FPGA-based hardware acceleration of machine learning algorithms for particle accelerators.

Submission #5

10.10.2022

Low Level RF Workshop 2022

9-13 Oct 2022, Brugg-Windisch, Switzerland

PAUL SCHERRER INSTITUT PSI

CERN

Gianluca Martino, Ahmad Al Zoubi, Julien Branlard, Holger Schlarb, Goerschwin Fey

TUHH Hamburg University of Technology

DESY.

Universität Hamburg DER FORSCHUNG | DER LEHRE | DER BILDUNG

DASHH

CDCS CENTER FOR DATA AND COMPUTING IN NATURAL SCIENCES

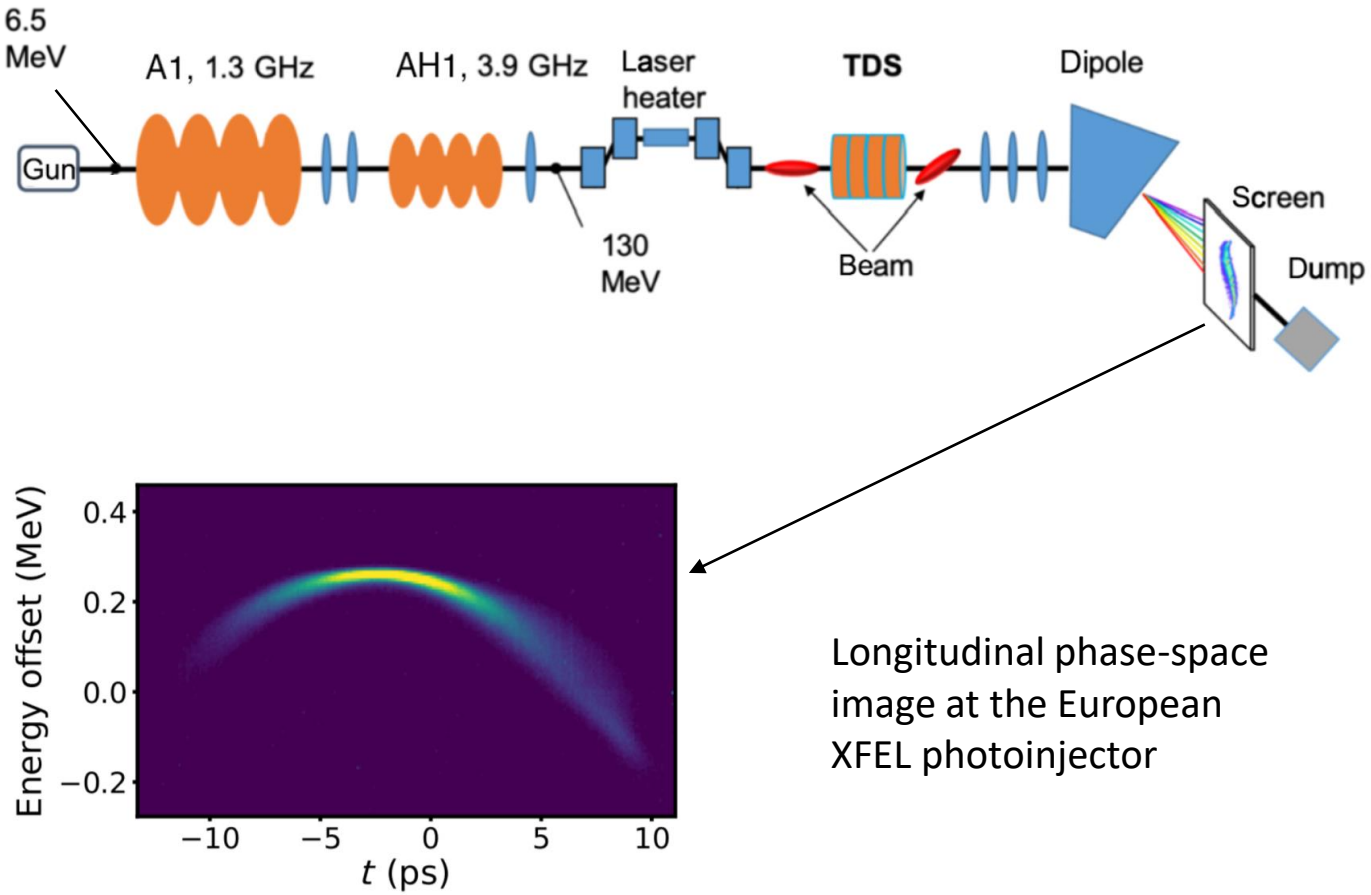
Motivation

Applications

- Some of our current diagnostics are not suitable for future particle accelerators

  ↓

  high beam intensity, very short pulses



Longitudinal phase-space image at the European XFEL photoinjector

Motivation

Applications

- Some of our current diagnostics are not suitable for future particle accelerators

  ↓

  high beam intensity, very short pulses

- The increasing **precision and reliability requirements** require **new approaches** to measuring the **beam properties** and **monitoring** the LLRF system.

High-Fidelity Prediction of Megapixel Longitudinal Phase-Space Images of Electron Beams Using Encoder-Decoder Neural Networks

J. Zhu, Y. Chen, F. Brinker, W. Decking, S. Tomin, and H. Schlarb
Phys. Rev. Applied **16**, 024005 – Published 3 August 2021

Open Access Article

Virtual Diagnostic Suite for Electron Beam Prediction and Control at FACET-II

by Claudio Emma [1,*,†], Auralee Edelen [1,†], Adi Hanuka [1,†], Brendan O'Shea [1,†] and Alexander Scheinker [2,†]

[1] SLAC National Accelerator Laboratory, Menlo Park, CA 94025, USA
[2] Los Alamos National Laboratory, Los Alamos, NM 87545, USA
[*] Author to whom correspondence should be addressed.
[†] These authors contributed equally to this work.

Motivation

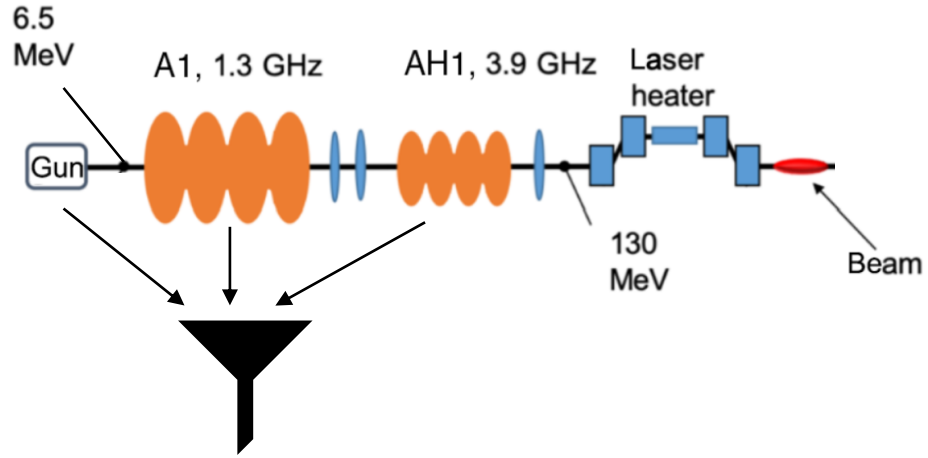
Applications

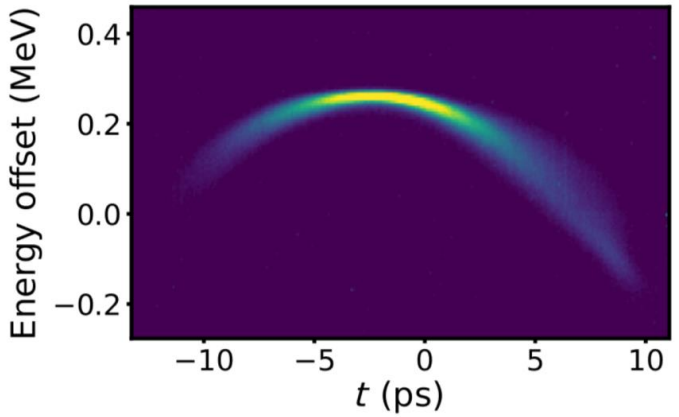
- Some of our current diagnostics are not suitable for future particle accelerators

  ↓

  high beam intensity, very short pulses

- The increasing **precision and reliability requirements** require **new approaches** to measuring the **beam properties** and **monitoring** the LLRF system.





Longitudinal phase-space image at the European XFEL photoinjector

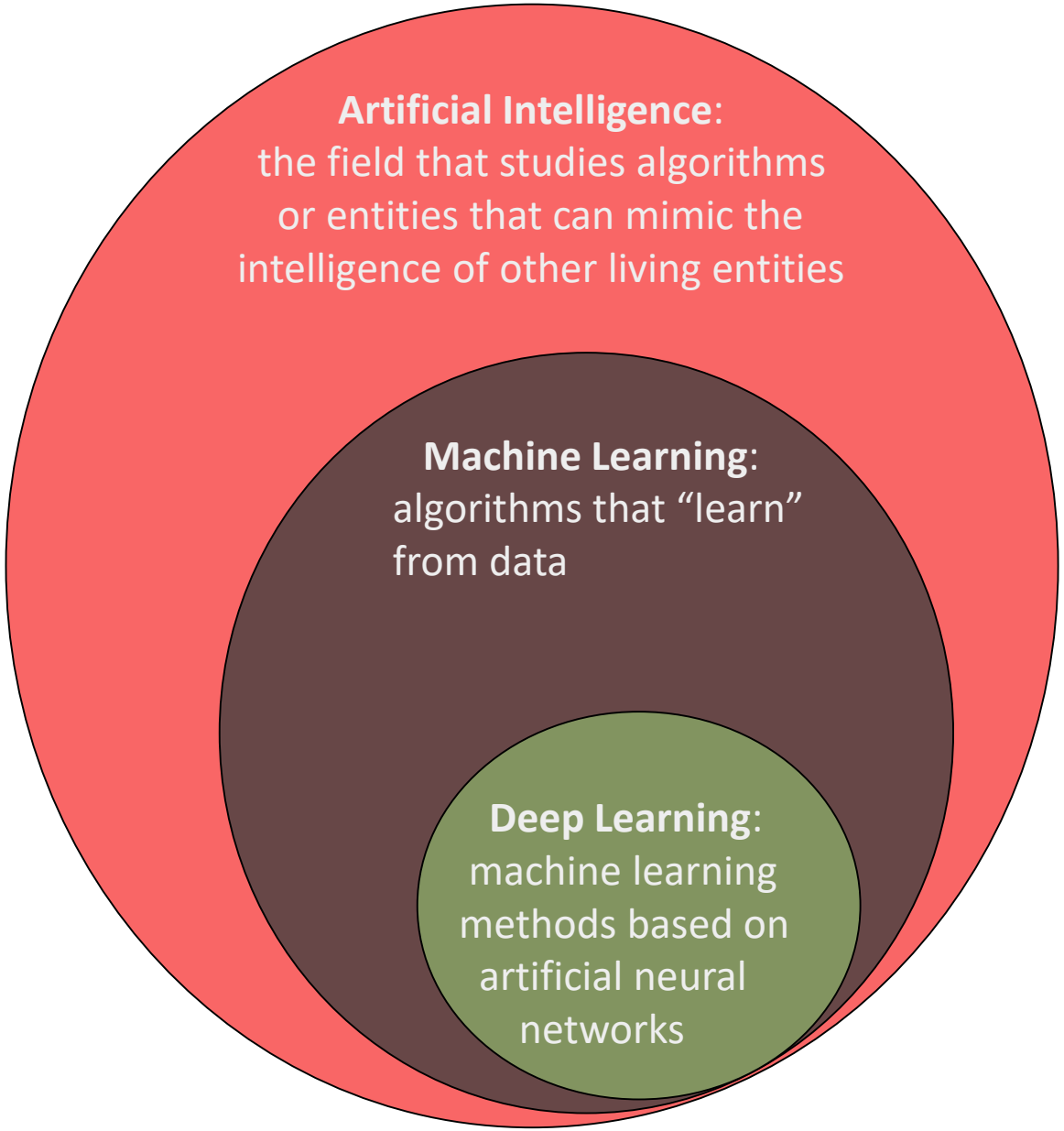
Motivation

Applications

- Some of our current diagnostics are not suitable for future particle accelerators

  ↓

  high beam intensity, very short pulses
- The increasing **precision and reliability requirements** require **new approaches** to measuring the **beam properties** and **monitoring** the LLRF system.
- We would like to use the same algorithms in real-time applications.



Motivation

Applications

- Some of our current diagnostics are not suitable for future particle accelerators

  ↓

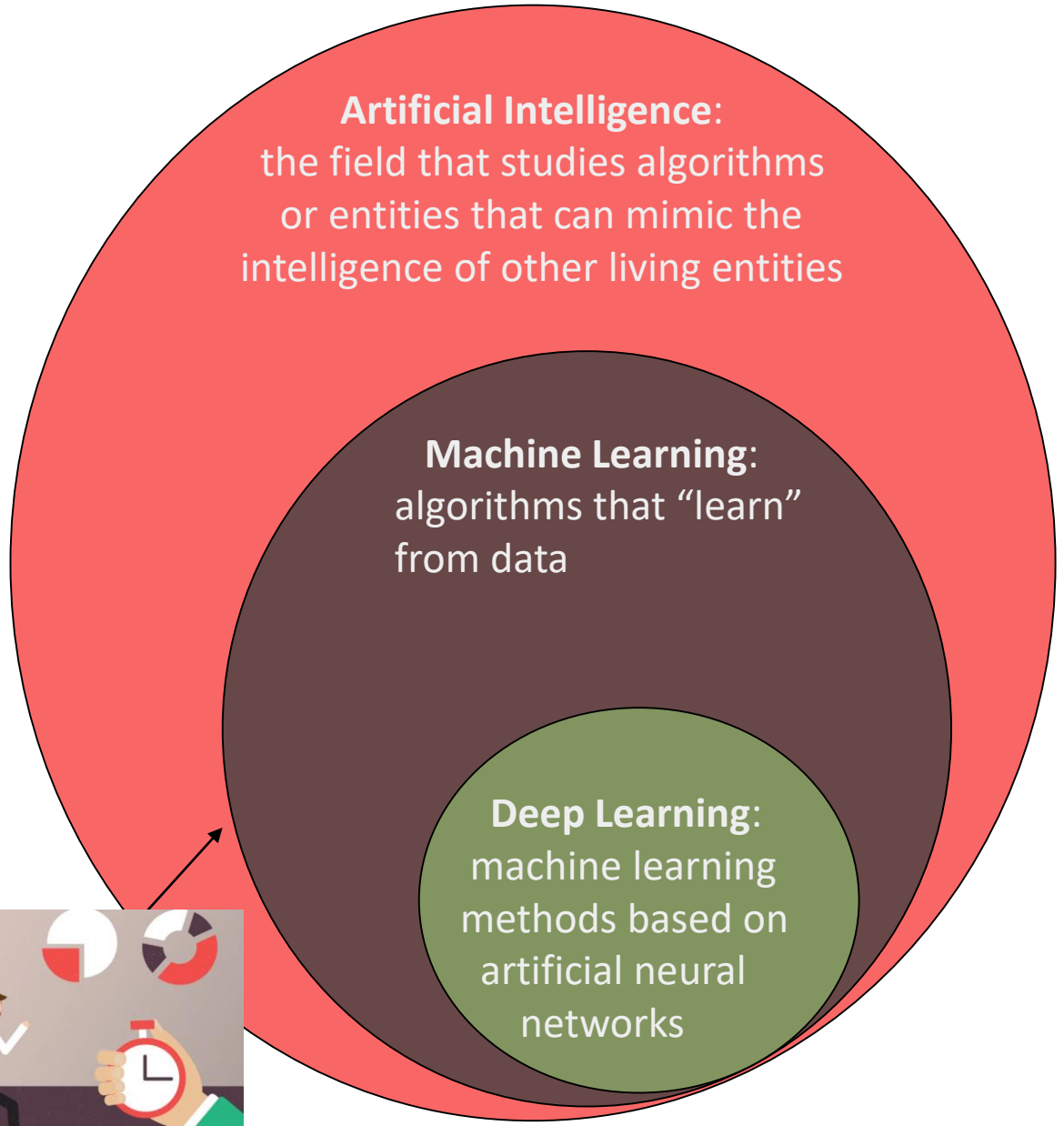
  high beam intensity, very short pulses

- The increasing **precision and reliability requirements** require **new approaches** to measuring the **beam properties** and **monitoring** the LLRF system.

- We would like to use the same algorithms in real-time applications.

  ↓

- This requires the acceleration of **machine learning algorithms**.



Hardware accelerators

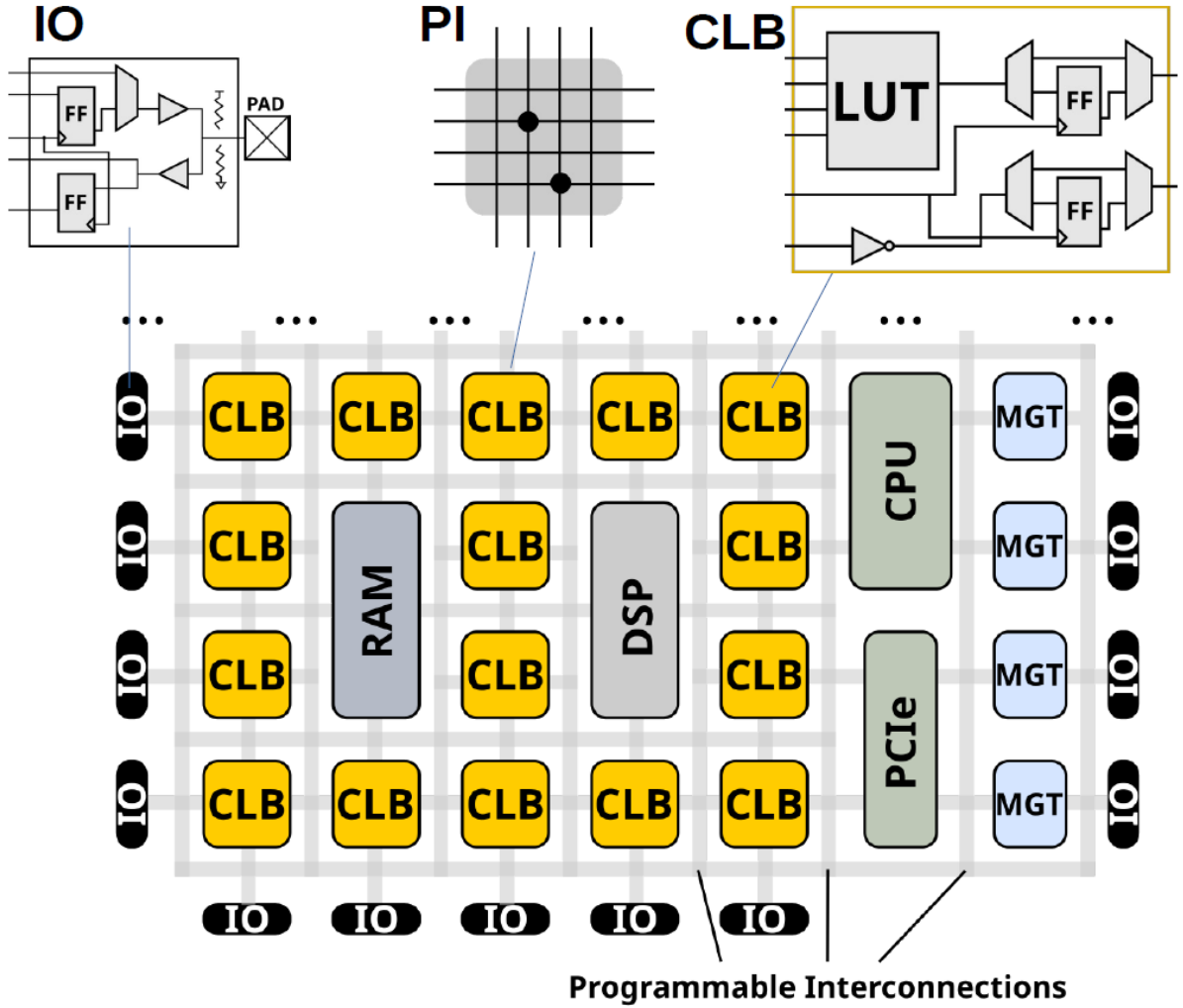
Field-programmable gate arrays

Field-programmable gate arrays (FPGAs) are composed of processing elements, i.e., configurable logic blocks (CLBs), connected by programmable interconnections (PIs).

Each element is programmable through SRAM cells controlling CMOS transistor gates.

Modern FPGAs include also:

- CPUs
- Digital Signal Processing (DSP) units
- Memory controllers
- Artificial Intelligence (AI) engines



Curtesy of L. Butkowski

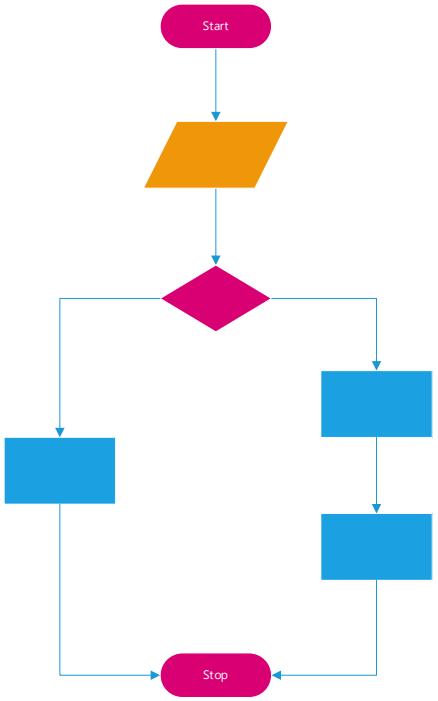
High-Level Synthesis

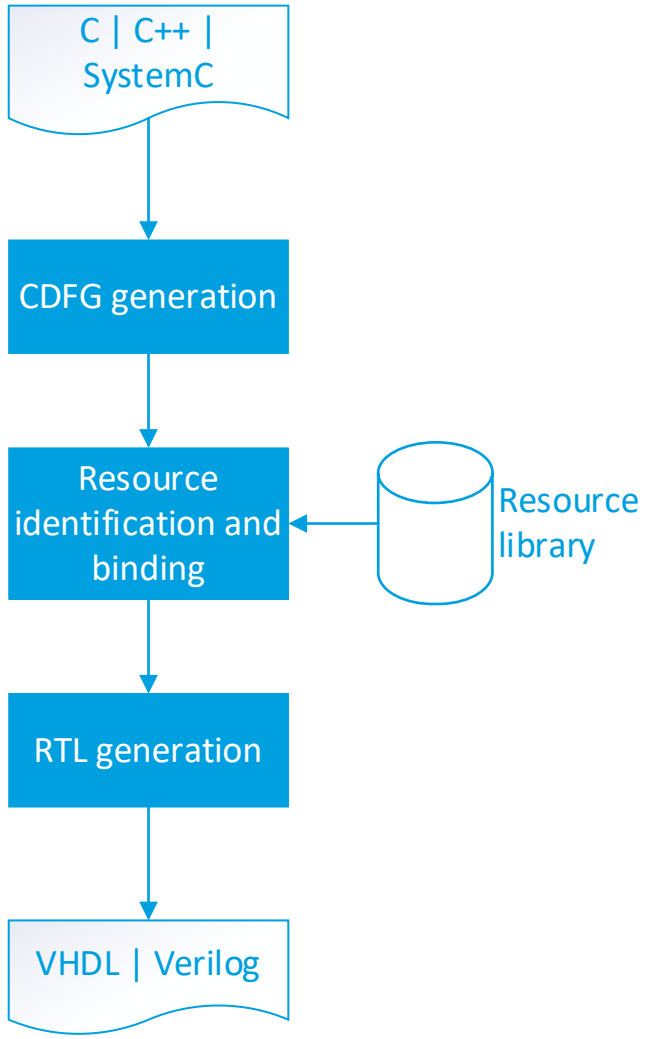
Introduction

High-Level Synthesis (HLS) generates HDL code from a behavioral specification.

The HLS main steps are:

- The creation of a *control and data flow graph* (CDFG) requires parsing and identification of each functional unit





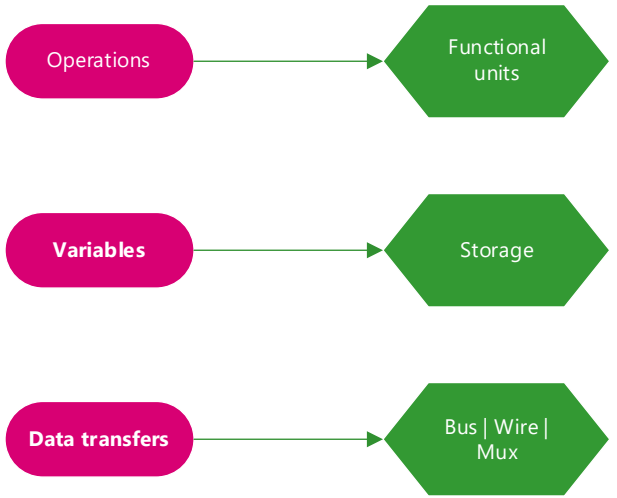
High-Level Synthesis

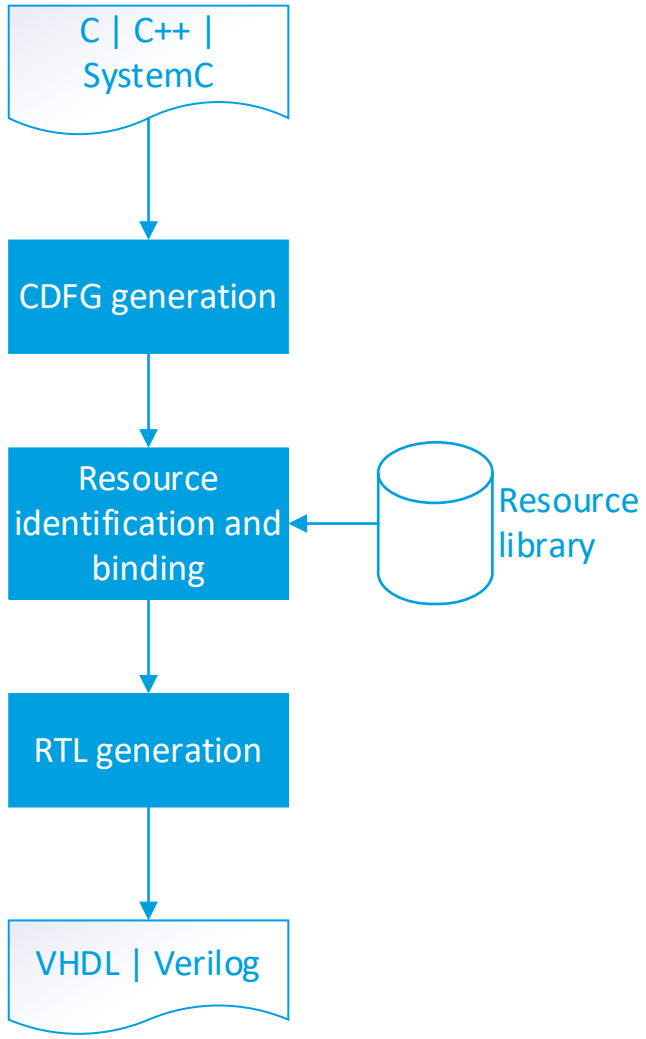
Introduction

High-Level Synthesis (HLS) generates HDL code from a behavioral specification.

The HLS main steps are:

- The creation of a *control and data flow graph* (CDFG) requires parsing and identification of each functional unit
- Resource identification and binding maps the resources available to the targeted FPGA to each element of the CDFG





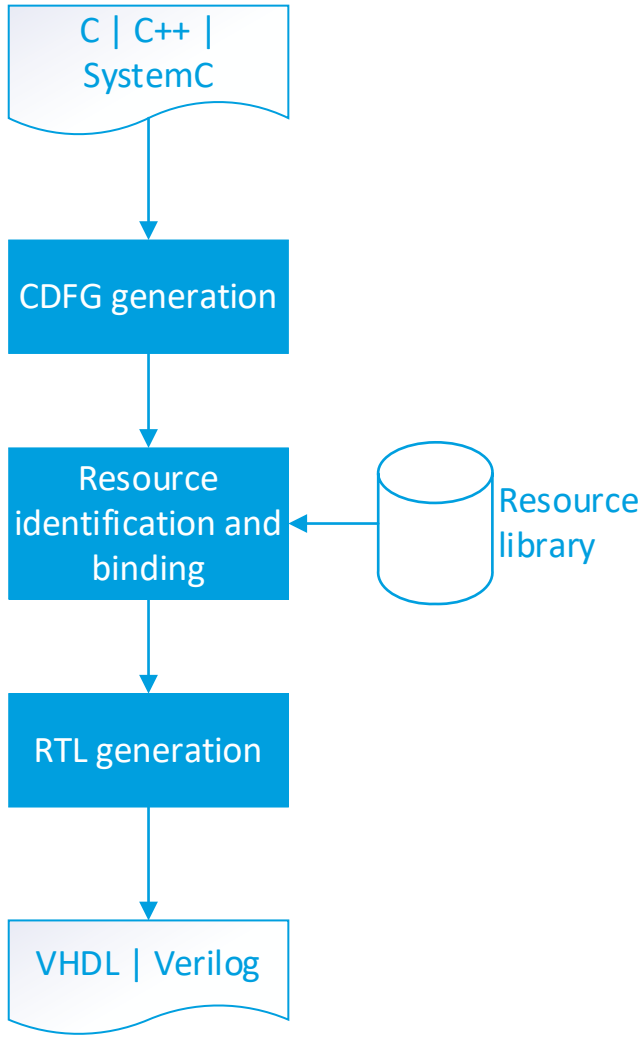
High-Level Synthesis

Introduction

High-Level Synthesis (HLS) generates HDL code from a behavioral specification.

The HLS main steps are:

- The creation of a *control and data flow graph* (CDFG) requires parsing and identification of each functional unit
- Resource identification and binding maps the resources available to the targeted FPGA to each element of the CDFG
- The RTL generation creates HDL code that can be directly used in the target FPGA



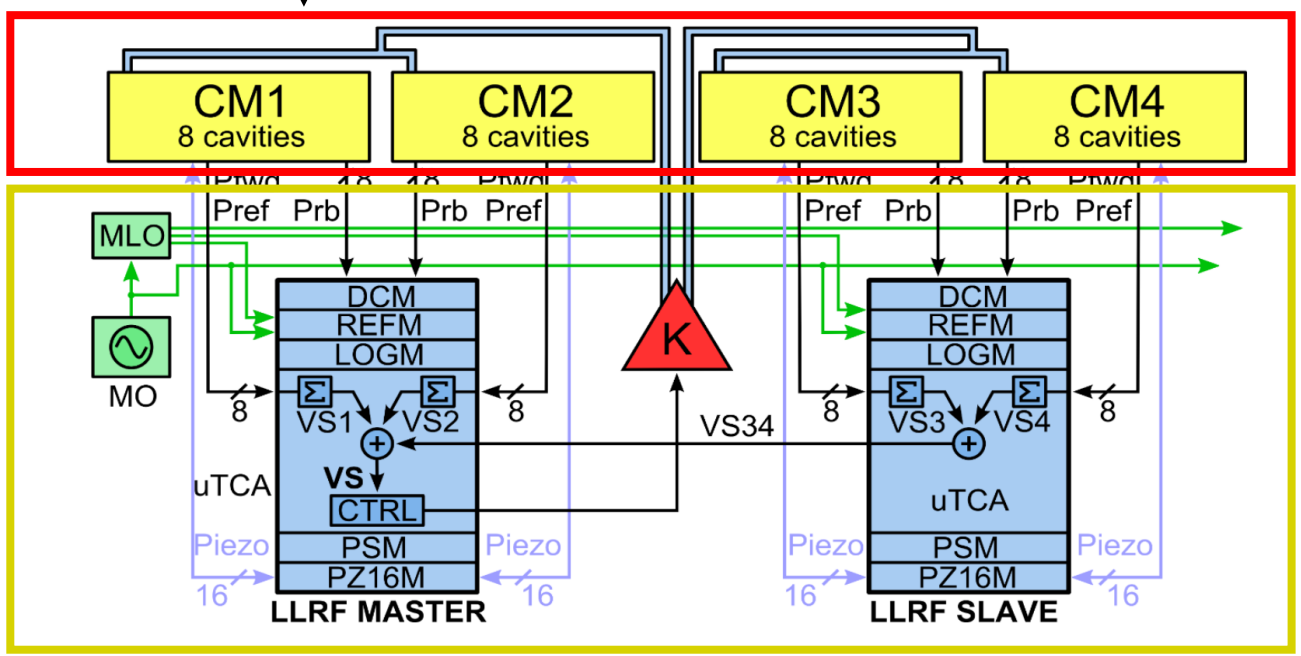
Using HLS for anomaly detection

Applications

Correctness of operations at both the system level and the machine level can be checked using **anomaly detection**.

Anomaly detection techniques include:

- Neural networks acceleration
  - Vitis AI [1], hls4ml
- Property-based anomaly detection
  - c-LTL runtime monitoring [2]
- Statistical anomaly detection
  - K-means, clustering, SVM, …



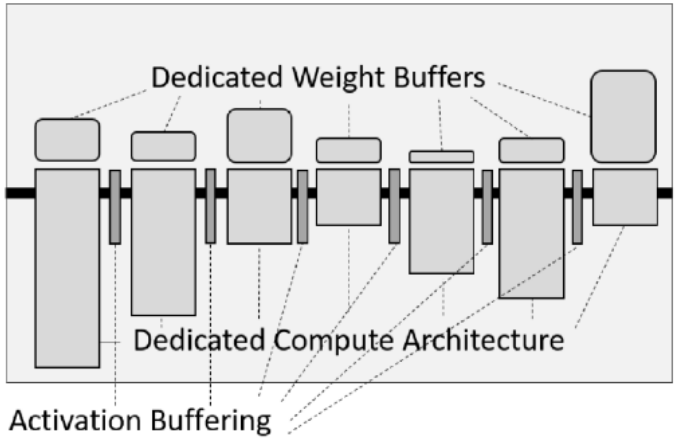
[1] Ahmad Al-Zoubi, Gianluca Martino, Fin Bahnsen, Jun Zhu, Görschwin Fey, Holger Schlarb, CNN Implementation and Analysis on Xilinx Versal ACAP at European XFEL, In International System-On-Chip Conference (SOCC), Belfast, Northern Ireland, 2022.

[2] Gianluca Martino, Görschwin Fey, Runtime Monitoring of c-LTL Specifications on FPGAs using HLS, In International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), Villasimius, Italy, 2022.

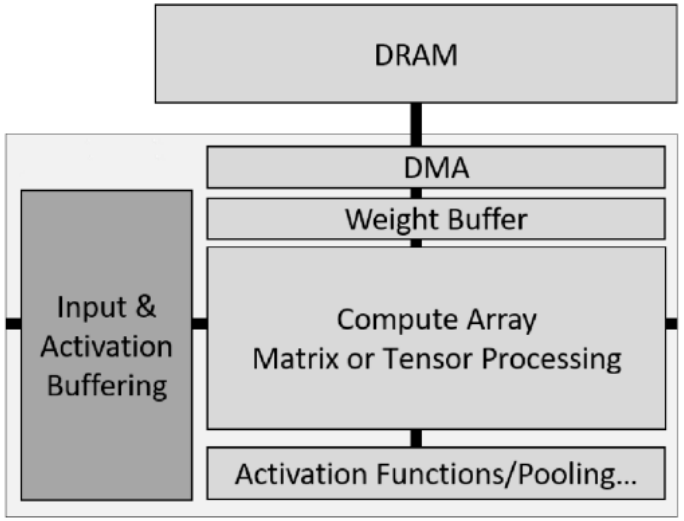
Using HLS for modeling NNs

DPUs vs Spatial Accelerators

Spatial accelerators ⟷ spectrum ⟷ DPU/MPE/SA



- One size does not fit all
  - Generate tailored hardware for a model
    - Few-bit weights and activations
    - Map each layer to HLS description
    - Connect with FIFOs/streams
- Stay on-chip
  - Higher energy efficiency and bandwidth



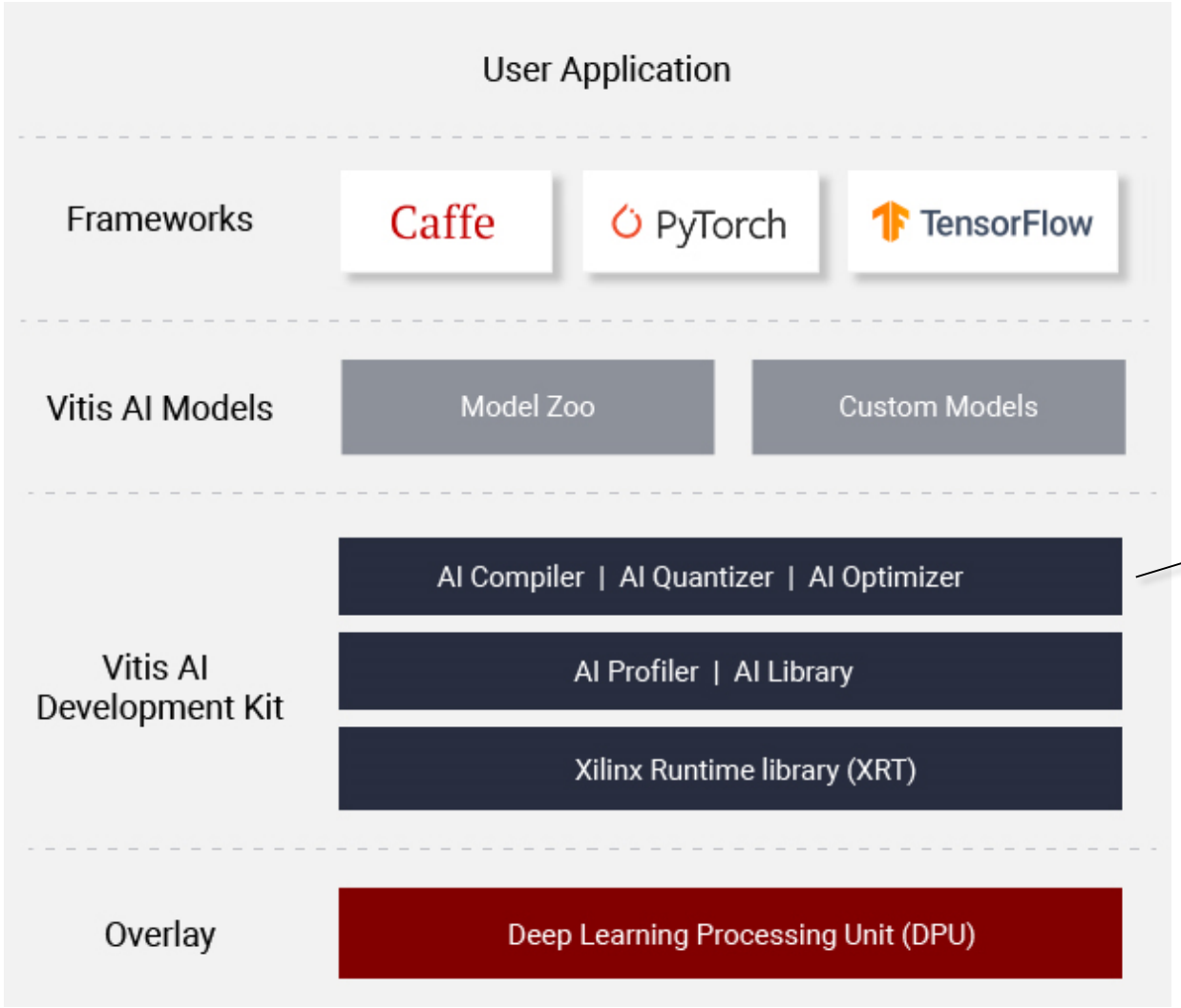
- Matrix of Processing Elements

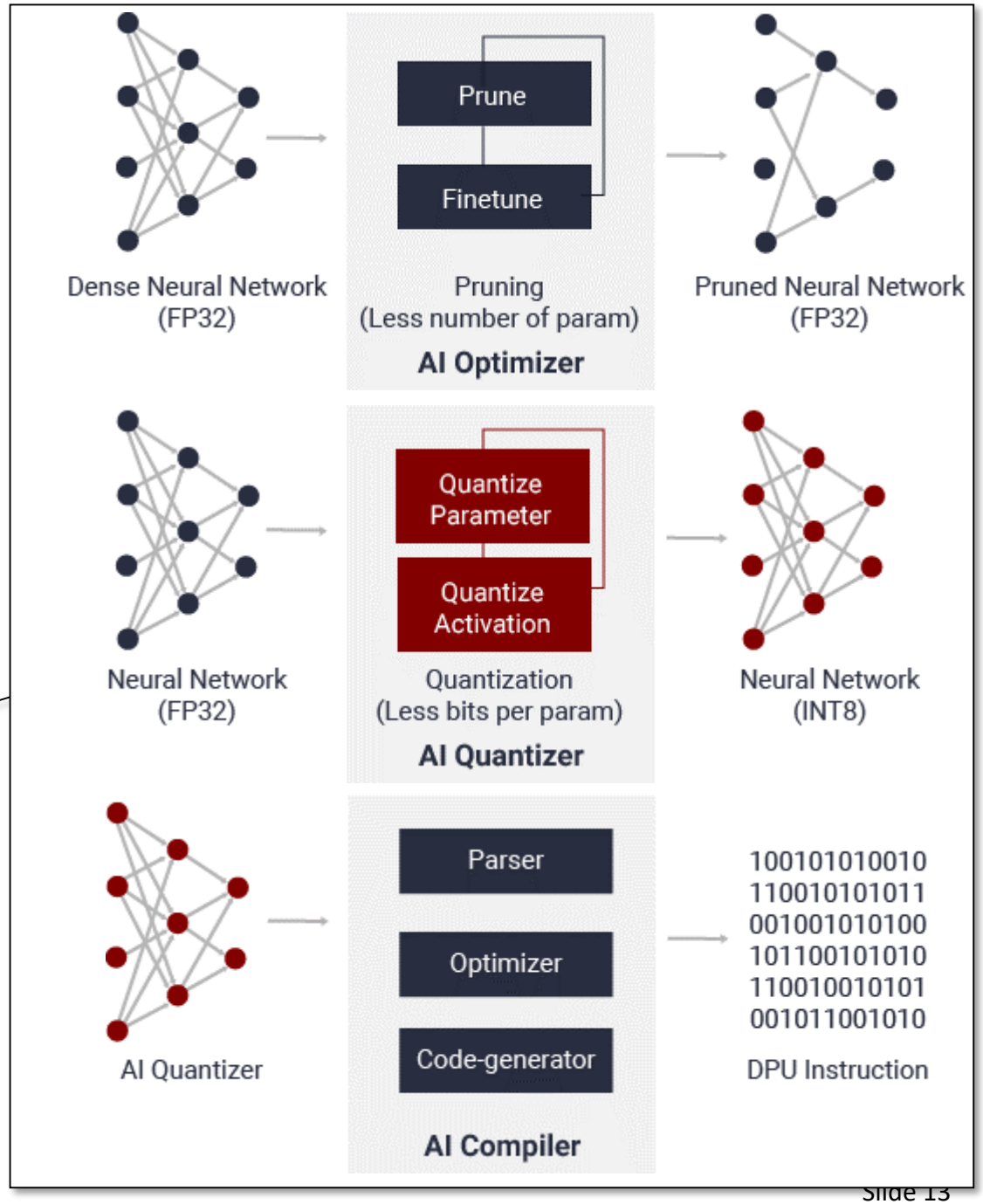
Courtesy of Nhan Tran, Giuseppe Di Guglielmo, and Farah Fahim

[Applications and Techniques for Fast Machine Learning in Science, (2022)]

Using DPUs for modeling NNs

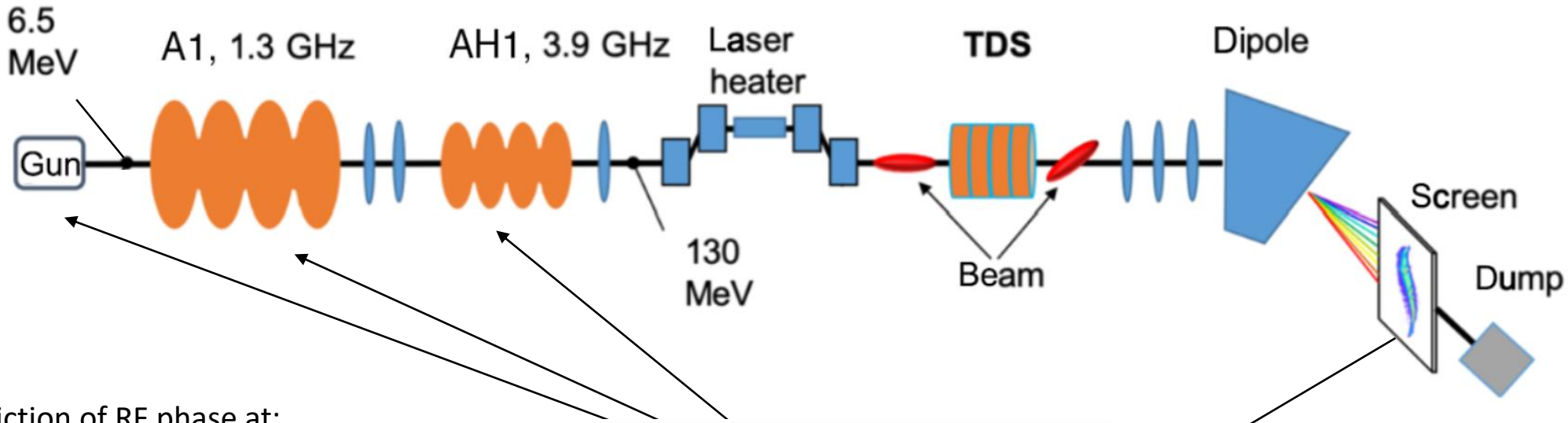
Vitis AI





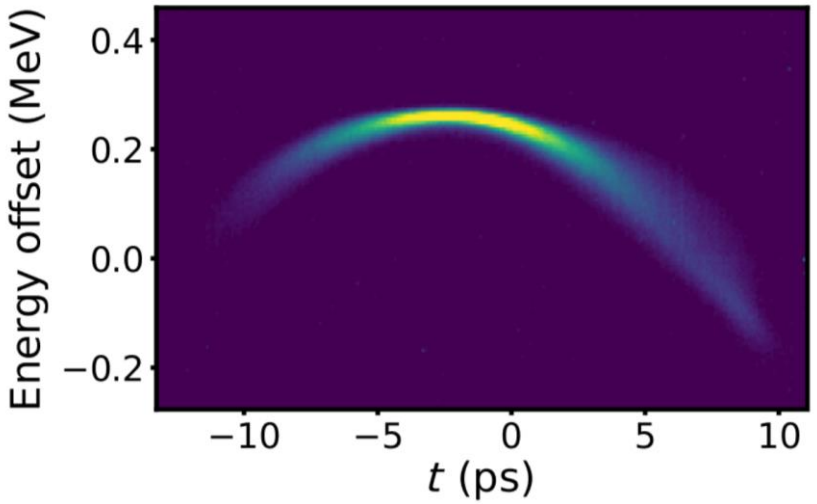
Using DPUs for modeling NNs

Vitis AI



Prediction of RF phase at:

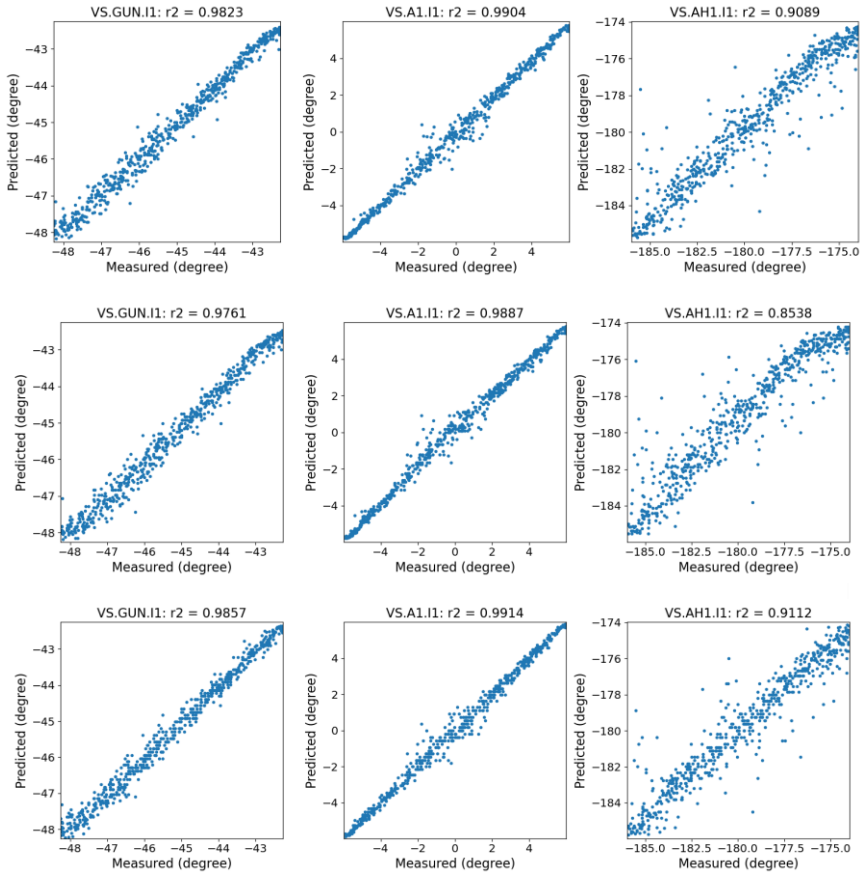
- RF gun
- A1
- AH1



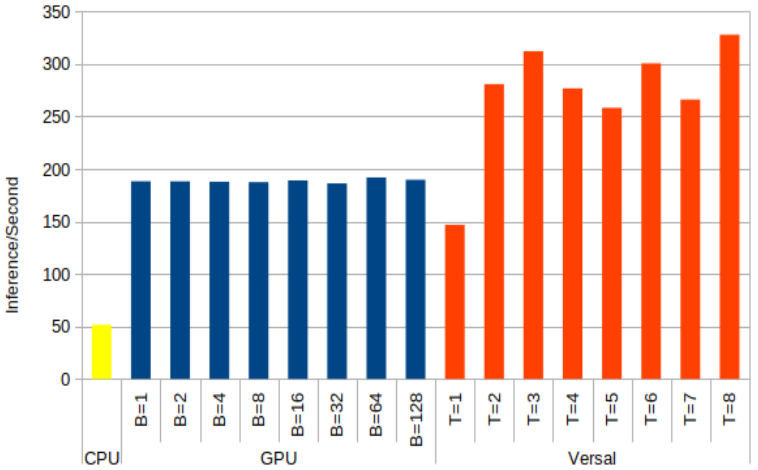
Longitudinal phase-space image at the European XFEL photoinjector

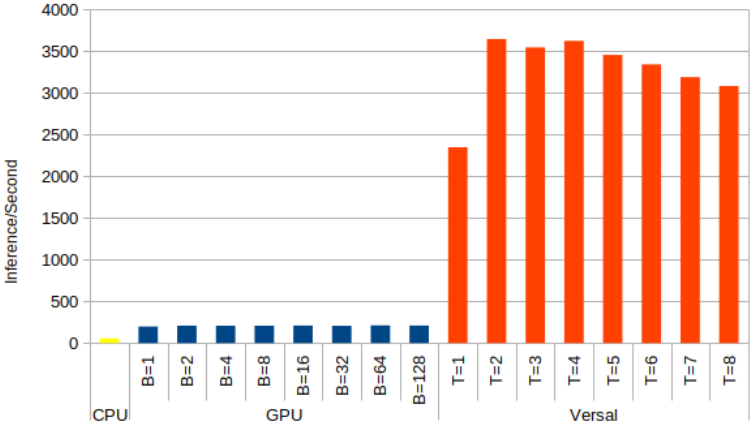
Using DPUs for modeling NNs

Vitis AI



Configurations	Device Latency (ms)			
	CPU	GPU	B4096	C32B3
Partial Support	51.40	55.68	53.04	24.01
Full Support	50.8	54.30	4.60	2.98

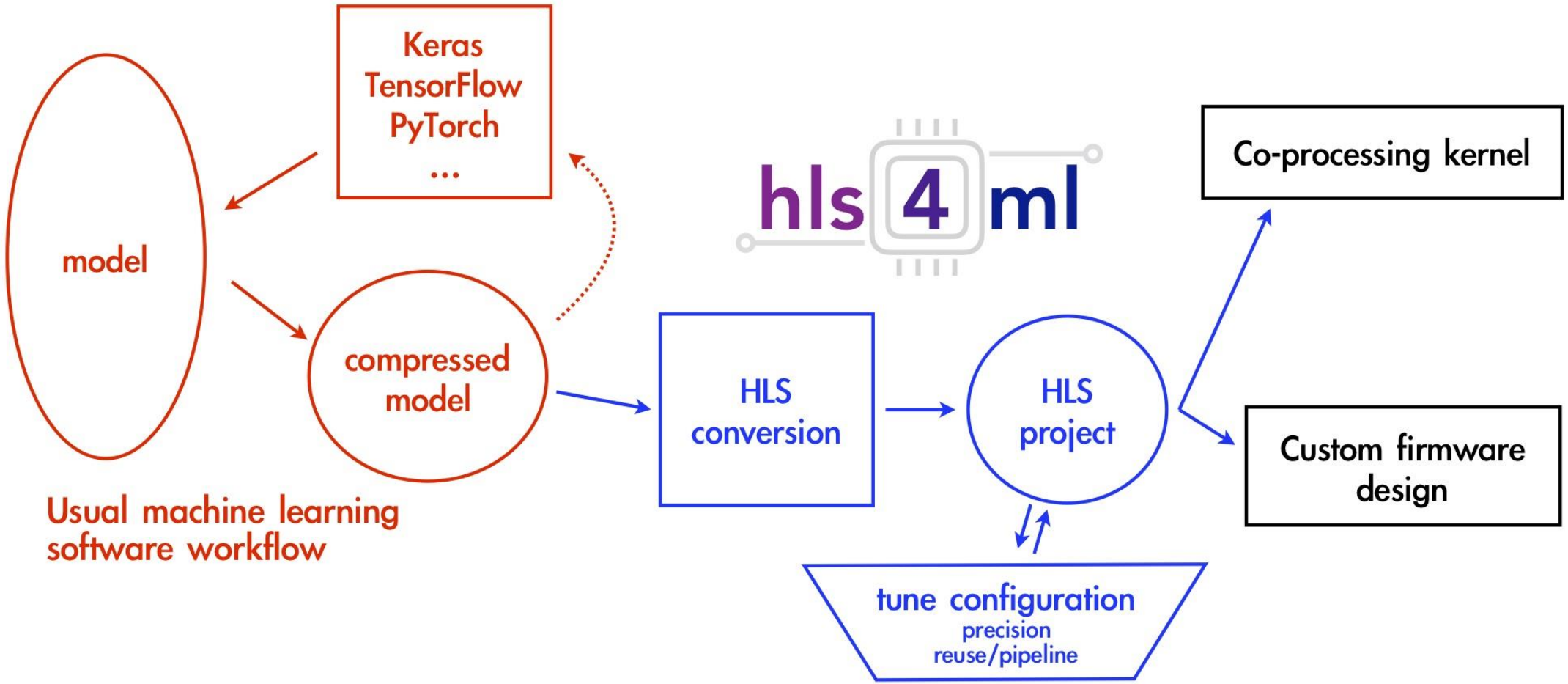




[1] Ahmad Al-Zoubi, Gianluca Martino, Fin Bahnsen, Jun Zhu, Görschwin Fey, Holger Schlarb, CNN Implementation and Analysis on Xilinx Versal ACAP at European XFEL, In International System-On-Chip Conference (SOCC), Belfast, Northern Ireland, 2022.

Using HLS for modeling NNs

hls4ml



[1] Duarte, J., Han, S., Harris, P., Jindariani, S., Kreinar, E., Kreis, B., Ngadiuba, J., Pierini, M., Rivera, R., Tran, N. and Wu, Z., 2018. Fast inference of deep neural networks in FPGAs for particle physics. *Journal of Instrumentation*, *13*(07), p.P07027.

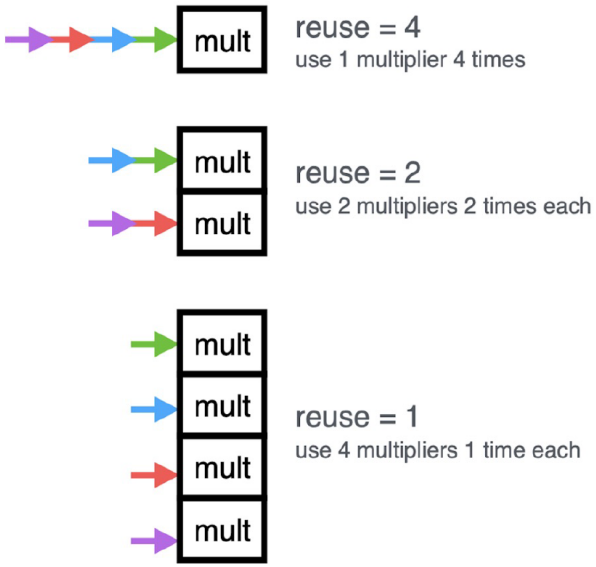
Using HLS for modeling NNs

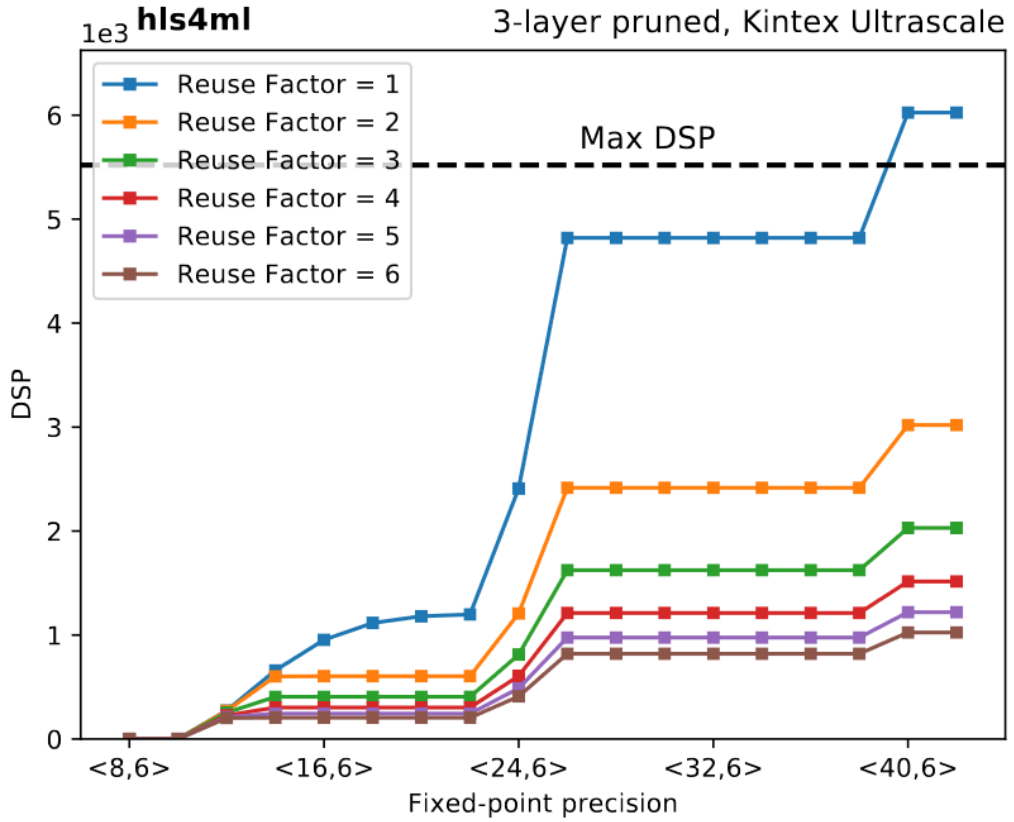
hls4ml

hls4ml uses a set of configurable HLS implementations for each layer that composes a supported model.

The main characteristics are:

- Configurable reuse factor





Using HLS for modeling NNs

hls4ml

hls4ml uses a set of configurable HLS implementations for each layer that composes a supported model.

The main characteristics are:

- Configurable reuse factor
- Configurable precision

config.yml

```
HLSConfig:
  Model:
    Precision: ap_fixed<16,6>
    ReuseFactor: 1
  LayerType:
    Dense:
      Precision: ap_fixed<14,5>
```

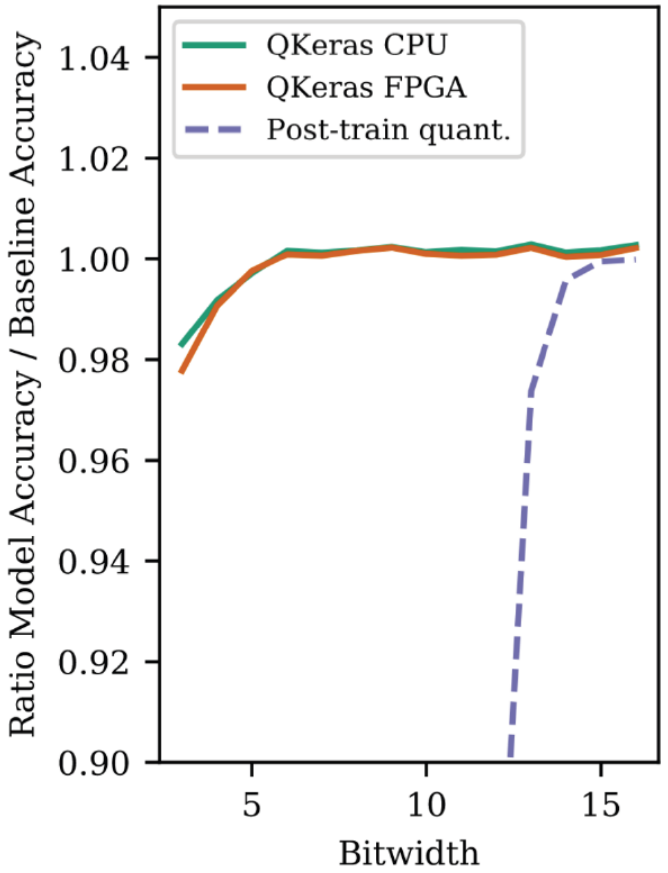
Using HLS for modeling NNs

hls4ml

hls4ml uses a set of configurable HLS implementations for each layer that composes a supported model.

The main characteristics are:

- Configurable reuse factor
- Configurable precision
- Support for pre-quantized networks (using QKeras)



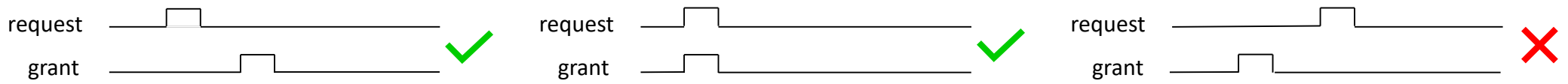
Using HLS for anomaly detection

c-LTL property checkers generation using HLS

Counting semantics for the Linear Temporal Logic (c-LTL) [1] properties express behaviors of a running system.

$$r \rightarrow \mathbf{F}(g)$$

- $\varphi_1 \rightarrow \varphi_2$: φ_1 implies φ_2
- **F (φ)** : φ will become true eventually (immediately or in the future)

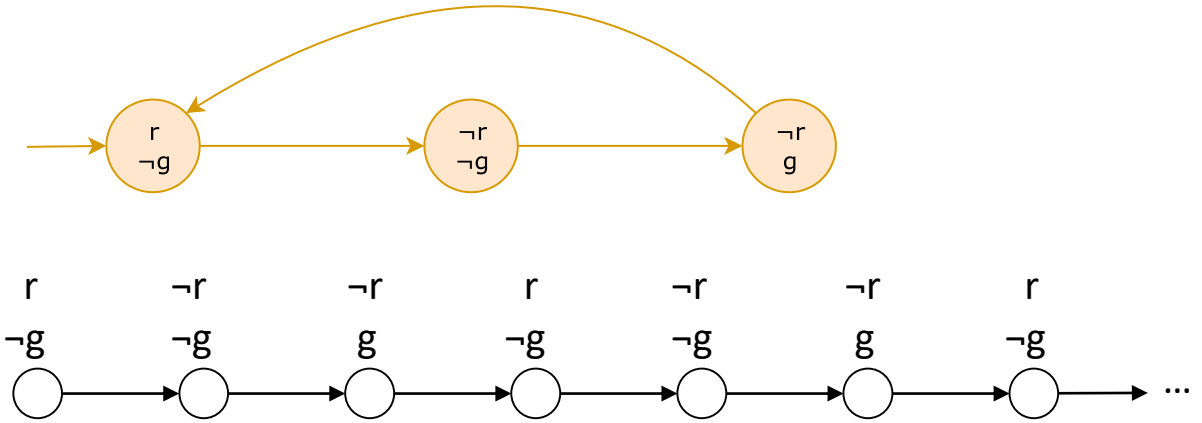


[1] Bartocci, E., Bloem, R., Nickovic, D. and Röck, F., 2018, July. A counting semantics for monitoring LTL specifications over finite traces. In *International Conference on Computer Aided Verification* (pp. 547-564). Springer, Cham.

Using HLS for anomaly detection

c-LTL property checkers generation using HLS

Counting semantics for the Linear Temporal Logic (c-LTL) properties express behaviors of a running system.

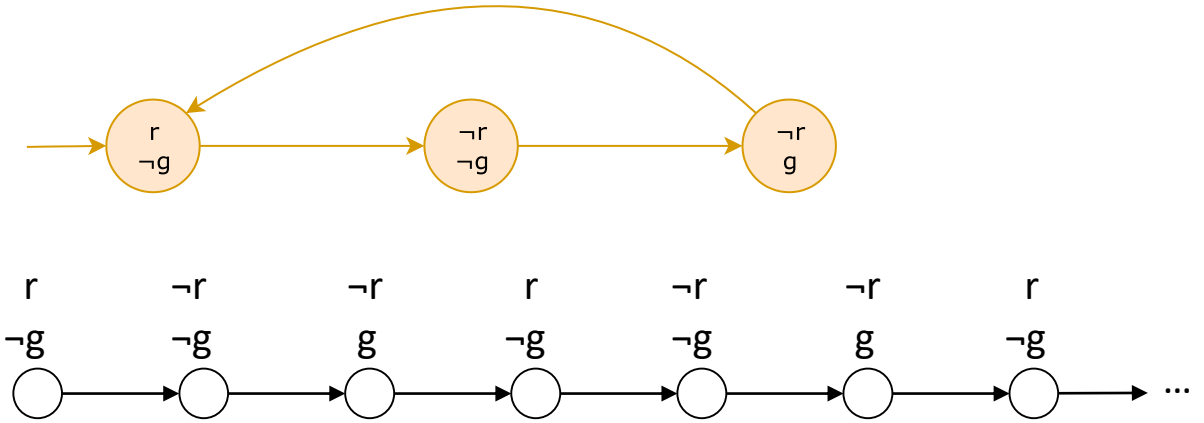


r	T	-	-	T	-	-	T	
g	-	-	T	-	-	T	-	

Using HLS for anomaly detection

c-LTL property checkers generation using HLS

Counting semantics for the Linear Temporal Logic (c-LTL) properties express behaviors of a running system.



r	T	-	-	T	-	-	T	
g	-	-	T	-	-	T	-	

Using HLS for anomaly detection

c-LTL property checkers generation using HLS

Counting semantics for the Linear Temporal Logic (c-LTL) properties express behaviors of a running system.

Trace

r	T	-	-	T	-	-	T	
g	-	-	T	-	-	T	-	

Evaluation

	1	2	3	4	5	6	7	EOT
g	⊥	⊥	T	⊥	⊥	T	⊥	?
r → **F** g	?	?	T	T_p	T_p	T	T_p	T_p ← presumably

The generated monitors use the counting semantics to predict the next state.

↓

It is possible to raise early warnings about immediate future failures of the system.

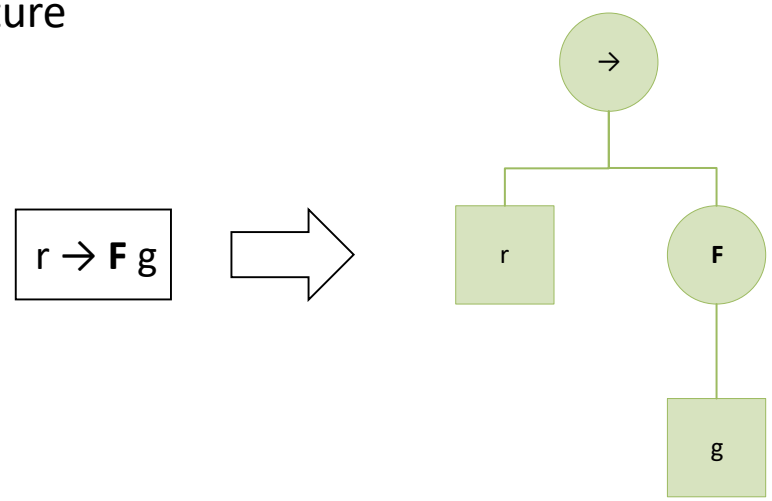
Using HLS for anomaly detection

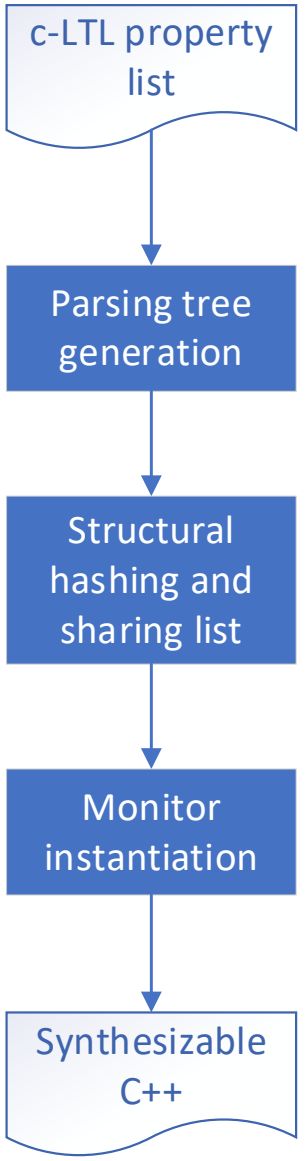
c-LTL property checkers generation using HLS

The c-LTL framework [1] generates synthesizable C++ code starting from LTL properties.

The creation of a c-LTL runtime monitor requires the following steps:

- The parsing tree generation step creates a structure for each c-LTL property





[1] Gianluca Martino, Görschwin Fey, Runtime Monitoring of c-LTL Specifications on FPGAs using HLS, In International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), Villasimius, Italy, 2022.

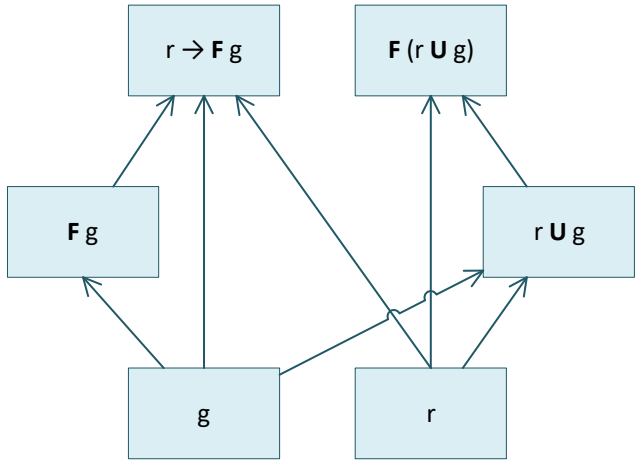
Using HLS for anomaly detection

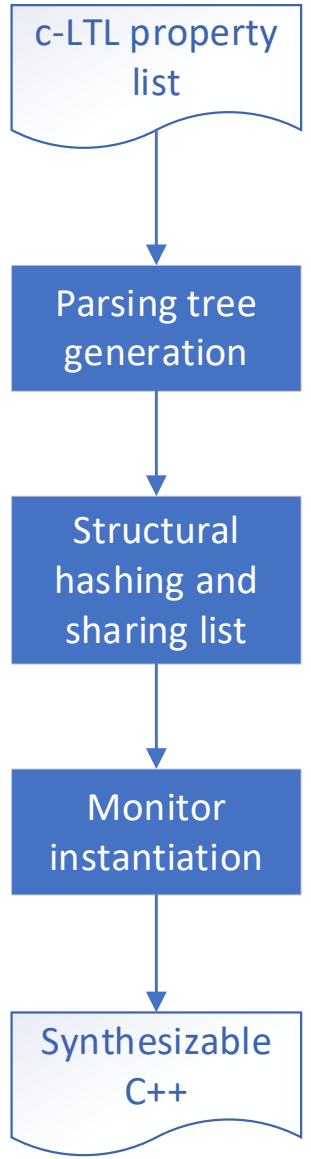
c-LTL property checkers generation using HLS

The c-LTL framework generates synthesizable C++ code starting from LTL properties.

The creation of a c-LTL runtime monitor requires the following steps:

- The parsing tree generation step creates a structure for each c-LTL property
- The structural hashing and sharing list creation step enables the sharing of common parts among different properties





[1] Gianluca Martino, Görschwin Fey, Runtime Monitoring of c-LTL Specifications on FPGAs using HLS, In International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), Villasimius, Italy, 2022.

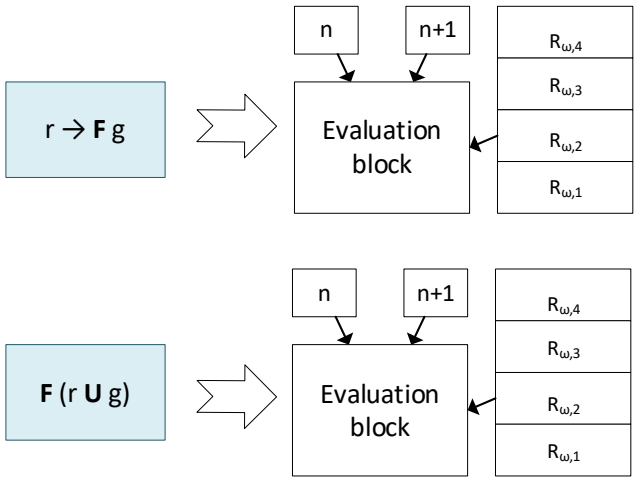
Using HLS for anomaly detection

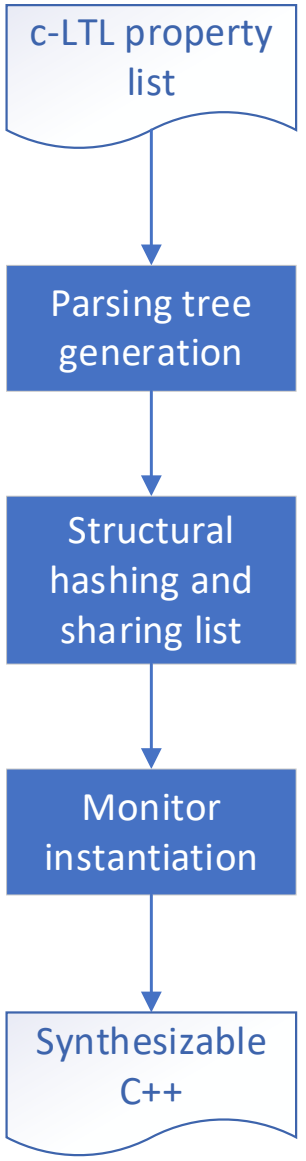
c-LTL property checkers generation using HLS

The c-LTL framework generates synthesizable C++ code starting from LTL properties.

The creation of a c-LTL runtime monitor requires the following steps:

- The parsing tree generation step creates a structure for each c-LTL property
- The structural hashing and sharing list creation step enables the sharing of common parts among different properties
- The monitor instantiation generates the optimized synthesizable C++ code





[1] Gianluca Martino, Görschwin Fey, Runtime Monitoring of c-LTL Specifications on FPGAs using HLS, In International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), Villasimius, Italy, 2022.

Conclusion

Current and future work

- There is a strong push for the acceleration of machine-learning algorithms

  ↓

  Today it is possible to automate the conversion from software to hardware implementation

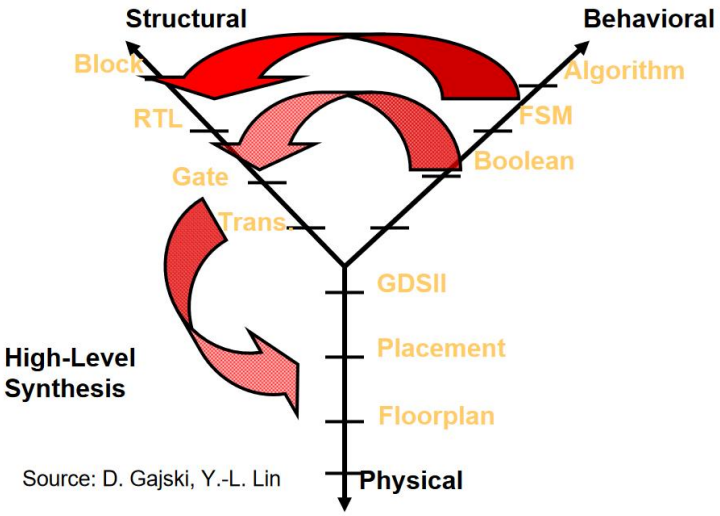
- Results show good accuracy despite quantization

- Acceleration of machine learning algorithms NOT based on neural networks requires additional research effort

  ↓

  Thanks to HLS, it is simpler to generate generic synthesizable code

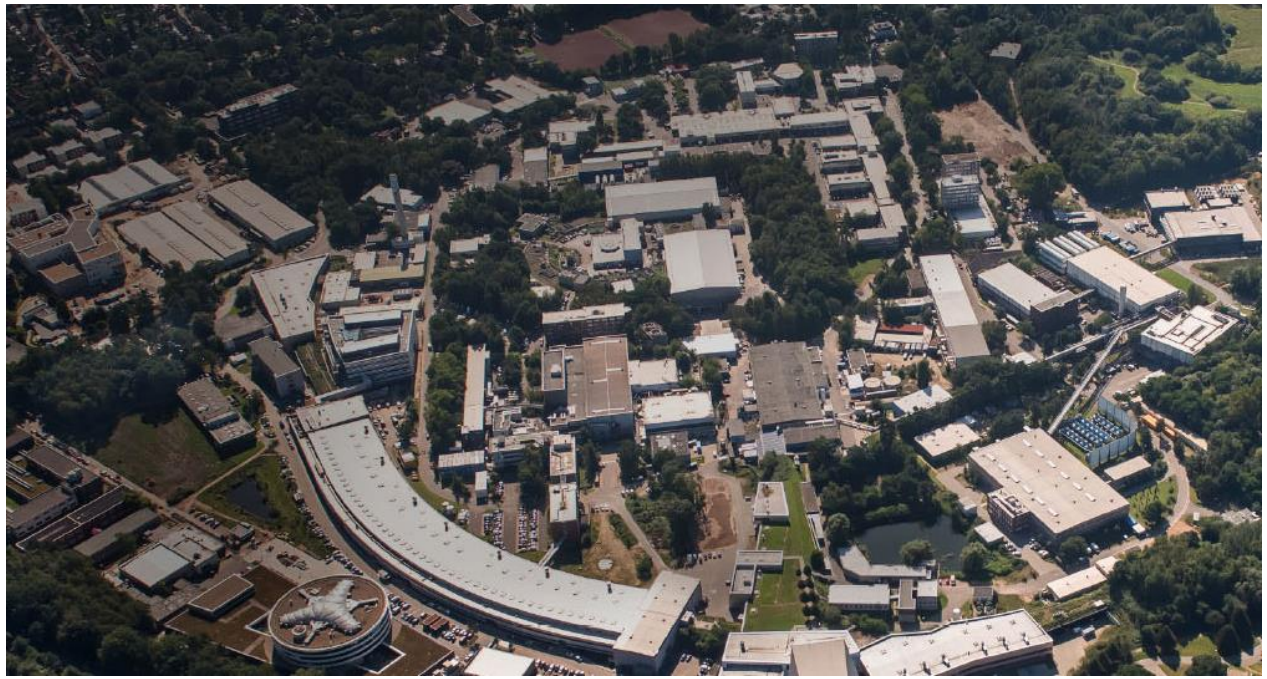
- Next steps:
  - Evaluation of **real-time monitoring** schemes based on HLS implementations
  - Framework for the **drop-in** replacement of standard **anomaly detection libraries**



Thanks for your attention!

Gianluca Martino
MSK, DESY
gianluca.martino@desy.de
+49 (0)40 8998 1718

Gianluca Martino, Ahmad Al Zoubi, Julien Branlard, Holger Schlarb, Goerschwin Fey



TUHH Hamburg University of Technology

DESY.

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

DASHH

CDCS
CENTER FOR DATA AND COMPUTING IN NATURAL SCIENCES