SWIFT-HEP
offline trigger & reconstruction on FPGAs

https://indico.cern.ch/event/1466097/

Alison Elliot *(STFC-RAL)*
Sam Harper *(STFC-RAL)*

Status and plans for the future

- This talk is primarily to highlight the status of the work done in the FPGA trigger & reconstruction area over the last ~year
- A brief look is taken at other FPGA developments in the community
- Ideas and directions for the future are presented - with the hopes for feedback and discussion

- While the status is largely unchanged since last SWIFT-HEP in March (see next slide), it is useful to frame what has been done in the context of where we are thinking of going

Summary of the last six months

Where we were in March

summary of Reco status at SWIFT-HEP #7 https://indico.cern.ch/event/1366954/

- implemented simple ECAL crystal energy reconstruction algo on FPGA
  - directly produced the rec-hits in standard CMSSW data format
  - time comparable to CPU but likely room for improvement

- explored FPGA libraries for standard functions (eg minimisers, matrix multiplications etc)
  - Did not quite get the performance we were hoping for but should try and spend some time optimising it

Review of other FPGA activity

General thoughts:

- The biggest area of activity in FPGA development is using ML at the hardware trigger level (L1)
  - main tool kit: https://fastmachinelearning.org/hls4ml/
- Other activity on use of FPGAs to offload processing as an accelerator

There have been some key meetings:

- FPGA developers' forum: https://indico.cern.ch/event/1381060/
- CHEP (see next slide): https://indico.cern.ch/event/1338689/

Review: what's going on in the world?

CHEP - summary of FPGA talks in CHEP:
[Track #2 summary: Online and real-time computing]

- interesting chep talks:
  - Low-latency AI for triggering on electrons at High Luminosity LHC with the CMS Level-1 hardware Trigger.
    https://indico.cern.ch/event/1338689/contributions/6015453/
  - FPGA implementation of the General Triplet Track Fit
    https://indico.cern.ch/event/1338689/contributions/6015445/
  - Evaluating FPGA Acceleration with Intel® oneAPI Toolkit for High-Speed Data Processing
    https://indico.cern.ch/event/1338689/contributions/6015441/
  - Real-time pattern recognition with FPGA at LHCb, an O(n) complexity architecture
    https://indico.cern.ch/event/1338689/contributions/6015410/
  - Porting MADGRAPH to FPGA using High-Level Synthesis (HLS)
    https://indico.cern.ch/event/1338689/contributions/6015985/

Vision of where we are going

Algorithms

1. acceleration of AI inference
2. offloading library functions
   - eg common functions like non-negative least squares, matrix multiplication
3. offloading end to end algorithms
   - eg producing tracks fully on the FPGA, producing ECAL rec hits

Hardware

Xilinx / AMD
currently the area we are using

Intel
interest in testing portability

Xilinx AI Core

evaluation criteria:

cost (including cooling and power costs): physics / CHF
power efficiency: physics / watt
maintainability and portability: physics / FTE

These are the areas we are potentially interested in exploring in the context of SWIFT-HEP

Algorithms: ML/AI

- acceleration of ML/AI inference
  - Makes sense for FPGA based hardware triggers (L1), lots of effort here outside SWIFT-HEP
  - Unclear if it beats GPUs / dedicated silicon, may not make sense for offline reconstruction
  - L1 has to run on a FPGA while offline could run on a CPU, GPU, or AI specific accelerators (NPU, TPU)
    - GPU or NPU may out class FPGA
  - L1 is also more resource constrained so models are significantly pared down
- CMS RAL group very active in L1 ML algos
  - vertexing algos, about to start focusing on anomaly detection
  - extremely active area of research in CMS as whole
  - while L1 based ML feels not quite SWIFT-HEP, strong synergy here to see if those efforts could be applied to software base reconstruction

Algorithms: hardcoded, minimal, repeatable

- acceleration of common functions:
  - library of common functions which code can call, eg
    - minimisers
    - fitters
    - etc
- pro: could be maximally generally applicable cross-experiment
- con: probably any gains would be erased by loss in transfer times
- Some common core algorithms are at the centre of most/all of the end-to-end functions (eg, matrix multiplication from Lucy at Sussex)

Algorithms: hardcoded, dedicated, end-to-end

- accelerator of complete algorithms
  - eg tracking, ecal rec hits

  Largely experimental specific

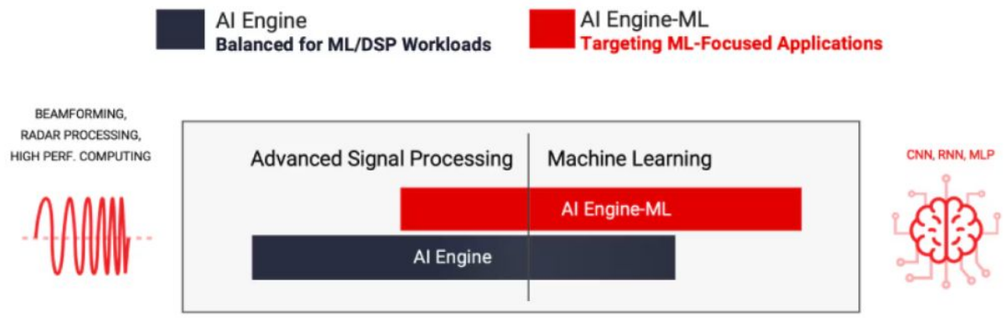
- A lot of ongoing work in ATLAS for offline tracking
  - Hough transform (Lucy)
  - EF tracking for Phase II
- Our work has been on CMS ecal rec hits
  - Takes in ADC samples, spits out rec-hits
  - the core of the simplest version is a multiplication
  - more complicated version is a least squares minimisation

Hardware: Xilinx & Intel cards

- Alveo u250 AMD Xilinx:
  - We have experience here, this has been our card to benchmark algorithms on
  - This hardware using specific Xilinx hardware level synthesis tools
- Intel Altera FPGA:
  - want to evaluate Intel FPGAs and how they work with oneAPI/SYCL in practice
    - RAL has a Stratix-10 GX dev kit on site to facilitate this
    - a comparison to AMD's ecosystem would be very valuable
  - critical to understand how portable code is between different FPGAs vendors
    - suspect not very portable
    - knowing how susceptible we are to vender lock-in is a key part of portability evaluation

Hardware: Xilinx AI Core

- Xilinx Versal AI engine:
  - https://www.xilinx.com/products/intellectual-property/versal-ai-engine.html
  - interesting to learn more about, particularly with AI
  - currently RAL does not have access to such hardware but considering a purchase



Target: next SWIFT-HEP

Explore some possibilities of where to go in algorithms and hardware

Build on what we have done in CMS algorithms, targeting both end-to-end and library functions

- Any investigations of AI inference likely not possible on this timescale

Investigate other hardware beyond Xilinx

Connect/reconnect with ongoing efforts in other areas and beyond SWIFT-HEP

Target: slightly longer term

Where we are going longer term depends on several criteria, including discussions and feedback from this SWIFT-HEP meeting

Documentation

If work is not documented, the work has not been done.

Any thoughts about:

- How best to lay out our experience
- How specific to get about algorithm development (cmssw specific tasks may not interest everyone, or maybe they will as a use case)
- How to then share it when it has been documented (probably on the SWIFT-HEP website)
- How everyone else is documenting? :)

Summary

Goals to deliver for next SWIFT-HEP:

- Finish investigations into “common function offload” and report on its feasibility
- (very) initial Intel portability studies
- Documentation
- Liaise with relevant parties

RAL group will be going more on L1 AI FPGA uses and in the future we will synergise with that effort and see what can be applied to reconstruction problems

Thanks, and your input would be greatly appreciated

backup