A Quick Introduction to the Micron Automata Chip

Peter M. Kogge
Univ. of Notre Dame

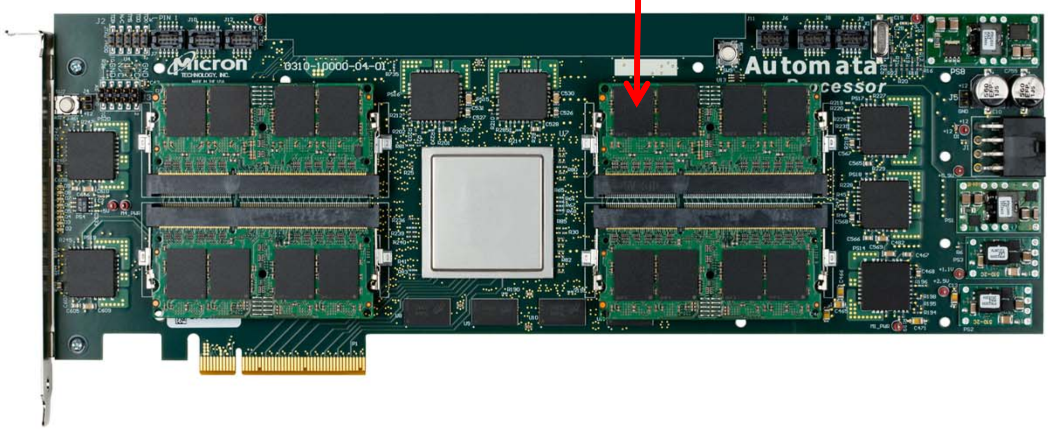
See also:

- Dlugosch, et al. "An Efficient and Scalable Semiconductor Architecture for Parallel Automata Processing," IEEE Trans. PDS, Dec. 2014
- **Center for Automata Processors**, UVA, http://cap.virginia.edu/

Several slides copied from K. Skadron, M. Stan. "Automata Processing: Massively-Parallel Acceleration for Approximate Pattern Matching and String Processing" http://www.clsac.org/uploads/5/0/6/3/50633811/skadron-clsac-2016.pdf

The Automata Processor

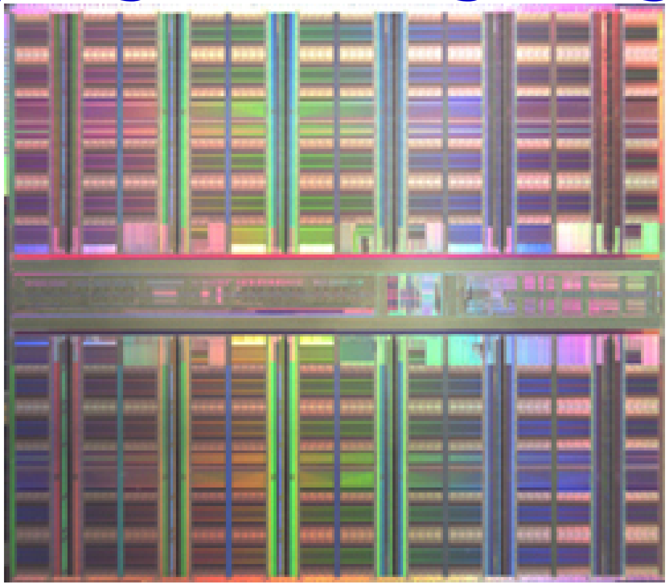
- Hardware accelerator specifically for symbolic pattern matching
- Hardware implementation of *non-deterministic finite automata* (*NFA*) (plus some extra features)
- A highly parallel, reconfigurable fabric comprised of ~50,000 pattern-matching elements per chip. First-generation boards have 32 chips, giving ~1.5M processing elements
- Exploits the very high and natural level of parallelism found in memory arrays
- On-board FPGA will allow sophisticated processing pipelines



AUTOMATA PROCESSING

The Automata Chip

- Massively parallel set of NFAs on a chip
- Designed for complex regular expressions
- Can be expanded to multi-chip systems
- Includes a programming language ANML



https://si.wsj.net/public/resources/images/BA-BJ332A_Tech__NS_20151030225643.jpg

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
- Many combinatorial problems
- Unstructured data

Video Analytics:
- Highly parallel operation
- Real-time operation
- Unstructured data

Data Analytics:
- Highly parallel operation
- Real-time operation
- Complex patterns
- Many combinatorial problems
- Unstructured data

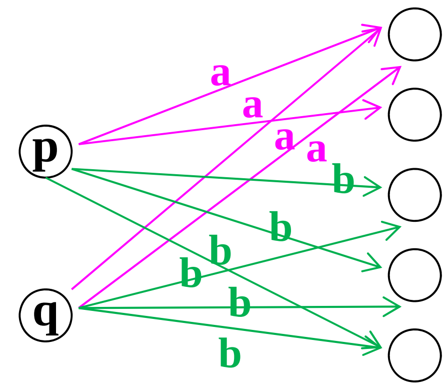
So far: 10-100sX speedups possible!

NFAs and Automata

- Normal definition 5-tuple $(Q, \Sigma, \delta, q_0, F)$
  - Q = set of states
  - Σ: alphabet of input symbols
  - δ: $Q \times \Sigma$ -> P(Q) (set of all subsets of Q)
  - q_0: start state
  - F: set of final accepting states
- Extension to δ: $\delta(C,a)$ = Union of $\delta(q,a)$
  - where C = set of states, q is in C
- Also δ: $\delta(C)$ = **follow**(C) = Union of $\delta(C,a)$
  - Set of all states that you can get to from any state in C
  - Where a is in Σ

Homogeneous Automaton

- All transitions entering a state must occur on same input symbol(s), i.e.
  - If a & b are in Σ, and p & q are in Q
  - Then $\delta(p,a) \cap \delta(q,b) = \delta(q,a) \cap \delta(p,b)$



- State q **accepts** a if
  - a is on some incoming transition to q
- **symbols**(a) = set of all states that accept a
  - $\text{symbols}(a) = \cup_{q \text{ in } Q} \delta(q,a)$
  - q accepts a iff q in symbols(a)
- Thus $\delta(C,a) = \text{follow}(C) \cap \text{symbols}(a)$
  - Remember C is some subset of states

Execution: Input string is S

$C = \delta(q_0)$ **C is all possible next states**
if $q_0 \in F$ **then**
  match the empty string
end if
for each input character α in S **do**
  $T = C \cap symbols(\alpha)$
  if $T \cap F \neq \emptyset$ **then**
    we have a match
  end if
  if T is empty **then**
    stop processing S
  end if
  $C = \delta(T)$
end for

T is set of states that accept next input, given we could be in any of set of states C

Test to see if in accepting state

Test to see if we reject

Update C for next symbol from input

State Sets as Bit Vectors

- Assume |Q| = m (i.e. m states)
- Represent state sets as m-bit bit vectors
  - bit j=1 implies state j is in current set
  - 2^q: m bit vector where position corresponding to q is 1
  - 2^C: OR of all 2^q where q is in C
- $\Delta: 2^Q \times \Sigma \rightarrow 2^Q$ is bit vector equivalent of δ
- Set intersection is now a bit-wise AND

Bit Vector Execution

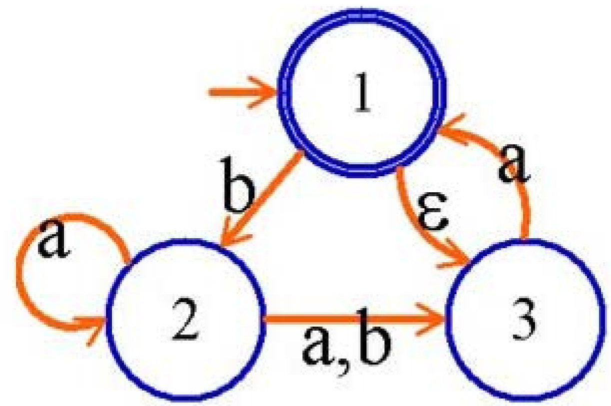
$2^C = 2^I$
if $2^F \,\&\, 0 \times 1 \neq 0$ **then**
    match the empty string
end if
for each input character α in S **do**
    $2^T = 2^C \,\&\, symbols[\alpha]$
    if $2^T \& 2^F \neq 0$ **then**
        we have a match
    end if
    Set $2^C = 0, \ \forall_q \in T, \ 2^C = 2^C | follow[q]$
    if $2^C = 0$ **then**
        stop processing S
    end if
end for

& = bit vector AND

| = bit vector OR

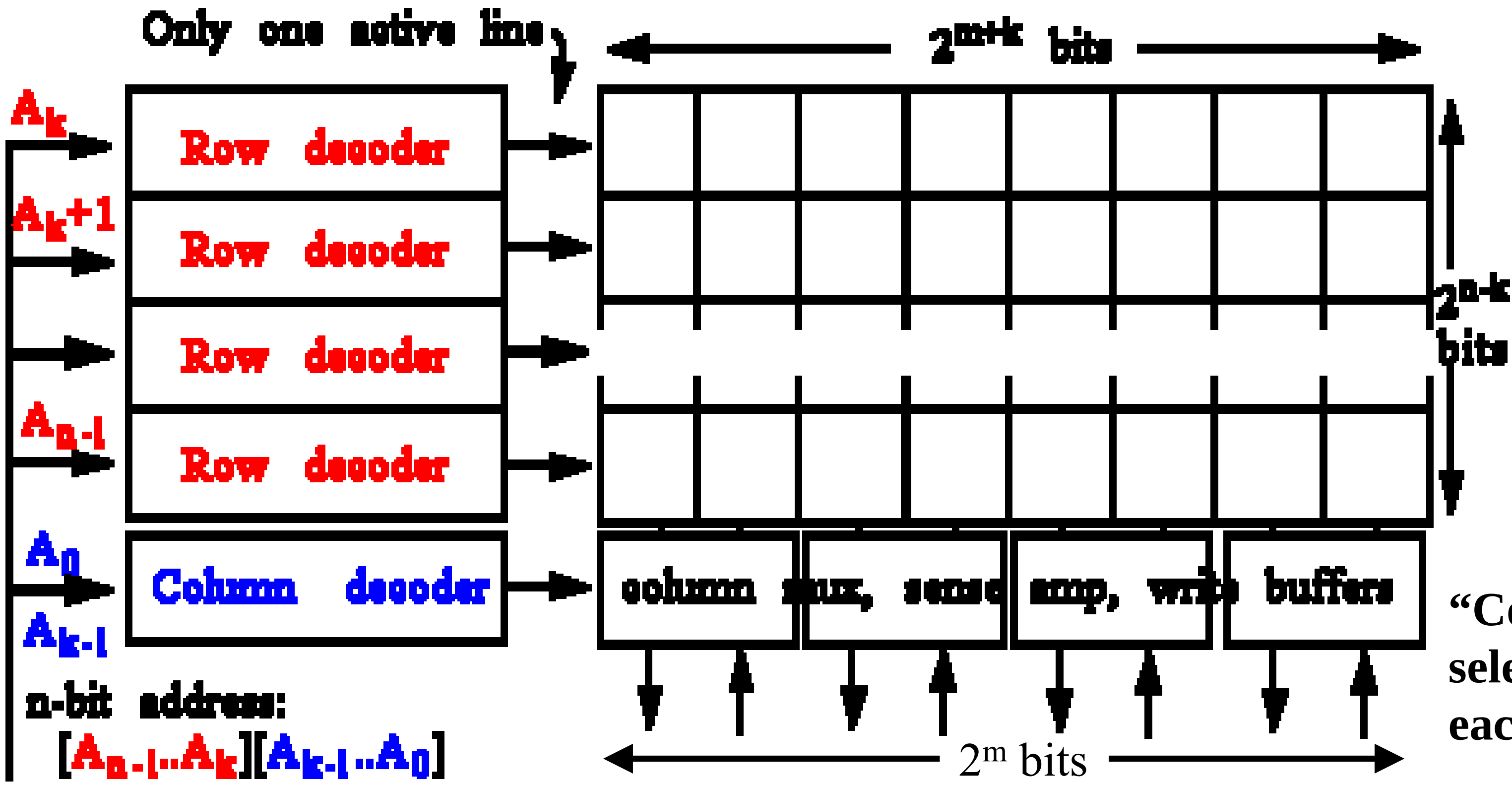
Bit Vector Example

- Consider the NFA N4 Fig. 1.36 on p. 53
  - δ(1,b)={2}
  - δ(1,ε)={3}
  - δ(2,a)={2,3}
  - δ(2,b)={3}
  - δ(3,a)={1}



- m=3 (3 states) so bit vector is 3 bits long
  - If C is 011 (either in state 2 or 3)
  - And next input is "a"
  - Then next state is 111: the OR of
    - 100 (from δ(3,a)={1})
    - 011 (from δ(2,a)={2,3})

Conventional Memory Block



"Column mux" selects 1 out of each 2^k bits

For example: Let N = 1,048,576 and M = 8 bits for a 1 million byte memory.

$n = \log_2 N = 20$, $k = 8$ and $m = \log_2 M = 3$.

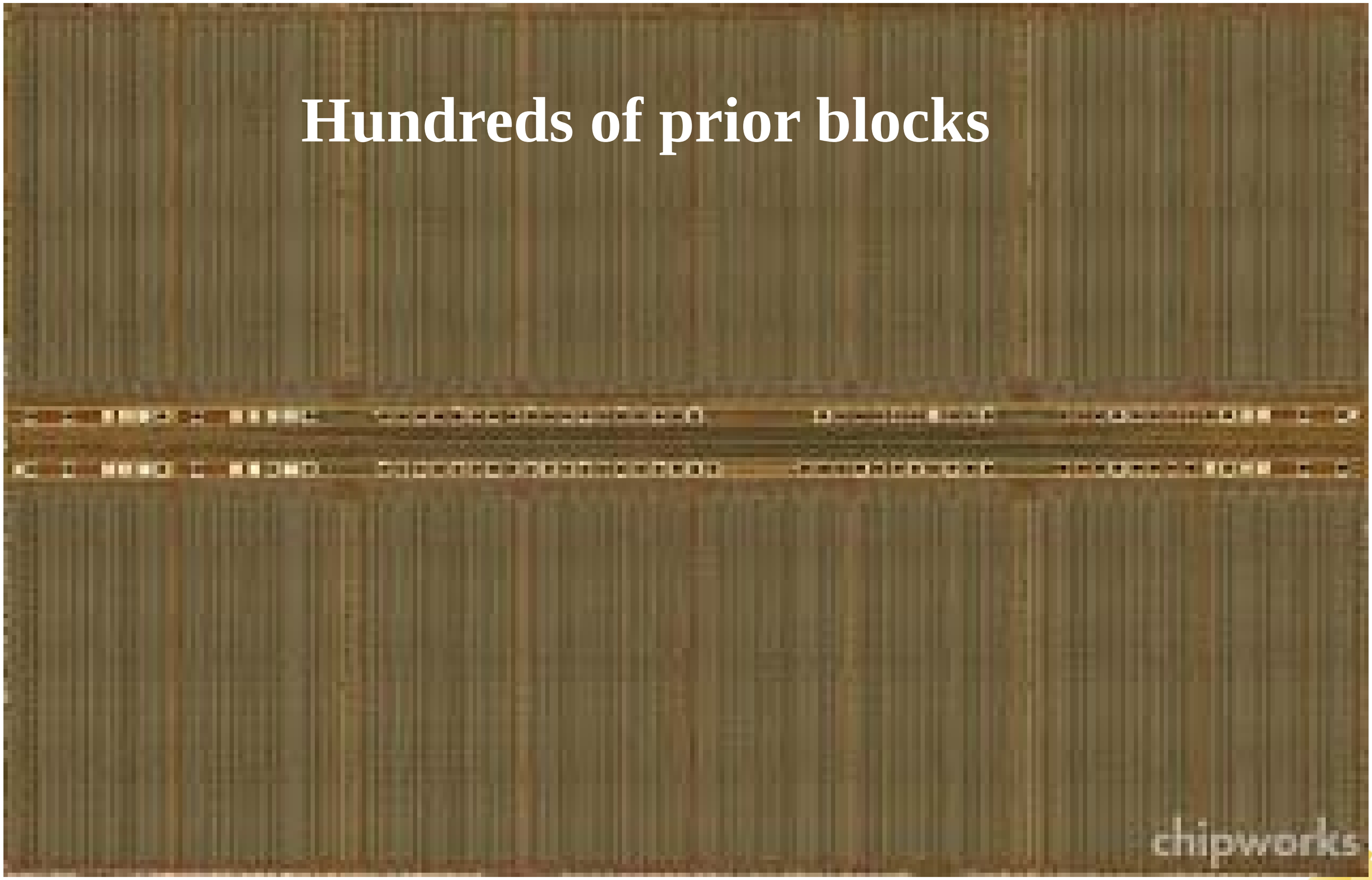
Then there are 2^{n-k} rows $= 2^{12} = 4096$ and

2^{k+m} columns/2^3 bits per word $= 2^8 = 256$ words.

Example = ~8 Mbits

http://ece-research.unm.edu/jimp/vlsi/slides/chap8_2-1.gif

A 2Gb DRAM Chip



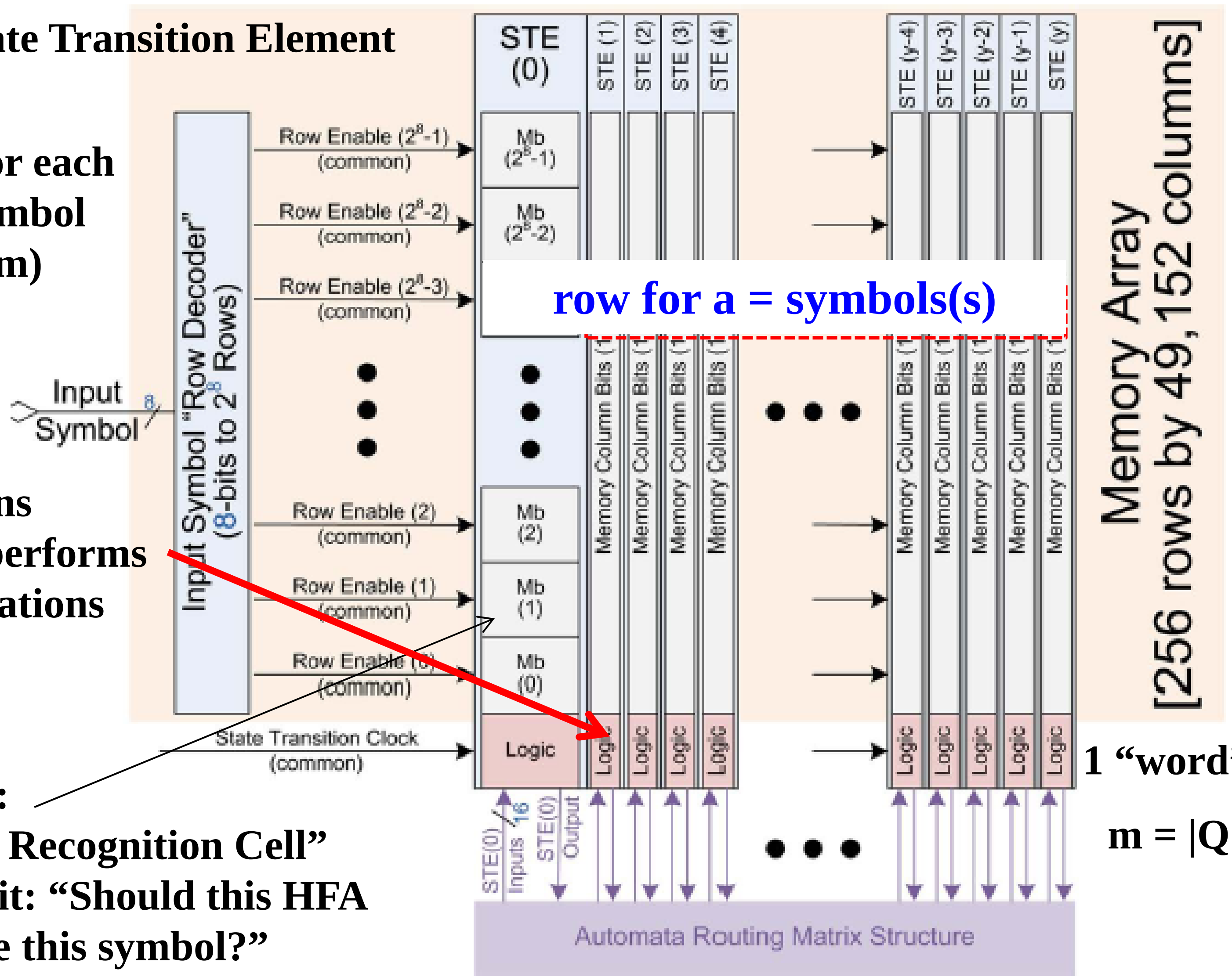
http://2.bp.blogspot.com/__ZSc4tYyVNI/TUbE3RjzDfI/AAAAAAAAAFo/oP7UqXefwtU/s320/K4B2G0846B-HCH9_K4B2G0846D-s-r_branded.jpg

Automata Chip

STE = State Transition Element

One row for each possible symbol (256 of them)

Logic contains C,T … and performs & and | operations



1 "word" = 49Kbits

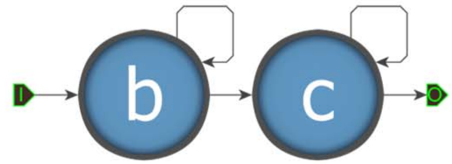
m = |Q| = 48K

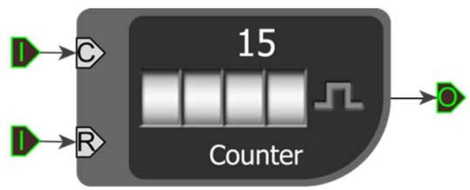
Each "Mb":

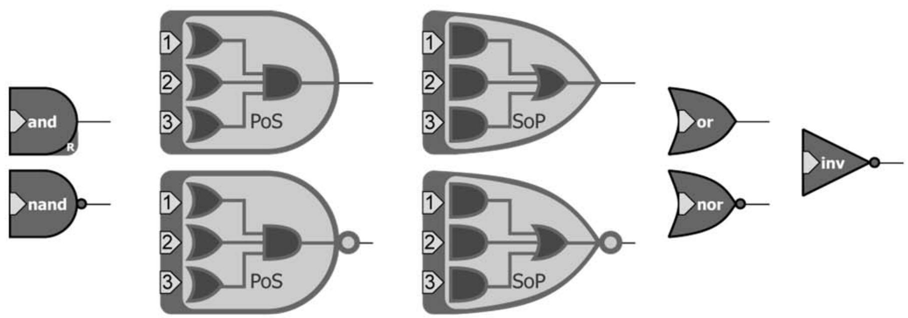
- **"Symbol Recognition Cell"**
- **1 State bit: "Should this HFA recognize this symbol?"**
- **Row driven only if symbol ok**

Automata Processor Hardware Building Blocks

	per chip
State Transition Element (STE)	49,152
Counter Element	768
Boolean Logic Element Nine Programmable Functions	2,304
Report buffer	6,144





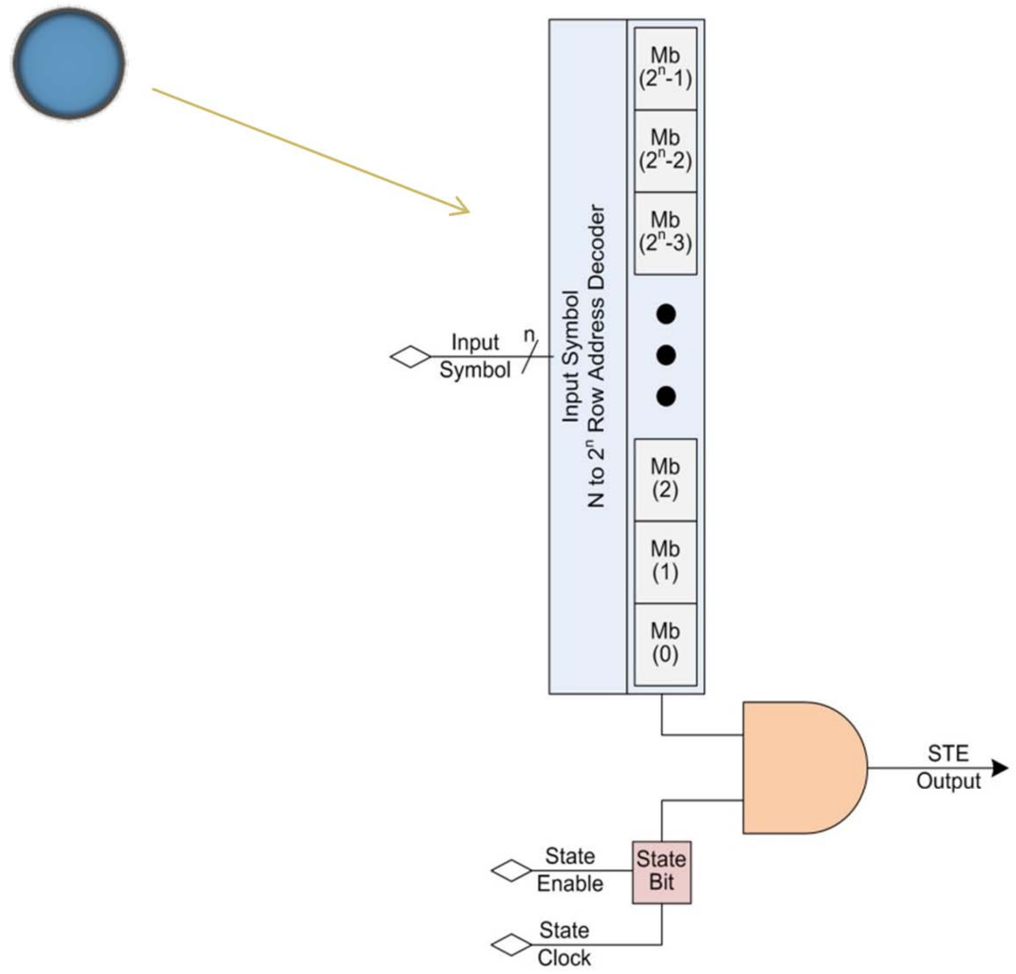


Figures courtesy of Micron

- Important: ALL elements on all chips see input symbol every cycle

Automata Processor: Basic Operation

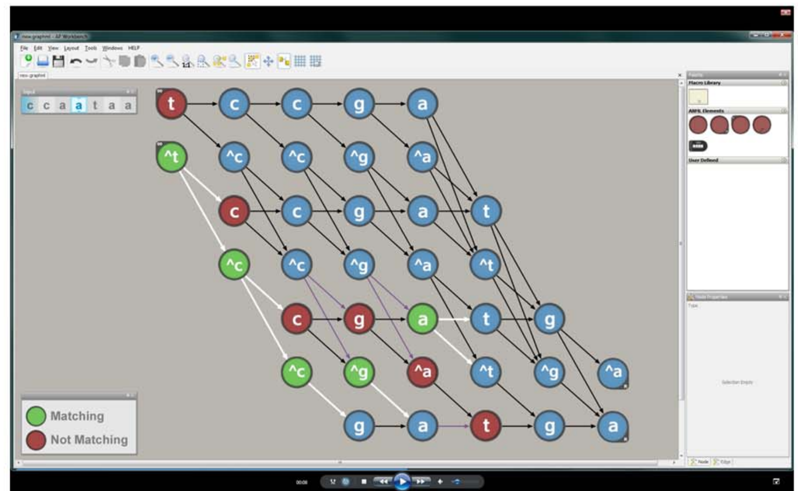
- STE "fires" when
  - Symbol match
  - AND the STE is active

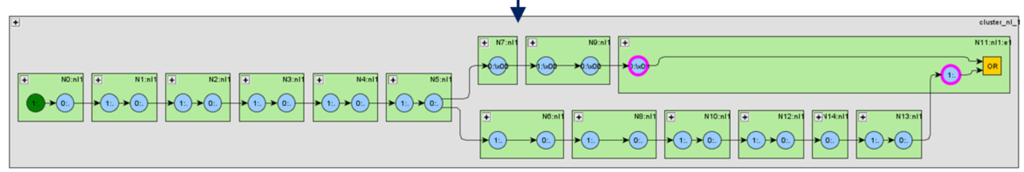


Figures courtesy of Micron

Programming Options

- Currently, like other PCIe-attached accelerators
  - Offload model, mediated by device driver
- Input
  - RegEx
  - GUI – Workbench
  - C/Python APIs
  - RAPID – C-like language
  - ANML
- Compiling
  - Input → ANML
  - ANML→ Netlist
  - Netlist → Place & route

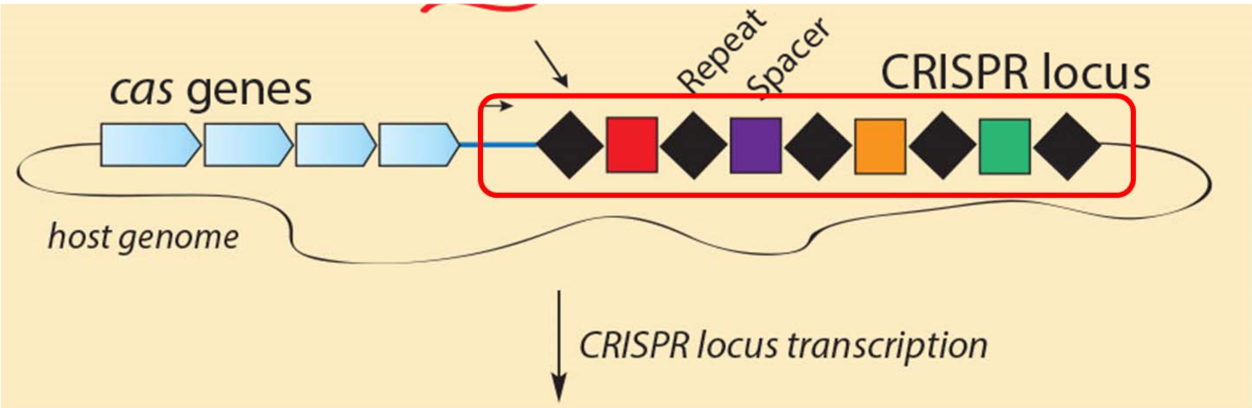




Figures courtesy of Micron

Bioinformatics: CRISPR Sites Discovery

- CRISPR: Clustered Regularly Interspaced Short Palindromic Repeats
- Each repeat is followed by a spacer DNA and the spacer could be either the same or different
- Mismatches/gaps may be allowed in repeats
- Potential applications: genome engineering, RNA editing, Biomedicine, etc.



Preliminary Results

- Find 100 and 500 CRISPRs
- Allow different number of mismatches (1~5)
- Promising speedup achieved, from 40.7x to 402x
- Speedup is better for larger database