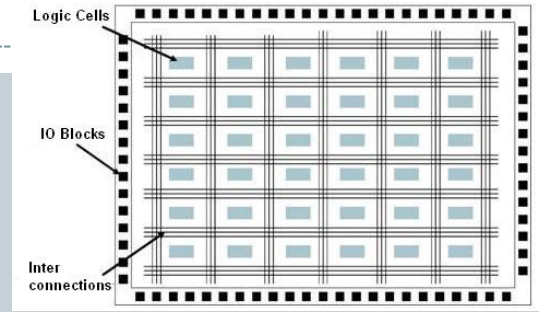
Accelerators and Alternative Architectures in Trigger and Reconstruction: FPGA

A. CERRI – UNIVERSITY OF SUSSEX

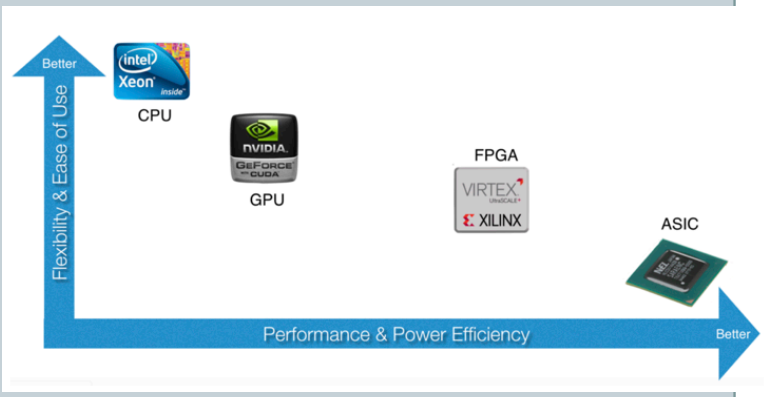
UNIVERSITY OF SUSSEX

ATLAS EXPERIMENT

FPGA Accelerators: "emerging" alternative



- FPGA: programmable array… of simple (logic, I/O, memory & specialised) logic blocks
- Once firmware ("program") is deployed, the implementation runs in hardware
  - More performing than processor
    - Logic blocks can parallelise tasks and calculations
    - But harder to program and optimise
  - Less performing than ASIC
    - Less efficient and optimal logic layout
    - …but re-programmable (debugging, improvements)



- Programming mostly through HDL (logic functional description) → specialized EE job

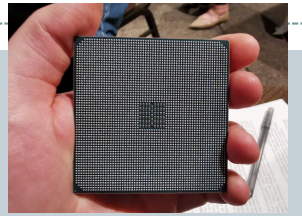
How is it emerging then?

- High-gain environments [e.g. cluster computing applications (think MS Azure) stemming from networking applications]
- Development of "High Level Synthesis": bridge towards portability and software universe

FPGA: HDL

- HDL: description of device behaviour in terms of logic operations on signals, expressions, statements, inputs, outputs
  - Implementation approached by describing Data flow, structure, behaviour
  - Approach much closer to designing a digital logic circuit than writing procedural programs
  - Performance gauged and tuned in terms of resource usage, latency, speed
  - Experienced developer can adapt code to features of a specific device
  - Development and results are device-dependent
- To a seasoned programmer this may sound like the analogous of assembly in microprocessors: powerful but system-dependent [although the more proper parallel is with RTL]

FPGA: HLS



- Nowadays FPGAs can be produced with O(1-10)M logic blocks, GB of fast access memory, embedded CPU, DSP blocks, Tb/s I/O etc.
- Resources can be sacrificed ("assembly → C++") in favour of:
  - "Brute force" performance
  - Development approachability
  - Portability (different FPGA)
  - Shareability of resources ("FPGA farms")
- Advantages retained:
  - Less expertise to achieve performance gain vs CPU
  - Flexibility vs ASIC
- Drawbacks (currently):
  - HLS still somewhat vendor-specific
  - Less control on low-level resources (e.g. latency)

Firmware Performance: aside remark

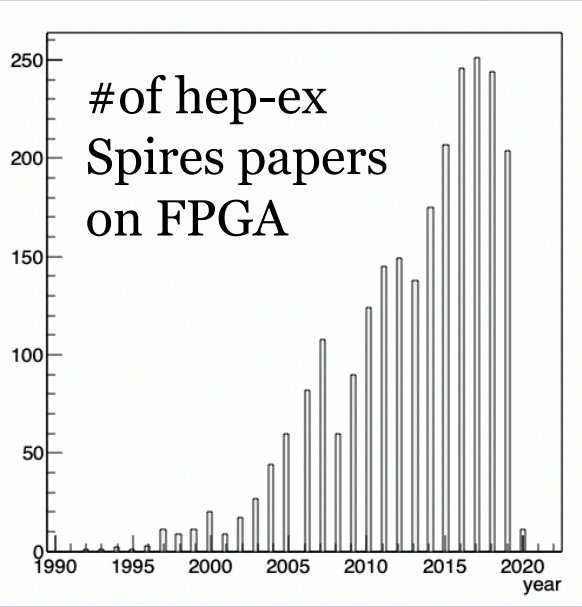
- No matter what the design language choice is, compilers will produce low-level firmware
  - HLS “compile” cycle much slower than for your typical CPU
  - Exact bit-level representation of the FPGA configuration
  - “post synthesis” simulations are natural part of FPGA development cycle
  - Performance predictions from these simulations is exact: latency and speed of a certain operation are known with certainty without deployment
    - This seems to be the source of some confusion, especially among non-experts, who insist on live tests
    - …probably stemming from software/CPU based experience?

Example of Farm implementation

- FPGA coprocessors distributed across farm nodes
- Interface analogous to GPU/OpenCL
  - Accelerator runs (from a choice of) pre-compiled algorithms on demand
  - Algorithm interface abstracted from accelerator
    - Different devices or CPU can be interchanged based on resources
  - Common stack across PCIe platforms → little portability effort
  - OpenCL stack manages: arbitration and configuration of resources, concurrent usage etc.
  - User application interfaces to accelerated algorithm
    - Typically provides or points OpenCL stack to pre-compiled implementation to be loaded

EPP Applications?

- FPGA so far exploited almost exclusively in application specific hardware: experiment/detector specific TDAQ electronics



Spires papers
on FPGA

Spires papers on
FPGA and computing

A prototype problem: Tracking

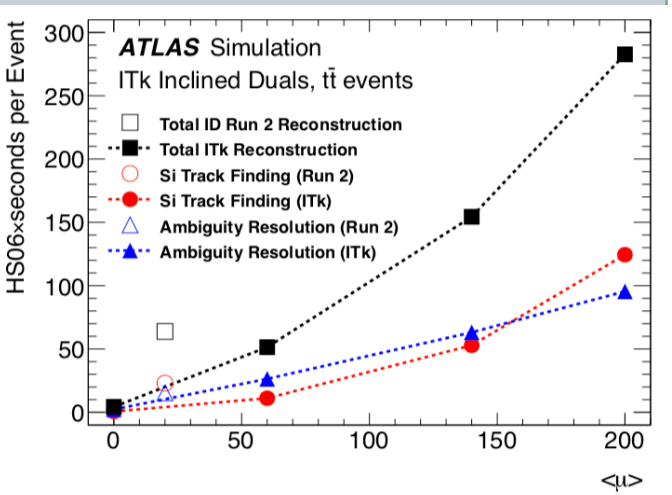
- Already discussed in Benjamin's and other talks
- Excellent benchmark, as implementation studies exist for:
  1. Commercial CPU
  2. GPU accelerators
  3. FPGA accelerators
  4. Real-time hardware
- (1) and (2) discussed elsewhere: will focus on (3) and (4)
- Caveat: different levels of maturity for different parts of these studies. Acceptable IMHO for today's discussion.
- Other very interesting studies being pursued and published (e.g. NN/triggering, NN co-processors, pID etc.)

HTT: Real-time Tracking in ATLAS @HL-LHC

- FPGA accelerator study based on ATLAS' Hardware Track Trigger design
  - Goal: full detector volume reconstruction of tracks at 100 kHz [coprocessor for trigger CPU farm]
  - Pattern recognition [ASIC] + linearised track fitting [FPGA] implemented in hardware

Detector Channels	Pile-up	Input Rate [kHz]	HW Footprint
0.2M	~1	30	4 Racks
100M	30-70	100	~6 Racks
800M	200	1000	25 Racks

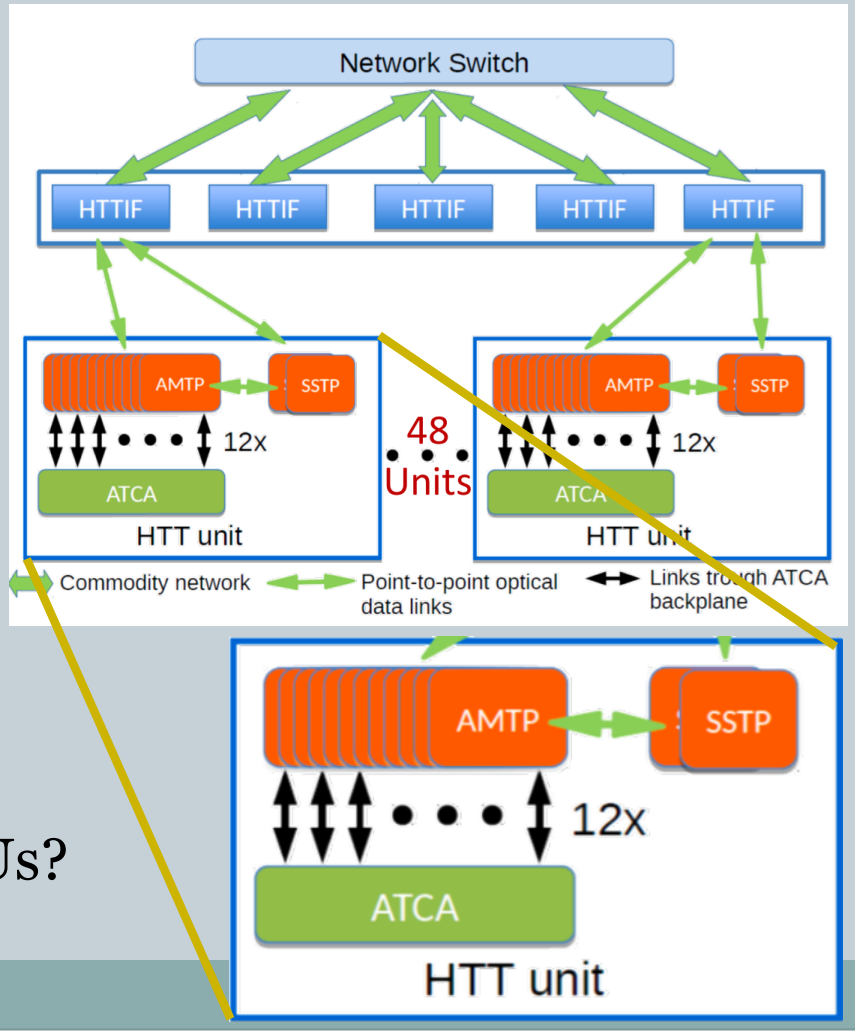
- System size has become limited by power density (~500 kW total power)
- CPU Farm based counterpart estimate: x2-x10 the ATLAS trigger processing farm and >x10 power consumption



HTT in one slide

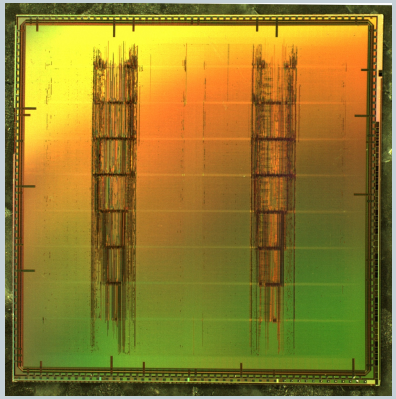
- O(500) custom hardware boards (ATCA)
- Modular structure:
  - HTT unit (~1 ATCA shelf) processes one of ~50 projective regions of the detector
  - Each unit performs:
    - Clustering
    - Pattern recognition
    - "1st stage" 8-layer fit
    - "2nd stage" high resolution fit
- Each unit contains:
  - 240 pattern recognition "Associative Memory" ASICs
  - 24 FPGA for Clustering+PR+1st stage fit
  - 6 FPGA for Clustering+2nd stage fit

Can this be implemented with FPGA coprocessors on commercial CPUs?



HTT→FPGA Accelerators

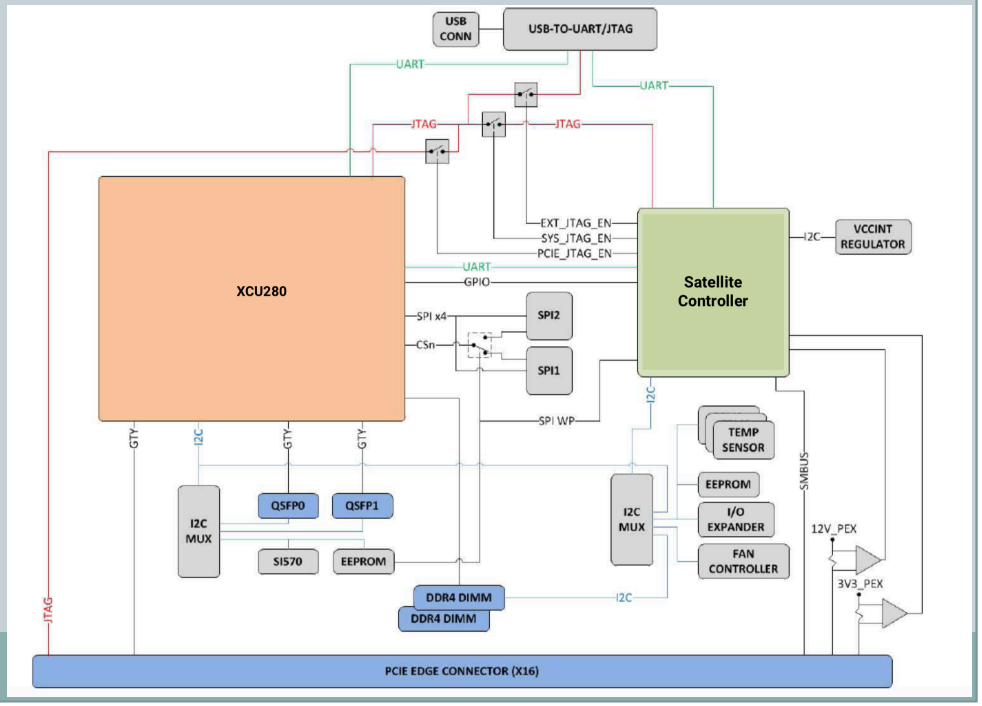
- Most steps already FPGA-based: firmware can be ported to a large extent to accelerators
  - Performance figures can be borrowed from hardware system TDR
  - Attention must be paid to I/O, less customisable
- Let's focus on "processing power"
  - Need to identify replacement for ASIC (i.e. pattern recognition) stage:
    - AM ASIC is a fully parallel template-matching device
    - Not most suitable to replicate with typical FPGA logic modularity
    - Hough transform provides a more suitable alternative (see e.g. by CMS in R. Aggleton *et al* 2017 *JINST* **12** P12019)



FPGA Accelerator

- I/O: O(100-800) Gb/s
- Local fast storage ~8GB
- O(3M) logic cells, 1M LUTs, 9k DSP blocks…
- Power usage 70W typ, 200W max
- 6-8 k$/board

DRAM Memory	
HBM2 Total Capacity	8GB
HBM2 Total Bandwidth	460GB/s
DDR Format	2x 16GB 72b DIMM DDR4
DDR Memory Capacity	32GB
DDR Total Bandwidth	38GB/s
SRAM Memory	
Internal SRAM Capacity	41MB
Internal SRAM Total Bandwidth	30TB/s
Interfaces	
PCI Express	Gen4x8 with CCIX
Network Interfaces	2x QSFP28 (100GbE)
Logic Resources	
Look-up Tables (LUTs)	1,079,000



Back of the envelope (NOT pretty!):

- Clustering: FPGA implementations (C L Sotiropoulou *et al* 2014 *JINST* **9** C10018) suggest O(50-100) FPGA accelerators needed to process the ATLAS ITk data
- Track Fitting:
  - 1st stage:
    - scale w DSP block availability 12Gfit/s/HTT unit
    - 1 Gfit/s ~ 1900 DSP blocks
    - Can accommodate in 300-400 accelerator cards
  - 2nd stage: same number of FPGA as HTT → 100 accelerator cards
- Pattern matching?
  - Critical parameters are #matches (→efficiency) and #fakes

Hough Transform in FPGA

Possible starting points for estimate:

- CMS L1 tracking trigger implements Hough transform on FPGA
  - Initiated with tracklets rather than single hits
  - 288 "HT arrays", each requiring 6kLUTs, 64 DSP, 6.7k FF, 33x36k=1Mb RAM
- LHCb VELO studies (JINST 11 (2016), C11040)
  - Substantially different detector geometry etc.
- ATLAS studies for HL-LHC (arXiv:1709.01034v1 and arXiv:1907.09846v3)
  - Replace 20 AM ASICs with 7 kRAM blocks and 2-6M ALM → x3/x10 → full ITk coverage with 1500-5000 accelerators
- Full Hough transform implemented
  - URAM is limiting resource
  - Currently estimate 3-6 accelerators to replace 20 AM ASICs → 1500-3000 accelerators

Adding All Up

Task	Accelerators
Clustering	100
Track Finding	400
Track Fitting	1500-3000
Total	2000-3500
Power (200w/board)	1.5-2.5 MW [incl. farm power]

- To be compared with
  - 2.5-15 MW in the commercial CPU option
  - ~25 racks and ~500 kW in custom electronics (ASIC+FPGA)

Conclusions

- FPGA accelerators are entering data centers
  - High Level Synthesis tools are key for exploitability, but
    - still vendor-specific
    - optimization requires deeper knowledge of device features
- Current devices are reaching comparable performance to custom electronics
  - Higher costs
  - Scalability with tech evolution
- Several HEP experiments are looking at possible use in farms as well as real-time environments
  - We are definitely not at the forefront in this: ML, real-time transactions, Bitcoin mining…