JLU
NEUE WEGE. SEIT 1607.
JUSTUS-LIEBIG-UNIVERSITÄT GIESSEN

JENNIFER2
EU grant n.822070

Bundesministerium
für Bildung
und Forschung
ErUM-FSP T09 Belle II

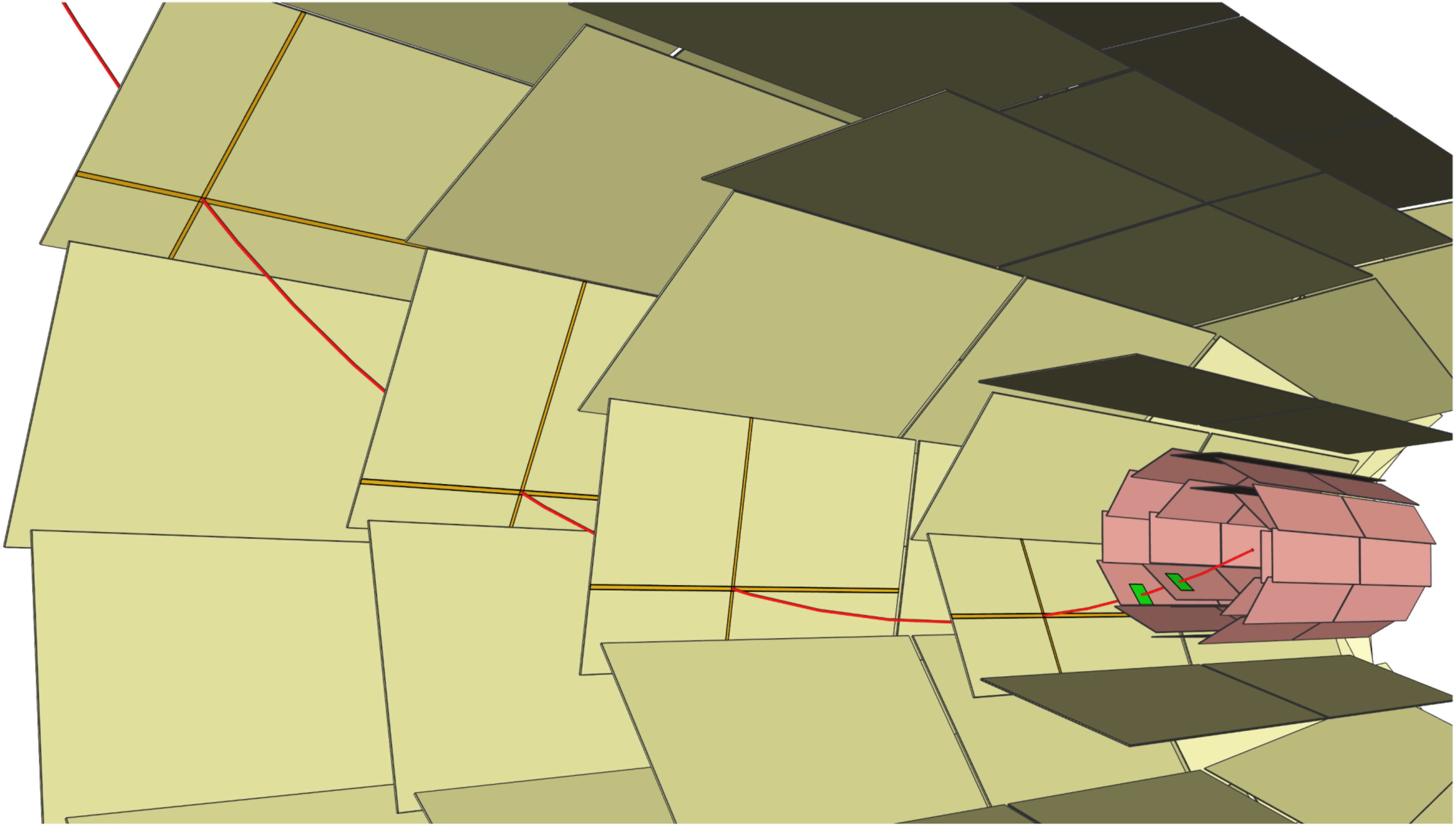
Versal @ Giessen: very, very first tests and experiences and possible application for the Belle II PXD (related to anomaly detection)

Jens Sören Lange
(Justus-Liebig-Universität Giessen)

WORKSHOP ON FAST REALTIME SYSTEMS AND REALTIME MACHINE LEARNING
Justus–Liebig–Universität Giessen
08.04.–11.04.2024

SLOW PION CLASSIFICATION

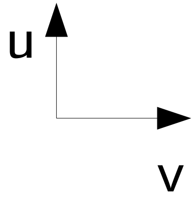
ROI selection in PXD DAQ

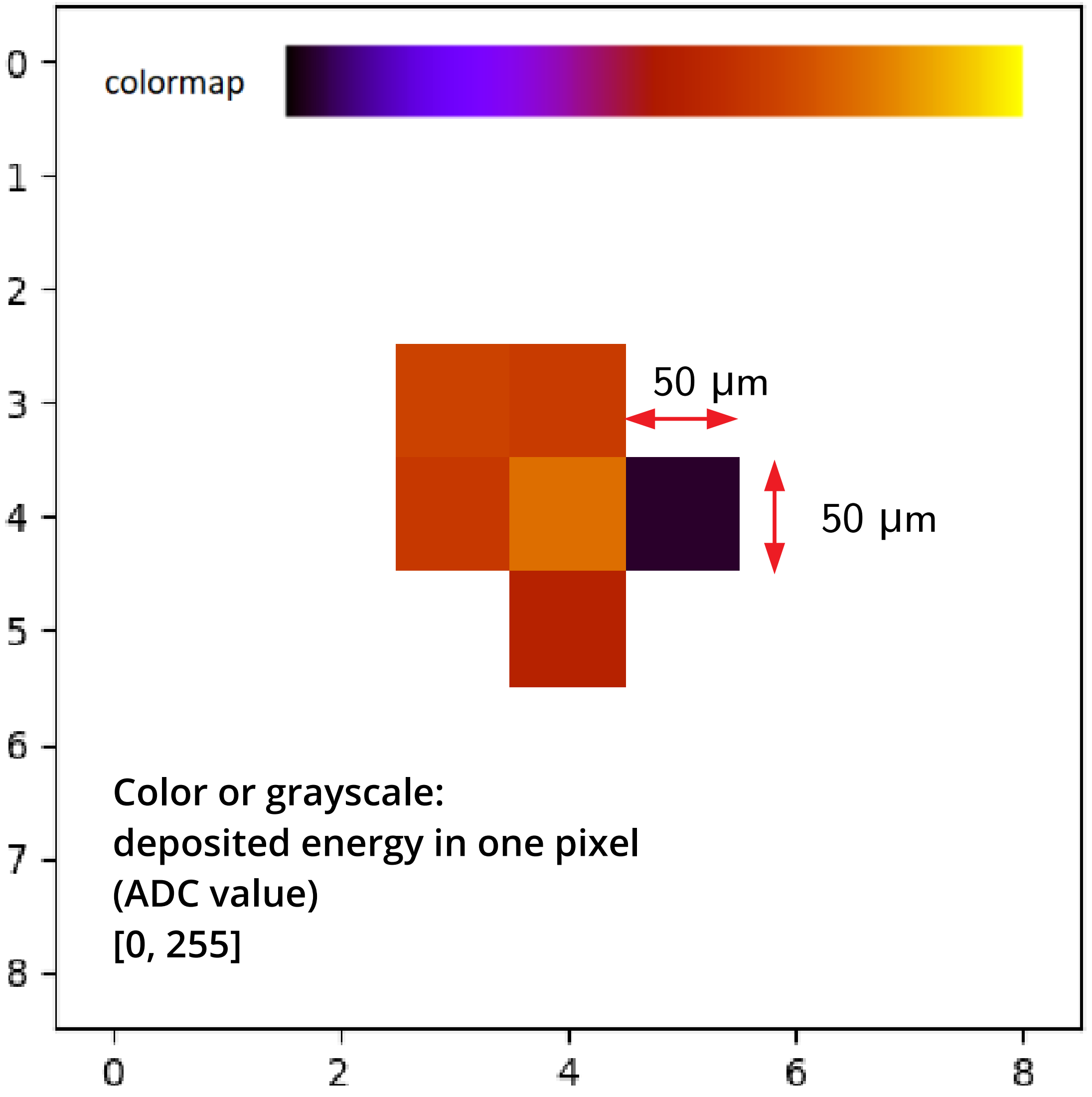


ROI selection in PXD DAQ

- prepared, tested and will be switched on at higher luminosities, when output of PXD DAQ saturates event builder input (1 GB/s total for all 32 links, 2025–2027, depending on accelerator)
  - disadvantage for highly ionising particles
    - high dE/dx, stopped in PXD
    - no HLT track generated
    - no ROI generated
    - related PXD hits are deleted (before reaching tape)
  - Signal example:
    - slow pions $p_T < 200$ MeV/c

Input for neural networks (NN), 9x9 pixel matrix



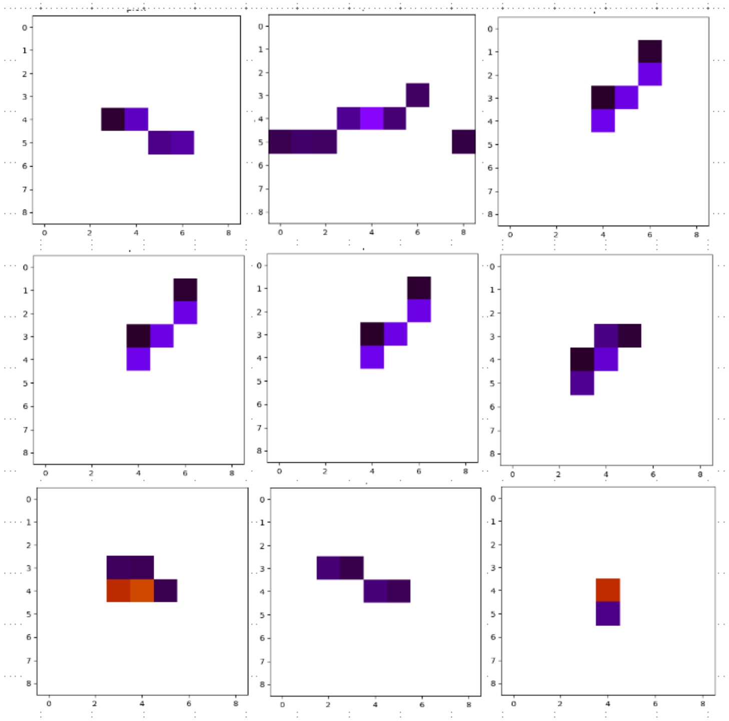


Color or grayscale:
deposited energy in one pixel
(ADC value)
[0, 255]

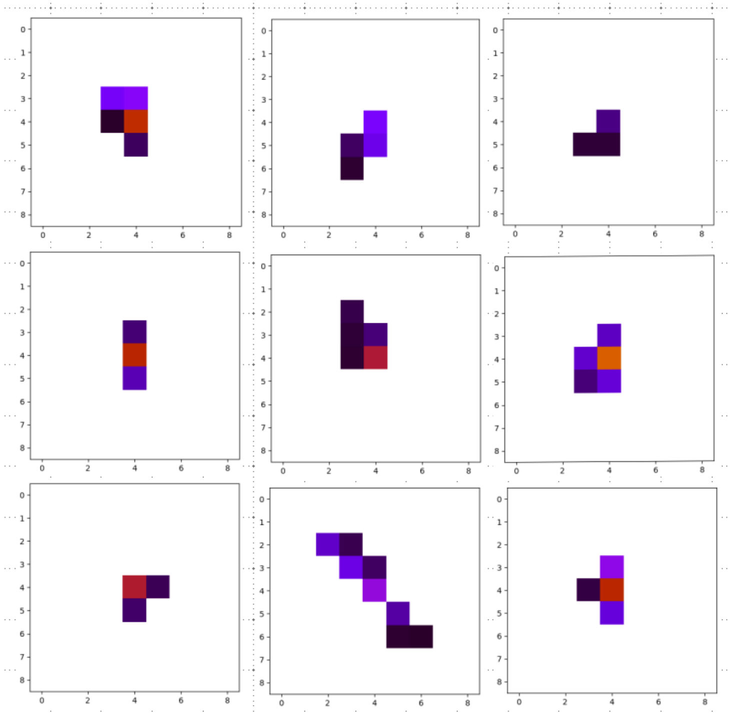
PXD2 contains ~8 million pixels.

9x9 pixel matrix
Signal & background

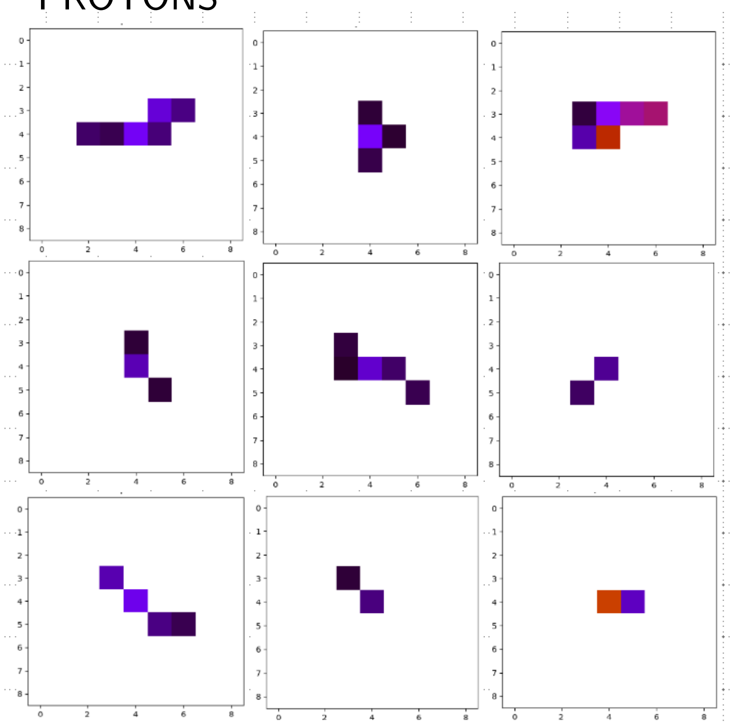
PIONS



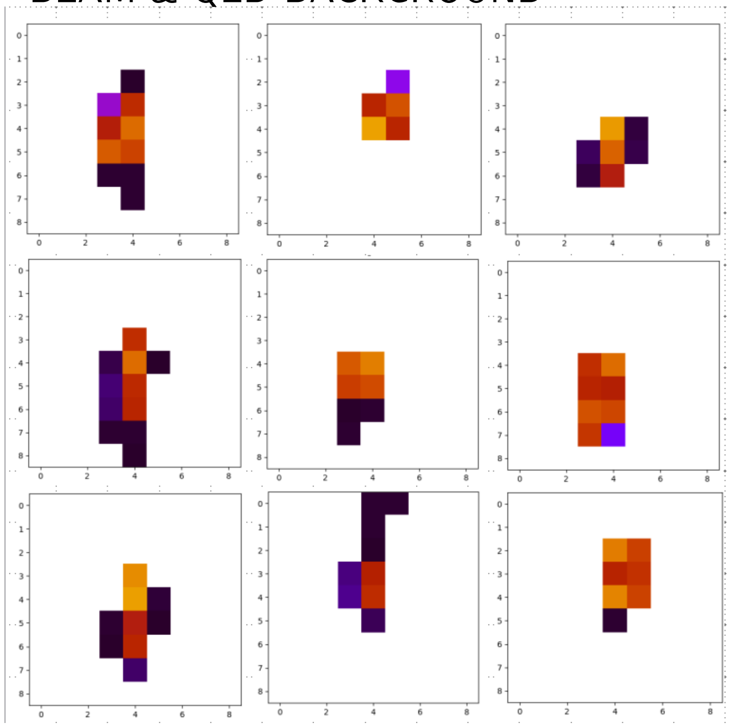
ANTIDEUTERONS



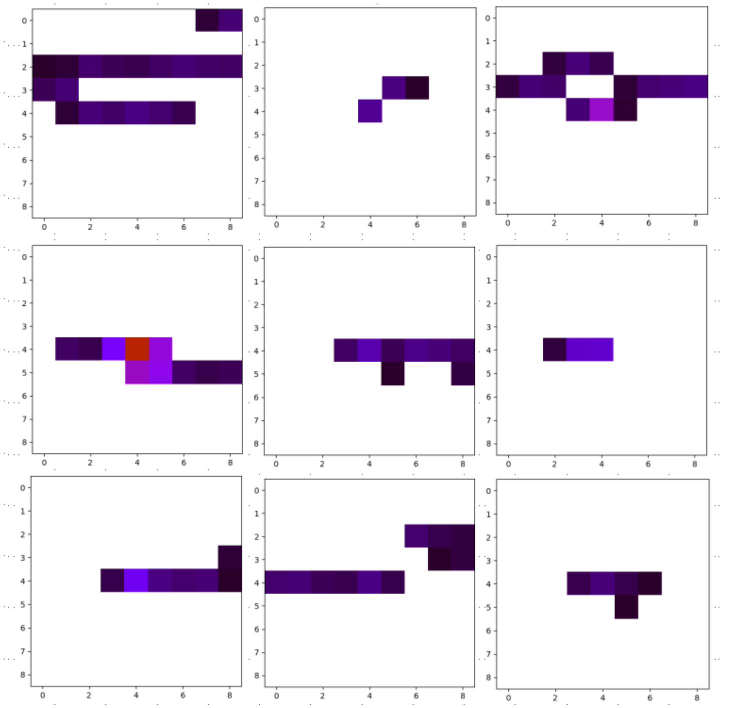
PROTONS



BEAM & QED BACKGROUND



MAGNETIC MONOPOLES



Slow pions vs. electrons (from QED processes)

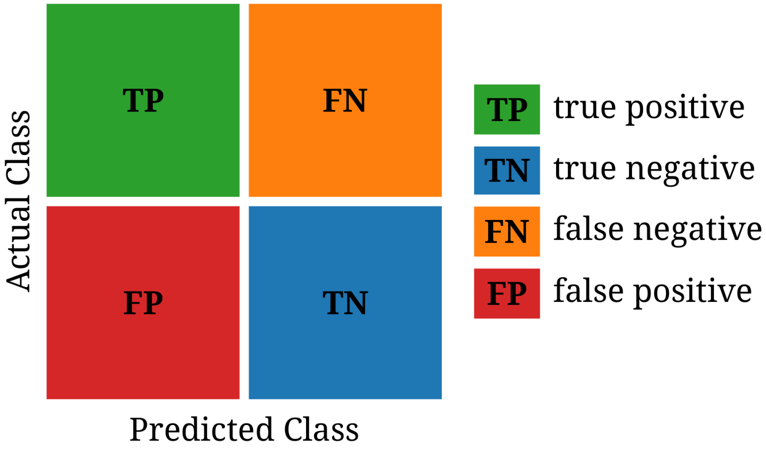
Sensitivity

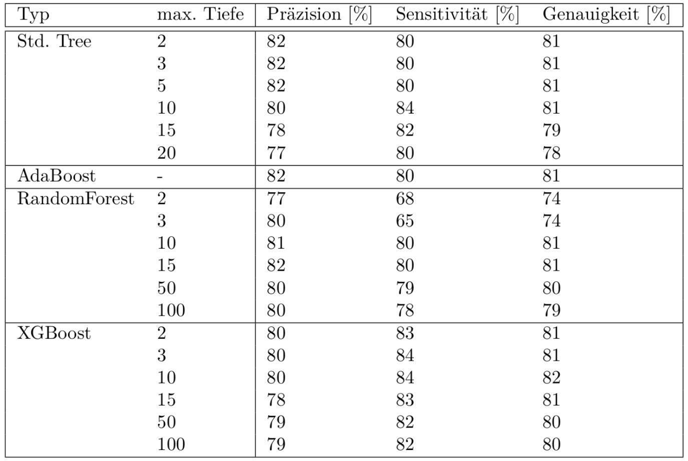
Number of events correctly identified by the NN as a signal (TP), divided by all real signal events (TP + FN), also called "effciency" in particle physics

Precision

Number of events correctly identified by the NN as a signal (TP), divided by all events identified by the NN as a signal (TP + FP), also called "purity" in particle physics

		Sensititivity (Pion Efficiency)	Precision (Pion Purity)
CNN	Johannes Bilk (master thesis)	82%	81%
Support Vector Machines	Timo Schellhaas (bachelor thesis)	83%	69%
Decision Trees, RandomForest	Stephanie Käs (master thesis)	82%	80%
GNN	Martin Beyer (master student project)	86%	82%

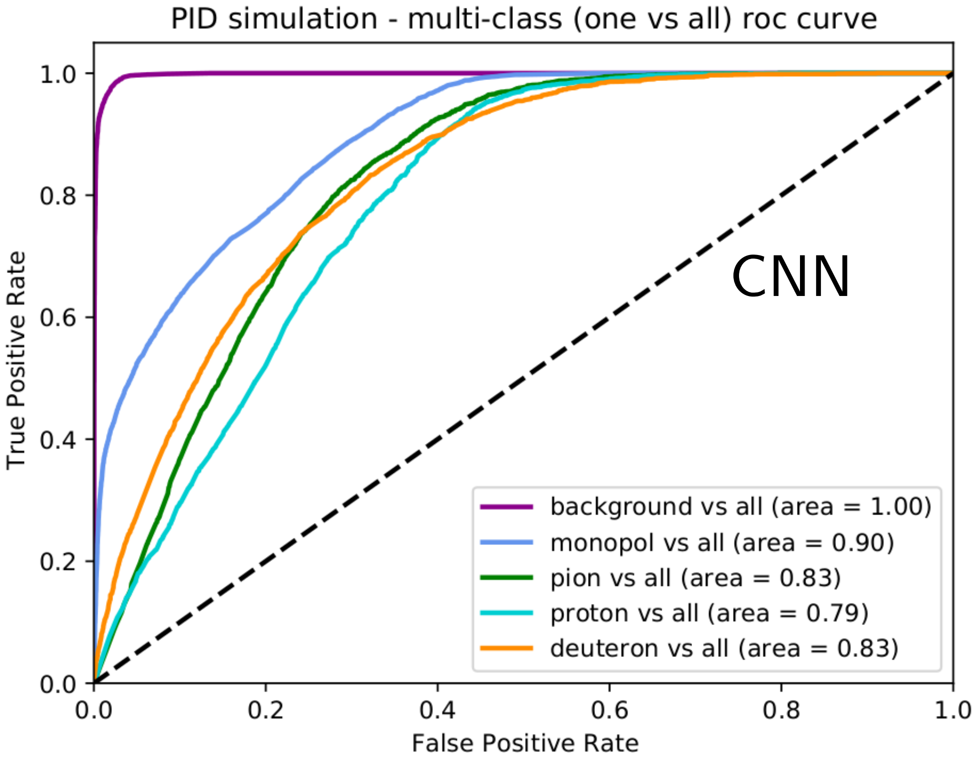


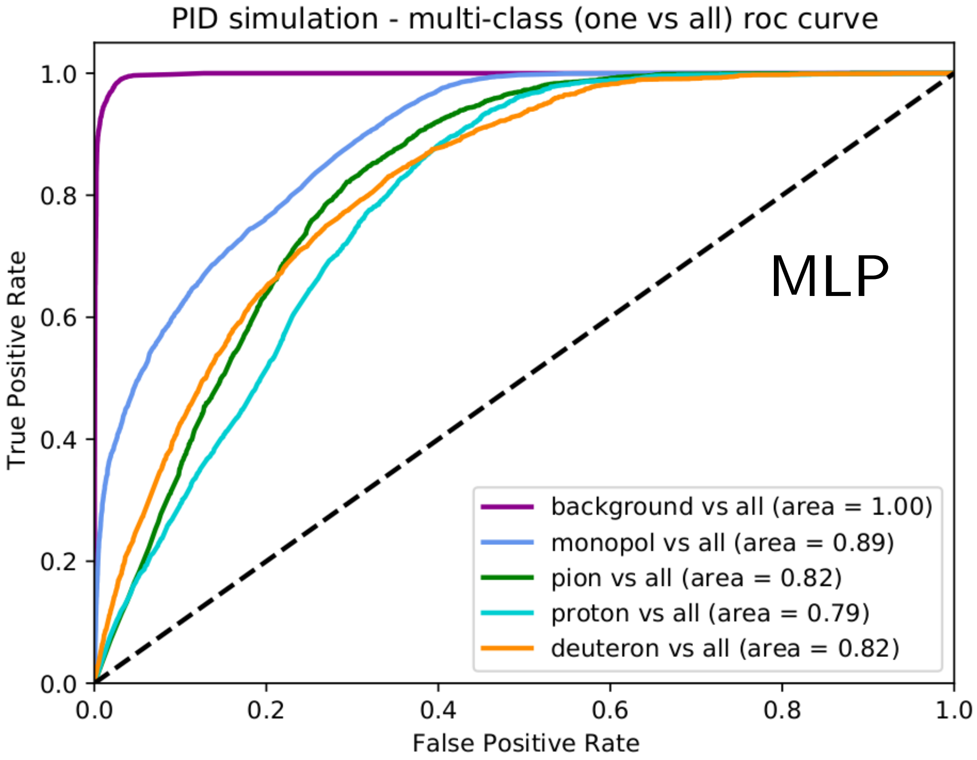


Typ	max. Tiefe	Präzision [%]	Sensitivität [%]	Genauigkeit [%]
Std. Tree	2	82	80	81
	3	82	80	81
	5	82	80	81
	10	80	84	81
	15	78	82	79
	20	77	80	78
AdaBoost	-	82	80	81
RandomForest	2	77	68	74
	3	80	65	74
	10	81	80	81
	15	82	80	81
	50	80	79	80
	100	80	78	79
XGBoost	2	80	83	81
	3	80	84	81
	10	80	84	82
	15	78	83	81
	50	79	82	80
	100	79	82	80

Slow pions (pT<200 MeV/c) vs. other particles

Marvin Peter, master student project, 2020, TensorFlow & Keras, CPU and GPU

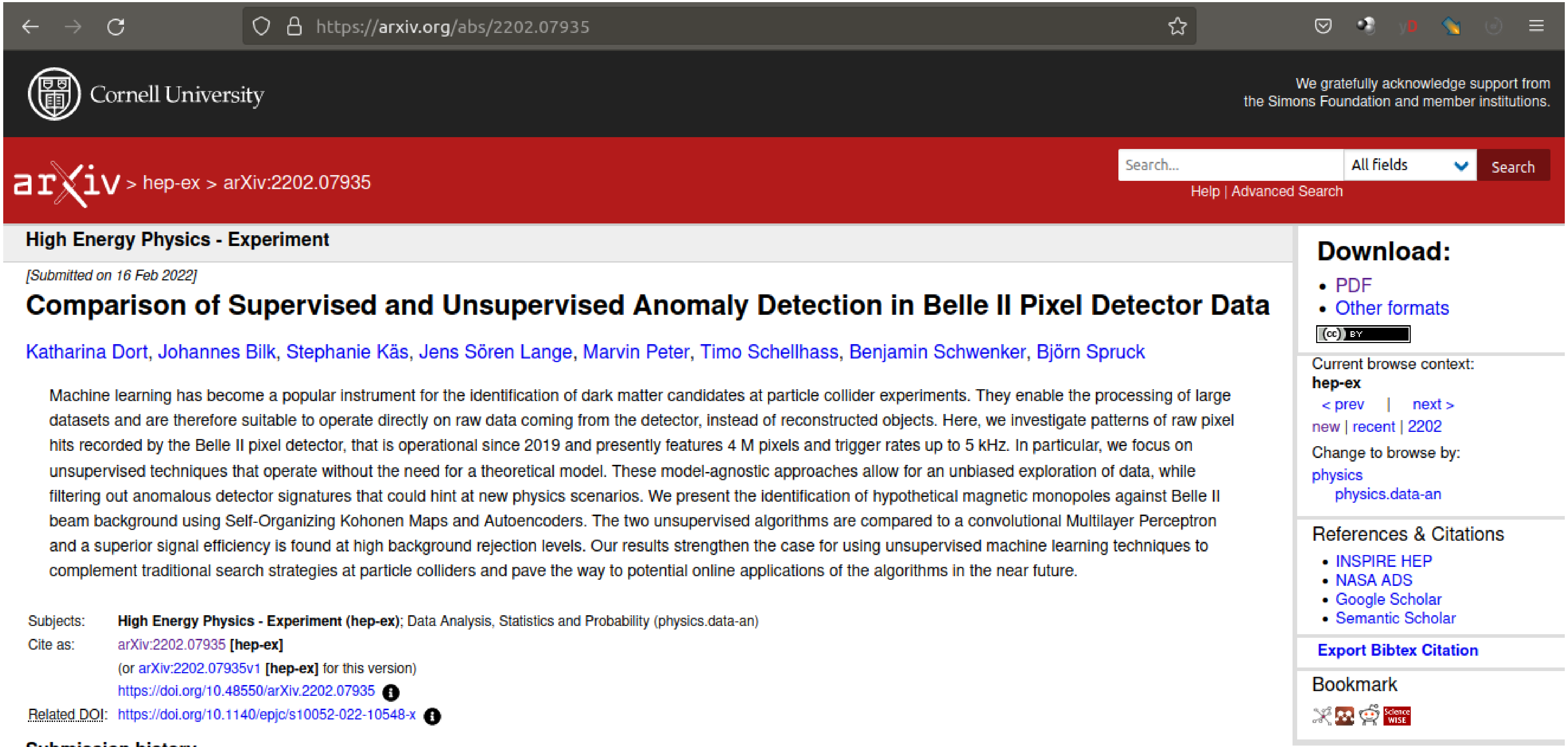




What it has to do with
ANOMALY DETECTION?

We also used 9x9 PXD pixel matrices (“images”) for a classification task of different data (see next slide).

Magnetic monopoles vs. beam background



High Energy Physics - Experiment

[Submitted on 16 Feb 2022]

Comparison of Supervised and Unsupervised Anomaly Detection in Belle II Pixel Detector Data

Katharina Dort, Johannes Bilk, Stephanie Käs, Jens Sören Lange, Marvin Peter, Timo Schellhass, Benjamin Schwenker, Björn Spruck

Machine learning has become a popular instrument for the identification of dark matter candidates at particle collider experiments. They enable the processing of large datasets and are therefore suitable to operate directly on raw data coming from the detector, instead of reconstructed objects. Here, we investigate patterns of raw pixel hits recorded by the Belle II pixel detector, that is operational since 2019 and presently features 4 M pixels and trigger rates up to 5 kHz. In particular, we focus on unsupervised techniques that operate without the need for a theoretical model. These model-agnostic approaches allow for an unbiased exploration of data, while filtering out anomalous detector signatures that could hint at new physics scenarios. We present the identification of hypothetical magnetic monopoles against Belle II beam background using Self-Organizing Kohonen Maps and Autoencoders. The two unsupervised algorithms are compared to a convolutional Multilayer Perceptron and a superior signal efficiency is found at high background rejection levels. Our results strengthen the case for using unsupervised machine learning techniques to complement traditional search strategies at particle colliders and pave the way to potential online applications of the algorithms in the near future.

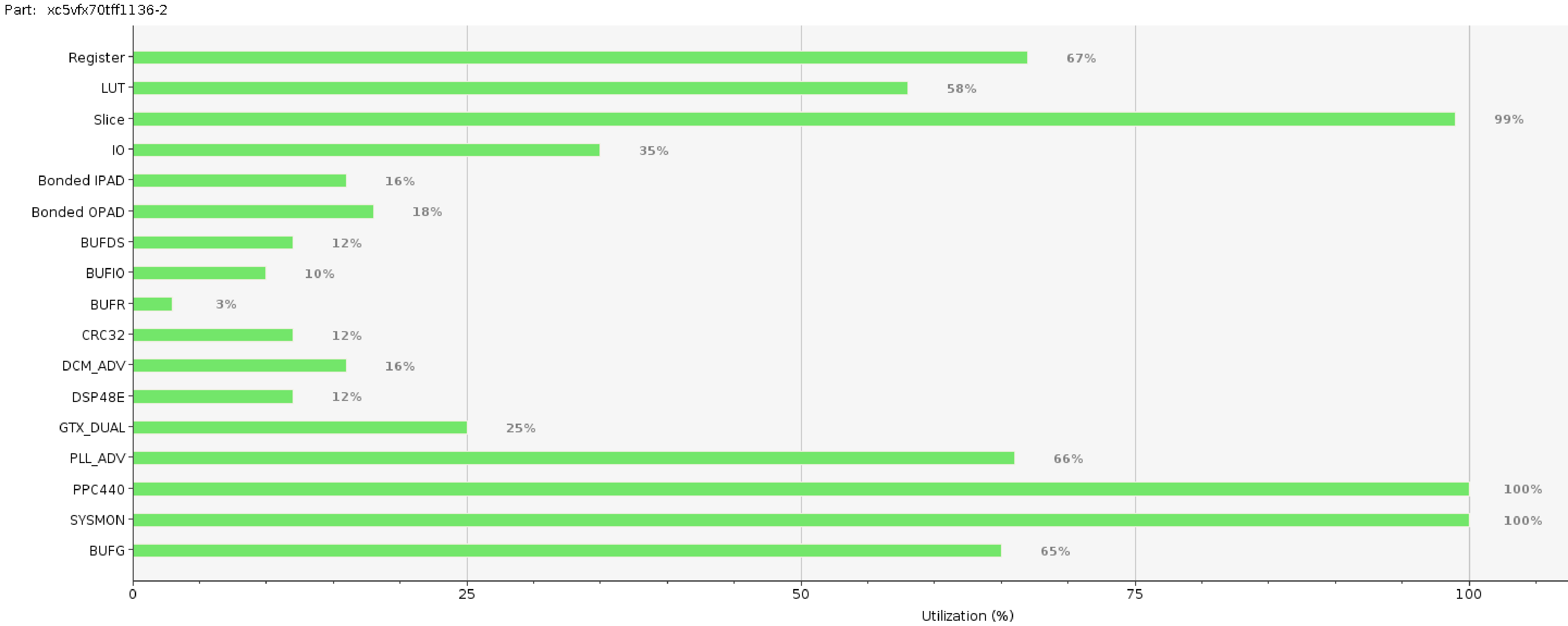
Subjects:	**High Energy Physics - Experiment (hep-ex)**; Data Analysis, Statistics and Probability (physics.data-an)
Cite as:	arXiv:2202.07935 **[hep-ex]**
	(or arXiv:2202.07935v1 **[hep-ex]** for this version)
	https://doi.org/10.48550/arXiv.2202.07935
Related DOI:	https://doi.org/10.1140/epjc/s10052-022-10548-x

Submission history

From: Katharina Dort [view email]
[v1] Wed, 16 Feb 2022 08:54:41 UTC (11,033 KB)

K. Dort et al., Eur. Phys. J. C 82 (2022) 7, 587

FPGA Resources of one PXD DAQ Selector Card



Multiport memory controller can only use 2 GB RAM so far, due to limit of resource

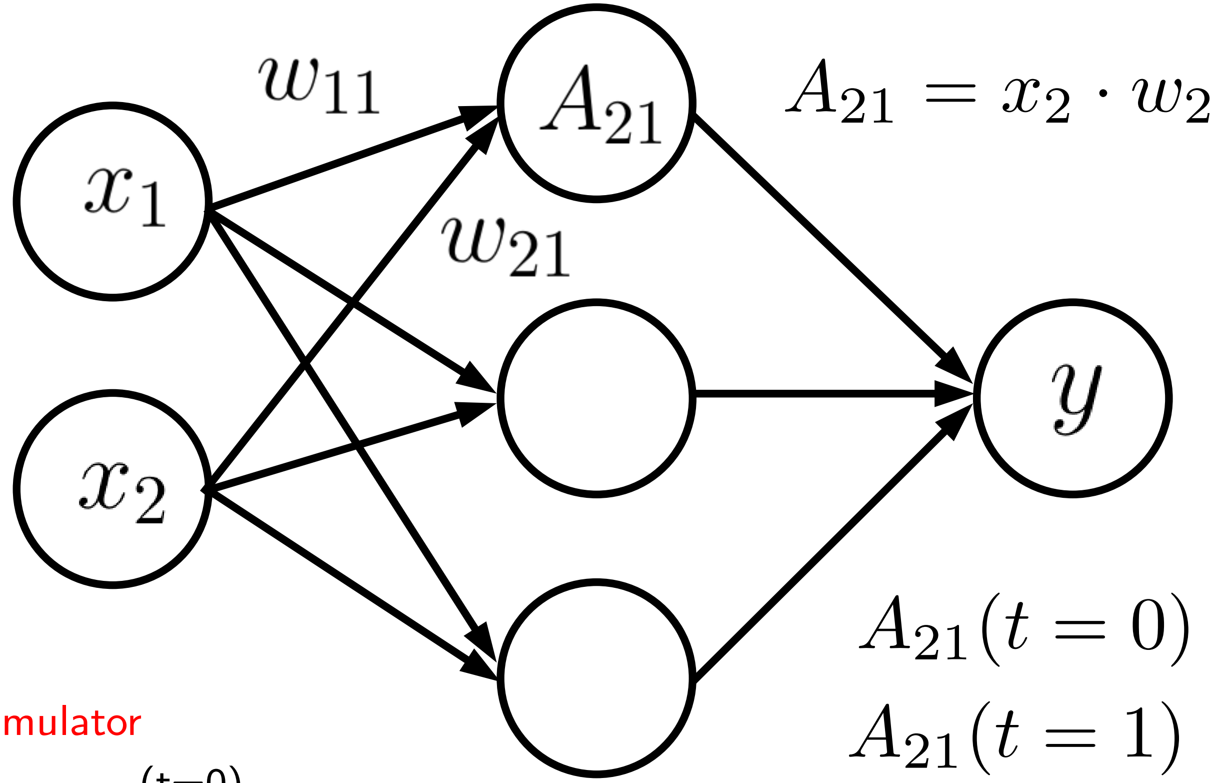
PPC440 is a “hard” processor (not existing anymore in newer FPGAs)

99% of (conventional) slices are used (!)

12% of DSP slices used (e.g. coordinate transforms), i.e. 88% available (possible solution)

Simon Reiter, Giessen

What operations do we need?



$$A_{21} = x_2 \cdot w_{21} + x_2 \cdot w_{21}$$

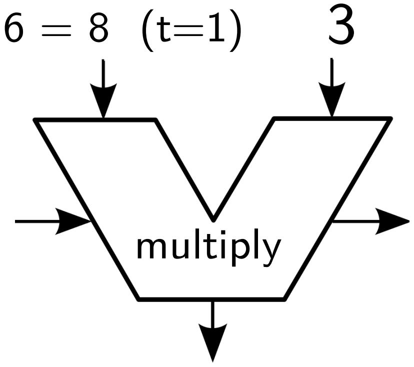
$$A_{21}(t=0) = x_1 \cdot w_{11}$$
$$A_{21}(t=1) = \underbrace{x_2 \cdot w_{21} + A_{21}(t=0)}_{}$$

This is called a “multiply/accumulate” (MAC) operation, and DSP elements in FPGAs are suited.

Accumulator

2 (t=0)

2 + 6 = 8 (t=1)



2 x 3 = 6

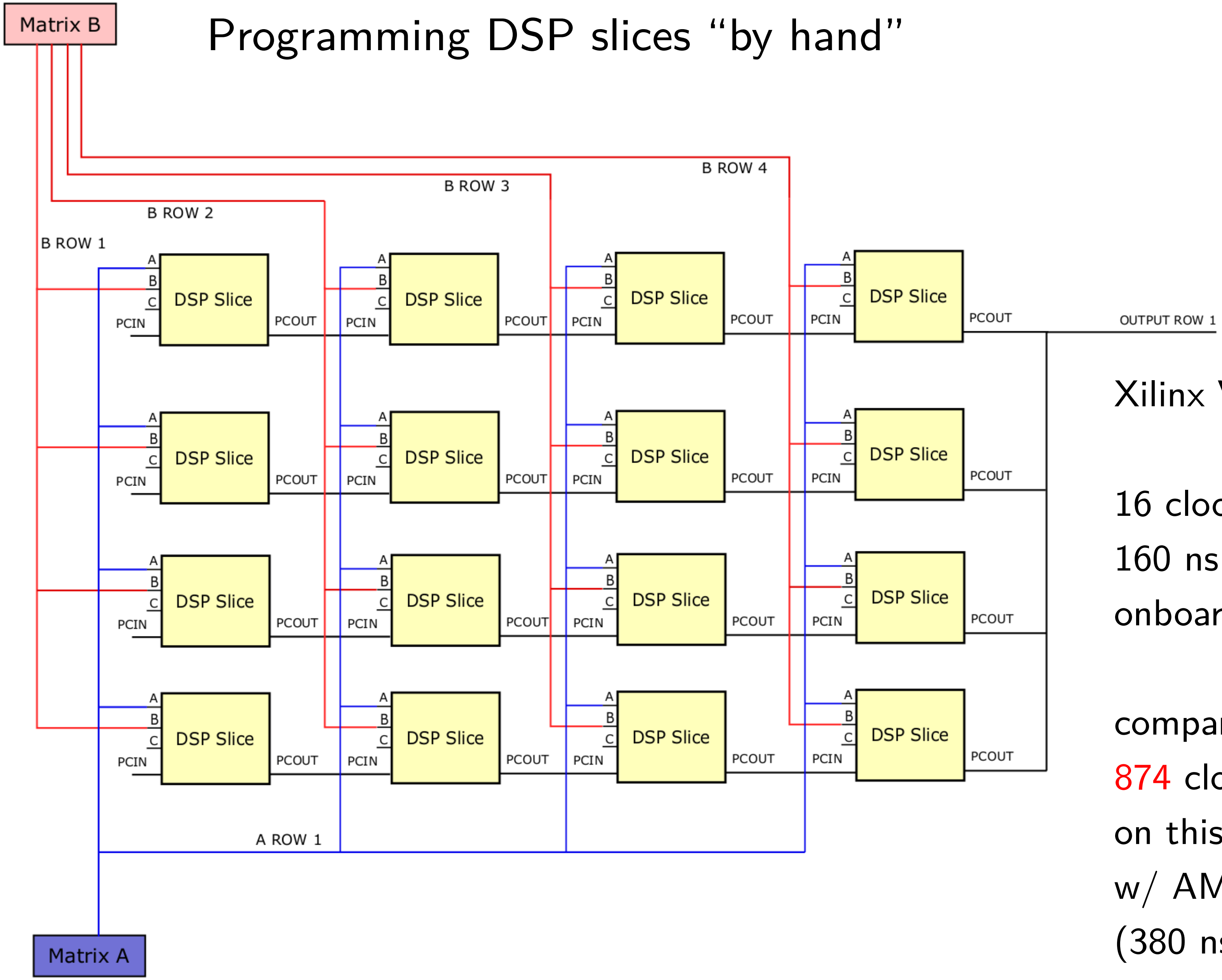
Side remarks:

– Node bias values need one more addition (can be done with LUTs)

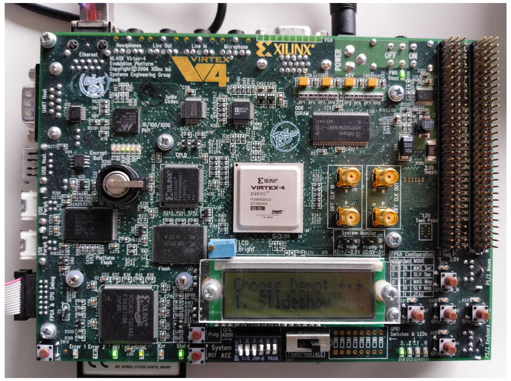
– Activation function needs lookup tables (can be done with block RAM)

Matrix multiplication on an FPGA

Programming DSP slices "by hand"



Falk Zorn, JLU Giessen



ML403 evaluation board

Xilinx Virtex-4 FX12

16 clock cycles
160 ns @ 100 MHz
onboard clock

compare:
874 clock cycles
on this HP notebook
w/ AMD Ryzen 5
(380 ns @ 2.3 GHz)

January 30, 2024: Versal unboxing

Versal Prime 1202 (XCVP1202–2MSEVSVA2785)

compared to Virtex–5 FX70T

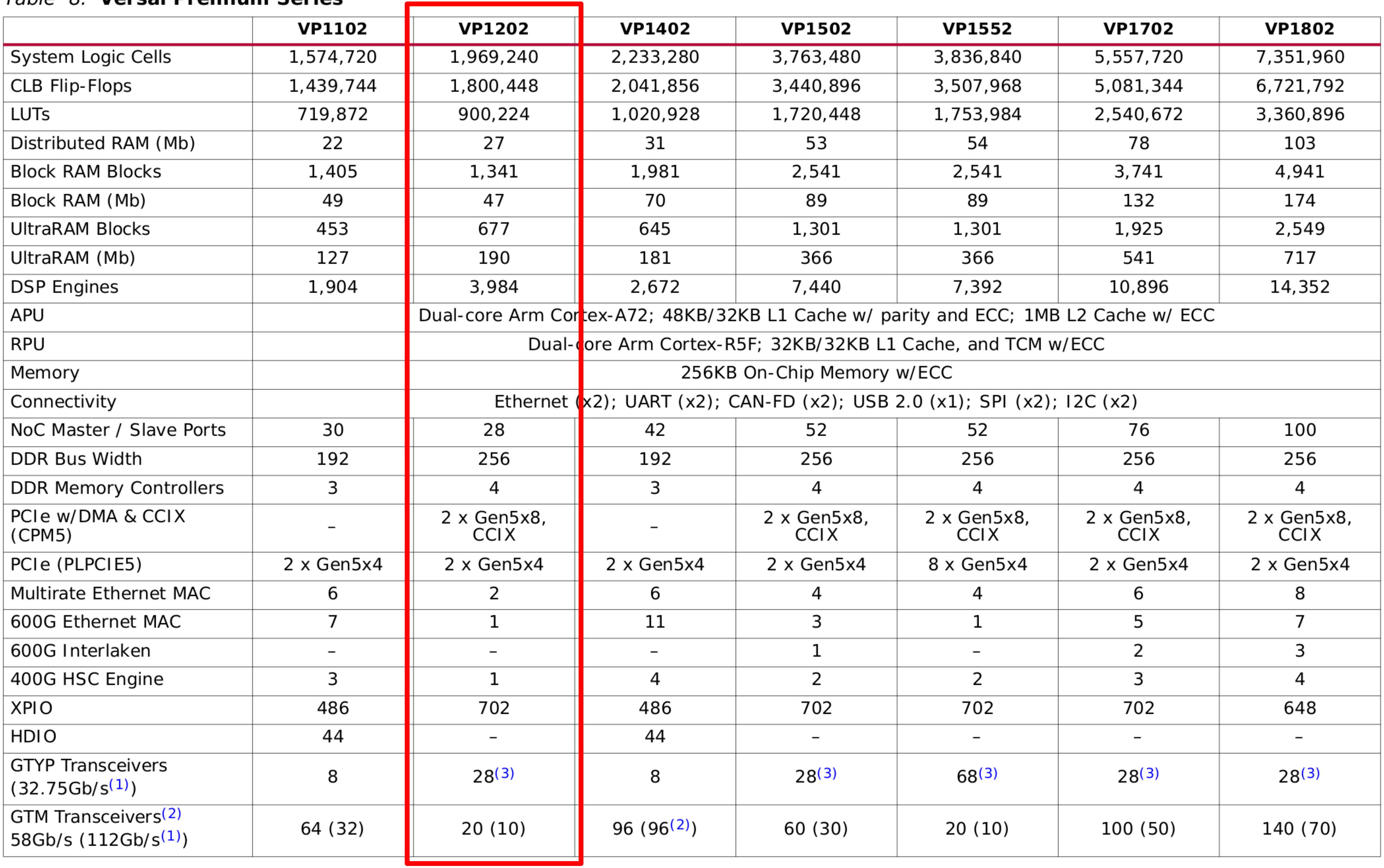
factor ∼30 more DSP slices

factor ∼40 more LUTs

XILINX novel Versal architecture

- Adaptive Compute Acceleration Platform (ACAP):
  - a multicore scalar processing system (PS), a dual-core Arm Cortex-A72 (APU) and a dual-core Arm Cortex-R5F (RPU). RPU is a “realtime” CPU.
  - A programmable logic (PL), made up of configurable blocks (as before in e.g. Virtex or Kintex), but there are adaptable engines, e.g.
    – SIMD VLIW AI engine accelerators
    (single instruction multiple data, very large instruction word)
    – DSP engines, 27 × 24 bit multiplier and a 58-bit accumulator,
    each 32-bit floating point
  - Massive data I/O e.g. 600G ethernet

Table 8: **Versal Premium Series**



	VP1102	VP1202	VP1402	VP1502	VP1552	VP1702	VP1802
System Logic Cells	1,574,720	1,969,240	2,233,280	3,763,480	3,836,840	5,557,720	7,351,960
CLB Flip-Flops	1,439,744	1,800,448	2,041,856	3,440,896	3,507,968	5,081,344	6,721,792
LUTs	719,872	900,224	1,020,928	1,720,448	1,753,984	2,540,672	3,360,896
Distributed RAM (Mb)	22	27	31	53	54	78	103
Block RAM Blocks	1,405	1,341	1,981	2,541	2,541	3,741	4,941
Block RAM (Mb)	49	47	70	89	89	132	174
UltraRAM Blocks	453	677	645	1,301	1,301	1,925	2,549
UltraRAM (Mb)	127	190	181	366	366	541	717
DSP Engines	1,904	3,984	2,672	7,440	7,392	10,896	14,352
APU		Dual-core Arm Cortex-A72; 48KB/32KB L1 Cache w/ parity and ECC; 1MB L2 Cache w/ ECC					
RPU		Dual-core Arm Cortex-R5F; 32KB/32KB L1 Cache, and TCM w/ECC					
Memory		256KB On-Chip Memory w/ECC					
Connectivity		Ethernet (x2); UART (x2); CAN-FD (x2); USB 2.0 (x1); SPI (x2); I2C (x2)					
NoC Master / Slave Ports	30	28	42	52	52	76	100
DDR Bus Width	192	256	192	256	256	256	256
DDR Memory Controllers	3	4	3	4	4	4	4
PCIe w/DMA & CCIX (CPM5)	–	2 x Gen5x8, CCIX	–	2 x Gen5x8, CCIX	2 x Gen5x8, CCIX	2 x Gen5x8, CCIX	2 x Gen5x8, CCIX
PCIe (PLPCIE5)	2 x Gen5x4	2 x Gen5x4	2 x Gen5x4	2 x Gen5x4	8 x Gen5x4	2 x Gen5x4	2 x Gen5x4
Multirate Ethernet MAC	6	2	6	4	4	6	8
600G Ethernet MAC	7	1	11	3	1	5	7
600G Interlaken	–	–	–	1	–	2	3
400G HSC Engine	3	1	4	2	2	3	4
XPIO	486	702	486	702	702	702	648
HDIO	44	–	44	–	–	–	–
GTYP Transceivers (32.75Gb/s[1])	8	28[3]	8	28[3]	68[3]	28[3]	28[3]
GTM Transceivers[2] 58Gb/s (112Gb/s[1])	64 (32)	20 (10)	96 (96[2])	60 (30)	20 (10)	100 (50)	140 (70)

Programming our Versal board, first steps

- Using VITIS 2023.2 (as newest as possible)
- Side remark: we used the different Xilinx tools beforehand
  - planAhead 14.7 (frozen) for Virtex (see my talk on 09.04.)
  - VIVADO for Kintex (new carrier board, see talk of Matthäus Krein on 09.04.)
- Installed also the two update packages, containing a lot of new Versal support
- Our board is AMD XILINX EK-VPK120-G Evaluation Kit
- FPGA is a Versal Premium 1202 (XCVP1202–2MSEVSVA2785)
- Versal has different chip series: Versal AI Core series has an array of signal processing cores that are highly optimized for functions in ML.
  The array consists of up to 400 AI engines, each comprising a 32-bit scalar RISC processor, fixed and floating point vector units.
- However, Versal premium has no "AI engines".
  It is better described as a hardware accelerator using e.g. "DSP engines"

Programming our Versal board, first steps

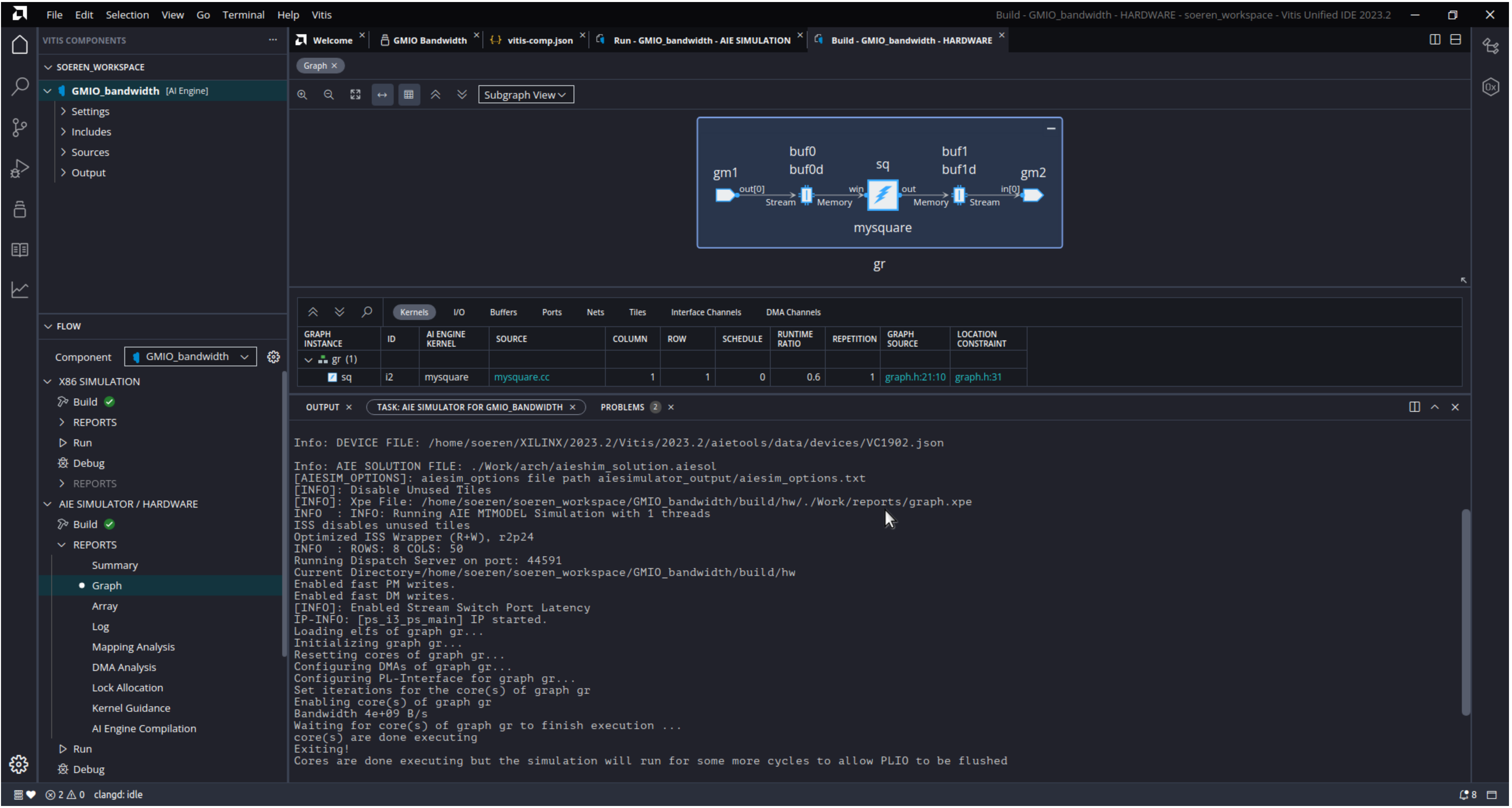
- Our VITIS has Prime 1202 support, however there is no VPK120 platform (which is the first thing you would have to select in a project)
- I decided for now to start development using VCK190 platform (needs to be adapted later) and running the “AI Engine Emulator” (but not the “X86 Simulator”)
- VITIS download and installation takes up to ~60 hours
- Installation gets stuck in Ubuntu 22.04,
  endless loop in "Generating Device List" (no error message)
  solution:

  ```
  sudo apt-get install libtinfo5
  sudo apt-get install libncurses5
  ```
- There are three flavors of VITIS:
  1. VITIS “unified” (black GUI, modern style)
  2. VITIS “classic” (grey GUI, reminds me of Microsoft Windows GUI)
  3. VITIS “console”

1. Test of VITIS unified

- VITIS unified has example projects, but there is no matrix multiplication
- Other examples, e.g. GMIO_bandwidth example project works out of the box (GMIO connects AI Engines and the logical global memory ports of a hardware platform design)

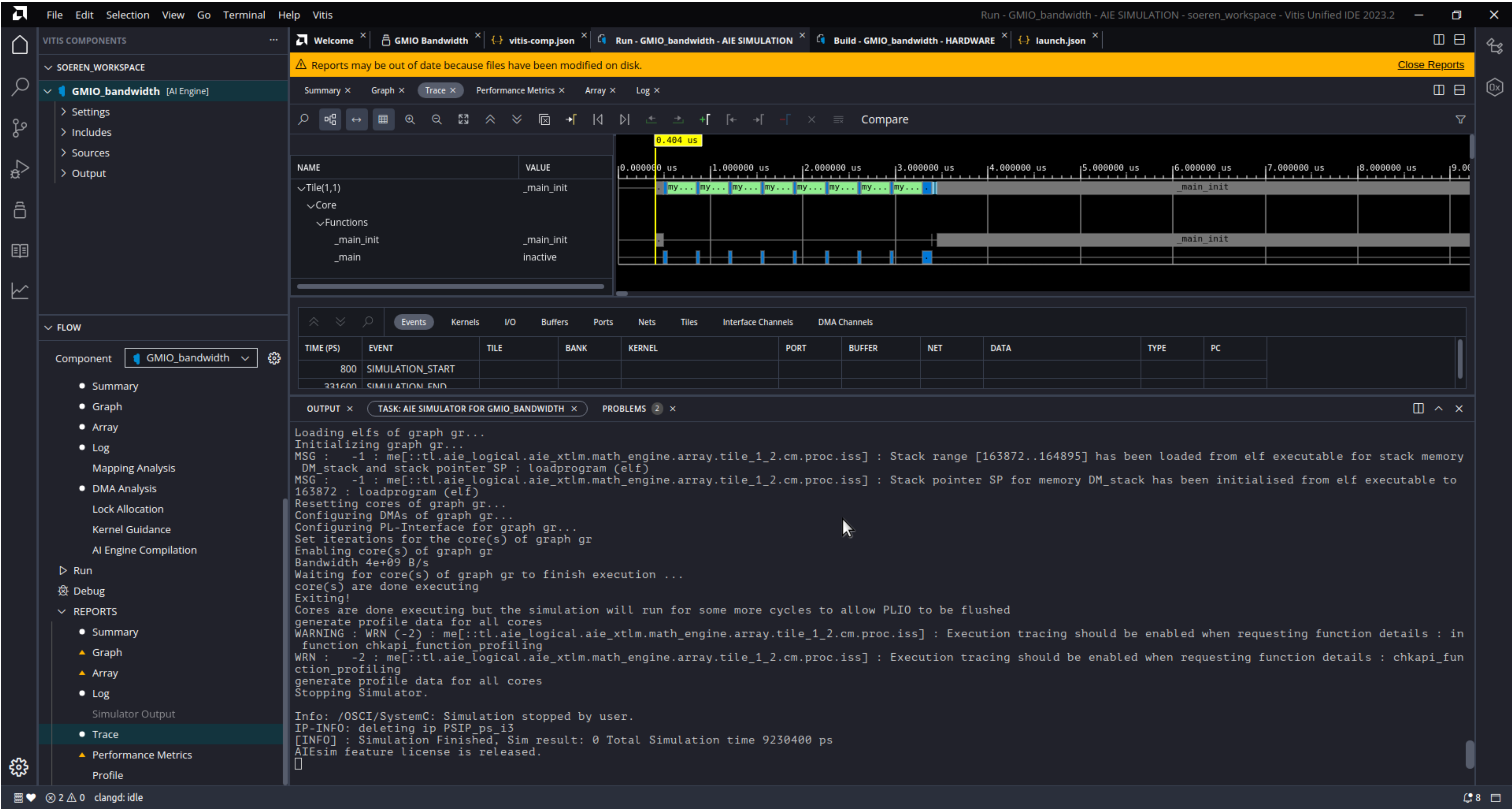
1. Test of VITIS unified



GRAPH INSTANCE	ID	AI ENGINE KERNEL	SOURCE	COLUMN	ROW	SCHEDULE	RUNTIME RATIO	REPETITION	GRAPH SOURCE	LOCATION CONSTRAINT
gr (1)										
sq	i2	mysquare	mysquare.cc	1	1	0	0.6	1	graph.h:21:10	graph.h:31

```
Info: DEVICE FILE: /home/soeren/XILINX/2023.2/Vitis/2023.2/aietools/data/devices/VC1902.json

Info: AIE SOLUTION FILE: ./Work/arch/aieshim_solution.aiesol
[AIESIM_OPTIONS]: aiesim_options file path aiesimulator_output/aiesim_options.txt
[INFO]: Disable Unused Tiles
[INFO]: Xpe File: /home/soeren/soeren_workspace/GMIO_bandwidth/build/hw/./Work/reports/graph.xpe
INFO  : INFO: Running AIE MTMODEL Simulation with 1 threads
ISS disables unused tiles
Optimized ISS Wrapper (R+W), r2p24
INFO  : ROWS: 8 COLS: 50
Running Dispatch Server on port: 44591
Current Directory=/home/soeren/soeren_workspace/GMIO_bandwidth/build/hw
Enabled fast PM writes.
Enabled fast DM writes.
[INFO]: Enabled Stream Switch Port Latency
IP-INFO: [ps_i3_ps_main] IP started.
Loading elfs of graph gr...
Initializing graph gr...
Resetting cores of graph gr...
Configuring DMAs of graph gr...
Configuring PL-Interface for graph gr...
Set iterations for the core(s) of graph gr
Enabling core(s) of graph gr
Bandwidth 4e+09 B/s
Waiting for core(s) of graph gr to finish execution ...
core(s) are done executing
Exiting!
Cores are done executing but the simulation will run for some more cycles to allow PLIO to be flushed
```

1. Test of VITIS unified

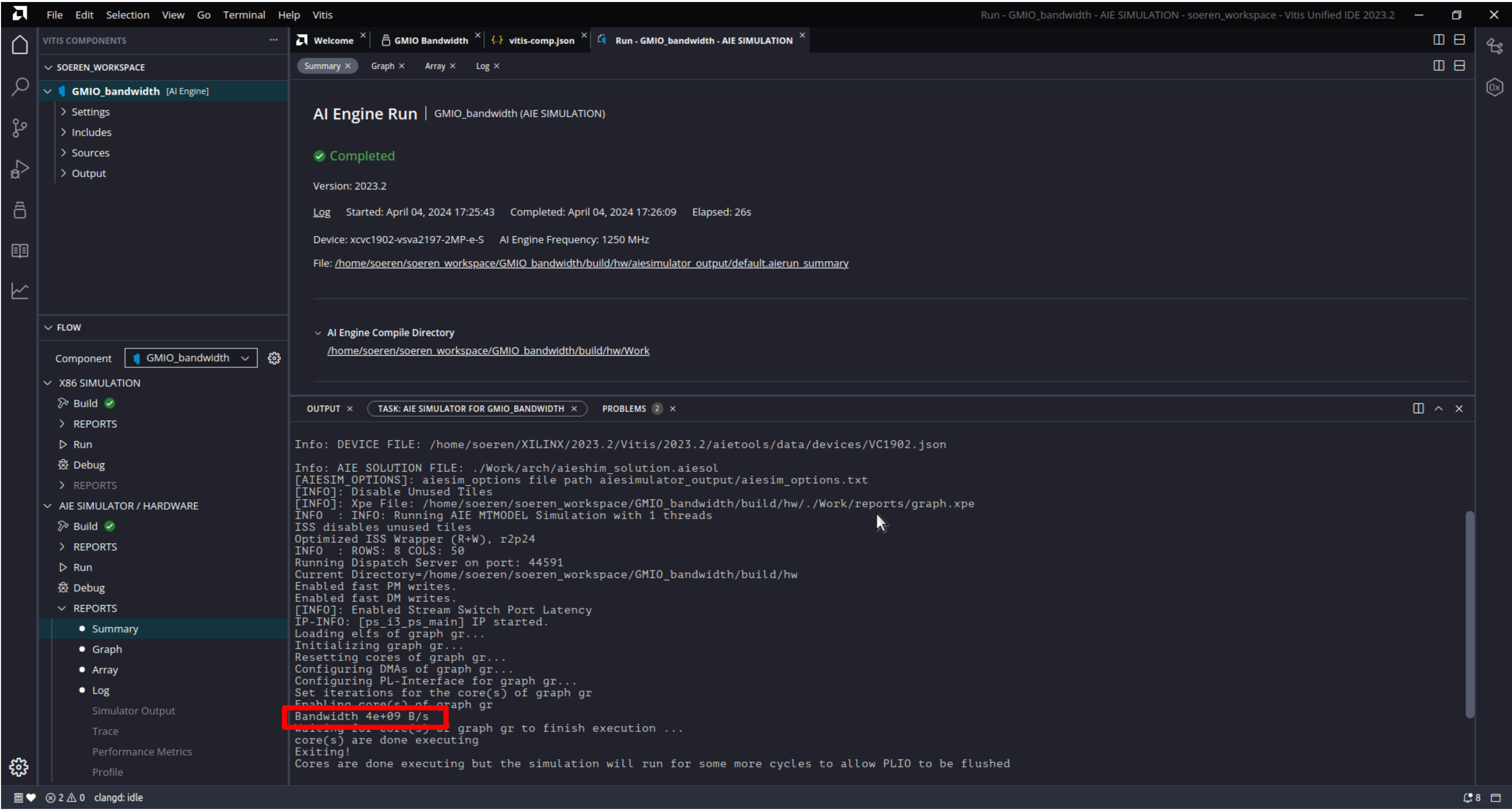


NAME	VALUE
Tile(1,1)	_main_init
Core	
Functions	
_main_init	_main_init
_main	inactive

TIME (PS)	EVENT	TILE	BANK	KERNEL	PORT	BUFFER	NET	DATA	TYPE	PC
800	SIMULATION_START									
331600	SIMULATION_END									

```
Loading elfs of graph gr...
Initializing graph gr...
MSG :   -1 : me[::tl.aie_logical.aie_xtlm.math_engine.array.tile_1_2.cm.proc.iss] : Stack range [163872..164895] has been loaded from elf executable for stack memory
 DM_stack and stack pointer SP : loadprogram (elf)
MSG :   -1 : me[::tl.aie_logical.aie_xtlm.math_engine.array.tile_1_2.cm.proc.iss] : Stack pointer SP for memory DM_stack has been initialised from elf executable to
163872 : loadprogram (elf)
Resetting cores of graph gr...
Configuring DMAs of graph gr...
Configuring PL-Interface for graph gr...
Set iterations for the core(s) of graph gr
Enabling core(s) of graph gr
Bandwidth 4e+09 B/s
Waiting for core(s) of graph gr to finish execution ...
core(s) are done executing
Exiting!
Cores are done executing but the simulation will run for some more cycles to allow PLIO to be flushed
generate profile data for all cores
WARNING : WRN (-2) : me[::tl.aie_logical.aie_xtlm.math_engine.array.tile_1_2.cm.proc.iss] : Execution tracing should be enabled when requesting function details : in
 function chkapi_function_profiling
WRN :   -2 : me[::tl.aie_logical.aie_xtlm.math_engine.array.tile_1_2.cm.proc.iss] : Execution tracing should be enabled when requesting function details : chkapi_fun
ction_profiling
generate profile data for all cores
Stopping Simulator.

Info: /OSCI/SystemC: Simulation stopped by user.
IP-INFO: deleting ip PSIP_ps_i3
[INFO] : Simulation Finished, Sim result: 0 Total Simulation time 9230400 ps
AIEsim feature license is released.
```

1. Test of VITIS unified



AI Engine Run | GMIO_bandwidth (AIE SIMULATION)

✓ Completed

Version: 2023.2

Log Started: April 04, 2024 17:25:43 Completed: April 04, 2024 17:26:09 Elapsed: 26s

Device: xcvc1902-vsva2197-2MP-e-S AI Engine Frequency: 1250 MHz

File: /home/soeren/soeren_workspace/GMIO_bandwidth/build/hw/aiesimulator_output/default.aierun_summary

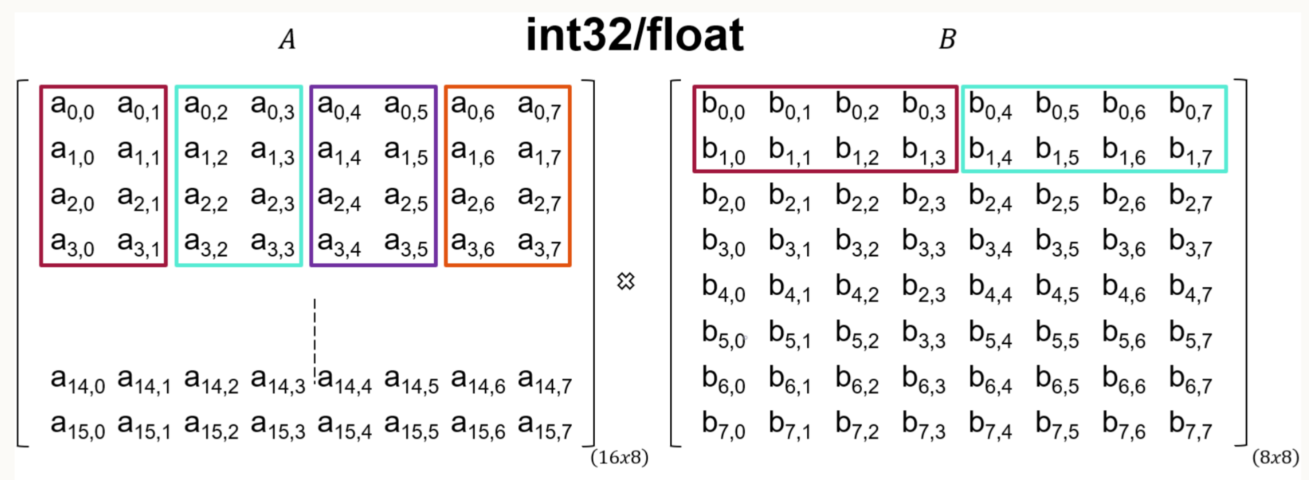
AI Engine Compile Directory

/home/soeren/soeren_workspace/GMIO_bandwidth/build/hw/Work

```
Info: DEVICE FILE: /home/soeren/XILINX/2023.2/Vitis/2023.2/aietools/data/devices/VC1902.json

Info: AIE SOLUTION FILE: ./Work/arch/aieshim_solution.aiesol
[AIESIM_OPTIONS]: aiesim_options file path aiesimulator_output/aiesim_options.txt
[INFO]: Disable Unused Tiles
[INFO]: Xpe File: /home/soeren/soeren_workspace/GMIO_bandwidth/build/hw/./Work/reports/graph.xpe
INFO  : INFO: Running AIE MTMODEL Simulation with 1 threads
ISS disables unused tiles
Optimized ISS Wrapper (R+W), r2p24
INFO  : ROWS: 8 COLS: 50
Running Dispatch Server on port: 44591
Current Directory=/home/soeren/soeren_workspace/GMIO_bandwidth/build/hw
Enabled fast PM writes.
Enabled fast DM writes.
[INFO]: Enabled Stream Switch Port Latency
IP-INFO: [ps_i3_ps_main] IP started.
Loading elfs of graph gr...
Initializing graph gr...
Resetting cores of graph gr...
Configuring DMAs of graph gr...
Configuring PL-Interface for graph gr...
Set iterations for the core(s) of graph gr
Enabling core(s) of graph gr
Bandwidth 4e+09 B/s
Waiting for core(s) of graph gr to finish execution ...
core(s) are done executing
Exiting!
Cores are done executing but the simulation will run for some more cycles to allow PLIO to be flushed
```

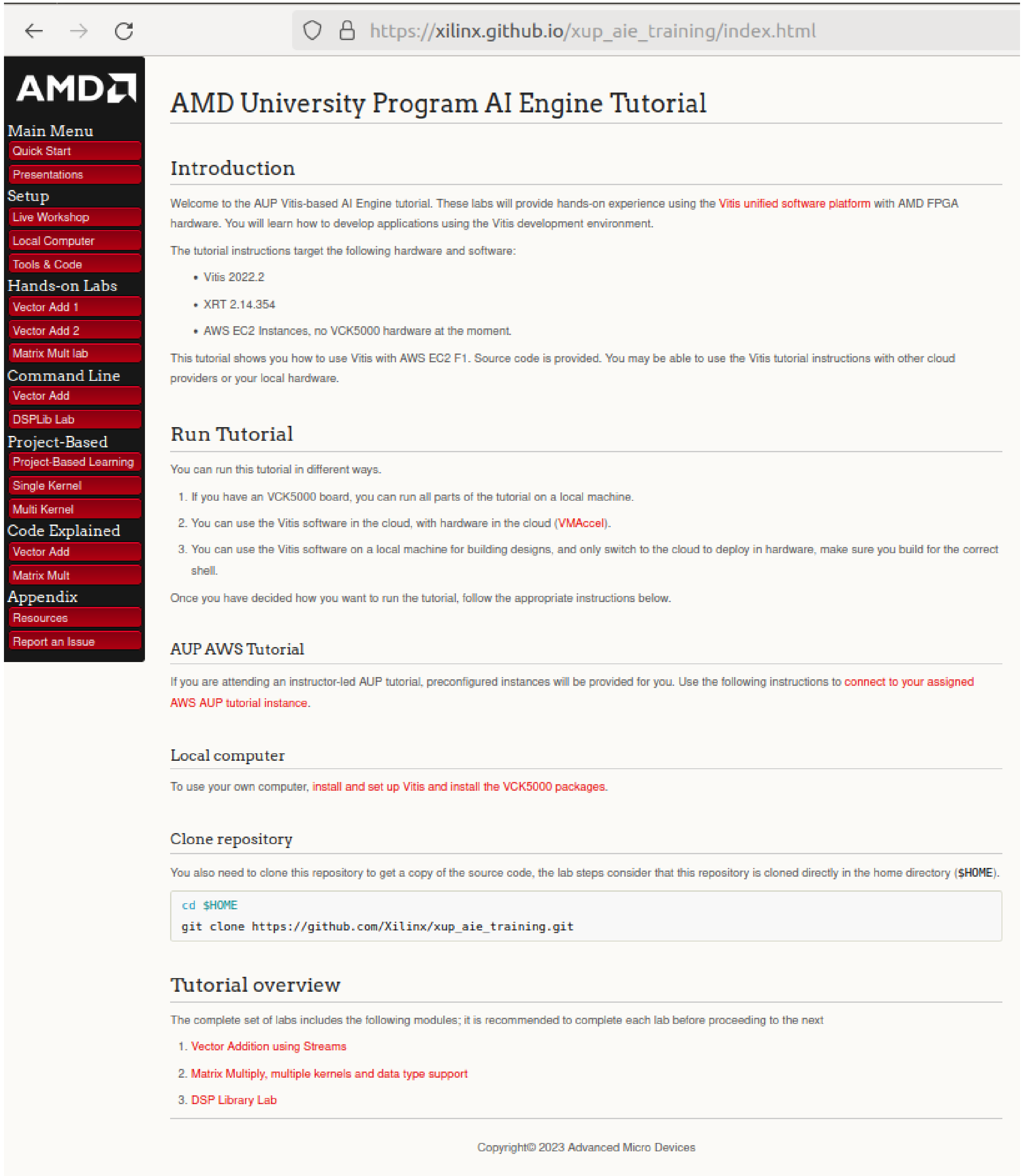
2. Test of VITIS classic

- AMD University Program AI Engine Tutorial https://xilinx.github.io/xup_aie_training/index.html
- Matrix multiplication (16 x 8) x (8 x 8)
- Integer and floating point
- examples run only in “classic” (opening in VITIS unified gives error message)
- Warning when opening (deprecated), but build project successful
- Run with trace (for result visualisation), but classic vitis_analyzer (classic) crashes sometimes when opening the trace files, but they can be opened with vitis_analyzer (unified)



$$A = \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} & a_{0,4} & a_{0,5} & a_{0,6} & a_{0,7} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} & a_{1,6} & a_{1,7} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} & a_{2,6} & a_{2,7} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} & a_{3,6} & a_{3,7} \\ & & & \vdots & & & & \\ a_{14,0} & a_{14,1} & a_{14,2} & a_{14,3} & a_{14,4} & a_{14,5} & a_{14,6} & a_{14,7} \\ a_{15,0} & a_{15,1} & a_{15,2} & a_{15,3} & a_{15,4} & a_{15,5} & a_{15,6} & a_{15,7} \end{bmatrix}_{(16x8)} \otimes B = \begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} & b_{0,4} & b_{0,5} & b_{0,6} & b_{0,7} \\ b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} & b_{1,4} & b_{1,5} & b_{1,6} & b_{1,7} \\ b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} & b_{2,4} & b_{2,5} & b_{2,6} & b_{2,7} \\ b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} & b_{3,4} & b_{3,5} & b_{3,6} & b_{3,7} \\ b_{4,0} & b_{4,1} & b_{4,2} & b_{2,3} & b_{4,4} & b_{4,5} & b_{4,6} & b_{4,7} \\ b_{5,0} & b_{5,1} & b_{5,2} & b_{3,3} & b_{5,4} & b_{5,5} & b_{5,6} & b_{5,7} \\ b_{6,0} & b_{6,1} & b_{6,2} & b_{6,3} & b_{6,4} & b_{6,5} & b_{6,6} & b_{6,7} \\ b_{7,0} & b_{7,1} & b_{7,2} & b_{7,3} & b_{7,4} & b_{7,5} & b_{7,6} & b_{7,7} \end{bmatrix}_{(8x8)}$$

2. Test of VITIS classic



AMD University Program AI Engine Tutorial

Introduction

Welcome to the AUP Vitis-based AI Engine tutorial. These labs will provide hands-on experience using the Vitis unified software platform with AMD FPGA hardware. You will learn how to develop applications using the Vitis development environment.

The tutorial instructions target the following hardware and software:

- Vitis 2022.2
- XRT 2.14.354
- AWS EC2 Instances, no VCK5000 hardware at the moment.

This tutorial shows you how to use Vitis with AWS EC2 F1. Source code is provided. You may be able to use the Vitis tutorial instructions with other cloud providers or your local hardware.

Run Tutorial

You can run this tutorial in different ways.

1. If you have an VCK5000 board, you can run all parts of the tutorial on a local machine.
2. You can use the Vitis software in the cloud, with hardware in the cloud (VMAccel).
3. You can use the Vitis software on a local machine for building designs, and only switch to the cloud to deploy in hardware, make sure you build for the correct shell.

Once you have decided how you want to run the tutorial, follow the appropriate instructions below.

AUP AWS Tutorial

If you are attending an instructor-led AUP tutorial, preconfigured instances will be provided for you. Use the following instructions to connect to your assigned AWS AUP tutorial instance.

Local computer

To use your own computer, install and set up Vitis and install the VCK5000 packages.

Clone repository

You also need to clone this repository to get a copy of the source code, the lab steps consider that this repository is cloned directly in the home directory (`$HOME`).

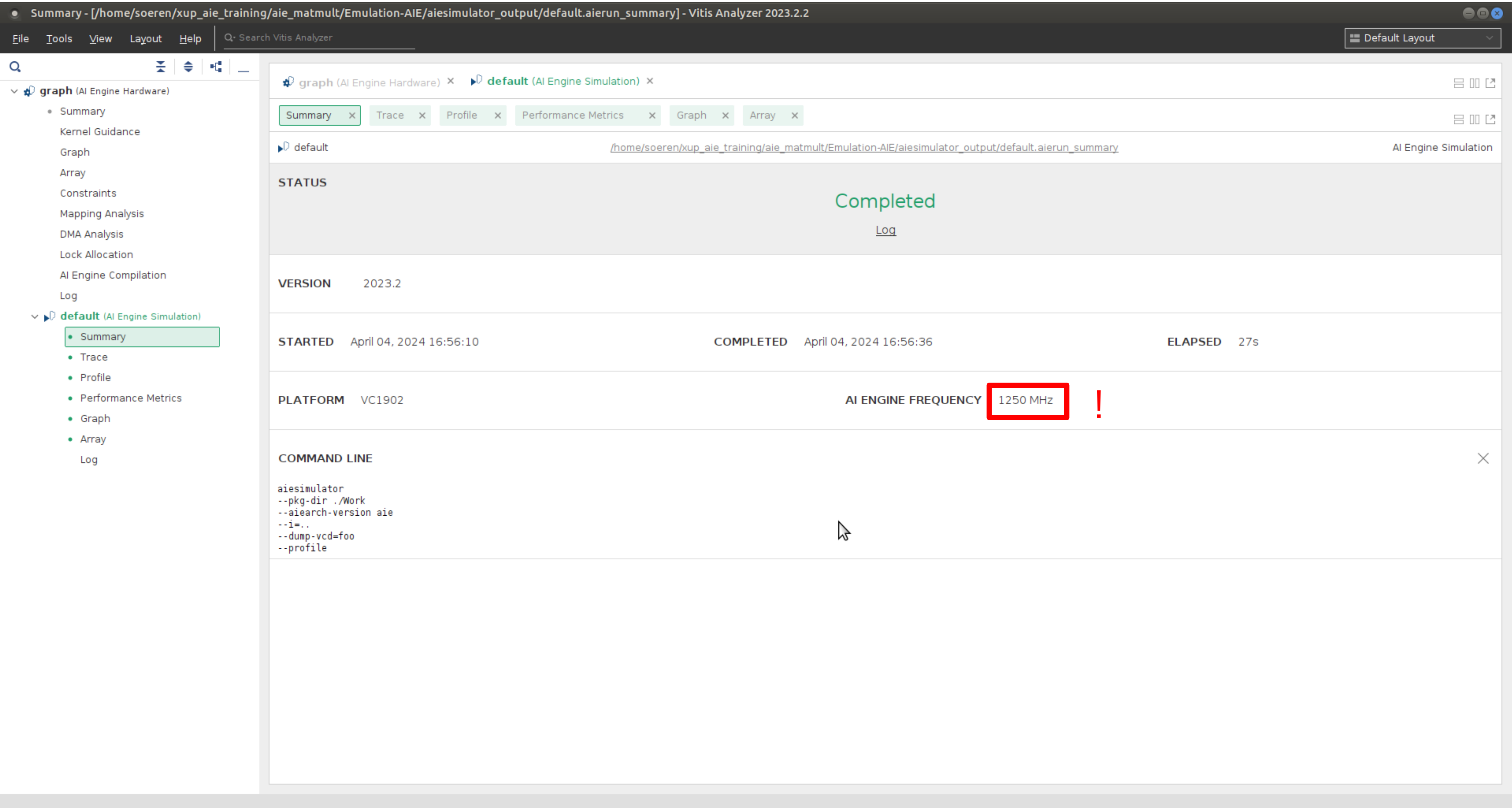
```
cd $HOME
git clone https://github.com/Xilinx/xup_aie_training.git
```

Tutorial overview

The complete set of labs includes the following modules; it is recommended to complete each lab before proceeding to the next

1. Vector Addition using Streams
2. Matrix Multiply, multiple kernels and data type support
3. DSP Library Lab

2. Test of VITIS classic



Summary - [/home/soeren/xup_aie_training/aie_matmult/Emulation-AIE/aiesimulator_output/default.aierun_summary] - Vitis Analyzer 2023.2.2
File Tools View Layout Help
Search Vitis Analyzer
Default Layout
graph (AI Engine Hardware)
Summary
Kernel Guidance
Graph
Array
Constraints
Mapping Analysis
DMA Analysis
Lock Allocation
AI Engine Compilation
Log
default (AI Engine Simulation)
Summary
Trace
Profile
Performance Metrics
Graph
Array
Log
graph (AI Engine Hardware)
default (AI Engine Simulation)
Summary
Trace
Profile
Performance Metrics
Graph
Array

default /home/soeren/xup_aie_training/aie_matmult/Emulation-AIE/aiesimulator_output/default.aierun_summary AI Engine Simulation

STATUS

Completed

Log

VERSION 2023.2

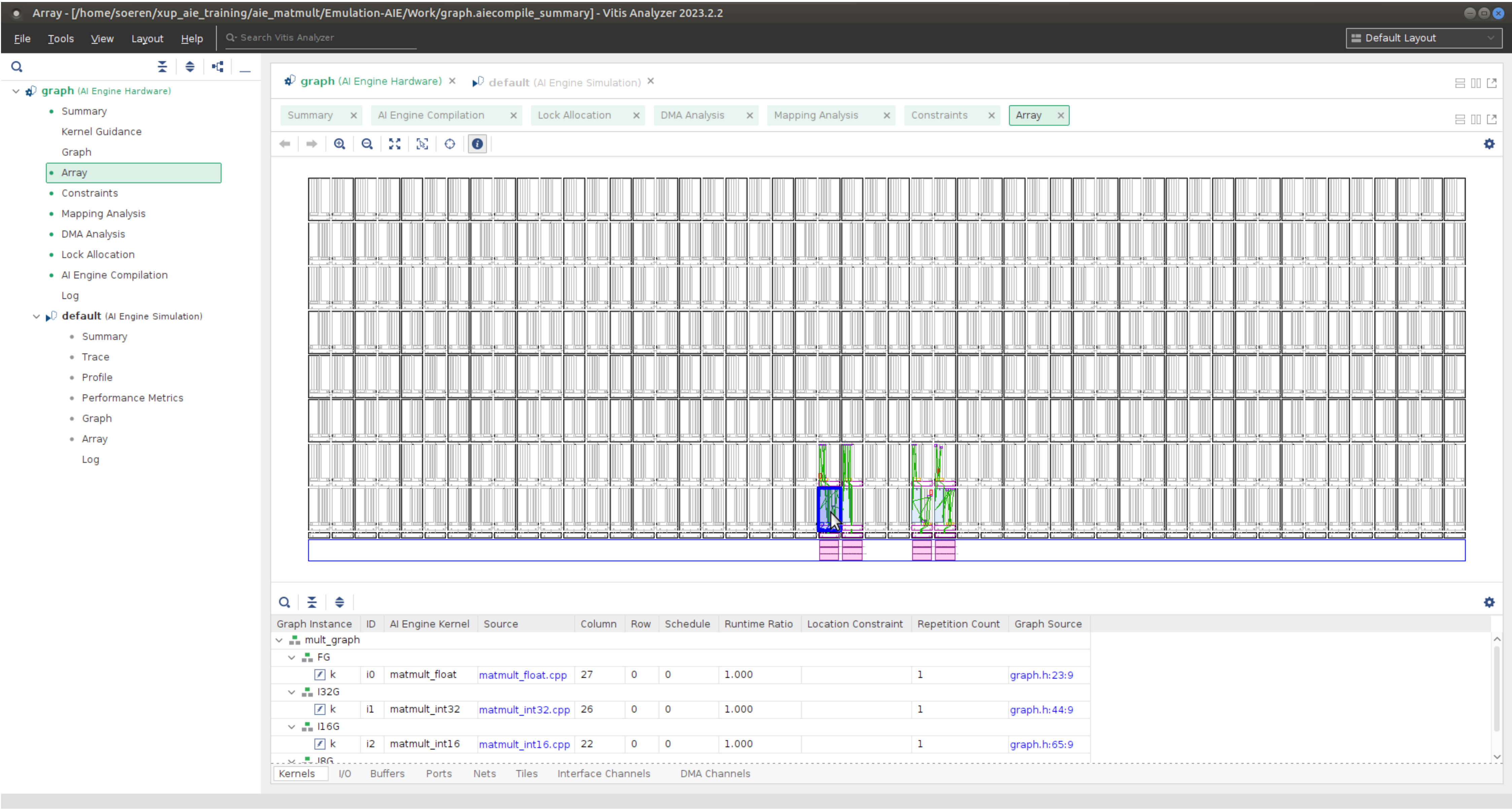
STARTED April 04, 2024 16:56:10 COMPLETED April 04, 2024 16:56:36 ELAPSED 27s

PLATFORM VC1902 AI ENGINE FREQUENCY 1250 MHz !

COMMAND LINE

```
aiesimulator
--pkg-dir ./Work
--aiearch-version aie
--i=..
--dump-vcd=foo
--profile
```

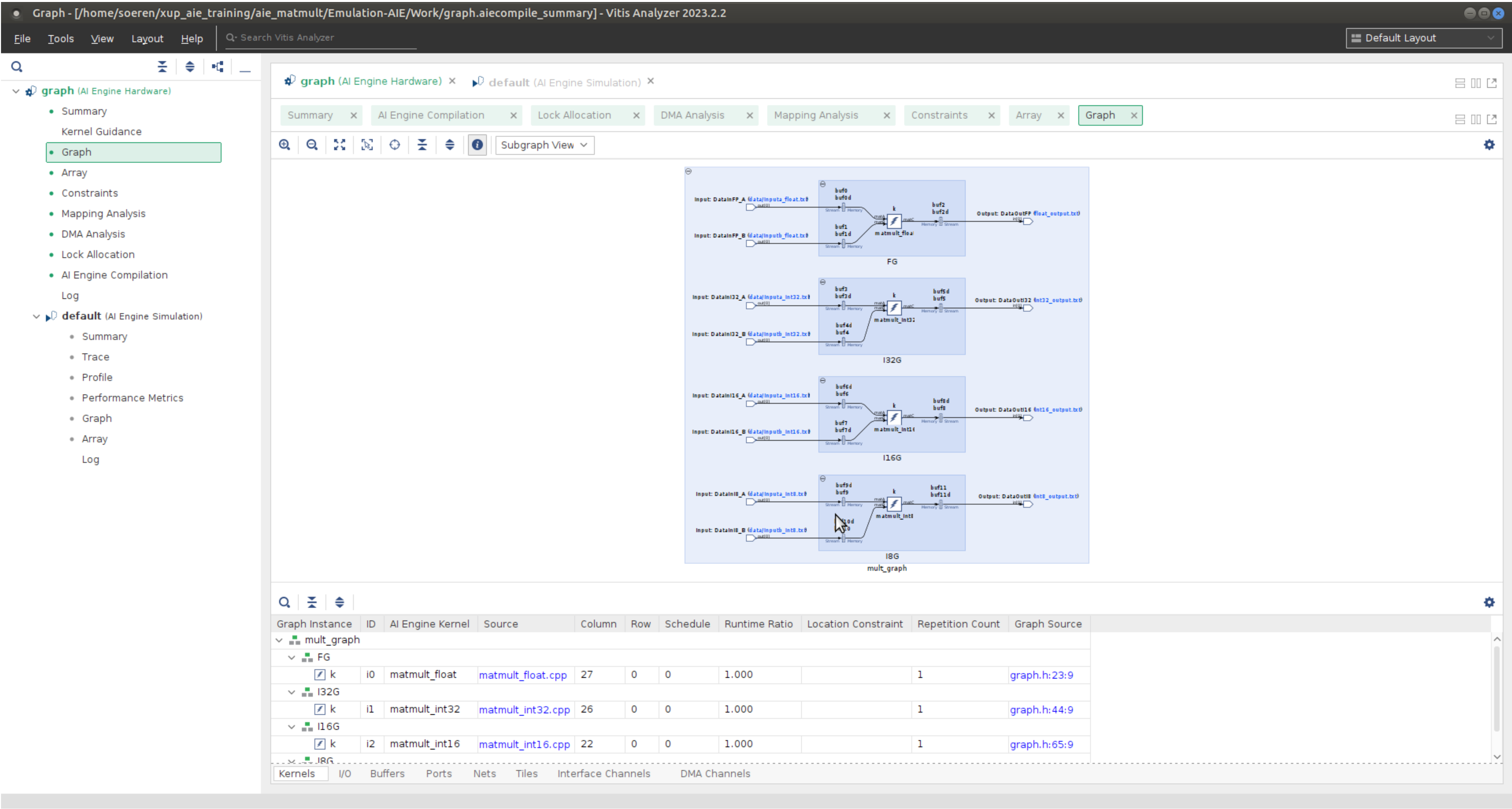
2. Test of VITIS classic



Array - [/home/soeren/xup_aie_training/aie_matmult/Emulation-AIE/Work/graph.aiecompile_summary] - Vitis Analyzer 2023.2.2

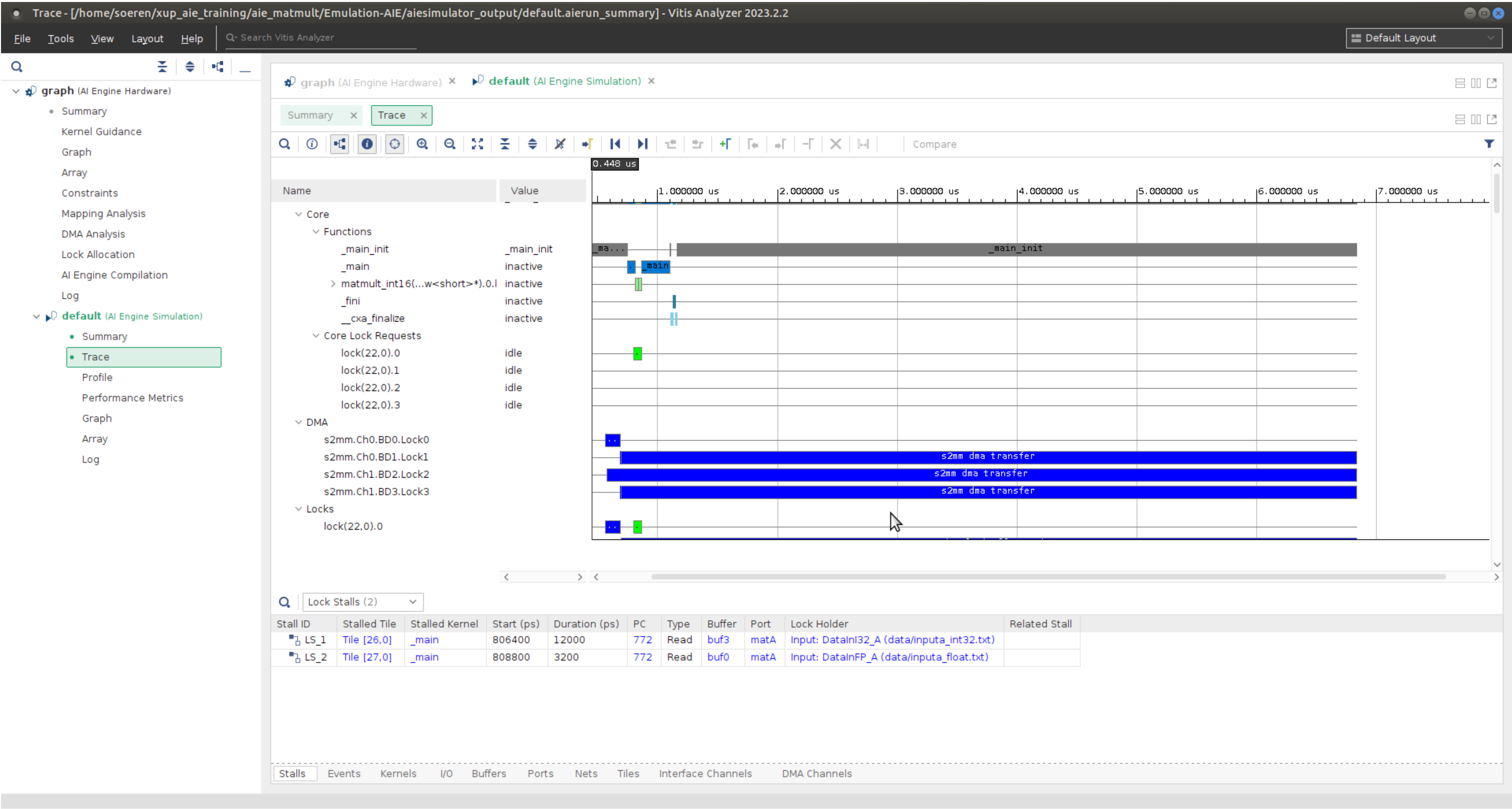
Graph Instance	ID	AI Engine Kernel	Source	Column	Row	Schedule	Runtime Ratio	Location Constraint	Repetition Count	Graph Source
mult_graph										
FG										
k	i0	matmult_float	matmult_float.cpp	27	0	0	1.000		1	graph.h:23:9
I32G										
k	i1	matmult_int32	matmult_int32.cpp	26	0	0	1.000		1	graph.h:44:9
I16G										
k	i2	matmult_int16	matmult_int16.cpp	22	0	0	1.000		1	graph.h:65:9
I8G										

2. Test of VITIS classic


Graph - [/home/soeren/xup_aie_training/aie_matmult/Emulation-AIE/Work/graph.aiecompile_summary] - Vitis Analyzer 2023.2.2
File
Tools
View
Layout
Help
Search Vitis Analyzer
Default Layout
graph (AI Engine Hardware)
Summary
Kernel Guidance
Graph
Array
Constraints
Mapping Analysis
DMA Analysis
Lock Allocation
AI Engine Compilation
Log
default (AI Engine Simulation)
Summary
Trace
Profile
Performance Metrics
Graph
Array
Log
graph (AI Engine Hardware)
default (AI Engine Simulation)
Summary
AI Engine Compilation
Lock Allocation
DMA Analysis
Mapping Analysis
Constraints
Array
Graph
Subgraph View
Input: DataInFP_A (data/inputa_float.txt)
Input: DataInFP_B (data/inputb_float.txt)
buf0
buf0d
buf1
buf1d
k
matmult_float
buf2
buf2d
Output: DataOutFP (float_output.txt)
FG
Input: DataInI32_A (data/inputa_int32.txt)
Input: DataInI32_B (data/inputb_int32.txt)
buf3
buf3d
buf4d
buf4
k
matmult_int32
buf5d
buf5
Output: DataOutI32 (int32_output.txt)
I32G
Input: DataInI16_A (data/inputa_int16.txt)
Input: DataInI16_B (data/inputb_int16.txt)
buf6d
buf6
buf7
buf7d
k
matmult_int16
buf8d
buf8
Output: DataOutI16 (int16_output.txt)
I16G
Input: DataInI8_A (data/inputa_int8.txt)
Input: DataInI8_B (data/inputb_int8.txt)
buf9d
buf9
k
matmult_int8
buf11
buf11d
Output: DataOutI8 (int8_output.txt)
I8G
mult_graph
Stream
Memory
matA
matB
matC
out[0]
in[0]
Kernels
I/O
Buffers
Ports
Nets
Tiles
Interface Channels
DMA Channels

Graph Instance	ID	AI Engine Kernel	Source	Column	Row	Schedule	Runtime Ratio	Location Constraint	Repetition Count	Graph Source
mult_graph										
FG										
k	i0	matmult_float	matmult_float.cpp	27	0	0	1.000		1	graph.h:23:9
I32G										
k	i1	matmult_int32	matmult_int32.cpp	26	0	0	1.000		1	graph.h:44:9
I16G										
k	i2	matmult_int16	matmult_int16.cpp	22	0	0	1.000		1	graph.h:65:9
I8G										

2. Test of VITIS classic

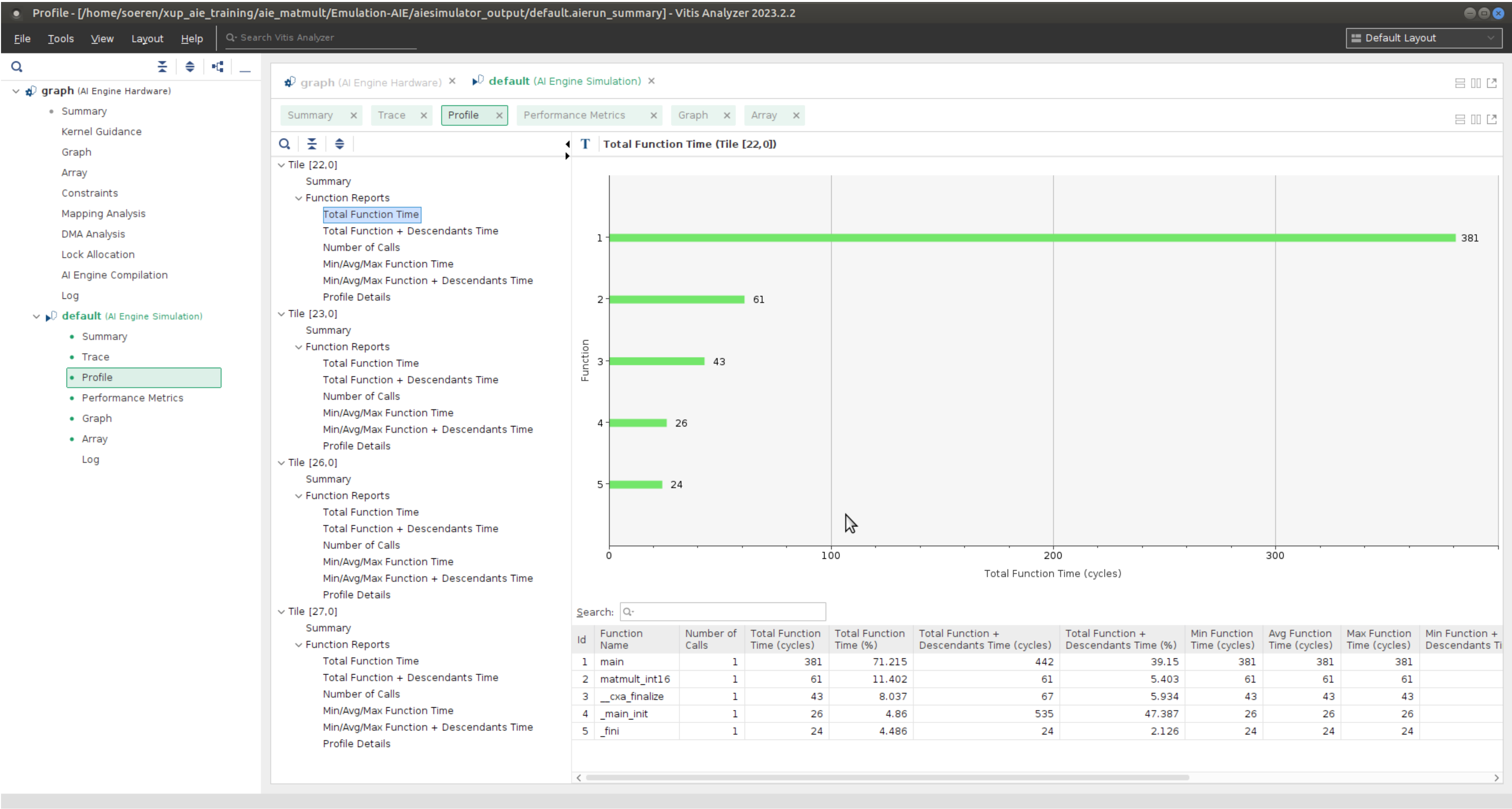


Trace - [/home/soeren/xup_aie_training/aie_matmult/Emulation-AIE/aiesimulator_output/default.aierun_summary] - Vitis Analyzer 2023.2.2

Name	Value
Core	
Functions	
_main_init	_main_init
_main	inactive
matmult_int16(...w<short>*).0.l	inactive
_fini	inactive
__cxa_finalize	inactive
Core Lock Requests	
lock(22,0).0	idle
lock(22,0).1	idle
lock(22,0).2	idle
lock(22,0).3	idle
DMA	
s2mm.Ch0.BD0.Lock0	
s2mm.Ch0.BD1.Lock1	
s2mm.Ch1.BD2.Lock2	
s2mm.Ch1.BD3.Lock3	
Locks	
lock(22,0).0	

Stall ID	Stalled Tile	Stalled Kernel	Start (ps)	Duration (ps)	PC	Type	Buffer	Port	Lock Holder	Related Stall
LS_1	Tile [26,0]	_main	806400	12000	772	Read	buf3	matA	Input: DataInI32_A (data/inputa_int32.txt)	
LS_2	Tile [27,0]	_main	808800	3200	772	Read	buf0	matA	Input: DataInFP_A (data/inputa_float.txt)	

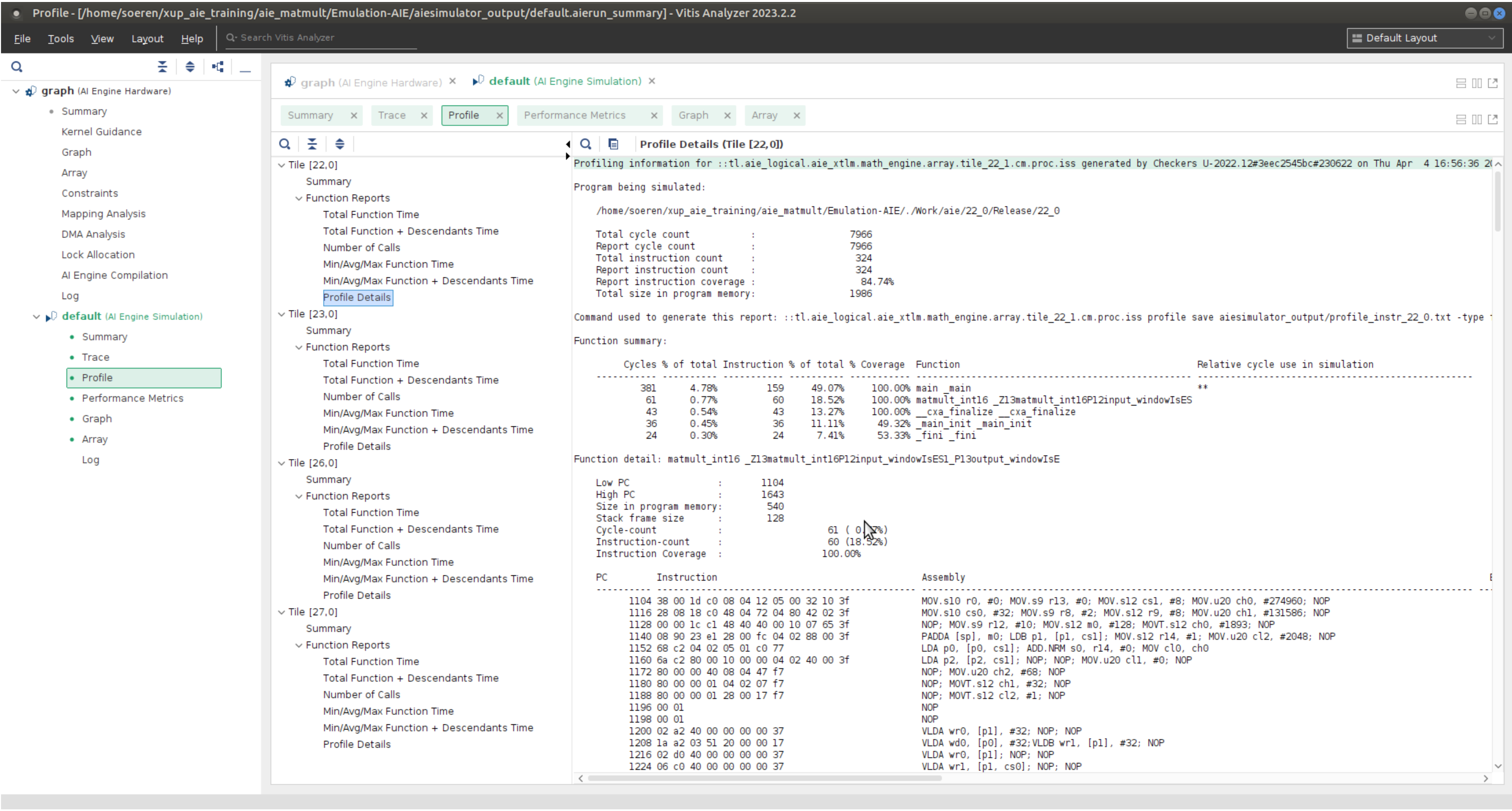
2. Test of VITIS classic



Profile - [/home/soeren/xup_aie_training/aie_matmult/Emulation-AIE/aiesimulator_output/default.aierun_summary] - Vitis Analyzer 2023.2.2

Id	Function Name	Number of Calls	Total Function Time (cycles)	Total Function Time (%)	Total Function + Descendants Time (cycles)	Total Function + Descendants Time (%)	Min Function Time (cycles)	Avg Function Time (cycles)	Max Function Time (cycles)	Min Function + Descendants Ti
1	main	1	381	71.215	442	39.15	381	381	381	
2	matmult_int16	1	61	11.402	61	5.403	61	61	61	
3	__cxa_finalize	1	43	8.037	67	5.934	43	43	43	
4	_main_init	1	26	4.86	535	47.387	26	26	26	
5	_fini	1	24	4.486	24	2.126	24	24	24	

2. Test of VITIS classic

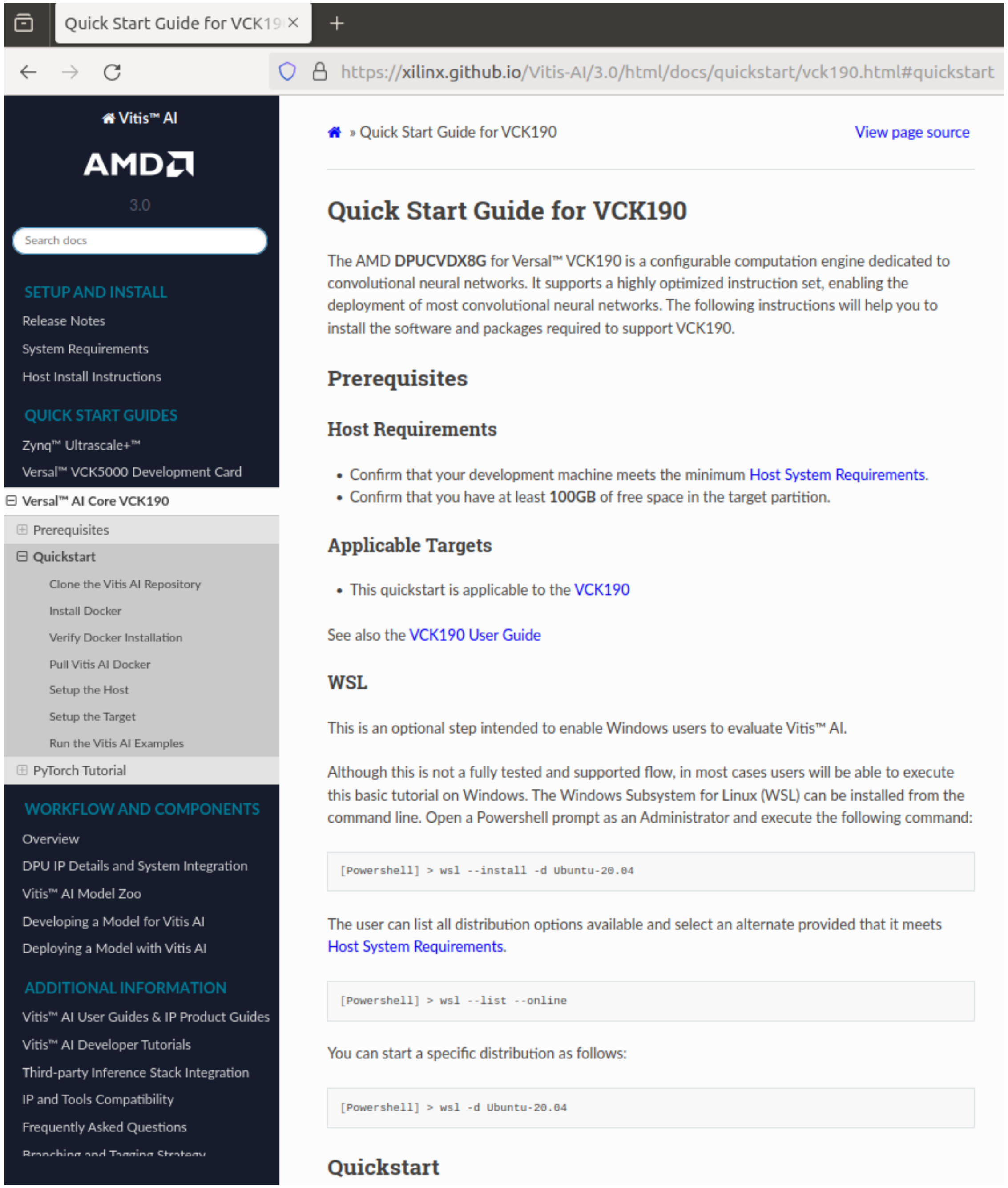

Profile - [/home/soeren/xup_aie_training/aie_matmult/Emulation-AIE/aiesimulator_output/default.aierun_summary] - Vitis Analyzer 2023.2.2
File
Tools
View
Layout
Help
Search Vitis Analyzer
Default Layout
graph (AI Engine Hardware)
Summary
Kernel Guidance
Graph
Array
Constraints
Mapping Analysis
DMA Analysis
Lock Allocation
AI Engine Compilation
Log
default (AI Engine Simulation)
Summary
Trace
Profile
Performance Metrics
Graph
Array
Log
graph (AI Engine Hardware)
default (AI Engine Simulation)
Summary
Trace
Profile
Performance Metrics
Graph
Array
Tile [22,0]
Summary
Function Reports
Total Function Time
Total Function + Descendants Time
Number of Calls
Min/Avg/Max Function Time
Min/Avg/Max Function + Descendants Time
Profile Details
Tile [23,0]
Summary
Function Reports
Total Function Time
Total Function + Descendants Time
Number of Calls
Min/Avg/Max Function Time
Min/Avg/Max Function + Descendants Time
Profile Details
Tile [26,0]
Summary
Function Reports
Total Function Time
Total Function + Descendants Time
Number of Calls
Min/Avg/Max Function Time
Min/Avg/Max Function + Descendants Time
Profile Details
Tile [27,0]
Summary
Function Reports
Total Function Time
Total Function + Descendants Time
Number of Calls
Min/Avg/Max Function Time
Min/Avg/Max Function + Descendants Time
Profile Details
Profile Details (Tile [22,0])

```
Profiling information for ::tl.aie_logical.aie_xtlm.math_engine.array.tile_22_1.cm.proc.iss generated by Checkers U-2022.12#3eec2545bc#230622 on Thu Apr  4 16:56:36 2(

Program being simulated:

    /home/soeren/xup_aie_training/aie_matmult/Emulation-AIE/./Work/aie/22_0/Release/22_0

    Total cycle count          :                 7966
    Report cycle count         :                 7966
    Total instruction count    :                  324
    Report instruction count   :                  324
    Report instruction coverage :                  84.74%
    Total size in program memory:                1986

Command used to generate this report: ::tl.aie_logical.aie_xtlm.math_engine.array.tile_22_1.cm.proc.iss profile save aiesimulator_output/profile_instr_22_0.txt -type 

Function summary:

         Cycles % of total Instruction % of total % Coverage  Function                                              Relative cycle use in simulation
    ----------- ---------- ----------- ---------- ----------- ----------------------------------------------------- ----------------------------------------------------
            381      4.78%         159     49.07%     100.00% main _main                                            **
             61      0.77%          60     18.52%     100.00% matmult_int16 _Z13matmult_int16P12input_windowIsES
             43      0.54%          43     13.27%     100.00% __cxa_finalize __cxa_finalize
             36      0.45%          36     11.11%      49.32% _main_init _main_init
             24      0.30%          24      7.41%      53.33% _fini _fini

Function detail: matmult_int16 _Z13matmult_int16P12input_windowIsES1_P13output_windowIsE

    Low PC                  :        1104
    High PC                 :        1643
    Size in program memory:          540
    Stack frame size        :        128
    Cycle-count             :                  61 ( 0.77%)
    Instruction-count       :                  60 (18.52%)
    Instruction Coverage    :                100.00%

    PC         Instruction                                       Assembly                                                                                        E
    ---------- ------------------------------------------------- ---------------------------------------------------------------------------------------------- ---
          1104 38 00 1d c0 08 04 12 05 00 32 10 3f              MOV.s10 r0, #0; MOV.s9 r13, #0; MOV.s12 cs1, #8; MOV.u20 ch0, #274960; NOP
          1116 28 08 18 c0 48 04 72 04 80 42 02 3f              MOV.s10 cs0, #32; MOV.s9 r8, #2; MOV.s12 r9, #8; MOV.u20 ch1, #131586; NOP
          1128 00 00 1c c1 48 40 40 00 10 07 65 3f              NOP; MOV.s9 r12, #10; MOV.s12 m0, #128; MOVT.s12 ch0, #1893; NOP
          1140 08 90 23 e1 28 00 fc 04 02 88 00 3f              PADDA [sp], m0; LDB p1, [p1, cs1]; MOV.s12 r14, #1; MOV.u20 cl2, #2048; NOP
          1152 68 c2 04 02 05 01 c0 77                          LDA p0, [p0, cs1]; ADD.NRM s0, r14, #0; MOV cl0, ch0
          1160 6a c2 80 00 10 00 00 04 02 40 00 3f              LDA p2, [p2, cs1]; NOP; NOP; MOV.u20 cl1, #0; NOP
          1172 80 00 00 40 08 04 47 f7                          NOP; MOV.u20 ch2, #68; NOP
          1180 80 00 00 01 04 02 07 f7                          NOP; MOVT.s12 ch1, #32; NOP
          1188 80 00 00 01 28 00 17 f7                          NOP; MOVT.s12 cl2, #1; NOP
          1196 00 01                                            NOP
          1198 00 01                                            NOP
          1200 02 a2 40 00 00 00 00 37                          VLDA wr0, [p1], #32; NOP; NOP
          1208 1a a2 03 51 20 00 00 17                          VLDA wd0, [p0], #32;VLDB wr1, [p1], #32; NOP
          1216 02 d0 40 00 00 00 00 37                          VLDA wr0, [p1]; NOP; NOP
          1224 06 c0 40 00 00 00 00 37                          VLDA wr1, [p1, cs0]; NOP; NOP
```

3. Test of VITIS console

- Runs ai-compiler in terminal
- Example: importing an PyTorch NN
- Requires Docker; many problems in Ubuntu installation
- RESNET50: CNN with 50 layers and 3x3 filter
- Task is image processing;
  AMD provides example images for download with wget
- I was able to reach the final point of hardware deployment
  (but, as mentioned, my hardware platform is incorrect)

3. Test of VITIS console


Quick Start Guide for VCK19
https://xilinx.github.io/Vitis-AI/3.0/html/docs/quickstart/vck190.html#quickstart
Vitis™ AI
AMD
3.0
Search docs
SETUP AND INSTALL
Release Notes
System Requirements
Host Install Instructions
QUICK START GUIDES
Zynq™ Ultrascale+™
Versal™ VCK5000 Development Card
Versal™ AI Core VCK190
Prerequisites
Quickstart
Clone the Vitis AI Repository
Install Docker
Verify Docker Installation
Pull Vitis AI Docker
Setup the Host
Setup the Target
Run the Vitis AI Examples
PyTorch Tutorial
WORKFLOW AND COMPONENTS
Overview
DPU IP Details and System Integration
Vitis™ AI Model Zoo
Developing a Model for Vitis AI
Deploying a Model with Vitis AI
ADDITIONAL INFORMATION
Vitis™ AI User Guides & IP Product Guides
Vitis™ AI Developer Tutorials
Third-party Inference Stack Integration
IP and Tools Compatibility
Frequently Asked Questions
Branching and Tagging Strategy
» Quick Start Guide for VCK190
View page source

Quick Start Guide for VCK190

The AMD **DPUCVDX8G** for Versal™ VCK190 is a configurable computation engine dedicated to convolutional neural networks. It supports a highly optimized instruction set, enabling the deployment of most convolutional neural networks. The following instructions will help you to install the software and packages required to support VCK190.

Prerequisites

Host Requirements

- Confirm that your development machine meets the minimum Host System Requirements.
- Confirm that you have at least **100GB** of free space in the target partition.

Applicable Targets

- This quickstart is applicable to the VCK190

See also the VCK190 User Guide

WSL

This is an optional step intended to enable Windows users to evaluate Vitis™ AI.

Although this is not a fully tested and supported flow, in most cases users will be able to execute this basic tutorial on Windows. The Windows Subsystem for Linux (WSL) can be installed from the command line. Open a Powershell prompt as an Administrator and execute the following command:

```
[Powershell] > wsl --install -d Ubuntu-20.04
```

The user can list all distribution options available and select an alternate provided that it meets Host System Requirements.

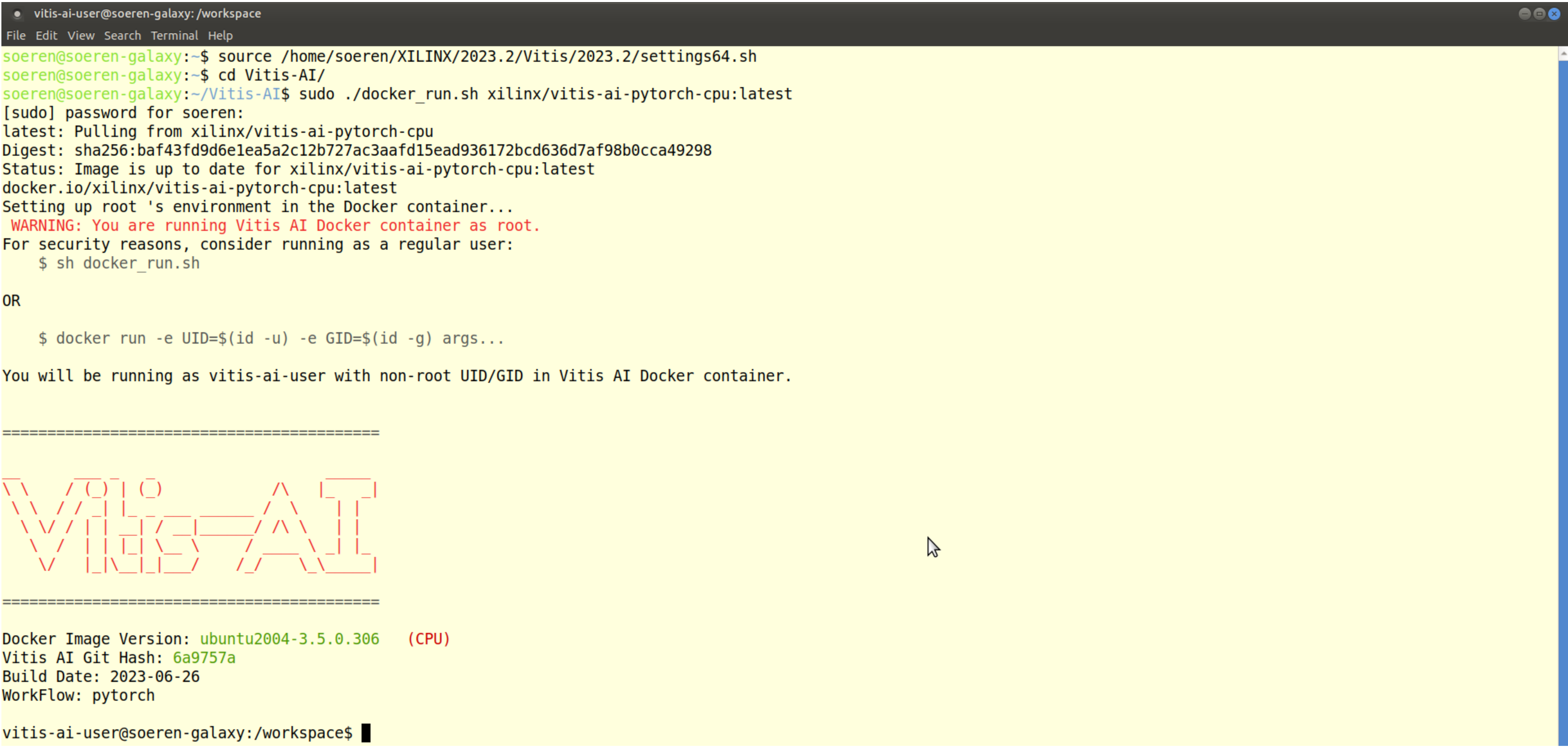
```
[Powershell] > wsl --list --online
```

You can start a specific distribution as follows:

```
[Powershell] > wsl -d Ubuntu-20.04
```

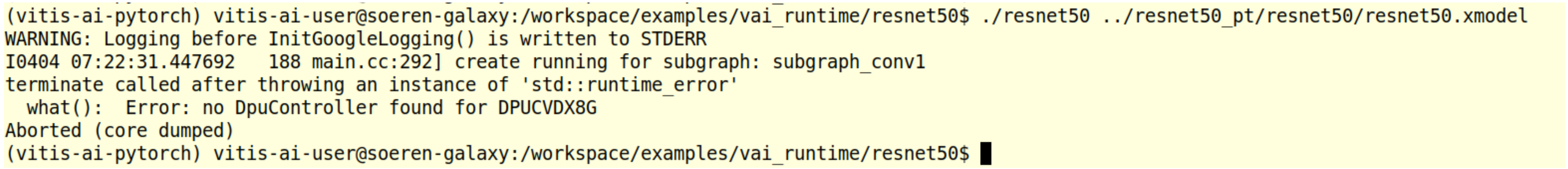
Quickstart

Test of VITIS console



vitis-ai-user@soeren-galaxy: /workspace
File Edit View Search Terminal Help

```
soeren@soeren-galaxy:~$ source /home/soeren/XILINX/2023.2/Vitis/2023.2/settings64.sh
soeren@soeren-galaxy:~$ cd Vitis-AI/
soeren@soeren-galaxy:~/Vitis-AI$ sudo ./docker_run.sh xilinx/vitis-ai-pytorch-cpu:latest
[sudo] password for soeren:
latest: Pulling from xilinx/vitis-ai-pytorch-cpu
Digest: sha256:baf43fd9d6e1ea5a2c12b727ac3aafd15ead936172bcd636d7af98b0cca49298
Status: Image is up to date for xilinx/vitis-ai-pytorch-cpu:latest
docker.io/xilinx/vitis-ai-pytorch-cpu:latest
Setting up root 's environment in the Docker container...
 WARNING: You are running Vitis AI Docker container as root.
For security reasons, consider running as a regular user:
    $ sh docker_run.sh

OR

    $ docker run -e UID=$(id -u) -e GID=$(id -g) args...

You will be running as vitis-ai-user with non-root UID/GID in Vitis AI Docker container.


==========================================

__      ___ _   _                   _____
\ \    / (_) | (_)            /\   |_   _|
 \ \  / / _| |_ _ ___ ______ /  \    | |
  \ \/ / | | __| / __|______/ /\ \   | |
   \  /  | | |_| \__ \     / ____ \ _| |_
    \/   |_|\__|_|___/    /_/    \_\_____|

==========================================

Docker Image Version: ubuntu2004-3.5.0.306   (CPU)
Vitis AI Git Hash: 6a9757a
Build Date: 2023-06-26
WorkFlow: pytorch

vitis-ai-user@soeren-galaxy:/workspace$ 
```



```
(vitis-ai-pytorch) vitis-ai-user@soeren-galaxy:/workspace/examples/vai_runtime/resnet50$ ./resnet50 ../resnet50_pt/resnet50/resnet50.xmodel
WARNING: Logging before InitGoogleLogging() is written to STDERR
I0404 07:22:31.447692   188 main.cc:292] create running for subgraph: subgraph_conv1
terminate called after throwing an instance of 'std::runtime_error'
  what():  Error: no DpuController found for DPUCVDX8G
Aborted (core dumped)
(vitis-ai-pytorch) vitis-ai-user@soeren-galaxy:/workspace/examples/vai_runtime/resnet50$ 
```

Conclusion and next steps

- Board arrived only a few weeks ago; testing just started
- Need to solve the hardware platform support problem (VPK120 instead of VCK190)
- Then repeat timing measurements for matrix multiplication on hardware

  (hoping that 1250 MHz is correct frequency)