CRISP
Center for Research on Intelligent
Storage and Processing in Memory

Accelerating Complex Pattern Recognition Processing with In-Memory Accelerator Architectures

Elaheh Sadredini

University of Virginia

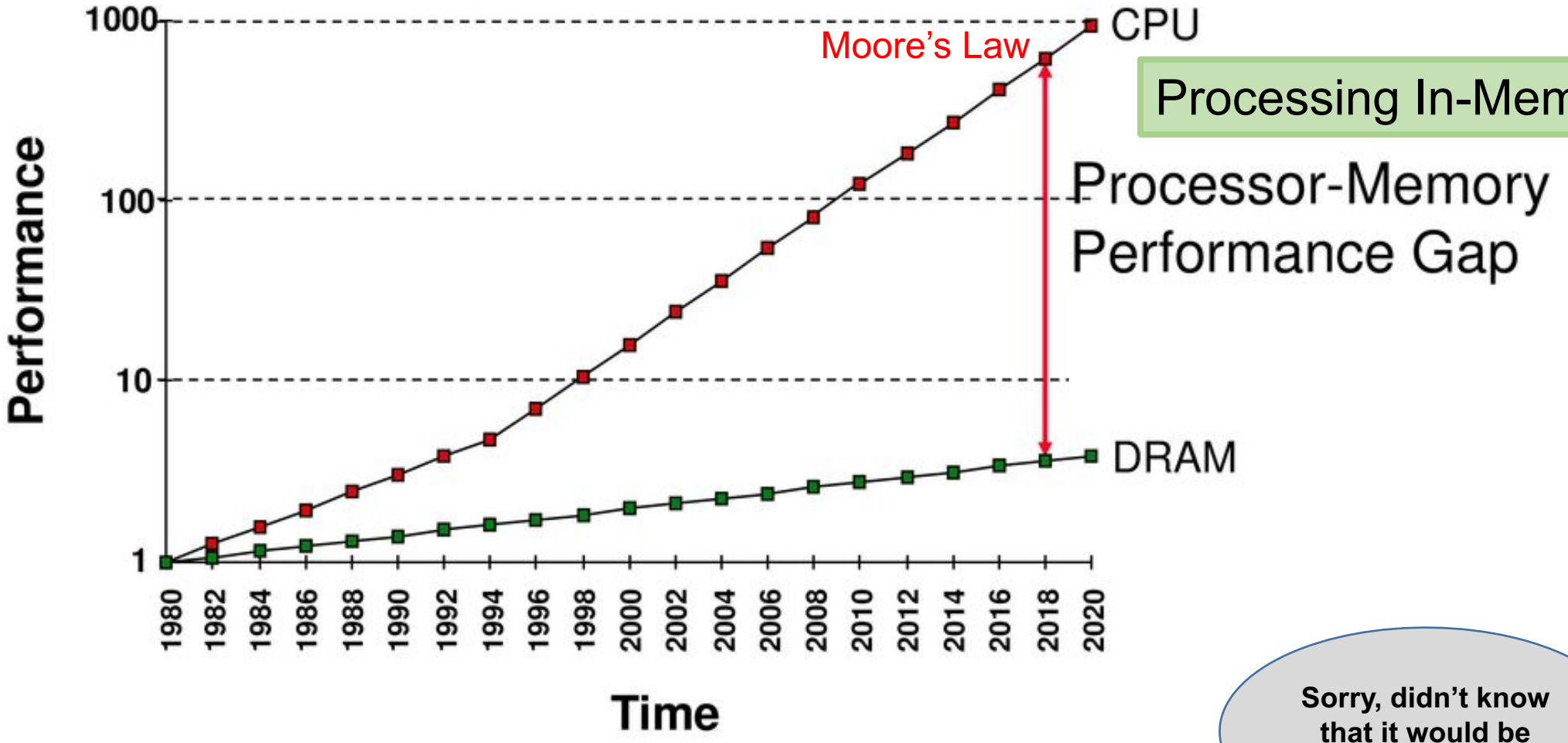
Department of Computer Science

Ph.D. Dissertation Defense

Advisor: Professor Kevin Skadron

April17th 2019

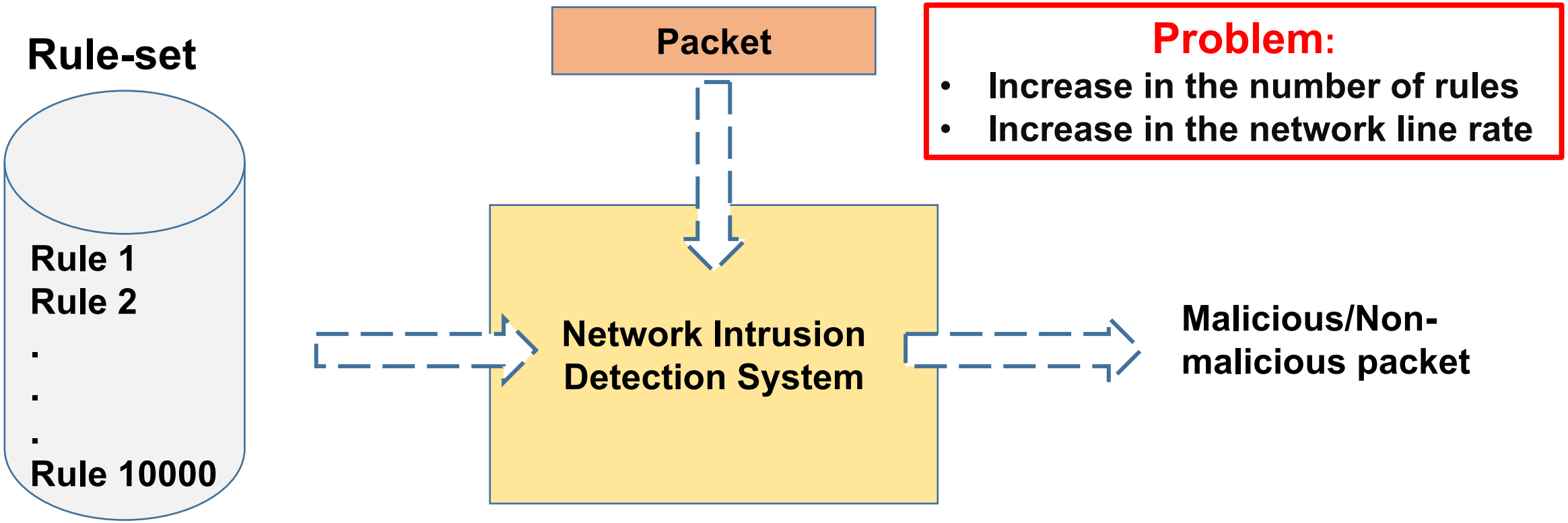
Problem: Processor / Memory Performance Gap



Source: David Patterson, UC Berkeley

Scalable and High-Performance Techniques Are Needed for Pattern Processing

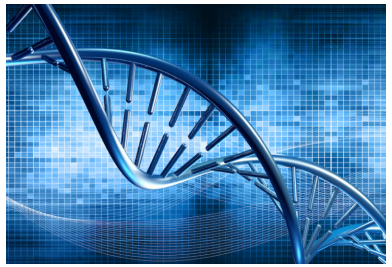
- Incoming packet is checked against every single rule of the database



Pattern Recognition Importance

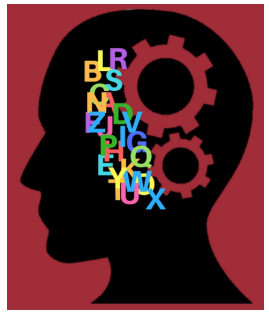
Network security

Bioinformatics



Data mining

NLP

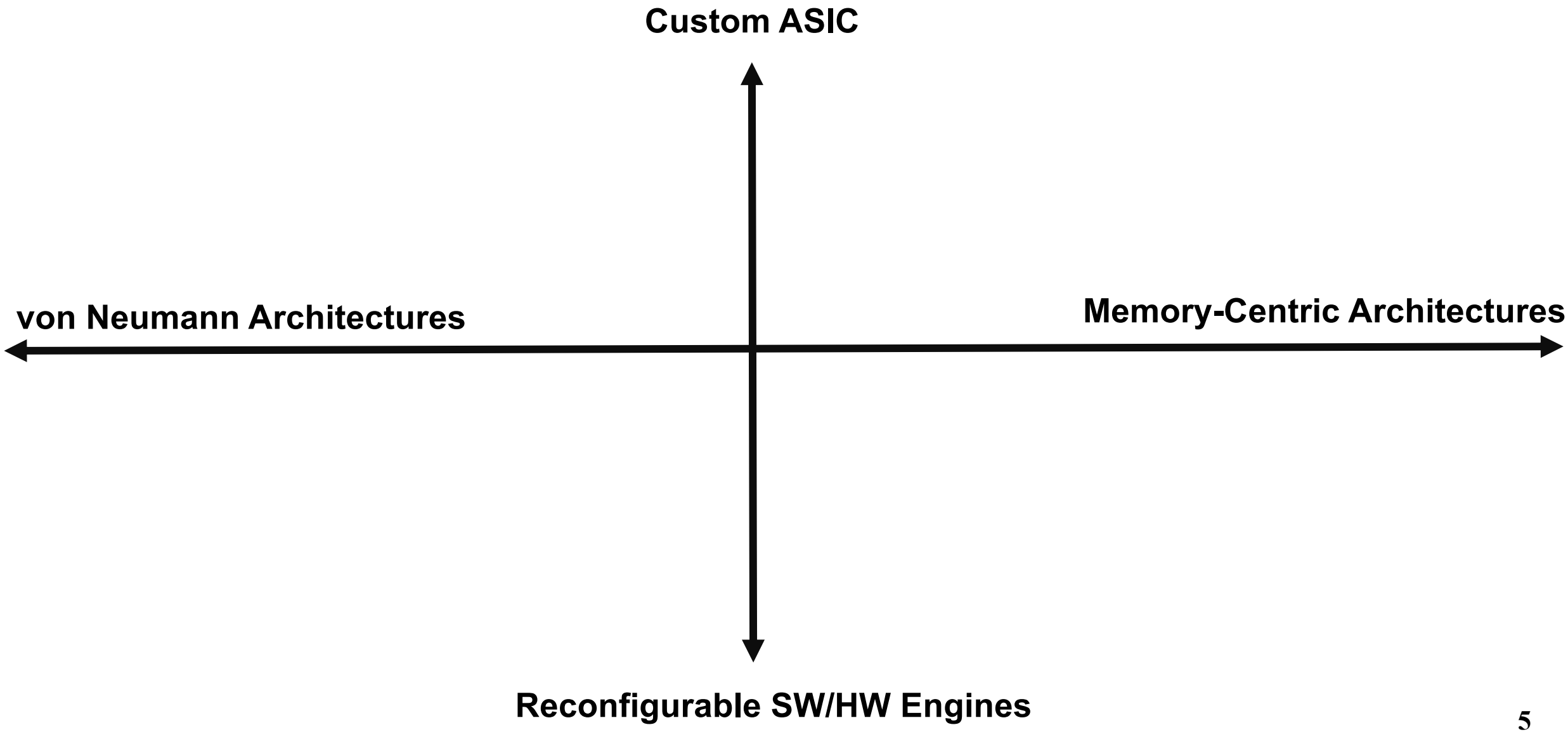


Patterns are often complex

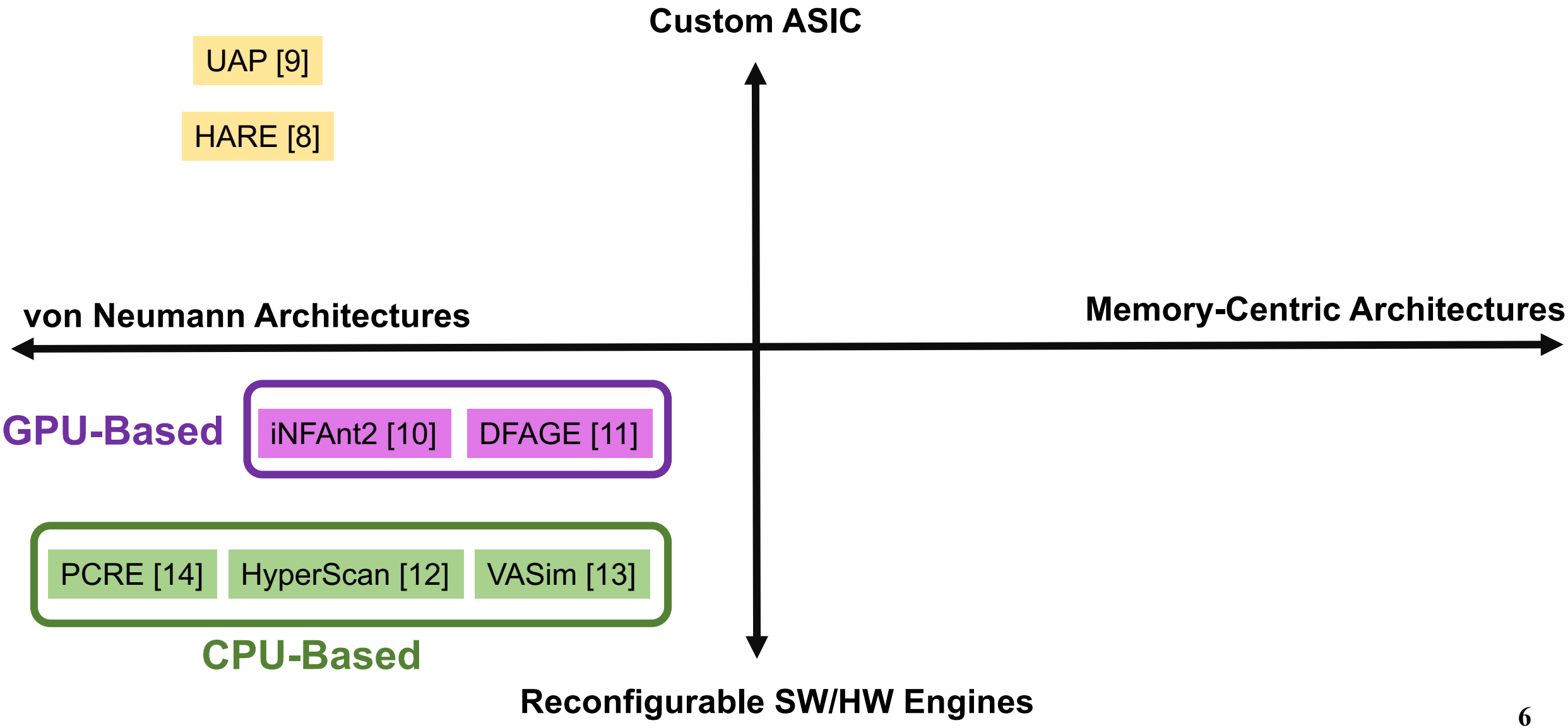
Thousands of patterns need to be processed in parallel

Regular Expressions = **Finite Automata**

Existing Automata Processing Solution



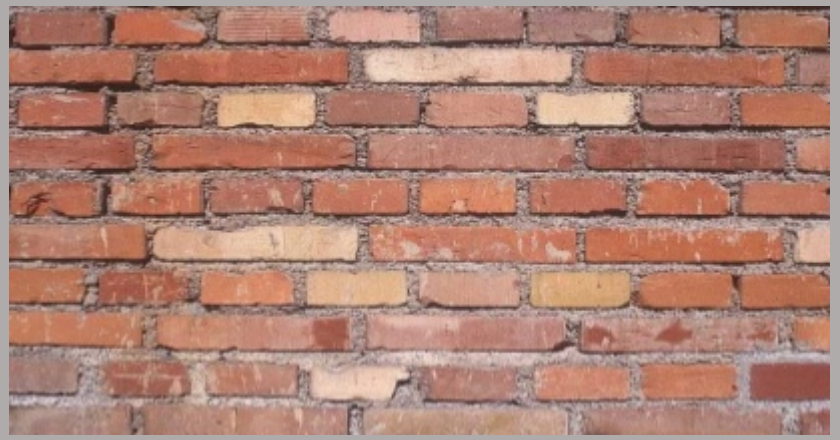
Existing Automata Processing Platforms



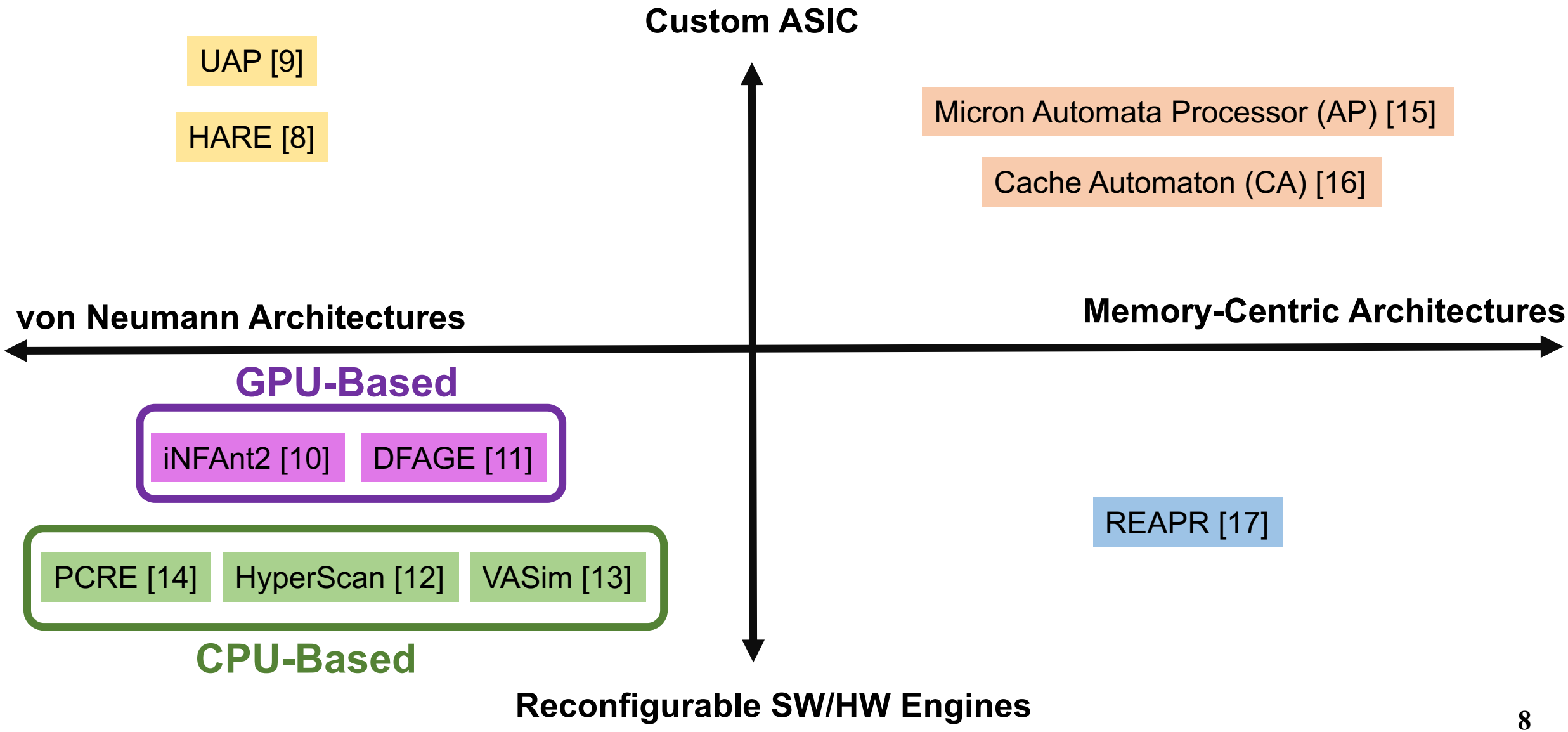
Existing Automata Processing Platforms

Problem: von Neumann processors easily become **memory bound**

- Unpredictable behavior
  - Branch mispredictions
- Irregular access pattern
  - Cache-miss
- Many parallel state transitions
  - Saturate memory bandwidth

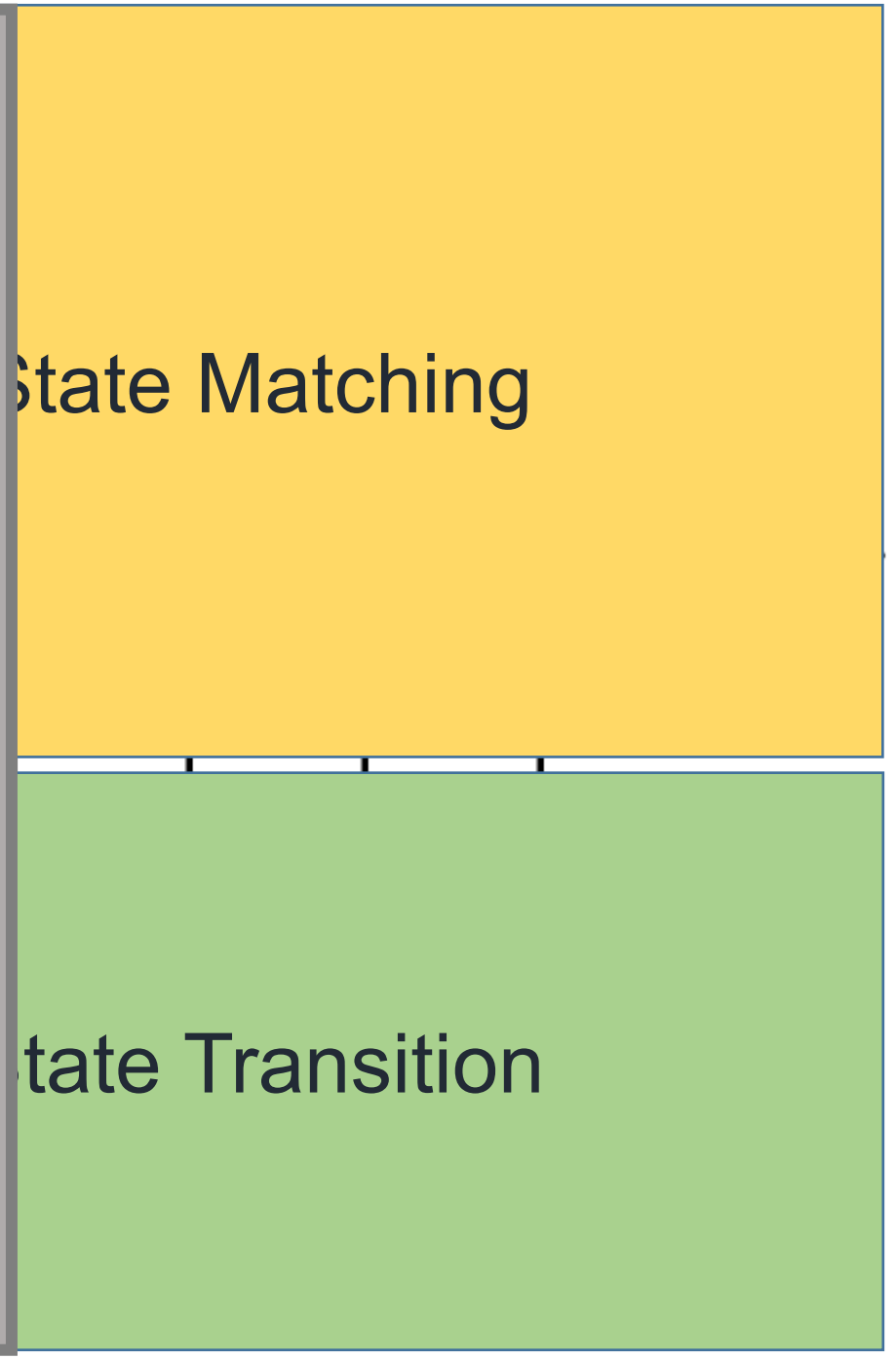


Existing Automata Processing Platforms

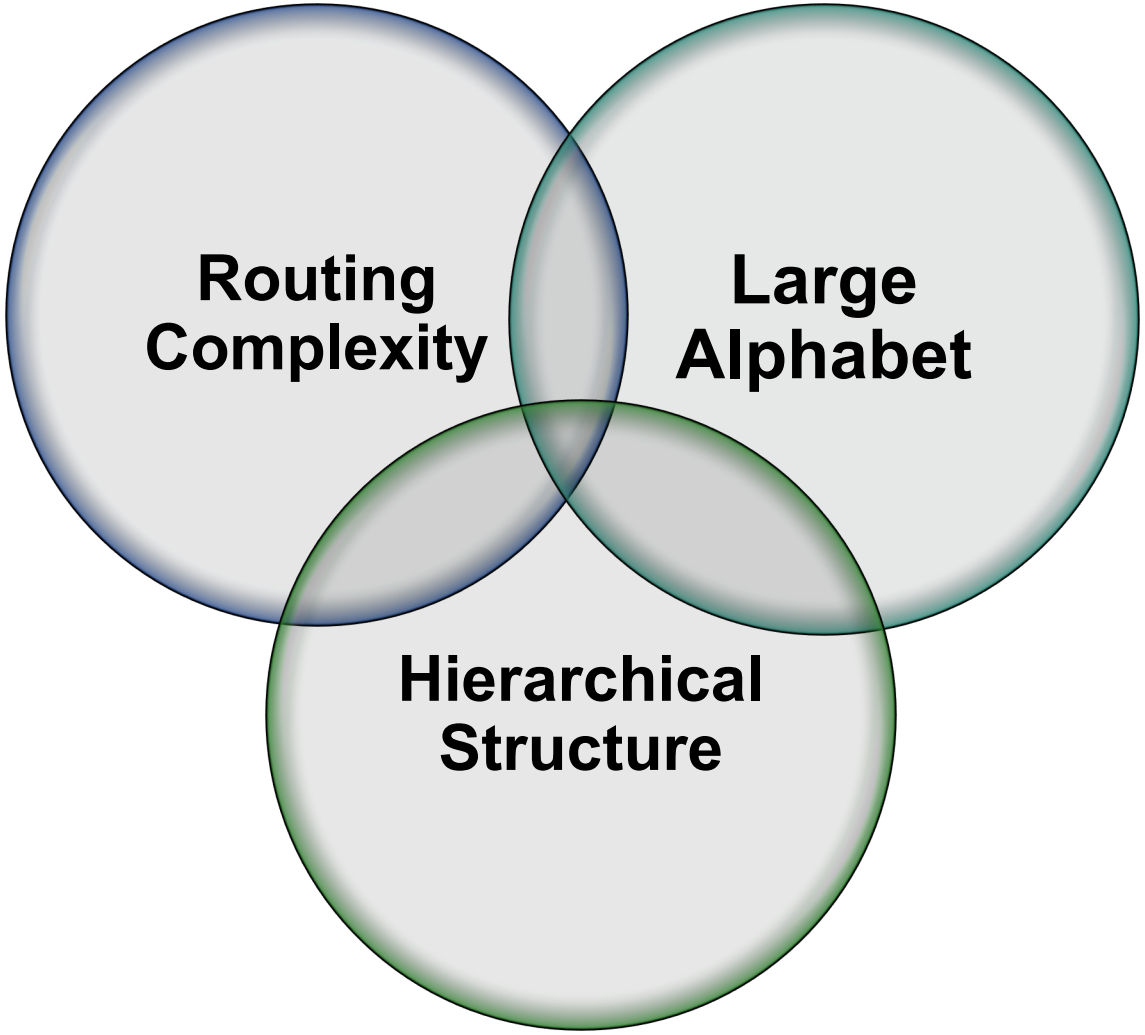


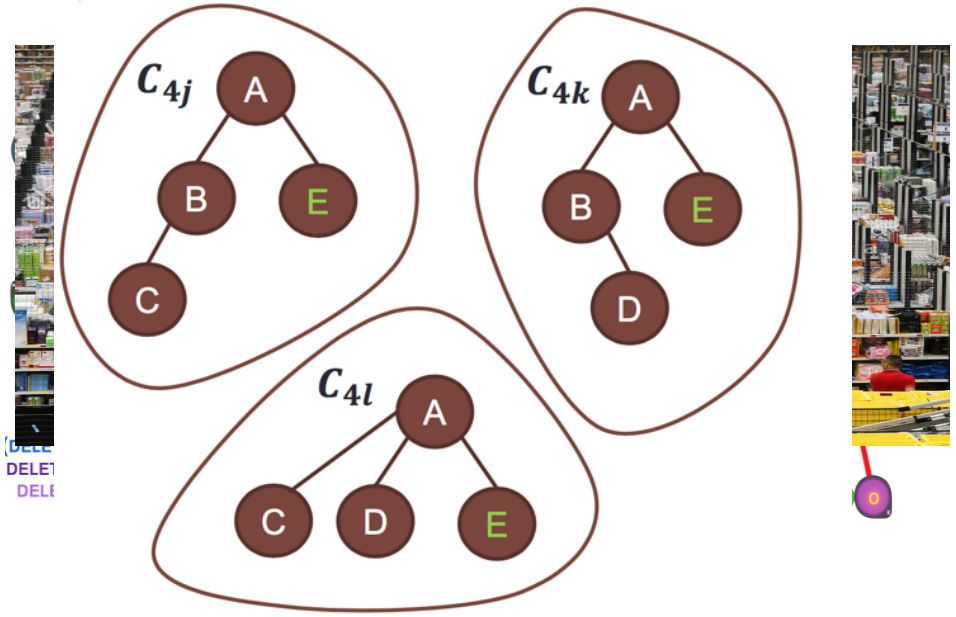
Problem with existing memory-centric architectures

- State Matching
  - High overhead SRAM or high-latency DRAM
- Interconnect architecture
  - Congestion
  - Underutilization
- Alphabet size
  - Feasibility? Overhead?
- Computation power?



Problem: Patterns can be very complex!





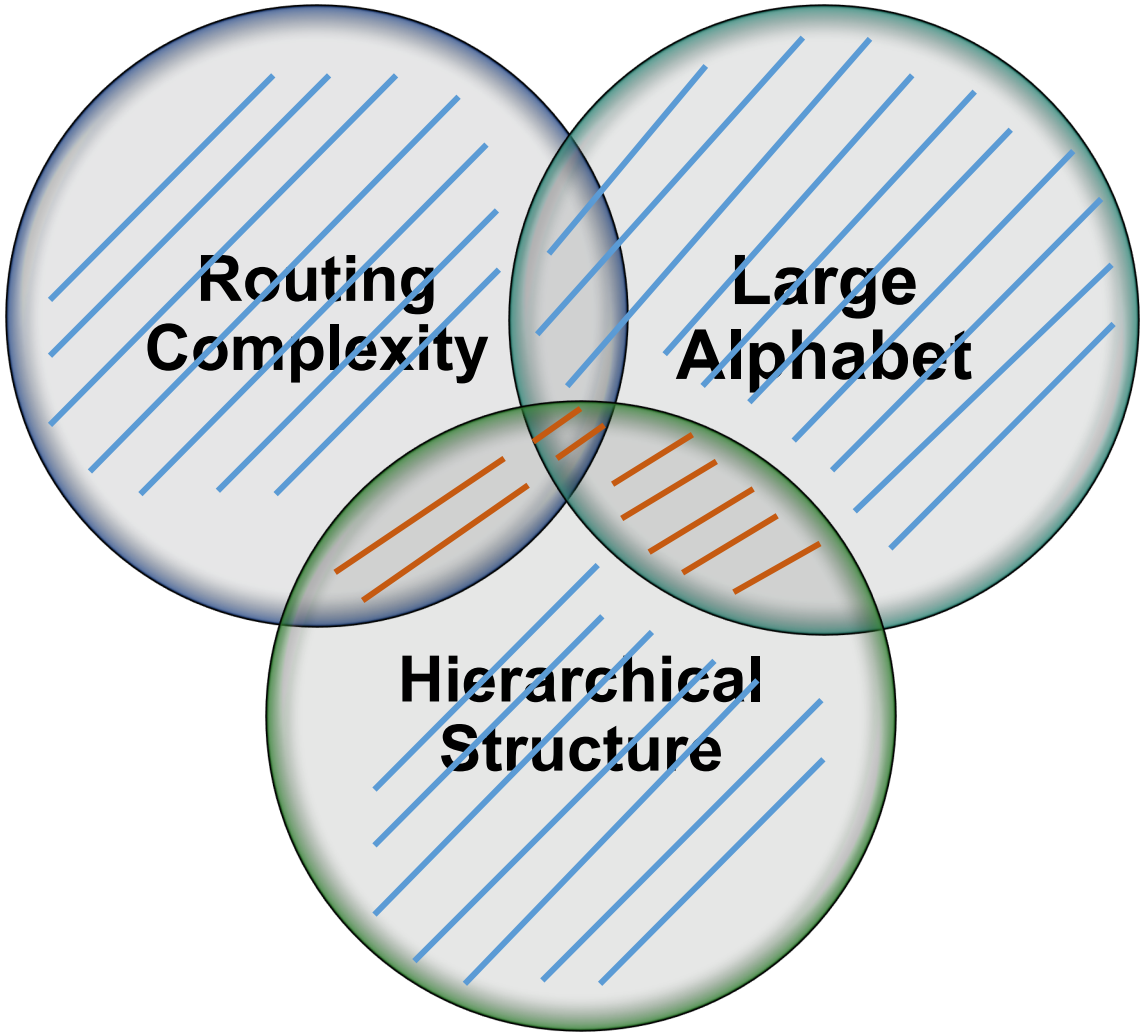
Research Questions

We hypothesize that **an efficient interconnect architecture**, **a more computationally powerful design**, **and flexible bitwidth processing** **can unleash in-memory processing benefits for more complex pattern recognition tasks.**

Overview of My Dissertation Work

1	• **Rule-Based Part-of-Speech Tagging Using Spatial Architectures**	KDD'18, IISWC'18
2	• **Hierarchical Pattern Mining on the Automata Processor**	ICS'17, CF'16, IJPP'17, IISWC'16
3	• **A Scalable In-SRAM Architecture for Pushdown Automata**	MICRO'18
4	• **A Scalable In-Memory Interconnect for Automata Processing**	CAL'19 MICRO'19
5	• **FlexAmata: A Flexible Automata Processing Engine**	MICRO'19

We provide solutions to these problems



Proposed and Evaluated

Future work

- **Novel Automata Application in Natural Language Processing**

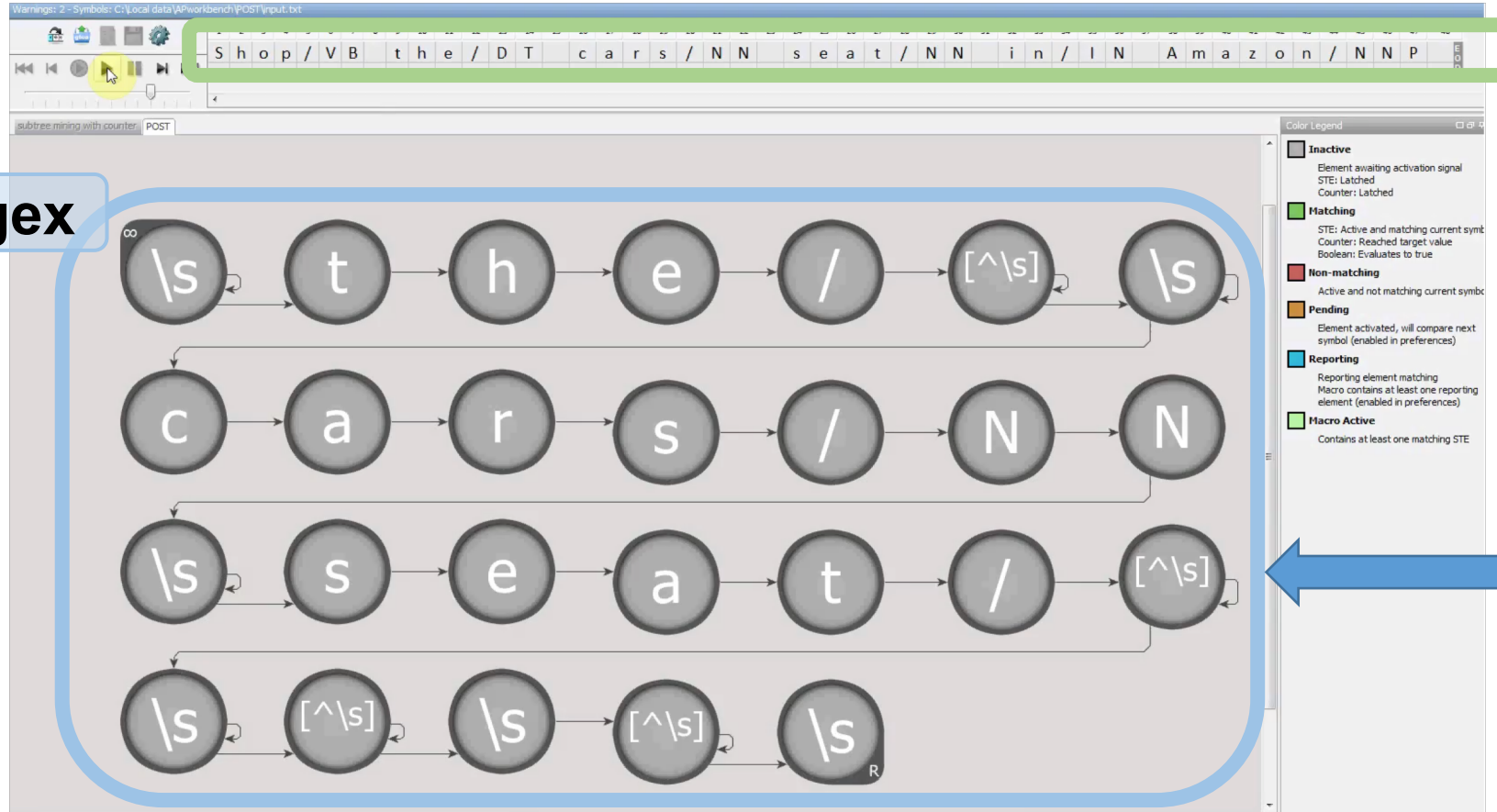
Accelerating Rule-Based Methods in Natural Language Processing on Automata Hardware Accelerators

Main Idea: Re-evaluating Rule-based Methods in NLP

Rule → NN → JJ if Word:the@[-1] & Word:car@[0] & Word:seat@[1]

Regex → NN → JJ : /\s+the\/[^\s]+\s+car\/NN\s+seat\/[^\s]+\s+[^\s]+\s+[^\s]+\s/

Input → Shop/VB the/DT car/NN seat/NN in/IN Amazon/NNP



Many of these rules can be run in parallel

Main insights from this study

- Learning a larger number of more complex rules **increases the accuracy** of rule-based approaches
- Automata hardware accelerators can run **thousands of these patterns** in parallel with **minimal overhead**
- Our solution is **two orders of magnitude faster** than ML-based taggers on GPU, while achieving **competitve accuracy**

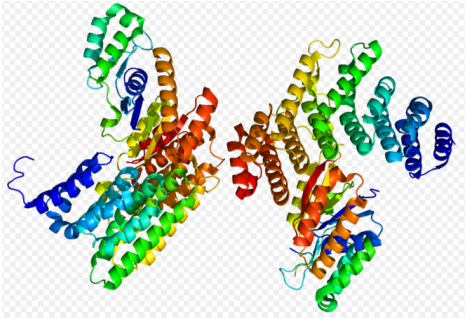
KDD'18 | ANMLZoo benchmark suite, IISWC'16 | AutomataZoo benchmark suite, IISWC'18

Accelerating Subtree Mining on the Automata Processor

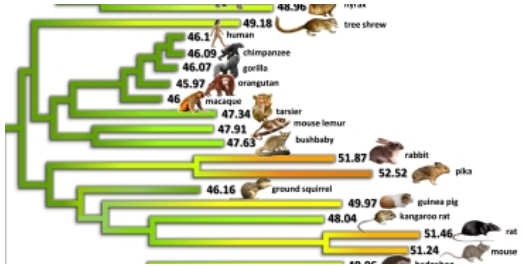
Subtree Mining: challenges and opportunities

Web mining
(sentiment analysis)

Semi-structured data
(NLP, parsers)



Bioinformatics
(protein sequences)



Phylogenetic
(crop improvement)

Processing tree-shaped patterns is **more complex** than sequences

Many of these applications **need high-throughput** processing

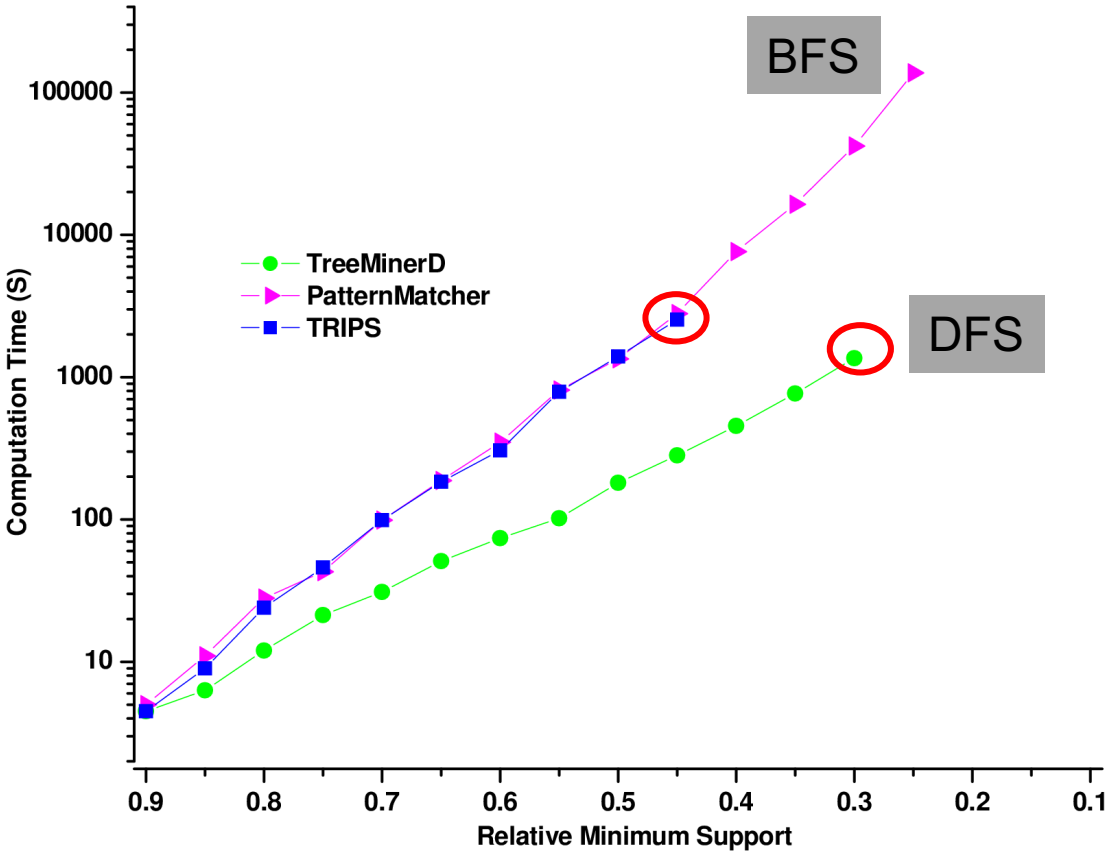
Tree-shaped patterns **cannot** be represented with finite automata

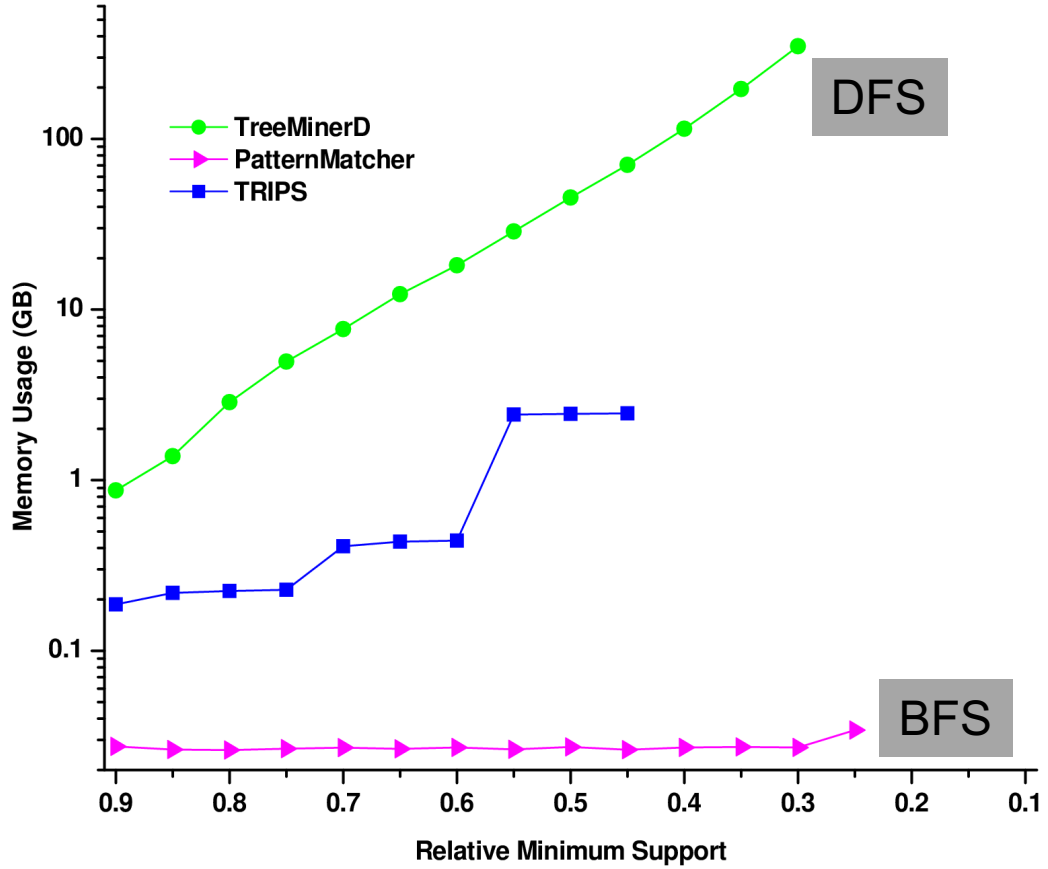
We propose an approximate solution for tree-shaped pattern processing

Problems with Current Solutions on von Nemumann architectures

	Pros	Cons
BFS	Massive pruning, Memory efficient	Multi-pass of dataset, slow
DFS	Fast	Little pruning opportunity, Memory-hungry

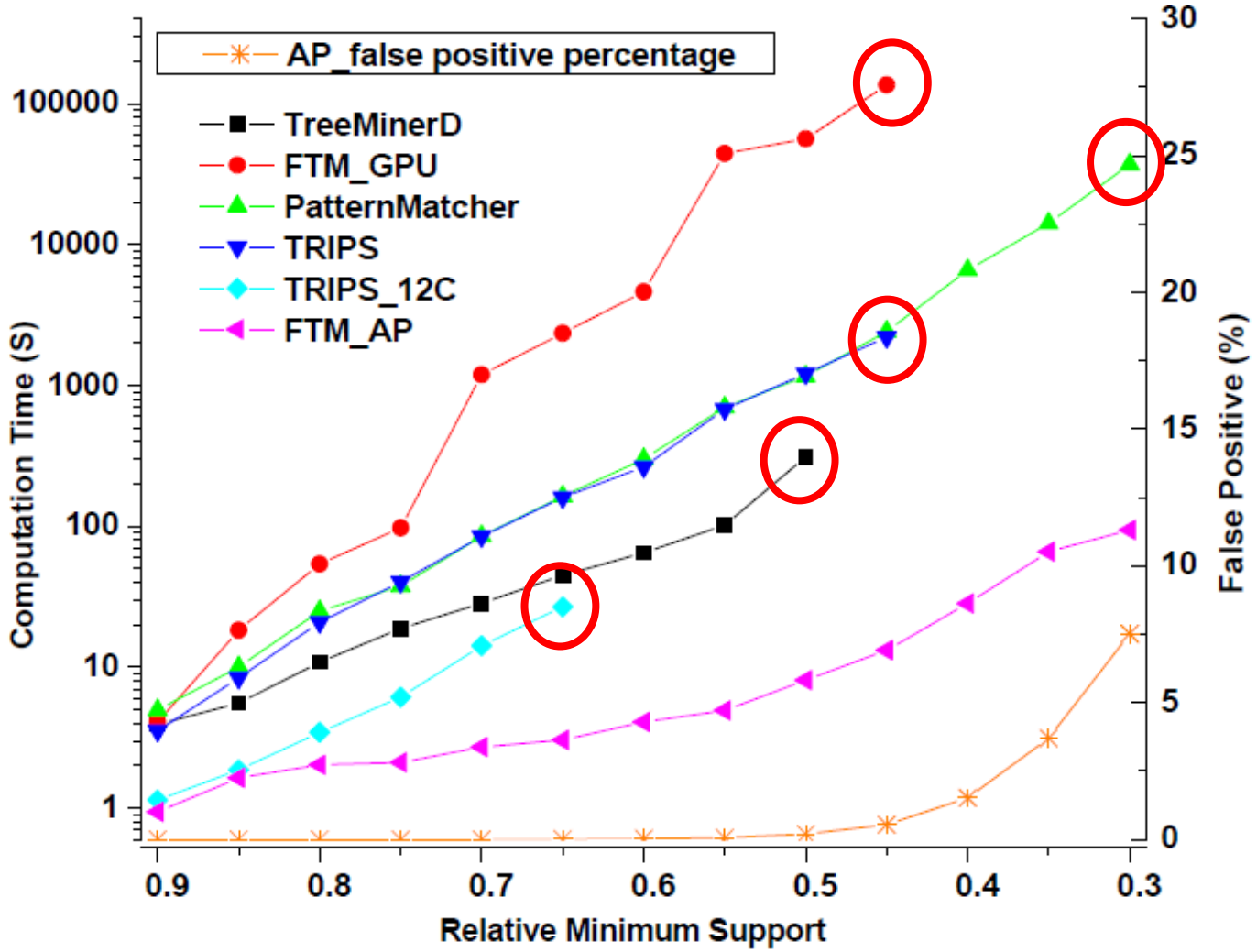
*BFS and DFS refer to candidate generation approach, not tree traversal





FTM-AP vs Other FTM Algorithms

Trade-off between *speed* and *accuracy* of the AP solution vs the existing FTM implementation



Dataset: Treebank

Main insights from this study

- Existing CPU/GPU solutions fail to process big datasets
- **A scalable** approximate solution on the Automata Processor
  - Up to 262X speedup over the best running solution
  - Up to 7% false positives
- Hybrid approach for exact solution
  - 6X speedup over the best running solution
- Structure independent
  - CPU/GPU performance → **depends** on the tree features
  - Automata processing performance → **independent** from structure

Automata Application in Data Mining: Additional Contribution

Elaheh Sadredini, Reza Rahimi, Ke Wang, Kevin Skadron.
ACM International Conference on Supercomputing (ICS'17)

Ke Wang, **Elaheh Sadredini**, Kevin Skadron.
"Sequential pattern mining with the Micron Automata Processor"
ACM International Conference on Computing Frontiers (CF'16), **won best paper award**

Ke Wang, **Elaheh Sadredini**, Kevin Skadron.
"Hierarchical Pattern Mining with the Micron Automata Processor"
International Journal of Parallel Programming (IJPP'17)

AutomataZoo benchmark suite, IISWC'18

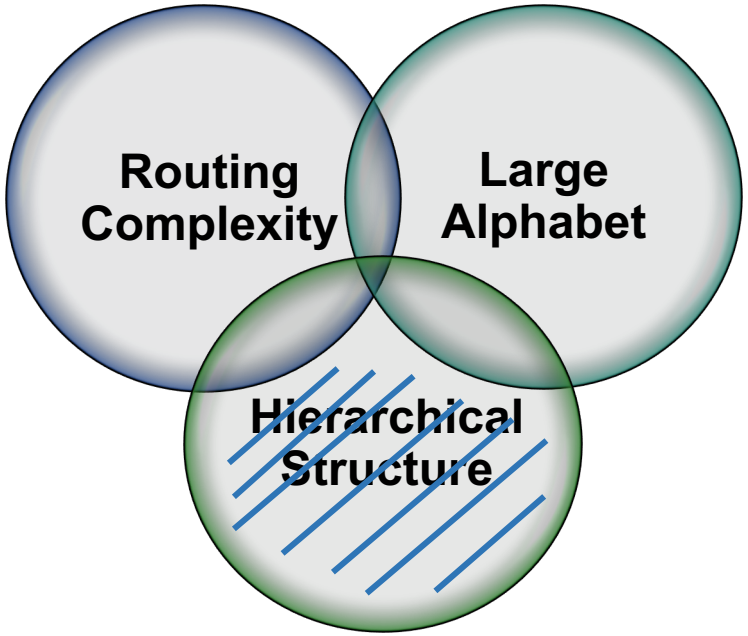
ANMLZoo benchmark suite, IISWC'16

Overview of My Dissertation Work

1	• Rule-Based Part-of-Speech Tagging Using Spatial Architectures	KDD'18, IISWC'18
2	• Hierarchical Pattern Mining on the Automata Processor	ICS'17, CF'16, IJPP'17, IISWC'16
3	• **A Scalable In-SRAM Architecture for Pushdown Automata**	MICRO'18
4	• A Scalable In-Memory Interconnect for Automata Processing	CAL'19 MICRO'19
5	• FlexAmata: A Flexible Automata Processing Engine	MICRO'19

- **Novel Architecture Exploration**

A Scalable In-SRAM Architecture for Pushdown Automata



Kevin Angstadt, Arun Subramaniyan
Westley Weimer, Reetuparna Das
University of Michigan

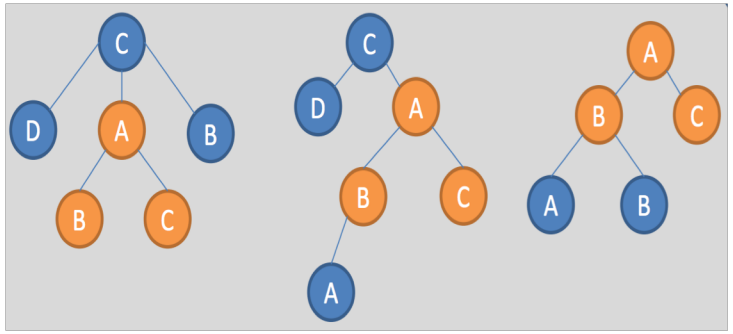
Elaheh Sadredini, Reza Rahimi
Kevin Skadron
University of Virginia

IEEE/ACM International Symposium on Microarchitecture (MICRO'51)
October 2018

Problem: Existing Automata Processing Platforms Cannot Support Computation for Tree-Structured Data

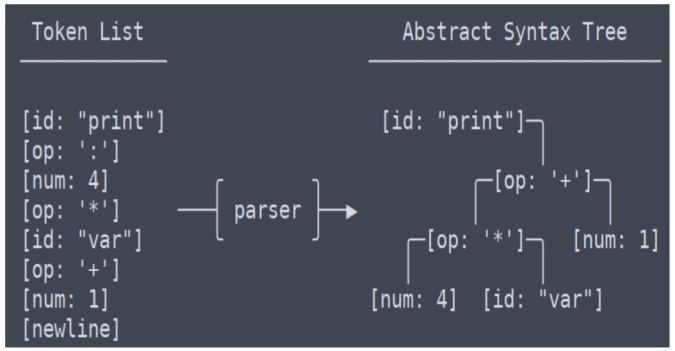
von Neumann Architectures:

- Irregular access patterns
- Branch misprediction



Memory-Centric Architectures:

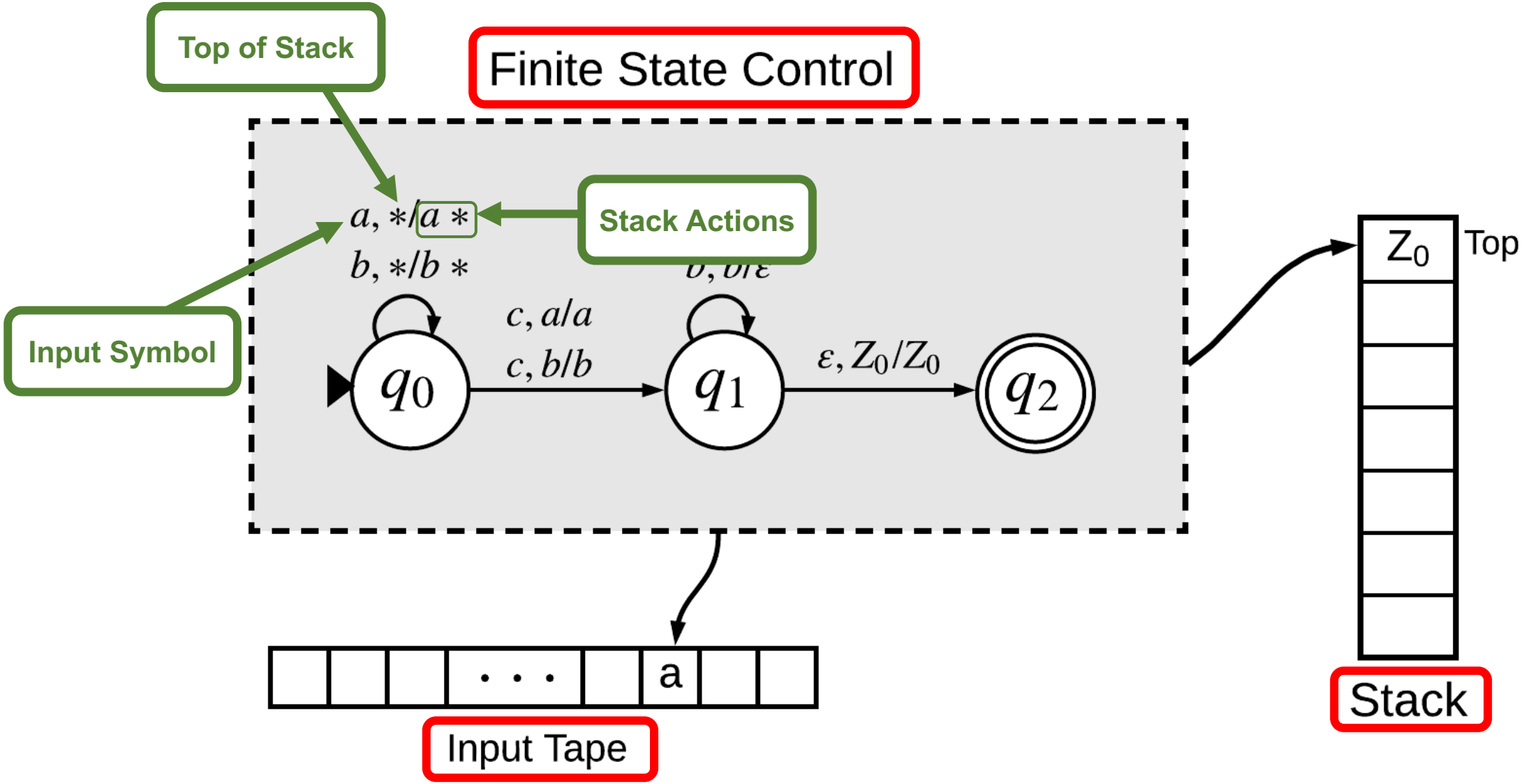
- Originally designed for NFA processing
- Do not support PDA (pushdown automata)



Main idea:

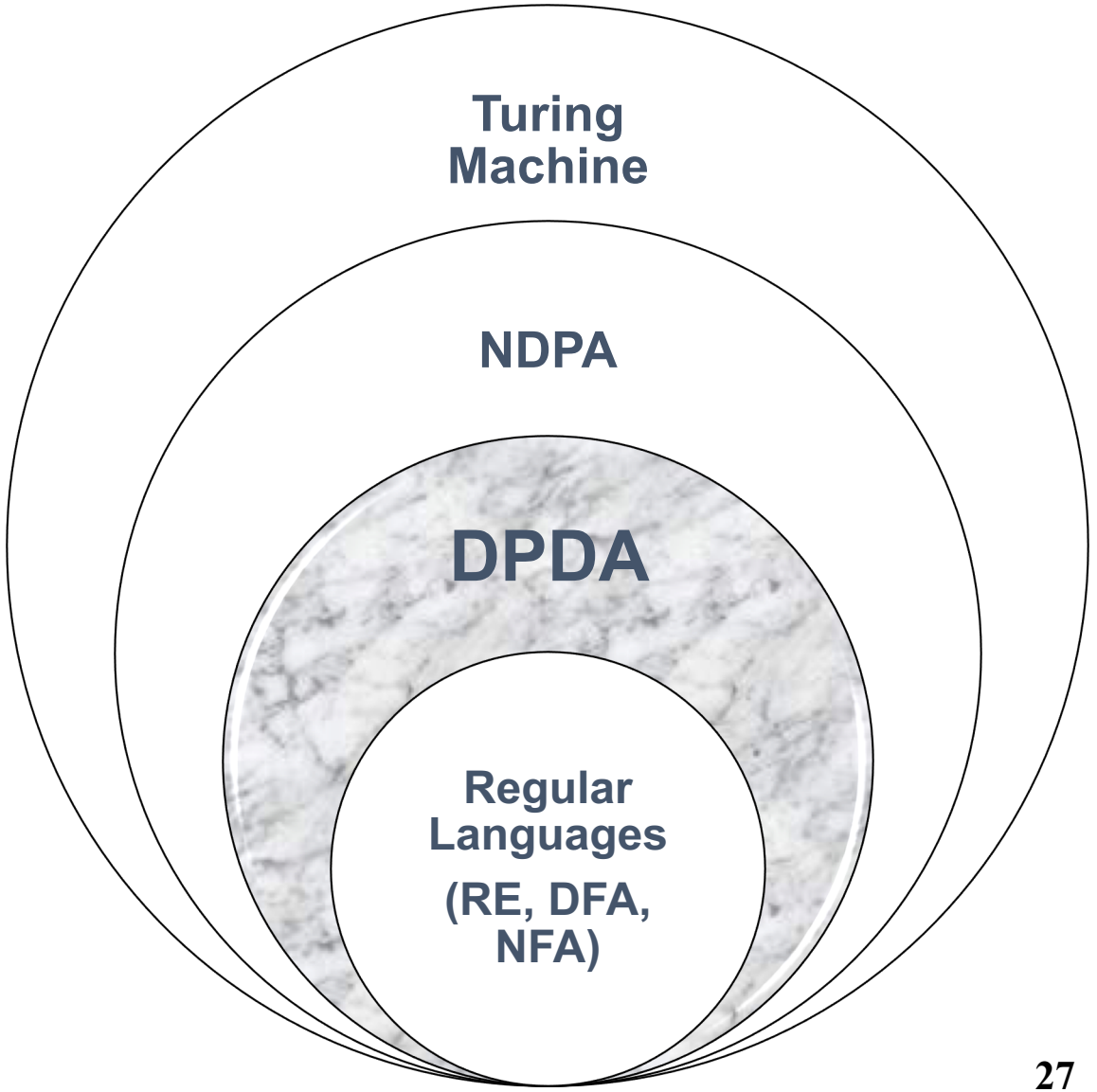
Proposing a scalable in-memory solution for parallel pushdown automata processing

Pushdown Automata Refresher



Solution: Deterministic Pushdown Automata (DPDA)

DPDA are powerful enough to process tree-shape structures in data mining and parse most programming languages



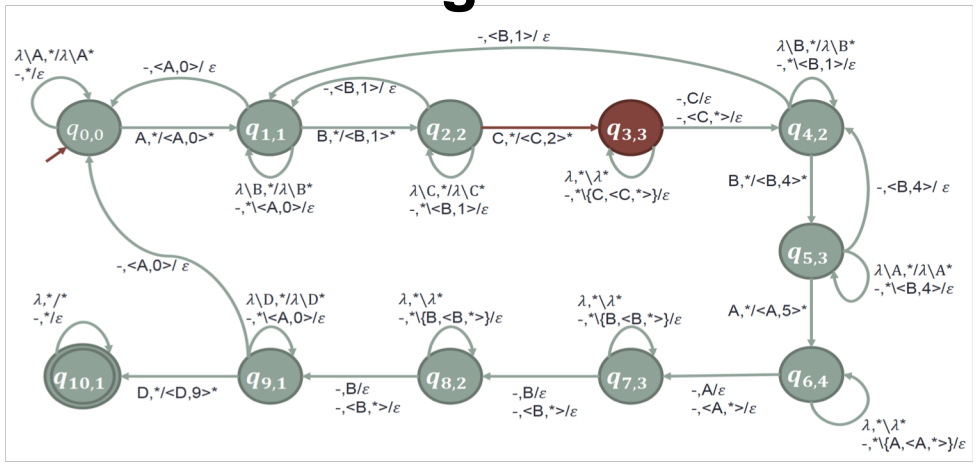
Grammar

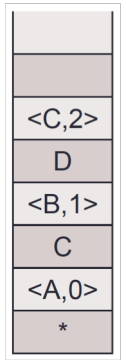
$(q_{00}, \tau_0, *) \rightarrow (q_{11}, (\tau_0, 0)*)$
$(q_{00}, \{\lambda \backslash \tau_0\}, *) \rightarrow (q_{00}, \{\lambda \backslash \tau_0\}*)$
$(q_{00}, -, *) \rightarrow (q_{00}, \varepsilon)$

$(q_{ij}, \tau_i, *) \rightarrow (q_{i+1j+1}, (\tau_i, i)*)$
$(q_{ij}, \{\lambda \backslash \tau_i\}, *) \rightarrow (q_{ij}, \{\lambda \backslash \tau_i\}*)$
$(q_{ij}, -, (\tau_p, p)) \rightarrow (q_{pj-1}, \varepsilon)$ where q_{pj-1}
$(q_{ij}, -, \lambda \backslash (\tau_p, p)) \rightarrow (q_{ij}, \varepsilon)$

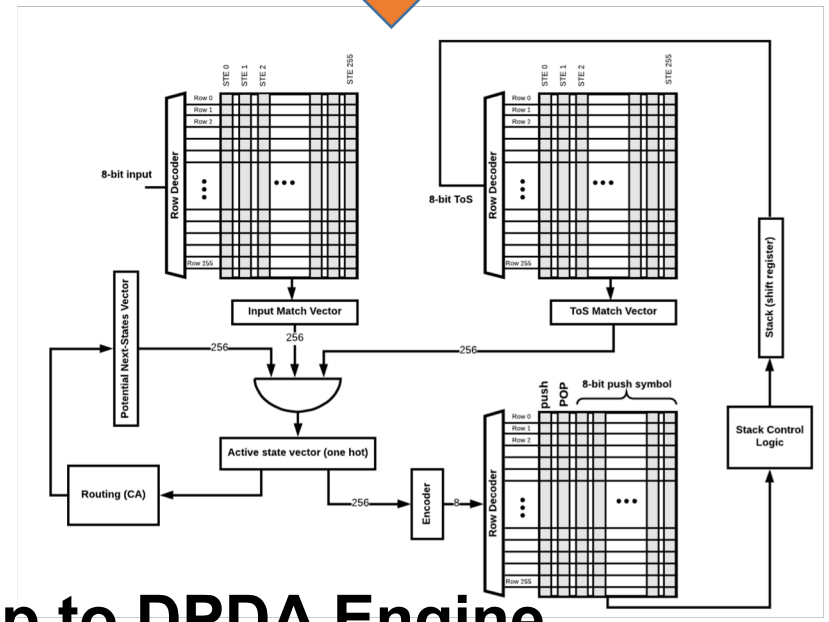
$(q_{ij}, -, *) \rightarrow (q_{i+1j+1}, \varepsilon)$
$(q_{ij}, \{\lambda \backslash -\}, *) \rightarrow (q_{ij}, \{\lambda \backslash -\}\varepsilon)$

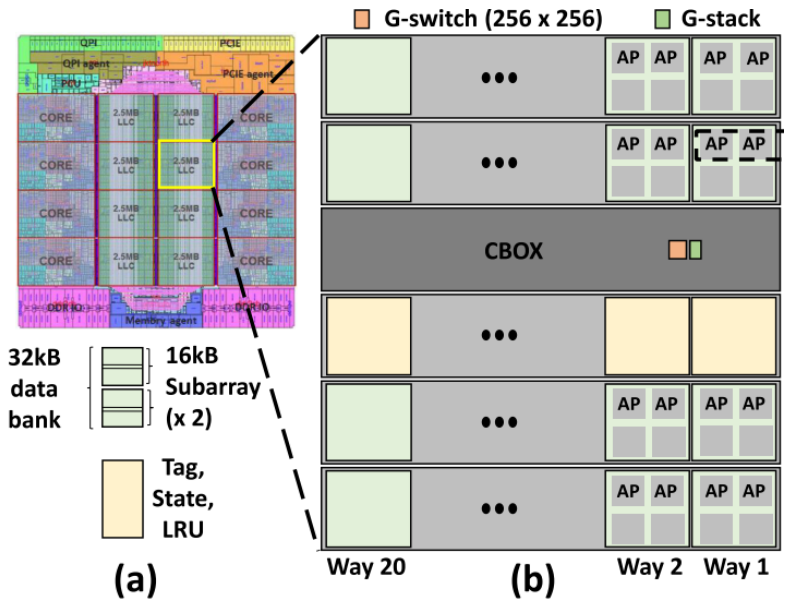
Parsing Automata





Homogeneous DPDA



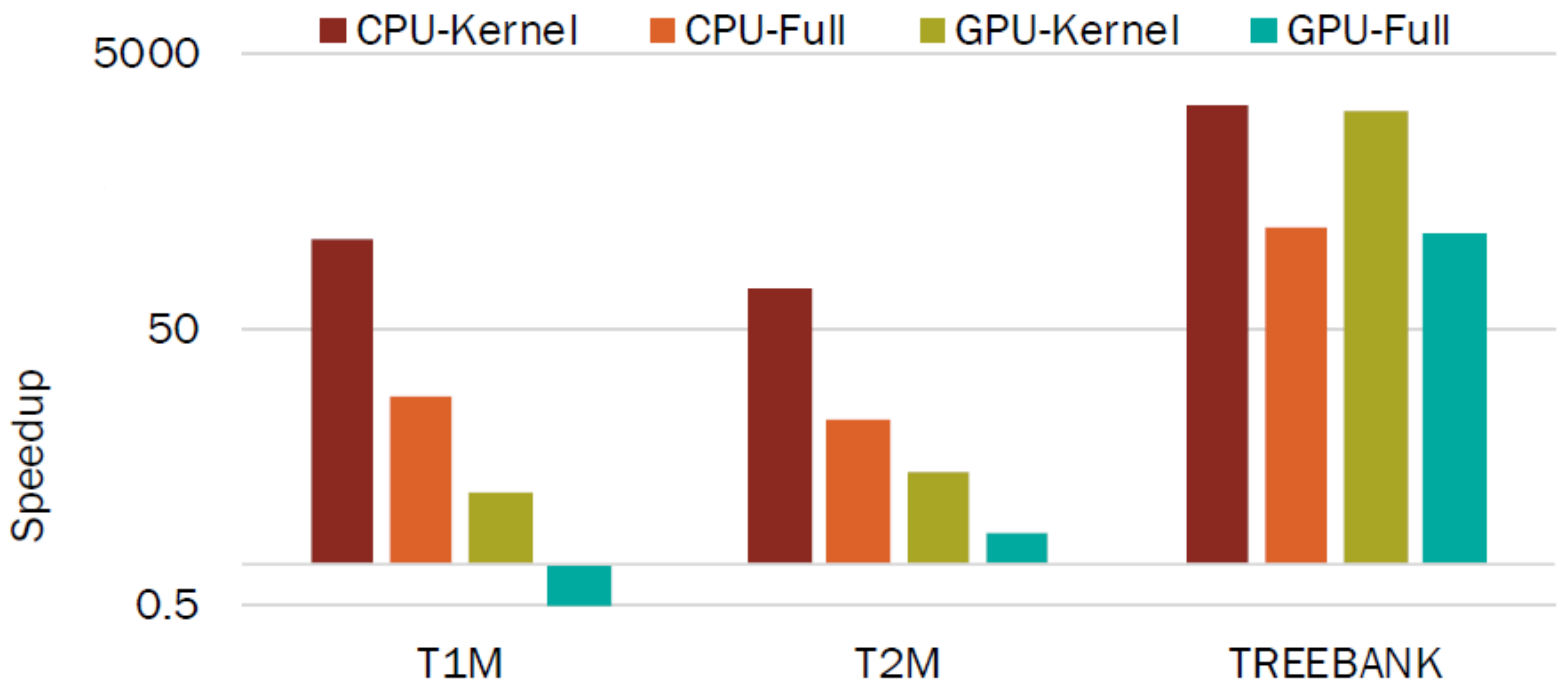


Processor for DPDA Acceleration

Map to DPDA Engine

Performance Results: Subtree Inclusion

- 67x faster than CPUs and 6x faster than GPUs for end-to-end application
- Performance is independent from tree size and complexity
- No epsilon transitions



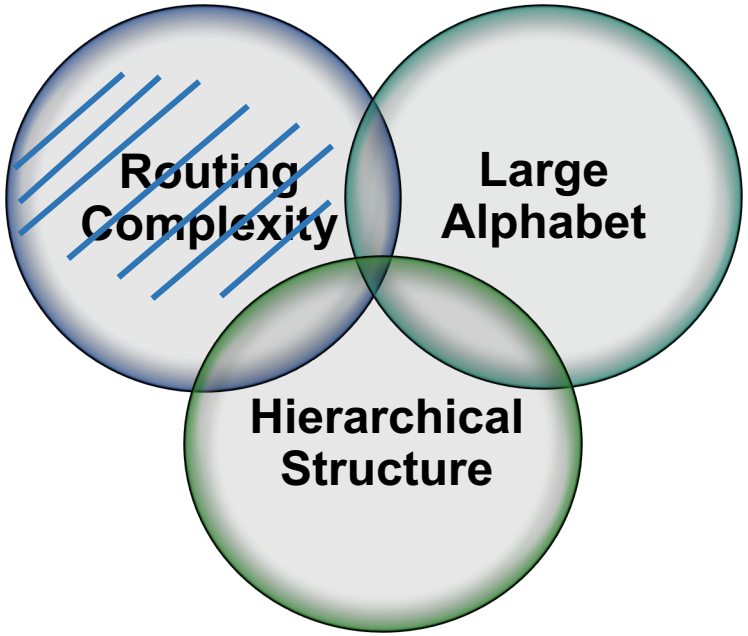
Tree mining on DPDA accelerator is about **10X faster** than tree mining on the hybrid exact method on the AP+CPU

Overview of My Dissertation Work

1	• Rule-Based Part-of-Speech Tagging Using Spatial Architectures	KDD'18, IISWC'18
2	• Hierarchical Pattern Mining on the Automata Processor	ICS'17, CF'16, IJPP'17, IISWC'16
3	• A Scalable In-SRAM Architecture for Pushdown Automata	MICRO'18
4	• **A Scalable In-Memory Interconnect for Automata Processing**	CAL'19 MICRO'19
5	• FlexAmata: A Flexible Automata Processing Engine	MICRO'19

• Novel Architecture Exploration

Scalable and Efficient in-Memory Interconnect for Automata Processing



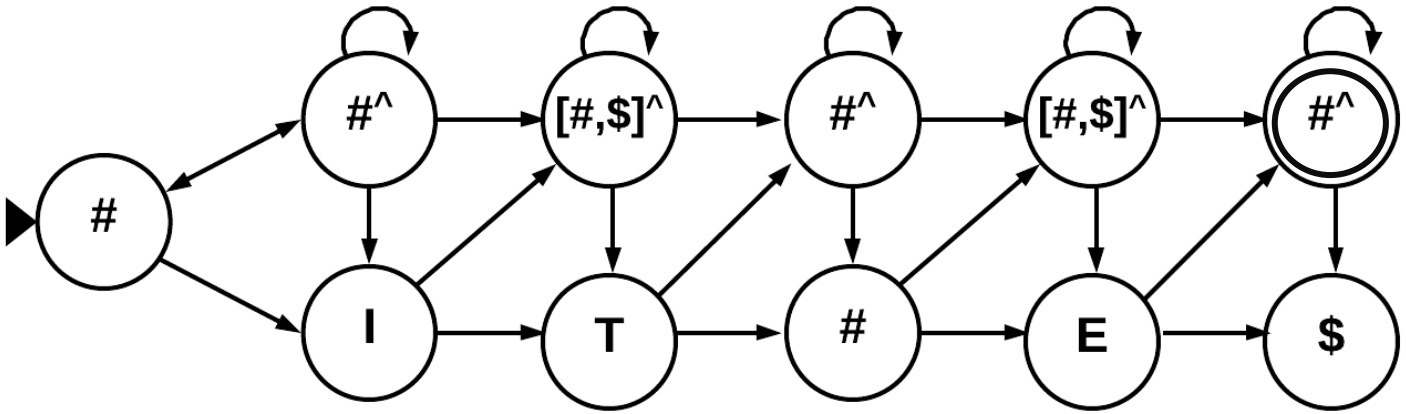
Problem: Routing Inefficiency in Existing In-Memory Automata Processing Architectures

- Automata Processor [15]
  - Routing matrix congestion
    - 13% state utilization in Levenshtein
- Cache Automaton [16]
  - Full-crossbar is excessive for interconnect
  - 0.53% of switches are utilized by ANMLZoo automata benchmark suite

Main Idea:

Designing a **low-overhead, yet flexible** routing architecture for automata processing and mapping it to a **right memory technology**

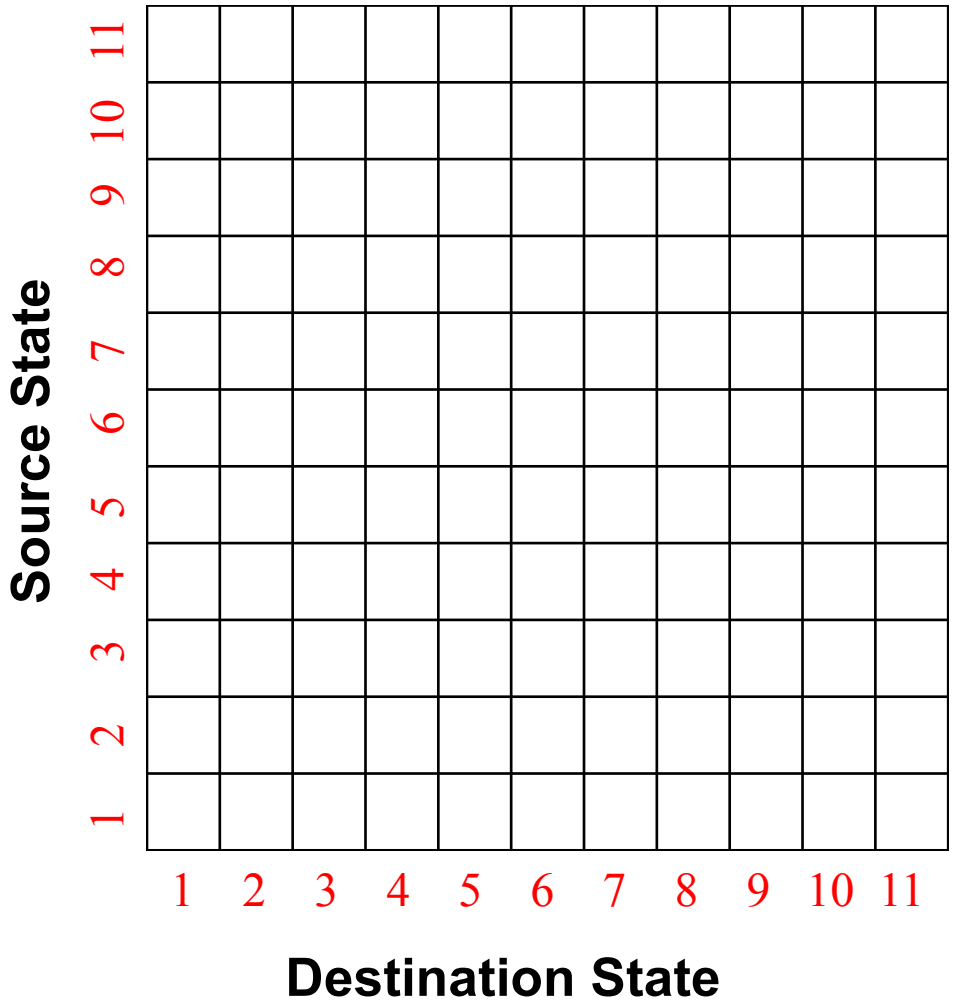
Solution: Minimizing Full-Crossbar



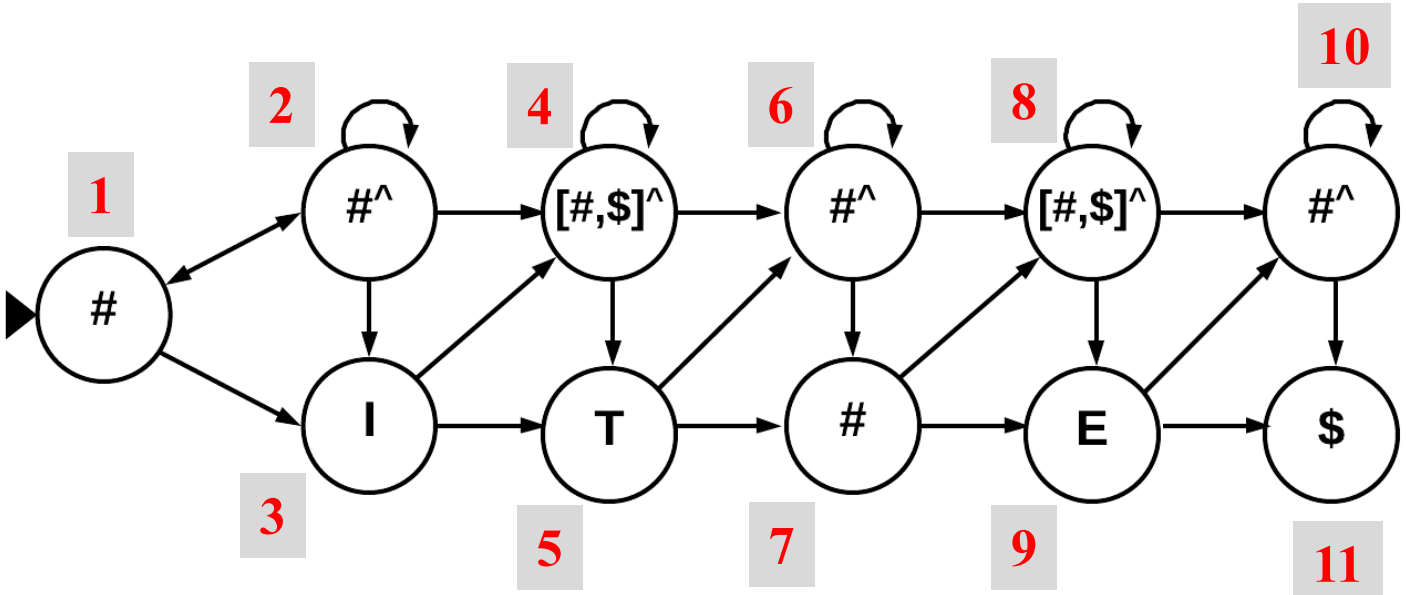
SPM Automaton

Solution: Minimizing Full-Crossbar

Connectivity Matrix



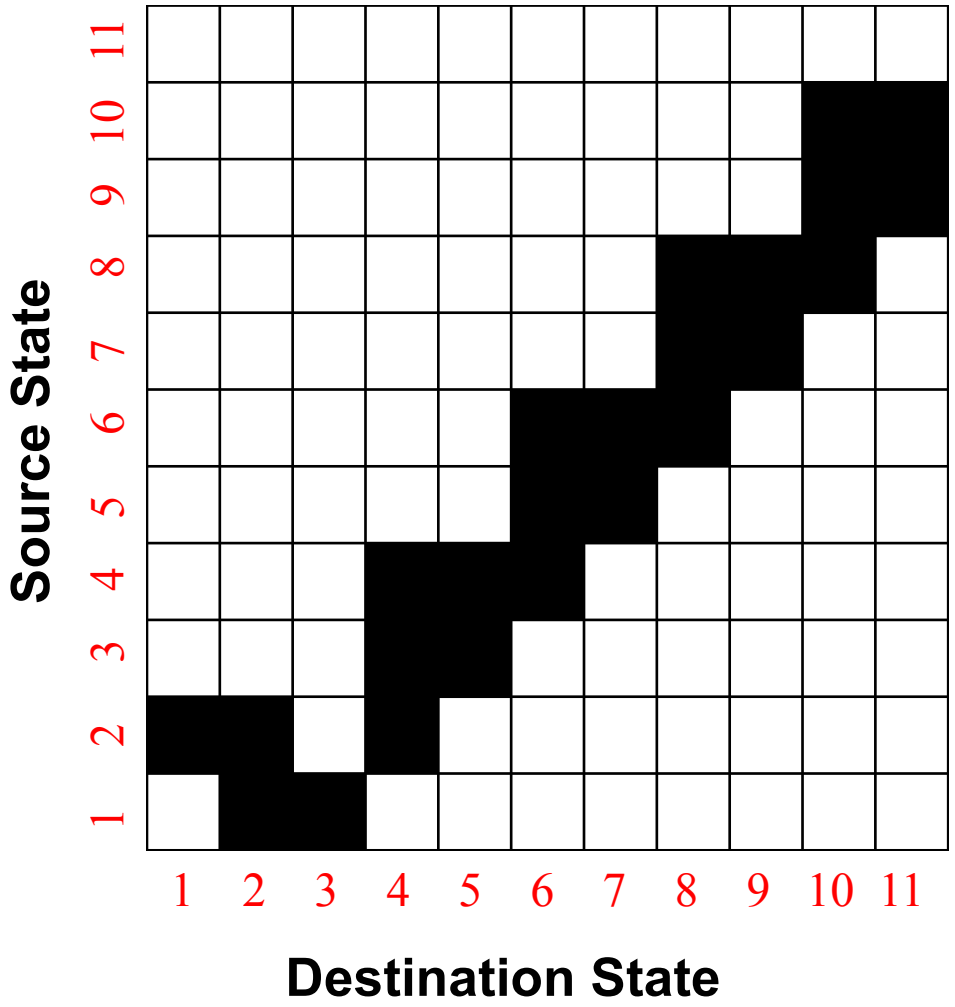
BFS Labeling



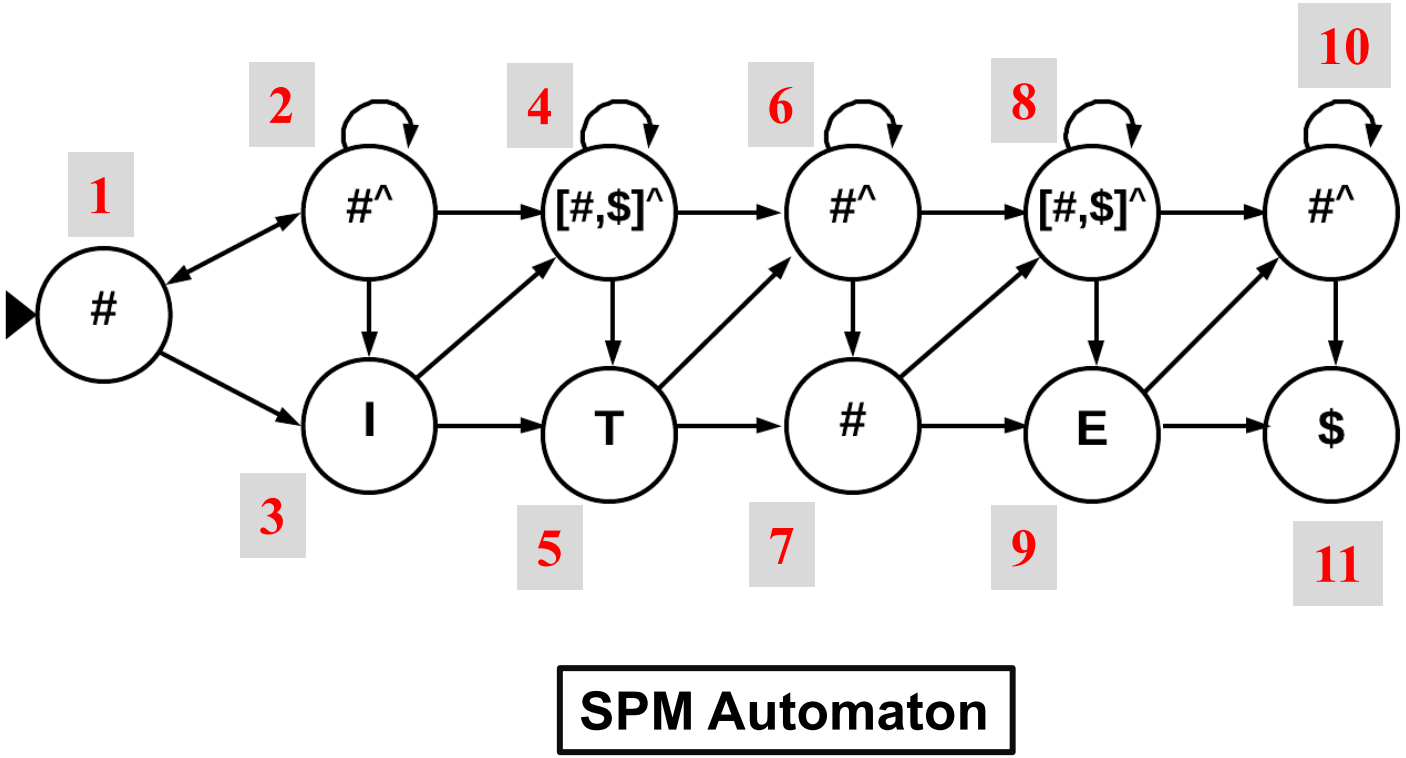
SPM Automaton

Solution: Minimizing Full-Crossbar

Connectivity Matrix



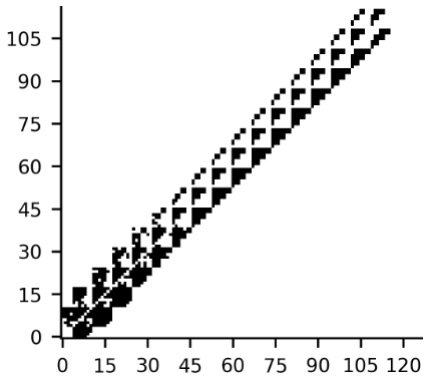
BFS Labeling



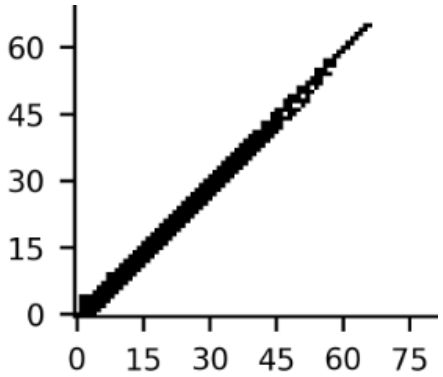
SPM Automaton

Union Heatmap of Routing Switches with BFS Labeling

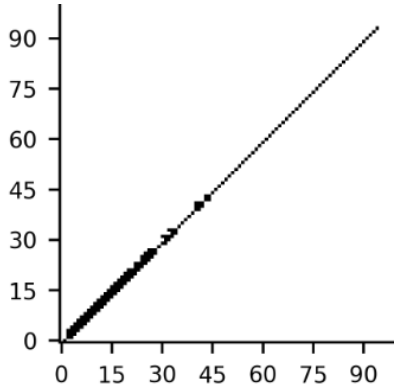
- 17 out of 19 benchmark applications show **diagonal property**



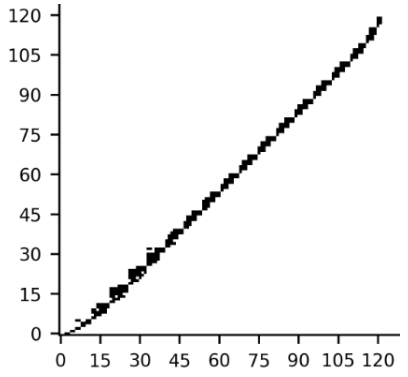
Levenshtein



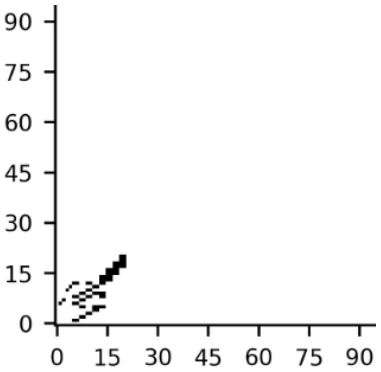
Brill



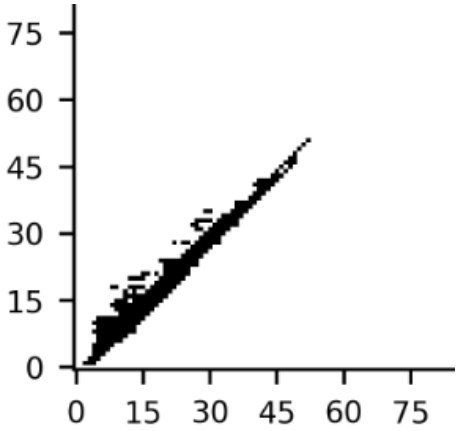
Dotstar



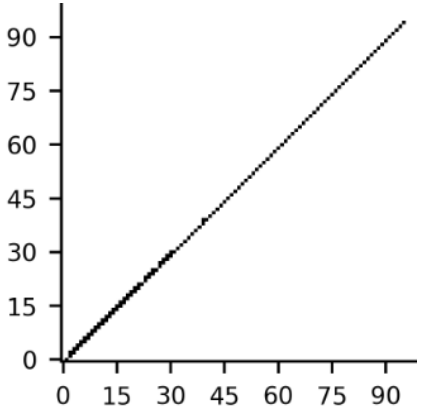
Hamming



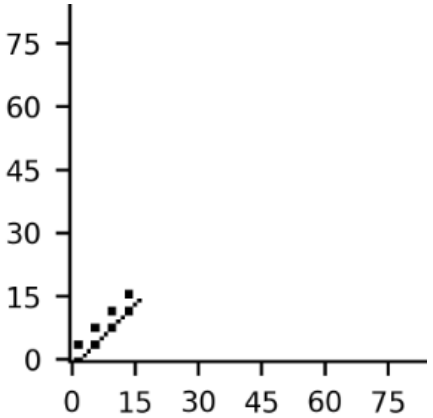
SPM



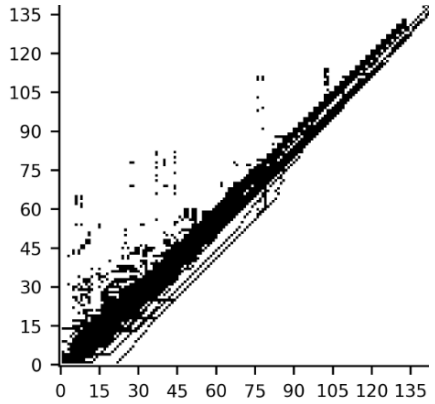
PowerEN



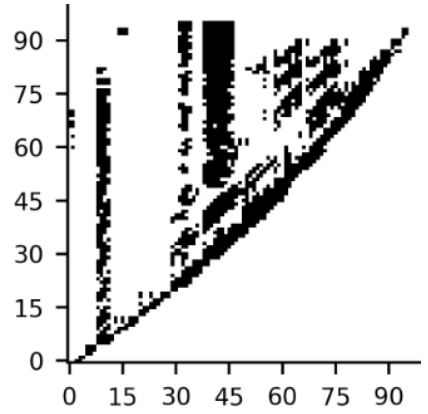
Ranges



Fermi

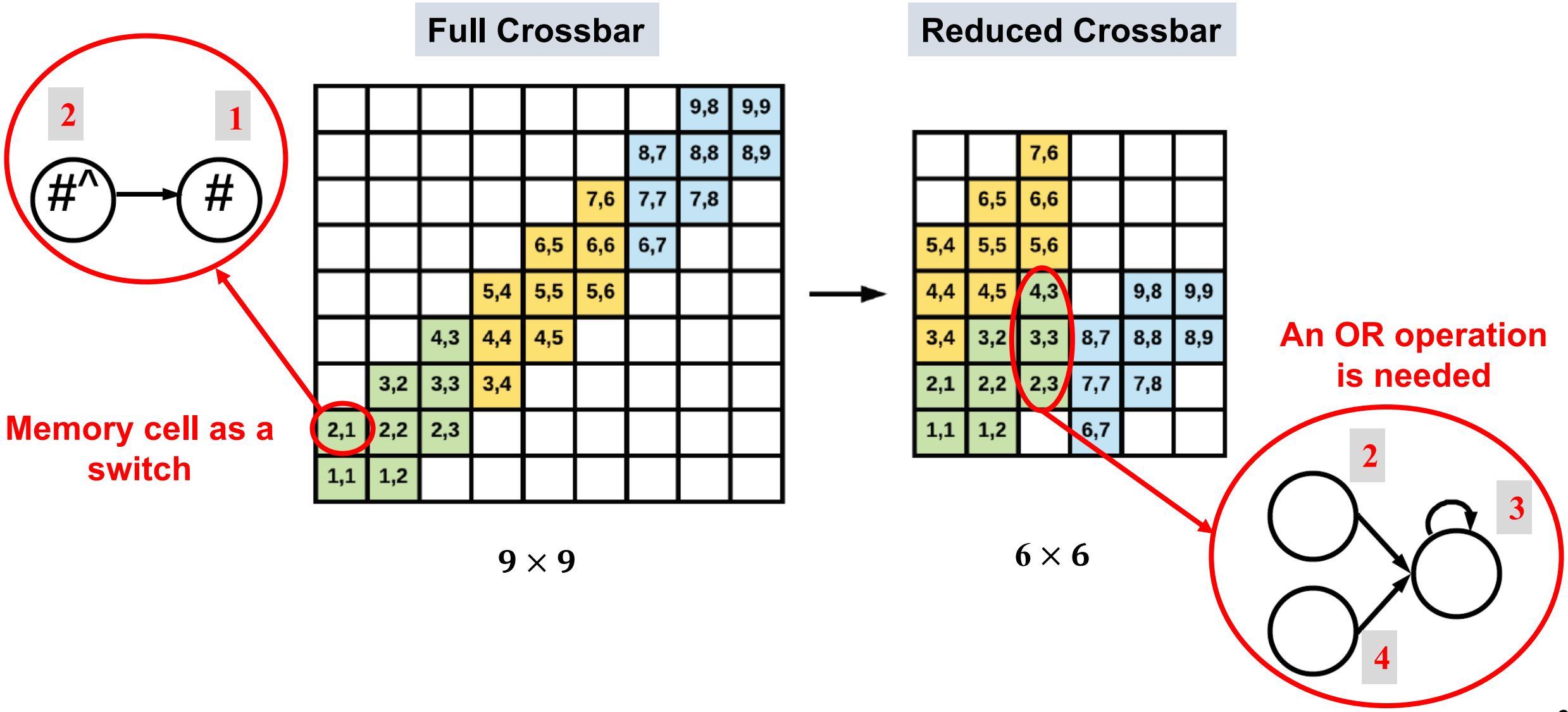


Snort



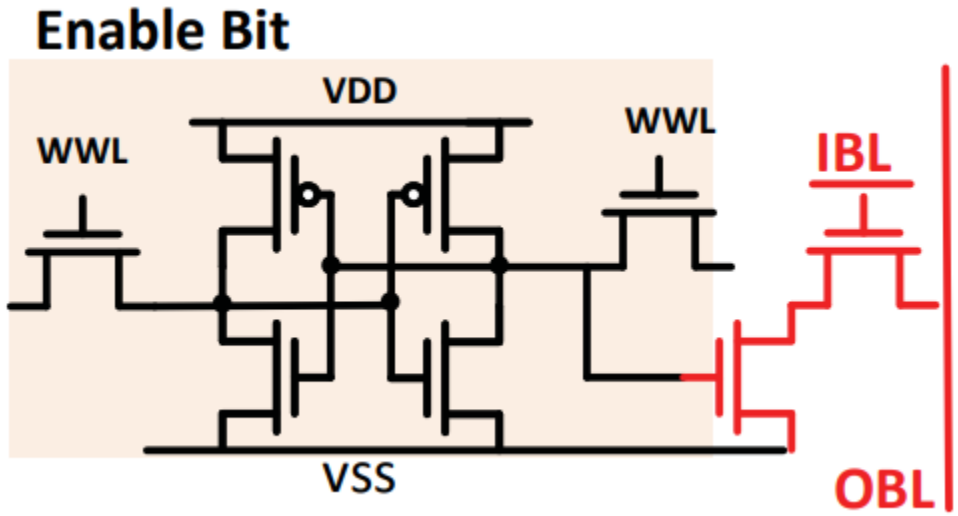
Entity resolution

Reduced Crossbar Interconnect

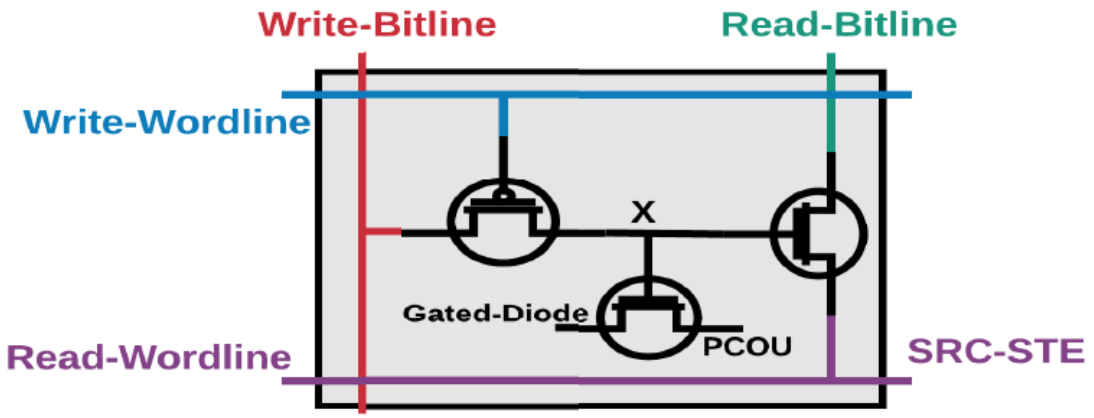


Mapping to Memory Technology

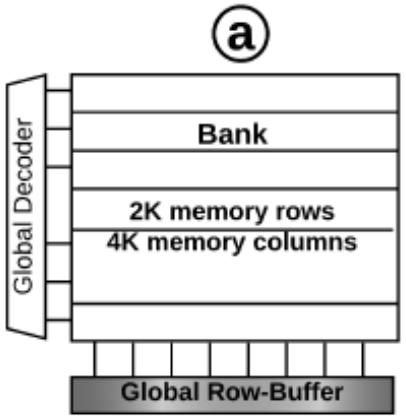
- Non-destructive read is necessary to implement OR functionality
- 2T1D cell has lower area overhead than 8T cell



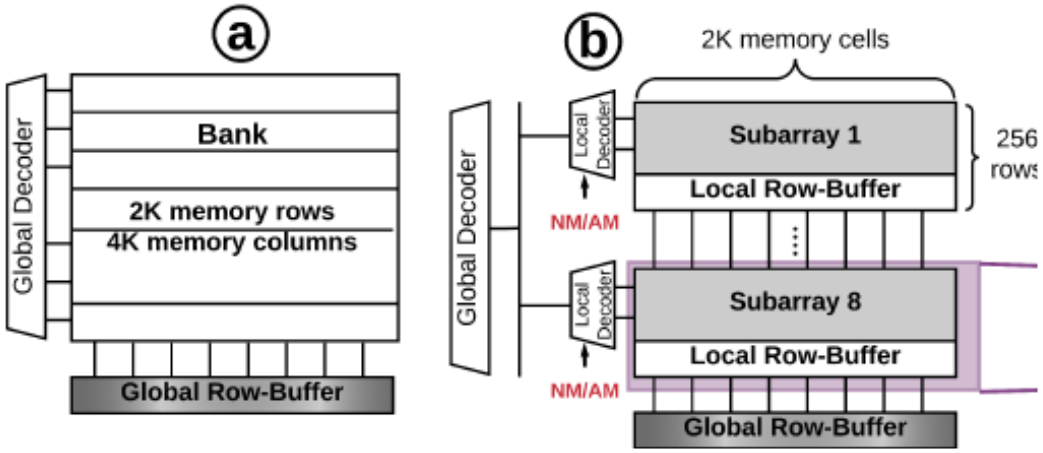
Cache Automaton use 8T SRAM cell



We propose to use 2T1D cell

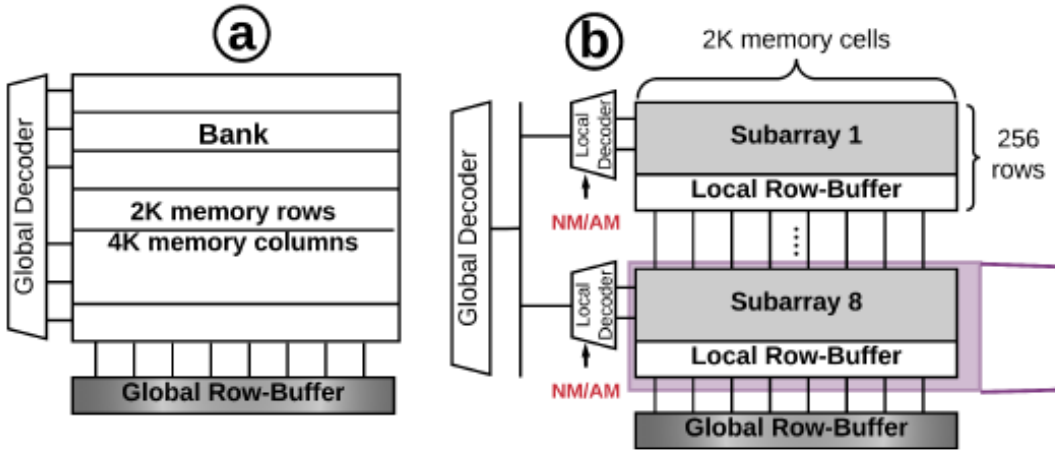


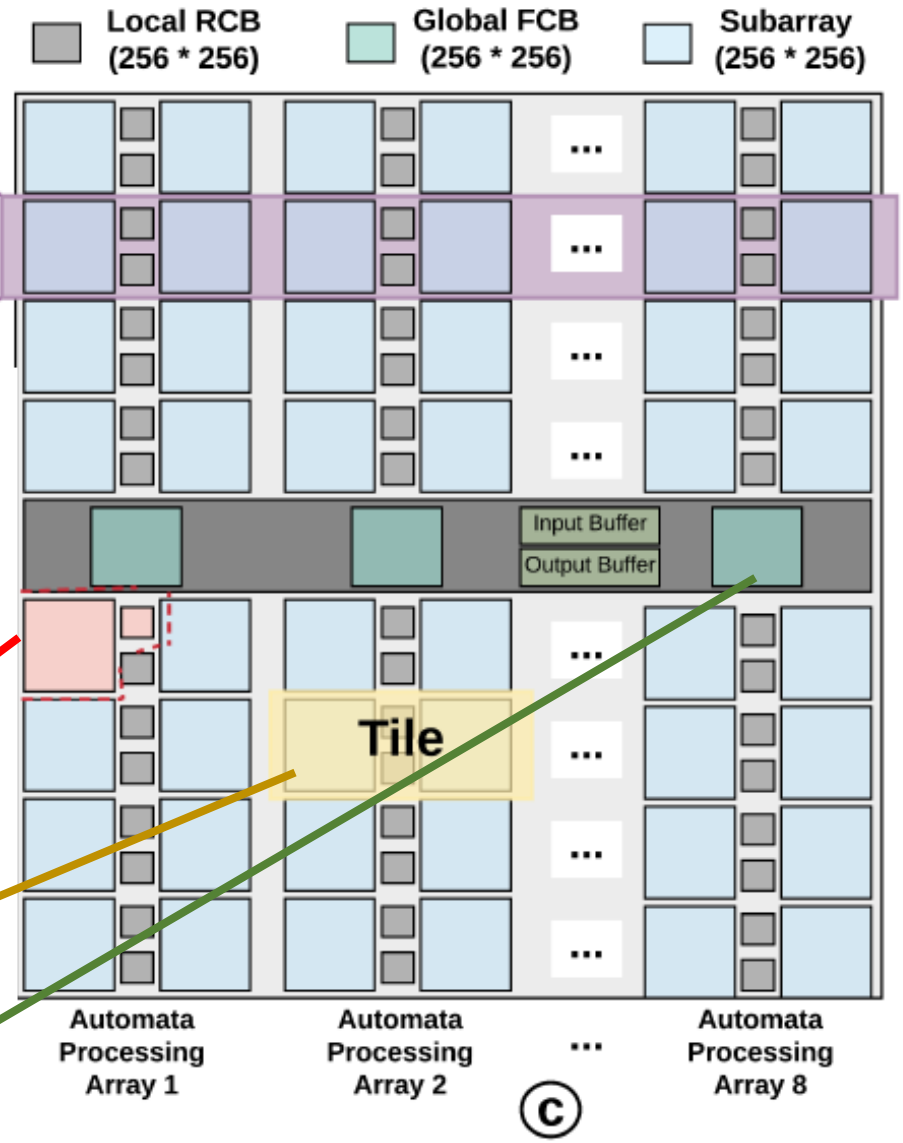
Re-purpose eDRAM bank for automata processing



Re-purpose eDRAM bank for automata processing

Utilize subarray level parallelism of a memory bank





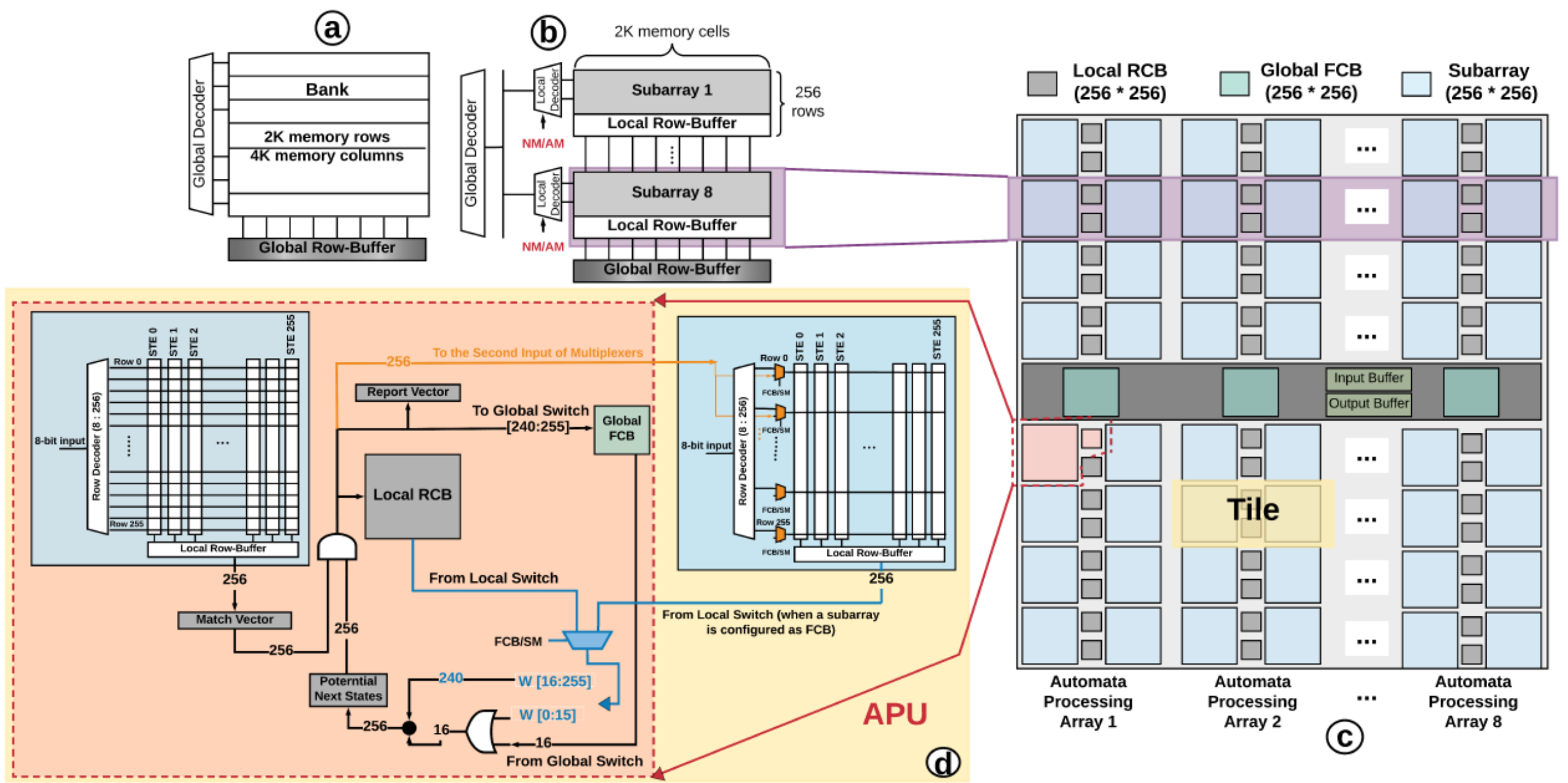
Re-purpose eDRAM bank for automata processing

Utilize subarray level parallelism of a memory bank

Process an automata with up to 256 states using local interconnect

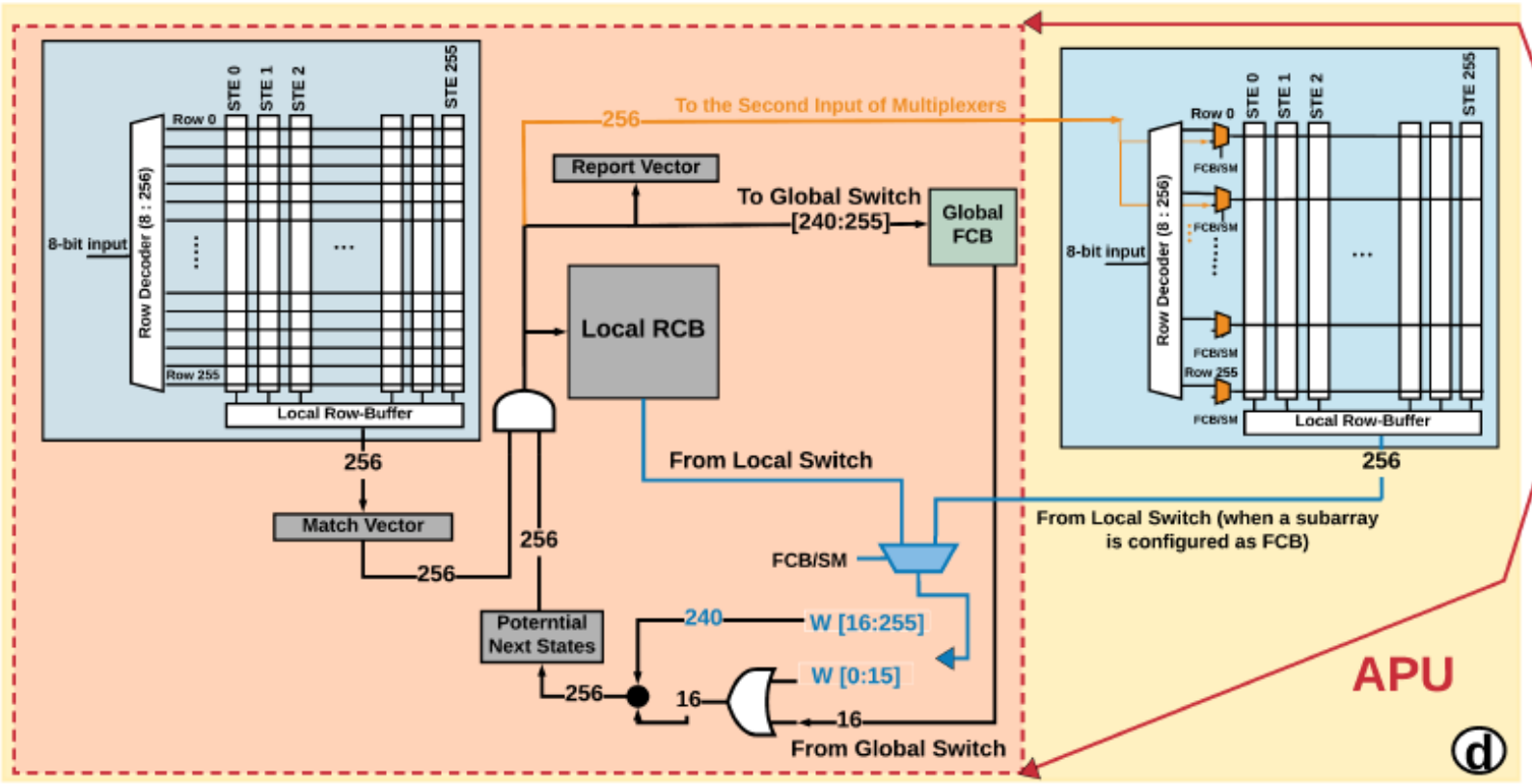
Tile process an automata with non-diagonal shape interconnect

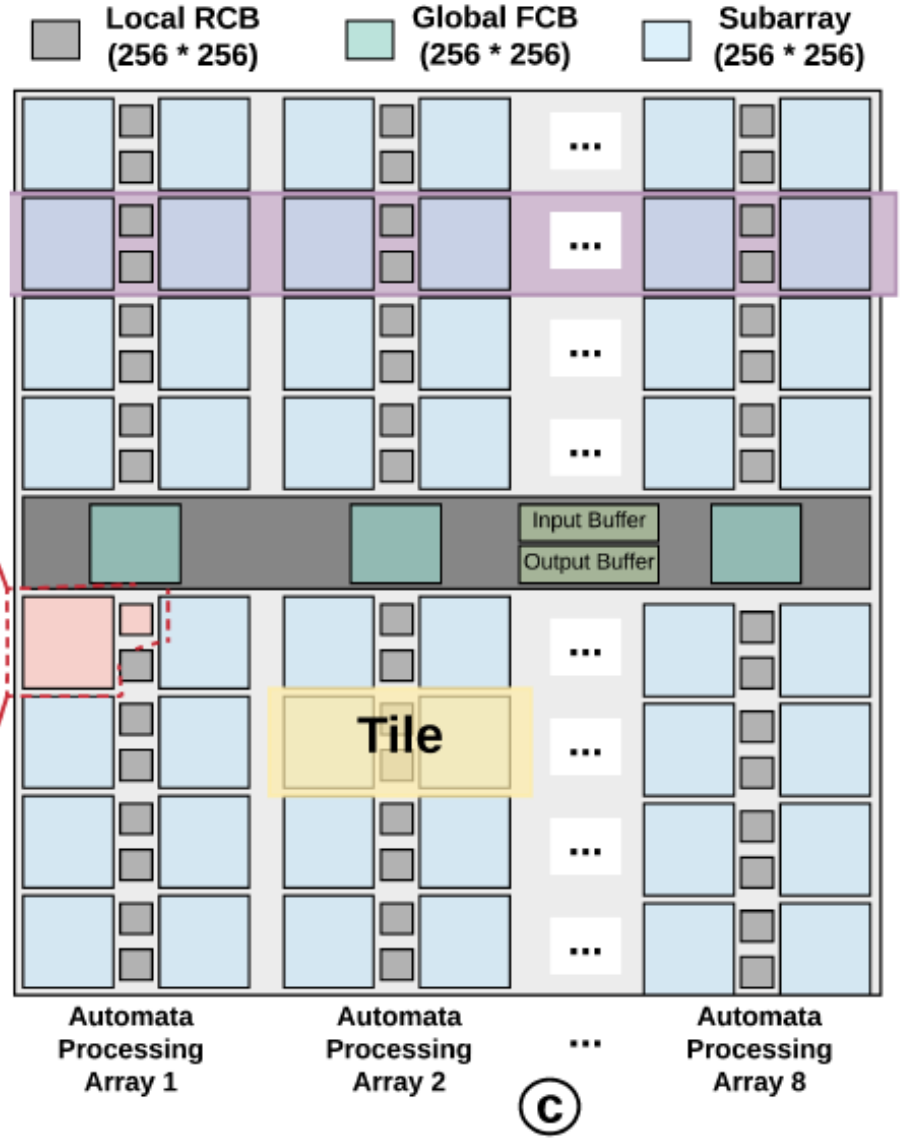
Global switches provide large automata processing


a
Global Decoder
Bank
2K memory rows
4K memory columns
Global Row-Buffer
b
2K memory cells
Global Decoder
Local Decoder
Subarray 1
Local Row-Buffer
256 rows
NM/AM
Subarray 8
Local Row-Buffer
NM/AM
Global Row-Buffer
Local RCB (256 * 256)
Global FCB (256 * 256)
Subarray (256 * 256)
Input Buffer
Output Buffer
Tile
Automata Processing Array 1
Automata Processing Array 2
Automata Processing Array 8
c
STE 0
STE 1
STE 2
STE 255
Row 0
Row 255
8-bit input
Row Decoder (8 : 256)
Local Row-Buffer
256
To the Second Input of Multiplexers
Report Vector
To Global Switch [240:255]
Global FCB
Local RCB
From Local Switch
Match Vector
256
256
256
FCB/SM
From Local Switch (when a subarray is configured as FCB)
Poterntial Next States
240
W [16:255]
W [0:15]
256
16
16
From Global Switch
APU
d

Design an efficient two stage pipeline

- Improves eAP clock frequency 2X
- Improves Cache Automaton clock frequency 1.5X





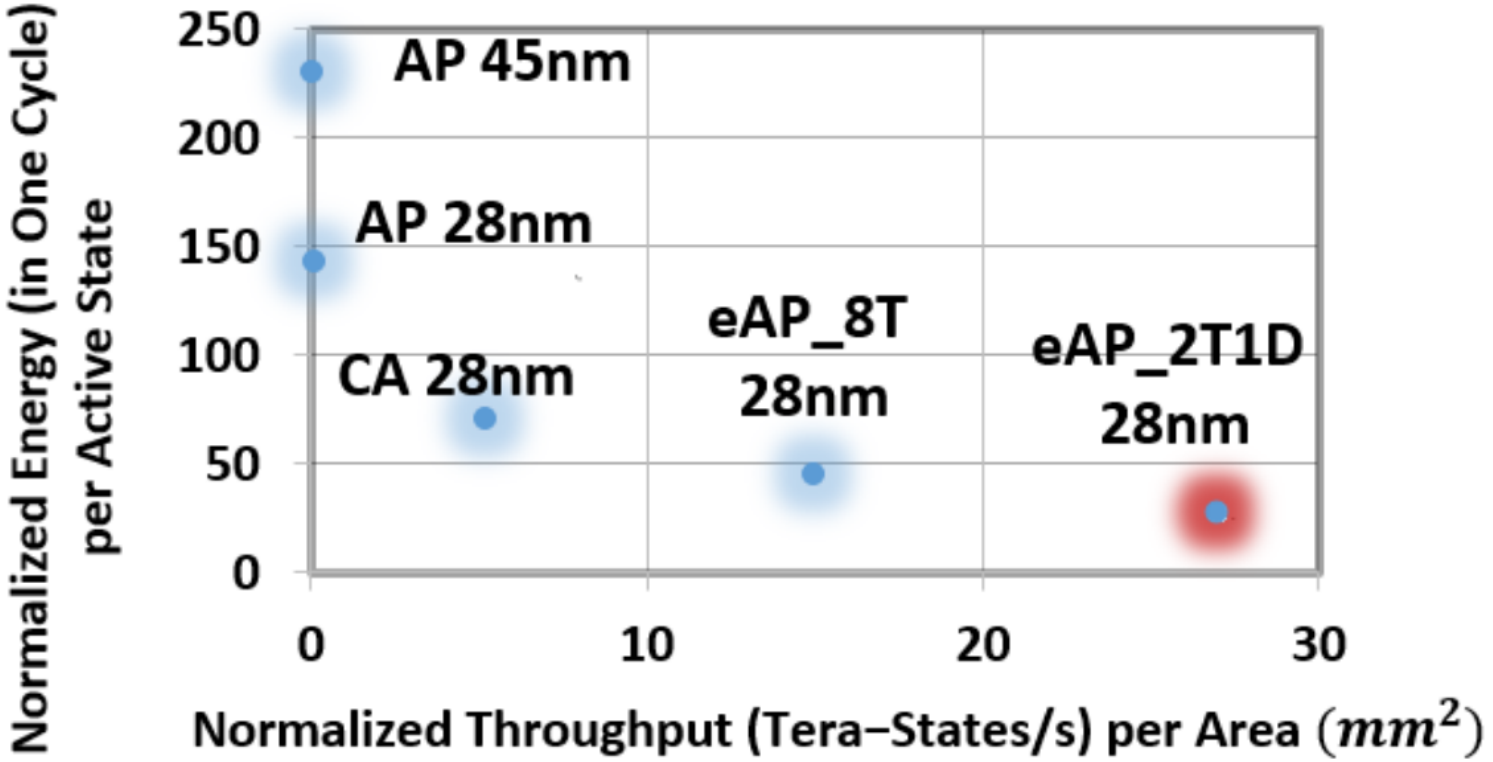
APSim (**A**utomata **P**rocessing **Sim**ulator)

- Parse automata
- Convert to homogenous representation
- Perform optimizations
- Map to hardware resources based on:
  - Connected component size
  - Interconnect shape
- Calculate activities for energy/power estimations

- **Two orders of magnitude faster** than AP-compiler

Summary of Performance Evaluation

- Incorporate both architectural contribution and technology contribution
- eAP_2D1D has **1.7X**, **3.3X** and **210X** better throughput per unit area than **eAP_8T**, **CA,** and the **AP**

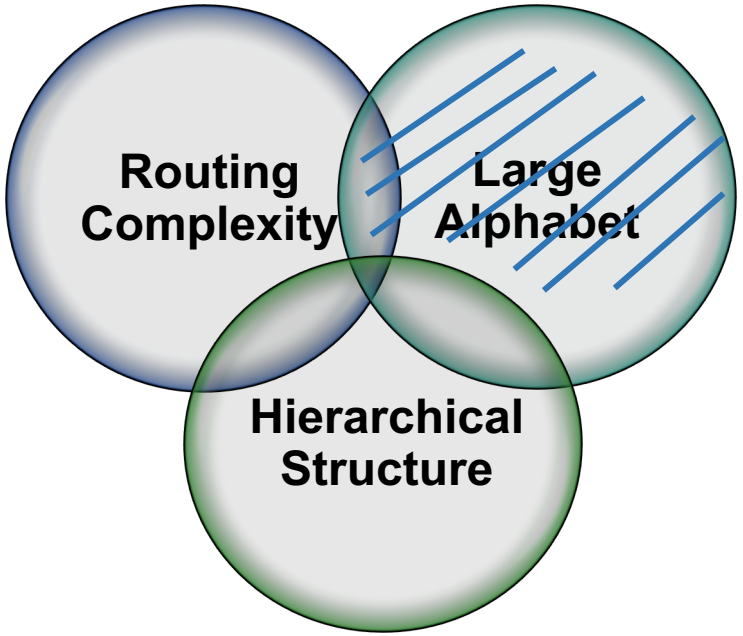


Overview of My Dissertation Work

1	• Rule-Based Part-of-Speech Tagging Using Spatial Architectures	KDD'18, IISWC'18
2	• Hierarchical Pattern Mining on the Automata Processor	ICS'17, CF'16, IJPP'17, IISWC'16
3	• A Scalable In-SRAM Architecture for Pushdown Automata	MICRO'18
4	• A Scalable In-Memory Interconnect for Automata Processing	CAL'19 MICRO'19
5	• **FlexAmata: A Flexible Automata Processing Engine**	MICRO'19

- **Novel Architecture Exploration**

FlexAmata:
A Flexible Automata Processing Engine

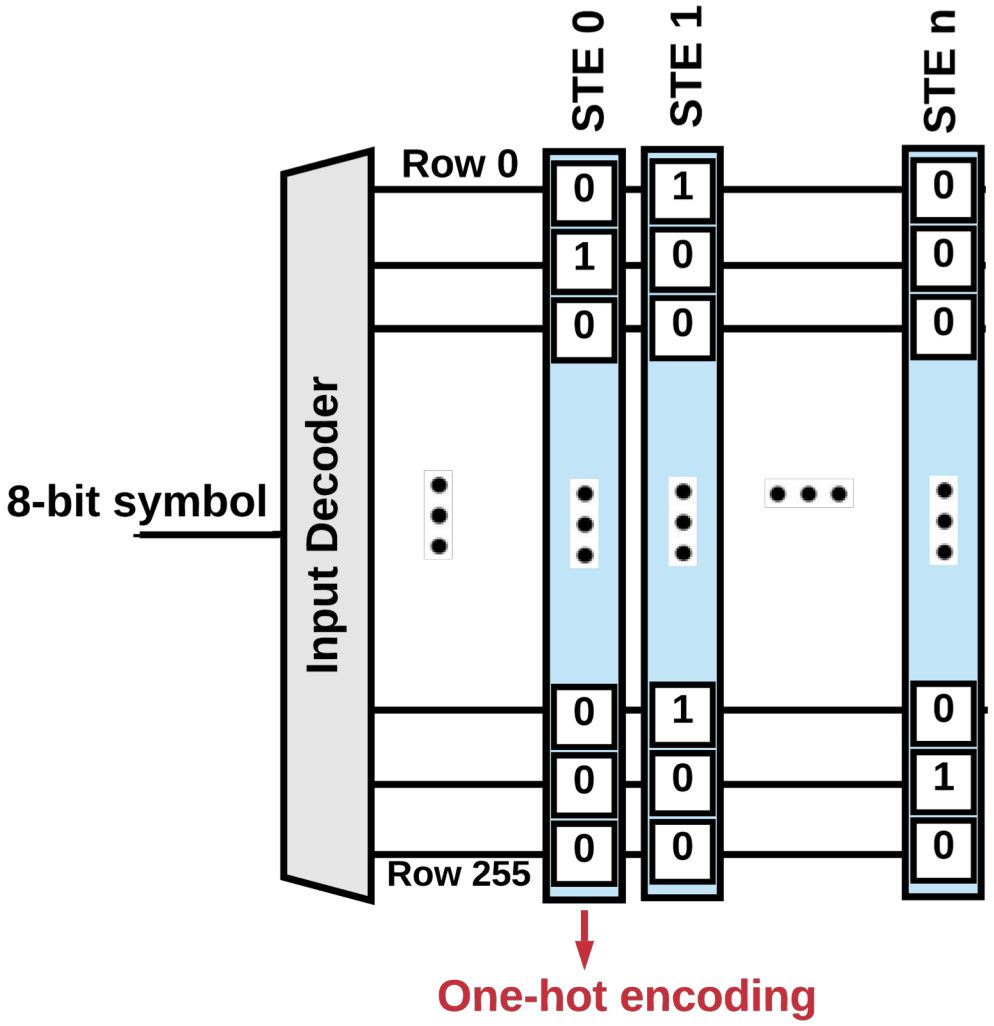


Potential problems with fixed 8-bit processing?

Problem 1: applications with **small alphabets** cannot fully utilize the existing 8-bit hardware accelerators

- Symbols are encoded with one-hot encoding

- Genomics applications
  - 2-bit processing is enough

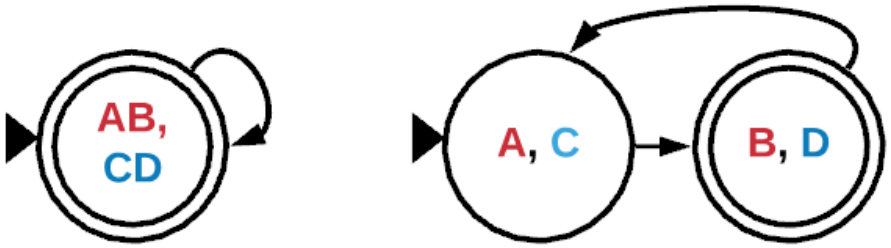


8-bit architectures **underutilize** hardware resources!

Problem 2: applications with **very large alphabets** are not able to use the existing 8-bit hardware accelerators

- What if an application has millions or billions of symbols?
- Increasing memory column size?
- Chaining states?

Amazon inventory



16-bit symbols

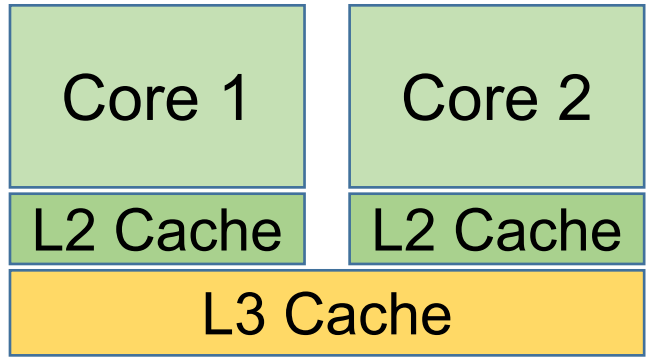
8-bit symbols

AD is a **false positive**

8-bit architectures are **not general** for applications with large alphabets!

Problem 3: application dependency to memory subarray size

- Cache Automaton [1]
  - Re-purposes a portion of L3 cache for automata processing
  - What if the subarray size of underlying memory technology changes?



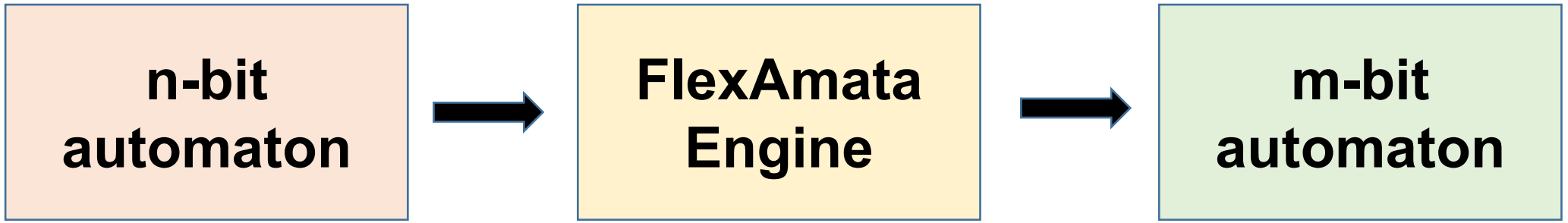
Applications may **not be compatible** with future memory technologies!

[1] Subramaniyan, Arun, et al. "Cache Automaton." *MICRO,* 2017.

Research Questions

- How to **efficiently** support applications with very large or very small alphabets on existing 8-bit architectures?
- How can an application make **better use** of existing hardware for automata acceleration?
- What is the **best bitwidth-size** for automata processing on spatial platforms?
- How to design **next-generation** automata accelerators with higher throughput?

Solution: FlexAmata

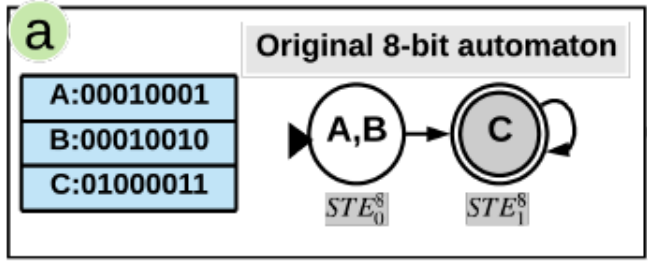


- n, m: arbitrary size
- Fine grain, bit-level optimizations

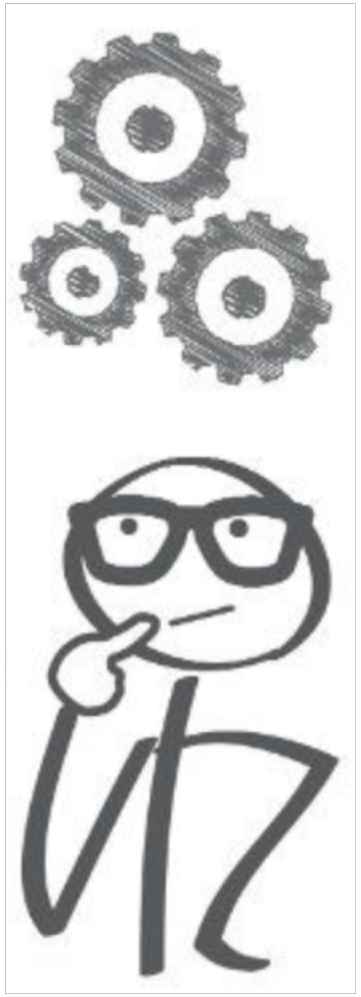
This enables

- General solution for any application on existing 8-bit architectures
- Design space exploration for various bitwidths on spatial hardware accelerators

FlexAmata: temporal transformation

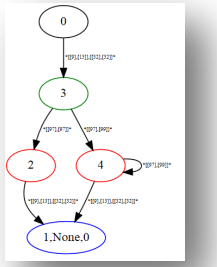


Transformation overhead?

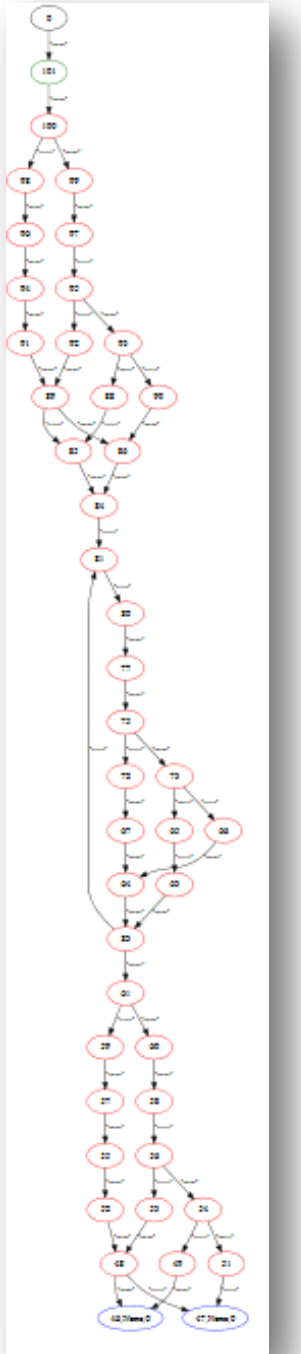


Original

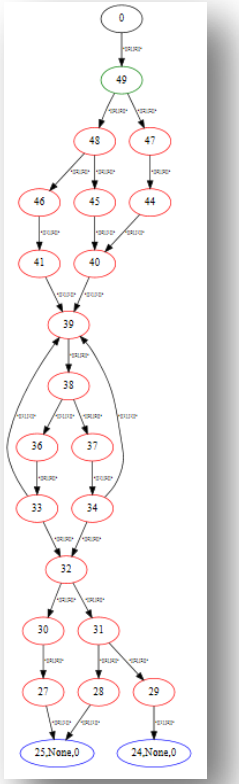
8-bit design



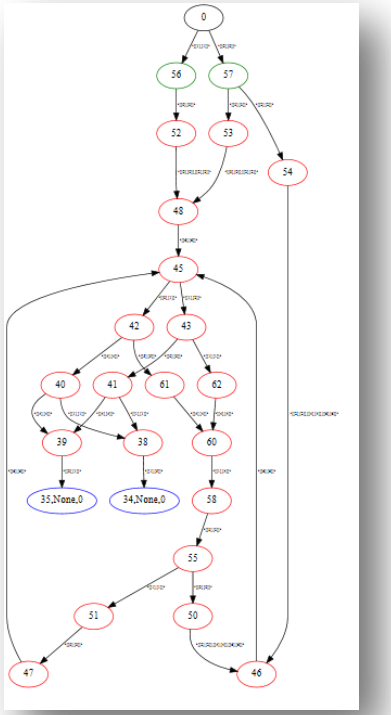
1-bit design



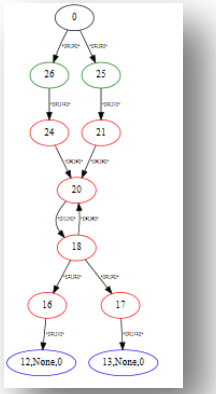
2-bit design



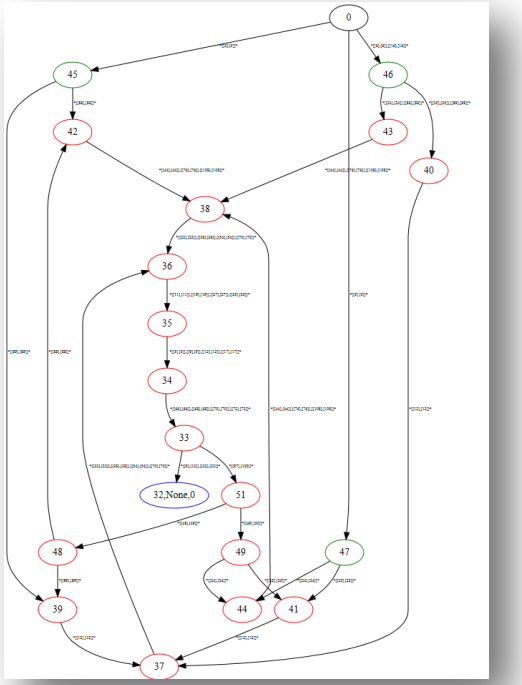
3-bit design



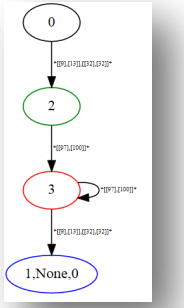
4-bit design



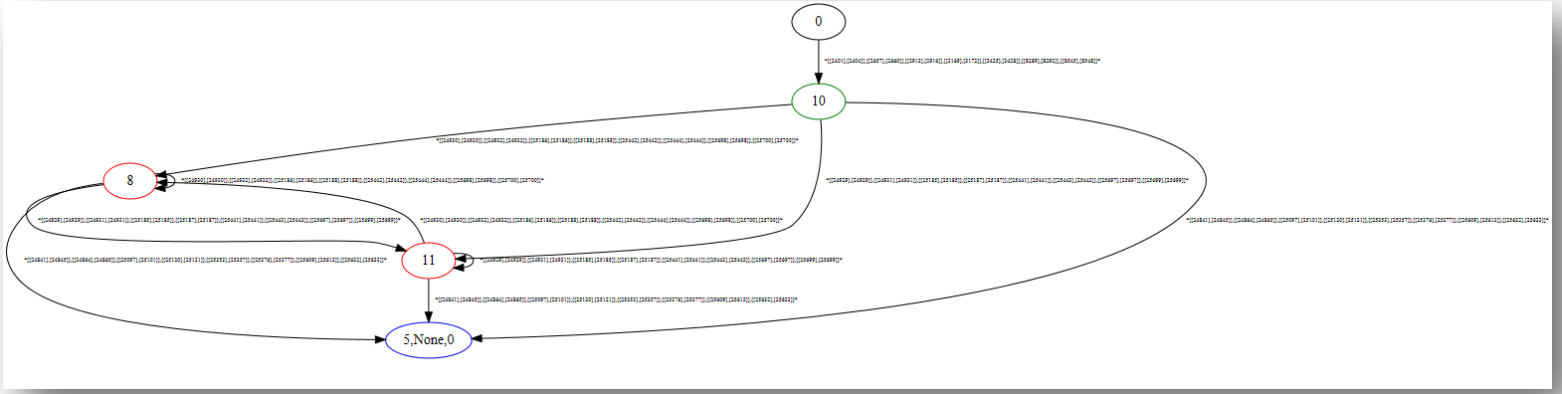
7-bit design



8-bit design

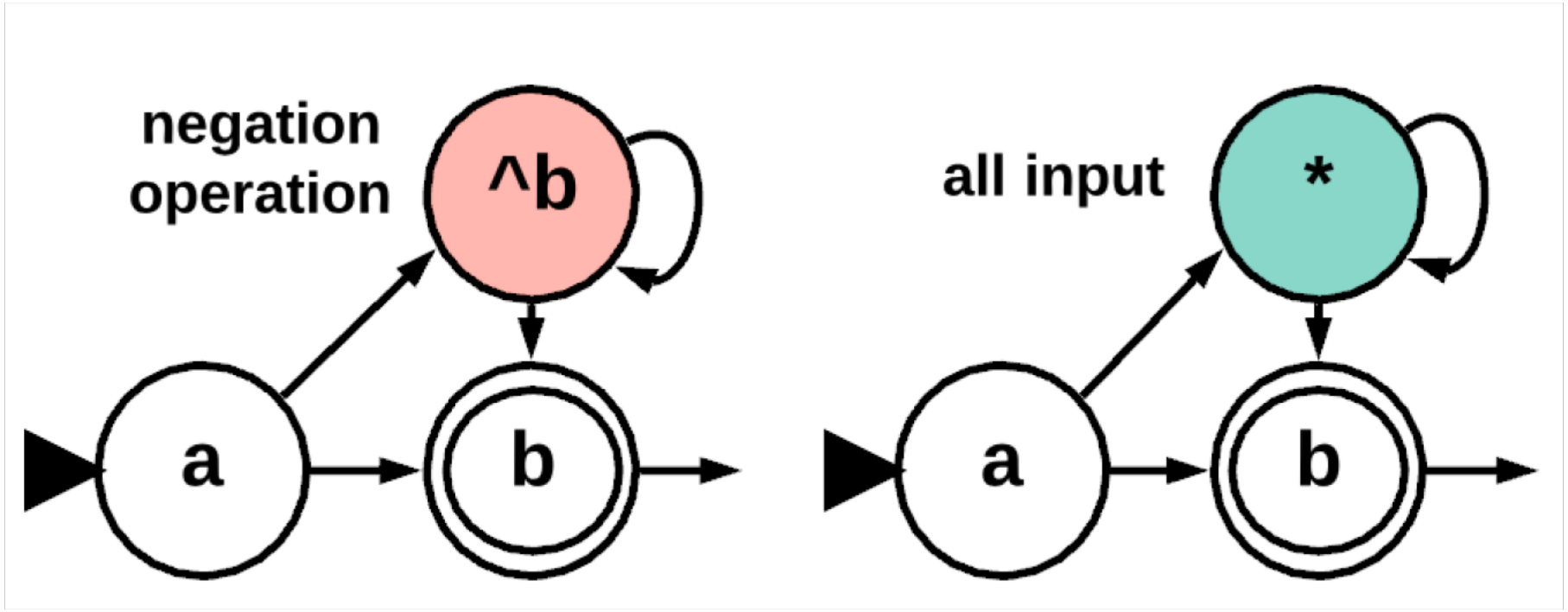


16-bit design

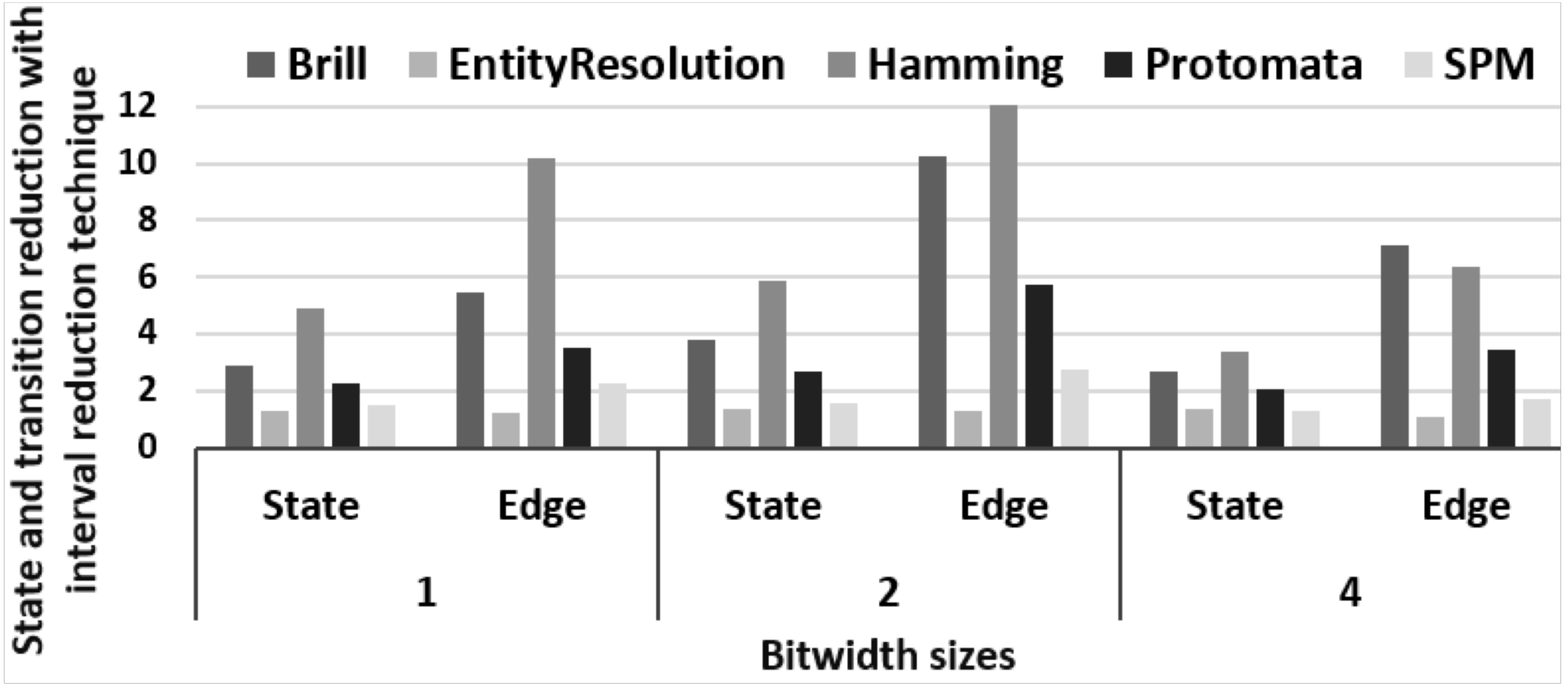


Software optimization: negation operation

- Both automata detect language a[^b]*b

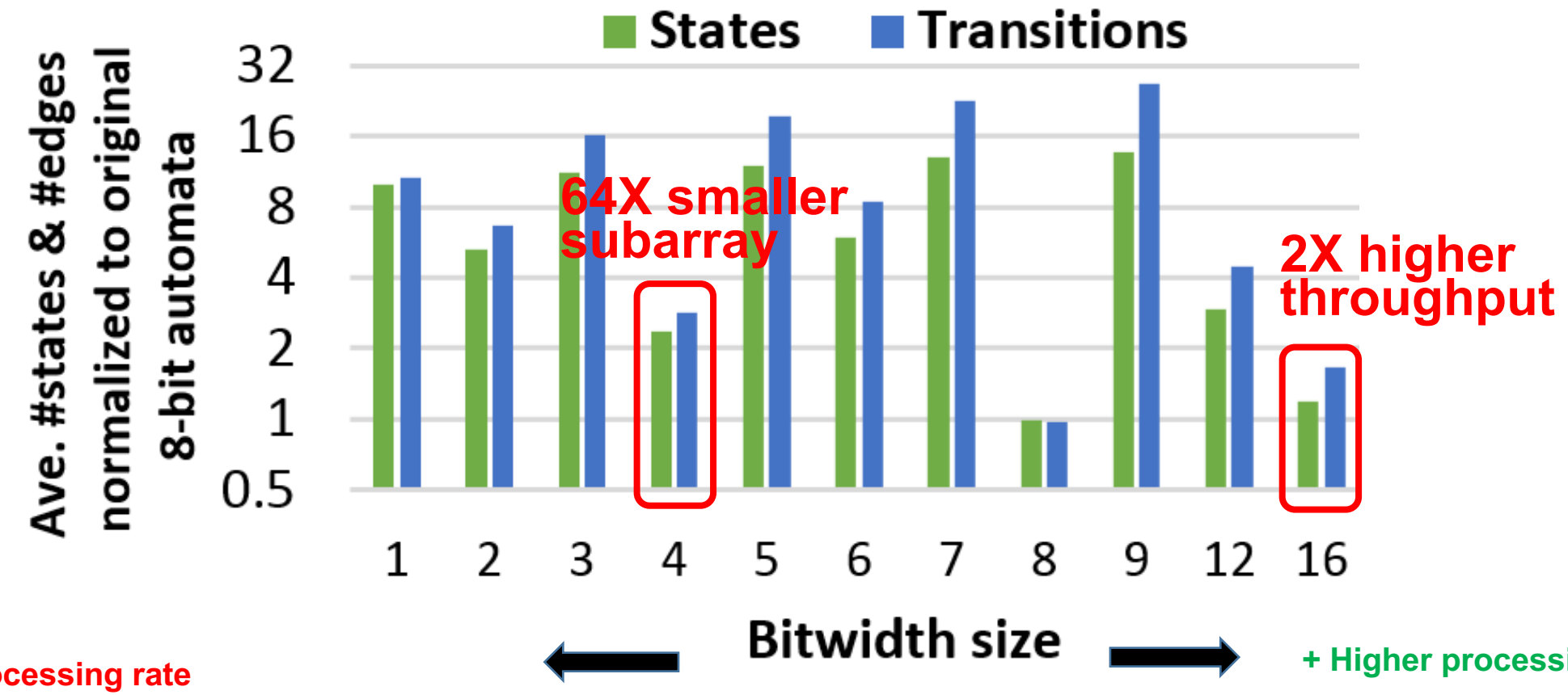


Negation Operation Removal



State and Transition Overhead

- Average on 20 automata applications from ANMLZoo and Regex benchmark suites



- Lower processing rate

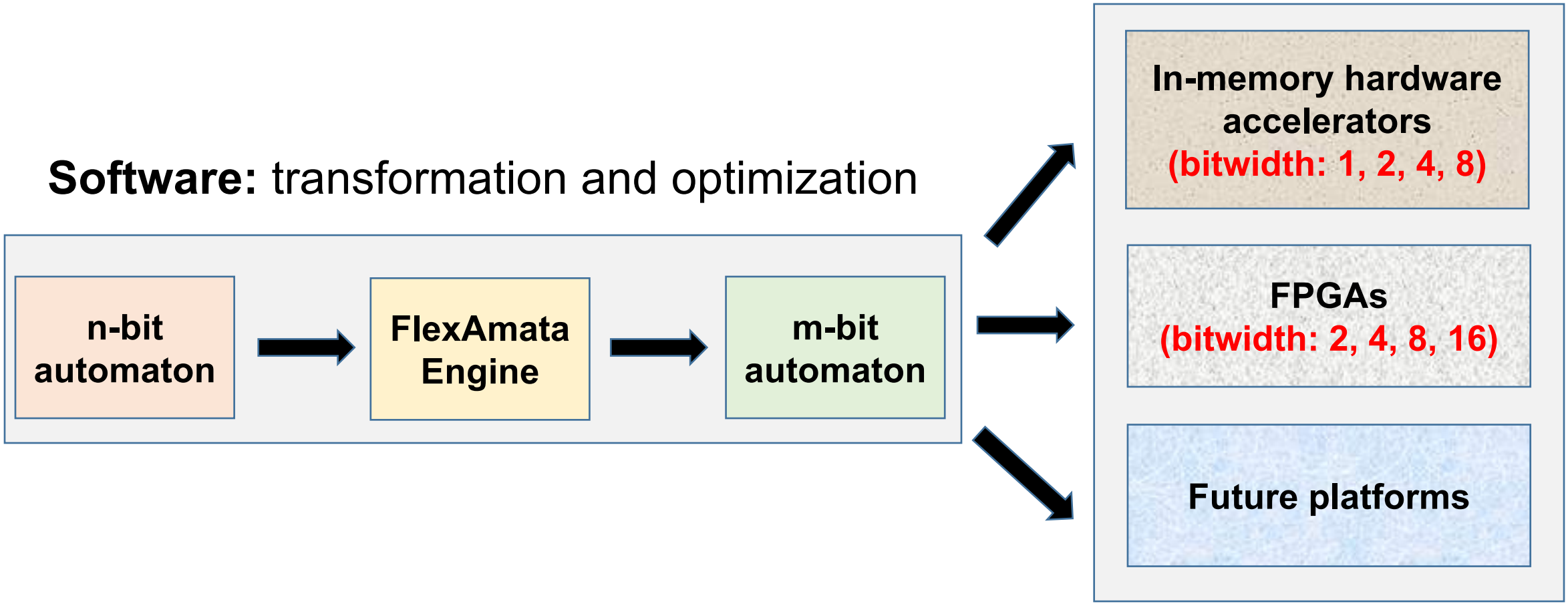
+ Smaller memory subarrays (reduced delay and power consumption)

+ Higher processing rate

- Larger memory subarrays

FlexAmata: Hardware Implications

Software: transformation and optimization



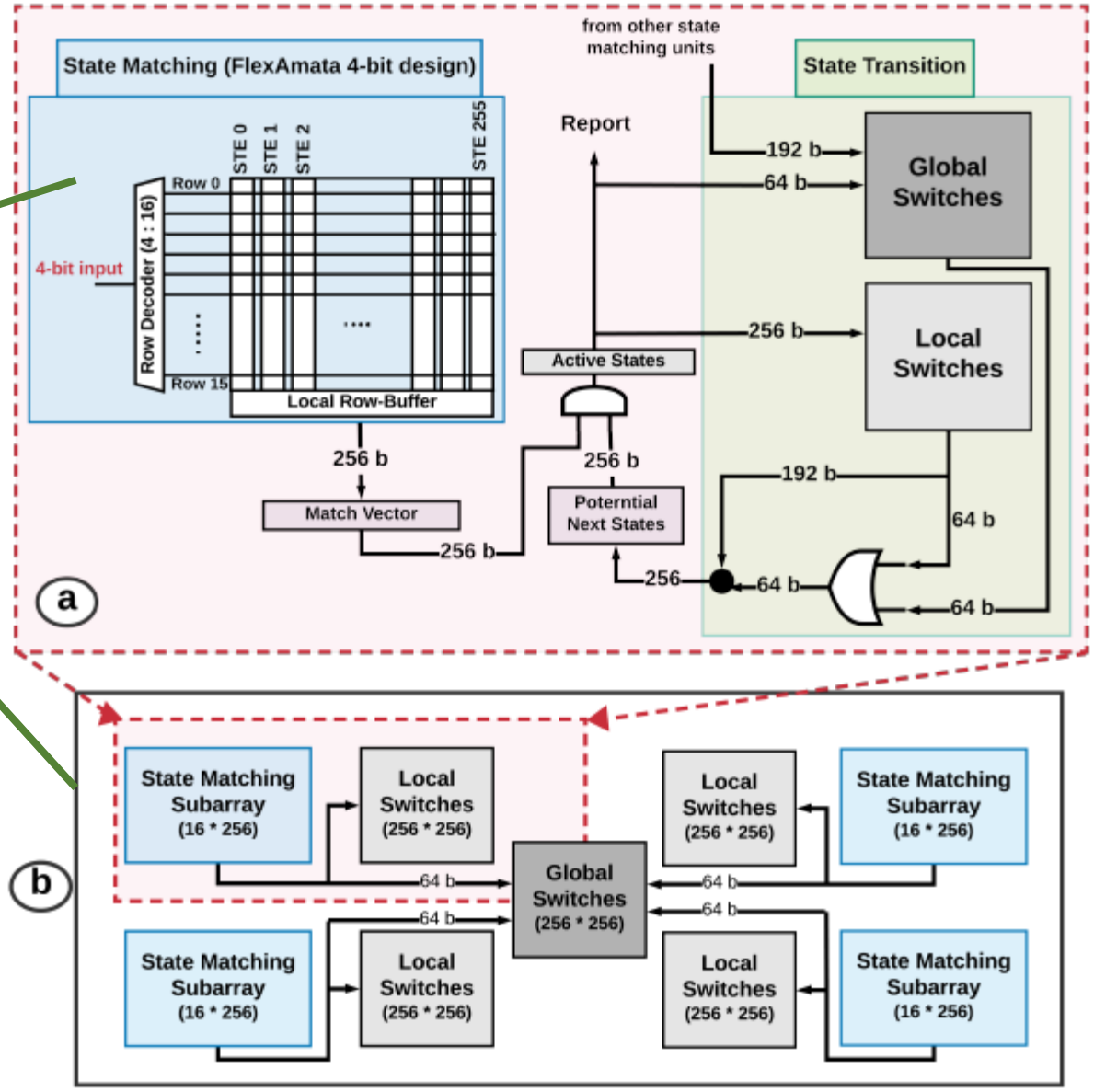
In-memory Architecture: Reduced Bitwidth Designs (RDBs)

- 4-bit design

State matching subarray (16 rows)
16X less than 8-bit design

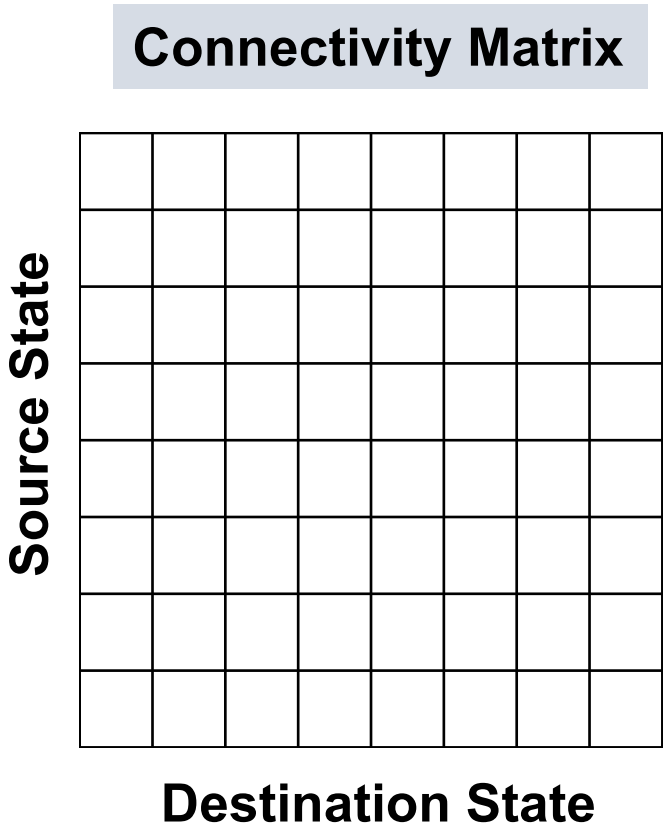
Hierarchical routing to support larger automaton
(up to 1024 states in a connected component)

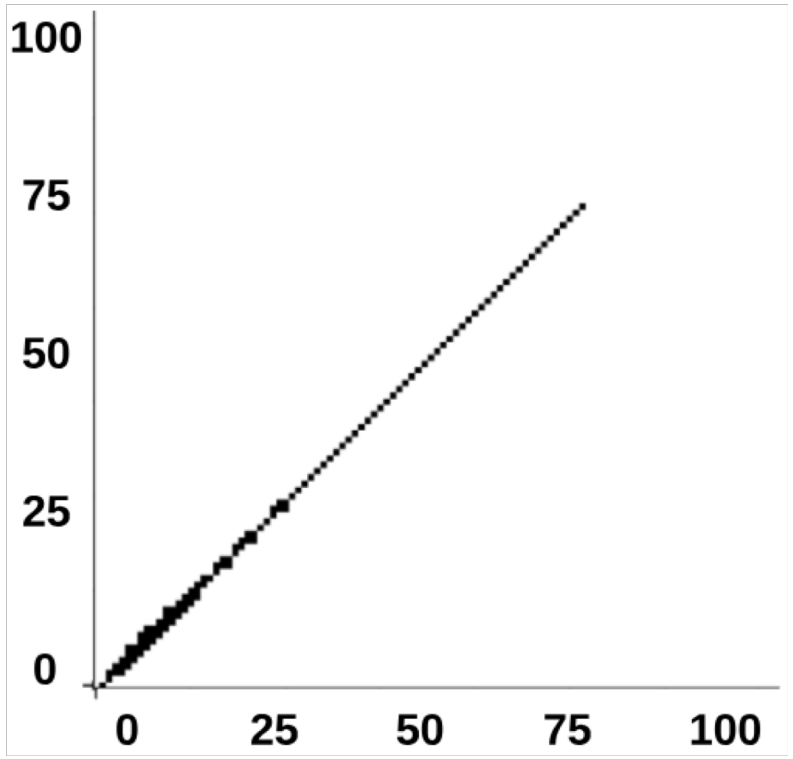
To support other sizes, only state matching subarray size changes



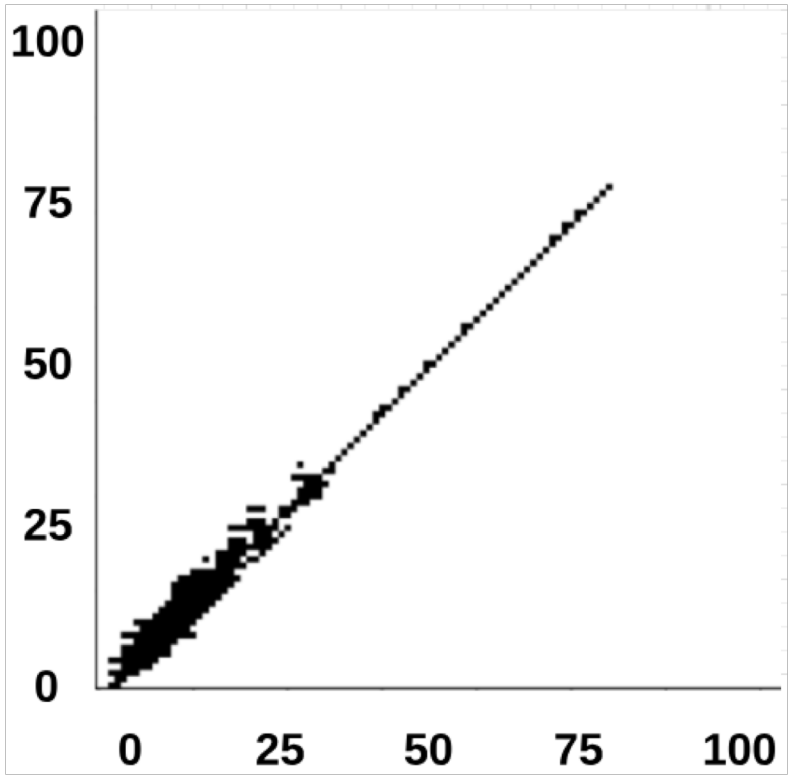
Zero/minimal interconnect overhead

- In-memory full-crossbar interconnect topology
- 4-bit design increases the interconnect utilization





8-bit design



4-bit design

Evaluation Metric: Throughput per unit area

- Both designs assume 4MB memory
- Frequency: 2.5GHz

	Number of rows	Number of subarrays	Number of States	Area (mm^2)
4-bit processing	16	8,192	128K	7.76
8-bit processing	256	512	2048K	5.06

128K states

One hardware unit
(in-memory 8-bit design)

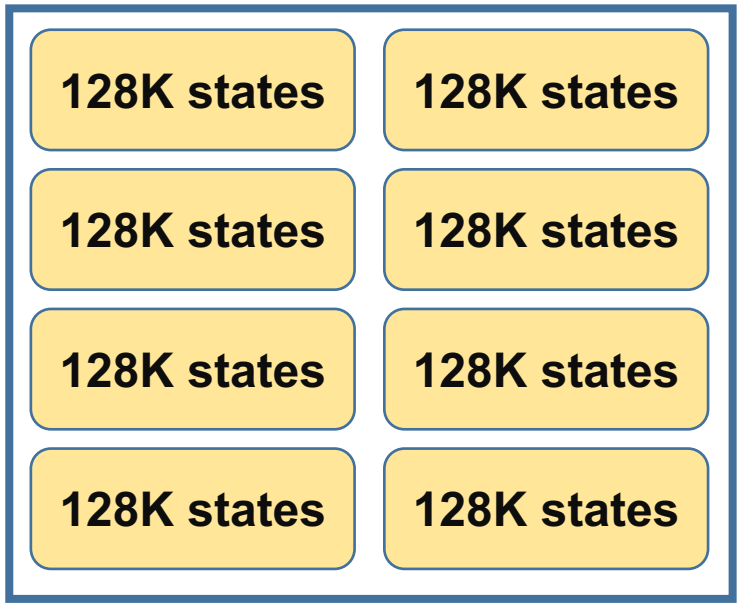
2048K states

One hardware unit
(in-memory 4-bit design)

In-memory automata processing:
4-bit design performs better than 8-bit design

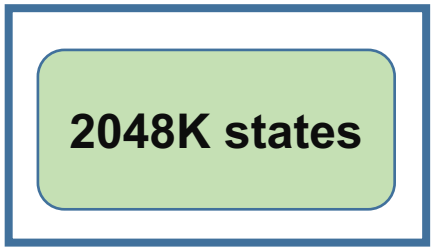
in-memory 8-bit design

1024K states



in-memory 4-bit design

2048K states



$$\frac{Throughput}{Area} = \frac{8 \times f}{8 \times 5.06} < \frac{Throughput}{Area} = \frac{4 \times f}{1 \times 7.76}$$

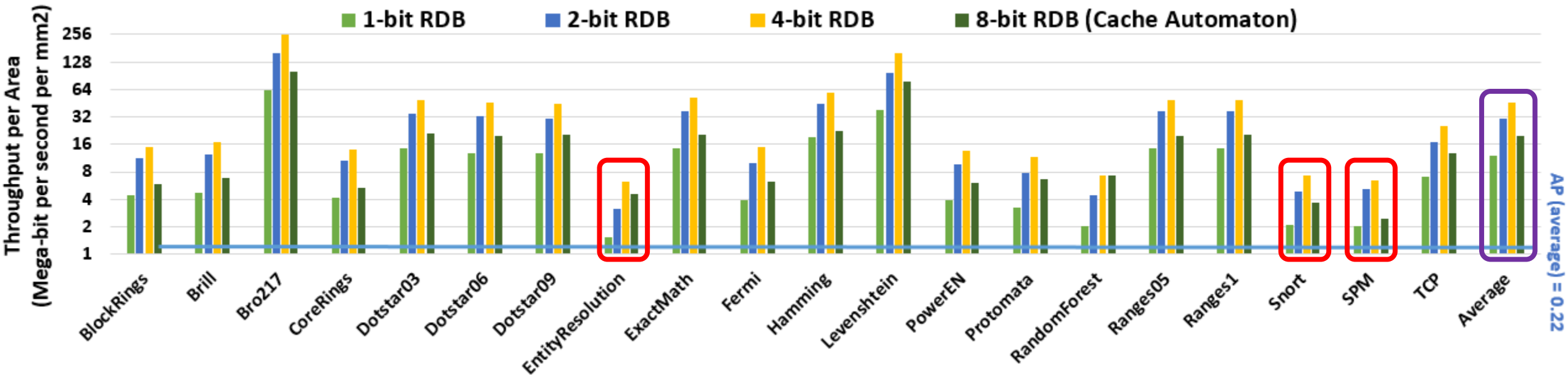
2.6X

Workload overview

Benchmark	#Family	#States	#Transitions	Ave. Node Degree	Symbol Density
Brill [24]	Regex	42658	62054	2.90	52.2
Bro217 [2]	Regex	2312	2130	1.84	1.8
Dotstar03 [2]	Regex	12144	12264	2.01	3.1
Dotstar06 [2]	Regex	12640	12939	2.04	4.8
Dotstar09 [2]	Regex	12431	12907	2.07	6.7
ExactMath [2]	Regex	12439	12144	1.95	1
PowerEN [24]	Regex	40513	40271	1.98	5.8
Protomata [24]	Regex	42009	41635	1.98	116
Ranges05 [2]	Regex	12621	12472	1.97	1.2
Ranges1 [2]	Regex	12464	12406	1.99	1.2
Snort [24]	Regex	100500	81380	1.61	7.5
TCP [2]	Regex	19704	21164	2.14	10.1
Hamming [24]	Mesh	11346	19251	3.39	113
Levenshtein [24]	Mesh	2784	9096	6.53	1
EntityResolution [24]	Widget	95136	219264	4.60	47
Fermi [24]	Widget	40783	57576	2.82	7.1
RandomForest [24]	Widget	33220	33220	2	179
SPM [24]	Widget	69029	211050	6.11	26.5
BlockRings [24]	Synthetic	44352	44352	2	1
CoreRings [24]	Synthetic	48002	48002	2	1

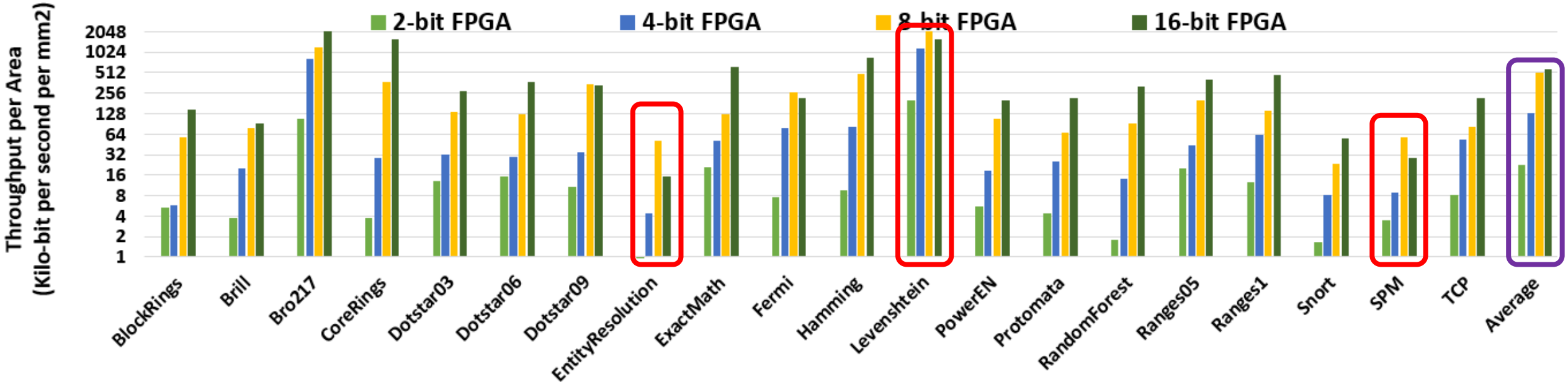
In-memory Design Results

On average, 2-bit and 4-bit designs have 1.6X and 2.3X higher throughput per area than original 8-bit design, respectively.



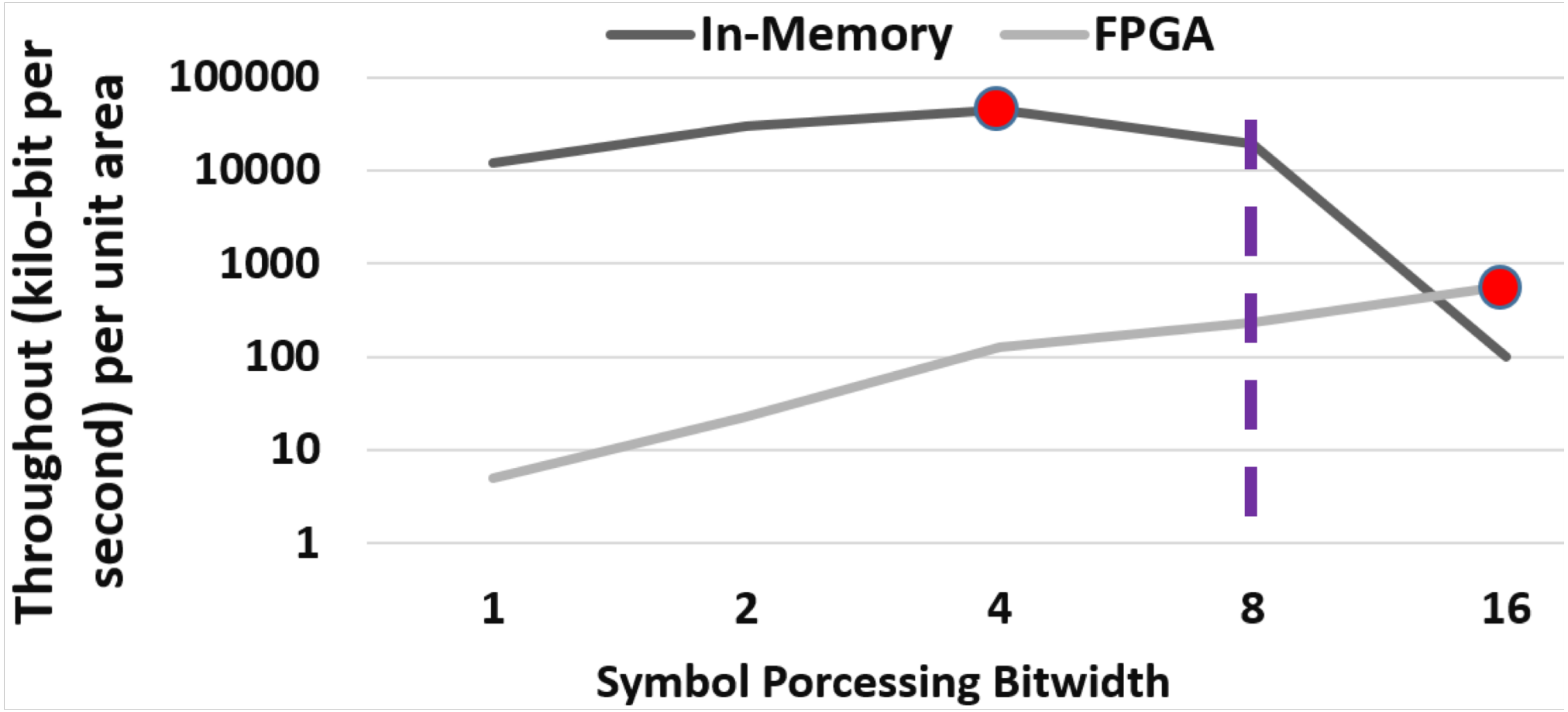
FPGA Results

- LUT based implementation
- 16-bit has 2.5X higher throughput per unit area than 8-bit design



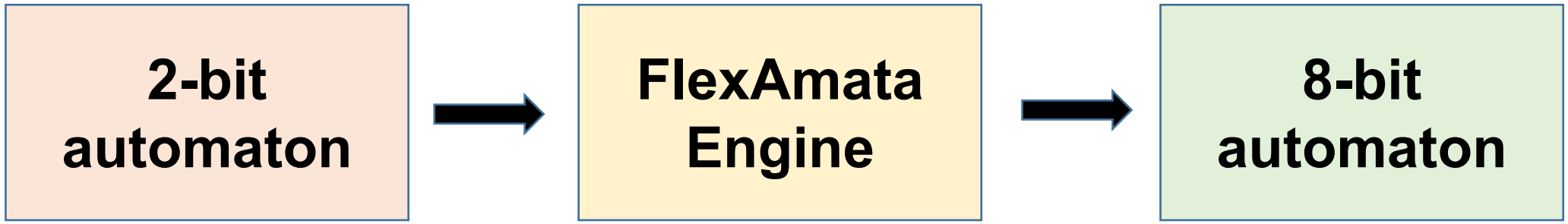
16-bit design has 1.8X more LUTs, 11% fewer FFs and 15% lower frequency but, 2X higher processing rate

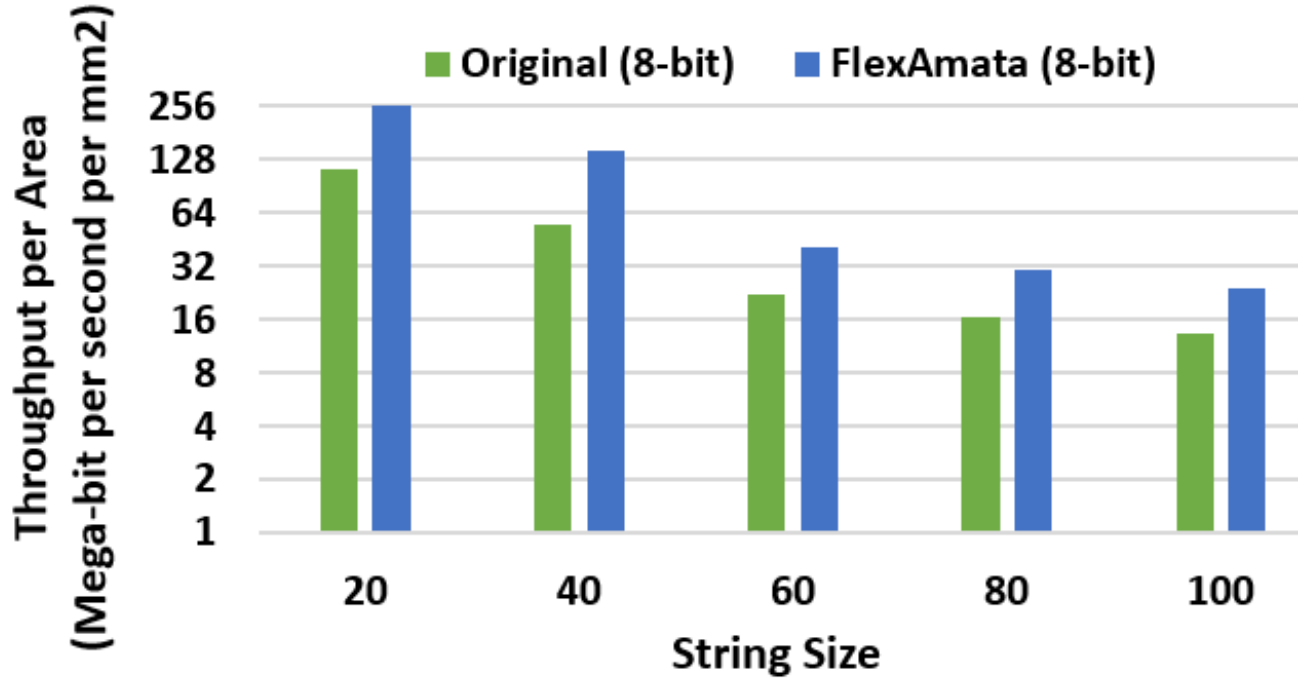
Insights for future automata processing



FlexAmata: Application Implications (1)

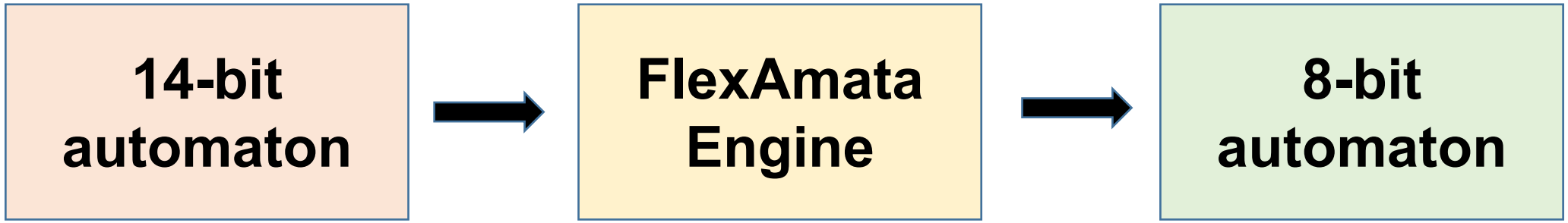
- FlexAmata enables full resource utilization for small symbol-set

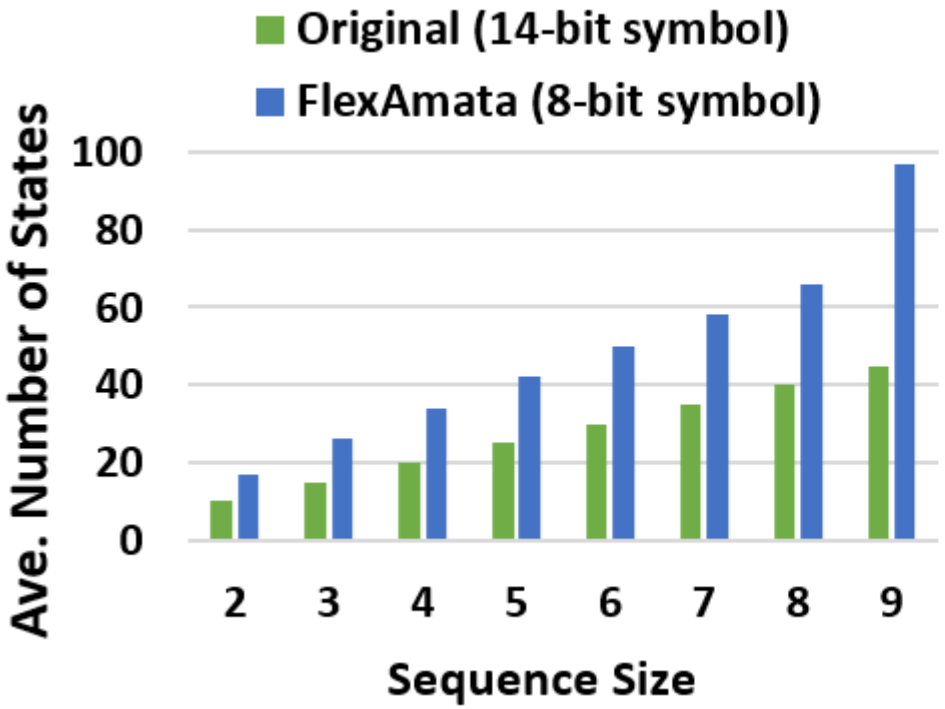


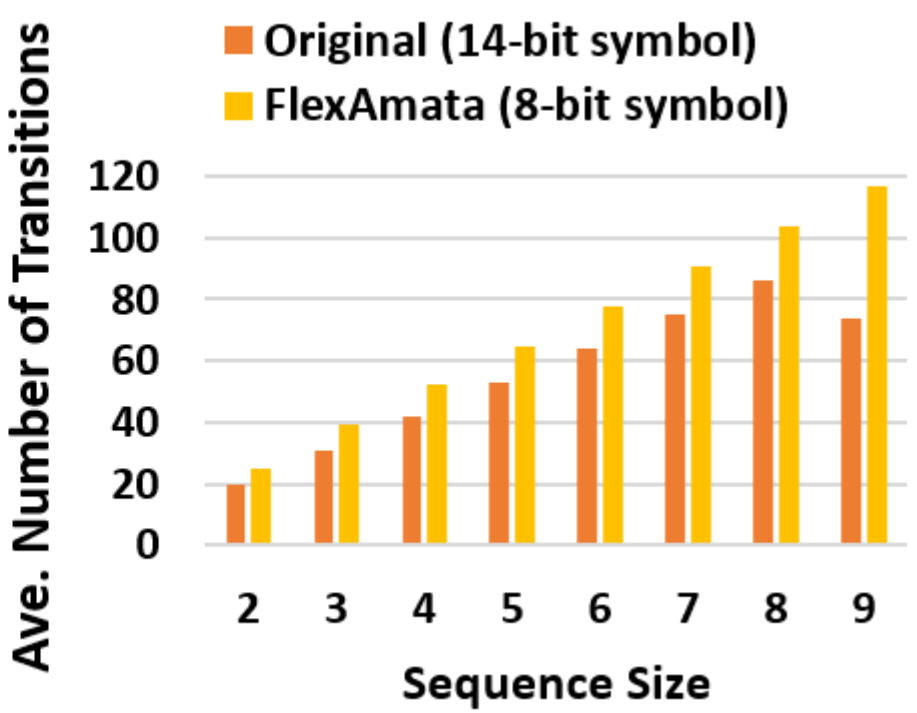


FlexAmata: Application Implications (2)

- FlexAmata enables feasibility support for large symbol-set







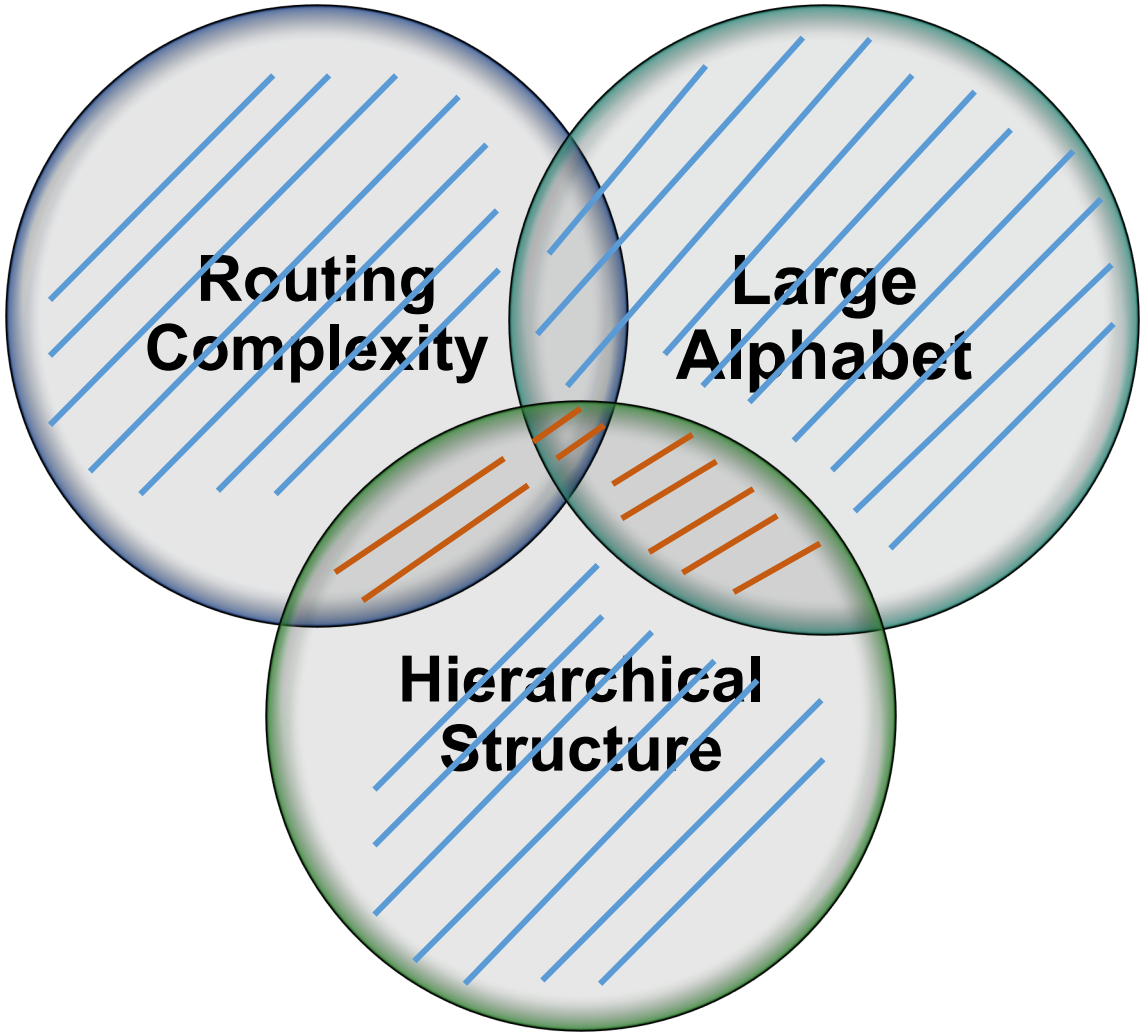
Research Questions

We hypothesize that **an efficient interconnect architecture**, **a more computationally powerful design**, **and flexible bitwidth processing** **can unleash in-memory processing benefits for more complex pattern recognition tasks.**

Overview of My Dissertation Work

1	• **Rule-Based Part-of-Speech Tagging Using Spatial Architectures**	KDD'18, IISWC'18
2	• **Hierarchical Pattern Mining on the Automata Processor**	ICS'17, CF'16, IJPP'17, IISWC'16
3	• **A Scalable In-SRAM Architecture for Pushdown Automata**	MICRO'18
4	• **A Scalable In-Memory Interconnect for Automata Processing**	CAL'19 MICRO'19
5	• **FlexAmata: A Flexible Automata Processing Engine**	MICRO'19

Future Directions

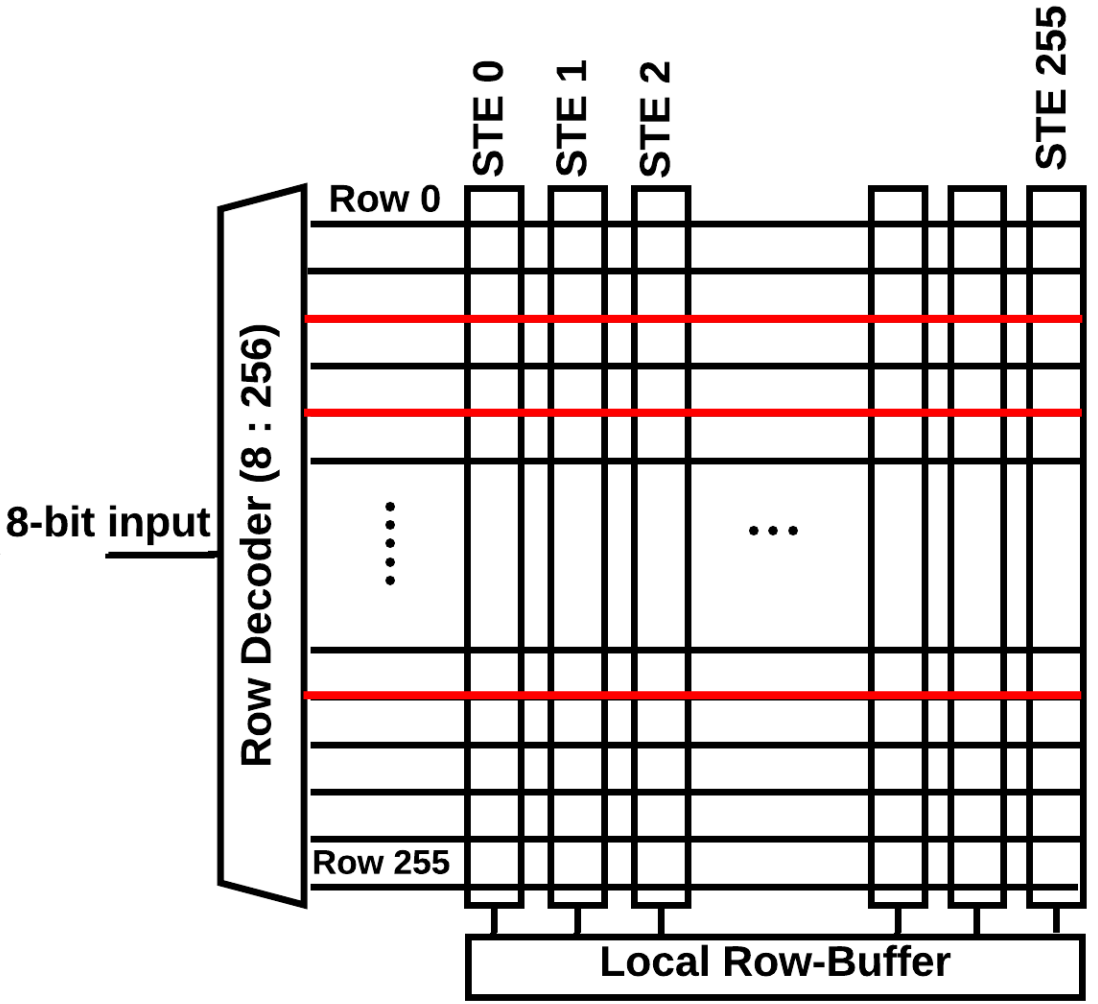


Proposed and Evaluated

Future work

Future Directions: Encoding with Multi-Row Activation

- Commodity DRAM
  - One-row activation
- SRAM or gain-cells memory array
  - Multi-row activation
- This provides the possibility to process more than one symbol per cycle by
  - Encoding the automata
  - Encoding the input



Future Directions

- Mapping graph processing applications to our in-memory DPDA architecture
- Reporting architecture
- BRAM-based FPGA for different bitwidths

Broader Implications

- Network security
  - Increase in network line rate
  - Cloud adoption
- System reliability
  - Execution behavior is expressed as automata
- Social media
  - Rumor debunking
  - Online language translation

Full List of Publications

1. **Elaheh Sadredini,** Reza Rahimi, Marzieh Lenjani, Mircea Stan, and Kevin Skadron. "FlexAmata: A Flexible Automata ProcessingEngine."51th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'51), under review.
2. **Elaheh Sadredini,** Reza Rahimi, Vaibhav Verma, Mircea Stan, and Kevin Skadron. "eAP: A Scalable and Efficient in MemoryAccelerator for Automata Processing." 51th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'51), under review.
3. **Elaheh Sadredini**, Reza Rahimi, Vaibhav Verma, Mircea Stan, and Kevin Skadron. "A Scalable and Efficient in-Memory InterconnectArchitecture for Automata Processing." IEEE Computer Architecture Letters, 2019. DOI: 10.1109/LCA.2019.2909870.
4. **Elaheh Sadredini**, Deyaun Gue, Chunkun Bo, Reza Rahimi, Kevin Skadron, and Hongning Wang. " A Scalable Solution for Rule-Based Part-of-Speech Tagging on Novel Hardware Accelerators." ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'18), 2018.
5. Kevin Angstadt, Arun Subramaniyan, **Elaheh Sadredini**, Reza Rahimi, Kevin Skadron, Westley Weimer, and Reetu Das. "ASPEN: A Scalable In-SRAM Architecture for Pushdown Automata."51th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'51), 2018**.**
6. **Elaheh Sadredini**, Reza Rahimi, ke Wang, Kevin Skadron. "Frequent Subtree Mining on the Automata Processor: Challenges and Opportunities." *In Proceedings of the International Conference on Supercomputing (ICS),* 2017
7. Ke Wang, **Elaheh Sadredini**, Kevin Skadron. "Sequential pattern mining with the Micron Automata Processor." In *Proceedings of the ACM International Conference on Computing Frontiers (CF)*. ACM, 2016.
8. Ke Wang, **Elaheh Sadredini**, Kevin Skadron. "Hierarchical Pattern Mining with the Micron Automata Processor. "International Journal of Parallel Programming (IJPP), Jan. 2017, DOI: 10.1007/s10766-017-0489-y.
9. Jack Wadden, Tommy Tracy II, **Elaheh Sadredini**, Lingxi Wu, Chunkun Bo, Jesse Du, Yizhou Wei, Matthew Wallace, Jeffrey Udall, Mircea Stan, Kevin Skadron. "AutomataZoo: A Modern Automata Processing Benchmark Suite. "IEEE International Symposium on Workload Characterization (IISWC), 2018.
10. J. Wadden, V. Dang, N. Brunelle, T. Tracy II, D. Guo, **E. Sadredini**, K. Wang, C. Bo, G. Robins, M. Stan, K. Skadron. "ANMLZoo: A Benchmark Suite for Exploring Bottlenecks in Automata Processing Engines and Architectures. "IEEE International Symposium on Workload Characterization (IISWC), October 2016.
11. Chunkun Bo, Vinh Dang, **Elaheh Sadredini**, kevin Skadron. "Searching for Potential gRNA Off-Target Sites for CRISPR/Cas9 usingAutomata Processing across Different Platforms." IEEE International Symposium on High-Performance Computer Architecture (HPCA),2018.
12. Ke Wang, Kevin Angstadt, Chunkun Bo, Nathan Brunelle, **Elaheh Sadredini**, Tommy Tracy II, Jack Wadden, Mircea Stan, Kevin Skadron. "An Overview of Micron's Automata Processor. "Proceeding of the Eleventh IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, 2016.

Patent Applications

1. **Elaheh Sadredini,** Reza Rahimi, Ke Wang, and Kevin Skadron. "Methods, Circuits, and Articles of Manufacture for Frequent Sub-Tree Mining using Non-Deterministic Finite State Machines "U.S. Patent Application No. 16/246,641.
2. **Elaheh Sadredini**, Reza Rahimi, Mircea Stan, and Kevin Skadron. "Methods, Circuits, Systems, and Manufacture for State Machine Interconnect Architecture Using Embedded DRAM." U.S. Patent Application No. 16/246,742.
3. Ke Wang, **Elaheh Sadredini**, and Kevin Skadron. ”Sequential Pattern Mining with the Micron Automata Processor.” U.S. Patent Application Serial No. 15/198521.
4. Ke Wang, **Elaheh Sadredini**, and Kevin Skadron. ”Disjunctive Rule Mining with Finite Automaton Hardware.” U.S. Patent Application, publication number: 20/180285424.
5. Chunkun Bo**, Elaheh Sadredini**, Vinh Dang, and Kevin Skadron. "Methods, Circuits, Systems, and Articles of Manufacture for Searching a Reference Sequence for a Target Sequence within a Specified Distance."U.S. Patent Application No. 15/932,287.

References

1. Yang, Yi-Hua, and Viktor Prasanna. "High-performance and compact architecture for regular expression matching on FPGA." IEEE Transactions on Computers 61.7 (2012): 1013-1025.
2. Tracy II, Tommy, et al. "Nondeterministic Finite Automata in Hardware-the Case of the Levenshtein Automaton.“ In Proceedings of the 2015 Workshop on Applications and Systems for Big Data (ASBD) in conjunction with ISCA 2015.
3. Roy, Indranil, et al. "Finding Motifs in Biological Sequences Using the Micron Automata Processor," *2014 IEEE 28th International Parallel and Distributed Processing Symposium*, Phoenix, AZ, 2014, pp. 415-424.
4. Wang, Ke, et al. "Association rule mining with the micron automata processor.“ In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2015.
5. Zhou, Keira, et al. "Brill tagging using the micron automata processor." (2014).
6. Bo, Chunkun, et al. "Entity Resolution Acceleration using Automata Processor." In *Proceedings of the 2015 Workshop on Applications and Systems for Big Data (ASBD) in conjunction with ISCA. 2015*.
7. Tracy II, Tommy, et al. "Towards machine learning on the Automata Processor." *International Conference on High Performance Computing*. Springer International Publishing, 2016.
8. Vaibhav Gogte, Aasheesh Kolli, Michael J Cafarella, Loris D’Antoni, and Thomas F Wenisch. Hare: Hardwareaccelerator for regular expressions. In Microarchitecture (MICRO), 2016 49th Annual IEEE/ACM International Symposium on, pages 1–12. IEEE, 2016.
9. uanwei Fang, Tung T Hoang, Michela Becchi, and Andrew A Chien. Fast support for unstructured data process-ing: the unified automata processor. InMicroarchitecture (MICRO), 2015 48th Annual IEEE/ACM InternationalSymposium on, pages 533–545. IEEE, 2015.
10. https://github.com/vqd8a/iNFAnt2
11. https://github.com/vqd8a/DFAGE
12. https://github.com/intel/hyperscan
13. https://github.com/jackwadden/VASim
14. https://www.pcre.org/
15. Dlugosch, Paul, et al. "An efficient and scalable semiconductor architecture for parallel automata processing." *IEEE Transactions on Parallel and Distributed Systems* 25.12 (2014): 3088-3098.
16. Subramaniyan, Arun, et al. "Cache automaton." *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 2017.
17. Xie, Ted, et al. "REAPR: Reconfigurable engine for automata processing." *Field Programmable Logic and Applications (FPL), 2017 27th International Conference on*. IEEE, 2017.
18. Stephens, Z. D., et al. "Big Data: Astronomical or Genomical." *PLoS Biol* 13.7 (2015): e1002195.

Thanks for Listening!

Questions?