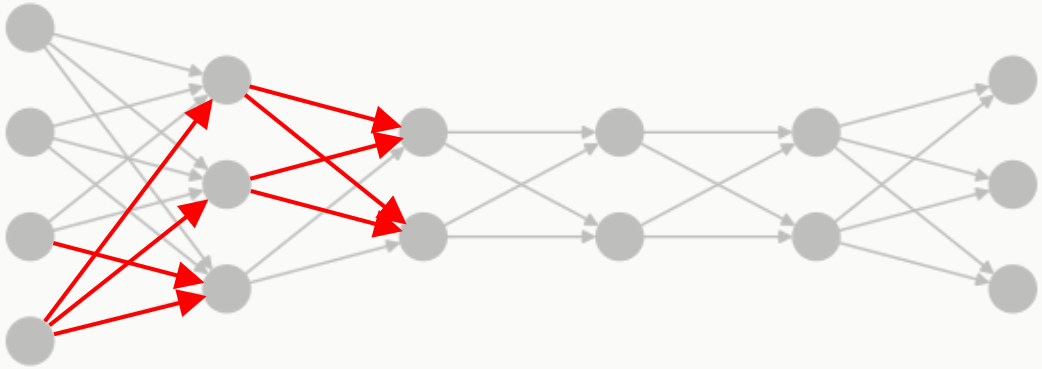
1) Loop possible connections/8:
   - Run neural network "R1" → AIE 1

...with 8 accumulators!

Con. 1:	ToA_{Start}	ToA_{End}	ToT_{Start}	ToT_{End}
Con. 2:	ToA_{Start}	ToA_{End}	ToT_{Start}	ToT_{End}
Con. 3:	ToA_{Start}	ToA_{End}	ToT_{Start}	ToT_{End}
Con. 4:	ToA_{Start}	ToA_{End}	ToT_{Start}	ToT_{End}
Con. 5:	ToA_{Start}	ToA_{End}	ToT_{Start}	ToT_{End}
Con. 6:	ToA_{Start}	ToA_{End}	ToT_{Start}	ToT_{End}
Con. 7:	ToA_{Start}	ToA_{End}	ToT_{Start}	ToT_{End}
Con. 8:	ToA_{Start}	ToA_{End}	ToT_{Start}	ToT_{End}

UNIVERSITÄT BONN

GNN FOR PATTERN RECOGNITION



1) Loop possible connections/8:
   - Run neural network "R1" → AIE 1

...with 8 accumulators!

Con. 9:	ToA_{Start}	ToA_{End}	ToT_{Start}	ToT_{End}
Con.10:	ToA_{Start}	ToA_{End}	ToT_{Start}	ToT_{End}
Con.11:	ToA_{Start}	ToA_{End}	ToT_{Start}	ToT_{End}
Con. 12:	ToA_{Start}	ToA_{End}	ToT_{Start}	ToT_{End}
Con.13:	ToA_{Start}	ToA_{End}	ToT_{Start}	ToT_{End}
Con.14:	ToA_{Start}	ToA_{End}	ToT_{Start}	ToT_{End}
Con. 15:	ToA_{Start}	ToA_{End}	ToT_{Start}	ToT_{End}
Con. 16:	ToA_{Start}	ToA_{End}	ToT_{Start}	ToT_{End}

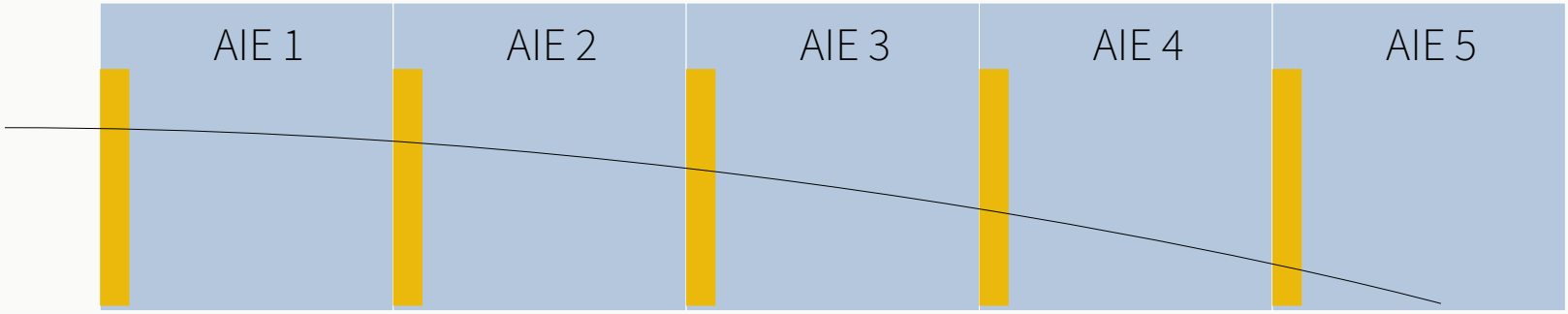
UNIVERSITÄT BONN

CURRENT STATUS & OUTLOOK

- Status: Implemented, currently improving performance
- Targeting O(µs) to classify each graph
- 32b floating-point → Quantize the NN? Use 8b integer, x16 speedup
- Could also include FPGA-part of ACAP
- Only using 5 AI engines! Space to go faster or do more

CURRENT STATUS & OUTLOOK

- Status: Implemented, currently improving performance
- Targeting O(µs) to classify each graph
- 32b floating-point → Quantize the NN? Use 8b integer, x16 speedup
- Could also include FPGA-part of ACAP
- Only using 5 AI engines! Space to go faster or do more:
  - Build tracks
  - Fit tracks, e.g. using a Kalman filter:



UNIVERSITÄT BONN

Thanks for your attention!

Questions?