Particle Physics in Real Time

Benjamin Nachman
Lawrence Berkeley National Laboratory

Real-Time Decision Making Reunion, June 2, 2018
Goal: **We want to study the structure of the smallest building blocks of matter.** For this, we need the most powerful microscope ever built!

![Diagram](image)

- Proton radius: $10^{-14}$ meters
- Atomic radius: $10^{-11}$ meters
- Human hair: $10^{-3}$ meters

**Large Hadron Collider (LHC)**: 2010+

- Quark discovery at Stanford: 1970
- Rutherford's gold foil Experiment: 1910
- Optical microscope
- Electron microscope
- STM
- Human eye
O(100) $pp$ collisions

(sub-)nuclear physics

out-going particles interact with detector

detector response (signal formation + digitization)

hardware-based trigger decision

software-based trigger decision

event reconstruction

event processing (skim, thin, augment)

final data analysis (uses millions of events)

25 ns

$10^{-19}-10^{-15}$ s

$\sim$ms

0.01-20 ns

$\sim$min

1-100 ns

$\sim$100 ms

2.5 $\mu$s

200 ms

$\sim$100 ms

$\sim$months

~few TB/s (99% thrown away in real time)
Data pipeline at the LHC

- 25 ns
- $10^{-19}$-$10^{-15}$ s
- 0.01-20 ns
- 1-100 ns
- 2.5 μs
- 200 ms
- ~100 ms
- ~months

$O(100)$ $pp$ collisions

(sub-)nuclear physics

Out-going particles interact with detector

Detector response (signal formation + digitization)

Hardware-based trigger decision

Software-based trigger decision

Event reconstruction

Event processing (skim, thin, augment)

Final data analysis (uses millions of events)

~few TB/s (99% thrown away in real time)
Collider-based HEP detectors are like leeks

~100 million readout channels
First decisions have to be on-detector and only with local information.

Detector is not constant with time - real time online calibration

Want to use offline-like algorithms as soon as possible

…but don’t have time or resources to run all our deepest NN’s, etc.
Real time tools

Fast decisions with incomplete information (ASICs)

*On-detector, radiation hard and ultra fast - single purpose hardware*

Fast decisions with full information (FPGAs)

*Off-detector, re-programmable*

(less) Fast decisions with full information (Software)

*Far off-detector, offline-like algorithms*
Fast decisions with incomplete information (ASICs)

*On-detector, radiation hard and ultra fast - single purpose hardware*

Fast decisions with full information (FPGAs)

*Off-detector, re-programmable*

(less) Fast decisions with full information (Software)

*Far off-detector, offline-like algorithms*
Innermost layer: high bandwidth, hit rate, rad. damage

\[ \text{GHz/cm}^2 \sim 0.1\%\text{/pixel/BC} \]
\[ \text{Gbps/cm}^2 \sim \text{streaming live audio from each pixel} \]

1 Grad (TID) and \(10^{16} \text{n}_\text{eq}/\text{cm}^2\) (NIEL)
# Innermost layer: high bandwidth, hit rate, rad. damage

<table>
<thead>
<tr>
<th>Generation</th>
<th>Run 1  (FEI3, PSI46)</th>
<th>Runs 2+3 (FEI4, PSI46DIG)</th>
<th>Runs 4+5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chip Size</td>
<td>7.5 x 10.5 mm$^2$</td>
<td>20 x 20 mm$^2$</td>
<td>&gt; 20 x 20 mm$^2$</td>
</tr>
<tr>
<td></td>
<td>8 x 10 mm$^2$</td>
<td>8 x 10 mm$^2$</td>
<td></td>
</tr>
<tr>
<td>Transistors</td>
<td>3.5 M 1.3 M</td>
<td>87 M</td>
<td>~1 G</td>
</tr>
<tr>
<td>Hit Rate</td>
<td>100 MHz/cm$^2$</td>
<td>400 MHz/cm$^2$</td>
<td>~2 GHz/cm$^2$</td>
</tr>
<tr>
<td>Hit Memory / Chip</td>
<td>0.1 Mb</td>
<td>1 Mb</td>
<td>~16 Mb</td>
</tr>
<tr>
<td>Trigger Rate</td>
<td>100 kHz</td>
<td>100 kHz</td>
<td>200 kHz - 1MHz</td>
</tr>
<tr>
<td>Trigger Latency</td>
<td>2.5 μs 3.2 μs</td>
<td>2.5 μs 3.2 μs</td>
<td>6 - 20 μs</td>
</tr>
<tr>
<td>Readout rate</td>
<td>40 Mb/s</td>
<td>320 Mb/s</td>
<td>1-4 Gb/s</td>
</tr>
<tr>
<td>Radiation</td>
<td>100 Mrad</td>
<td>200 Mrad</td>
<td>1 Grad</td>
</tr>
<tr>
<td>Technology</td>
<td>250 nm 130 nm 250 nm</td>
<td>65 nm</td>
<td></td>
</tr>
<tr>
<td>Power</td>
<td>~1/4 W/cm$^2$</td>
<td>~1/4 W/cm$^2$</td>
<td>1/2 - 1 W/cm$^2$</td>
</tr>
</tbody>
</table>
Innermost layer: high bandwidth, hit rate, rad. damage

e.g. the camera in your phone on steroids, next to a nuclear reactor (unfortunately, Apple doesn’t make one of these)

<table>
<thead>
<tr>
<th>Generation</th>
<th>Run 1 (FEI3, PSI46)</th>
<th>Runs 2+3 (FEI4, PSI46DIG)</th>
<th>Runs 4+5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chip Size</td>
<td>7.5 x 10.5 mm²</td>
<td>20 x 20 mm²</td>
<td>&gt; 20 x 20 mm²</td>
</tr>
<tr>
<td></td>
<td>8 x 10 mm²</td>
<td>8 x 10 mm²</td>
<td></td>
</tr>
<tr>
<td>Transistors</td>
<td>3.5 M</td>
<td>87 M</td>
<td>~1 G</td>
</tr>
<tr>
<td></td>
<td>1.3 M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hit Rate</td>
<td>100 MHz/cm²</td>
<td>400 MHz/cm²</td>
<td>~2 GHz/cm²</td>
</tr>
<tr>
<td>Hit Memory / Chip</td>
<td>0.1 Mb</td>
<td>1 Mb</td>
<td>16 Mb</td>
</tr>
<tr>
<td>Trigger Rate</td>
<td>100 kHz</td>
<td>100 kHz</td>
<td>200 kHz - 1 MHz</td>
</tr>
<tr>
<td>Trigger Latency</td>
<td>4.9 µs</td>
<td>2.5 µs</td>
<td>6 - 20 µs</td>
</tr>
<tr>
<td></td>
<td>3.2 µs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Readout rate</td>
<td>40 Mb/s</td>
<td>320 Mb/s</td>
<td>1-4 Gb/s</td>
</tr>
<tr>
<td>Radiation</td>
<td>100 Mrad</td>
<td>200 Mrad</td>
<td>1 Grad</td>
</tr>
<tr>
<td>Technology</td>
<td>250 nm</td>
<td>130 nm</td>
<td>65 nm</td>
</tr>
<tr>
<td></td>
<td></td>
<td>250 nm</td>
<td></td>
</tr>
<tr>
<td>Power</td>
<td>~1/4 W/cm²</td>
<td>~1/4 W/cm²</td>
<td>1/2 - 1 W/cm²</td>
</tr>
</tbody>
</table>
Pixels at the heart of the detector

Pixel ASIC for the innermost layer of the LHC detectors

11.8 mm ; 192 pixels
50 x 50 µm² pixels
20 mm ; 400 pixels
Specific RTDM Problem: Find hits, ignore noise

- Top down view of pixel detector
- iid Gaussian noise in each pixel
- Rate requirement on what we can read out
- Record only what exceeds a threshold
Specific RTDM Problem: Find hits, ignore noise

Top down view of pixel detector

iid Gaussian noise in each pixel

rate requirement on what we can read out

record only what exceeds a threshold

real particle goes through the detector
Specific RTDM Problem: Find hits, ignore noise

charge deposited by the particle
Specific RTDM Problem: Find hits, ignore noise

apparent charge after adding noise
Specific RTDM Problem: Find hits, ignore noise

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2.1</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>9.6</td>
<td>3.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

apparent charge after charge sharing
(diffusion + electronics are capacitively coupled)
Specific RTDM Problem: Find hits, ignore noise

charge over threshold that is **observed**

(threshold = 2.5)
Specific RTDM Problem: Find hits, ignore noise

charge over threshold that is observed (threshold = 2)

Question: can we do better?
Specific RTDM Problem: Find hits, ignore noise

<table>
<thead>
<tr>
<th>Facts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prob(hit from real particle)</td>
</tr>
<tr>
<td>Prob(hit $</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dynamic thresholds?</th>
</tr>
</thead>
</table>

| 0.5 |  
| 7.1 | 0.6 |

charge over threshold that is *observed* (threshold = 2)
Two charge sharing schemes

Option 1: As a result of capacitative coupling, a charge $Q$ on one pixel adds $fQ$ on neighbors. $f$ depends on length of shared edge and is $\sim$ few \%.

One parameter: $f_{\text{share}}$

Usually want this to be small, but maybe can gain by artificially increasing it?

$$Q_{\text{primary}} + fQ_{\text{neighbor}} > \text{threshold}$$

$$Q_{\text{primary}} > \text{threshold} - fQ_{\text{neighbor}}$$

N.B. $\Pr(\text{hit}) \ll 1$ but $\Pr(\text{hit} | \text{neighbor}) \sim 1$ (effectively lowers threshold)
Two charge sharing schemes

Option 2:  Whenever a pixel is above threshold, lower the threshold of the neighbors.

One parameter:  $f_{\text{neighbor}}$

This is hard(er) to implement in practice because it requires more active logic (which means more power &/or more memory)

N.B. these are quite simple, but I’ll show that they work well. Can probably do even better by using less local information.
Charge sharing as a dynamic threshold

Filled: #
Open: charge-weighted

Geant4 (Allpix)
50 x 50 x 150 μm³, ηl = 1.5, 4 bits @ 32 ToT / MIP

Scheme 1: Give a fraction \( f_{\text{share}} \) of your charge to your neighbor

Scheme 2: Set the threshold of your neighbor to \( f_{\text{neighbor}} \) of your threshold.

Measured hits / produced hits
(we call this efficiency)

Noise Rate
Measured hits / produced hits (we call this efficiency)

Filled: #
Open: charge-weighted

Geant4 (Allpix)
50 x 50 x 150 μm³, ηl = 1.5, 4 bits @ 32 ToT / MIP

Scheme 1: Give a fraction \( f_{\text{share}} \) of your charge to your neighbor

Scheme 2: Set the threshold of your neighbor to \( f_{\text{neighbor}} \) of your threshold.

Noise Rate
Measured hits / produced hits (we call this efficiency)

Filled: #
Open: charge-weighted

Geant4 (Allpix)
50 x 50 x 150 μm³, ηl = 1.5, 4 bits @ 32 ToT / MIP

Scheme 1: Give a fraction \( f_{\text{share}} \) of your charge to your neighbor

Scheme 2: Set the threshold of your neighbor to \( f_{\text{neighbor}} \) of your threshold.

Noise Rate
Measured hits / produced hits (we call this efficiency)

Filled: #
Open: charge-weighted

Geant4 (Allpix)
50 x 50 x 150 μm³, ηl = 1.5, 4 bits @ 32 ToT / MIP

Scheme 1: Give a fraction \( f_{\text{share}} \) of your charge to your neighbor

Scheme 2: Set the threshold of your neighbor to \( f_{\text{neighbor}} \) of your threshold.

Noise Rate
Measured hits / produced hits (we call this efficiency)

Filled: #
Open: charge-weighted

Geant4 (Allpix)
50 x 50 x 150 μm³, ηl = 1.5, 4 bits @ 32 ToT / MIP

Scheme 1: Give a fraction \( f_{\text{share}} \) of your charge to your neighbor

Scheme 2: Set the threshold of your neighbor to \( f_{\text{neighbor}} \) of your threshold.

Noise Rate
Measured hits / produced hits (we call this efficiency)

Filled: #
Open: charge-weighted

Geant4 (Allpix)
50 x 50 x 150 μm³, ηl = 1.5, 4 bits @ 32 ToT / MIP

Scheme 1: Give a fraction \( f_{\text{share}} \) of your charge to your neighbor

Scheme 2: Set the threshold of your neighbor to \( f_{\text{neighbor}} \) of your threshold.

Noise Rate
Measured hits / produced hits (we call this efficiency)

Filled: #
Open: charge-weighted

Geant4 (Allpix)
50 x 50 x 150 μm³, ηl = 1.5, 4 bits @ 32 ToT / MIP

Scheme 1: Give a fraction \( f_{\text{share}} \) of your charge to your neighbor

Scheme 2: Set the threshold of your neighbor to \( f_{\text{neighbor}} \) of your threshold.

Noise Rate
Measured hits / produced hits (we call this efficiency)

Filled: #
Open: charge-weighted

Geant4 (Allpix)
50 x 50 x 150 μm³, ηl = 1.5, 4 bits @ 32 ToT / MIP

Scheme 1: Give a fraction \( f_{\text{share}} \) of your charge to your neighbor

Scheme 2: Set the threshold of your neighbor to \( f_{\text{neighbor}} \) of your threshold.

Noise Rate
Measured hits / produced hits (we call this efficiency)

Filled: #
Open: charge-weighted

Geant4 (Allpix)
50 x 50 x 150 μm³, ηl = 1.5, 4 bits @ 32 ToT / MIP

Scheme 1: Give a fraction \( f_{\text{share}} \) of your charge to your neighbor

Scheme 2: Set the threshold of your neighbor to \( f_{\text{neighbor}} \) of your threshold.

Noise Rate
Measured hits / produced hits (we call this efficiency)

Filled: #
Open: charge-weighted

Geant4 (Allpix)
50 x 50 x 150 μm³, ηl = 1.5, 4 bits @ 32 ToT / MIP

Scheme 1: Give a fraction \( f_{\text{share}} \) of your charge to your neighbor

Scheme 2: Set the threshold of your neighbor to \( f_{\text{neighbor}} \) of your threshold.

Noise Rate
Measured hits / produced hits (we call this efficiency)

Filled: #
Open: charge-weighted

Geant4 (Allpix)
50 x 50 x 150 μm³, ηl = 1.5, 4 bits @ 32 ToT / MIP

Scheme 1: Give a fraction \( f_{\text{share}} \) of your charge to your neighbor

Scheme 2: Set the threshold of your neighbor to \( f_{\text{neighbor}} \) of your threshold.

Noise Rate
Measured hits / produced hits (we call this efficiency)

Filled: #
Open: charge-weighted

Geant4 (Allpix)
50 x 50 x 150 μm³, ηl = 1.5, 4 bits @ 32 ToT / MIP

Scheme 1: Give a fraction \( f_{\text{share}} \) of your charge to your neighbor

Scheme 2: Set the threshold of your neighbor to \( f_{\text{neighbor}} \) of your threshold.

Noise Rate
Measured hits / produced hits (we call this efficiency)

Filled: #
Open: charge-weighted

Geant4 (Allpix)
50 x 50 x 150 μm³, ηl = 1.5, 4 bits @ 32 ToT / MIP

Scheme 1: Give a fraction \( f_{\text{share}} \) of your charge to your neighbor

Scheme 2: Set the threshold of your neighbor to \( f_{\text{neighbor}} \) of your threshold.

Noise Rate
Measured hits / produced hits (we call this efficiency)
Charge sharing as a dynamic threshold

Position resolution improves when more information is kept

estimated position = weighted average over hit pixels

Scheme 1: Give a fraction $f_{\text{share}}$ of your charge to your neighbor

Scheme 2: Set the threshold of your neighbor to $f_{\text{neighbor}}$ of your threshold.

Geant4 (Allpix)
50 X 50 x 150 \( \mu \text{m}^3 \), $\eta_l = 1.5$, 4 bits @ 32 ToT / MIP

- **Nominal**
- $f_{\text{share}} = 5\%$
- $f_{\text{neighbor}} = 50\%$

Can’t do worse than
\(~(\text{pixel size})/ \sqrt{12} \) ~ 29\%
Charge sharing as a dynamic threshold

Position resolution improves when more information is kept

Estimated position = weighted average over hit pixels

Scheme 1: Give a fraction $f_{\text{share}}$ of your charge to your neighbor

Scheme 2: Set the threshold of your neighbor to $f_{\text{neighbor}}$ of your threshold.
Charge sharing as a dynamic threshold

Position resolution improves when more information is kept

estimated position = weighted average over hit pixels

Scheme 1: Give a fraction $f_{\text{share}}$ of your charge to your neighbor

Scheme 2: Set the threshold of your neighbor to $f_{\text{neighbor}}$ of your threshold.

Optimal value depends on size of pixels.

Geant4 (Allpix)
150 μm thick, thresh. = 600 e, 4 bits

Resolution / pitch

Charge Sharing [%]
Scheme 1: Give a fraction $f_{\text{share}}$ of your charge to your neighbor

...information transferred ~instantly to neighbors

Scheme 2: Set the threshold of your neighbor to $f_{\text{neighbor}}$ of your threshold.

...need time to tell neighbor to lower threshold.

ToT = time over threshold
Penultimate slide: timeline

**LHC / HL-LHC Plan**

- **Run 1**: 2011-2014
- **Run 2**: 2015-2018
- **Run 3**: 2019-2022
- **Run 4-5**: 2023-2038

**Now**

- **LS1**: 7 TeV splicing consolidation, button collimators, R2E project
- **LS2**: 13 TeV EYETS
- **LS3**: 14 TeV

**Firmware + software updates**

- **Hardware updates**

- **Software updates ~only!**
Conclusions and outlook

The LHC is a unique science tool with extreme challenges related to the data rate: real time / ultra fast algorithms are required.

There are many exciting opportunities and ideas for fully exploiting our data we must make sure no stone is left unturned!
Normalized Pixel Energy Difference

η

Translating

Pseudorapidity

φ

Azimuthal Angle

b → 1, 8

pp = 125 GeV

Fin.