Accelerated Deep Learning Inference on FPGAs in the Space Domain

SpacE FPGA Users Workshop 2023

DEFENCE AND SPACE

<Name>

March 16, 2023

AIRBUS

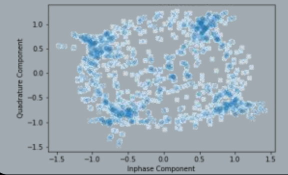
Part 1:

Deep Learning in the Space Domain

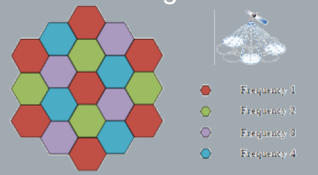
Artificial Intelligence in Space

Radio Frequency

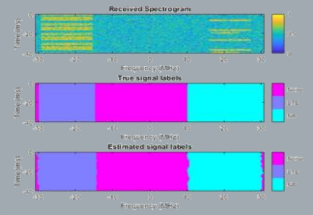
Modulation recognition



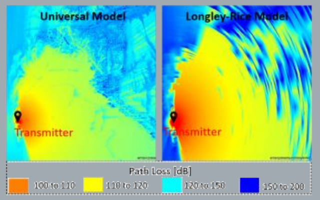
Beamforming



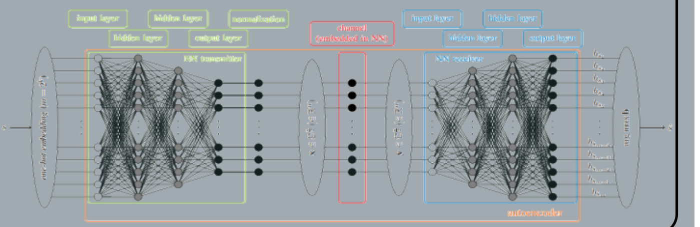
Spectrum sensing



Terminal localization



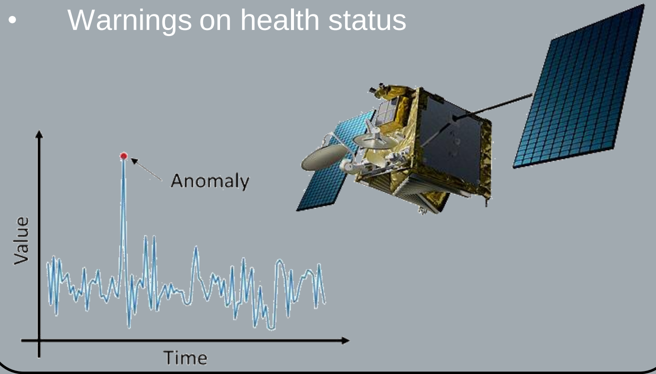
ML-augmented signal processing



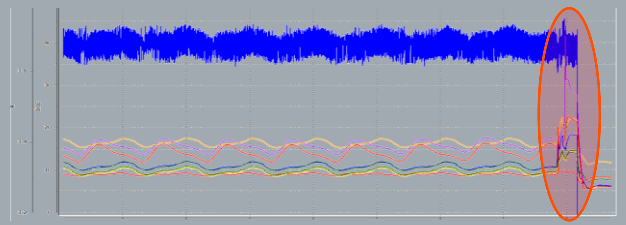
Anomaly Detection

ML-based telemetry data processing

- Prediction of known failures
- Detection of unknown anomalies
- Warnings on health status



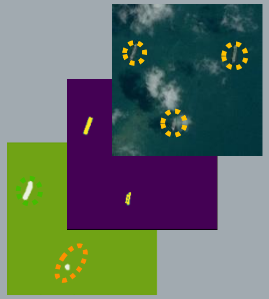
Corona discharge detection



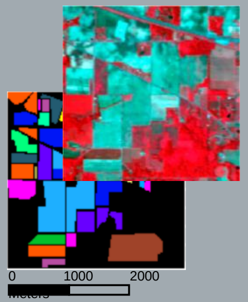
Computer Vision

On-board processing of satellite images

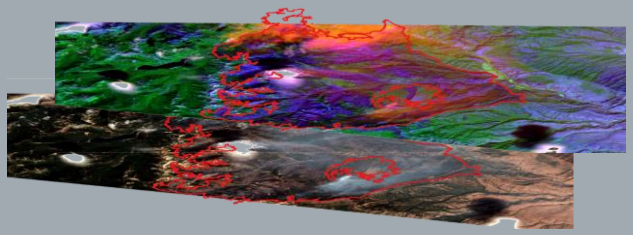
Ship detection



Segmentation



Wildfire detection



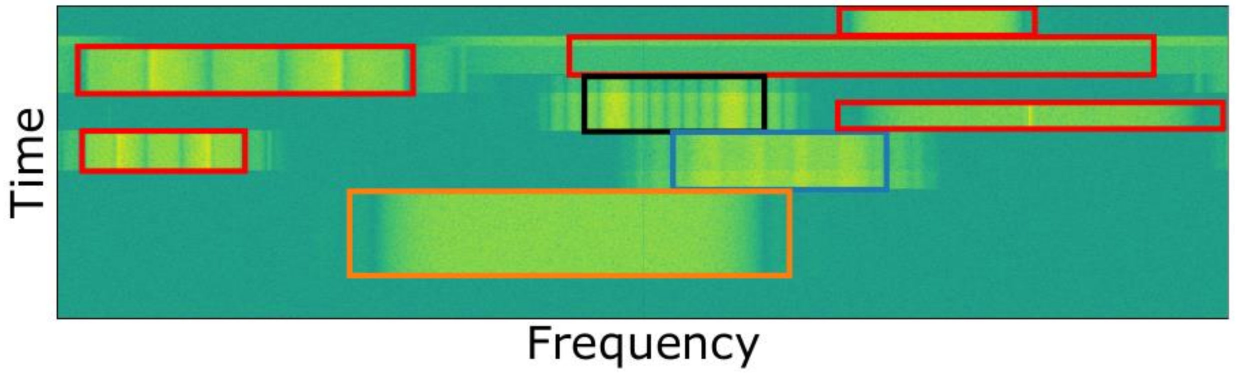
March 2023

Use Case: Spectrum Analysis

Goals:

1) Detecting the **presence** of signals in the electromagnetic spectrum (Signal-of-Interest / Interference)
2) Estimating the "**location**" of present signals (Center frequency / Bandwidth / Duration)

Spectrogram (based on FFT):



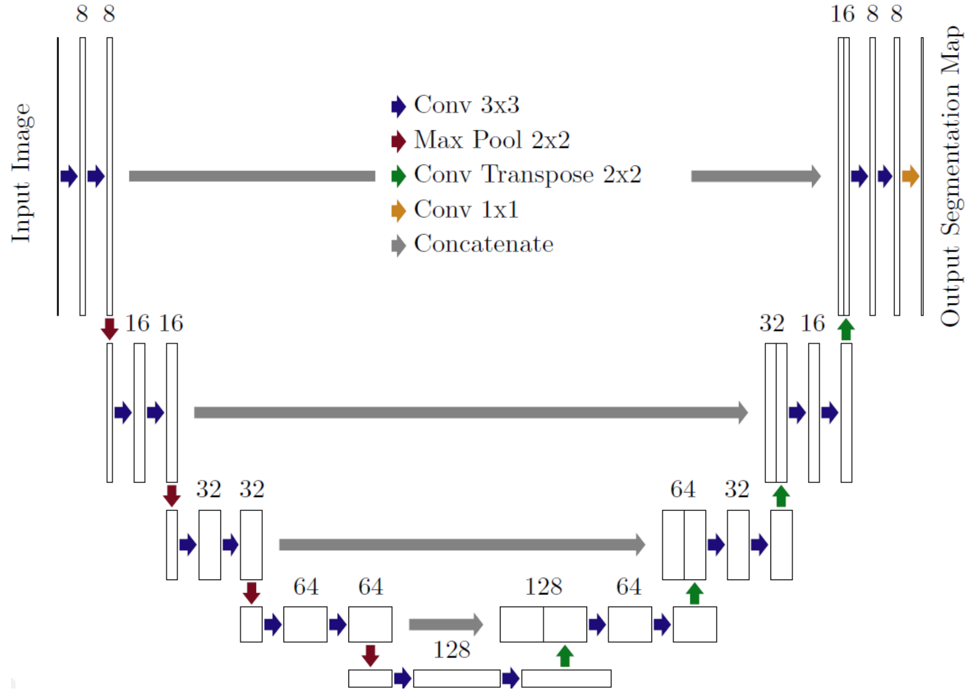
Applications:

⇨ Monitoring of the electromagnetic spectrum from space (Regulatory purposes)

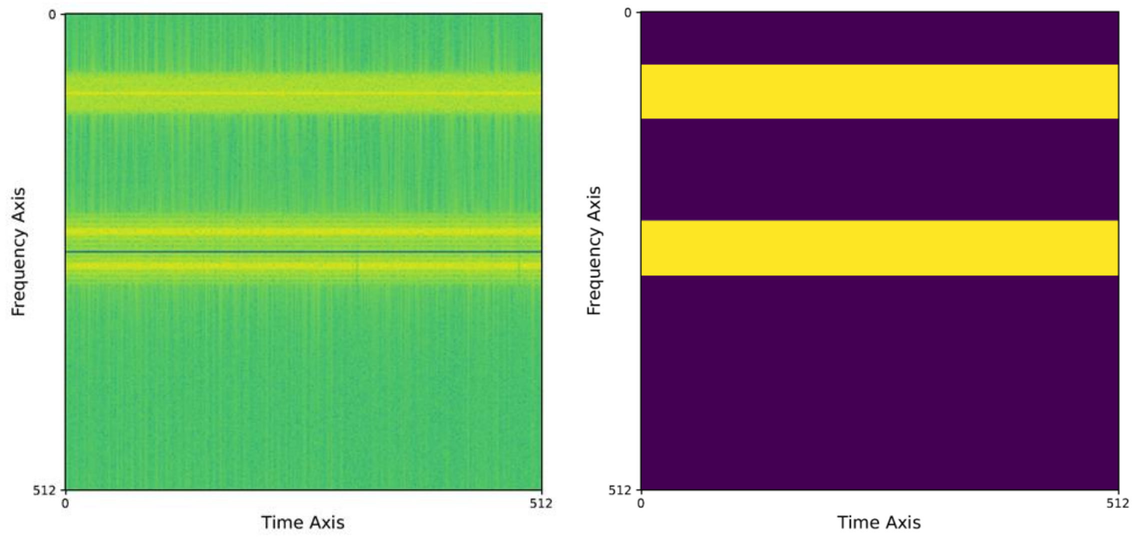
⇨ "Intelligent" radios can use the information about spectrum occupancy for opportunistically accessing unused / underutilized frequency bands ("Dynamic Spectrum Access")

Use Case: Spectrum Analysis

ML Approach: U-Net based **Convolutional Neural Network** for image segmentation



Inputs: 512 x 512 Spectrogram "Images"



Outputs: 512 x 512 Segmentation Maps

Part 2:

Xilinx Versal for Space Applications

Space-Grade Versal ACAP

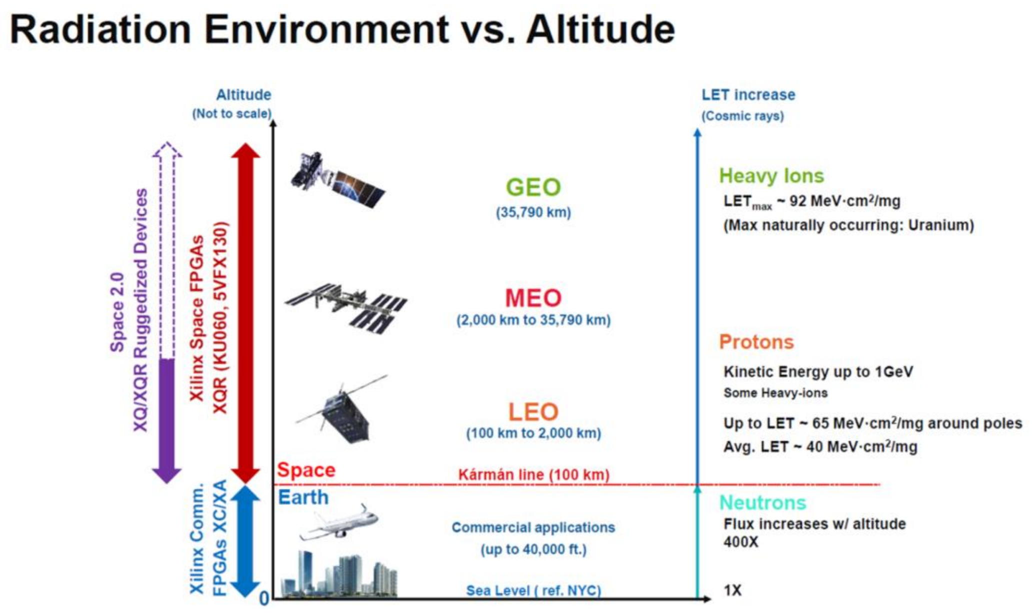
The space-grade Versal is a **radiation tolerant System-on-Chip** intended for Satellite & Space applications

Its powerful, heterogeneous processing architecture enables a multitude of **Space 2.0 applications**:

- Machine Learning & Artificial Intelligence
- Broadband Internet
- High-Speed Networks
- Cloud & Object Detection
- …

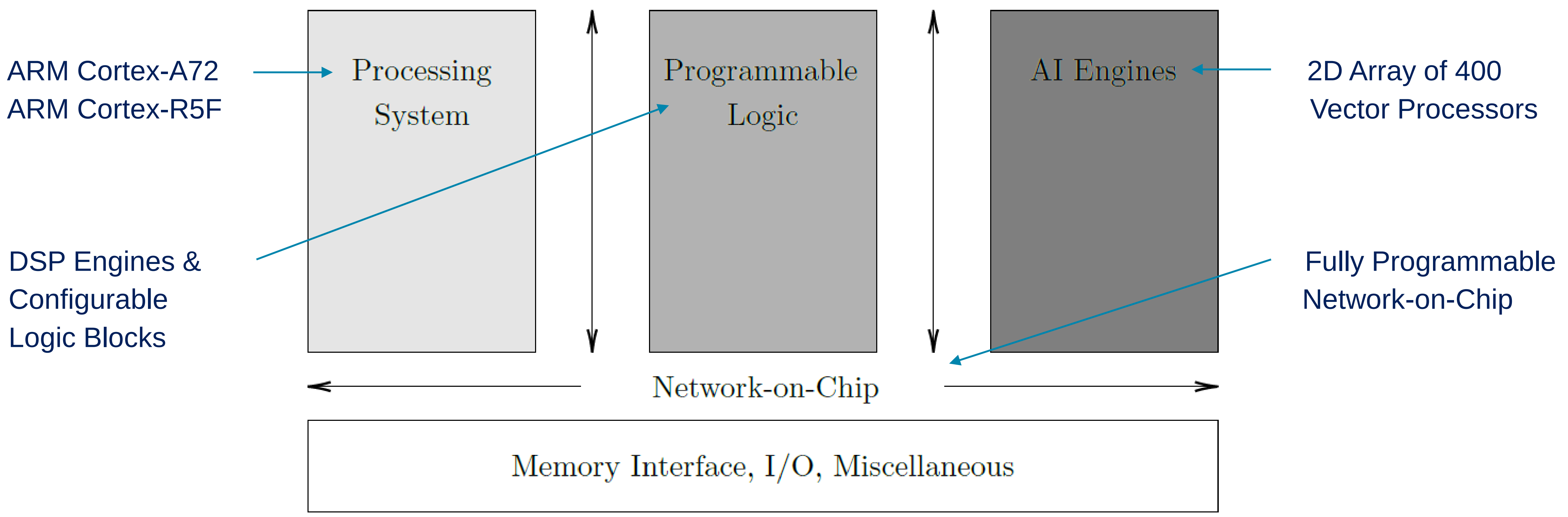
Features of the XQR Versal:

- Designed for LEO missions with a duration of 5 to 7 years
- Xilinx Soft Error Mitigation (XilSEM) Library for Detecting & Correcting Soft Errors (Single Event Upsets)
- Ruggedized Organic Packaging
- ITAR-free, but US Technology



Versal Architecture

Architectural Overview:



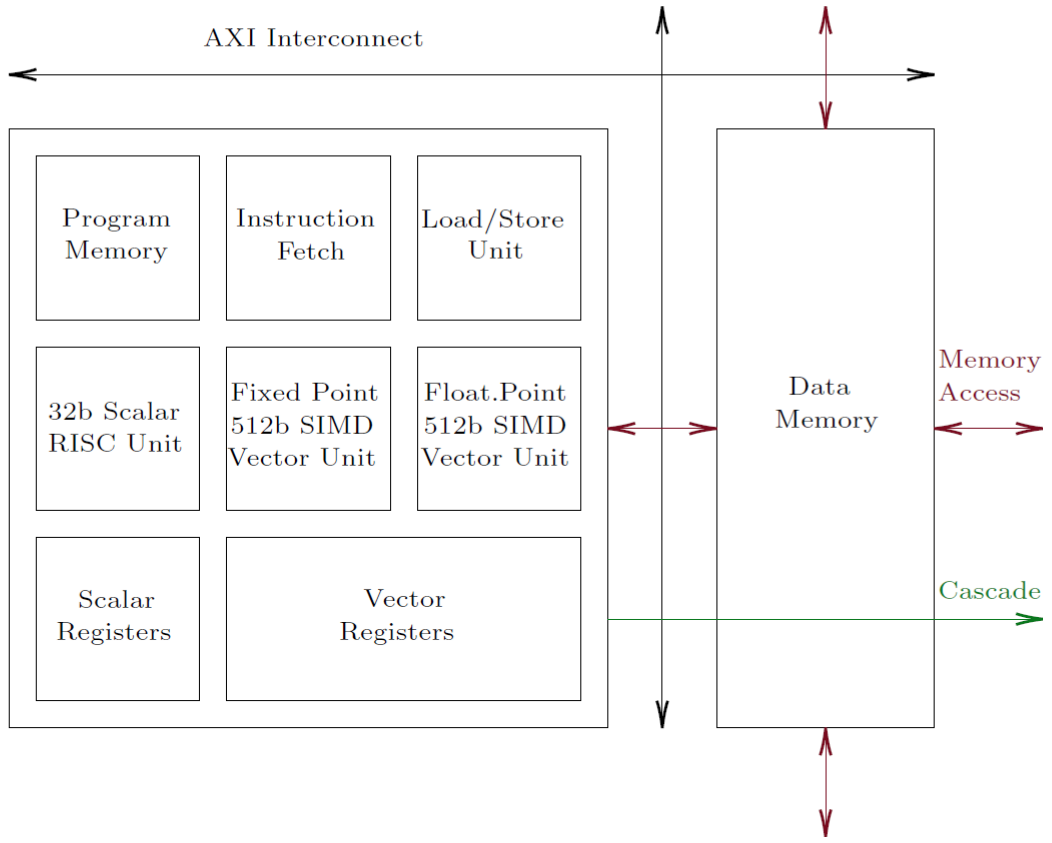
Versal AI Engines

Make use of three levels of computing parallelism:

1. **Data-Level (SIMD):**
   Vector operations, e.g. addition of two int32 vectors with 16 elements -> add16(v16int32 x, v16int32 y)

2. **Instruction-Level (VLIW):**
   Execution of up to 7 operations in parallel
   (Load x2, Store, Scalar Op, Move x2, Vector Op)

3. **Multicore-Level:**
   Up to 400 AI Engines working in parallel in a 2D array



Machine Learning on FPGAs

Three general approaches for **Accelerating ML Applications on FPGA**-based systems:

1. Use of a predesigned, generic (programmable) co-processor IP Core for executing neural networks

   e.g. Xilinx Deep Learning Processing Unit

2. Use of an automatic framework to generate HDL/HLS design for a co-processor that targets a specific neural network

   e.g. MATLAB HDLCoder, FINN

   (Currently not well suited for the Versal due to its heterogeneous architecture)

3. Design of a custom co-processor in HDL/HLS for executing a specific neural network

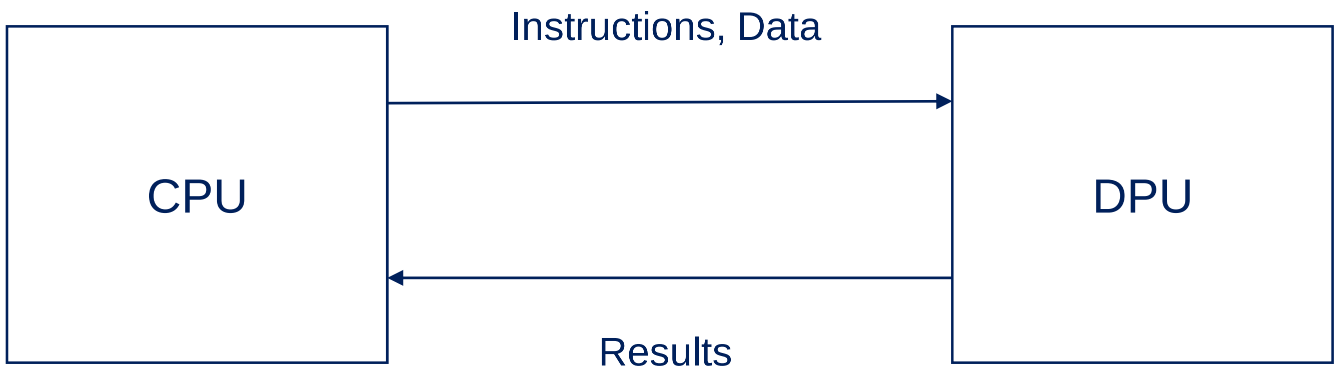
   e.g. VHDL, Vitis HLS

Part 3:

Inference via Predesigned IP Cores

Xilinx Deep Learning Processing Unit

- Xilinx Deep Learning Processing Units (DPUs) are **predesigned IP Cores** optimized for executing neural networks
- In particular, DPUs are **Co-Processors** / Hardware Accelerators controlled by means of dedicated instructions
- Neural networks are **automatically quantized and compiled** into instructions for the DPU via Xilinx tools

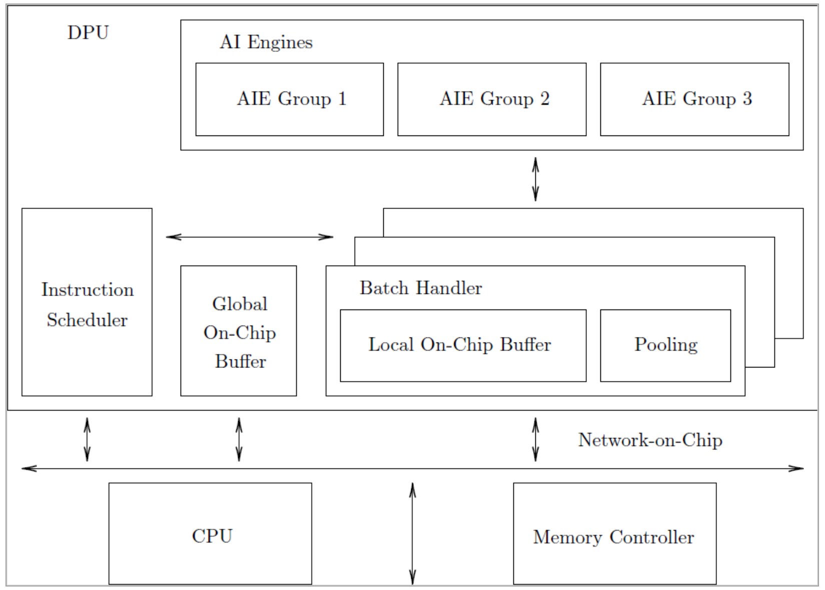


DPU uses both the PL resources as well as the AI Engines !

Overview of the Development Flow

Hardware

Xilinx Deep Learning Processing Unit (DPU)
- Programmable IP Core
- Supports a variety of network layers, e.g. Conv2D / 3D, Dense, Max Pooling



Software

AI application development
with Python & TensorFlow / PyTorch

↓

Vitis AI Optimizer
Pruning of neural networks

↓

Vitis AI Quantizer
Quantization of parameters

↓

Vitis AI Compiler
Generation of DPU instructions

Use Case: Spectrum Analysis

Performance Comparison:

	Zynq UltraScale+	Versal ACAP
- **DPU Configuration:**	Maximum Resources	Minimum Resources
	PL Frequency: 325 MHz	PL Frequency: 333 MHz
		AIE Frequency: 1250 MHz
- **Performance Metrics:**	Throughput: 50 frames / second	Throughput: 79 frames / second
	Latency: 19.4 milliseconds	Latency: 12.3 milliseconds

- **Main Challenge:** Higher power consumption of the Versal compared to the UltraScale+ !

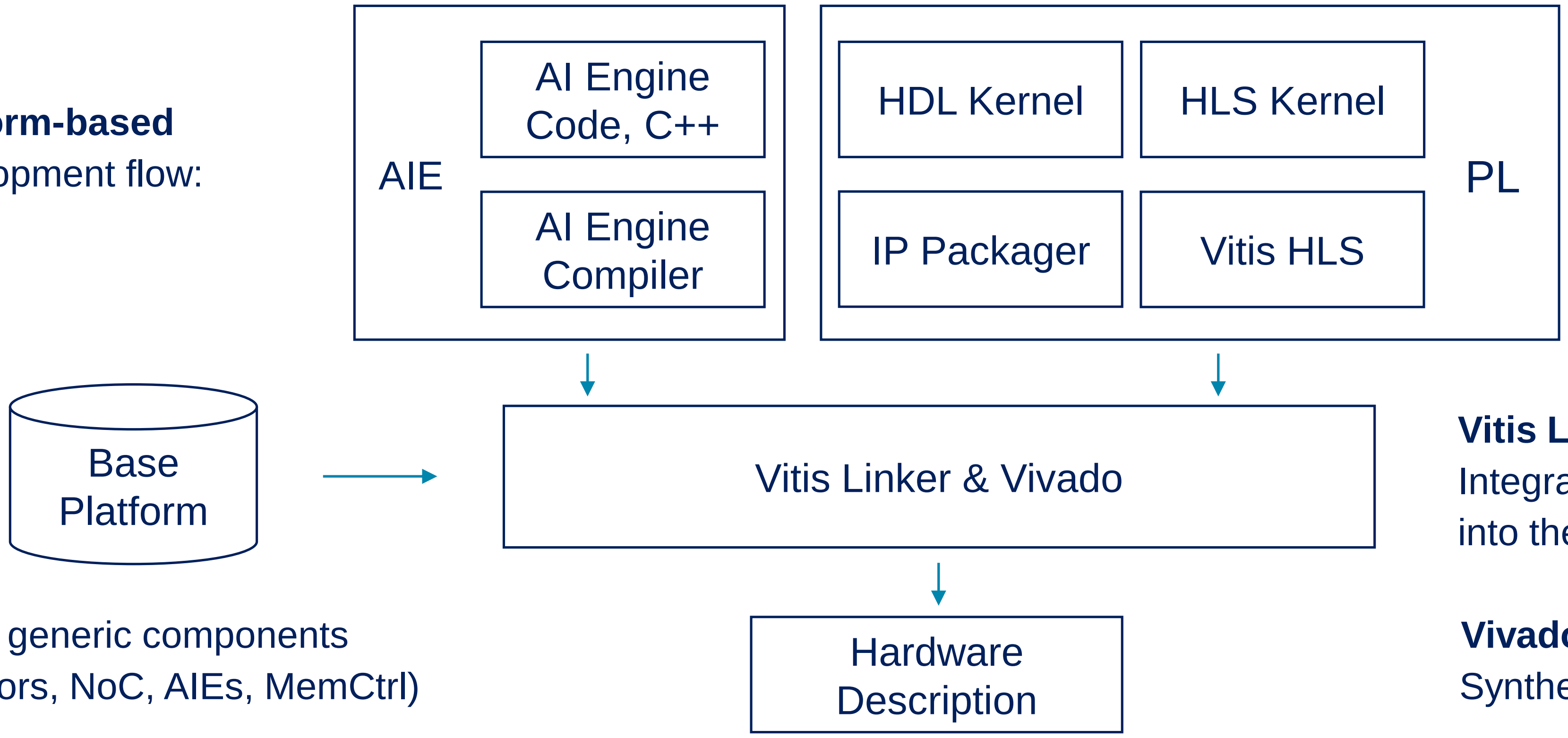
Part 4:

Inference via Custom Co-Processors

Application Acceleration on the Versal

Heterogeneous Versal platform requires a **Hardware/Software-Codesign** approach !
-> Partitioning of the application into functions that are executed on the PL resources and AIEs, respectively

Platform-based
development flow:



Only generic components
(Processors, NoC, AIEs, MemCtrl)

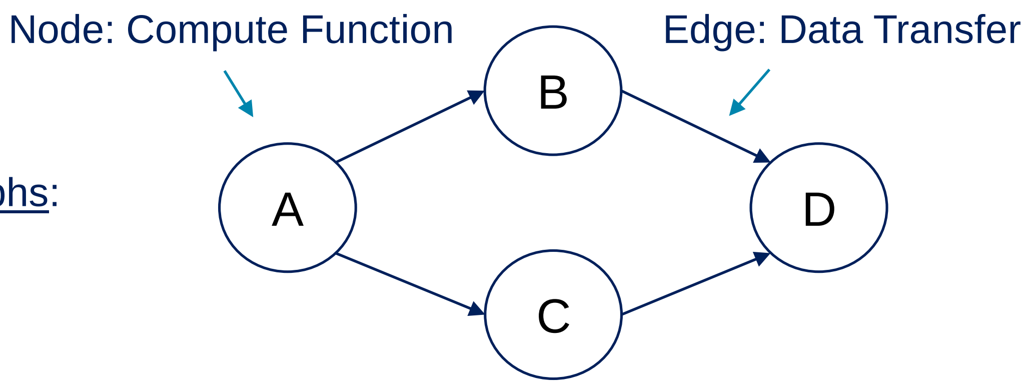
Vitis Linker:
Integration of kernels
into the base platform

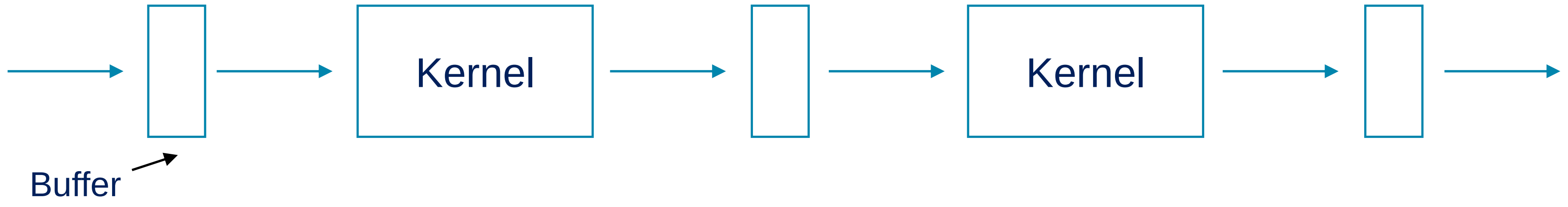
Vivado:
Synthesis & Implementation

AIRBUS

AI Engine Programming Model

Programming of the AI Engines is done based on Data-Flow Graphs:





Communication:
- Kernel waits until input buffer is full
- Kernel is executed and writes data into output buffer

Computation:
- Data is loaded into vector registers
- Vector functions operate on data in registers (e.g. add, mul, …)

AI Engine Programming

Example: Implementation of a 1D Convolution Layer on the AIEs

```
static int8 weights[128] = { … };                  // Weights and bias values
static int16 bias[8] = { … };                      // are permanently stored in
static v16int8 prev;                               // the AIE data memory

void conv1d(input_window_int8 * in, output_window_int8 * out) {
    v16int8 curr = window_readincr_v16( in );      // Read in data samples from the input buffer
    v32int8 X = concat( prev, curr );
    v64int8 Y;
    v8acc48 acc;                                   // Accumulator registers store the intermediate multiplication results

    for ( unsigned i=0; i<8; i++ ) {
        acc = ups( bias, B_SHFT );                 // Initialize accumulators with bias values
        acc = mac8( acc, weights, …, X, 2*i, … );  // Accumulate the results of the matrix-vector multiplication
        Y = upd_v( Y, i, srs( acc, S_SHFT ) );     // Place the results into the output vector
    }
    Y = maxdiff( Y, null_v64int8() );              // Apply the ReLU activation function to the output vector
    window_writeincr( out, Y );                    // Write the results to the output buffer
    prev = curr;
}
```

Appendix: 1D Convolution Layer

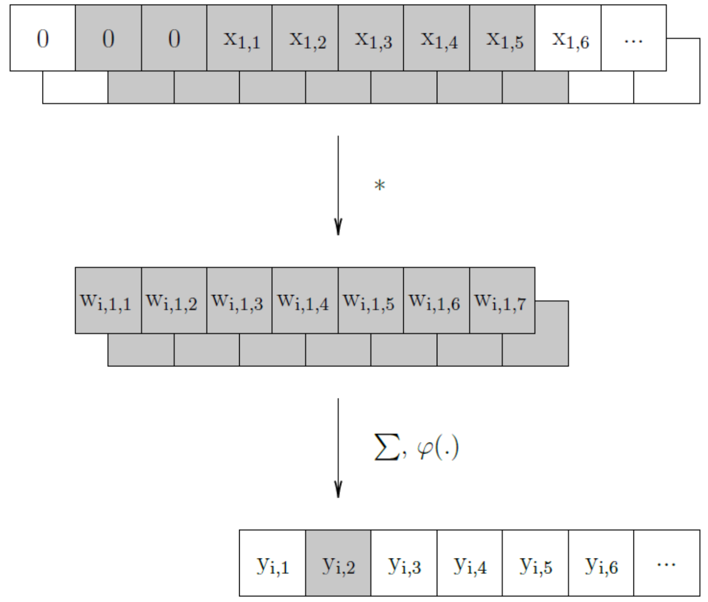
One-Dimensional Convolution Operation for CNNs:

Mathematical Description

$$
\begin{aligned}
y_{i,j} &= \varphi\left(\sum_{k=1}^{7} w_{i,1,k} x_{1,j-1+k} + \sum_{k=1}^{7} w_{i,2,k} x_{2,j-1+k} + b_i\right) \\
&= \varphi\left(\sum_{l=1}^{2}\sum_{k=1}^{7} w_{i,l,k} x_{l,j-1+k} + b_i\right) \\
&= \varphi\left(\sum_{k=1}^{7}\sum_{l=1}^{2} w_{i,l,k} x_{l,j-1+k} + b_i\right),
\end{aligned}
$$

-> Realization as matrix-vector-multiplication

Graphical Illustration



Thank you!

AIRBUS

Hardware Platforms

Versal AI Core VC1902

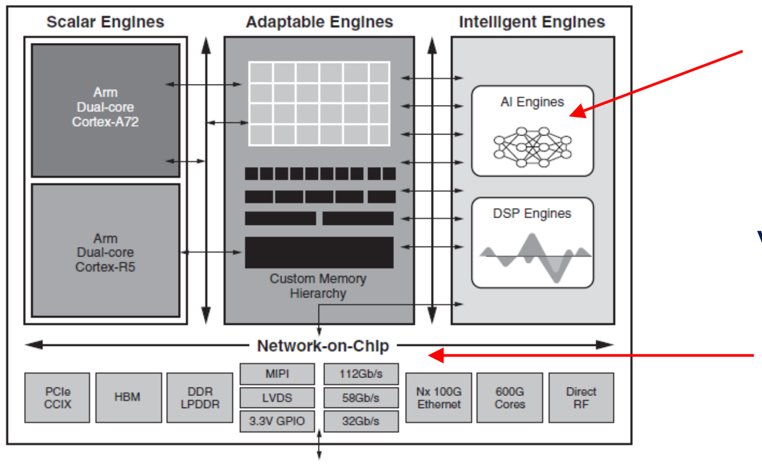
Processor System
Arm Cortex-A72 (x2)
Arm Cortex-R5F (x2)

Programmable Logic & Engines
Lookup Tables (900k)
DSP Engines (x1968)
AI Engines (x400)

Zynq UltraScale+ ZU9EG

Processor System
Arm Cortex-A53 (x4)
Arm Cortex-R5F (x2)

Programmable Logic & Engines
Lookup Tables (274k)
DSP Engines (x2520)



Versal Architecture

AIRBUS

Challenges: Power Consumption

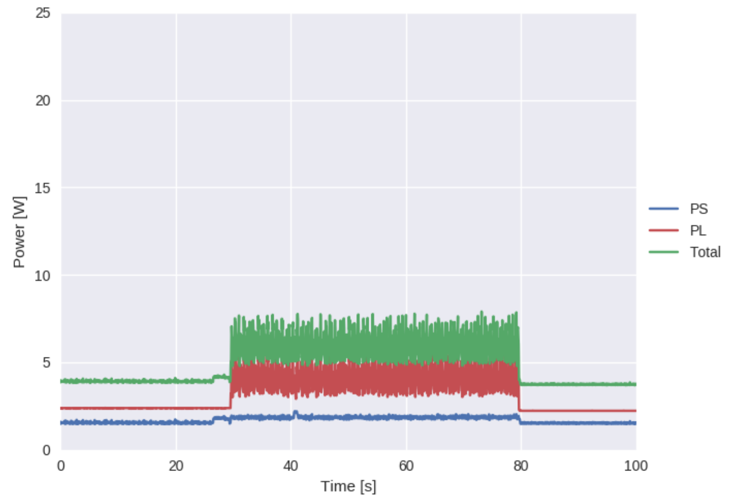
Example: Power Consumption for Inference with the DPU

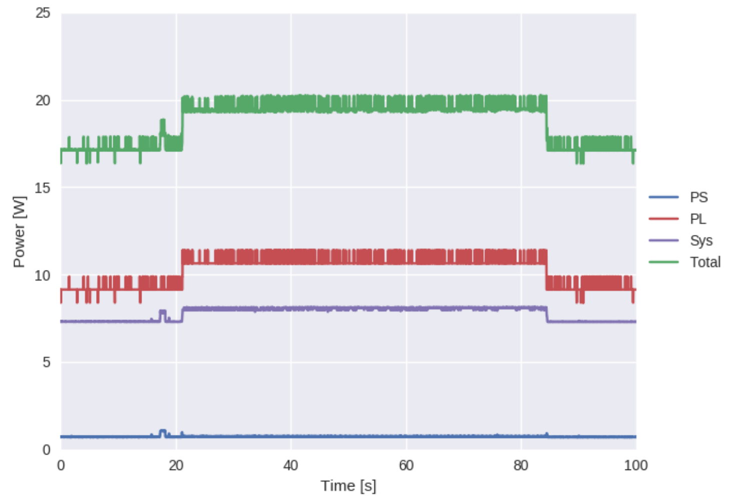
Zynq UltraScale+:

Resource		Utilization
LUT	(52k / 274k)	19 %
DSP	(710 / 2520)	28 %

Versal:

Resource		Utilization
LUT	(81k / 900k)	9 %
DSP	(139 / 1968)	7 %
AIE	(32 / 400)	8 %


Power [W]
Time [s]
PS
PL
Total


Power [W]
Time [s]
PS
PL
Sys
Total