



Status of the DEEP-EST Project and Outlook

Viktor Khristenko (CERN)

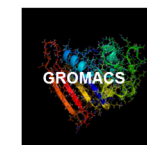
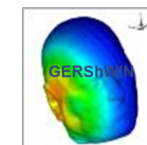
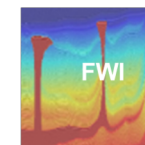
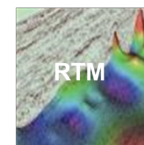
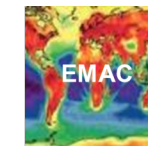


DEEP-EST Project: Objectives

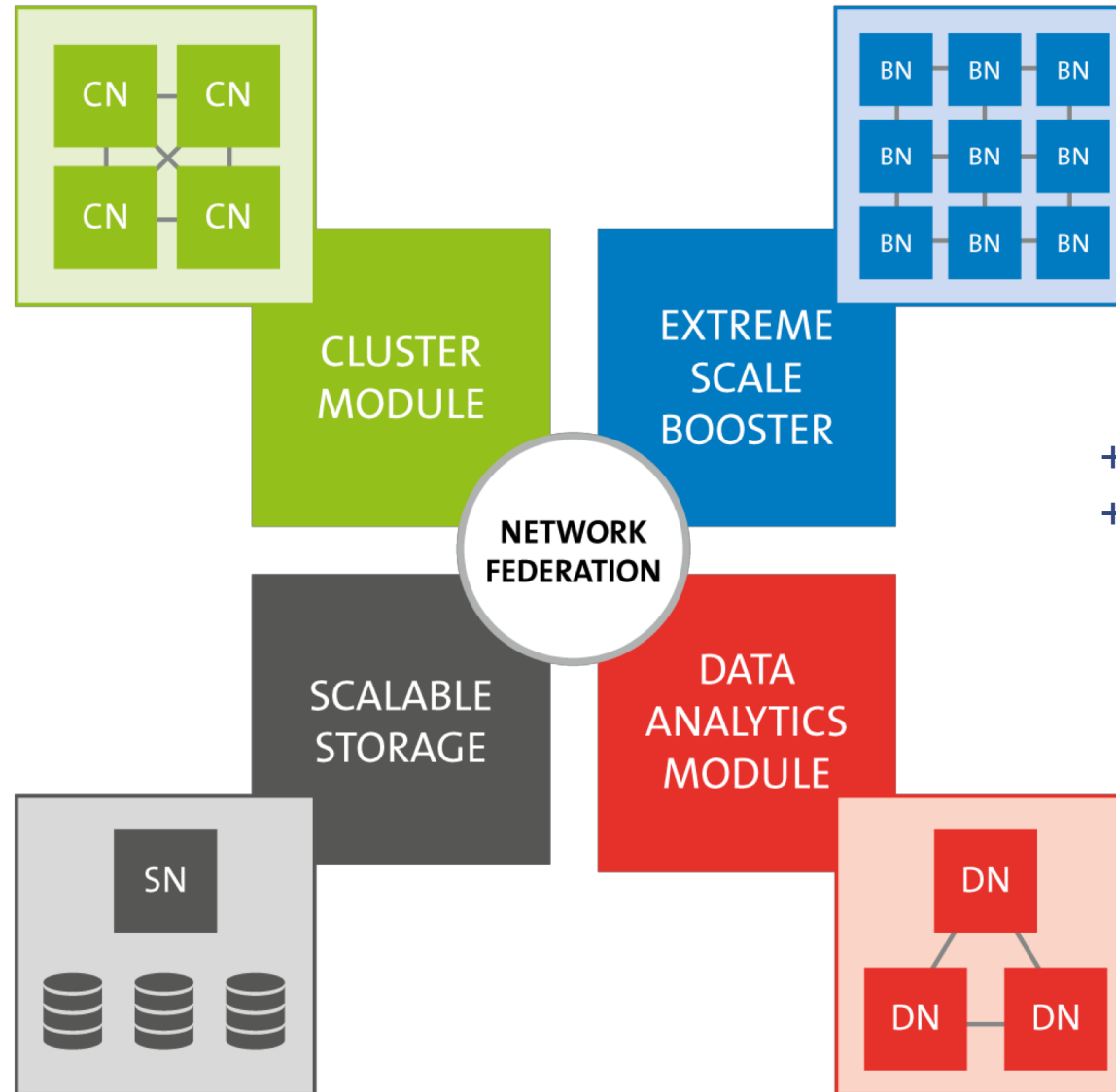
- In short -> Build Modular Supercomputing Architecture (MSA)
- Build a fully working, energy efficient prototype of the MSA
- Support HPC and HPDA convergence
- Extend a proven resource management and scheduling system to fully support the MSA
- Enhance and optimize the programming environment based on MPI and OpenMP. Add support for data analytics and machine learning frameworks
- Validate the full hardware / software stack with relevant HPC / HPDA applications
- Coordinator: Juelich Supercomputing Center
- 16 Partners
- 8 EU countries
- JSC, Intel, BADW-LRZ, BSC, ETH-Aurora, Megware, UHEI, EXTOLL, UEDIN, FHG-ITWM, KULeuven, ASTRON, NCSA, NMBU, UoI, CERN

The DEEP-EST Project: Applications

- 6 HPC and/or HPDA applications selected to drive the co-design process
- Used to evaluate hardware and software technologies developed within DEEP-EST
- HEP, Earth Science, Space Weather, Molecular Dynamics, Neuroscience, Radio Astronomy

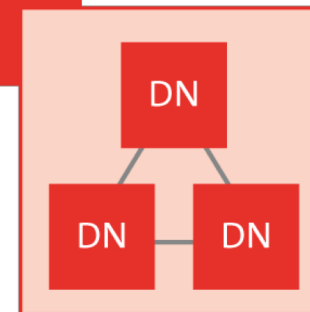
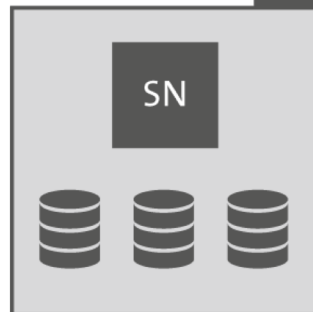
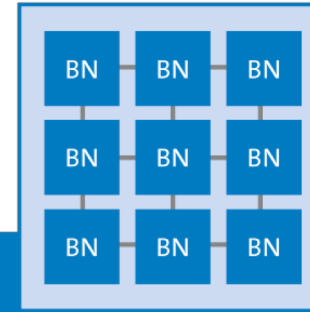
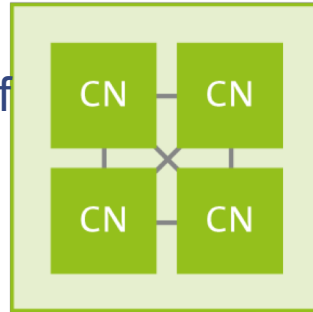


The DEEP-EST Project: Architecture



The DEEP-EST Project: Architecture

high-clocked CPU
to maximize single thread perf



GPUs + “Weak” driving CPUs.

Booster = Highly Scalable –
If app needs a lot of compute

3 types of interconnects overall:
Ethernet
EXTOLL
Infiniband

+ Network Attached Memory
+ Global Collective Engine

GPUs and FPGAs
Large DDR4 RAM capacities +
NVRAM as well

Goals and Motivation for HEP

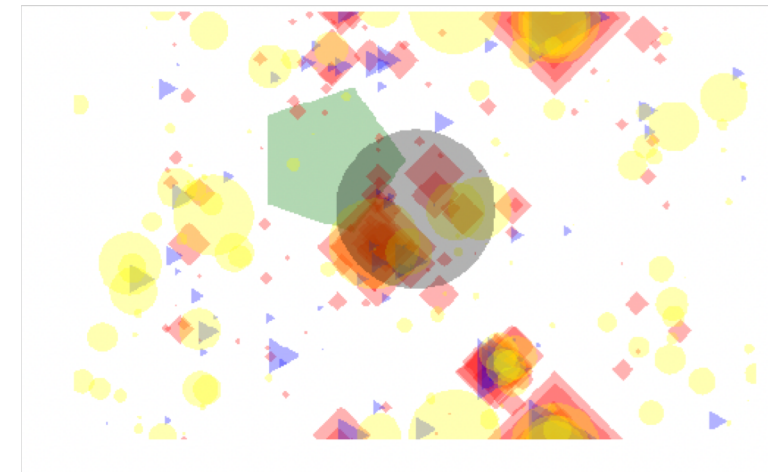
- Explore conventional HEP workflows on HPC infrastructure
 - Experiment with ways to deliver software stack
 - Experiment with new architectures (arch/uarch)
- Explore heterogenous options for data processing
 - CUDA / OpenCL / etc. devices
- Explore large scale ML/DL training/inference with HPC resources
 - usability of Apache Spark for HEP Data Analytics with HPDA resources
 - Other frameworks...

Status: Exploring HPC Infrastructure

- Several Deliverables were provided
 - Providing the specifics of HEP use case
 - Providing the preliminary mapping of the applications to the MSA.
- Complete CMS software stack delivery
 - To JSC (cvmfs + frontier squid + xrootd + etc.)
 - To Megware machines (binaries + frontier squid + xrootd + etc.)
 - Mostly to test and select CPUs for different cluster modules
- For CMSSW benchmarking
 - Employ official CMS Run 2 production workflows
 - Slurm (batch system) configurations were quickly implemented.

Status: ML/DL on HPC: Image Classification Pipeline

- Tight Integration of ROOT I/O with Apache Spark
 - New data source implementation
 - Allows pruning of deep nested structures to avoid bringing unnecessary fields
- The main use-case to be employed on the MSA is Deep Learning Image Classification Pipeline
 - 10TB simulated input
 - Feature Engineering pipeline is implemented
 - ML is implemented by @Mmiglio
- Multi-node tests of Apache Spark with Slurm
 - Needs further better integration
 - Currently just standalone clusters



Presented at Apache Spark Summit, London 2018

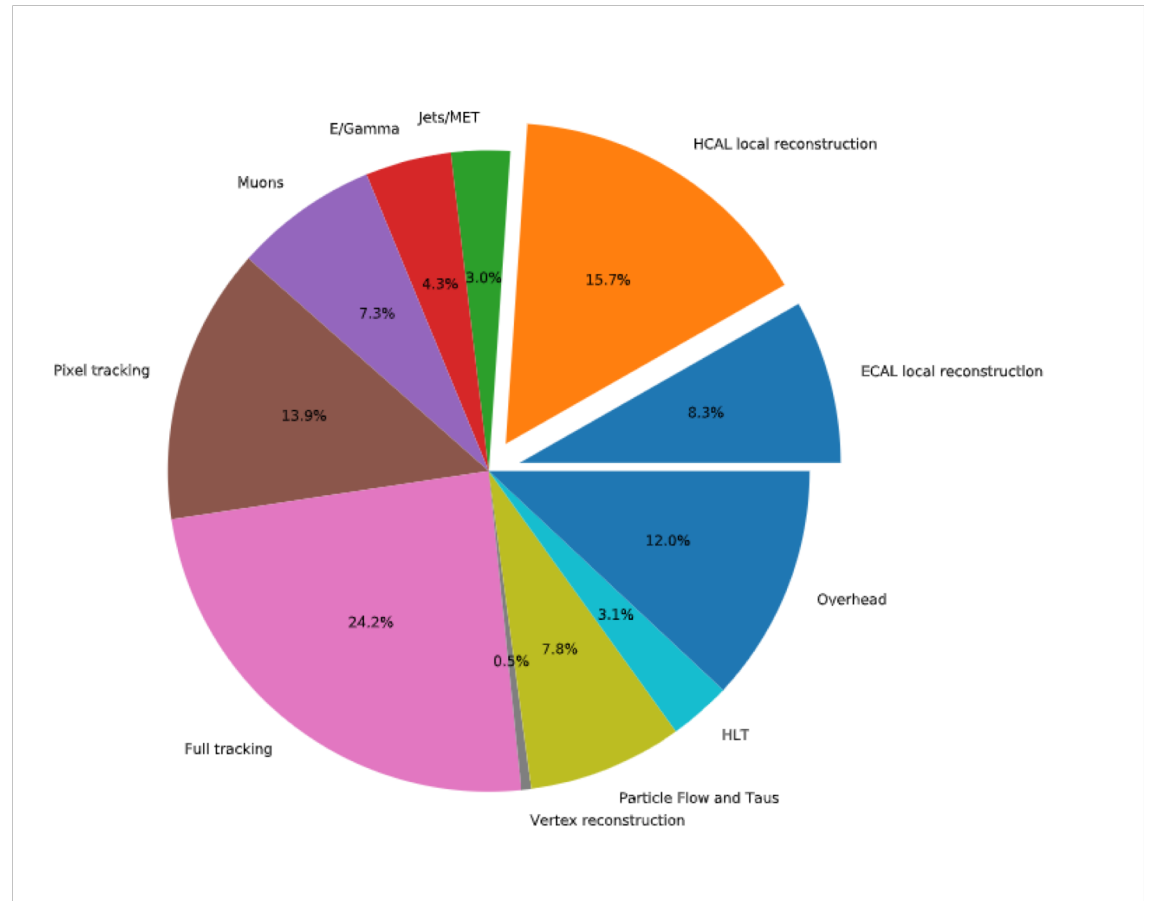
Heterogenous Resources for data processing

- Heterogenous Execution for CMSSW
 - Concentrating on HCAL / ECAL Local Energy Reconstruction

**Current Calorimeters take 20-25% RECO time
And both use the same algorithm -> fast NNLS**

Table 2.1: Time spent into the various HLT reconstruction steps

Step	Real-Time	Percentage
ECAL local reconstruction	38.9 ms	8.25%
HCAL local reconstruction	73.9 ms	15.67%
Jets/MET	14 ms	2.97%
E/Gamma	20.4 ms	4.33%
Muons	34.2 ms	7.25%
Pixel tracking	65.7 ms	13.93%
Full tracking	114.2 ms	24.22%
Vertex reconstruction	2.3 ms	0.49%
Particle Flow and Taus	36.8 ms	7.8%
HLT	14.7 ms	3.12%
Overhead	56.4 ms	11.96%
Total	471.5 ms	100%



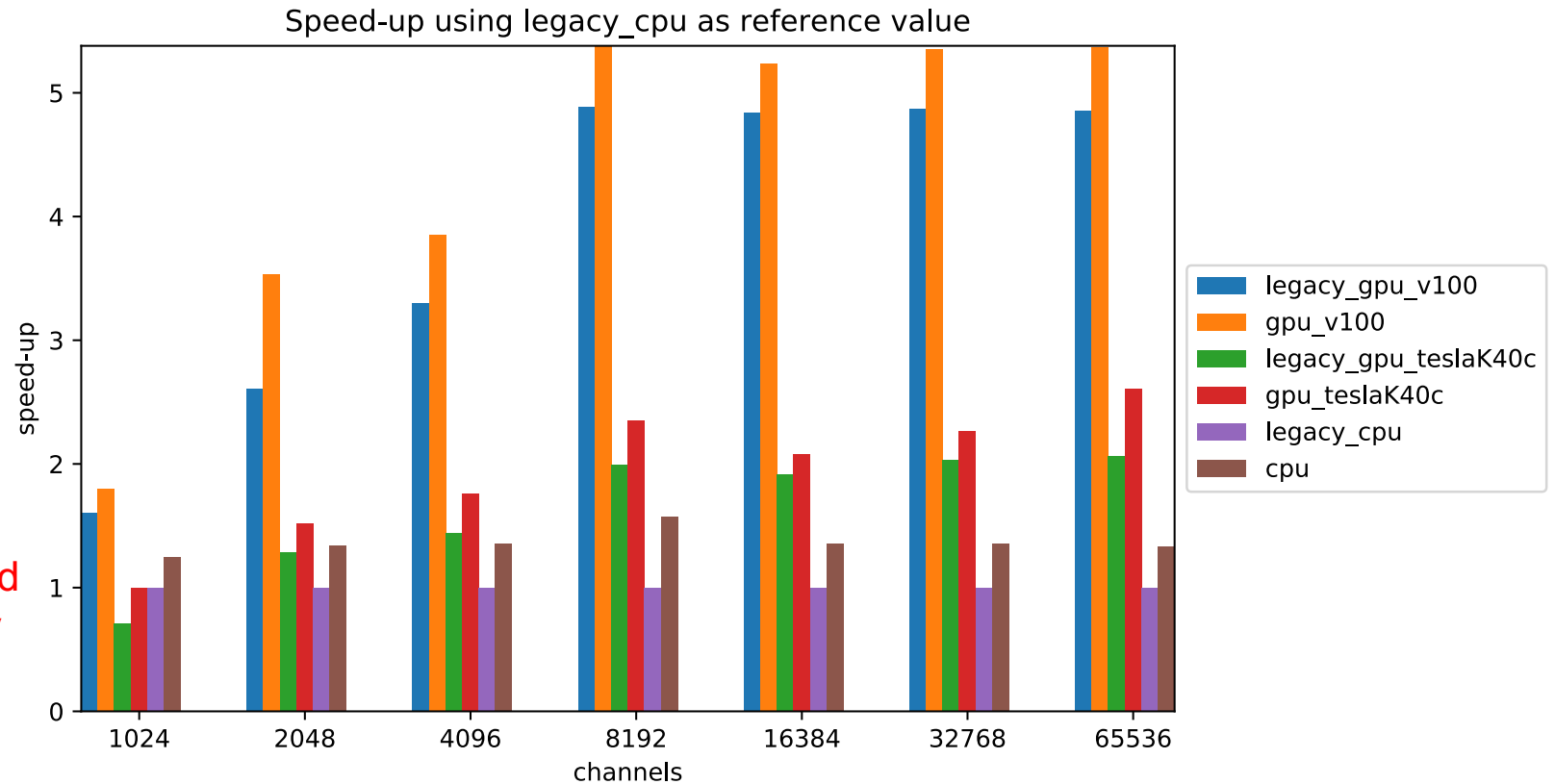
Standalone Implementation

CPU:
Intel(R) Core(TM) i7-4770K CPU
@ 3.50GHz

Tested with Tesla and Volta GPUs


CPU version runs single threaded as it is done for production jobs. Given a fully loaded CPU, no benefit from additional concurrency

The point is to remove this load from CPU
And understand if this removal is beneficial
(transfer + exe + transfer back)

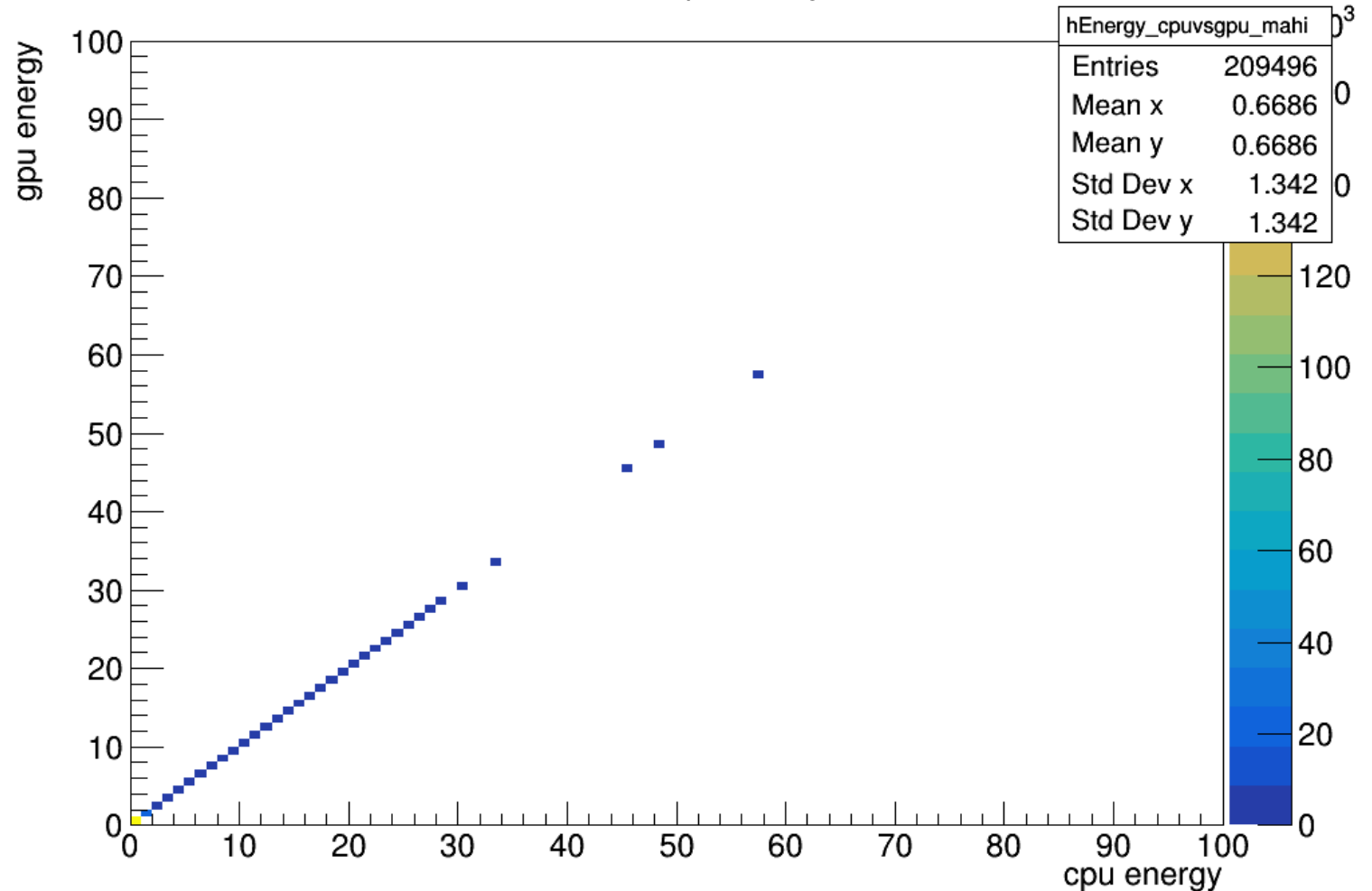


We observe factor of 5x speed up w.r.t.
Single threaded CPU implementation
In standalone version
For #calo channels \geq 8K with Volta cards

Porting to CMSSW

- Implemented and Integrated
 - apart from a couple (time slew) of corrections
- Physics Validation 
- tested in production job config with 8 tbb threads/streams {cmssw}
 - 1 cuda stream per cpu thread
- Next
 - Polishing
 - More Validation
 - Same for Ecal
 - Improve/Understand performance

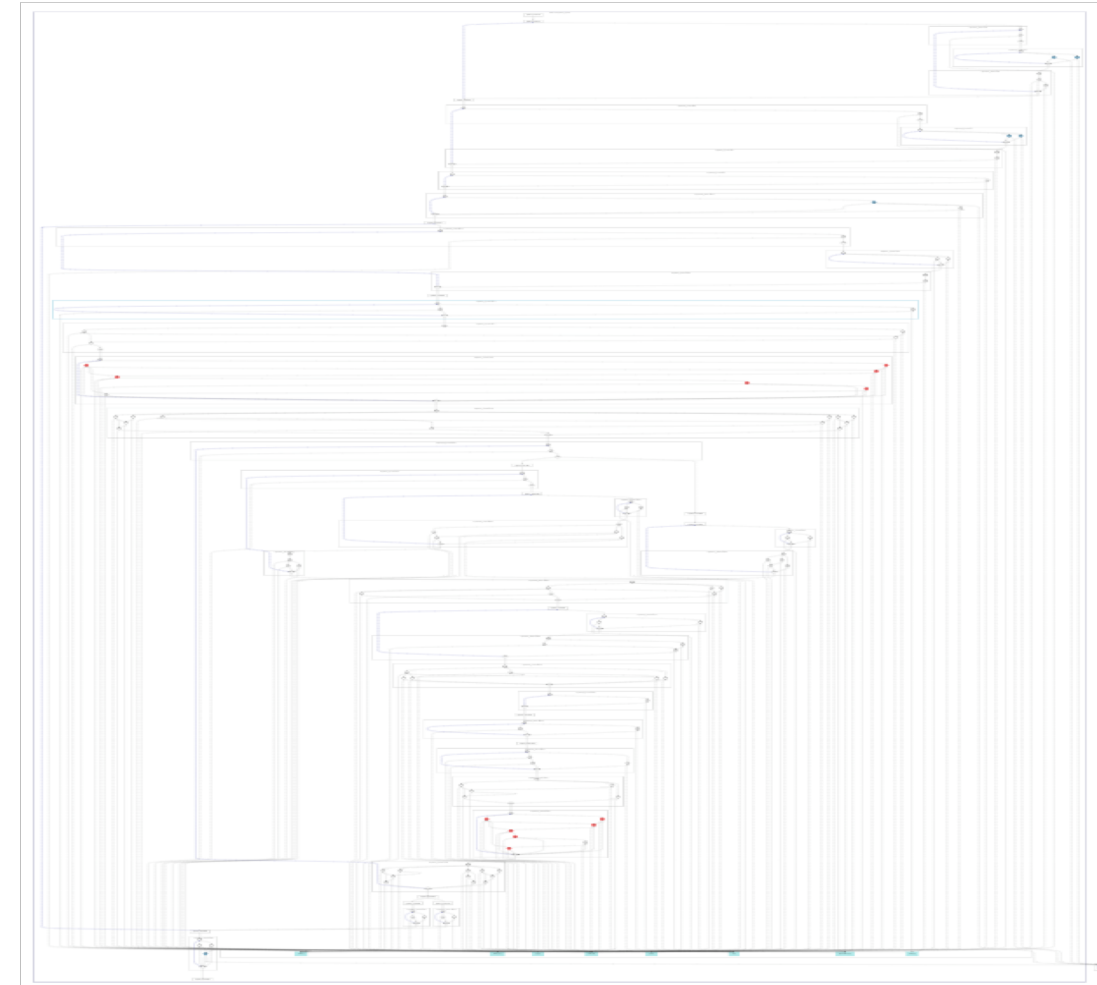
Hcal Reconstruction (MAHI) GPU vs CPU



More exotic? Testing Intel FPGAs

- Standalone implementation of Fast NNLS in OpenCL
- Offloading N channels
- impl details
 - Single-work item kernel and no replication (for now)
 - No pipes {yet}, monolithic {=> suboptimal}
 - Essentially c with fpga-specific pragmas

Data Flow for FNNLS



```

+-----+-----+
; Estimated Resource Usage Summary
+-----+-----+
; Resource           + Usage
+-----+-----+
; Logic utilization   ; 40%
; ALUTs               ; 22%
; Dedicated logic registers ; 19%
; Memory blocks       ; 27%
; DSP blocks          ; 4%
+-----+-----+

```

BSP is ~ half total logic



~5x slower than a cpu version with eigen (but no logic replication, etc...)
But with Identical results (up to 10^{-4})

DEEP *Projects*



The DEEP projects have received funding from the European Union's Seventh Framework Programme (FP7) for research, technological development and demonstration and the Horizon2020 (H2020) funding framework under grant agreement no. FP7-ICT-287530 (DEEP), FP7-ICT-610476 (DEEP-ER) and H2020-FETHPC-754304 (DEEP-EST).