FERMILAB-SLIDES-23-311-CSAID

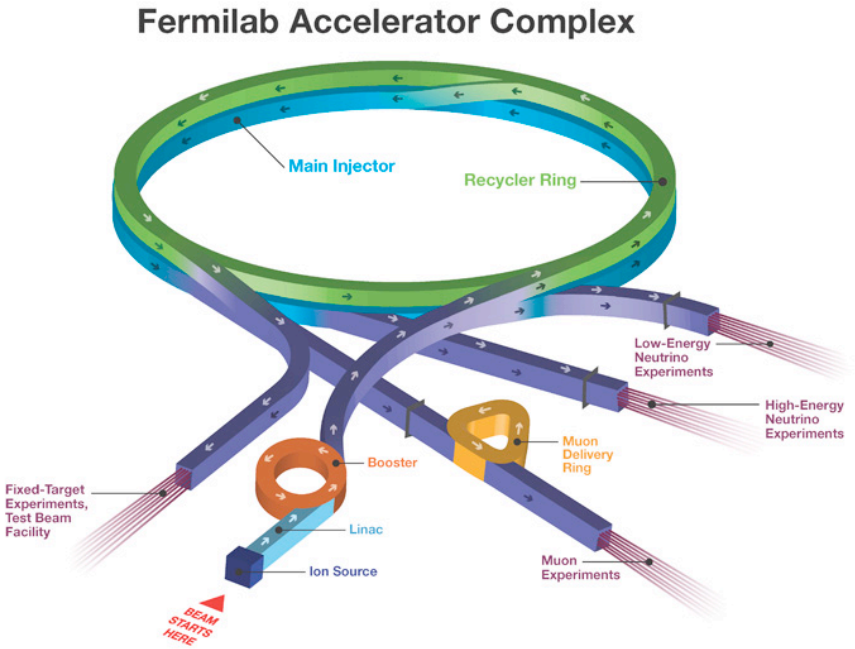
Edge AI for accelerator controls (READS): beam loss deblending

Jeremy Arnold, Mark Austin, Jose Berlioz, Dave Bracey, Pierrick Hanlet, Kyle Hazelwood, Jerry Yao-Chieh Hu, Aisha Ibrahim, Jing Jang, Han Liu, Seda Memik, Jovan Mitrevski, Vladimir Nagaslaev, Aakaash Narayanan, Dennis Nicklaus, Gauri Pradhan, Andrea Saewert, Brian Schupbach, Kiyomi Seiya, Rui Shi, Alexis Maya-Isabelle Shuping, Mattson Thieme, Randy Thurman-Keup, Nhan Tran, Chenwei Xu

Sept. 26, 2023 – Fast Machine Learning For Science Workshop 2023, London UK

Introduction

- At the Fermilab accelerator complex, the Main Injector (MI) and Recycler Ring (RR) share a tunnel.
  - MI: 8–120 GeV proton accelerator for high-energy neutrino experiments, etc.
  - RR: 8 GeV permanent magnet ring, proton stacker for MI; beam to g-2.
- Design from the Tevatron days: RR was a storage ring for low intensity antiprotons.
- Now both the MI and RR can have high-intensity beams.



Problem

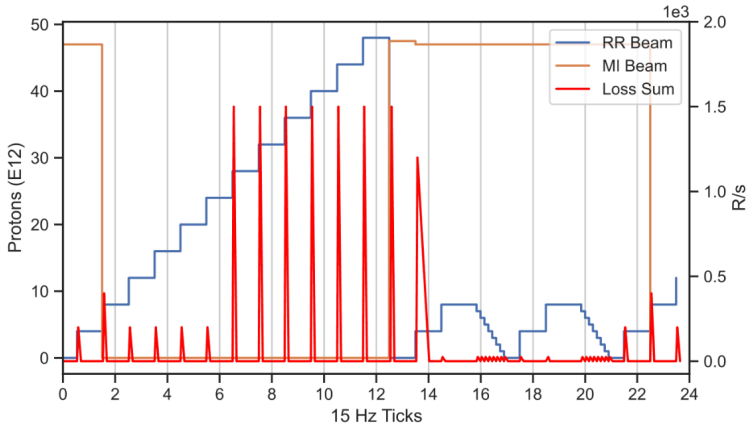
- Both the MI and RR can have significant beam loss
- About 260 Beam Loss Monitors (BLMs) spread around the tunnel to detect beam loss
- Often hard to know which ring is responsible for the beam loss: both machines can be unnecessarily tripped
- Purpose of project: better attribute beam loss to a particular machine to increase uptime

Beam Loss Monitor

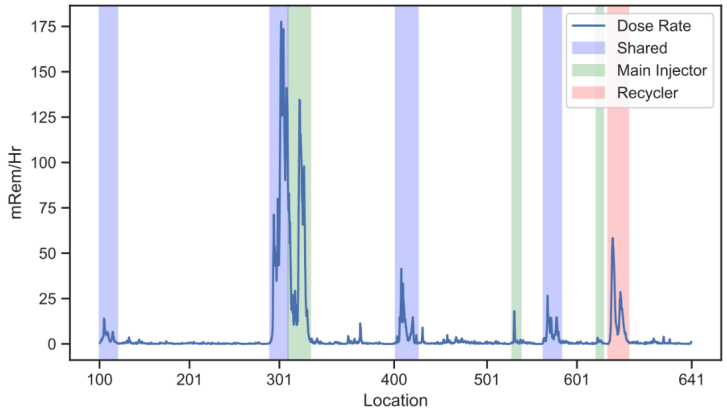
Fermilab

Beam Loss

- One can distinguish between MI or RR loss based on time / state of the machine or location
- Idea is to train a machine learning model to distinguish between the two



Example illustration of overlapping beam events and losses in the MI and RR accelerators

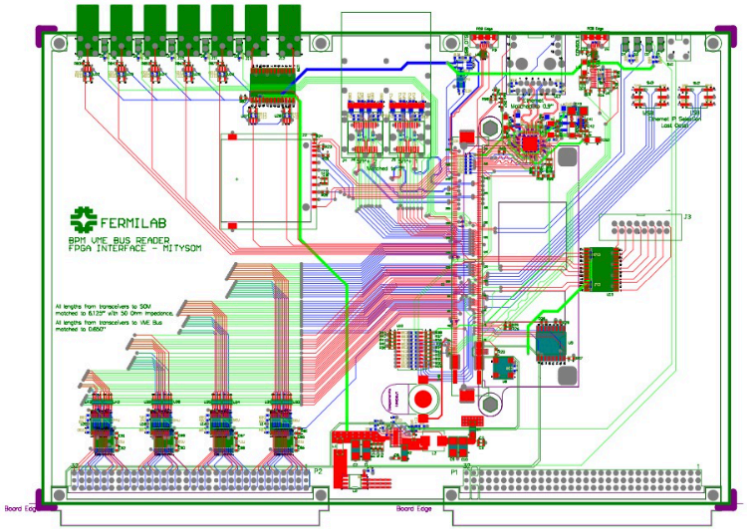


Location dependency of MI and RR beam loss as seen from tunnel activation residual doses

Fermilab

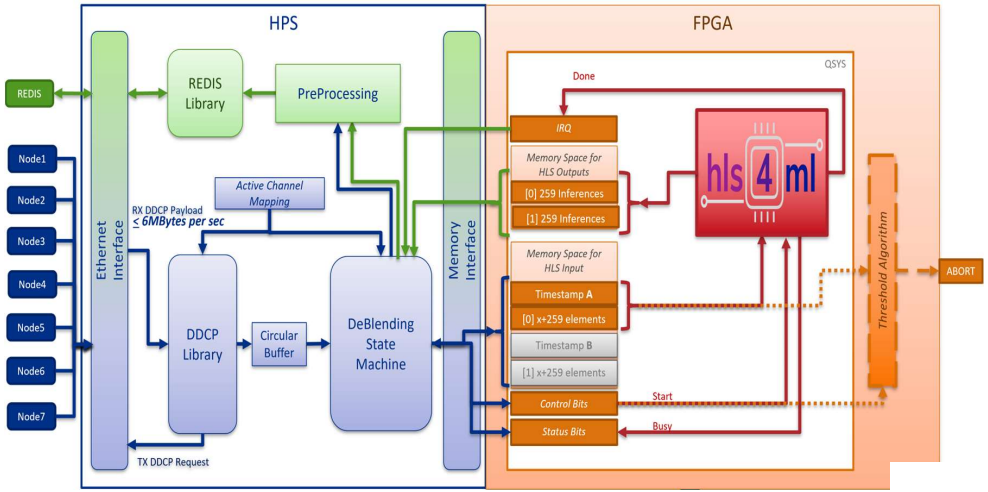
BLM Readout “Pirate” Cards

- BLMs are read out using 7 BLM nodes, tied to machine permit
- Standard method to read BLM values not suitable for ML deployment
- Added a custom VME reader cards (“pirate”) to each BLM node to stream out data
  - Cards use MitySOM Cyclone 5 FPGAs
  - Stream out data using DDCP protocol using UDP at 320 Hz



Central Node

- The data from the VME reader cards are collected by a central node
- Uses a REFLEX CES Achilles Arria10 SoC SOM
- The ARM cores collect the data streams, sends the samples to the ML (in the FPGA fabric), and streams the inference results out to Redis.

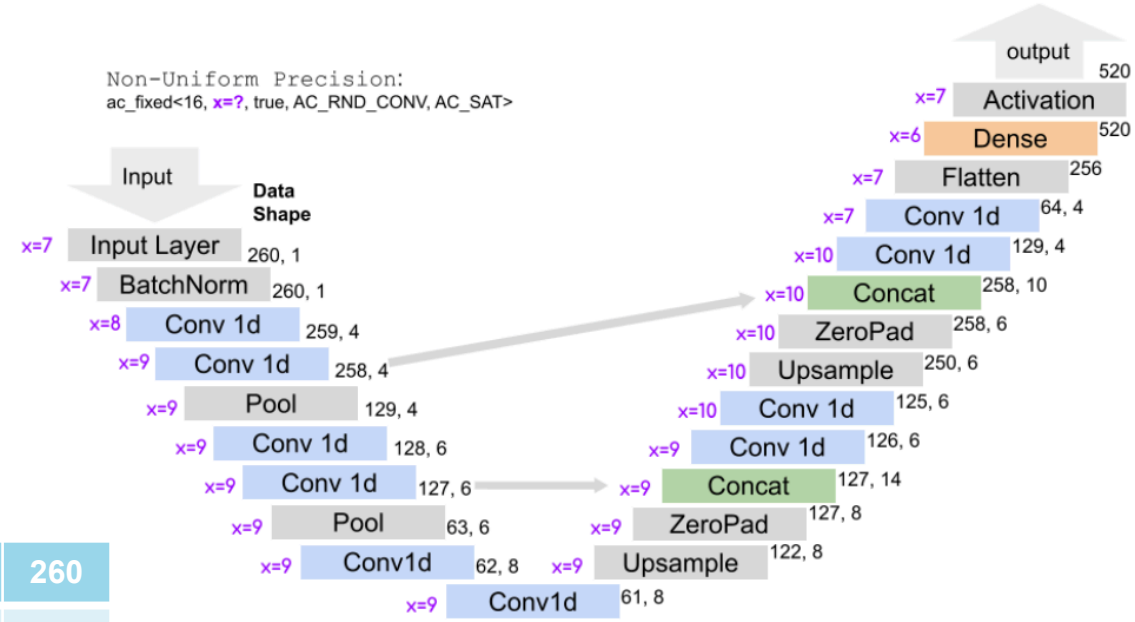


Fermilab

U-Net ML Algorithm

- After some tests settled on U-Net architecture: recognizes both global and local losses.
- Inputs: 260 BLM readings
- Outputs: 260 × 2 (nBLM × nMachines) probability that loss at a BLM is attributable to MI / RR.
- New inputs each 3 ms.

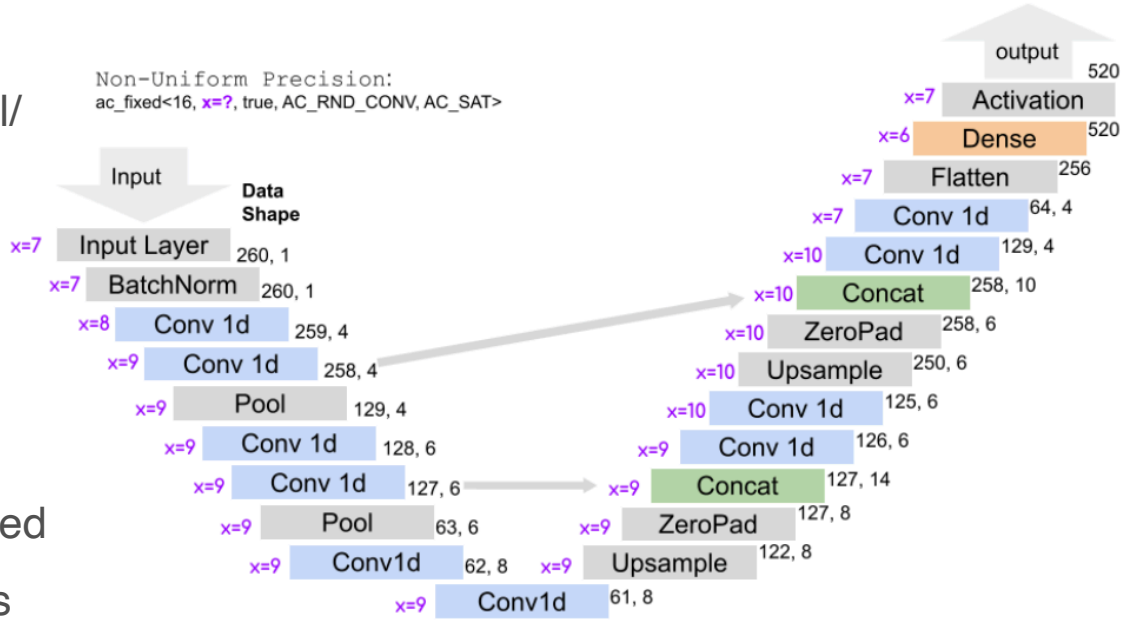
BLM	1	2	3	4	5	…	260
MI	p_1^{MI}	p_2^{MI}	p_3^{MI}	p_4^{MI}	p_5^{MI}	…	p_{260}^{MI}
RR	p_1^{RR}	p_2^{RR}	p_3^{RR}	p_4^{RR}	p_5^{RR}		p_{260}^{RR}



Fermilab

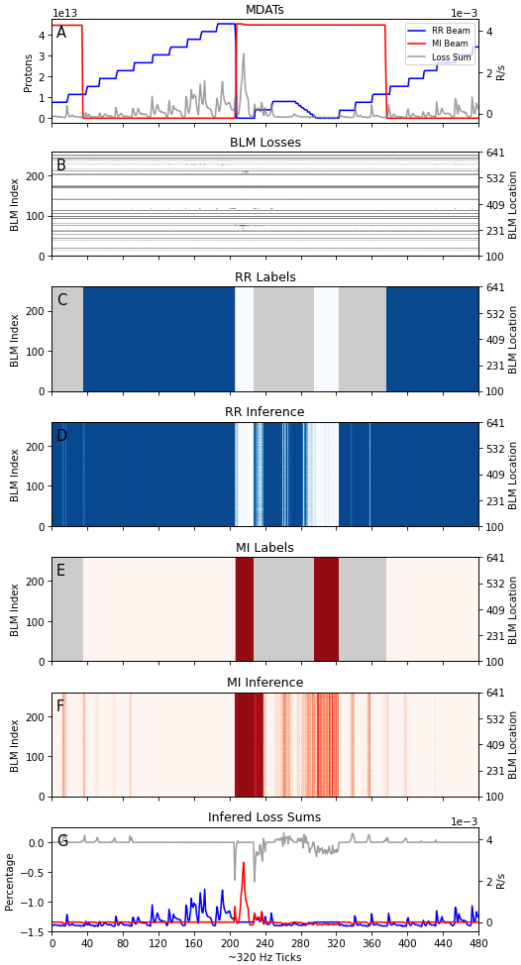
U-Net ML Algorithm

- Model is trained with Keras and uses post-training quantization
- Synthesized with hls4ml using Intel/ Quartus backend
- Used Intel Quartus Prime Pro 21.4
- In order to fit on an Arria 10 had to tune the model widths:
  - 8 bit weights
  - 16 bit layer outputs, int/frac varied
- System easily meets required 3 ms system latency: U-Net latency 1.57 ms, 1.74 ms system



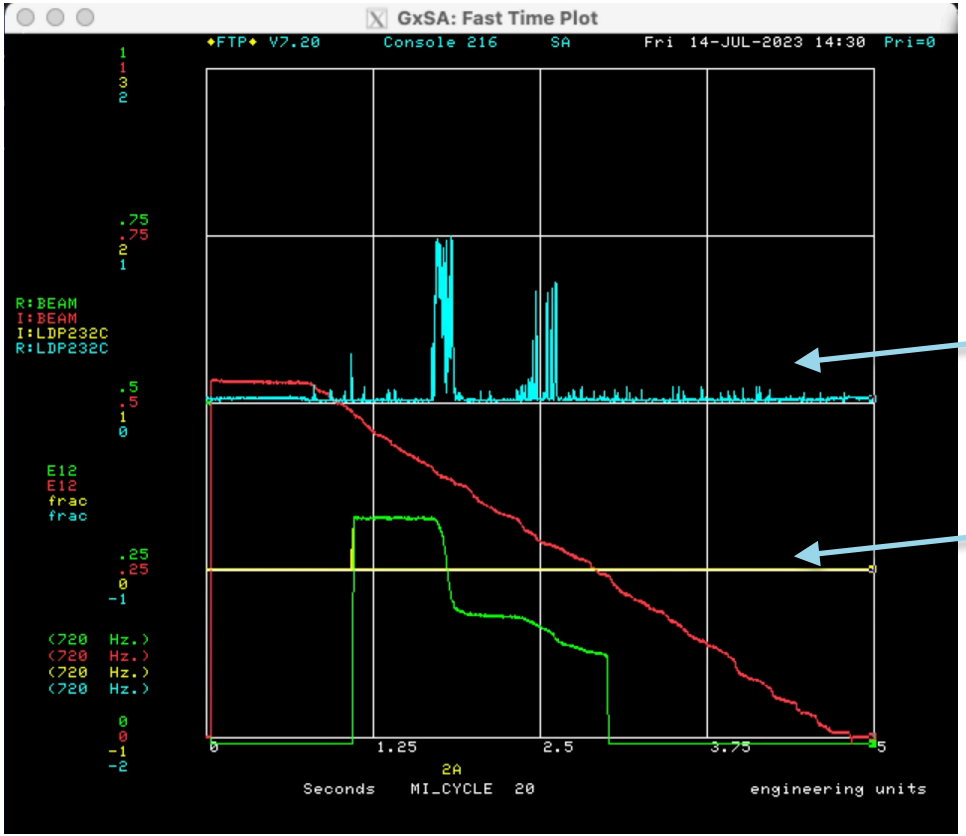
Offline Model Inference

- Attributes ring-wide losses well, but not local losses: a work in progress



Fermilab

Online performance, BLM 232C, purposeful loss in RR



Loss attributed to RR

Loss attributed to MI

Fermilab

Implementation details

- In order to work with the skip connections, had to add a buffer size to the streams

```
template <typename T, unsigned int N> using stream = ihc::stream<T, ihc::buffer<N>>;
template <typename T, unsigned int N> using stream_in = ihc::stream_in<T, ihc::buffer<N>>;
template <typename T, unsigned int N> using stream_out = ihc::stream_out<T, ihc::buffer<N>>;
```

- Required changing templates inside `nnet_utils` implementations
- Ran FIFO depth optimization using the Vivado backend in order to get initial buffer size estimates, increased some until cosim did not deadlock.
- Given that this is an extensive change and the Intel HLS compiler is being deprecated (with ihc::streams being replaced with pipes), decided to wait for the oneAPI version and not try to make a pull request.

Additional change

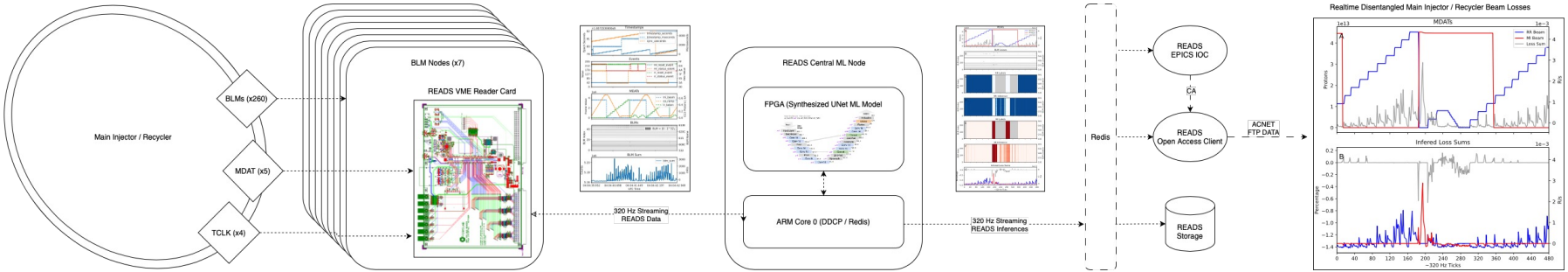
- The streaming zero-padding implementation caused errors that were not obvious:

```
Multiple reflexive accesses from stream 'layer3_out' is not allowed
```

- We wrote an alternate less optimized zero-padding implementation that did not produce this error.

Conclusion

- Created a full system with custom hardware and software
- Trained an ML model to distinguish between losses from the MI and RR
- Synthesized the model using hls4ml, the Intel HLS compiler, and Quartus
- Successfully deployed an ML model in hardware as part of accelerator controls
- The first deployment of Realtime Edge AI in the Fermilab accelerator complex



Fermilab

Acknowledgement

This work was produced by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics. Publisher acknowledges the U.S. Government license to provide public access under the DOE Public Access Plan DOE Public Access Plan.

Backup

Network details

System Properties	U-Net Model
Trainable Parameters	134434
Default Precision	ap_fixed<16, 7>
Precision Strategy	Layer-based
Default Reuse Factor	32
Dense/Sigmoid Reuse Factor	260
Average System Latency	1.74ms
FPGA U-Net Latency	1.57ms
Logic Utilization	223674 (89%)
Total Registers	406123
Total Pins	221 (37 %)
Total Block Memory Bits	25275808(58%)
Total RAM Blocks	1818 (85%)
Total DSP Blocks	273 (16%)
Total PLLs	3 (5 %)