PROCEEDINGS OF SCIENCE

Investigating the Micron Automata Processor for HEP Pattern Recognition Applications

Michael Wang*, **Christopher Green, and Ted Liu**
Fermilab
E-mail: mwang@fnal.gov, greenc@fnal.gov, thliu@fnal.gov

The Micron Automata Processor is a dedicated pattern matching engine that is based on a non-Von Neumann processor architecture, and was designed primarily to satisfy the growing needs of high-speed text-based pattern search applications. We investigate its suitability for HEP pattern recognition applications, using a sample track-confirmation trigger to demonstrate a proof-of-principle. We compare its performance, in this sample application, with that of other processor architectures, namely a general purpose CPU and an FPGA-based implementation using content-addressable memories.

The 25th International workshop on vertex detectors
September 26-30, 2016
La Biodola, Isola dâ?TElba, ITALY

*Speaker.

PoS(Vertex 2016)045

1. Introduction

The invention of the multiwire proportional chamber nearly half a century ago revolutionized experimental particle physics by introducing electronic tracking detectors that transformed the manual reconstruction of particle trajectories into an automated data processing task [1]. Since then, the field of electronic tracking detectors has continued to make huge strides. Today's Silicon pixel detectors offer unprecedented capabilities by providing true 3D space points and the ability to deal with high track rates and densities around the harsh interaction regions of modern collider detectors. However, the significantly higher channel densities associated with these detectors, especially in the context of the High Luminosity LHC, pose serious challenges for the task of track pattern recognition. Although the cutting edge solutions based on custom hardware being developed by the HEP community to address these challenges appear promising, it would be very valuable to explore recent developments in other sectors, such as industry, for new and innovative ideas. Since many computing challenges facing today's data-centric economies, such as those in the internet search and data-mining industries, involve pattern recognition problems fundamentally similar to those in HEP, solutions developed in those sectors may prove applicable in HEP with the obvious advantages afforded by the economies of scale. One particularly promising example is the Micron Automata Processor which we investigate in this talk for its suitability in HEP pattern recognition applications. Much of what is discussed in this talk is described in greater detail in Reference [2] and we refer the reader to that publication for more information.

2. What is the Automata Processor?

The Micron Automata Processor (AP) is the result of a clever repurposing of conventional SDRAM technology to realize the first direct hardware implementation of a non-deterministic finite automata in a commercial device [3]. The block diagram of the AP in Figure 1 bears some resemblance to a conventional memory array. The basic unit of the AP is a state transition element (STE) which is represented as a column in the block diagram. There are 48K STEs in the current version of the chip, each of which can individually be enabled or disabled. Each STE consists of 256 cells, each of which is associated with a unique 8-bit value. Referring to Figure 1, let us imagine that all the STEs, except for STE-0, are initially disabled. Imagine that we present the value 254 at the AP's 8-bit input bus. Much like conventional memory, this generates a *row address* which selects the second to the topmost cell in STE-0. If this particular STE is programmed to recognize 254, a "1" will be generated which propagates down the column, out the STE, through the routing matrix, and up to STE-3 to which STE-0 is connected. This enables STE-3 to await the next value to be presented at the 8-bit input bus to see if it matches what it is programmed to recognize. In this manner, one can imagine configuring a string of STEs where each STE is programmed to recognize a particular 8-bit value, thereby representing a particular pattern. The routing matrix structure at the bottom of the diagram is a programmable fabric that allows the user to define the interconnections between the STEs. It is responsible for generating the equivalent of the *column address* in conventional memory devices. This fabric allows the creation of automata networks where each STE's output can branch out to several others, while its own input can be enabled by multiple other STEs.

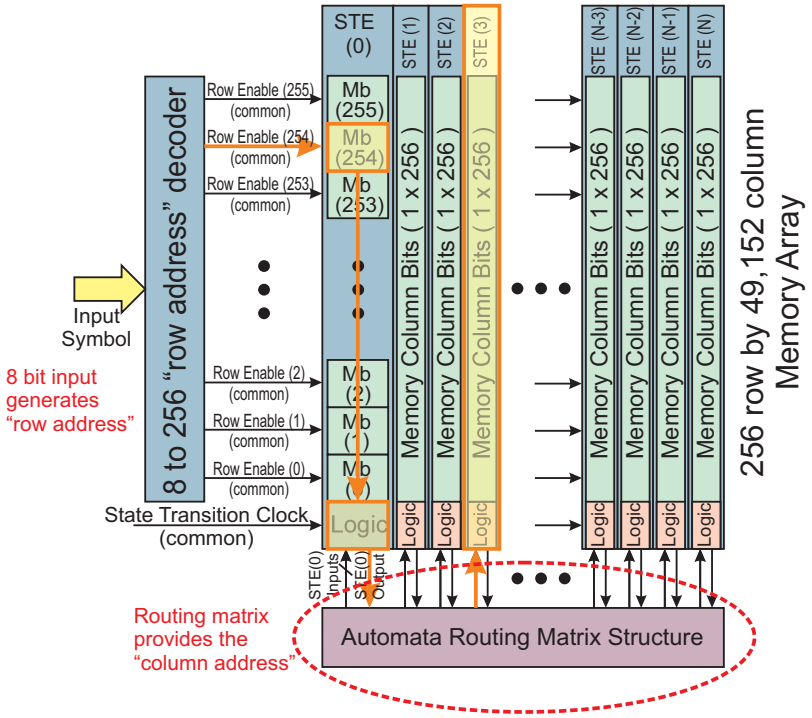

Figure 1: The block diagram of the Micron Automata Processor above shows its resemblance to the 2D memory array in traditional memory devices. One of the key things that differentiates it is the programmable routing matrix that allows user-defined interconnections between the 49,152 STEs represented by the columns.

3. From Regular Expressions to Particle Trajectories

The most obvious application of the Micron AP would be in the recognition of strings of characters or words. Figure 2 shows a network of STEs (drawn as circles) configured as a dictionary of words. The character in each circle represents the 8-bit value or symbol that particular STE is programmed to recognize. If the sequence of characters "susy" is presented to this automata network, a match would be found in exactly 4 clock or symbol cycles, irrespective of the size of the dictionary. Since the symbols are merely 8-bit quantities, they need not be restricted to ASCII characters, and could just as well represent other quantities, such as wire or pixel addresses in a HEP tracking detector. One could imagine the oversimplified case of a string of four STEs with each STE programmed to recognize a "hit" address in one layer of a 4-layer tracking detector and where all four addresses are associated with a physically possible particle trajectory. The idea, then, is to form a dictionary consisting of all such possible trajectories in the detector, where the sequence of addresses associated with each trajectory is stored in an automata network consisting of 4 STEs. There would, therefore, be as many of these automata networks as there are possible track patterns. When a sequence of hits from a real detector presented to the AP matches a sequence stored in the dictionary, it would immediately be found and recognized as a track. This is the same underlying principle used in the application of associative or content-addressable memories to HEP track pattern recognition [4].

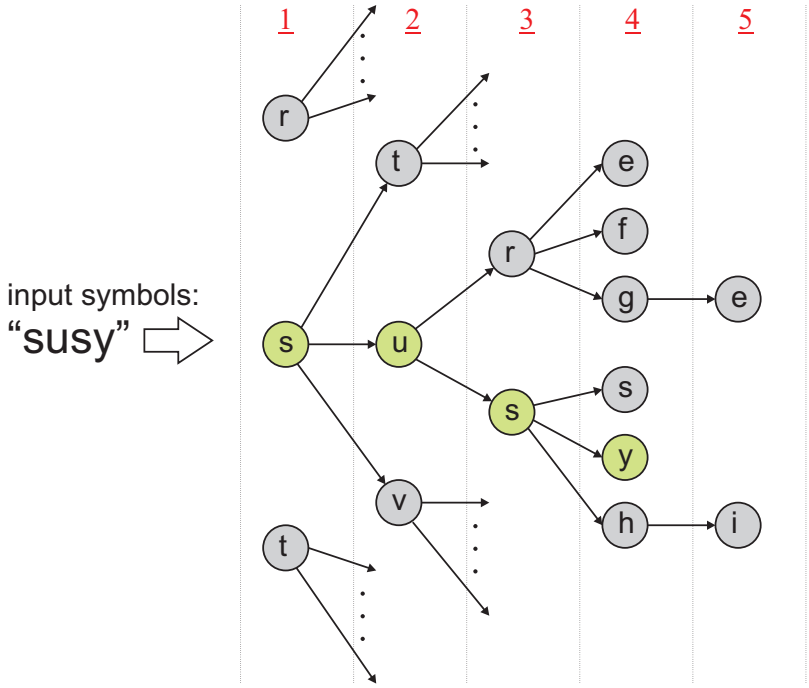

Figure 2: An automata network representing a dictionary of words and acronyms. Each STE, represented as a circle, is programmed to recognize a particular "symbol" which is an ASCII character in this case. These symbols could just as well be detector wire, strip, or pixel addresses.

4. Proof-of-Principle Application with a Toy CMS Detector

To see if we can actually use the Micron AP for HEP track pattern recognition, we developed a proof-of-principle application where we implement a hypothetical electron track confirmation trigger using a toy tracking detector that roughly approximates the actual 4-layer Phase-1 CMS pixel detector [5]. EM clusters found in the calorimeter are projected back into the interaction region to see if there are pixel detector hits consistent with those from a charged particle track pointing towards the cluster. If so, the cluster is identified as an electron instead of a photon.

4.1 Simulated Events for the Toy CMS Detector

To test the track trigger with the toy detector, we generated simulated $Z \rightarrow ee$ events and overlaid them with minimum-bias events to simulate pile-ups. 5 different samples, consisting of 1000 beam-crossings each, were created using the same set of $Z \rightarrow ee$ events overlaid with 50, 80, 110, and 140 pile-ups, respectively. Since this is just a proof-of-concept application, we kept things simple by not including stochastic processes like multiple scattering and energy loss. We also assumed 100% efficiency for the pixel detectors. However, all unstable particles were decayed randomly into the appropriate channel based on their branching ratios and with exponential decay length distributions based on their proper lifetimes. Photons were also converted into small-opening-angle pairs in the material of the detector.

4.2 Defining the Alphabet of our Dictionary: Hit Addresses

As we mentioned earlier, the basic idea behind using the AP for track recognition is to create a dictionary of all possible tracks. Before we can build this dictionary, we must first define our alphabet. Ideally, the tracks in our dictionary would be specified in full 3D coordinates. Although possible in principle, this would not be practical due to the huge number of tracks that need to be dealt with. Instead, we actually build two separate dictionaries, one for each 2D view - R-ϕ and R-z, which are operated in parallel on the same input data stream (the introduction to Section 5 below describes how we use the fact that pattern matches found in each view for a given track are correlated in time). We use alphabets with 16-bit and 14-bit symbols for the R-ϕ and R-z views, respectively. For R-ϕ, bits 14-15 determine the layer, bits 8-13, the face on that layer, bit 7, which of two Read-Out-Chip (ROC) rows in that face, and bits 0-6, the pixel row in the ROC. For R-z, bits 12-13 determine the layer, bits 9-11, the chip module in that layer, bits 6-8, which of 8 ROC columns in that module, and bits 0-5, the pixel column in the ROC. We also divide each view into sectors, in order to organize the patterns (words) in our dictionary into separate pattern banks.

4.3 Basic Automaton for the Track Finder

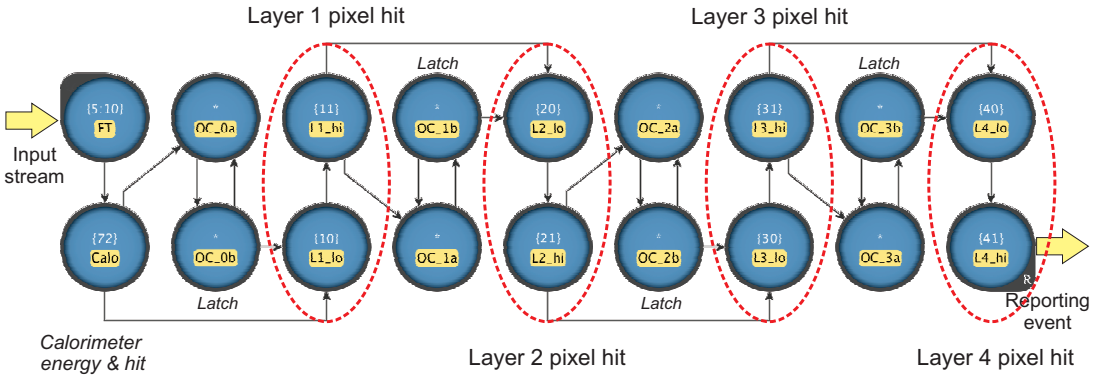

Figure 3: The basic automaton we use to represent one track pattern in our dictionary. Pairs of STEs are used to represent up to 16-bit addresses in order to circumvent the limitation imposed by the AP's native symbol size.

Having defined the dictionary and its alphabet, the next step is to define the automaton or automata network that acts as the place-holder for a single pattern or word in the dictionary. Each pattern in our dictionary consists of 4 16-bit (14-bit) pixel-hit addresses in the R-ϕ (R-z) view, one for each detector layer. The automaton we use for this purpose is shown in Figure 3. This design is significantly more complicated than the string of 4 STEs we described in our oversimplified example earlier, where we assumed 8-bit hit addresses were sufficient. The reason is that the current version of the AP has a native symbol size of only 8 bits and we require 16 bits (14 bits) for each of the 4 hits in one track pattern of our R-ϕ (R-z) dictionary. This requires using 4 STE pairs to represent the 4 hit addresses. This further necessitates an additional 4 STE pairs that ensure that the lower half of the STE pair representing the hit is enabled on the appropriate symbol cycle associated with the lower 8 bits of the hit address. The additional STEs are for the ϕ position of the "5th" hit associated with the calorimeter cluster and the energy range that cluster must fall within.

Pileup Interactions	EM Clusters			Track Match		Efficiency (%)	Rejection Factor	Purity (%)
	Total	Electrons	Photons	Electrons	Photons			
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

Table 1: This table summarizes the Automata Processor's ability to identify electrons and reject photons for the simulated samples used in our study. See text for more details.

5. Testing the Track Confirmation Trigger on the Toy Detector

Using this automata network design to represent the track patterns, we build our dictionary consisting of 72 banks of 1,163 patterns each for the *R*-ϕ view and 244 banks of $\sim$4662 patterns each for the *R*-*z* view. We use the simulated $Z \rightarrow ee$ samples described earlier to test our AP-based trigger. For the purposes of our proof-of-concept demonstration, we assume that we have perfect knowledge of the location of the primary interaction vertex. In an actual experiment like CMS, however, such knowledge would have to be provided by another sub-detector like the outer tracker. For each beam crossing, we begin by selecting all the EM clusters above $p_T = 5$ GeV. We then project a straight line from the each of the selected clusters back to the primary interaction vertex. Only those hits within a narrow Region-Of-Interest (ROI) around this line are considered to significantly reduce the total amount of data that needs to read out of the detector. All the hits in the ROIs are then fed into the AP to see if any sequence matches a pattern stored in the banks. A track is considered found, and hence a trigger accept signal "generated", if matches are found in both views that occur on the same clock or symbol cycle (in reality, there is a fixed offset in the cycle where matches are found between the two views due to the fact that the *R*-*z* view automatons representing a pattern have fewer STEs and hence require fewer clock cycles to find matches).

5.1 Results for Electron Identification and Photon Rejection

The results of our tests are shown in Table 1 which summarizes the AP-based trigger's ability to identify electrons and reject photons. Results are shown for each of the 4 samples described earlier. Columns 2-4 show the total number of EM clusters in each sample broken down into those that originated from electrons or photons. Columns 5 and 6 show the number of these clusters in each type for which the AP reported a track match. The remaining columns show that for this proof-of-concept application, we are able to identify electrons with 100% efficiency and still maintain a high purity all the way up to the sample with 140 pileups.

5.2 Processing Time Relative to Commodity CPU

For the 1K-crossing sample with 140 pileups, described in Section 4.1, the number of clock cycles it takes the AP to find track match and generate a trigger accept is 3.62 μs. For comparison, we created a functionally equivalent c-based algorithm and, using the same 140 pileup sample, executed it on a 3.3 GHz, 5th generation Intel core i7 processor. For a single core, it takes 32.1 μs

to execute the equivalent algorithm and generate the trigger accept. Using OpenMP with 6 cores, it takes 17.5 μs to do the same. For this application, the AP clearly has an advantage over the Intel CPU. Compared with this CPU, its execution time also scales much better as a function of the number of pileups, exhibiting almost linear response for the R-ϕ view [2].

5.3 Processing Time Relative to Associative-Memory Based FPGA Implementation

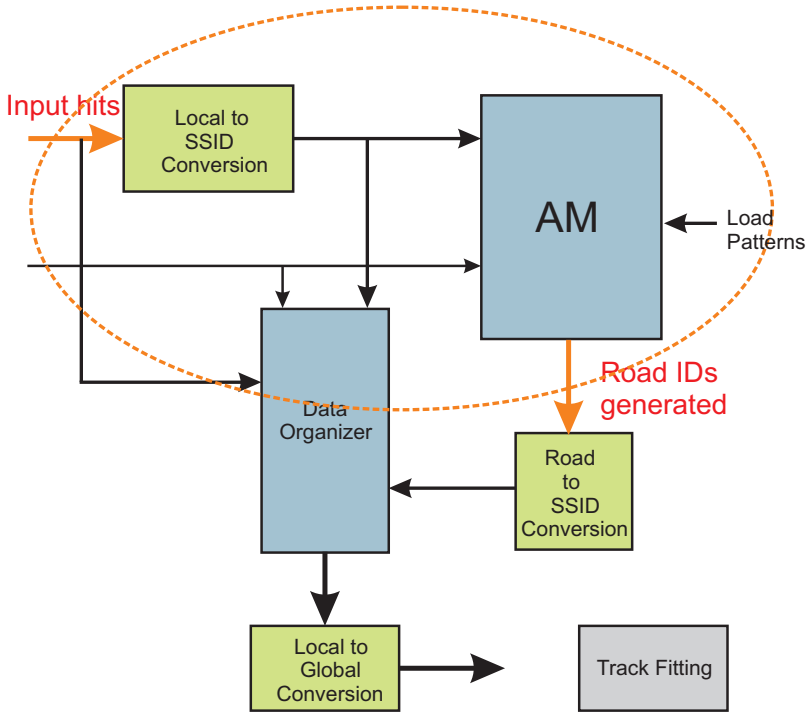

Figure 4: Block diagram of the FPGA-based Pattern Recognition Module used to compare with the AP. For our studies, we limit our attention to the portion enclosed in the oval.

Having seen how the AP compares with a traditional CPU at the commodity end of the spectrum, let us now see how it compares to a solution at the other end of the spectrum which includes ASICs and FPGAs. Specifically, we look at the FPGA-based Pattern Recognition Module (PRM) [6] developed at Fermilab as both a demonstrator for the custom VIPRAM ASIC [7] and as a tool used for optimizing its design. The PRM firmware has been tested extensively and its behavior, down to the clock cycles, is deterministic and well understood. With this knowledge, we can calculate the number of clock cycles it takes the PRM to find track matches on our samples without having to run our samples through the hardware.

As in the AP-based implementation, we assume two separate dictionaries or pattern banks for the FPGAs, one for each 2D view. Referring to the block diagram of the PRM shown in Figure 4, we ignore the track fitting stage and focus our attention only on the topmost 2 stages enclosed in the dotted oval. We start counting clock cycles the instant the first pixel hit is fed into the first stage of the PRM and stop counting as soon as the last road ID is generated from the Associative Memory (AM) stage. We calculate the total number of clock cycles as the sum of the number of hits in the detector layer having the greatest number of hits within the ROI, the execution latencies

in each of the two PRM stages considered (4 cycles for "local to ssid conversion" and 8 cycles for "road ID generation"), and the number of roads found. For the 140-pileup sample described earlier, the average number of pixel hits in the layer with the greatest number of hits within the ROI is 37.7 hits. Since the average number of roads found within an ROI is higher in the R-z view than in the R-ϕ view, we use the former, which is 5.66, to calculate our cycle counts. The total number of cycles is therefore $37.7+4+8+5.66=55.4$ cycles. For a Xilinx Kintex Ultrascale FPGA running at 250 MHz, this translates to 0.2 μs, which is over an order of magnitude faster than the AP.

6. Conclusions and Recommendations

We have provided an overview of the Micron AP, describing what it is and what it is capable of doing. We described a proof-of-concept application demonstrating its feasibility, in principle, for track pattern recognition applications in HEP. Finally, we compared its processing performance with both a commodity processor and an FPGA-based implementation. The AP bridges the gap between these two extremes in the processing spectrum. It appears to be suitable for offline pattern recognition applications and, perhaps, even moderately demanding real-time applications. We showed that it was superior to the CPU for low-latency real-time applications. On the other hand, relative to the FPGA, it was roughly an order of magnitude slower. As far as pattern densities are concerned, 2496 instances of the automaton representing one track pattern in our sample implementation can be programmed into the current version of the AP chip. The FPGA implementation described above can store roughly twice as many patterns. Associative memory based ASIC implementations currently under development can easily store over 100K patterns. In its current form, the AP is not a replacement for ASIC/FPGA solutions in the most demanding real-time applications requiring the best possible performance. However, it should also be noted that we are dealing here with the very first version of an entirely new and unconventional product category with unique and promising capabilities, and having substantial headroom for further improvements. For instance, one of the limiting aspects of the current design for HEP applications is the native 8-bit symbol size. Increasing this could significantly simplify the design of the automaton used to represent a single track pattern by drastically reducing STE resource usage. This will, in turn, cut down the number of symbol matching cycles and increase the density of patterns that can be stored per chip. Another limiting aspect is the latency associated with the current readout architecture which is responsible for a significant fraction of the total processing time. Substantial gains can be achieved by optimizing the readout of the match results. Furthermore, all these improvements do not take into account anticipated reductions of the semiconductor process node and increases in symbol processing clock frequencies in future generations of the AP chip. We look forward to such possible enhancements in this new technology and we anticipate that, as awareness of it grows within the HEP community, other novel and interesting applications will begin to sprout.

References

[1] G. Charpak, R. Bouclier, T. Bressani, J. Favier, Č. Zupančič, The use of multiwire proportional counters to select and localize charged particles, Nucl. Instrum. Methods Phys. Res. 62 (1968) 262.

[2] M.H.L.S. Wang, G. Cancelo, C. Green, D. Guo, K. Wang, Using the automata processor for fast pattern recognition in high energy physics experiments-a proof of concept, Nucl. Instrum. Methods Phys. A Res. 832 (2016) 219.

[3] P. Dlugosch, D. Brown, P. Glendenning, M. Leventhal, H. Noyes, An efficient and scalable semiconductor architecture for parallel automata processing, IEEE Trans. Parallel Distrib. Syst. 25 (2014) 3088.

[4] M. Dell'Orso, L. Ristori, Vlsi structures for track finding, Nucl. Instrum. Methods Phys. Res. A Accel. Spectrom, Detect, Assoc. Equip. 278 (1989) 436.

[5] A. Dominguez, et al., CMS Technical Design Report for the Pixel Detector Upgrade, CMS-TDR-011, 2012.

[6] J. Olsen, J. Hoff, T. Liu, J. Wu, Z. Xu, A new way to implement high performance pattern recognition associative memory in modern fpgas, poster presented at Topical Workshop on Electronics for Particle Physics (TWEPP), September 28–October 2, Instituto Superior Técnico, Lisbon, Portugal, 2015.

[7] T. Liu, J. Hoff, G. Deptuch, R. Yarema, A new concept of vertically integrated pattern recognition associative memory, in Proceedings of the 2nd International Conference on Technology and Instrumentation in Particle Physics (TIPP 2012), Phys. Procedia 37 (2012) 1973.

.