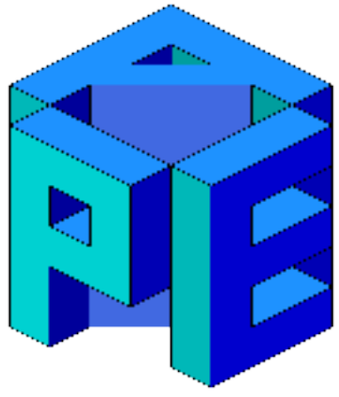
Innovative Workflows Astro & Particle Physics (IWAPP)

APEIRON

Abstract **P**rocessing **E**nvironment for **I**ntelligent **R**eadout systems based **O**n **N**eural networks

Neural Network on FPGA with High Level Synthesis



Matteo Turisini - APE group - INFN Roma

2021 March 9th

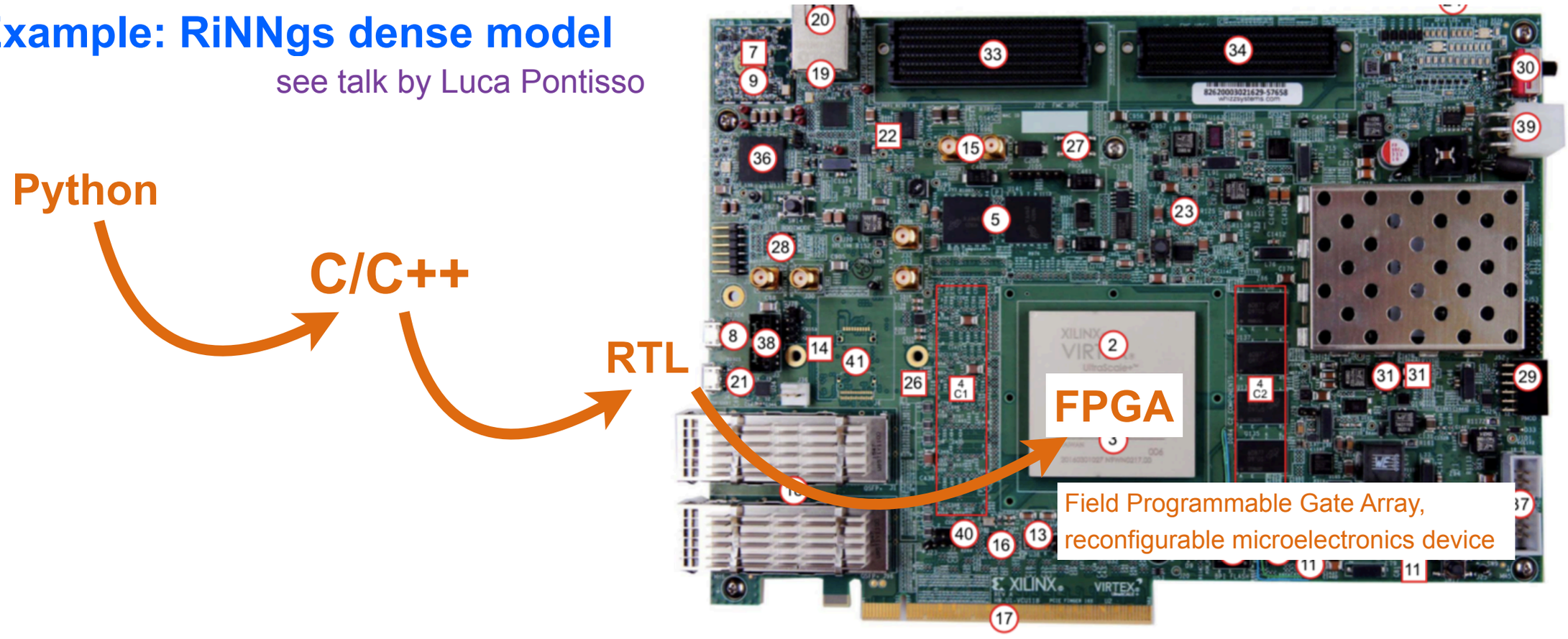
Hardware “programming”

Agenda:

1.Introduction to High Level Synthesis (HLS)

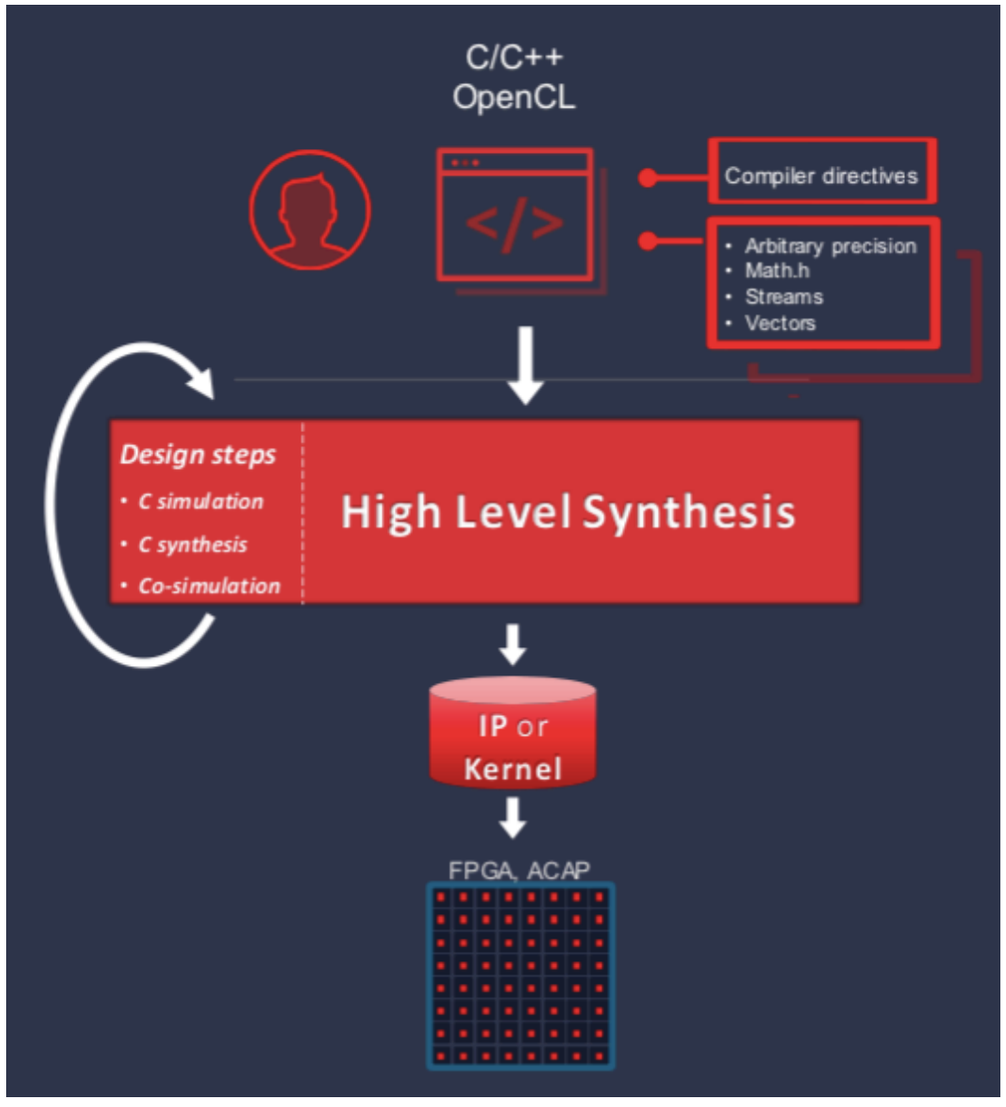
2.Example: RiNNgs dense model

see talk by Luca Pontisso



HLS is an FPGA design methodology convenient to describe complex function difficult or unpractical to write with traditional hardware flow, like Neural Network (NN)

Write software, think hardware



HLS flow representation by Xilinx

HLS: C based design entry

High level instructions transposed to hardware components
e..g. an array becomes an on-chip RAM
Code annotations (pragmas) can “guide" the compiler
e.g. array partition to access stored data in parallel

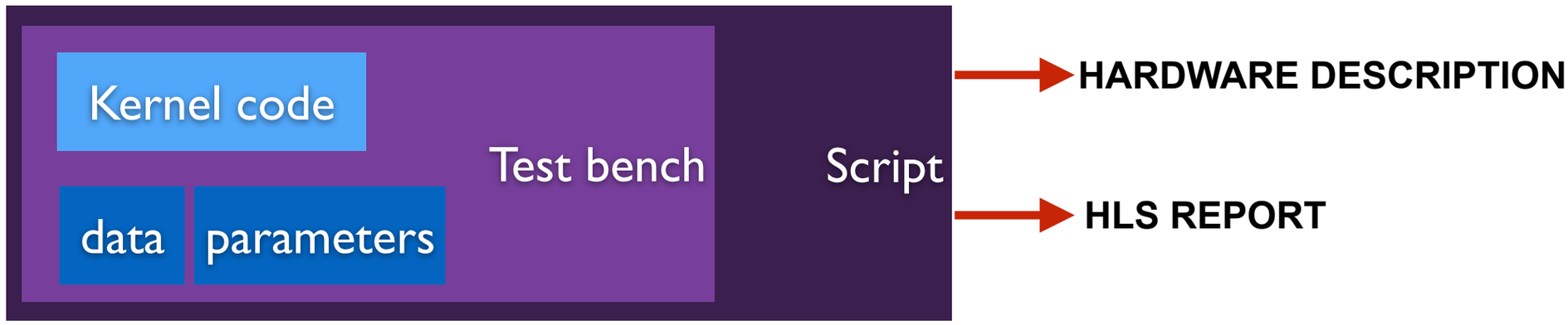
- Build tailored/custom processors
- Fast verification of the algorithm
- Detailed report about generated digital circuit
- Pragmas determines circuit topology

Our workflow for NN in HLS

1.Bottom-up — Fully connected, convolutional, activations, precision

2.Design Space Exploration — Size, random data and weights, easy/corner cases

3.Automation — Unassisted operations (e.g. scans)



NN model/architecture

Custom C/C++ code for a complete control over implementation

Automatic conversion using **HLS4ML*** library

* https://github.com/fastmachinelearning/hls4ml

Example: inference on FPGA with HLS

RiNNgs

INPUT

hit_list[64]

OUTPUT

0,1,2,3+ rings

in the next slides HLS results with two different algorithms for the same NN:

HLS4ML and **custom**

HLS REPORT

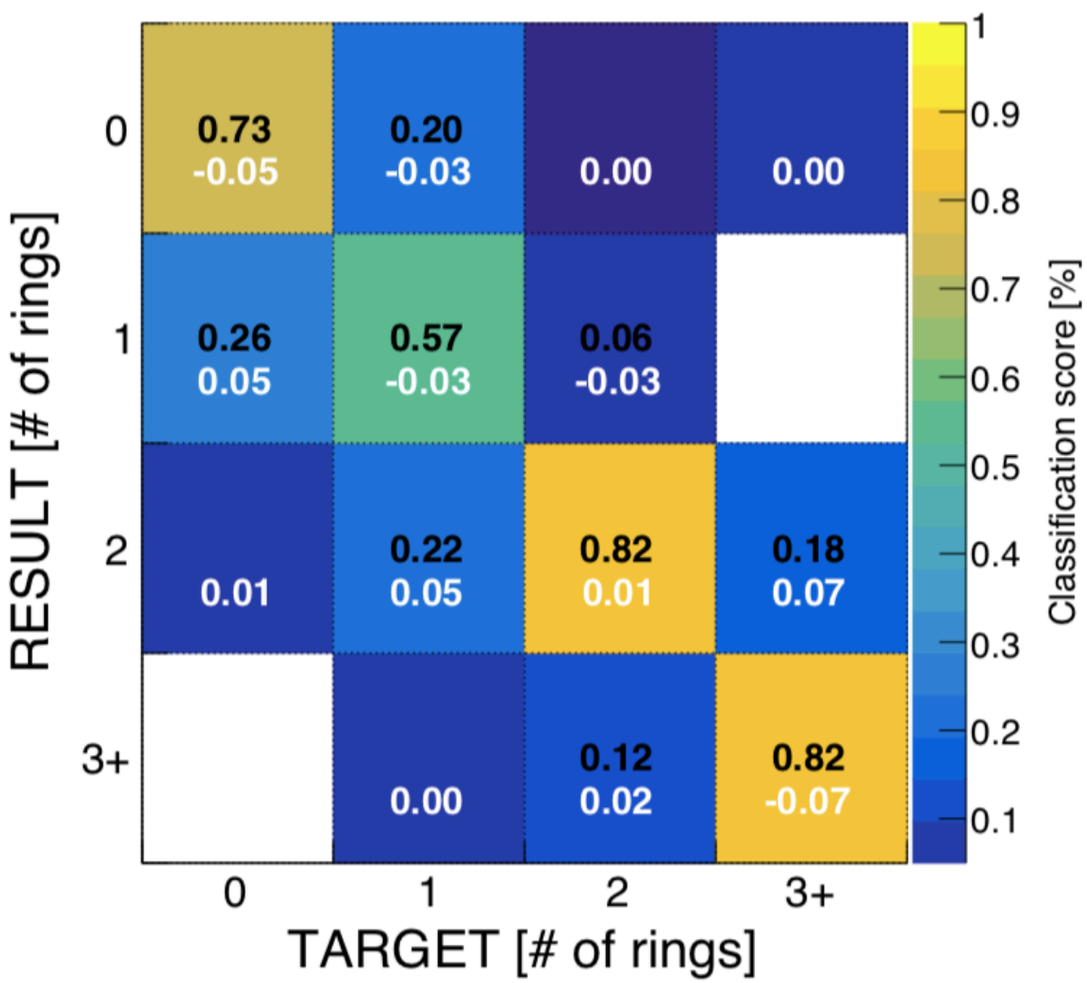
- Accuracy
- Fixed precision
- Resources
- Timing

FC - Fully Connected

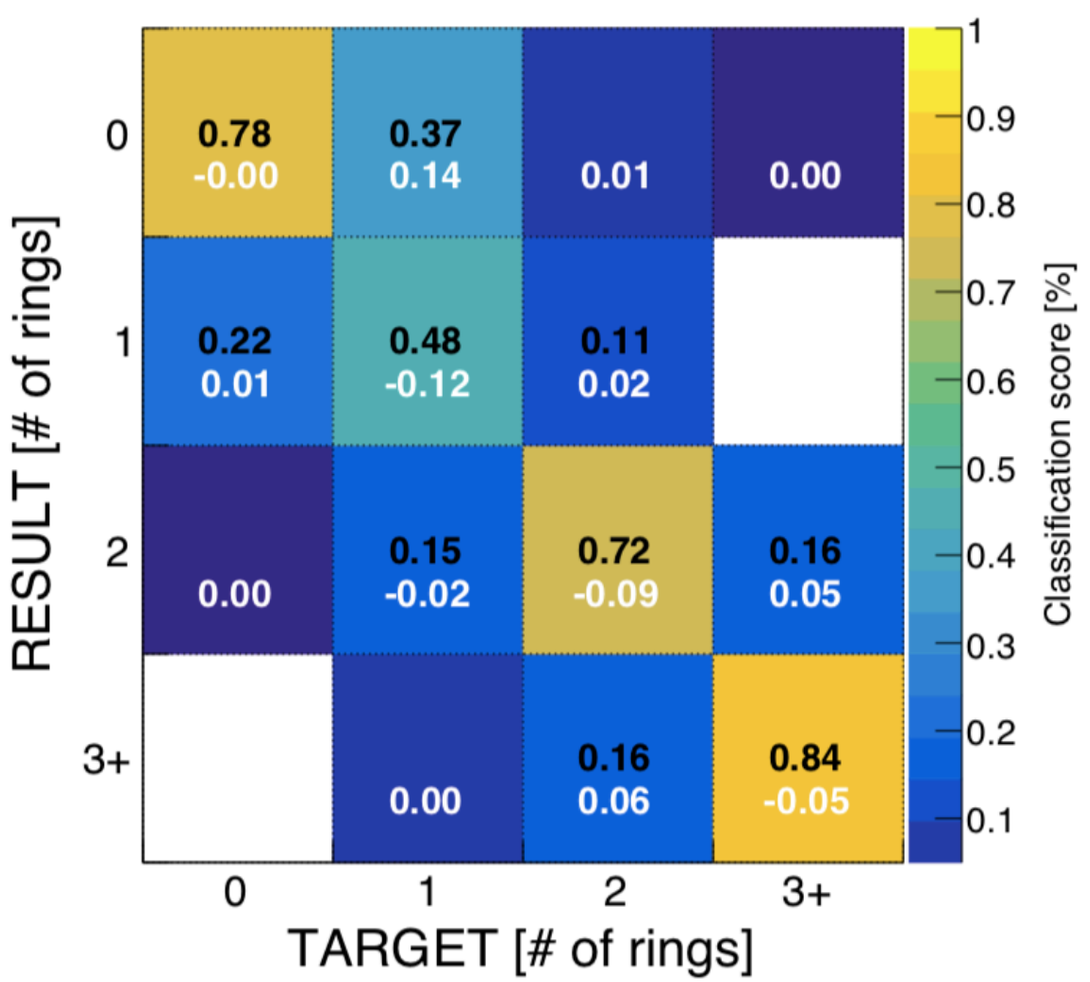
Accuracy verification

Plots representing NN confusion matrix of the hardware implementation obtained with HLS
Deviation from Tensorflow in white text

HLS4ML



custom

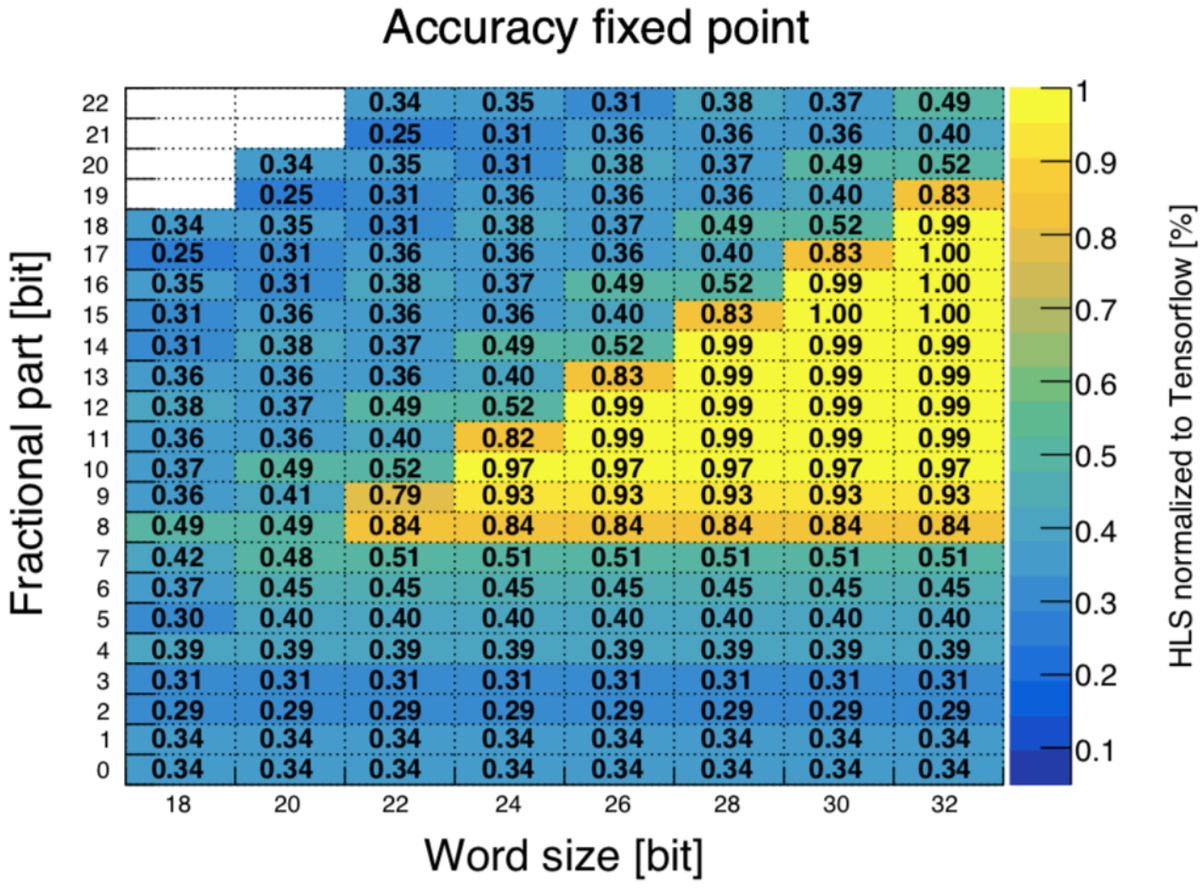


Model features reproduced nicely with fixed precision

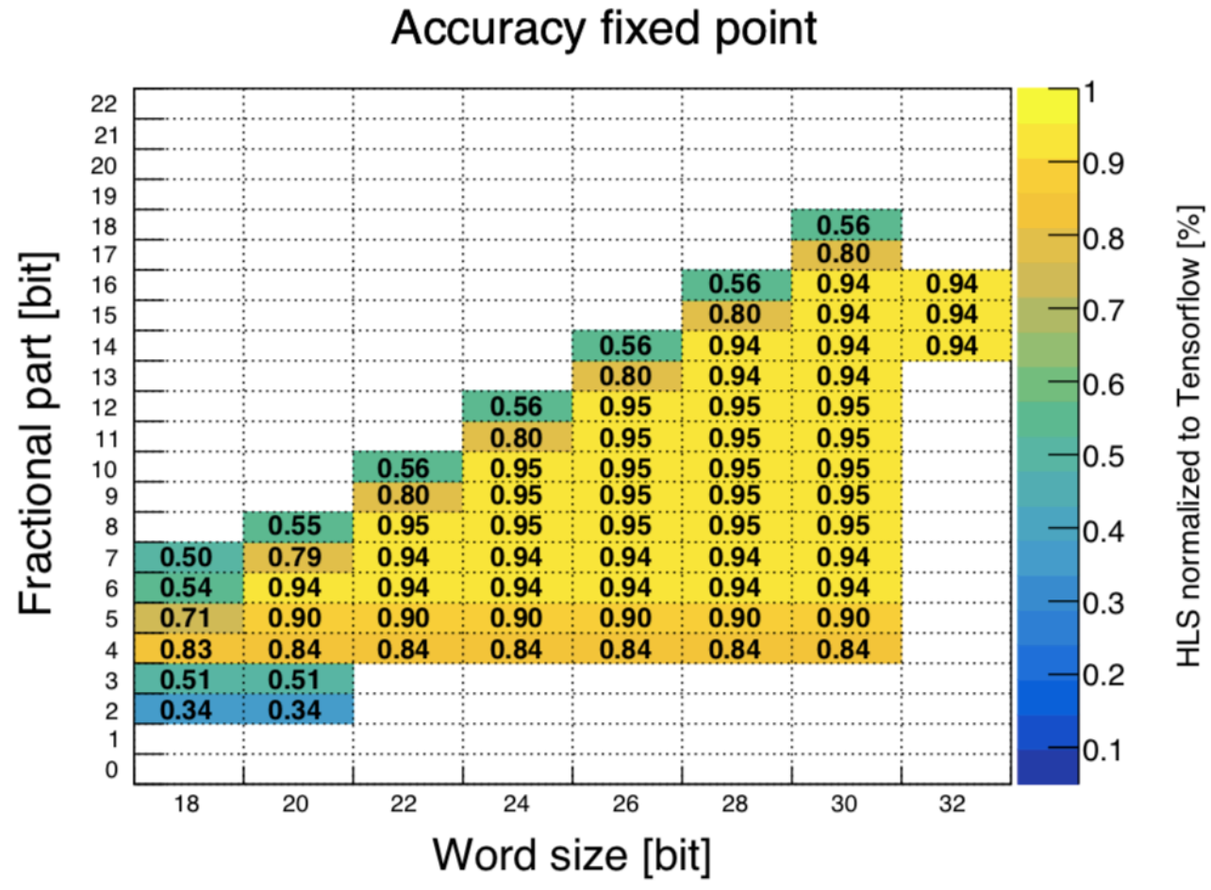
Playing with numerical precision

Reduced precision arithmetics (fixed point on FPGA) instead of floating point (in Tensorflow)
Each point in the plane represents a combination of bit-width and fractional part
Accuracy normalized to TensorFlow (74%)

HLS4ML



custom



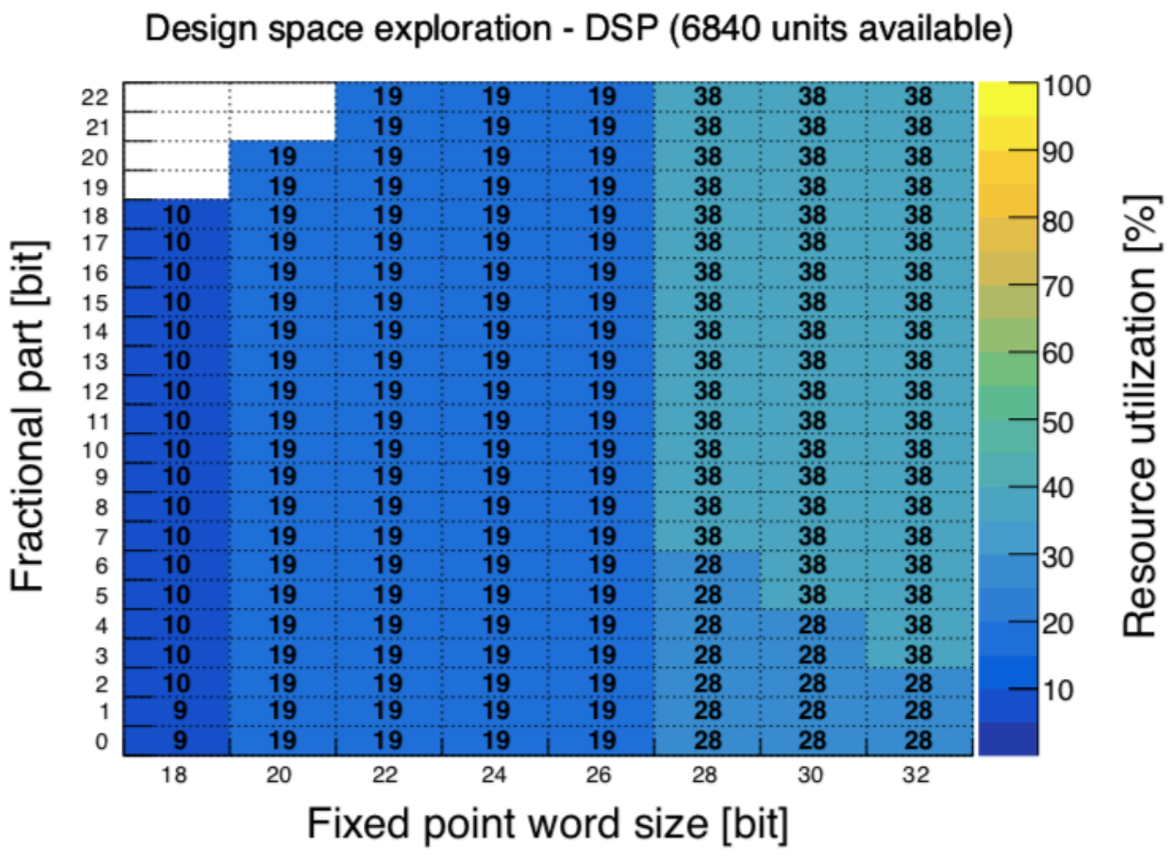
The more brilliant yellow the better (i.e. the closer to tensorflow model performance)

Different code, different hardware

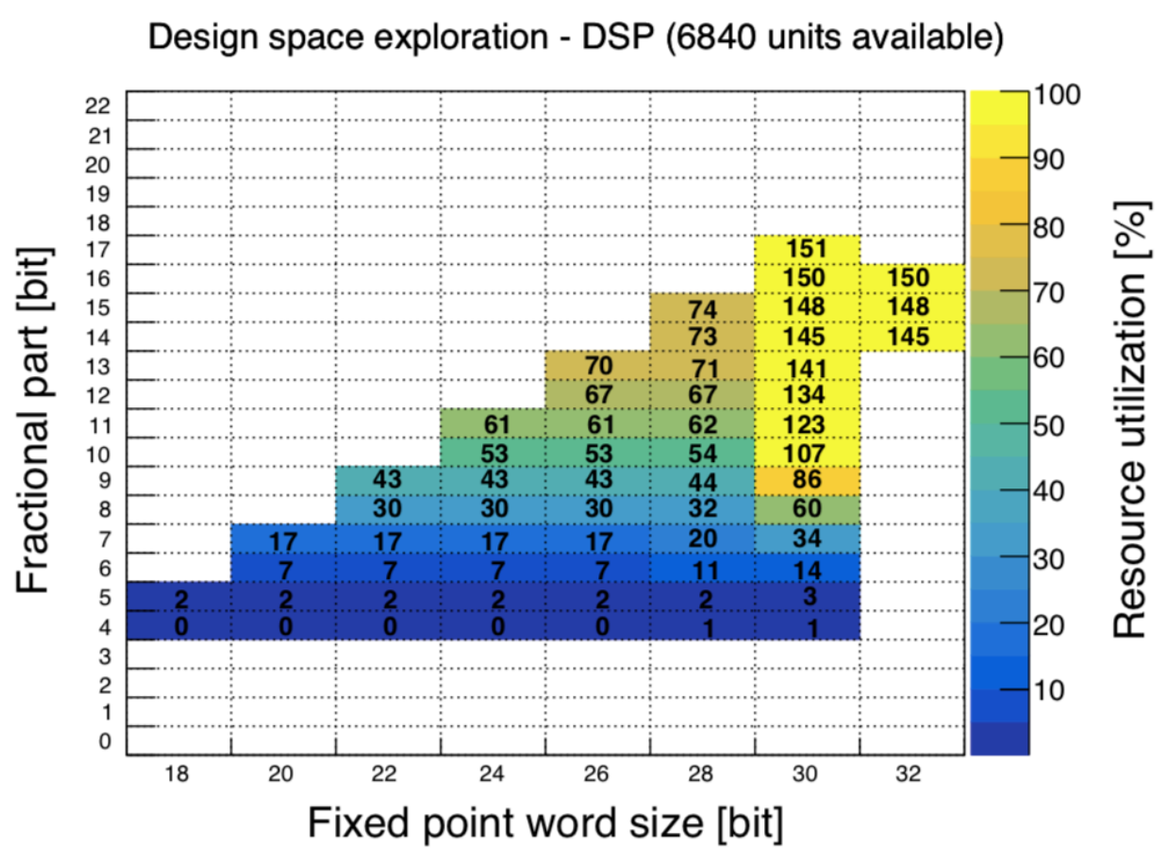
DSP digital signal processors, hardware units for arithmetics (e.g. multiply and accumulate)

Resources expressed in % of available in Xilinx VCU118

HLS4ML



custom



The more dark blue the better (i.e. smaller circuit area and faster design)

other resources report on backup slides

Timing Performance

HLS timing report:
(of the digital circuit)

- operating conditions (clock period and frequency)
- time delay to produce the output (latency)
- minimum time interval between inputs (throughput)

	Clock [ns]	Frequency [MHz]	Latency [clock ticks]	Interval [clock ticks]
HLS4ML (reuse 8)	>7.1	<140	18-25	11
CUSTOM	>2.5	<400	50-160	4

<< 1 ms

< 100 ns

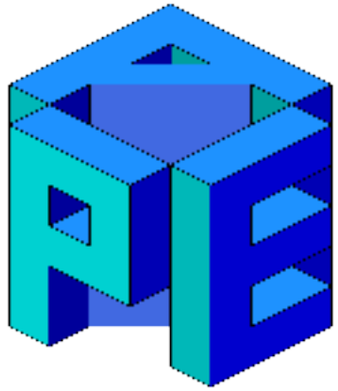
Both designs satisfy NA62 requirements

Summary and outlook

- **HLS** is a fast C-based FPGA design methodology
- Can be used for **NN** but requires hardware “aware " programming skills
- RiNNgs dense model implementation fulfills NA62 **online** requirements
- Next steps are on board integration and convolutional model development

APEIRON

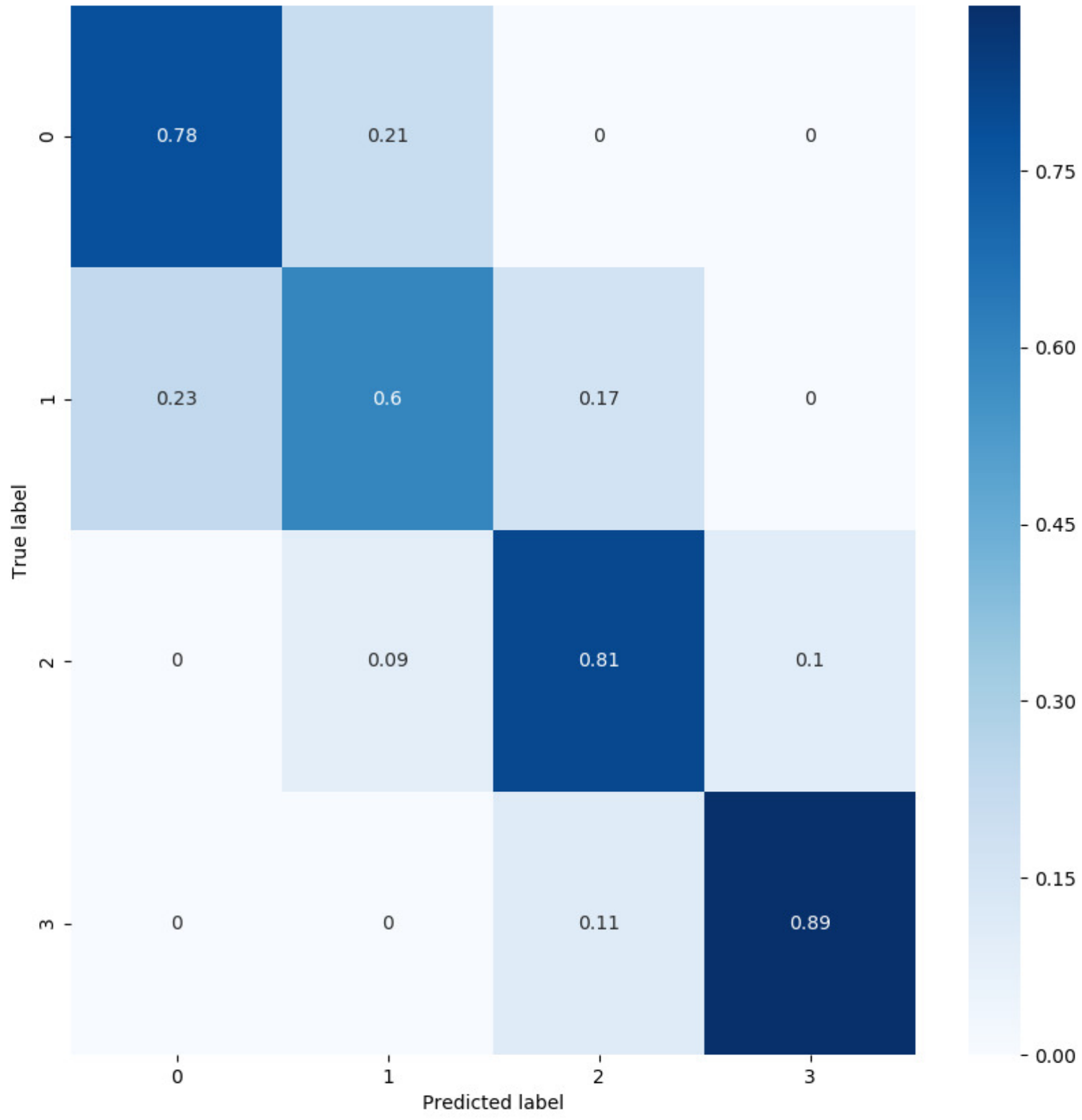
Abstract **P**rocessing **E**nvironment for **I**ntelligent **R**eadout systems based **O**n **N**eural networks



more info at https://apegate.roma1.infn.it

Back up 0

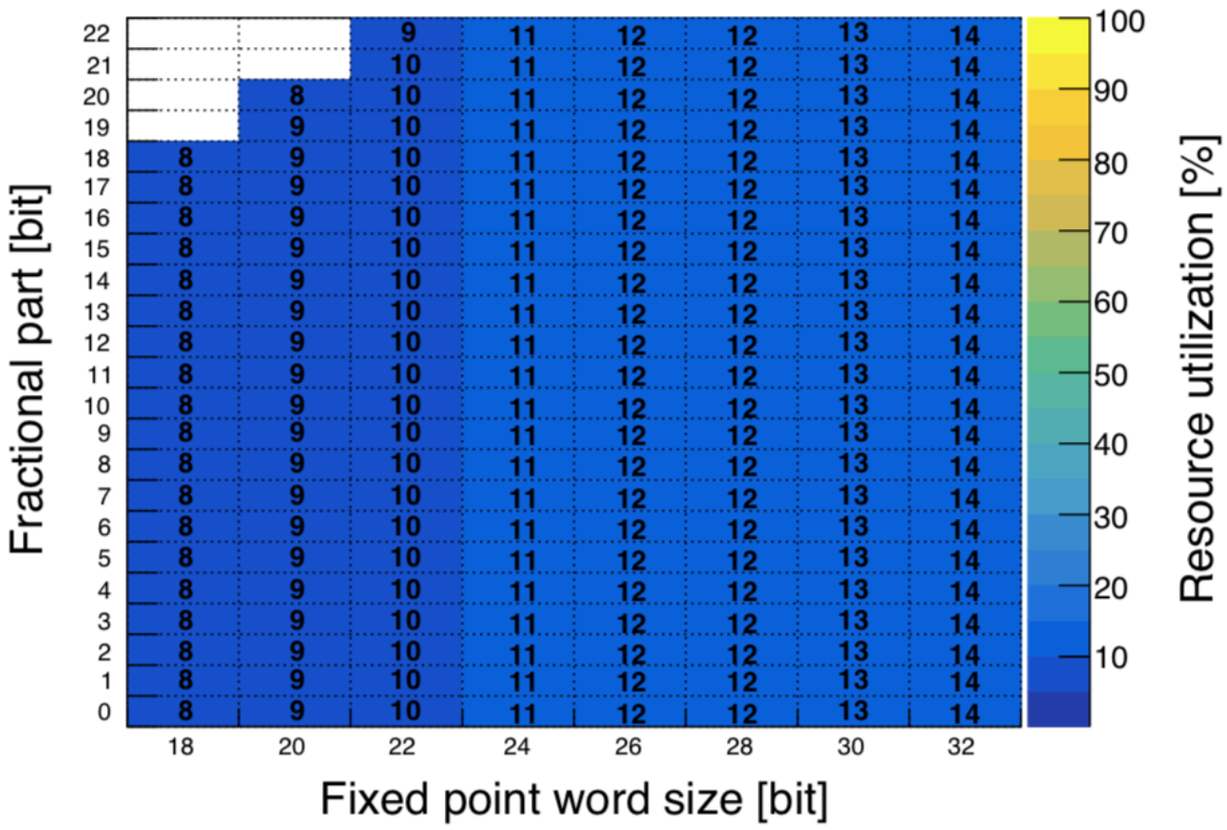
RiNNgs dense model confusion matrix from Tensorflow - Keras



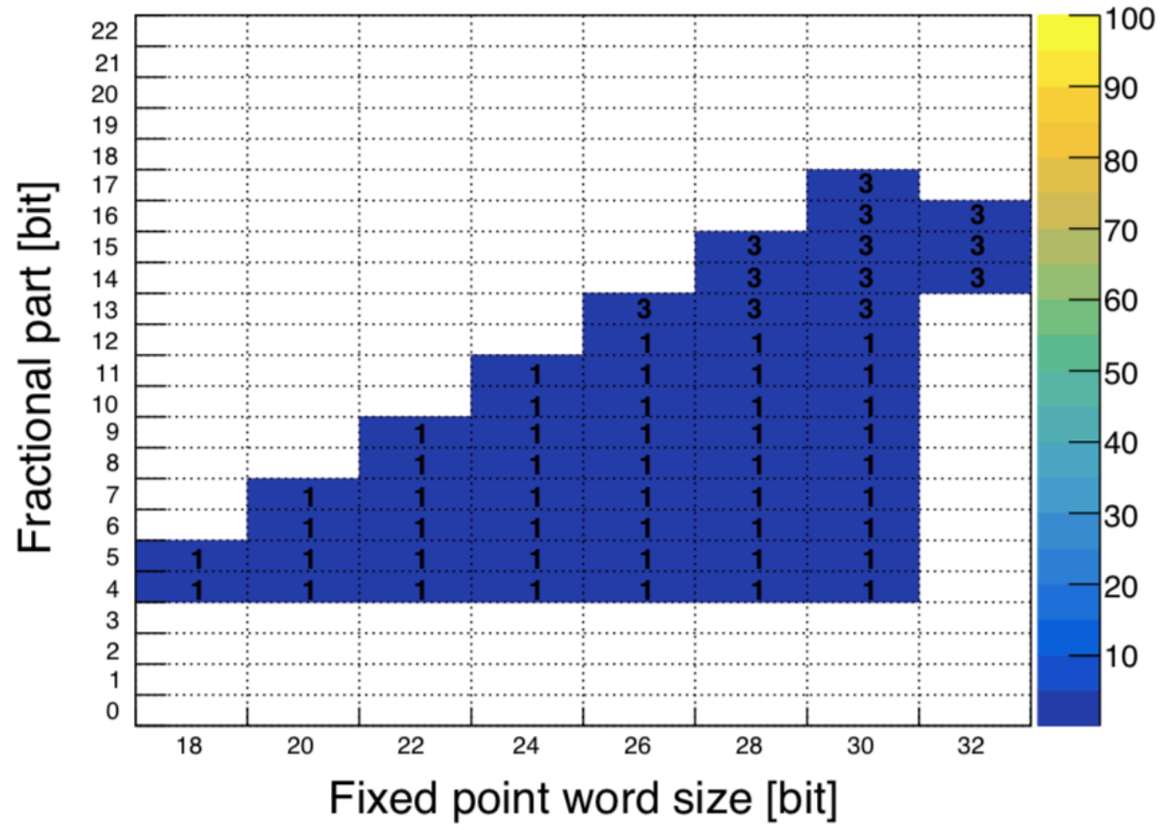
Back up 1

BRAM - Random Access Memory (on chip)

HLS4ML



custom



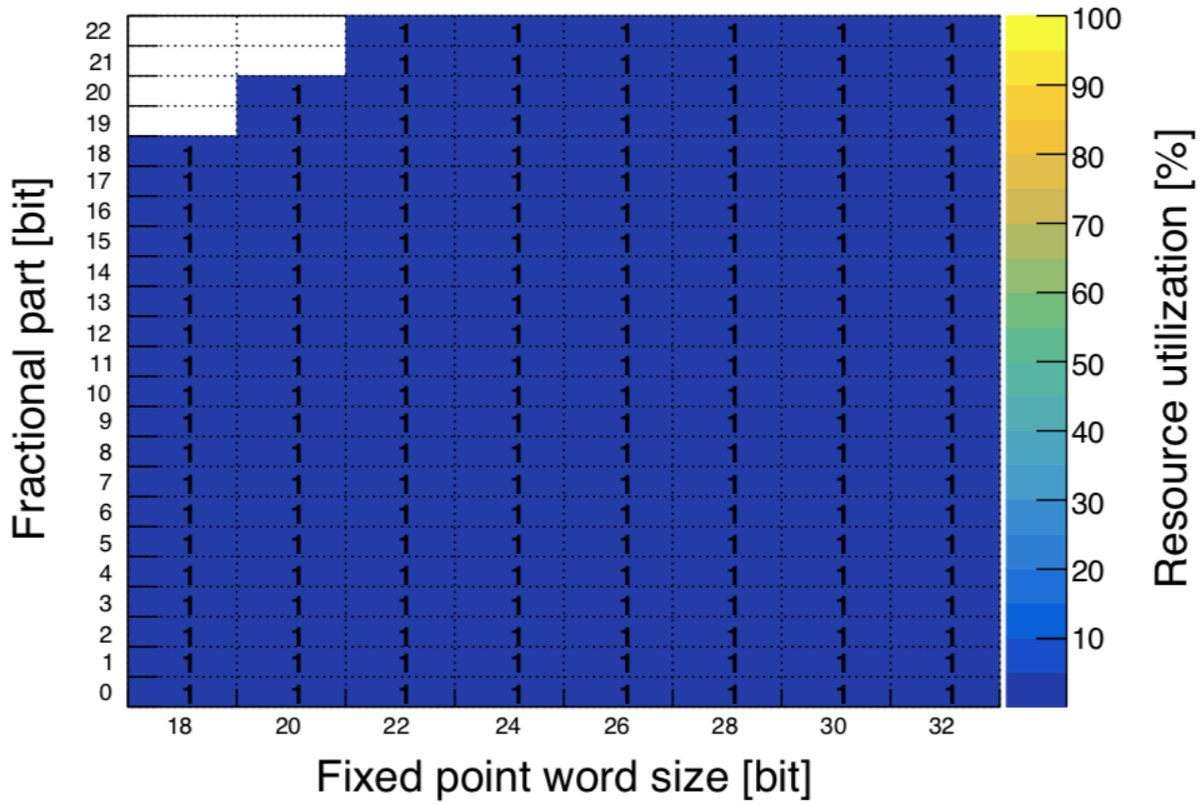
Resources expressed in % of available in Xilinx VCU118

not shown if accuracy < 50% (custom only)

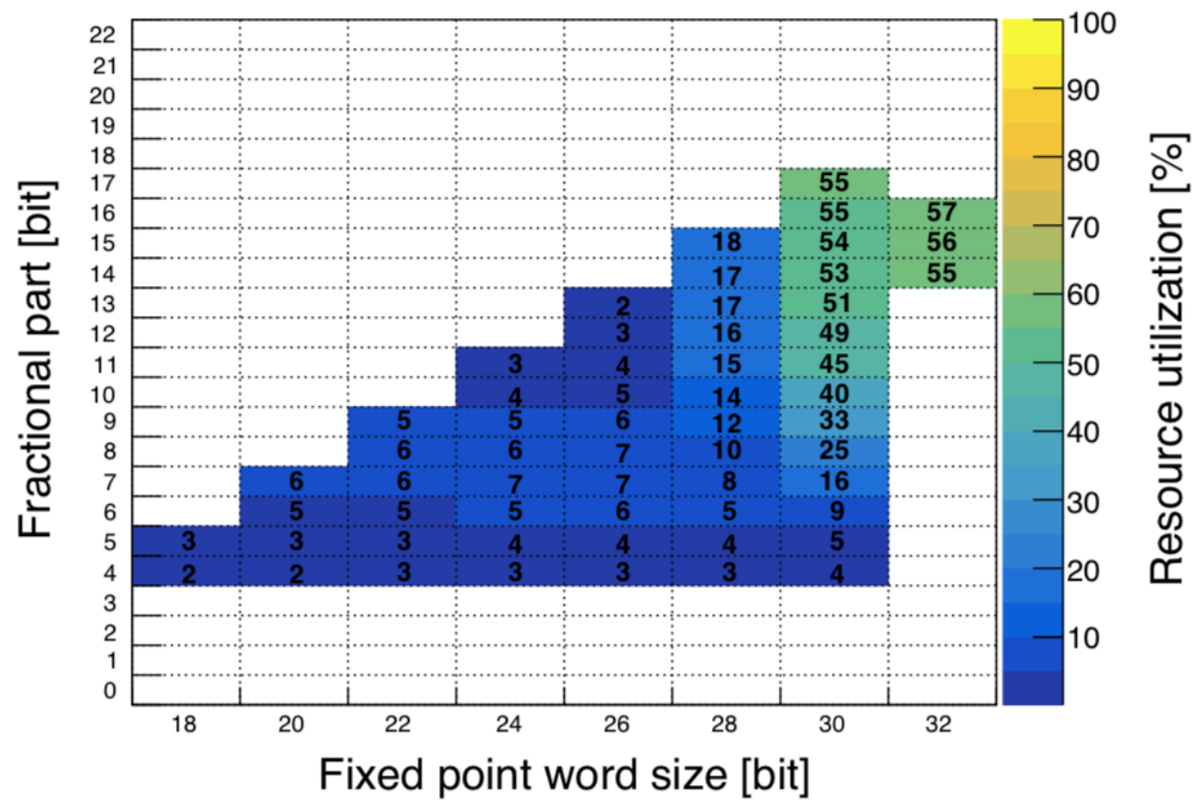
Back up 2

FF - Flip Flop, elementary register

HLS4ML



custom



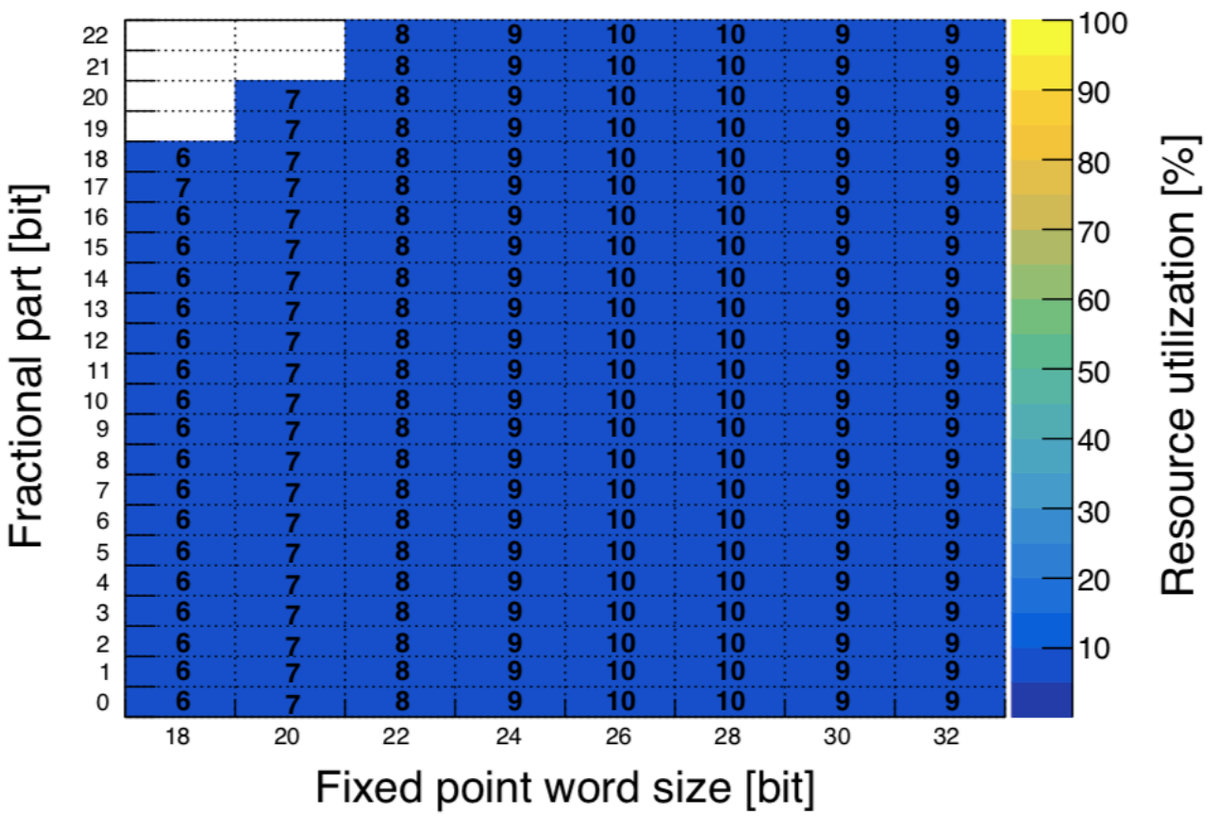
Resources expressed in % of available in Xilinx VCU118

not shown if accuracy < 50% (custom only)

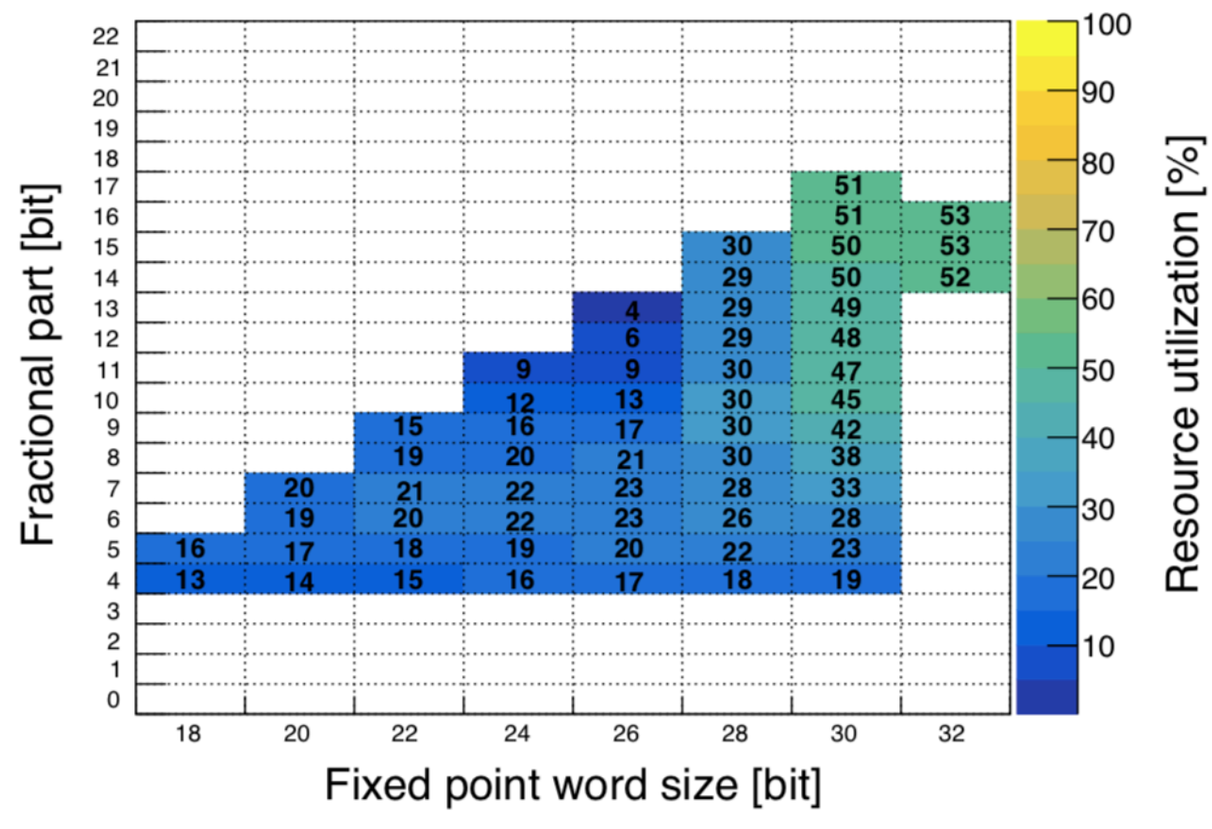
Back up 3

LUT - Lookup table, configurable logic elements

HLS4ML



custom



Resources expressed in % of available in Xilinx VCU118

not shown if accuracy < 50% (custom only)

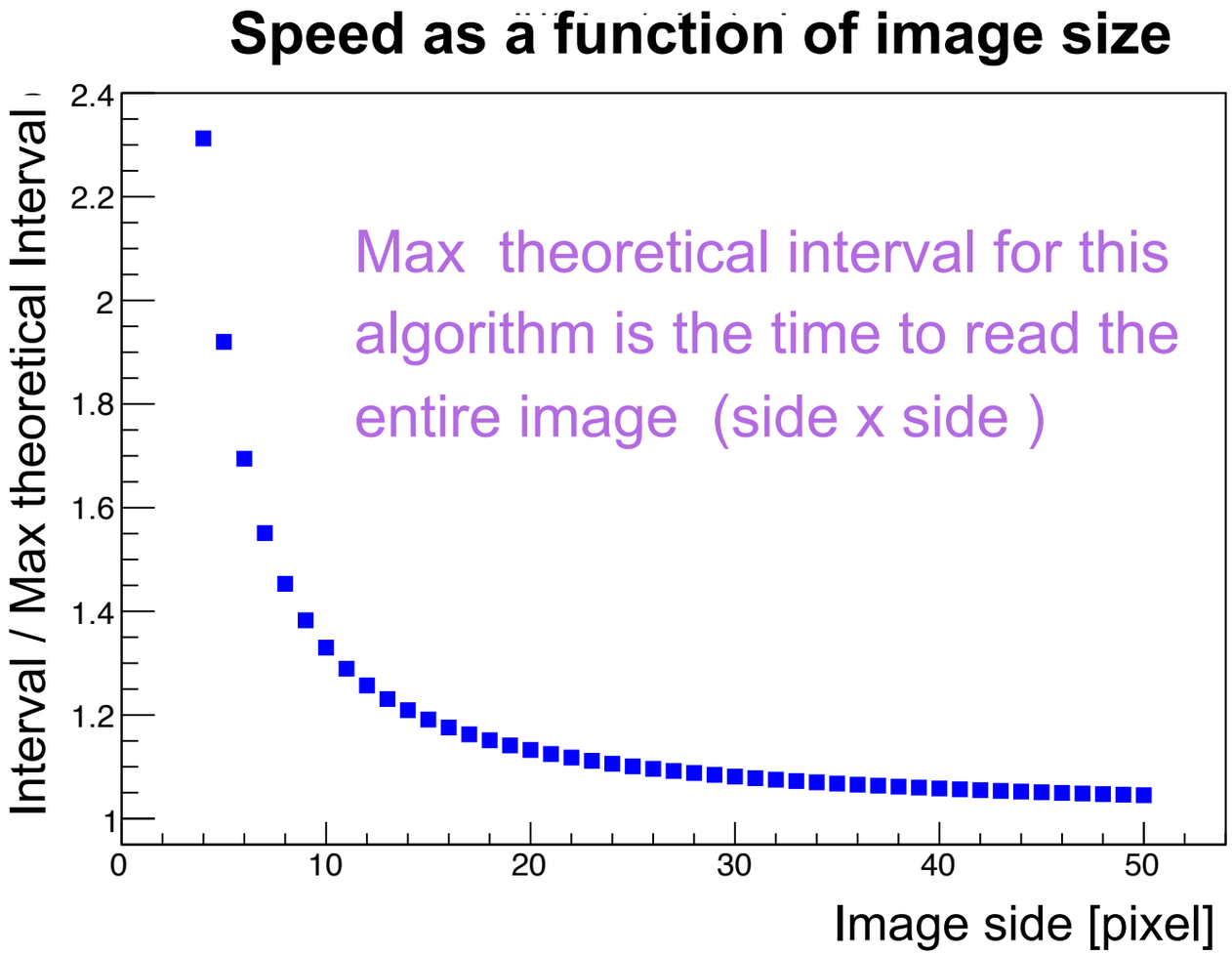
RiNNgs CONV

8 filters

Streaming convolution

Custom algorithm inspired by video streaming (data reuse, pipeline processing)

Design space exploration with random data set (no weight loaded yet)



Preliminary indications:

- Lightweight design
- Minimal dependency on image size
- Interval=Latency=Reading time
- not fast enough for NA62

(e.g. 2500 clock ticks for image 50 x 50 pixels)
@100 MHz interval is 25 microsec

ongoing exploration of a more parallel algorithm (process multiple pixel per clock ticks)