Micron’s Automata Processor

Beyond CMOS HPC Workshop

Terry Leslie
Director, Business Development
Distinguished Member of the Technical Staff
Advanced Computing Group
Micron Technology, Inc.
Agenda

- Introduction
- Architecture
- Execution Model
- Application Survey
Automata Processor

Micron’s **Automata Processor** is a revolutionary new class of programmable accelerator

- A hardware implementation of highly-parallel Non-deterministic Finite Automata (NFA)
- Orders of magnitude (>100x) faster than CPU’s for pattern matching and graph analytics
- Rapidly reconfigurable for complex algorithms
- Simple parallel programming with familiar tools

Automata is a Multiple Instruction – Single Data (MISD) processor

- Non-von Neumann architecture evaluates streaming data against all instructions in parallel
- Enables deep analysis of data streams containing spatial and temporal information
- Complexity of expressions (instructions) has no impact on execution time
NFAs

Any nondeterministic machine can be modeled as deterministic at the expense of exponential growth in the state count.

- Today’s computers model NFA as a DFA, requiring all state transitions to be explicitly enumerated. This creates an explosion in memory space.

**Automata Processor** - Nondeterministic Finite Automaton (NFA)

**Conventional CPU** – Deterministic Finite Automaton (DFA)
Agenda

- Introduction
- Architecture
- Execution Model
- Application Survey
Automata Processor – Basic Operation

Row Access results in **one** word being retrieved from memory.

Row Access results in **49,152** match & route operations (then Boolean AND with “active” bit-vector)
Automata Processor: The Fabric

Match Elements:

- State Transition Element (STE)
- Determine match of input symbol
- Can support high in/out degree

Boolean Logic Elements

- Programmable Functions

Counters

- 12 bit counters

Micron Automata Processor: Silicon

Key Device Parameters

- 129.3 mm² (12.15 × 10.64)
- 133M Symbols/Second
- 49,152 State Transition Elements
- 24,576 STE Max Automata Size
- 5-6W TPD
- 512 Entry State Cache
- 6,144 STE Max Match Capacity
Automata Processor PCIe Boards

AP PCIe gen 3 Board

32 AP devices with >1.5M states

100 & 133 MS/sec
ES boards – 2Q16
Production boards – 4Q16

(133 MS/s boards available 3Q and 1Q17)
Micron’s Automata Processor (AP) is a reconfigurable processing architecture that enables programmers to easily exploit massive parallelism. The AP is purpose-built to address the processing challenges associated with graph analysis, pattern matching, and data analytics.

Many of today’s most difficult computing problems require parallel code to search and analyze unstructured data, which may be both structured and unstructured. This class of computation is not handled well by traditional CPU and memory system architectures. It requires a fundamentally new approach to computing.

The Automata Processor (AP) is a completely new architecture for regular expression acceleration, including analysis, statistics, and regex operations. It solves the problem of thousands of unstructured data elements in the shortest possible time. It offers efficiency far greater than traditional CPUs and GPUs with a much easier programming model for parallel processing.

The AP adds new thrust to this class of computing. It’s a disruptive acceleration technology that can dramatically improve throughput in many Big Data and cloud applications. An ecosystem is already forming around this technology. The SDK allows developers to be created, perfected, and distributed, enabling collaborators to use in increasing scales of parallelism.

Request a copy of the evaluation SDK here

Review Research Results here
Agenda

- Introduction
- Architecture
- Execution Model
- Application Survey
Automata Representation

Edges become Symbols in State Transition Elements (STEs) – one STE per edge

Outgoing edges from a start state become start STEs

Incoming edges to an accept state become reporting STE
AP Layers of Parallelism

Each STE: test many different symbol matches per cycle, per input symbol
- Von Neumann (VN) architecture needs multiple instructions

Multiple active STEs: pursue different matching hypotheses in parallel
- Non-determinism very difficult in VN; exp. growth in space complexity or looping

Multiple activations: branching—activate many potential successor paths
- Non-determinism very difficult in VN; exp. growth in space complexity

Multiple automata: independent rules
- VN requires multiple threads, limited capacity

Multiple streams
- VN requires multiple threads
Design Phase and Runtime Phase

Two distinct phases of AP usage: **design phase** and **runtime phase**

**Design phase:**
create, simulate, debug, and compile automata designs

**Runtime phase:**
program the AP; scan input data; retrieve results
Programming Overview

DESIGN PHASE

1. Create automata designs
2. Compile designs
3. Program the AP
4. Write data to the AP
5. Process results

HOST

Patterns

AP APIs / Device Driver

AP Board

111010
101001

Data
RUNTIME PHASE

1. Create automata designs
2. Compile designs
3. Program the AP
4. Write data to the AP
5. Process results

HOST

AP APIs / Device Driver

Patterns

Data

111010
101001

AP Board

100100001100101101101100011011000101001010011101010110010001001001001000100010001010010100100100100100010010010010001001001001000100100100100010010010010001001001001000100100100100010010010010001001001001000100100100100010010010010001001001001000100100100100010010010010001001001001000100100100100010010010010001001001001000100100100100010010010010001001001001000100100100100010010010010001001001001000100100100100010010010010001001001001000100100100100010010010010001001001001000100100100100010010010010001001001001000100100100100010010010010001001001001000100100100100010010010010001001001001000100100100100010010010010001001001001000100100100100010010010010001001001001000100100100100010010010010001001001001000100100100100010010010010001001001001000100100100100010010010010001001001001000100100100100010010010010001001001001000100100100100010010010010001001001001000100100100100010010010010001001001001000100100100100010010010010001001001001000100100100100010010010010001001001001000100100100100010010010010001
Programming Overview

RUNTIME PHASE

Create automata designs

Compile designs

Program the AP

Write data to the AP

Process results

HOST

AP APIs / Device Driver

Data

Patterns

Host Program

AP Board
Overview of APIs

AP_AddAnmlElement()
AP_AddAnmlEdge()
...

AP_CompileAnml()

AP_Load()

AP_OpenFlow()

AP_GetMatches()

AP_ScanFlows()
Automata Design Methods

Automata designs can be created by:

- Directly converting from **regular expressions**
- Visually diagramming the patterns in the **Workbench** tool
- Using the **programming APIs** in either C, Python or Java
Agenda

- Introduction
- Architecture
- Execution Model
- Application Survey
Problems Aligned with the Automata Processor

Applications requiring **deep analysis** of **data streams** containing **spatial** and **temporal** information are often impacted by the **memory wall** and will benefit from the **processing efficiency** and **parallelism** of the Automata Processor.

**Network Security:**
- Millions of patterns
- Real-time results
- Unstructured data

**Bioinformatics:**
- Large operands
- Complex patterns
- Unstructured data

**Financial Services:**
- Highly parallel operation
- Real-time results
- Unstructured data

**Machine Learning:**
- Highly parallel operation
- Real-time results
- Unstructured data
AP Scope of Use

Approximate String Matching
- Regular Expressions
- Entity Resolution
- Edit distance
- Hamming Distance

Graph Analytics
- Hamiltonian paths/cycles
- Breadth First Search
- Clique Discovery
- Subgraph Mining

Machine Learning
- Random Forests
- Association Rule Mining
- Hierarchical Temporal Memory

Floating point Operations
- Interval Stabbing

Protein Motifs

Sequence Mapping

Boolean SAT

Network Security

MOTOMATA

Twitter Sentiment Analysis

Hand-written digit recognition
Approximate String Matching

Hamming distance
- SDK Sample code

Entity resolution
- Initial research at the UVA CAP (Center for Automata Processing)
  - Historical Social Networks and Archival Context database entity analysis

*Entity Resolution using the Micron Automata Processor*: University of Virginia - 5th International Workshop on Architectures and Systems for Big Data (ASBD)
Approximate String Matching

Approximate string matching API

- Micron Technology SDK API
- Search for string patterns (text or non-text) within defined error tolerances
  - Edit distance - number of symbol mismatches/substitutions, insertions & deletions
  - Variable error windows can be defined

Nondeterministic Finite Automata in Hardware – the Case of the Levenshtein Automaton – University of Virginia - 5th International Workshop on Architectures and Systems for Big Data (ASBD) 2015
Example: **Protein Automata** - Accelerating search for PROSITE protein motifs

- **Input Data:** *Proteomes of interest*
- **Automata:** 1308 ProSite protein signatures

Massive parallel pattern search of fuzzy signatures

- All 1308 ProSite automata fit into a single Automata Processor
- Every pattern evaluated in parallel
- Set of inexact NFAs – some with > 1 M possible matches
Prosite pattern motifs

\[ W-x(0,2)-[KDN]-\{Q\}-\{L\}-K-[KRE]-[LI]-E-[RKN]. \]

\[ C-x-C-x(3,5)-C-x(7)-G-x-C-x(9)-C-C. \]

\[ G-C-x(1,3)-C-P-x(8,10)-C-C-x(2)-[PDEN]. \]

\[ C-x(5,6)-[DENQKRHSTA]-C-[PASTDH]-[PASTDK]\ldots \]

Regular expression

\[ W.\{0,2\}[KDN][^Q][^L]K[KRE][LI]E[RKN] \]

High Performance Pattern Matching using the Micron Automata Processor: Georgia Institute of Technology & University of Missouri - Accepted for publication at the IEEE 30th International Parallel and Distributed Processing Symposium, May 2016.
Ps_scan vs. Protomata

Protomata Network Design

Micron Technology benchmarking on AP alpha PCIe board.
Implement the Snort rule set on the AP for Network Intrusion Detection and deep pattern inspection

- Snort ruleset written in a description language with 5310 active rules used to scan for network intrusions
- Rules contain location modifiers, distance modifiers and other modifiers

Method:
- Derive NFA automaton from SNORT ruleset
- 4312 (81%) of the active pattern matching rules can be efficiently implemented
- Snort rule NFAs fit in about ½ board which enables the ruleset to be replicated in another logical core - allows multi-thread processing of network packets at higher bandwidth

High Performance Pattern Matching using the Micron Automata Processor: Georgia Institute of Technology & University of Missouri - Accepted for publication at the IEEE 30th International Parallel and Distributed Processing Symposium, May 2016.
Genetics Motif Search

Planted Motif Search

- 20 base-pair strings (L=600) of genomic data
- Fuzzy string matching to find correlating sub-strings in the data with error distance=d
- Identify the motif

Method:

- Stage 1 – Identify all n-cliques
- Stage 2 - Build a search tree using one sequence & check if root to leaf path represents a motif using other sequences

Finding Motifs in Biological Sequences using the Micron Automata Processor: Georgia Institute of Technology – Presented at the 28th IEEE International Parallel & Distributed Processing Symposium (IPDPS 2014).
Automata Processor: Bioinformatics

MOTOMATA: de-novo (l,d) motif search

```
acgttagaactgcgatctcgatagctcgcttagctagcg
tcgatatgcgttggggatataacgatatgcattagct
cctggctaatgagtadtatgcataacgatagtacctaga
taccgatattaggtatatggagaaatactcgctagatac
gtactgatcgactcgaaatcagtcahgtattcagctagat
gtactgatcgacttcggagcttggtataatggtaatg
```

Planted Motif Search Problem

<table>
<thead>
<tr>
<th>Processors</th>
<th>Automata Processor</th>
<th>UCONN - BECAT Hornet Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processors</td>
<td>48 (PCIe Board)+CPU</td>
<td>48 CPU (Cluster/OpenMPI)</td>
</tr>
<tr>
<td>Power</td>
<td>245W-315W$^1$</td>
<td>&gt;2,000W$^1$</td>
</tr>
<tr>
<td>Cost</td>
<td>TBD</td>
<td>~$20,000$</td>
</tr>
<tr>
<td>Performance (25,10)</td>
<td>12.26 minutes$^2$</td>
<td>20.5 minutes</td>
</tr>
<tr>
<td>Performance (26,11)</td>
<td>13.96 minutes$^2$</td>
<td>46.9 hours</td>
</tr>
<tr>
<td>Performance (36,16)</td>
<td>36.22 minutes$^2$</td>
<td>Unsolved</td>
</tr>
</tbody>
</table>

$^1$ Micron Technology Estimates, Not including Memory of 4GB DRAM/Core
$^2$ Research conducted by Georgia Tech (Roy/Aluru)
Machine Learning

Implementing Supervised Machine Learning using Random Forest Models

- **Nodes**: features splits or leaf classification nodes
  - **Leaf** nodes return a classification & Internal **split nodes** split on a feature threshold value
- **Depth**: feature splits used by the decision tree to classify the result
- **Ensemble method**: combines the predictions of several decision trees

Method:
- Use **Scikit-learn** framework to train a Random Forest
- Convert trees into chains
- Convert numerical input features into symbol ranges
- Create compact model

Towards Machine Learning on the Automata Processor – University of Virginia & Micron Technology - Accepted for presentation at the International Supercomputing Conference (ISC) in June 2016
Machine Learning w/ Random Forests

Twitter Sentiment Analysis

@Micron is awesome. #micron
See the new SSD? #micron
My computer is blazing fast! #micron
I don’t like Micron’s new memory. #micron

Results
(Summer Internship Project)

<table>
<thead>
<tr>
<th>State-of-art</th>
<th>Automata Processor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processors</td>
<td>96 node cluster</td>
</tr>
<tr>
<td>Accuracy</td>
<td>72%</td>
</tr>
<tr>
<td>Performance</td>
<td>120 kTweets/sec</td>
</tr>
</tbody>
</table>

Hand written Numeral Analysis

<table>
<thead>
<tr>
<th>State-of-art (neural network)</th>
<th>Automata Processor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processors</td>
<td>GPU</td>
</tr>
<tr>
<td>Accuracy</td>
<td>99.7%</td>
</tr>
<tr>
<td>Performance</td>
<td>Learning (14 hours)</td>
</tr>
<tr>
<td></td>
<td>Processing (61kPred/secs)</td>
</tr>
</tbody>
</table>
Implementing Association rule mining (ARM) or frequent itemset mining (FIM)

- Identify strong rules discovered in databases
- The order of items within a transaction doesn’t matter
  - Web usage mining, Market basket analysis, Traffic accident analysis, Bioinformatics, Intrusion detection

<table>
<thead>
<tr>
<th>Transaction</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bread, Milk</td>
</tr>
<tr>
<td>2</td>
<td>Bread, Diaper, Beer, Eggs</td>
</tr>
<tr>
<td>3</td>
<td>Milk, Diaper, Beer, Coke</td>
</tr>
<tr>
<td>4</td>
<td>Bread,Milk, Diaper, Beer, Coke</td>
</tr>
<tr>
<td>5</td>
<td>Bread, Milk, Diaper, Coke</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bread</td>
<td>0</td>
</tr>
<tr>
<td>Milk</td>
<td>1</td>
</tr>
<tr>
<td>Diaper</td>
<td>2</td>
</tr>
<tr>
<td>Beer</td>
<td>3</td>
</tr>
<tr>
<td>Coke</td>
<td>4</td>
</tr>
<tr>
<td>Eggs</td>
<td>5</td>
</tr>
<tr>
<td>Separator</td>
<td>255</td>
</tr>
</tbody>
</table>

Method:
- Apriori framework: Downward-closure property - To be a frequent itemset, subsets must also frequent itemset
- Candidates of frequent (K+1)-itemsets are generated from K-itemsets
- Multi-pass - AP is used to accelerate each level
- More obscure associations drive increasing combinatorial search space

Association Rule Mining with the Micron Automata Processor – University of Virginia - 29th IEEE International Parallel & Distributed Processing Symposium (IPDPS 2015).
Interval Stabbing

Implementing parallel interval stabbing numerical analysis using the AP

- Often O(n) complexity
- AP can do comparison to all intervals in parallel in many formats

Method:

- Macro enables variable length literal interval to be evaluated
  - Multiple formats, Left inclusive/Right inclusive, Member/Non-member, Little/Big Endian, Signed/Unsigned, Floating point numbers

High Energy Physics

Fermi National Accelerator Lab wants to identify interesting high energy particles fast

- High energy particle paths are patterns
- Particle detectors are comprised of a geometrical array of pixels – particle paths are approximate patterns

Method:

- Create “interesting” particle automaton based upon physics & detector geometry
  - Simulate high momentum tracks (low curvature in 4T magnetic field)
  - Assign an address to each hit in each projection to build patterns of desired hit combinations.

Fast Track Pattern Recognition in High Energy Physics Experiments with the Automata Processor:
Automata Processor Research Activity

**Center for Automata Processing (CAP)**
- Created by the University of Virginia & Micron
- Create an eco-system of university research focused around a large scale AP cluster
- Directed by Dr. Kevin Skadron – Chair of the CS Department
- CAP web site: [www.cap.virginia.edu/research](http://www.cap.virginia.edu/research)

**Georgia Institute of Technology**
- Dr. Srinivas Aluru - Professor in the School of Computational Science and Engineering
- Bioinformatics & Graph Analytics
- Dr. Aluru’s web site: [www.cc.gatech.edu/~saluru/](http://www.cc.gatech.edu/~saluru/)

**University of Missouri**
- Dr. Michela Becchi – Professor in the Electrical and Computer Engineering Department
- Network Security
- Dr. Becchi’s web site: [web.missouri.edu/~becchim/](http://web.missouri.edu/~becchim/)
Automata Summary

Micron is delivering a massively parallel non-von Neumann MISD compute architecture

- A hardware implementation of highly-parallel Non-deterministic Finite Automata (NFA)
- Initial results indicate orders of magnitude faster for NFA pattern matching
- Rapidly reconfigurable for complex algorithms
- Simple parallel programming and reconfiguration with familiar tools

**Higher Performance:**
- >100x performance increase for complex NFAs

**Lower Power:**
- As little as 0.9 pJ/DecisionOp
- 5.8W TDP per device

**Lower Cost:**
- One PCIe card can outperform a cluster of processors

**Better Quality of Result:**
- Directly analyzes complex graphs without approximations

**Ease of Parallel Programming:**
- No special programming considerations required to perform parallel processing
- No vectorization of data; no timing loops; no race conditions
Automata Processor Contacts

**Micron Technology**
- Terry Leslie – tleslie@micron.com
- Micron Automata Processor web page: www.micronautomata.com

**UVA Center for Automata Processing (CAP)**
- Dr. Kevin Skadron - skadron@virginia.edu
- CAP web page: www.cap.Virginia.edu

**Georgia Institute of Technology**
- Dr. Srinivas Aluru - aluru@cc.gatech.edu
- Dr. Aluru’s web site: www.cc.gatech.edu/~saluru/

**University of Missouri**
- Dr. Michela Becchi - becchim@missouri.edu
- Dr. Becchi’s web site: web.missouri.edu/~becchim/