B2GM: Scalability of CaloClusterNet on Heterogenous SoC Devices (AMD Versal)

Marc Neu, Isabel Haide, Kai Unger, Timo Justinger, Till Rädler, Valdrin Dajaku, Torben Ferber, Jürgen Becker

Introduction Till Rädler

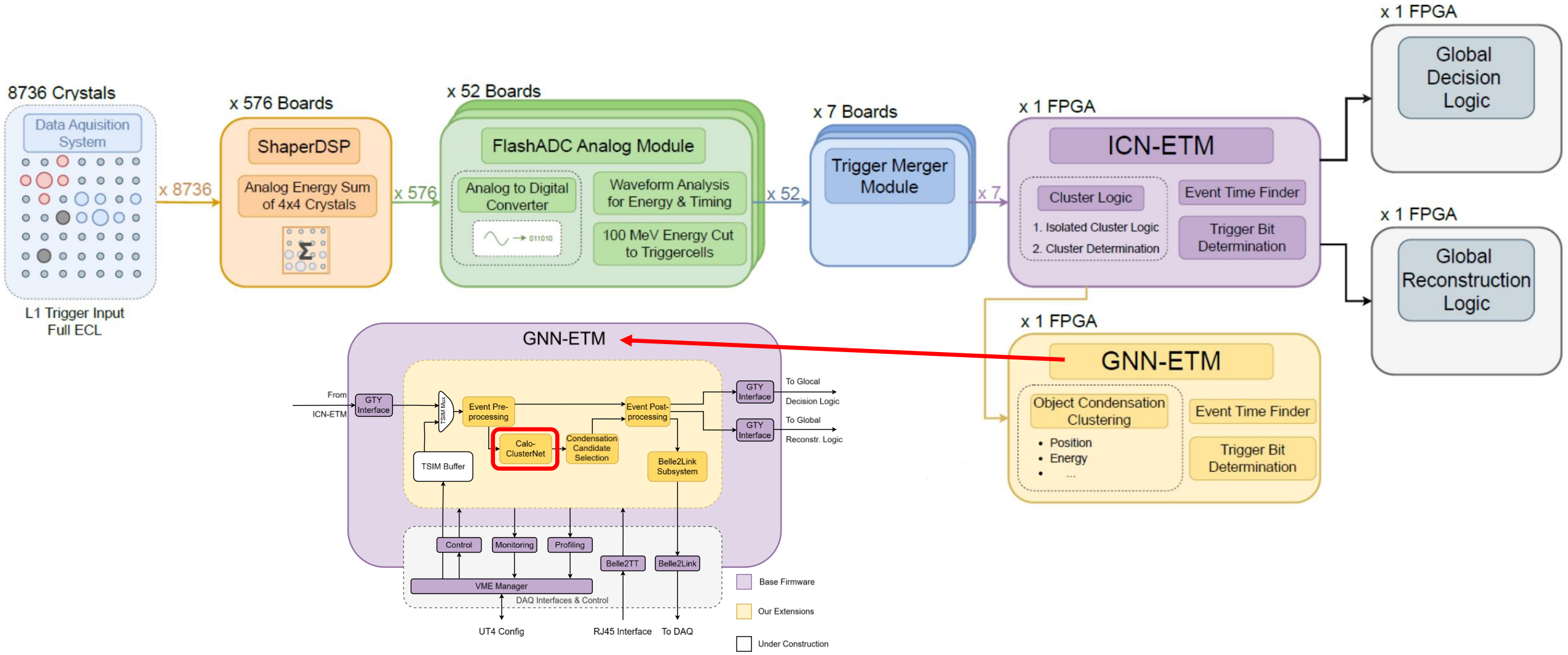
- Master Student KIT ITIV (Electrical Engineering)
- System Engineering

- Master Thesis:

„Scalable graph neural networks through CGRAs for Belle II Detector “

Start 1.10.2024

Current trigger system



Motivation

- I will perform a case study, evaluating latency and throughput of the CaloClusterNet in the Belle II Detector
- **Goal of the Thesis**: implementing CaloClusterNet on the Versal vck190 board
- The board uses AIEs (Artificial Intelligence Engines) to increase computation power

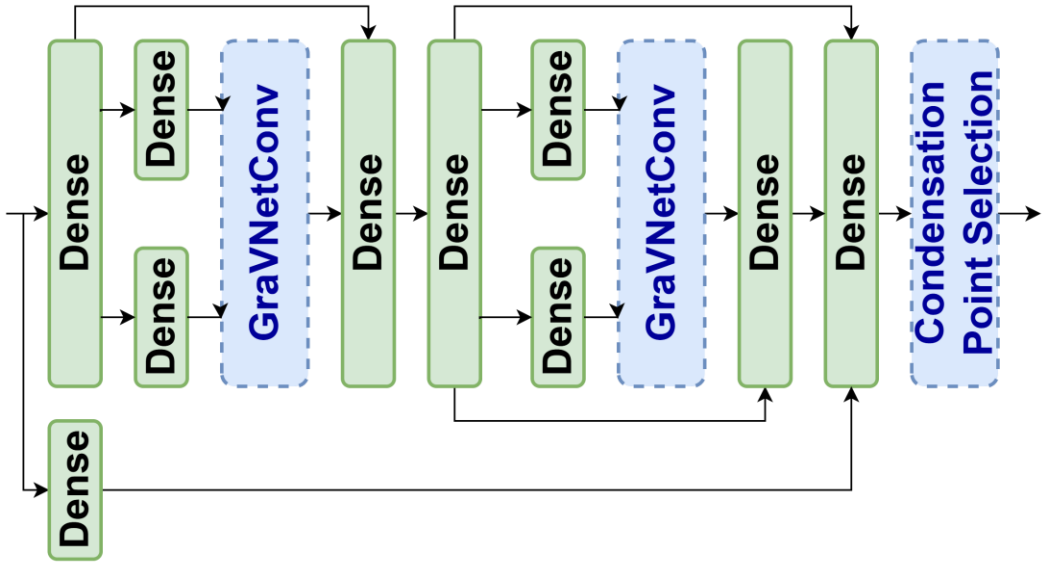


Figure by Marc Neu

UT4 vs VCK190

Resources	**UT4**: (Virtex UltraScale XCVU-FLGB2104-2-E)	**VCK190**: (Versal AI Core XCVC1902-2MSEVSVA2197)
AIE	0	**400**
LUTs (Look-up-tables)	1.074.240	899.840
DSPs (digital signal processors)	1.800	1.968

AMD: Versal AI Core Series VCK190 Evaluation Kit

- Versal AI Core has better technology, roughly the same PL (programmable Logic) size, but also includes 400 AIE

Current Limitations & Constraints

System constraints

- Data throughput of 8MHz
- Latency ~5us (for upgrade)
- Should fit on one SoC

Current Limitations UT4

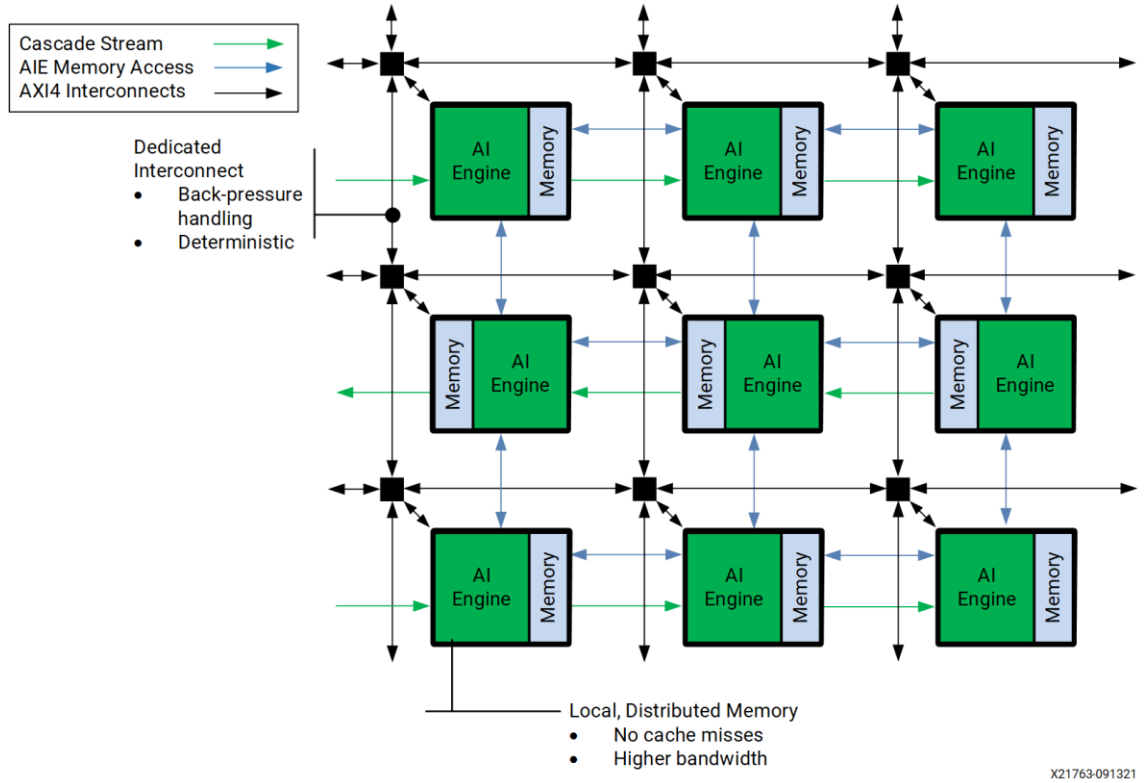
- Currently we use 32 out of 576 input points (trigger cells) coming in
- With increasing beam background, we have to increase the number of trigger cells

We would like to scale the number of trigger cells

→ **case study on Versal board**

Structure of the Versal Vck190

- 8x50 Array of AIEs
- Central NoC (Network on Chip)
- Between PL&AIEs AXI4 Streams
  - PL→AIE: 156
  - AIE→PL: 234

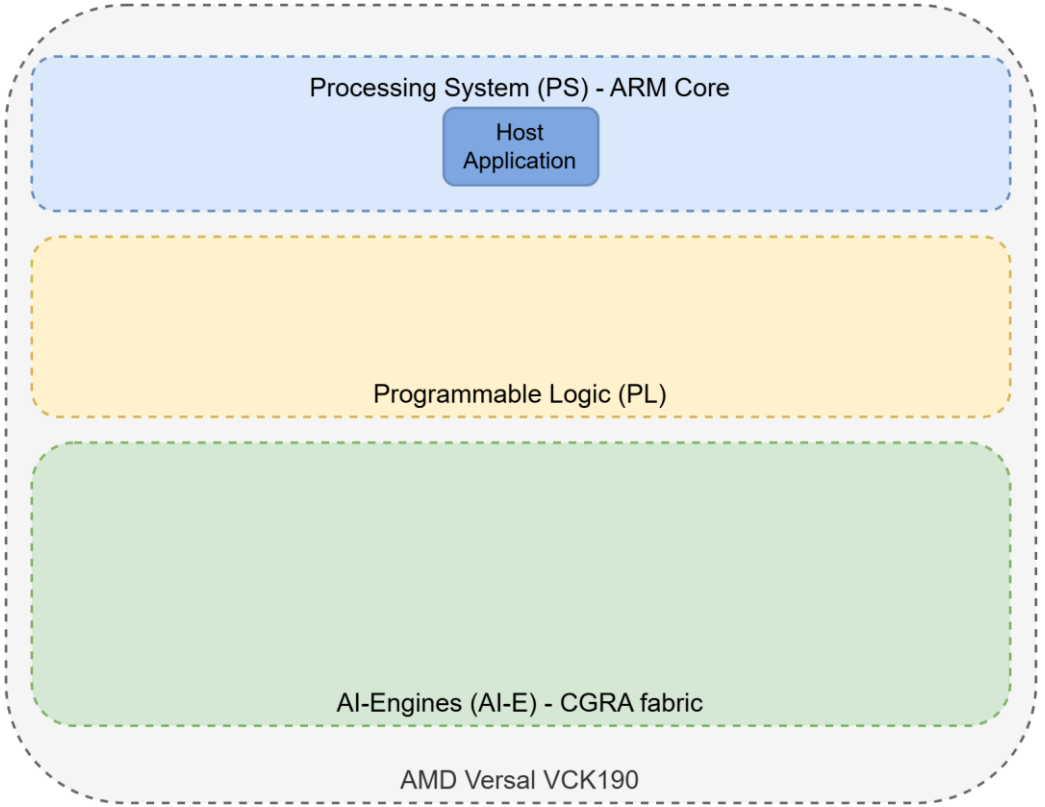


AMD: Versal Adaptive SoC AI Engine Architecture Manual AM009

- PLIOs as possible bottleneck (also see: Yun-Tsung Lais presentation)

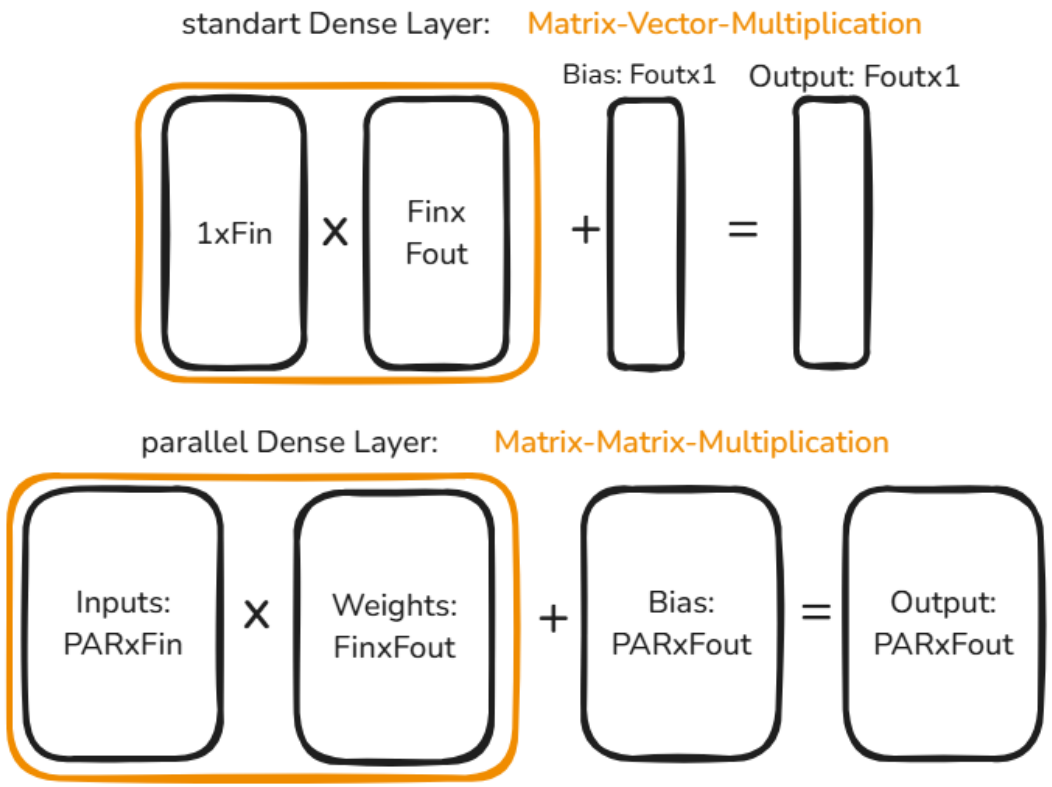
Problem Statement

- Which function can be implemented on the AIEs, and which must be implemented using the PL
- I will perform an evaluation of the single components
- Using this evaluation, I will develop a partitioning of the CaloClusterNet on the AMD Versal vck190



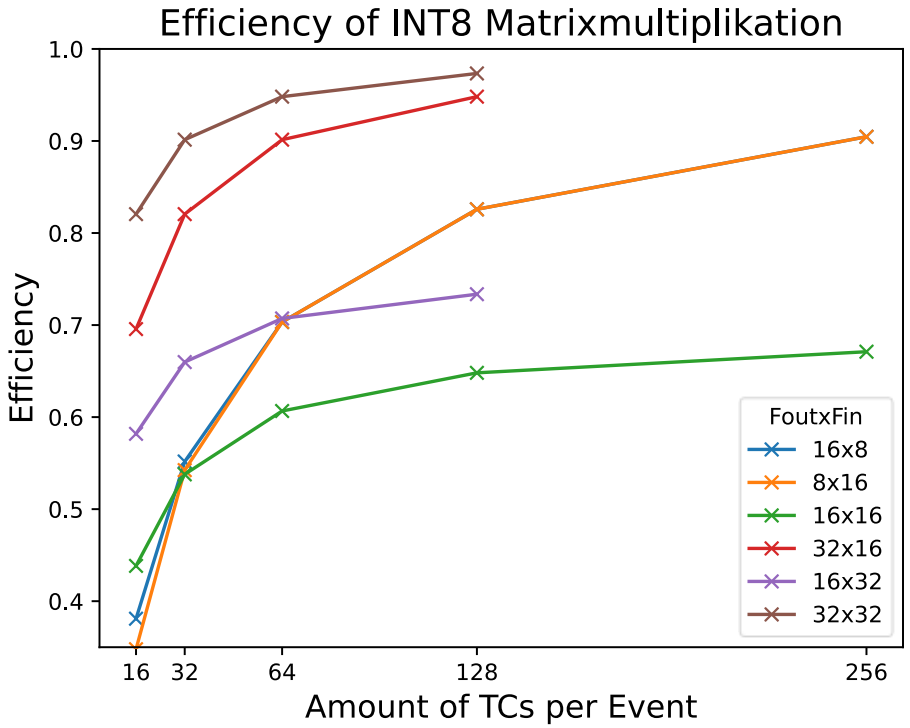
Matrix multiplication for Neural Networks

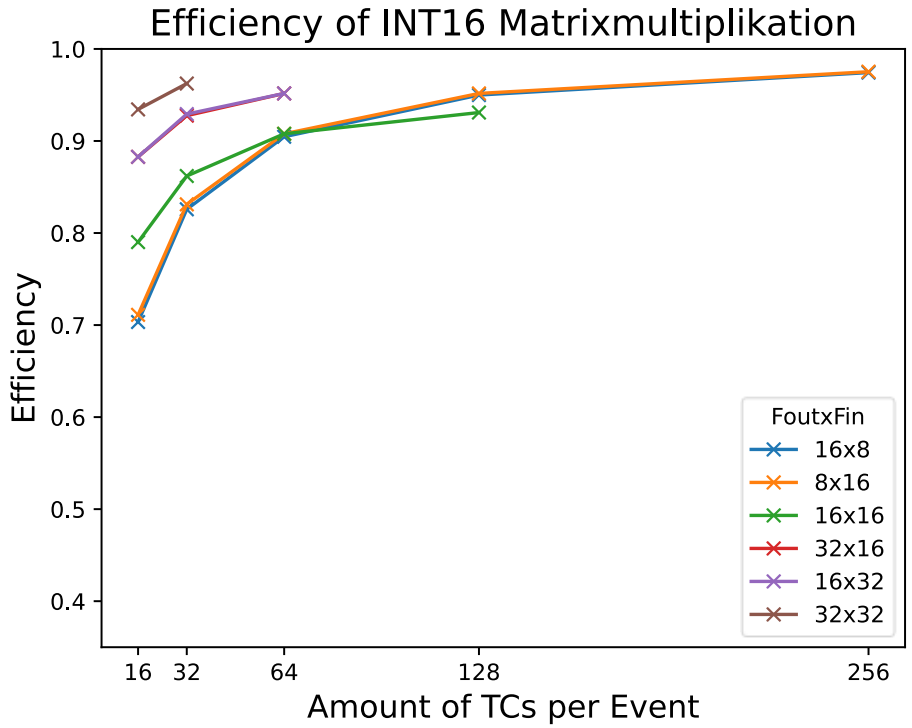
- Parallel incoming TCs are propagated through neural networks
- can be calculated in parallel and are independent of each other



$$Efficiency = \frac{cycle_{min}}{cycle_{real}}$$

Evaluation on the Versal Board



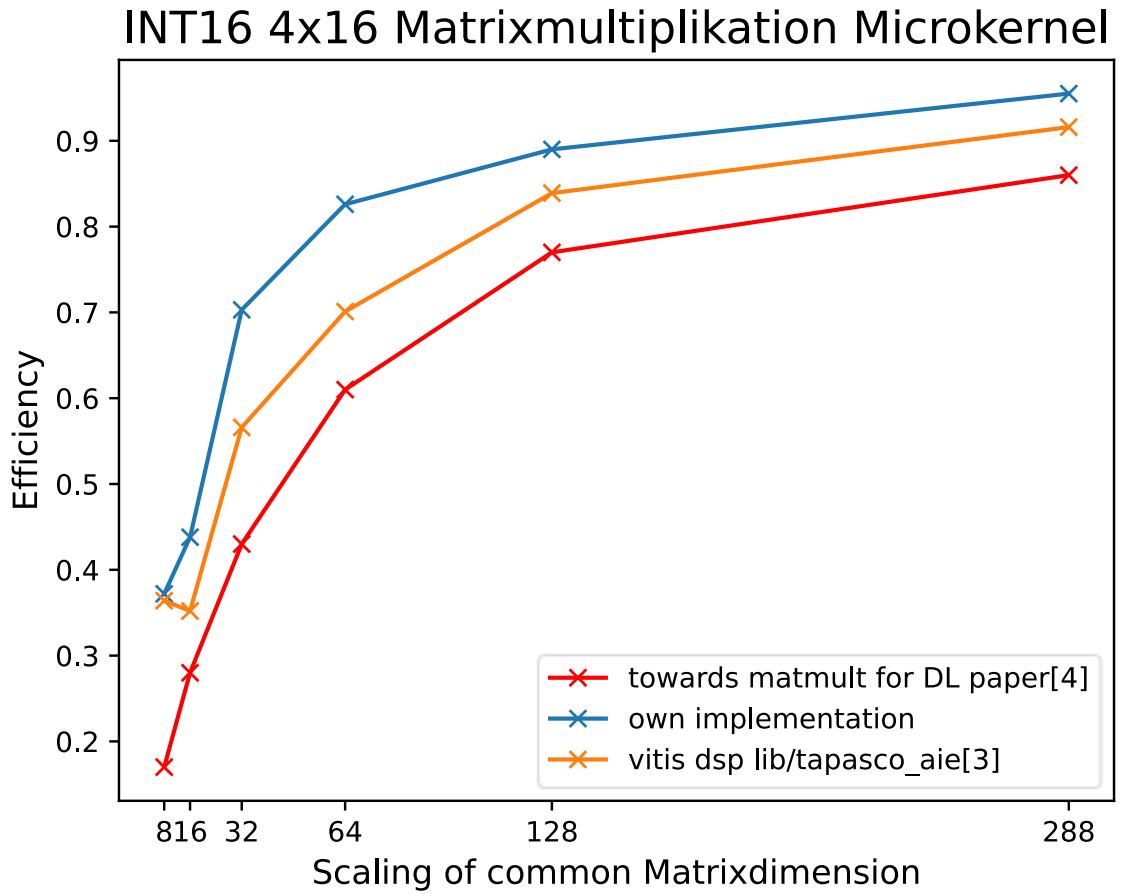


INT8: Efficiencies up to 95%;
for smaller Dimensions only ~50%
→ int8 not natively supported

INT16: Efficiencies up to 97%;
minimal ~70%

Comparison to State of the Art

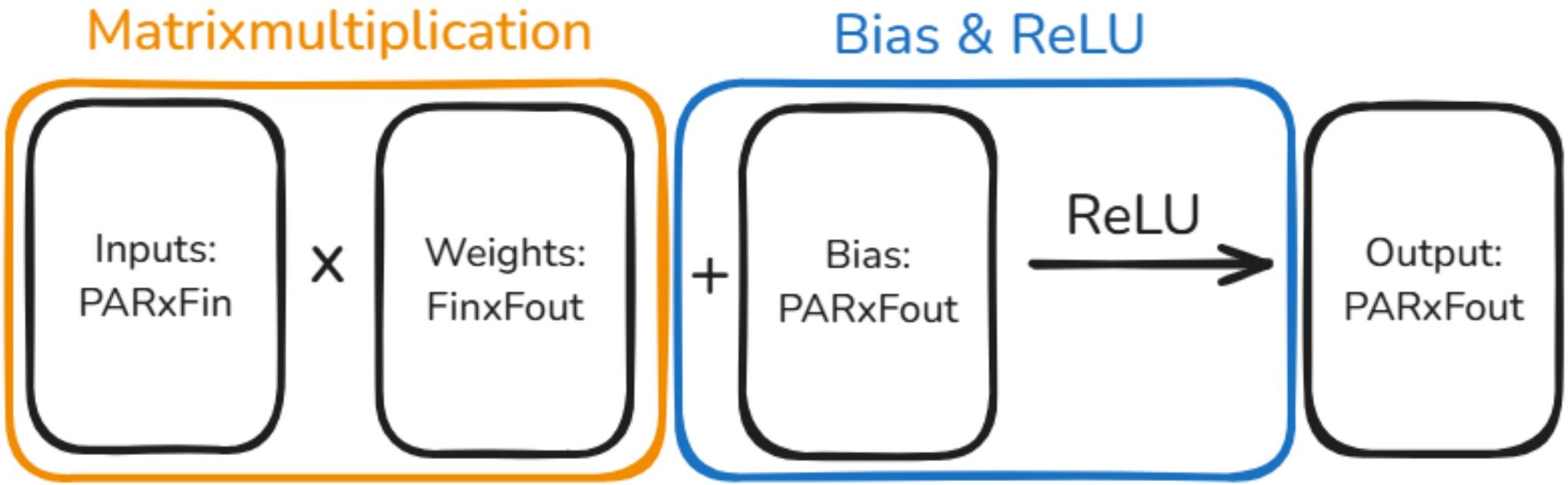
- Matrixmultiplication INT16 (Efficiencies up to 95%) [1], [2]
- Comparison of own implementation with State-of-the-Art paper for a 4x16 Microkernel



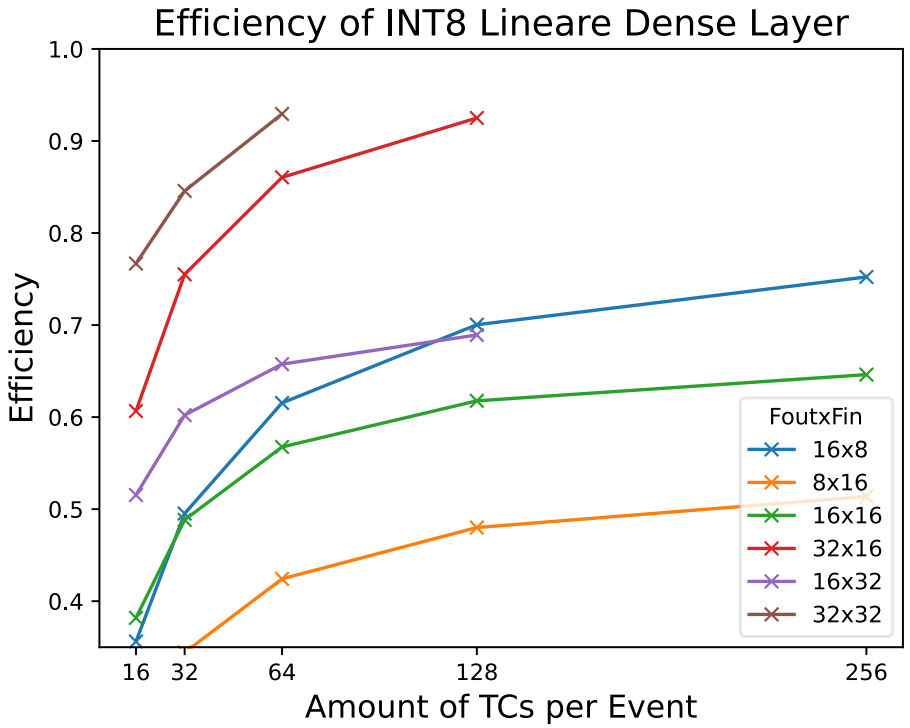
[1] Endri Taka et al. “MaxEVA: Maximizing the Efficiency of Matrix Multiplication onVersal AI Engine”.
[2] Xiaodong Deng et al. “AMA: An Analytical Approach to Maximizing the Efficiencyof Deep Learning on Versal AI Engine”
[3] Carsten Heinz et al. “TaPaSCo-AIE: An Open-Source Framework for Streaming-Based Heterogeneous Acceleration Using AMD AI Engines”
[4] Jie Lei, José Flich, and Enrique S. Quintana-Ortí. “Toward Matrix Multiplica-tion for Deep Learning Inference on the Xilinx Versal”. In: 2023 31st EuromicroInternational Conference on Parallel, Distributed and Network-Based Processing(PDP)

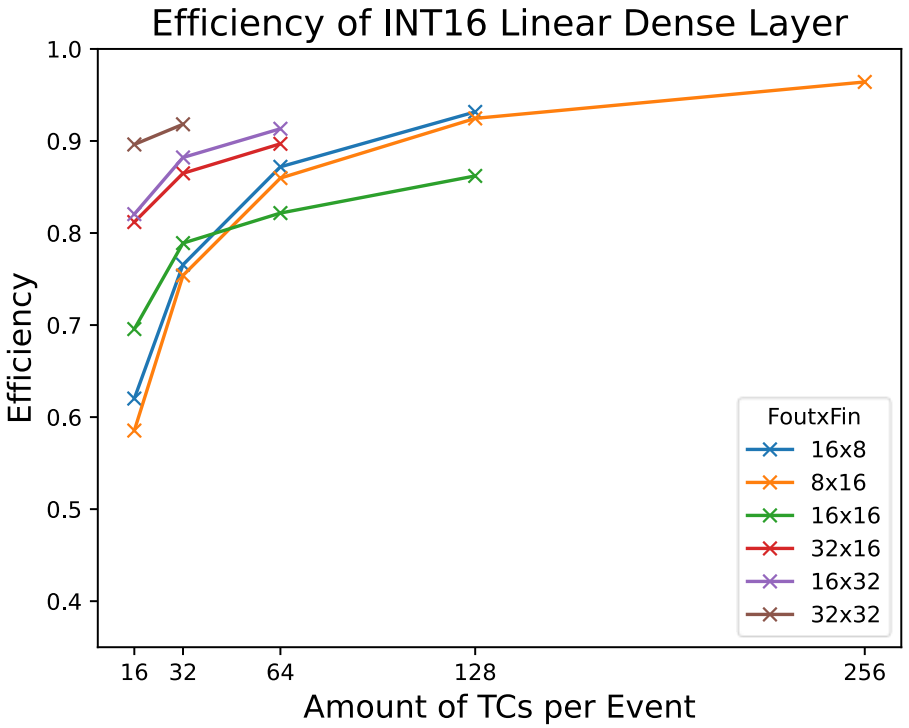
Dense Layer

- What is missing to go from only matrixmultiplication to a dense layer ?



Evaluation Dense Layer



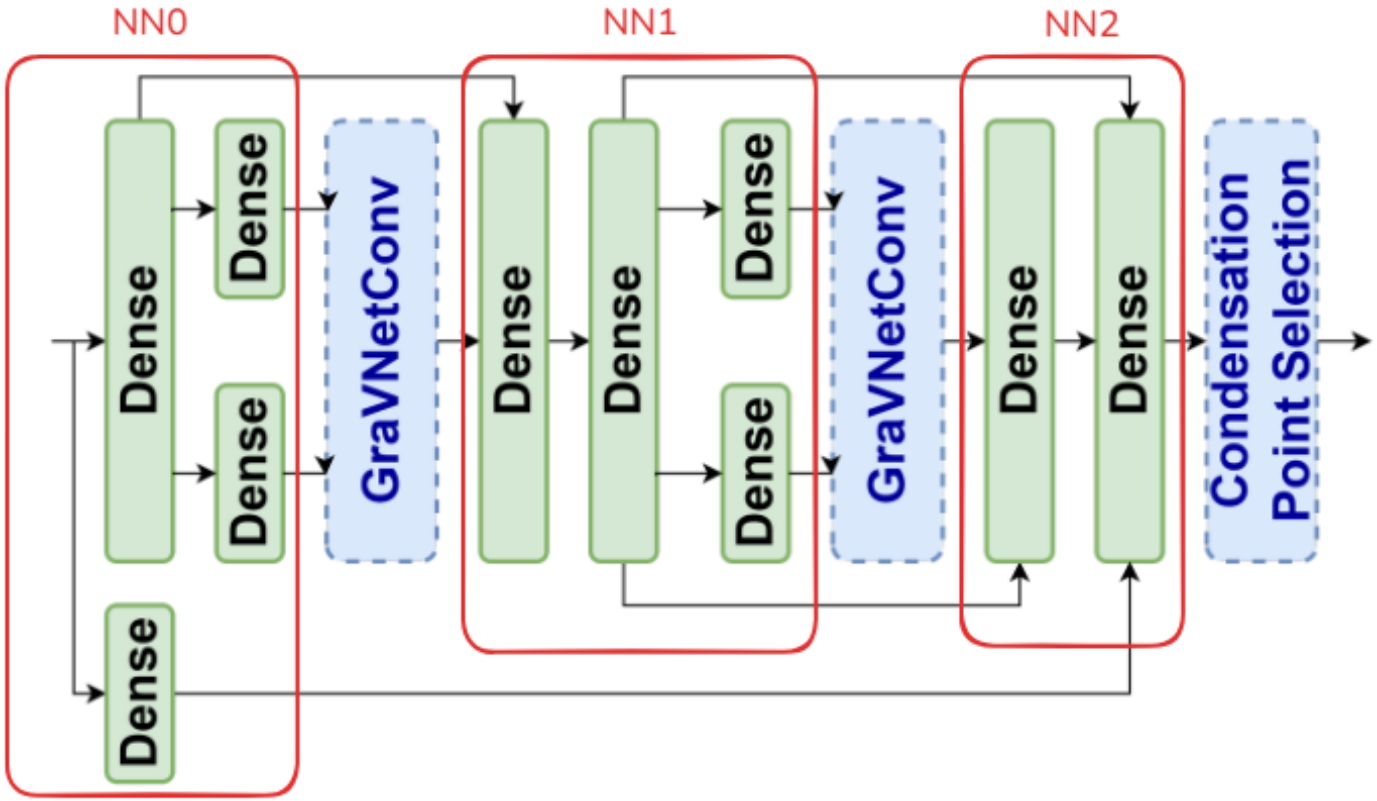


INT8: Efficiencies up to 92%;
for smaller Dimensions only ~40%
→ Scheduling problems with INT8

INT16: Efficiencies up to 95%;
minimal ~60%

NN in the CaloClusterNet

- The CaloClusterNet consists of three neural Networks separated by two GraVNet layers



GravNet

- Graphic shows the needed arithmetic operations
- **Questions**: can this be implemented on the Versal AI Engines

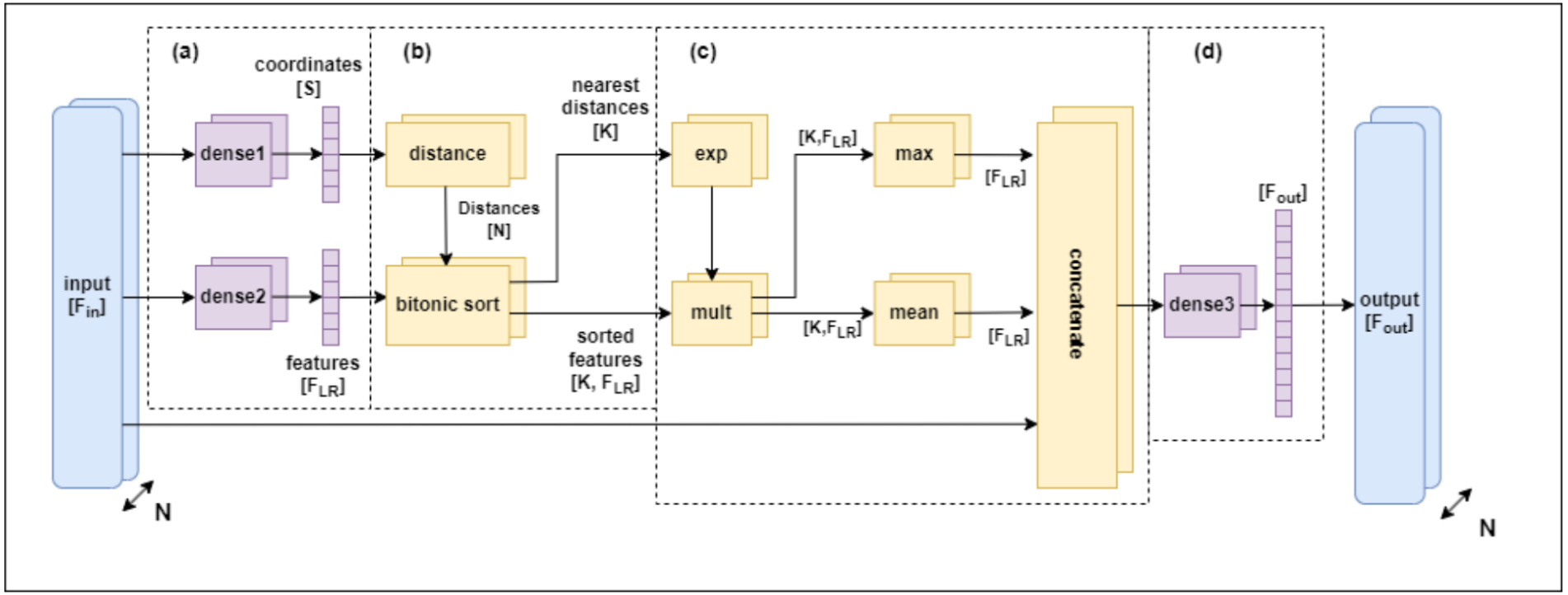
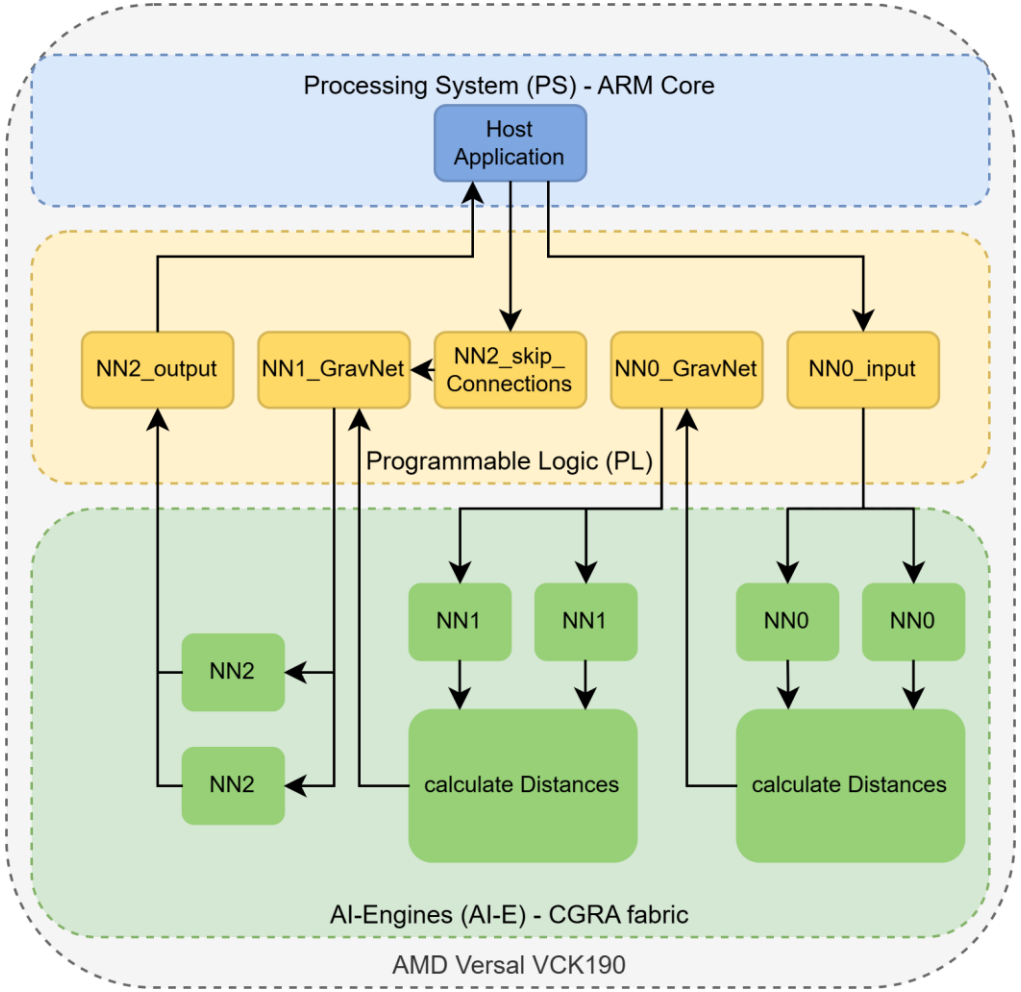


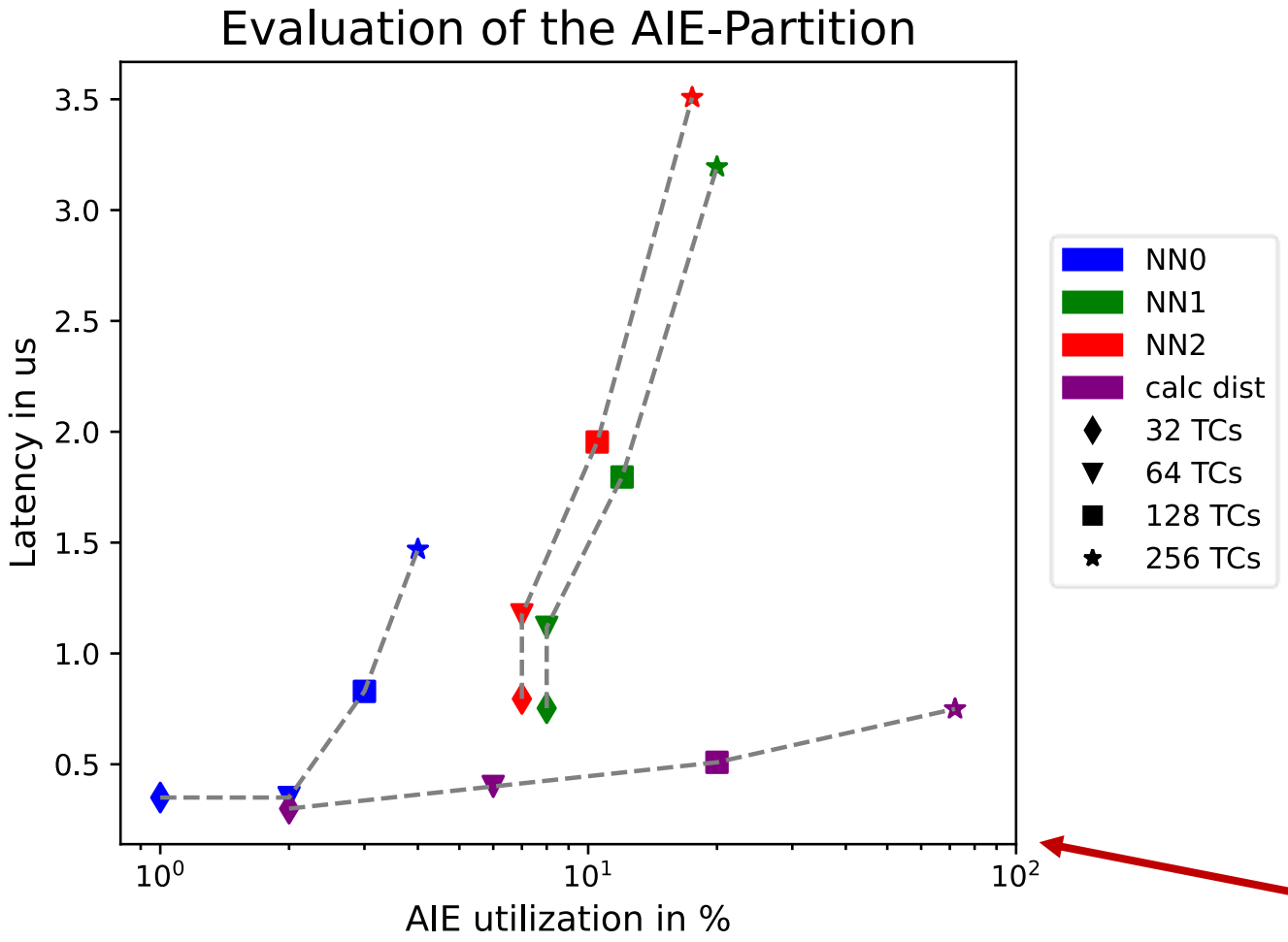
Figure by Shao Shen

Partitioning of the overall Design

- The sort functionality can‘t be implemented using the AIEs while matching the desired throughput
- Make use of the DSPs on the PL fabric & reduce PLIO data transfer → multiplication, max and mean can be moved to the PL
- Calculate distance function is the computational most expensive



Evaluation of the Design



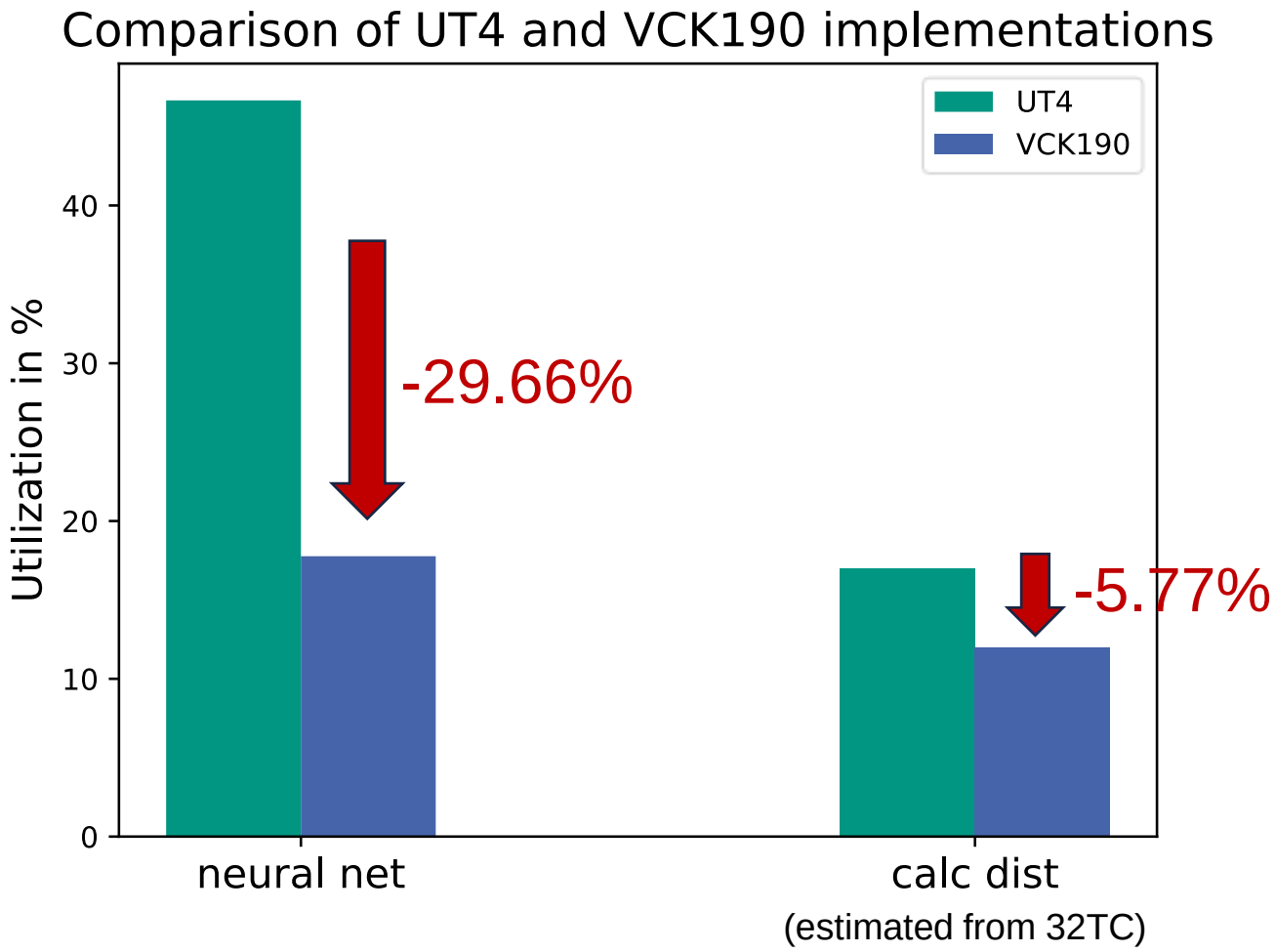
- All implementations satisfy the 8MHz constraint
- Main parts of the latency comes from the neural network
- Main part of the utilization comes from the calc dist

log. scale !

Comparison with UT4 Implementation (64 TCs)

- Comparison of the relative utilization
  - Vck190: AIE-Tiles
  - UT4: Clock Regions
- UT4 implementation by Timo Justinger

- Better utilization for the neural net and the calc dist than the UT4 implementation



Overall Design - Evaluation

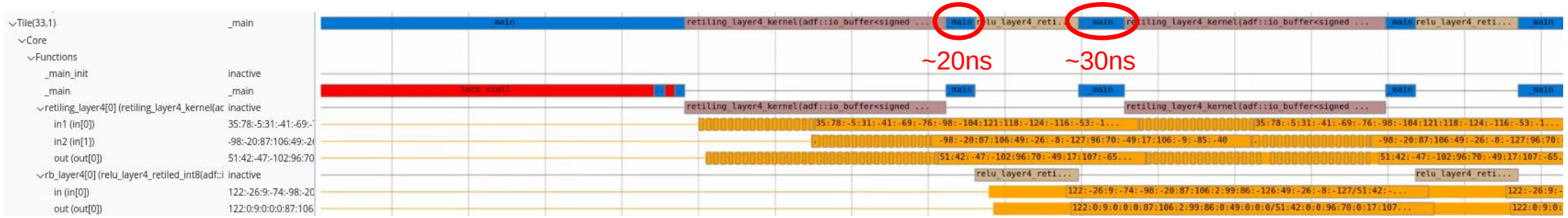
Utilization	Space Multiplexing	NN	Retiling	Calc dist	AIEs total
$\frac{320}{400} = 0.8$	2	34*2=68	3*4=12	2*40=80	160*2=320
Latency	Latency NN	Latency calc dist			
$3.812\mu s$	$2,794\mu s$	$1.018\mu s$			
Throughput					
9.26MHz					

Table 5.6.: Evaluation des Designs für 128 TCs

- Theoretical → in practice an AIE utilization of over 60% leads to a shortage of independent buffer which results in memory stalls (→ ~10% overhead)
- With the sort implemented on the PL **128 TCs could be achievable**

Future work - Challenges

- **Challenge**: free running AIEs → currently not possible for kernels with buffer interfaces (**as a user currently not changeable**)
- Overhead: starting and stopping kernels (min. ~15 cycle) & main-state between kernel executions (min ~20ns, scales with latency)



- →kernels with low latency have a relativly higher overhead than kernels with high latency
- **Solving this problem could reduce latency and increase throughput**

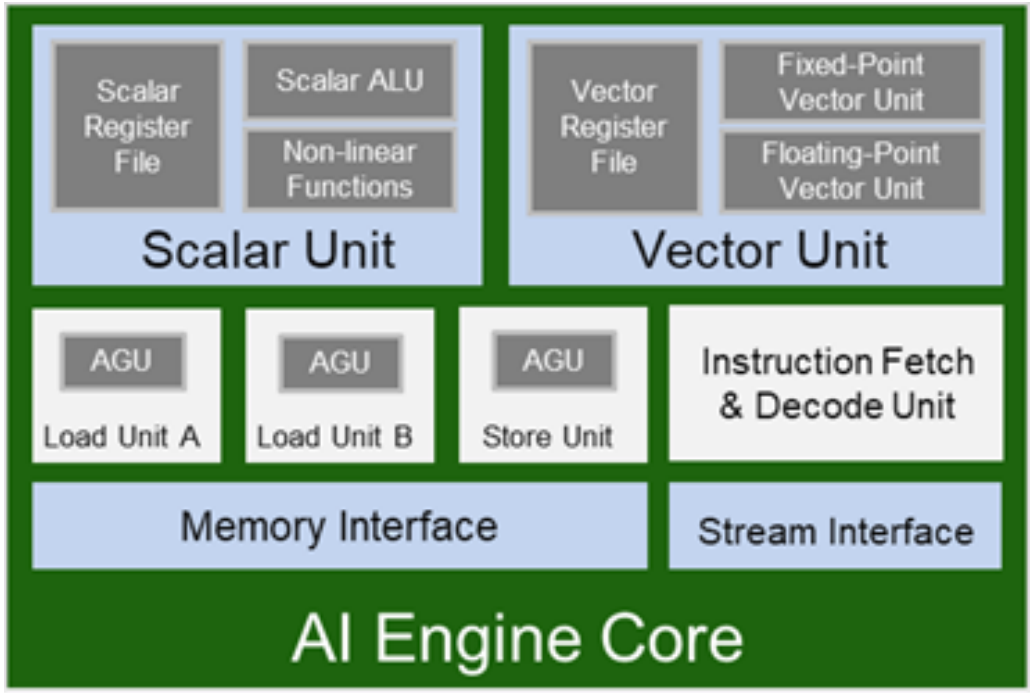
Conclusion

- I have conducted an Design Space Exploration regarding the amount of TCs, latency and throughput for the CaloClusterNet on the Versal platform
- I was able to develop an design of the CaloClusterNet for 128 TCs regarding the AIE-Partition
- I was able to show an increased performance for parts of the CaloClusterNet compared to the current UT4 implementation
- **Outlook:** using AIE-ML as an upgrade to the AI-Engines

Backup-Slides

Versal AI Engines – Backup

- SIMD (single instruction multiple data)
- VLIW (very long instruction word)
  - 7 operations in one
- Scalar & Vector Unit
  - 2048-bit vector register
  - Up to 32 16-bit MAC operations in 1 cycle
- 32KB data memory and 16KB program memory
- Stream interface
  - AXI4 Stream between AIE-tiles



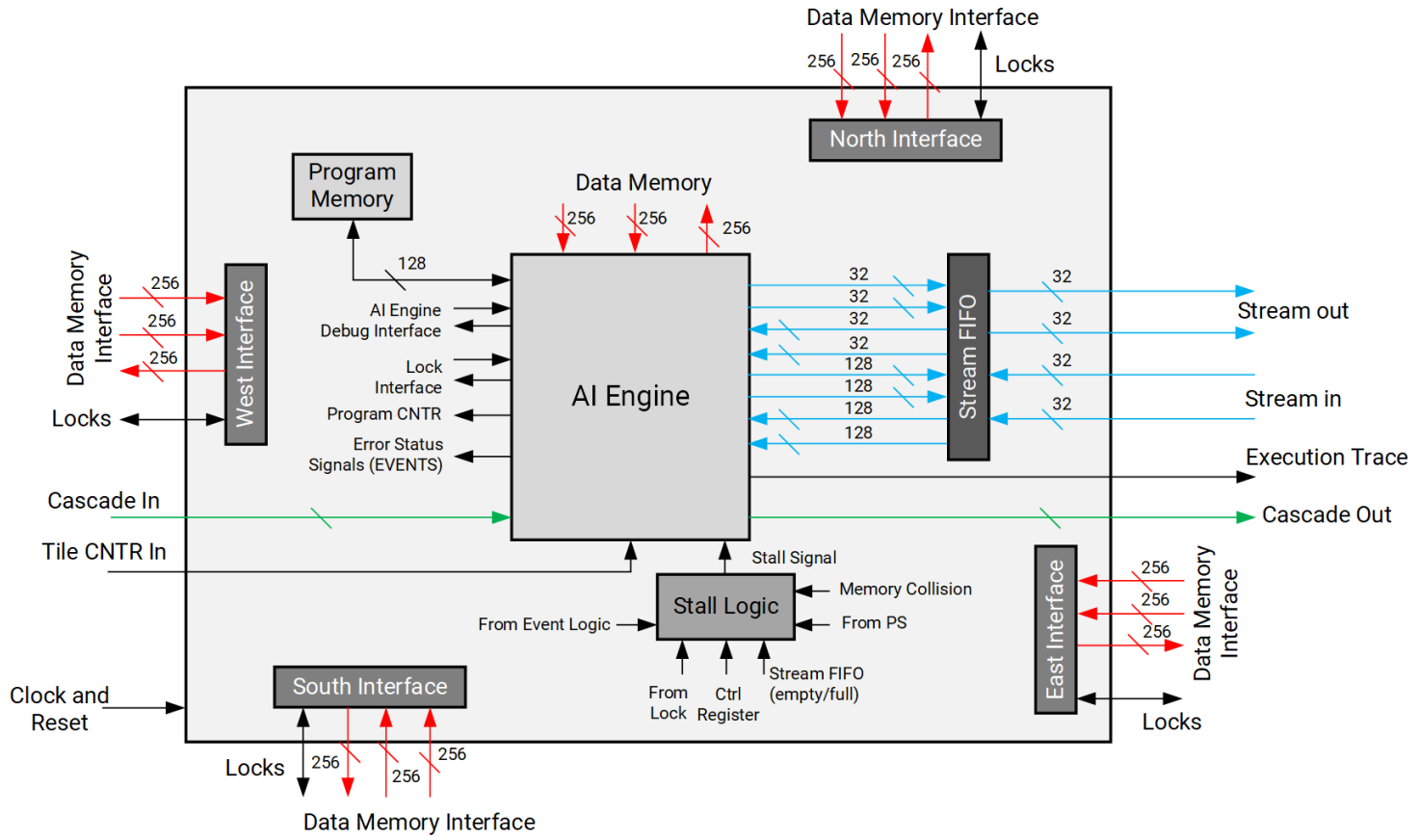
AMD: Versal Adaptive SoC AI Engine Architecture Manual AM009

Communication within the AIE-Array – Backup

Shared Memory with neighboring AIE

AXI 4 Interconnect to all other AIE

Cascade to the next AIE

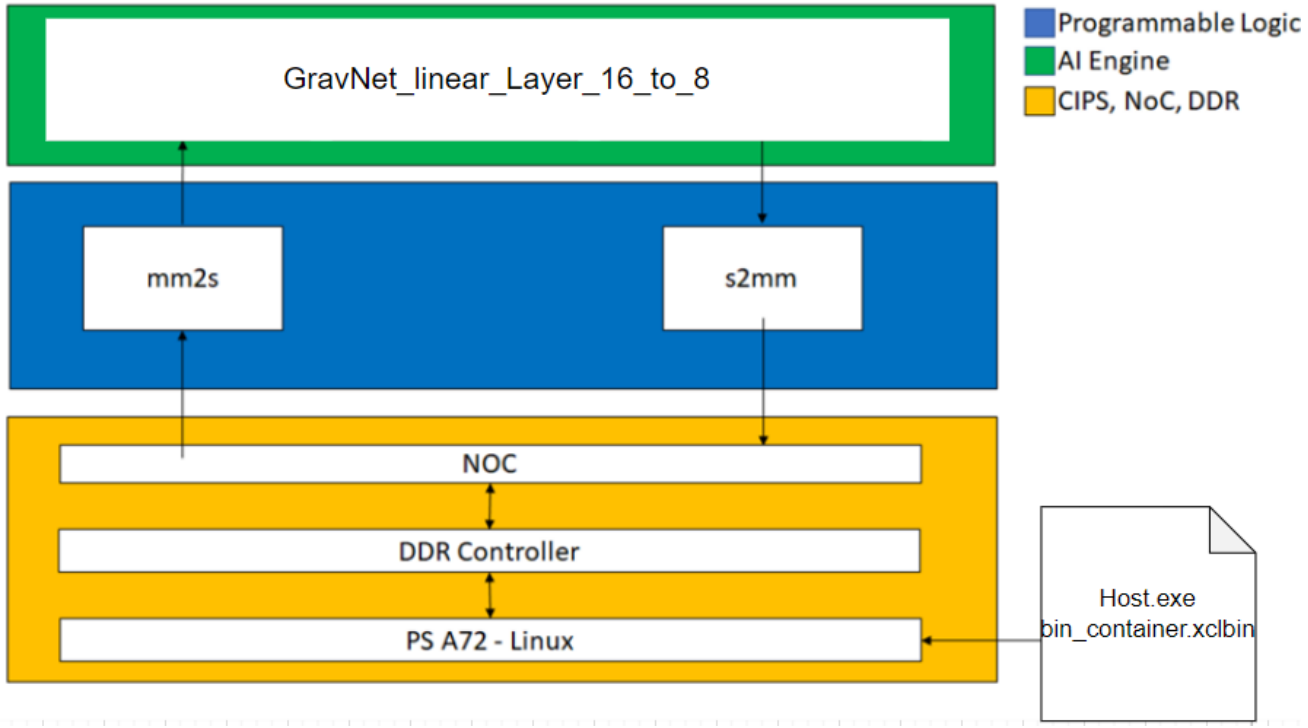


AMD: Versal Adaptive SoC AI Engine Architecture Manual AM009

X20812-071923

System Setup – Backup

- We have successfully build a custom Petalinux Image (runs on the application core) and executed a test application on the board
- An application consists of a .xclbin file that configures the PL and AIE-Array (linear_layer) and an .exe file that communicates with the PL (mm2s, s2mm)



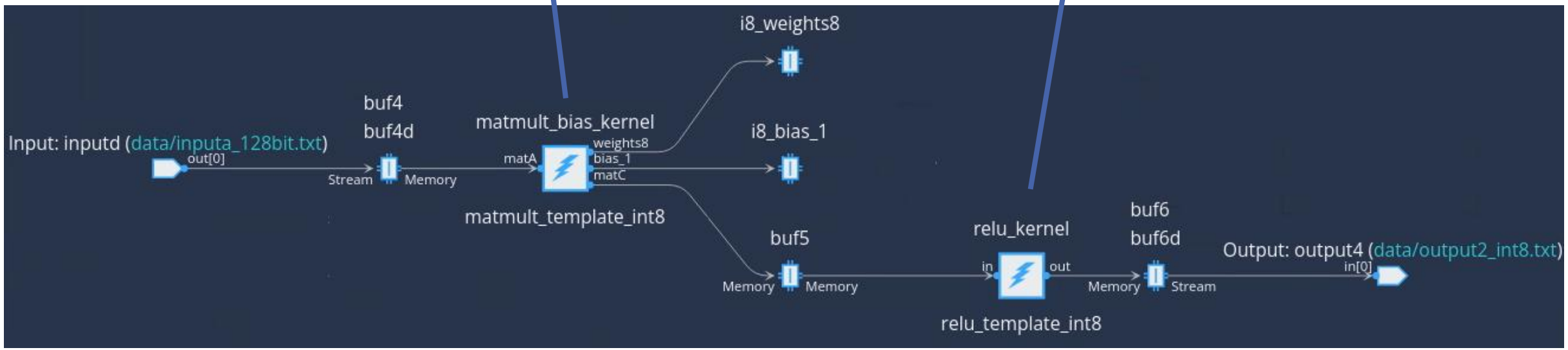
https://github.com/Xilinx/Vitis-Tutorials/tree/2024.1/AI_Engine_Development/AIE/Feature_Tutorials

Implementation & Metrics Lin Dense - Backup

- Bias and Weights implemented as Buffer
- ReLU Function needs to be implemented in an extra Kernel (but can be mapped on the same AIE)

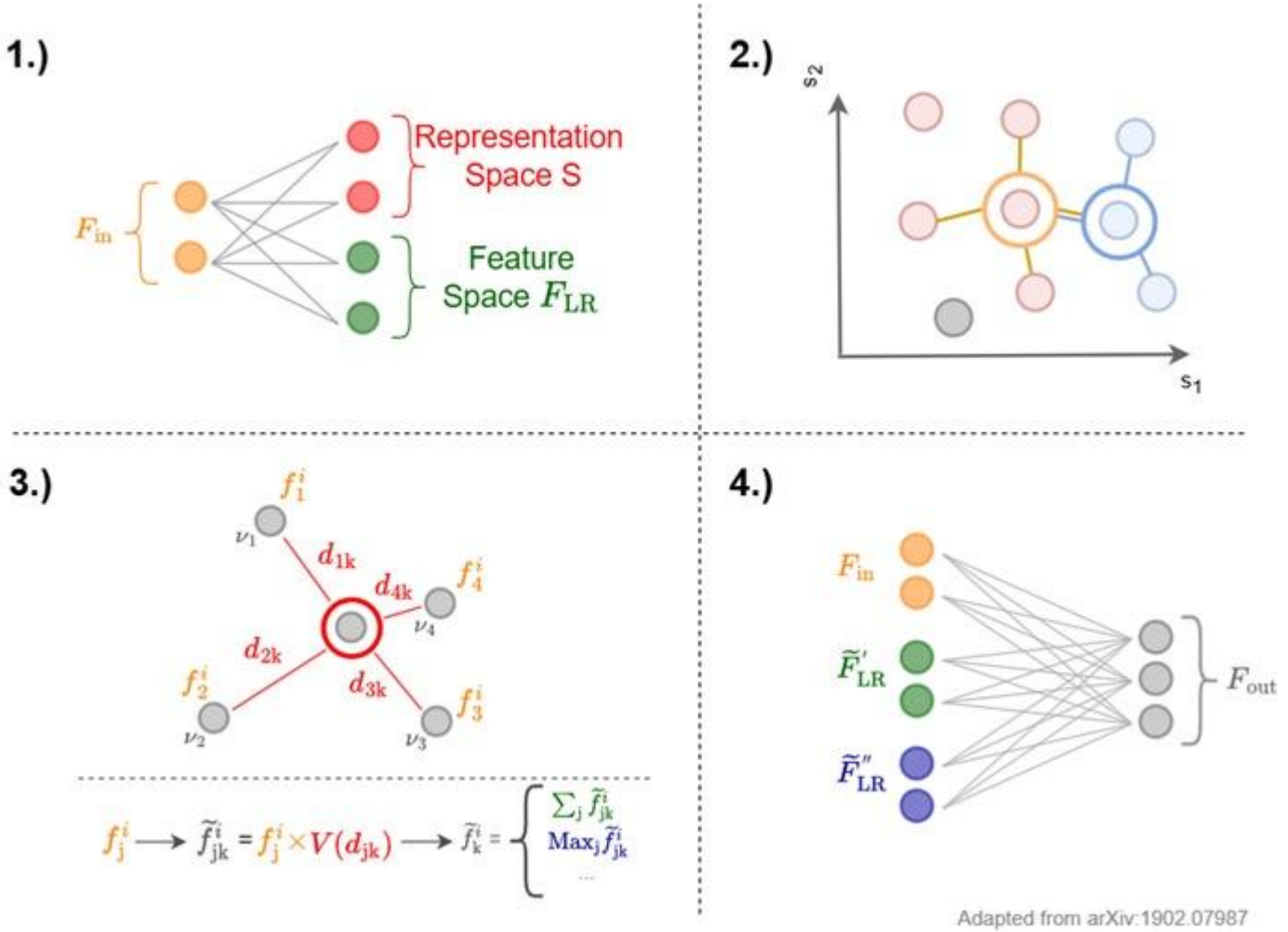
$$cycle_{min} = \frac{Ops}{cycle_{MACs}} + \frac{Fout * PAR}{cycle_{MACs}}$$

$$cycle_{min} = \frac{Fout * PAR}{32}$$



GravNet – Backup

- (1) implemented as linear **dense layer**
- (2) **calculate distances** to all points and **sort** them
- (3) **multiplication**, **max** and **mean** operations
- (4) **concat** data → useable in next linear dense layer

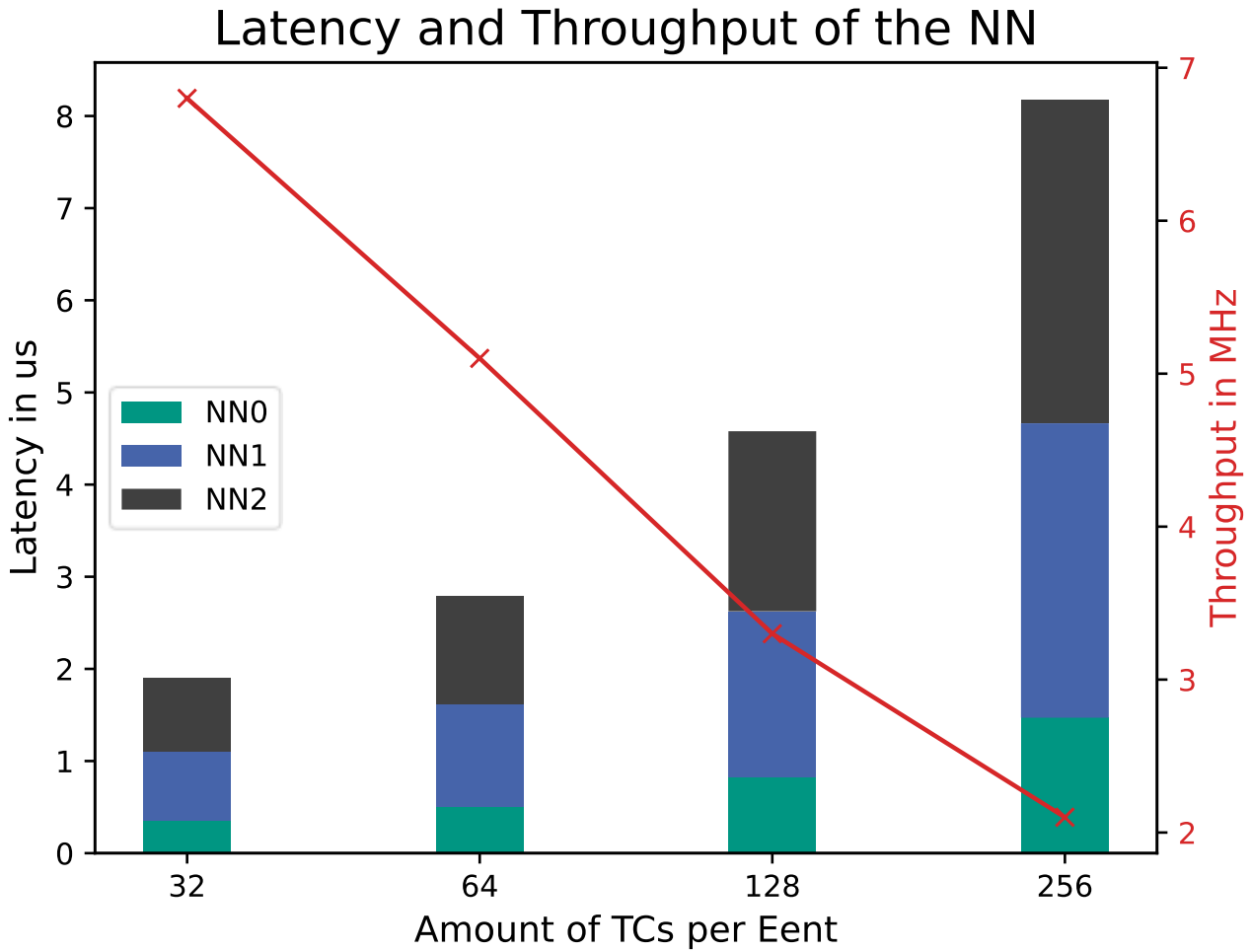


Evaluation of the NN – Backup

- Utilization:
  34/400 AIEs = 8,5%
- Pipeline utilization: 88,1%
- Space Multiplexing:
  Tradeoff
  Latency & Throughput ←→ Utilization

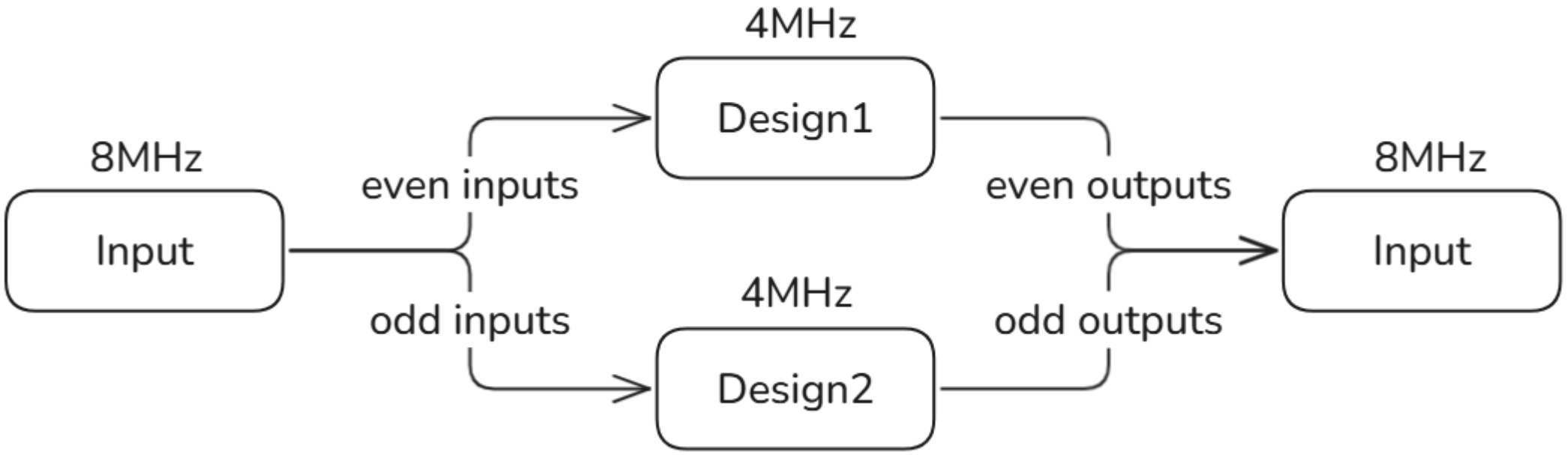
Calc Dist

- Latency for 128 TCs:
  0,509us



Space Multiplexing – Backup

- Uses multiple designs (higher utilization)
- Throughput of the designs add up

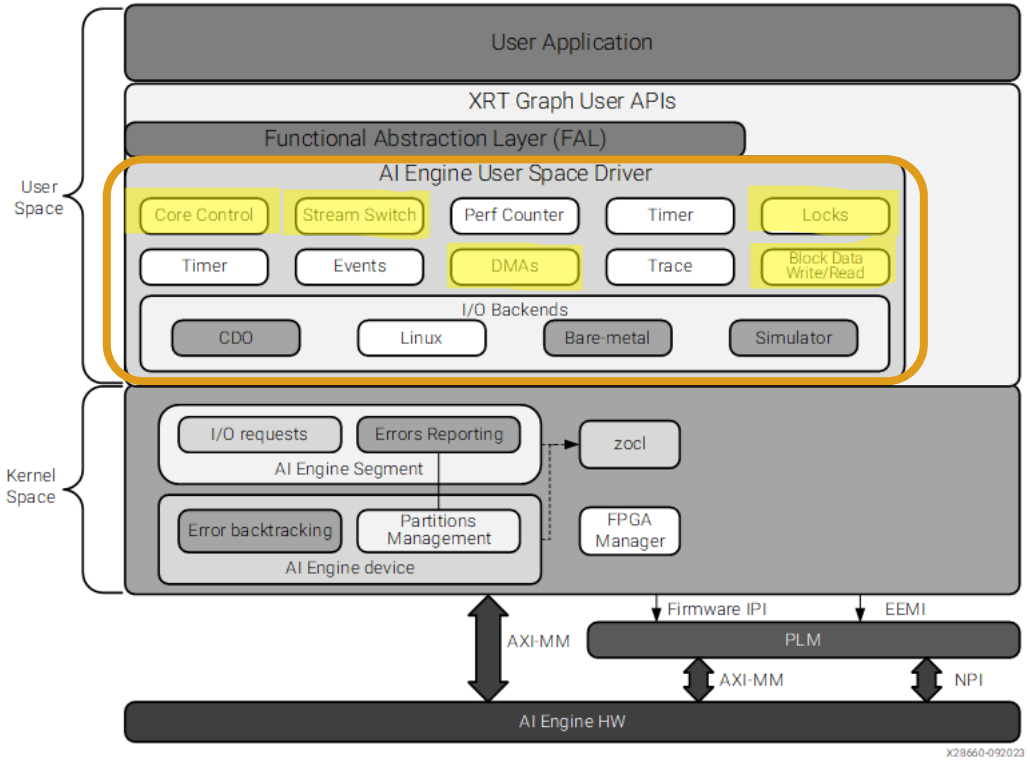


Versal AI Engines – Backup

- Execution of the kernels is handled by AI Engine driver

- Overhead when starting and stopping a kernel → initialization and „free“
- Always ~10+5 cycles → is more important for kernels with low latency
- Overhead between kernel executions at least 15ns

- Free running kernels only with stream interfaces
  - Not enough throughput for this application



AMD: UG1079

Outlook – Backup

- AIE-ML Engines as an upgrade to the AIE-Engines

- Double the amount of memory banks → no memory stalls
- for int8 four times more operations per cycle
- Less overhead for neural network implementation due to symmetric mmul intrinsic
- Supports sparse matrix multiplication and unsigned datatypes (both are used in the current UT4 implementation)

- Very new → only one possible Board (vek280) with 304 AIE-ML Tiles and smaller PL than the vck190