Managed by Fermi Research Alliance, LLC for the U.S. Department of Energy Office of Science

HEP Track Finding with the Micron Automata Processor and Comparison with an FPGA-based Solution

Michael Wang, Gustavo Canelo, Christopher Green, Ted Liu, Ted Zmuda (presented by John Freeman for the authors)

Fermilab

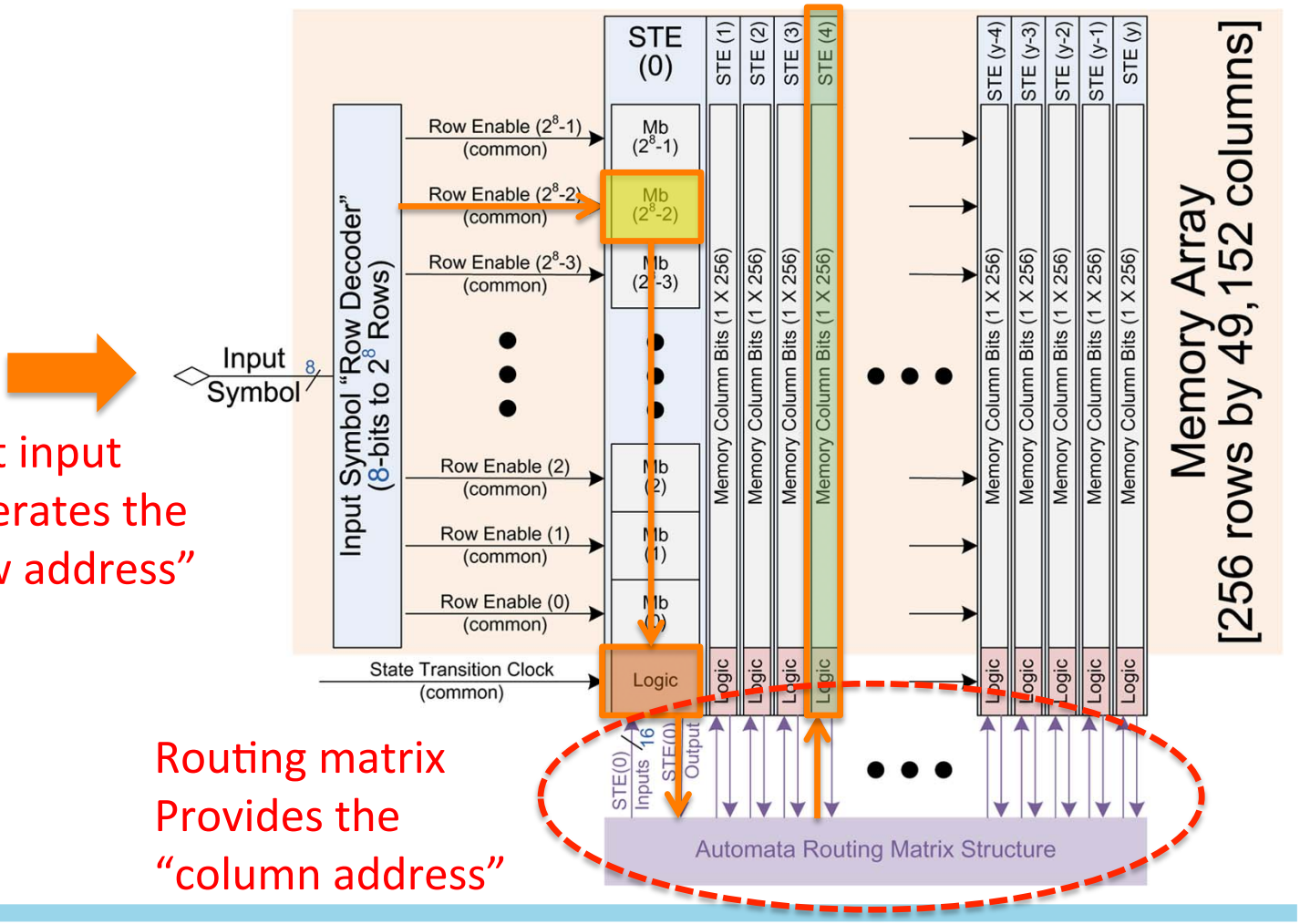
22nd International Conference on Computing in High Energy and Nuclear Physics, San Francisco, California, October 10-14, 2016

Introduction

- Outline of Talk
  - Motivation for investigating the Micron Automata Processor (AP)
  - What is the Automata Processor?
  - Brief description of a proof-of-principle application to investigate the AP's feasibility in HEP track recognition
  - Comparison with CPU, single and multi-threaded
  - Comparison with Content-Addressable-Memory based FPGA implementation
- Motivation for Exploring the Micron AP
  - Trend in HEP experiments, towards more complex event topologies and higher particle densities, makes fundamental task of pattern recognition in HEP more challenging
  - Conventional CPU/GPU architectures becoming less effective as we enter post-Moore's law era.
  - Need to find other off-the-shelf solutions based on novel architectures tuned for high-speed search applications like those in the Internet search industry.
  - Micron's AP is a good candidate

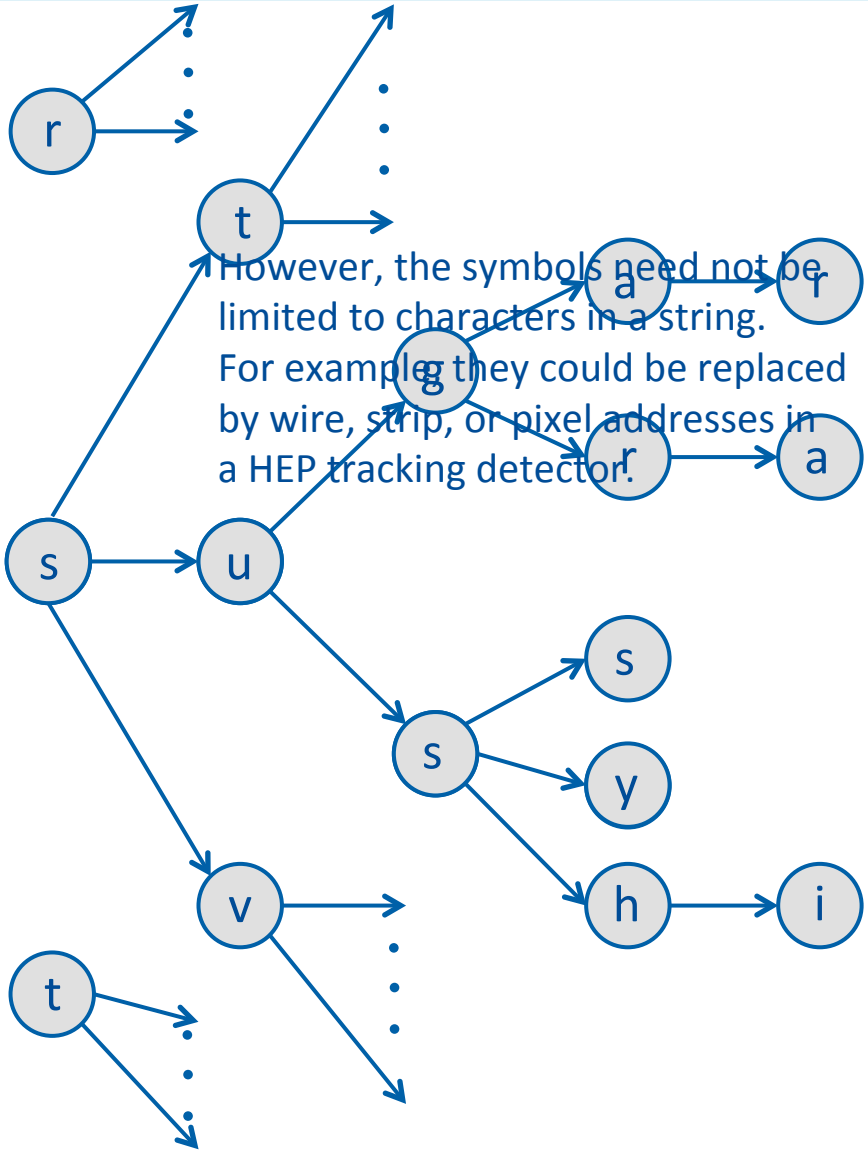
What is the Micron Automata Processor?

- Hardware realization of a Non-deterministic Finite Automata (NFA)
- Interesting adaptation of conventional SDRAM architecture



Fermilab

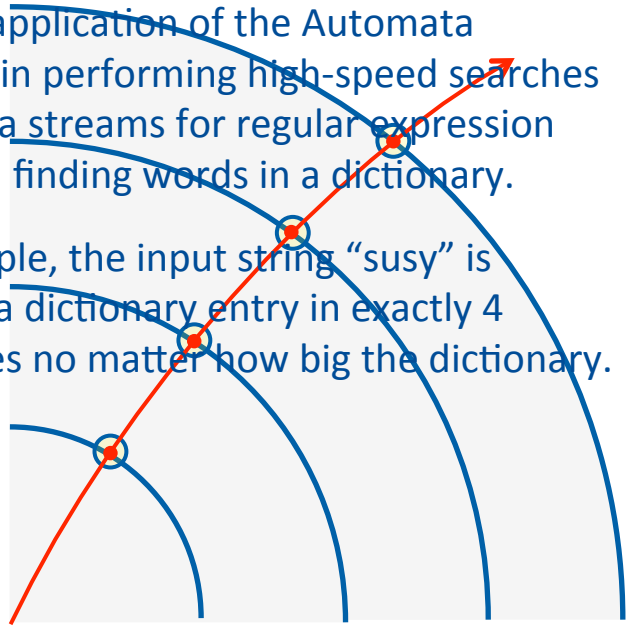
From regular expressions to particle trajectories



However, the symbols need not be limited to characters in a string. For example they could be replaced by wire, strip, or pixel addresses in a HEP tracking detector.

An obvious application of the Automata Processor is in performing high-speed searches on input data streams for regular expression matching or finding words in a dictionary.

In this example, the input string "susy" is matched to a dictionary entry in exactly 4 symbol cycles no matter how big the dictionary.



symbol A - hit address 1 "s u s y"

symbol B - hit address 2

symbol C - hit address 3

symbol D - hit address 4

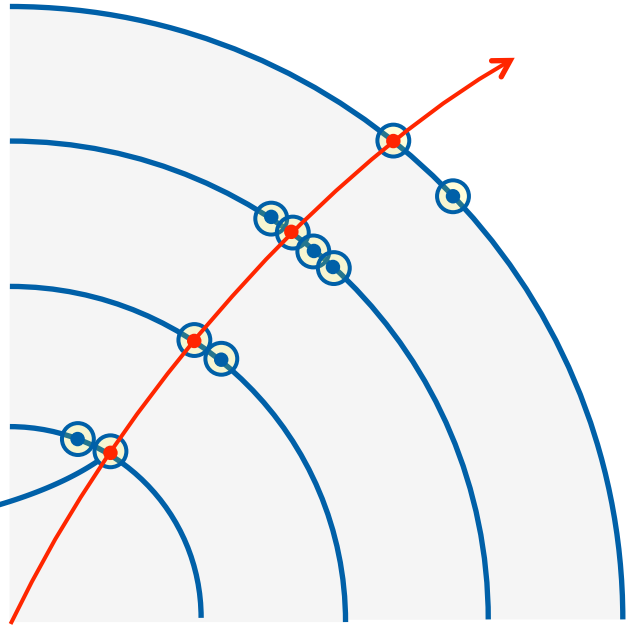
Fermilab

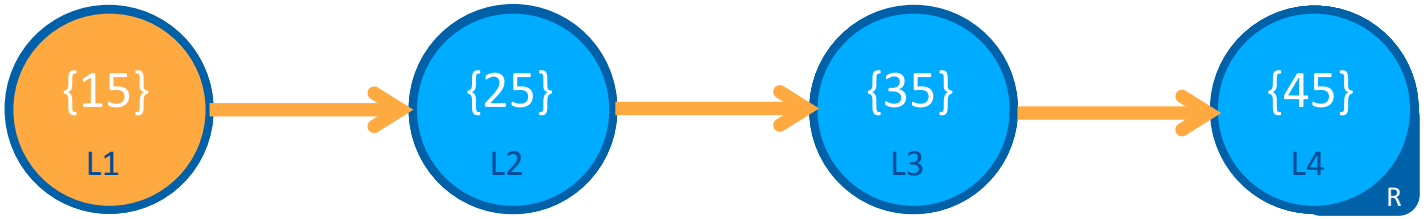
Basic operating principle of an automata track finder

Addresses of detector "hits" read out by layer:

15,17,23,25,31,32,35,36,41,45

Layer 1 Layer 2 Layer 3 Layer 4

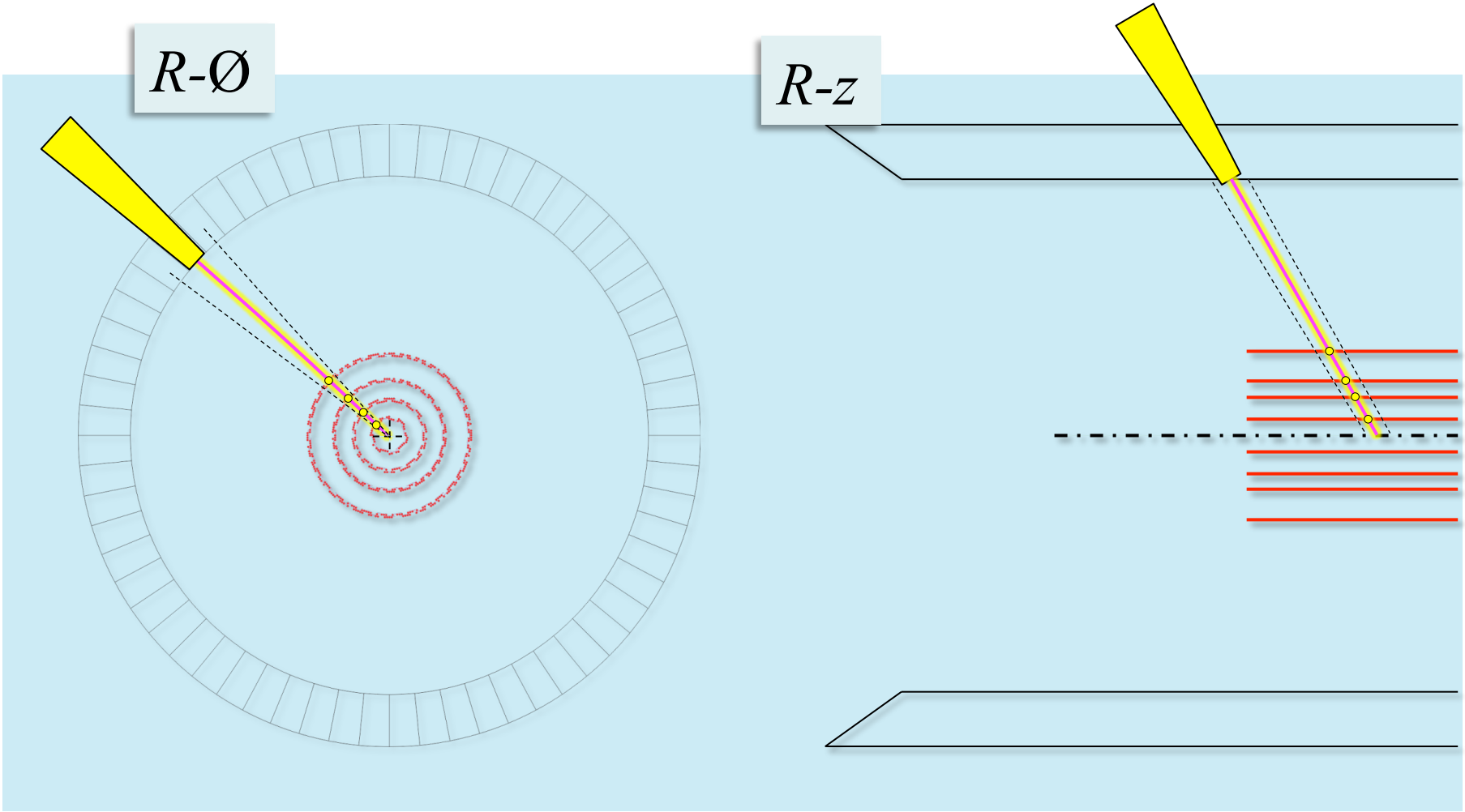




The idea is to create a pattern bank containing every possible track pattern. Each pattern is represented by an Automata network like the one showed above (with latch attributes enabled). Detector hits are fed into the AP sequentially by layer and all hit combinations with matching patterns in the bank are found.

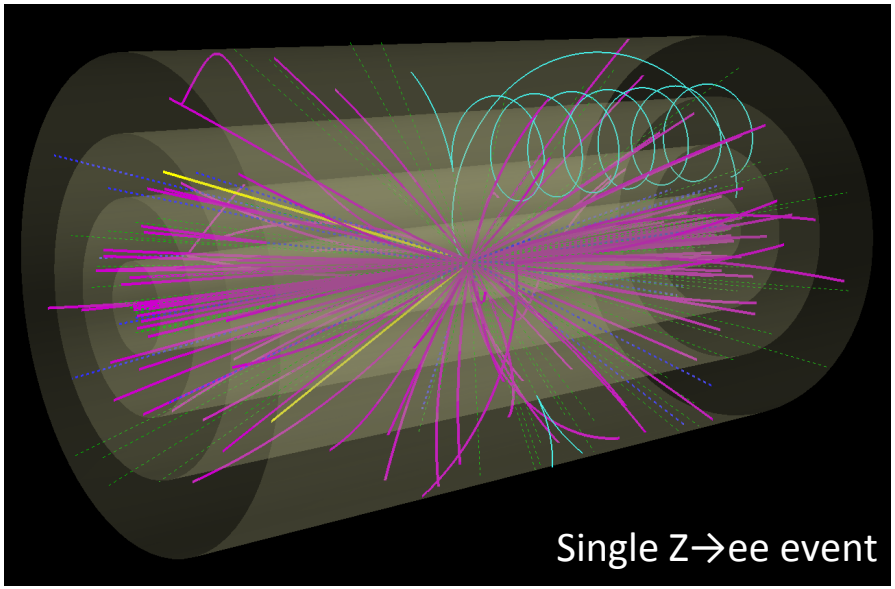
Fermilab

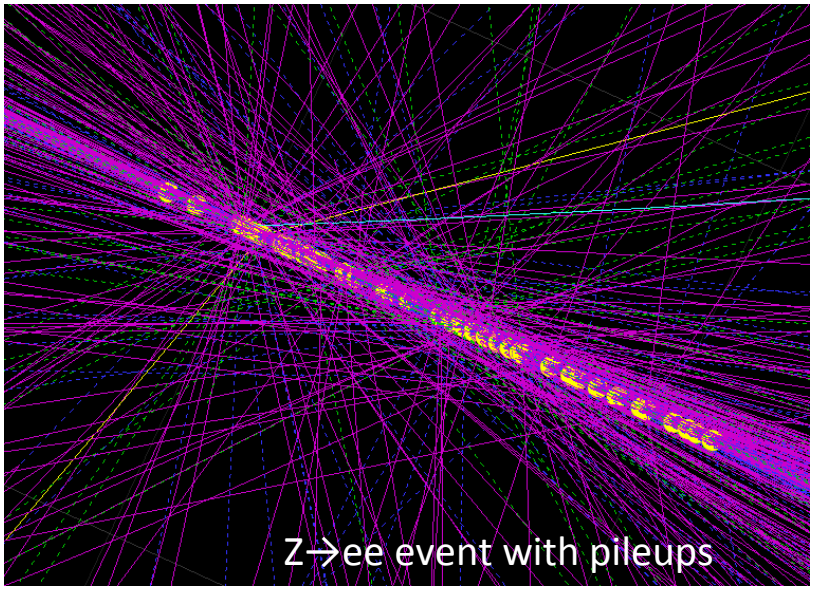
Proof-of-principle application with “toy” CMS Detector

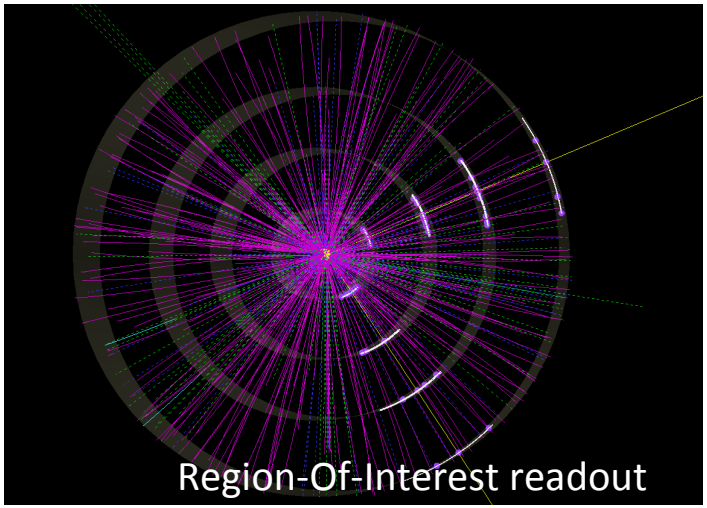


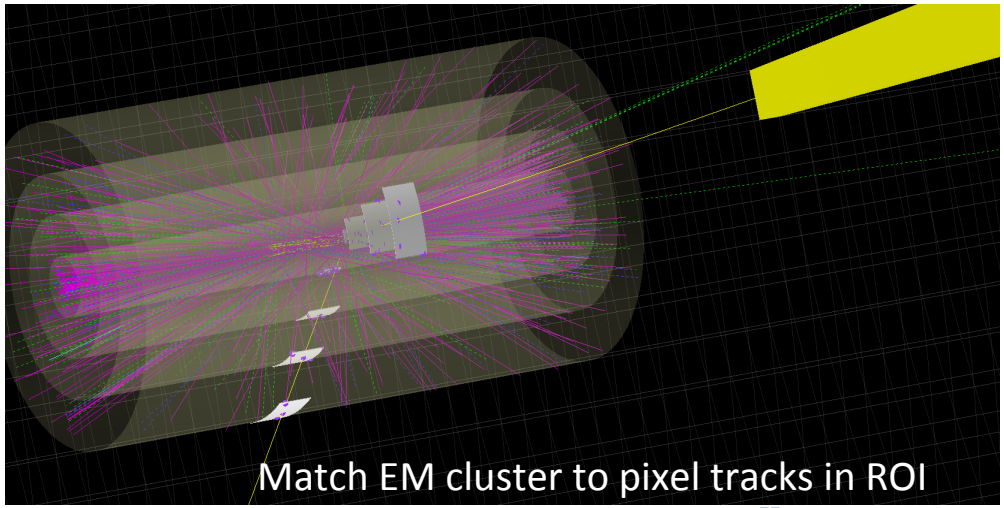
Proof of principle application: Implement a hypothetical pixel detector based electron track confirmation trigger on a “toy” CMS detector

Simulated Events in Toy CMS pixel detector

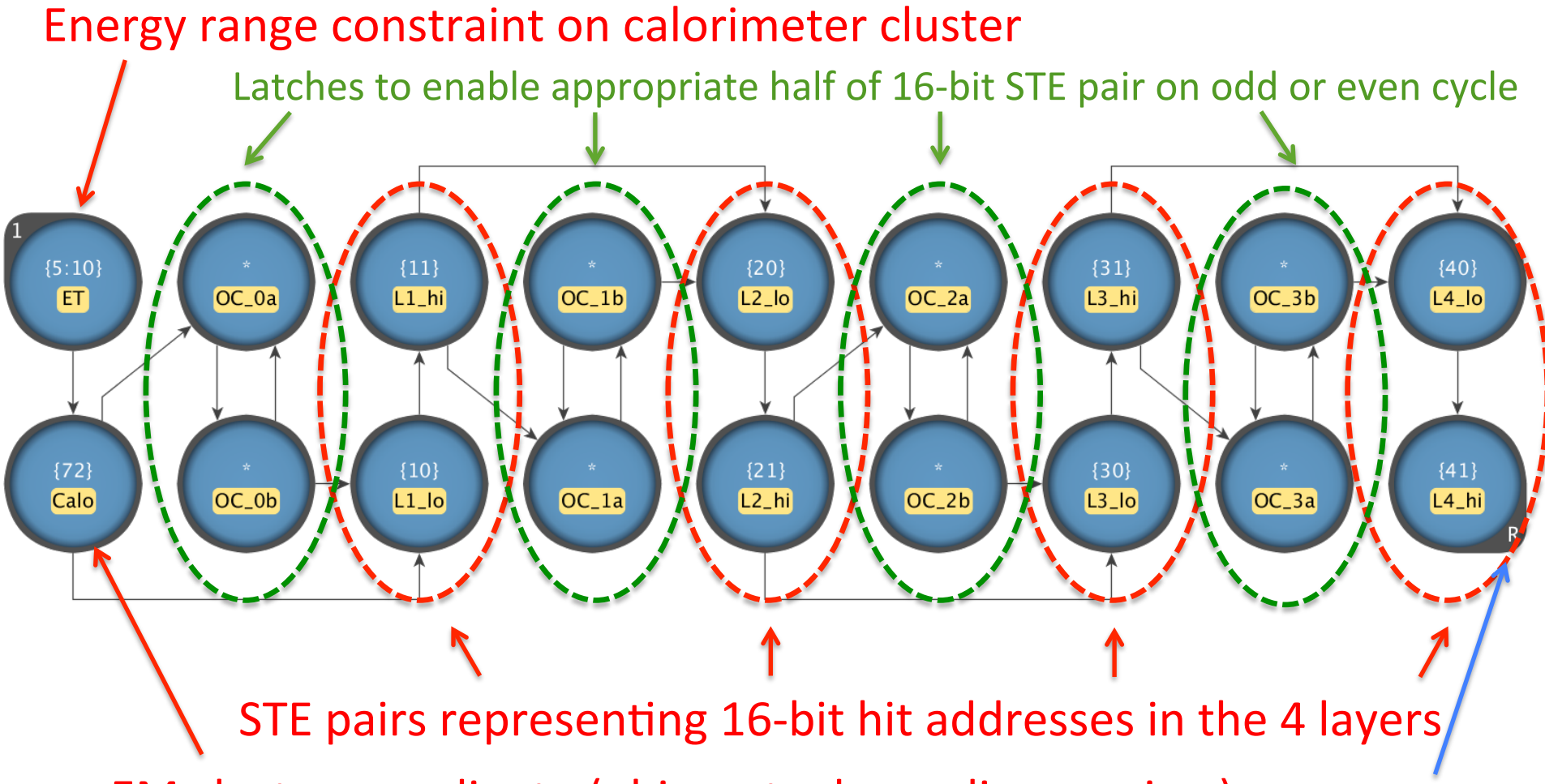








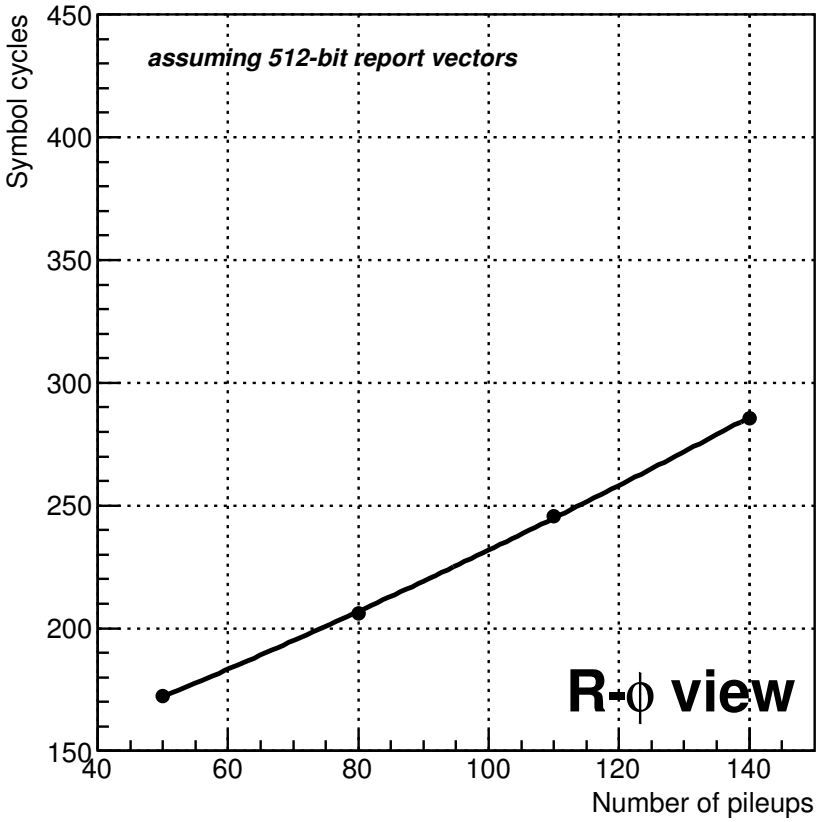
Basic “automaton” for track finder requiring all 4 hits

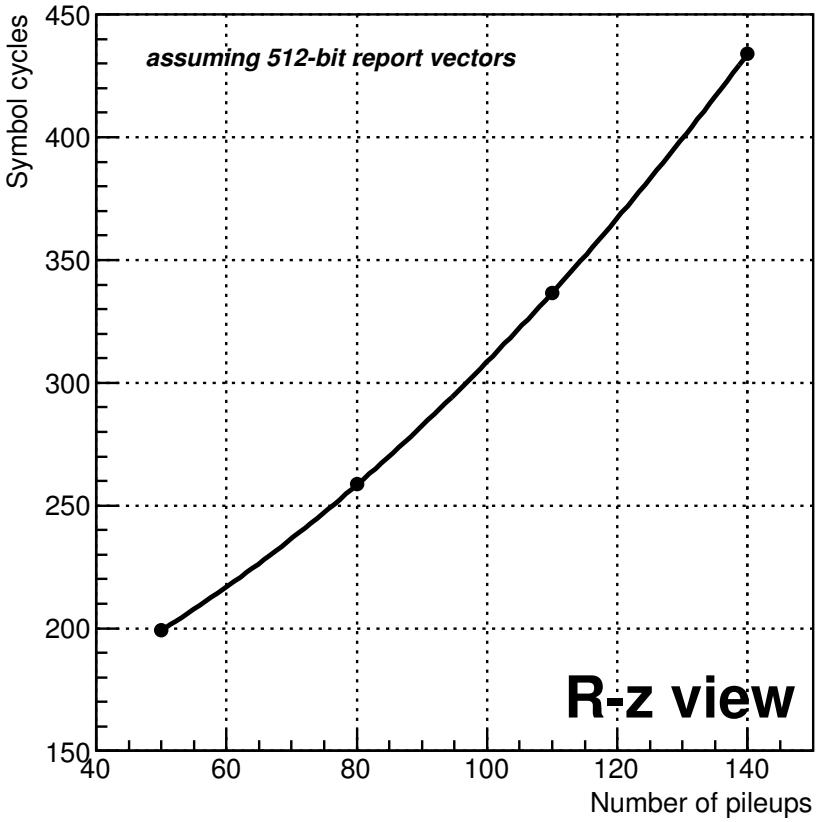


Results for Electron identification and photon rejection

Pileup	EM Clusters			Track Match		Eff.	Rejection	Purity
Inter.	Total	e	γ	e	γ	(%)	Factor	(%)
50	1242	837	405	837	9	100	45	99
80	1395	839	556	839	17	100	33	98
110	1515	844	671	844	26	100	26	97
140	1648	844	804	844	56	100	14	94

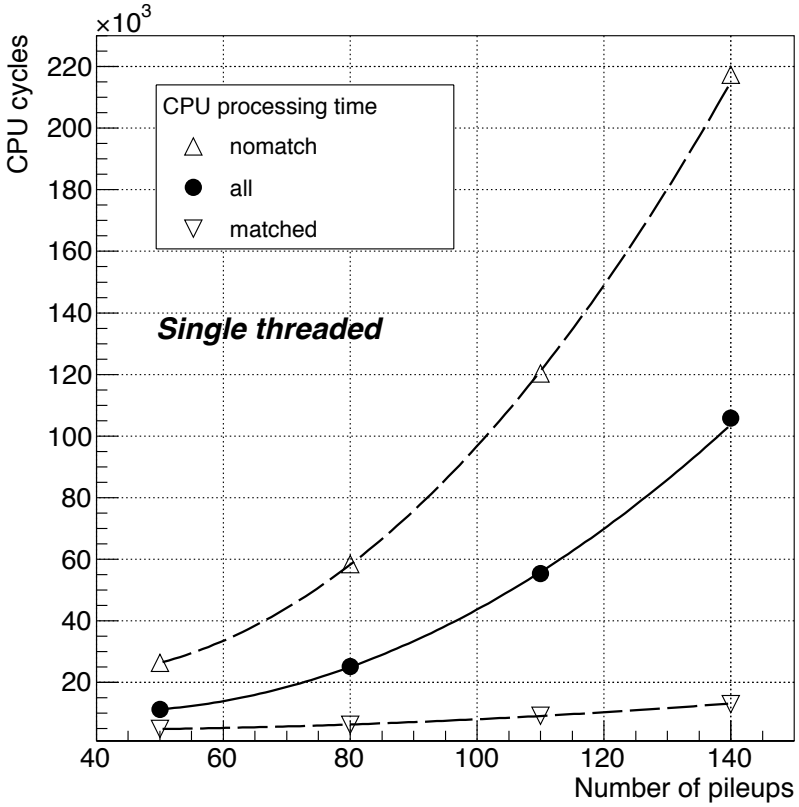
Processing time on automata processor

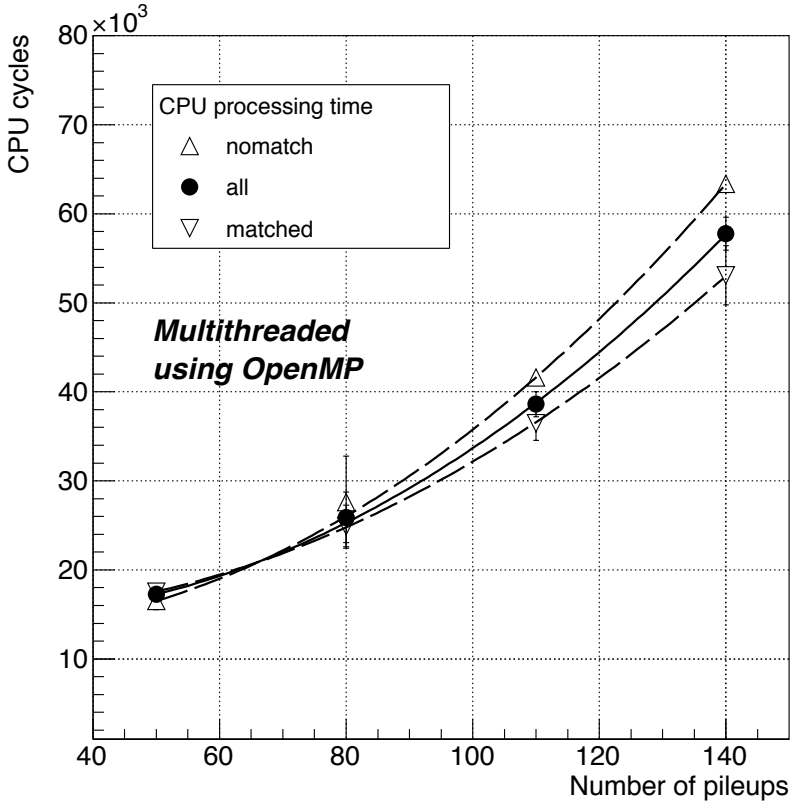




- Additional step, in external logic, needed to find coincident matches in both views.

Processing time on x86 CPU





CPU Cycles vs # of Pileups

Fermilab

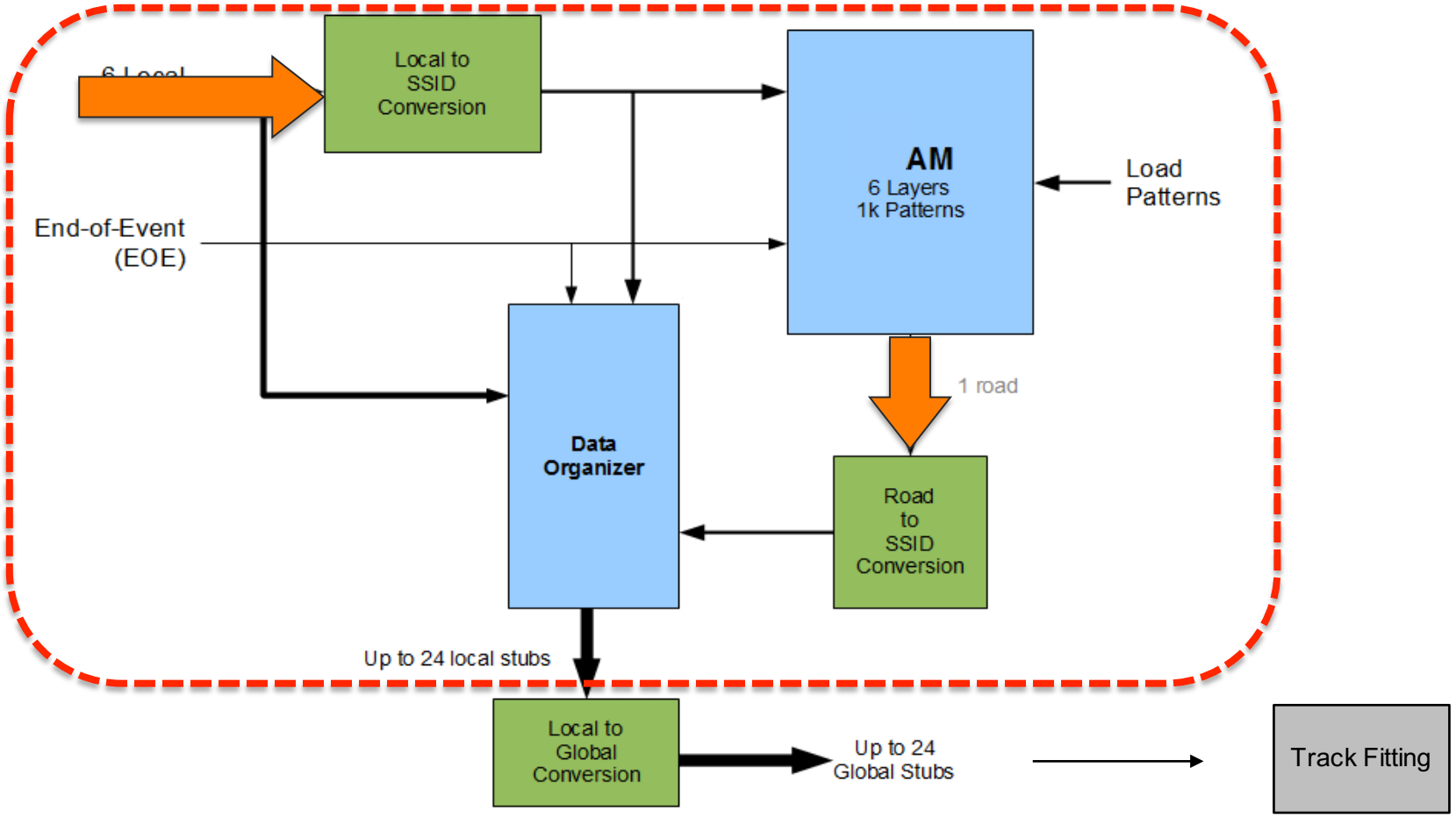
Processing Time on Automata Processor vs x86 CPU

- Used the simulated sample with 140 pileups
- ***Micron Automata Processor at 133 MHz***
  - 3.25 us + 0.37 us external processing time = 3.62 µs
- ***Intel i7, 5th generation at 3.3 GHz***
  - *Single core*: 32.1 µs
  - *OpenMP on 6 cores*: 17.5 µs

Comparison with CAM-based FPGA implementations

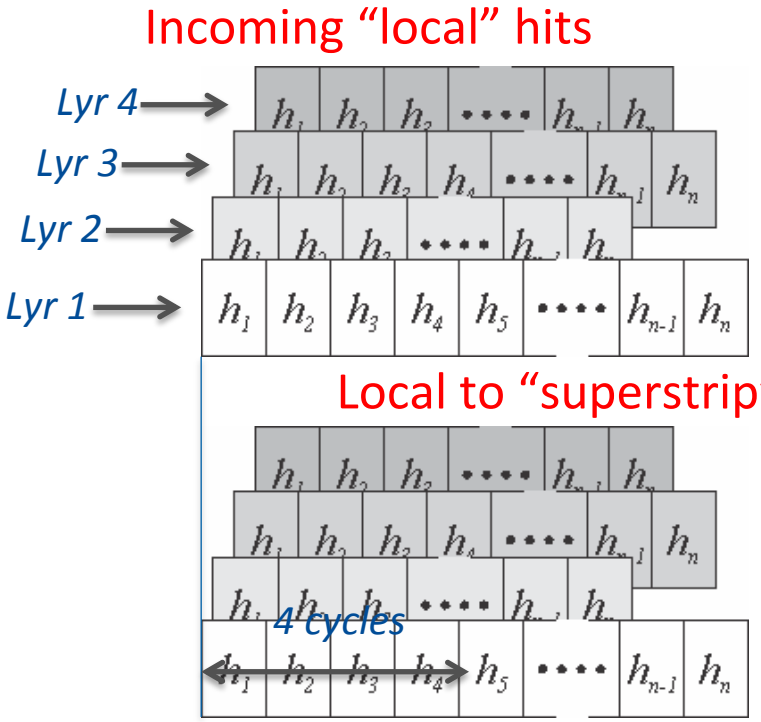
- We look at the FPGA-based PRM (Pattern Recognition Module) developed at Fermilab as a demonstrator for the VIPRAM ASIC and for optimizing its design (Ted Liu et al.)
- PRM firmware has been tested extensively and its behavior, down to clock-cycles, is deterministic and well understood.
- Possible to get very good idea of its performance relative to Micron AP without actually running it on the same data.
- With knowledge of its architecture and characteristics, we calculated the number of PRM cycles it would take to match pixel tracks to EM clusters:
  - on the same 1K event sample with 140PU used to test the Micron AP
  - using same definition of ROI associated with each EM cluster to provide same set of pixel hits as input

PRM Block Diagram



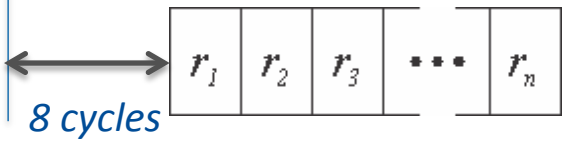
Calculate timing from the instant the first pixel hit is fed into the module up to instant the last "road id" is output from AM stage

PRM Timing Calculation



Ncycles = # of hits in layer with most # of hits
+ 4 cycles: local to ssid conversion
+ 8 cycles: road Id generation
+ # of roads found

End-of-Event



Fermilab

Processing Time on FPGA-PRM

- Average # hits in ROI in layer with largest number of hits: ~37.7
- Average # “roads” found in ROI: ~5.66
- Total number of cycles: 37.7 + 4 + 8 + 5.66 = 55.4 cycles
- For 250MHz clock: 0.2 us
- > 10x faster than automata processor

Conclusion

- Overview of Micron AP described architecture & capabilities.
- Demonstrated feasibility in HEP track recognition with a proof-of-principle application.
- Compared performance with commodity CPU and custom FPGA solution.
- Currently, AP bridges the gap between traditional CPU/GPUs and ASIC/FPGA solutions for fast pattern recognition applications.
- Areas of improvement in current AP architecture that can make it more competitive:
  - Larger symbol sizes, > 8 bits
  - Higher clock rates (> 133MHz)
  - More STEs per chip (> 48K)
  - More efficient readout architecture
- AP still in its infancy, improvements like those listed above in the next version will further enhance its suitability for HEP pattern recognition.
- Some of the results shown in this presentation are described in detail in:

  NIM A 832 (2016) 219-230.

End