Embedded FPGAs for HEP

Reconfigurable Digital Logic in 28nm CMOS for Smart Pixel Readout

Julia Gonski, Kenny Jia
SLAC National Accelerator Laboratory

July 29, 2024

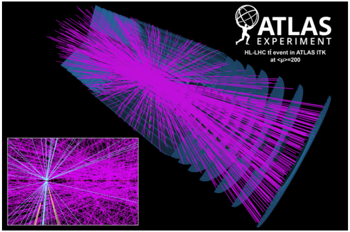
Outline

- Motivation and Physics Context
- eFPGA Technology
- Description of the Dataset
- Our strategy
  - Proof of Concept (resource constrained)
    - Software model
    - High Level Synthesis
  - High Performance Model
    - Software model
    - High Level Synthesis
- Future Plan

Motivation and Physics Context

- Collider Pixel Detectors (ITk):
  - each pixel is a silicon sensor with ASIC
  - O(100) million pixels, Pixel size O(μm^2)
  - Petabyte per second data rate (more for future colliders!)

Challenge: how to effectively reduce the data volume transmitted off-detector while preserving useful physics information as much as possible?



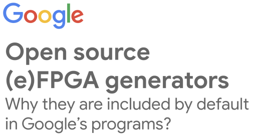
Goal of R&D

Lossy data compression with on-chip ML algorithm enables extraction of key physics info while minimizing data rate to be transmitted off-detector.

Embedded FPGAs (eFPGAs)

Basic idea is that you can put reconfigurable logic in your ASIC design.

- Full reconfigurability: can be configured just like a regular FPGA
- Power Efficiency: ASIC implementation means lower power than FPGA (“best of both worlds")
- Development Time: “plug-and-play" FPGA fabric into ASIC
- Cost: no need for costly engineer hours or licenses to design an ML chip



Google

Open source (e)FPGA generators

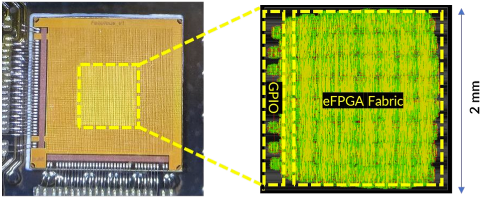
Why they are included by default in Google’s programs?

Also in use as hardware accelerators

See Larry Ruckman’s (CPAD 2023) talk for more

eFPGA Design

Original patents for several popular eFPGA architectures have expired. In 2021, University of Manchester has started an open-source project called “FABulous" to design ASICs with eFPGA fabric. Open source design framework reduces cost and lowers barrier to entry for institutions to participate in microelectronics design.

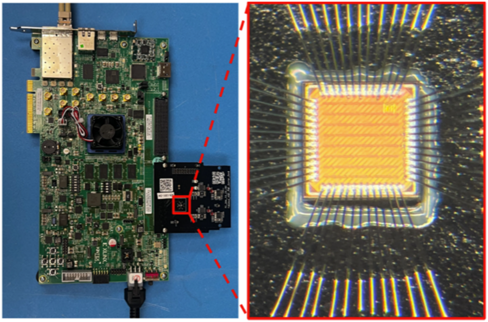


Fabulous v1 ASIC

Floorplan of eFPGA fabric after Place&Route

eFPGA Development at SLAC

- SLAC's Technology Innovation Directorate (TID) demonstrated an eFPGA design using FABulous framework in a 130nm CMOS Multi-Process Wafer (v0)
- Subsequently designed a second “proof-of-concept" eFPGA in 28nm CMOS in 2023 (v1), 1mm × 1mm

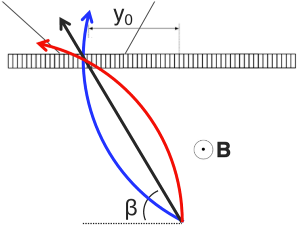


How do we do data reduction here?

Simplest way: train ML model to **classify** high p_T from low p_T tracks: reject fraction of data at source, save on data rate

Fancier ways: **regression** of cluster kinematic variables; **compression** via autoencoder and save only data in latent space.

- Starting from past smart pixel work! 2310.02474, 2312.11676



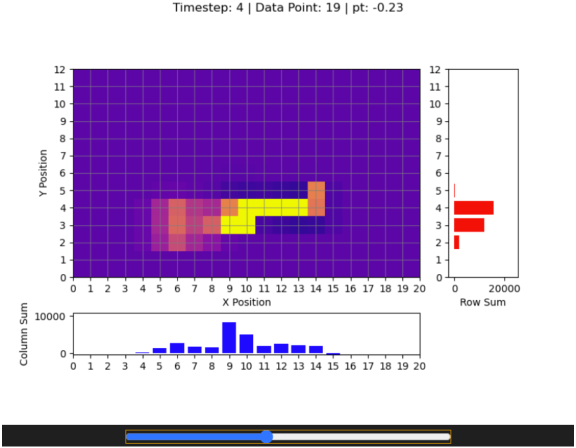
Description of the Dataset

We used the Smart Pixel[1] Dataset, which are pixel clusters produced by charged particles (pions) with real kinematics from CMS Run 2.

- Input:
  - 0.5 Millions of 20*13*21 (time × y position × X position) 2D "video" + y-local (y_0).
- Target:
  - 13 truth properties: x-entry, y-entry, z-entry, n_x, n_y, n_z, number_eh_pairs, pt, cotAlpha, cotBeta, y-midplane, x-midplane.

[1] https://zenodo.org/records/7331128

Description of the Dataset



Our strategy

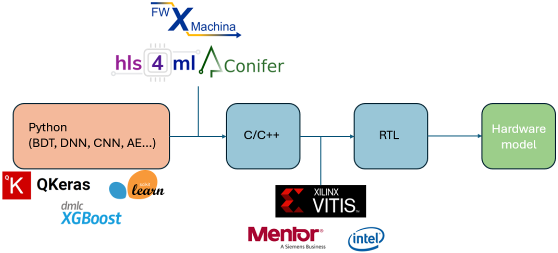
We wanted to achieve a **proof of concept (PoC)** with the current 28nm eFPGA chip first, then scale up to a more realistic logical capacity with a **high-performance model** and tape out a new v2 eFPGA.

1. For the PoC case: main challenge is the *extremely low logical capacity.* Number of LUTs is at the scale of 1/1000 in compare to regular commercial FPGA like Xilinx Virtex 7 (400 v.s. 400k-2M).
2. For the high performance case: explore power of reconfigurability with multiple algorithms (classification, dimension reduction, and regression) in *codesign* with eFPGA engineer team

Workflow

Four implementation steps:

- Software ML model: (Q)Keras, scikit-learn, XGBoost...
- C++ code generation: HLS4ML, SNL, conifer, fwXmachina...
- Synthesis and CSIM/COSIM: Catapult HLS, Intel HLS...
- Testing on chip

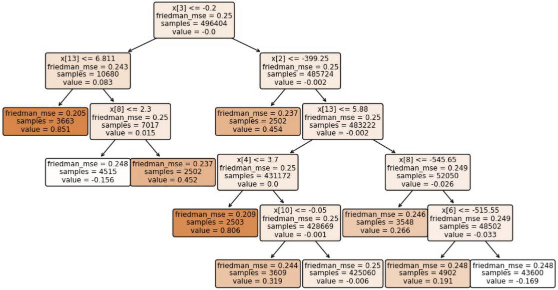


Step 1: Proof of concept

- Highly constrained (unrealistic) resources:
  - LUTs: 448
  - MUX2: 224
  - MUX4: 112
  - MUX8: 56
  - DSP_Slice: 4
  - Global_Clock: 1
- To achieve PoC, we simplify the model at every single level.
  - Input: 1D array with 14 features: 13 values summing over time and x_pos, with y_0
  - Output: probability score of whether track $p_T > 2$ GeV.

PoP software level: BDT

- Our first shot was looking for simple fully connected layers. Even with a few nodes, the LUTs needed exceeds far more than what we have on the current chip.
- Second attempt is Boosted Decision Tree. It is fast to train, powerful, and resource-efficient!



PoC results

Translation from python to C/C++ done with `conifer`[2].
Signal Efficiency:96.36% Background Rejections:5.76%
Not meant to be a realistic physics algorithm! Constrained resources mean that all the PoC can offer is a confirmation of the chip simulation & eFGPA reconfigurability.

Use only 294 LUTs and nothing else (BRAM_18K, DSP, FF, URAM). Latency under 25ns.

Successfully configured PoC model on eFPGA chip, matching perfectly to expected output!

Documented in SLAC eFPGA paper: 2404.17701 submitted to JINST

[2] https://github.com/thesps/conifer

Conclusions & Next Steps

- eFPGAs proving to be an interesting technology for intelligence and reconfigurability at-source in collider experiments
  - SLAC eFPGA paper submitted to JINST: 2404.17701
  - Discussing a variety of other applications: waveform classification eg. CalVision, straw tracker readout, ...
- Next steps
  - Exploration of high performance models (regression/dimensional reduction; new Smart Pixel datasets?)
  - Codesign and tape out of new v2 eFPGA
  - Develop FABulous for collider applications: Triple Modular Redundancy(TMR) for radiation hardness test at test beam, cryogenics, ...
- Many thanks to Smart Pixel team for the dataset & help along the way!

Backup

Power

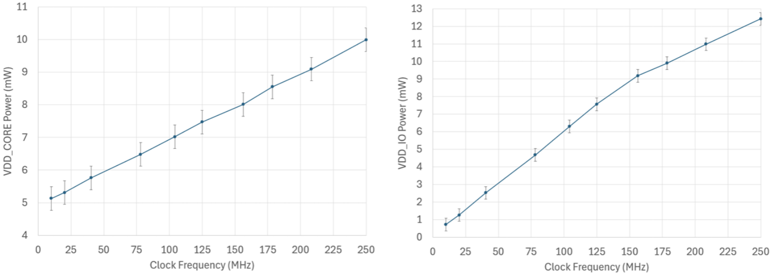


Figure 10. Plot of 28nm ASIC core voltage power draw versus clock frequency (left), and plot of 28nm ASIC I/O voltage power draw versus clock frequency (right).

PoP software level

An interesting finding: LUTs need depends more on number of estimators than the max depth.

BDT config:
’loss’: ’log_loss’,
’learning_rate’: 0.1,
’n_estimators’: 1,
’subsample’: 1.0,
’criterion’: ’friedman_mse’,
’min_samples_split’: 5000,
’min_samples_leaf’: 2500,
’max_depth’: 5,
’max_features’: None