The RETINA algorithm

Giovanni Punzi
University/INFN-Pisa

Luciano Ristori
Fermilab

DATA SCIENCE @LHC 2015
CERN, November 9-13, 2015
Introduction

- LHC has opened a new era in HEP - also for data processing
- Exploitation of the upcoming High-Luminosity LHC phase will pose even greater challenges
- Data reconstruction and storage will be the toughest issues
- A big part of the problem is reconstruction of charged particle trajectories – this is what I will concentrate upon

- Large combinatorial problem, calls for high parallelization
- In many cases, latencies are an issue due to need for buffering
Past Experience in HEP

- Trigger, DAQ, Computation, data storage – have been part of HEP since its earliest days, in one form or another
- Complexity and computational load kept increasing
- At the same time, advancements in electronics technology caused huge price/performance drops
- Nevertheless:
  → Real-time processing still today a major cost item of experiments
  → In many cases a major technical constraint

- What about the Future?

Indications that HEP will face a computing roadblock
A summary view of Data Processing in HEP

[S. Cittolin, Phil. Trans. R. Soc. A 2012 370]
The LHC (phase-1) era

- Most features of trigger systems already present in pre-LHC era, **but**:  
  - Larger data volume, although event rate not much larger  
  - Exploit the explosion of internet → telecom technology  
  - Conscious effort to exploit “commercial off-the-shelf “products to the maximum possible extent  
  - Increasing resources in event-building  
  - Moving most of the complexity to the latest stage
- Early processing did not increase in rejection power or sophistication - used traditional HEP Ingredients:
  - Timing, calorimetric sums, muons...(gave up tracking)  
  - LHC traded sophistication for size, cost, and simplicity

**But there are reasons to expect that further progress will require bigger steps forward**
In spite of increasing DAQ bandwidth and storage availability, the need for large data reduction factors to permanent storage keeps getting stronger.

Evolution of computing not necessarily going as fast as in the past

Problem compounded by the needs dictated by physics:
- Precision measurements becoming more important
- Event structure more complex ("pile-up") even at constant rate.
  → Need more computing power to take the same decisions
  → Need to feed more data into each decision
  → Implications for DAQ: need larger B/W to the trigger

Today no obvious solutions for doing such things at HL-LHC \( L=5\times10^{34} \)
Pressures on Real-Time computing

- Limit Storage
- Reduce Event size
- Higher rejection
- Increase Analysis quality
- Larger samples
- Precision physics
- Higher Pileup

Need more Real-Time Computing Power
HEP Trends, and Level-1

- Level-1 traditionally based on simple quantities, that can be calculated fast, to give an easy and cheap way to reduce rate. (“The HIGH-PT paradigm”: look only at the hardest events)
- Future: More complex physics, ”precision physics”, and more events at the same time => No more an easily-extracted, smaller portion of the event data that can be used to reduce data for a more detailed processing later.
  - Examples: ...LHCb has signal events at every collisions, CMS need to reduce data from the tracker to read it out...
  - In the future, all SM physics will be “low-Pt physics”
    - At FCC, the rate of top events will be 3kHz...
→ will need to actually process all data from each crossing.
    - with larger event size...
→ CMS and ATLAS are coping by developing real-time tracking systems intended to operate at level 1 – this is tough!
A summary view of Data Processing in HEP

This is NOT
The full rate!  
$>10^2$ reduction by Level-1 pre-selection

Going to be more of a challenge in future

[S. Cittolin, Phil. Trans. R. Soc. A 2012 370]
The issue with tracking at first level of processing

- It is “true real-time”: latency and local data availability requirements carry a weight
- Greater specificity to the detector structure, tighter optimization
- Less amenable to “plug-and-play” commercial solutions.
  - Requires larger development time, specialists
  - Less commonality with other solutions
  - In the old times it was implemented in “hardware”
  - Now the distinction between hardware and software is much more blurred... electronics boards are typically completely programmable in software, although the software may be more application-specific
  - More than anything else, architecture matters. Design not made of procedures, but of structures (happening to general-purpose software as well, where parallelization requires programmers to think about actual execution)

- Ideal goal: “detector-embedded” reconstruction of complex primitives – making the rest of computation more manageable. A tracker device producing tracks, not hits
Track reconstruction by pattern-matching in HEP

- The fastest approach to tracking that has been used up to now is direct matching to a bank of stored templates.

- First large system to use this method has been CDF, at the Tevatron, where a real-time processor named SVT was capable of reconstructing quality tracks in ~10µs.

- Based on custom ASICs implementing content-addressable memory (Associative Memory [NIM A278, (1989), 436-440])

- It actually worked! Allowed CDF to discover Bs oscillations (amongst other things)

- This same approach is continuing in FTK for ATLAS and in the planned Phase 2 upgrade for CMS
Track reconstruction by pattern-matching using “Associative Memory”

A pattern is a sequence of hits in the different layers, represented by coordinates. A particle trajectory is a specific sequence of hits. Hit are read out sequentially, and compared in parallel to a set of pre-calculated “track patterns” - NO combinatorics.

Track parameters found in a 2\textsuperscript{nd} step (more sequential, but fast if you used enough AM cells in the first stage)
Successful past examples of real-time tracking by pattern-matching

<table>
<thead>
<tr>
<th>Name</th>
<th>Tech.</th>
<th>Exp.</th>
<th>Year</th>
<th>Event rate</th>
<th>clock</th>
<th>cycles/event</th>
<th>latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>XFT</td>
<td>FPGA</td>
<td>CDF-L0</td>
<td>2000</td>
<td>2.5 MHz</td>
<td>200 MHz</td>
<td>80</td>
<td>&lt;4µs</td>
</tr>
<tr>
<td>SVT</td>
<td>AM</td>
<td>CDF-L2</td>
<td>2000</td>
<td>0.03 MHz</td>
<td>40 MHz</td>
<td>~1600</td>
<td>&lt;20µs</td>
</tr>
<tr>
<td>FTK</td>
<td>AM</td>
<td>ATLAS-L2</td>
<td>2015</td>
<td>0.1 MHz</td>
<td>~200 MHz</td>
<td>~2000</td>
<td>O(10µs)</td>
</tr>
</tbody>
</table>

Compare with the requirements of a L0@LHC:

|     |     | LHC-L0 | ~2020 | 40MHz | ~1GHz | ~25 | few µs |

The task of L0 tracking at LHC appears daunting despite the progress of electronics.

- Any complex tracking calls for $O(10^3)$ clock cycles/event in latency and throughput (still much faster than CPUs)
- No known example of a system making non-trivial pattern reconstruction in $O(25)$ time units

Maybe just an impossible task?
Inspiration from “Natural computing”: comparing natural vision with HEP

Many similarities:

• Lots of complex data/combinatorics
• Little time available
• Pressure to make accurate decisions
• Strongly constrained computing resources
A look at size and timing of the natural vision system

- The early visual areas in human brain produce a recognizable sketch of the image at 30-40Hz, with latencies <100ms

> $10^9$ neurons for vision, typical switching time ~1ms.
Complex tracking calls for $O(10^3)$ clock cycles/event (both in latency and throughput) – Vision works within just $\sim 25$

If we could do the same in an electronics device, we could easily do **real-time tracking of every LHC collision**: 25 cycles@1GHz → 25ns : 40MHz

The scaled flow of data would be **5 Pb/s** – enough for a huge detector

Brain outperforms HEP triggers greatly - WHY ?

---

**Performance: Natural vs Man-made**

<table>
<thead>
<tr>
<th>Name</th>
<th>Tech.</th>
<th>Exp.</th>
<th>Year</th>
<th>Event rate</th>
<th>clock</th>
<th>cycles/event</th>
<th>latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVT</td>
<td>AM</td>
<td>CDF-L2</td>
<td>2000</td>
<td>0.03 MHz</td>
<td>40 MHz</td>
<td>$\sim 1600$</td>
<td>&lt;20µs</td>
</tr>
<tr>
<td>FTK</td>
<td>AM</td>
<td>ATLAS-L2</td>
<td>2014</td>
<td>0.1 MHz</td>
<td>$\sim 200$ MHz</td>
<td>$\sim 2000$</td>
<td>O(10µs)</td>
</tr>
<tr>
<td>Vision (neural) (Brain)</td>
<td>old</td>
<td>~40 Hz</td>
<td>~1kHz</td>
<td>$\sim 25$</td>
<td>&lt;100ms</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**What is so special about the “brain algorithm” ?**

- Parallelism, of course - but Associative Memories are very parallel devices as well...
- Some important differences, though:
  - Hit processing in AM cells still happens serially, while in the visual system only relevant data reaches a cell. This is faster, and allows processing power to be **spread over a network**.
  - The AM has “rigid templates” with yes/no response, while the brain works by **interpolation of analog responses**. This saves internal storage and makes it easier to deal with “missing information”.

**Could these features be implemented in a viable artificial device ?**

**Is it at all possible to replicate neural-like algorithmic in electronics ?**

**Investigating these questions is the goal of the “RETINA project”**
One-slide digression: Sheila Nirenberg's retina encoder

www.pnas.org/cgi/doi/10.1073/pnas.1207035109

- Functionality of retinal circuitry was measured and replicated in standard digital devices
- Application to vision prosthetic being developed
- Different from our purpose – but suggestive
The RETINA project
https://web2.infn.it/RETINA

- R&D program supported by INFN CSN5 (Technological research division)

- Goal: study the possibility to build a specialized track processor based on a vision-like architecture and evaluate its performance for tracking in LHC environment

- Specialization is important: the success of GPUs stems from specialization for a narrow purpose. Our aim is to build something that does for Tracking what the GPU did for Graphics (just with a smaller market...) (a “TPU”).

- Not intended to replicate vision in detail - just exploit similar design principles.
Implementing a “neural-like” tracking algorithm

Each cell performs a weighted sum of hits “in the vicinity” (“graduated response”).

Moving beyond AM’s yes/no response allows using fewer cells, and yields immediate parameter estimates

Response of each cell is summed over all hits

\[ R = \sum_{\text{all hits}} e^{-\frac{s_i^2}{2\sigma^2}} \]
Implementing a “neural-like” tracking algorithm

- A valid track appears as a cluster of cell responses – parameters can then be determined by interpolation of nearby cells.

- First work in this direction in year 2000 [L. Ristori, “An Artificial retina for Fast Track Finding” NIM A453 (2000) 425-429] (historical reason for the name, although today we believe most of this processing actually happens in the primary visual cortex areas)

Implementing a “neural-like” tracking algorithm

- Toy-model study, using simple 2-D tracker (figure from original 2000 paper)

Fig. 1. Simulation of the response of the receptor array.
System Architecture is crucial

- Tracking layers
- Separate trigger-DAQ path
- Custom switching network delivers hits to appropriate cells
- Data organized by cell coordinates
- Blocks of cellular processors
- Track finding and parameter determination
- To DAQ
Hit delivery via programmable switch logic

- Hits must be delivered only to the cells that need them (there can be more than one)
- Switch network “knows” where to deliver hits
- All information embedded in the network via distributed LUTs

Data processing happens while data is being moved - not afterwards
Building a large custom switching network from uniform elementary blocks
**Cellular computing engine working principle**

Each node:
- Performs calculation of weights for a hit into a cell
- Handles time-skew between events

In second stage:
- Deals with surrounding cells → local clustering
- Queues results to output

All the above happens in pipeline without stops (data-flow)
Final stage: Parameter extraction

- Two (or 3!) parameters can be extracted directly from cluster centroid in 2D array of cells.

How about other 2 or 3 parameters?

- Add “lateral cells” and interpolate their response
  (Enough when parameter spread is limited)
- Perform local linearized fit (easy with hardware DSPs)

Tested with up to $3^5 = 243$ cells
(full 5-parameters tracks)
First-order estimation of needed resources: Generic 6-layer pixel tracking detector

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>crossing frequency</td>
<td>40 MHz</td>
</tr>
<tr>
<td>number of hits per layer per crossing</td>
<td>300</td>
</tr>
<tr>
<td>number of bits per hit</td>
<td>24</td>
</tr>
<tr>
<td>total hit bandwidth</td>
<td>1.7 Tb/s</td>
</tr>
<tr>
<td>link bandwidth</td>
<td>10 Gbit/s</td>
</tr>
<tr>
<td>engine cycles/s</td>
<td>1 GHz</td>
</tr>
<tr>
<td>engine cycles to process one hit</td>
<td>450</td>
</tr>
<tr>
<td>computing power required</td>
<td>32 THz</td>
</tr>
<tr>
<td>number of engines</td>
<td>49152</td>
</tr>
<tr>
<td>number of receptors</td>
<td>245760</td>
</tr>
<tr>
<td>latency</td>
<td>0.37µs</td>
</tr>
<tr>
<td>number of events in pipeline</td>
<td>15</td>
</tr>
</tbody>
</table>

→ Quite reasonable parameters for a system to build

HL-LHC crossing freq
Typical estimate for \( L = 10^{33} - 10^{34} \)
Generous allowance
O(100) optical links
Fits in a single FPGA...
Standard business today
(Turned out to be greatly overestimated, see later)

Nothing to be scared of.
50k cells: compare to millions of AM cells
Sub-µs latency. Fits well with typical LHC level-1
The bandwidth profile issue

- HEP DAQ typically works by progressively reducing the data bandwidth (funnel-like).

- The RETINA approach needs to increase the data flow in the initial stage, by making multiple data copies, and then the bandwidth is shrunk back to lower values when the maxima location is found.

- Curiously enough: evidence of similar process in the brain visual path.

- The process is dependent on the geometry of the tracking detector:
  - Correlated information between layers helps a lot
  - e.g. CMS’s double-layers
  - Possible future time-tagged hits

Best to build detector with Data-Processing in mind
Simulation studies on a HEP detector

- Multi-layer, precision pixel detector
- 3D reconstruction of tracks
- Take two parameters from intersection of tracks with a reference plane
- These two parameters can then be mapped to a 2D main grid
- Remaining track parameters implemented in a separate step
Getting down to details: Detector mapping to “retina” cells

- Intersection of “base tracks” with detectors gives a map of “nerve endings”
- Every hit on the detector produces a signal on nearby receptors, depending on distance
- Effective operation requires distribution to be non-uniform for load-balancing (curiously biologically-looking)
- It is all virtual in our case, that is, implemented in the network connections.
High-level simulation in C++:
Cell activation map of typical multi-track event
Tracking performance checks

**EFFICIENCY/UNIFORMITY**
Equivalent to offline reconstruction (fake track rates equivalent as well)

**MOMENTUM RESOLUTION**
Very close to offline.

Promise of quality reconstruction at LHC crossing frequency
Implementation Considerations

Most promising and accessible medium: large state-of-the-art FPGA devices.

• Large I/O capabilities: now $O(Tb/s)$ with optical links!
• Large internal bandwidth (a must!)
• Distributed computing resources: DSP slices, SoC...
• Low power consumption → critical in the current computing era
• Fully flexible, easy (!) to program and simulate in software
• Steep Moore's slope, easily upgradable
• Highly reliable, easy to maintain and update

→ Industry's method of choice for complex projects for small productions (CT scanners, high-end radars...), low-latency (finance, military)
Reality check: other experiences with custom-designed processing in FPGAs

Table 3.

Calculation time comparison.

<table>
<thead>
<tr>
<th>Algorithm and Platform</th>
<th>Execution Time</th>
<th>Processing Image Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSM of Ji et al. [3] on FPGA</td>
<td>15.57 ms</td>
<td>1,024 × 768</td>
</tr>
<tr>
<td>Chen et al. [40] on FPGA</td>
<td>2.07–3.61 ms</td>
<td>512 × 512</td>
</tr>
<tr>
<td>Proposed Method on FPGA</td>
<td>15.59 ms</td>
<td>1,024 × 768</td>
</tr>
<tr>
<td>Direct HT Computation on PC</td>
<td>(a-1) 0.93 s</td>
<td>1,024 × 768</td>
</tr>
<tr>
<td></td>
<td>(a-2) 1.26 s</td>
<td>1,024 × 768</td>
</tr>
<tr>
<td></td>
<td>(a-3) 1.62 s</td>
<td>1,024 × 768</td>
</tr>
<tr>
<td></td>
<td>(a-4) 1.45</td>
<td>1,024 × 768</td>
</tr>
</tbody>
</table>

Speedup factors of 70÷500 regularly obtained in vision, military, finance applications.
FPGA implementation, Timing simulation

- Processing time depends only on # of hits in the event - Results always available after fixed number of cycles
  - Time between hit delivery and accumulator update
  - Time between end sequence and accumulator output

- Turns out ~20 clock cycles are sufficient
- Require 1 – 5 kLE of logic → $O(10^3)$ cells/average FPGA
- Can build tracker with $O(100)$ medium-size FPGAs

Processing time depends only on # of hits in the event - Results always available after fixed number of cycles.
Boards based on 4 Stratix-IV ALTERA FPGAs
Events processed in the boards and bit-level checked with C++ simulation.
Reconstruction rate: 1.8 MHz (160MHz clock): <100 clock periods/event (board architecture not optimized for the purpose!)
Normal readout-only operation 1MHz - implication is track reconstruction in principle doable “on the fly” while reading detector

Reality check: implement within typical HEP frameworks: Lab Test with NA62 DAQ boards (TEL62)

Not a simulation: Real board signals
Boards based on 4 Stratix-IV ALTERA FPGAs
Events processed in the boards and bit-level checked with C++ simulation.
Reconstruction rate: 1.8 MHz (160MHz clock): <100 clock periods/event (board architecture not optimized for the purpose!)
Normal readout-only operation 1MHz - implication is track reconstruction in principle doable “on the fly” while reading detector

Next steps:
- Larger system
- Full-speed system with purposedly built boards
Summary

○ Future HEP experiments will increasingly depend on large computing power

○ A key to progress will be the capability of real-time reconstruction by special-purpose processors.

○ RETINA project aimed at designing better real-time tracking processors using architectures inspired by natural vision

○ Encouraging preliminary results may lead to a HEP future with detector-embedded data reconstruction
Bibliography

L. Ristori, An artificial retina for fast track finding, NIM A 453 (425-429), http://inspirehep.net/record/539203
M. M. Del Viva, G. Punzi, D. Benedetti, Information and Perception of Meaningful Patterns [PDF]
S. Stracka, A specialized processor for track reconstruction at the LHC crossing rate, Talk at Connecting The Dots Workshop 2015, https://indico.physics.lbl.gov/indico/getFile.py/access?contribId=14&sessionId=2&resId=0&materialId=slides&confId=149
R. Cenci, Artificial retina processor for track reconstruction, Talk at Connecting The Dots Workshop 2015, https://indico.physics.lbl.gov/indico/getFile.py/access?contribId=2&sessionId=9&resId=0&materialId=slides&confId=149
A. Abba et al., Progress Towards the First Prototype of a Silicon Tracker Using an 'Artificial Retina' for Fast Track Finding, Poster at TWEPP14, https://indico.cern.ch/event/299180/session/7/contribution/64
A. Abba et al., The artificial retina for track reconstruction at the LHC crossing rate, arXiv:1411.1281 [ICHEP 2014], https://inspirehep.net/record/1326137.
A. Abba et al, Simulation and performance of an artificial retina for 40 MHz track reconstruction, JINST 03-10 (C03008) [WIT2014], https://inspirehep.net/record/1314984.
A. Abba et al., A Specialized Processor for Track Reconstruction at the LHC Crossing Rate, JINST 9 (C09001) 2014 [INSTR14], https://inspirehep.net/record/1303542.