Automata Processor

Tobias Markus

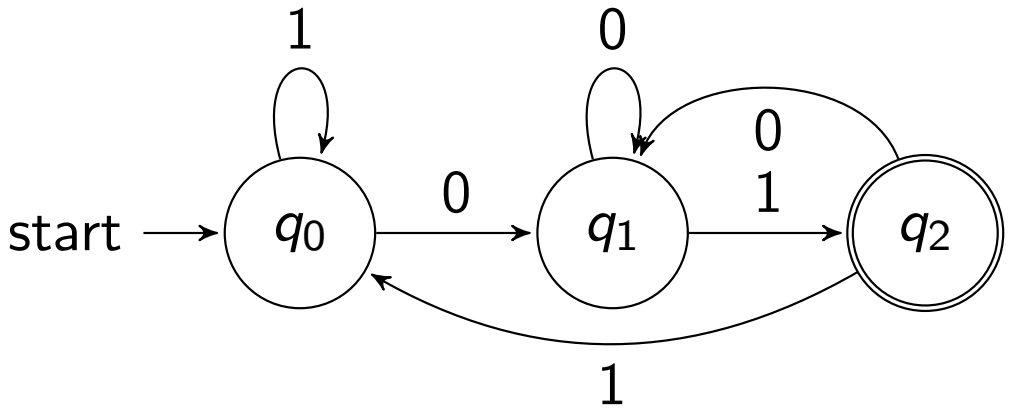
Advanced Seminar "Computer Engineering"

January 21, 2015

Future Accelerators

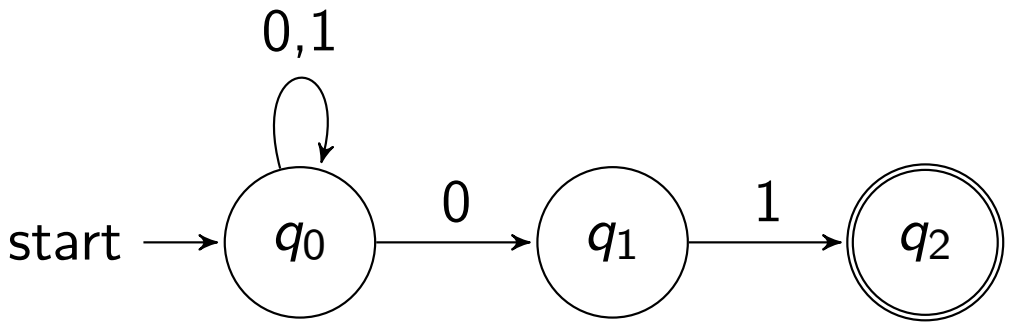
- Need for parallel systems
- GPUs, Cluster good for structured data
- Accelerator for difficult problem loads (e.g. pattern matching)

Deterministic Finite Automaton (DFA)



- Exactly one active state
- Exactly one active transition (transition function)
- In this example detecting binary strings ending with "01"

Nondeterministic Finite Automaton (NFA)



- Multiple active states are allowed
- Transition relation can result in multiple states
- Fewer transitions and states are possible compared to DFA

Mathematical model

NFAs can be described by a 5-tupel:

$$< Q, \Sigma, \delta, q_0, F > \tag{1}$$

- Q: set of automaton states
- Σ: input alphabet
- $\delta(q, \alpha)$: transition relation
- q_0: start state
- F: set of final states

Definitions

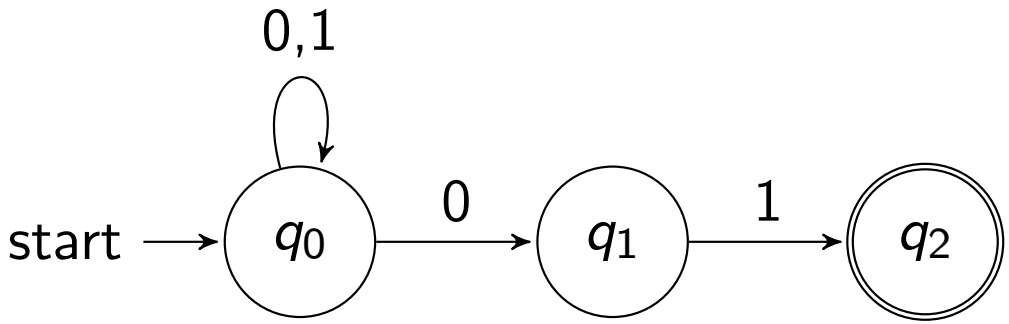
Accepting a word

- $r_0 = q_0$
- $r_{i+1} \in \delta(r_i, a_{i+1}), i = 0, ..., n-1$
- $r_n \in F$

The recognized Language $L(M)$

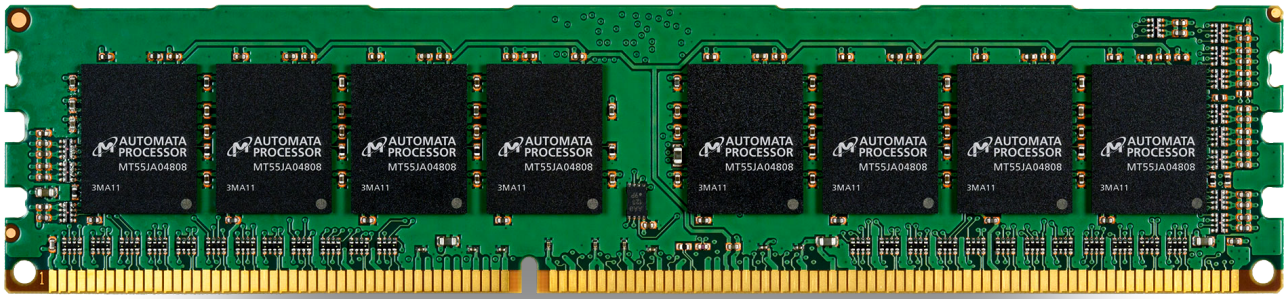
$$L(M) = \{w \mid \delta^*(q_0, w) \cap F \neq \emptyset\} \tag{2}$$

Example



- $\delta^*(q_0, 0) = \{q_0, q_1\}$
- $\delta^*(q_0, 00) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$
- $\delta^*(q_0, 001) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$
- $\delta^*(q_0, 0010) = \delta(q0, 0) \cup \delta(q_2, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$
- $\delta^*(q_0, 00101) = \delta(q0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$

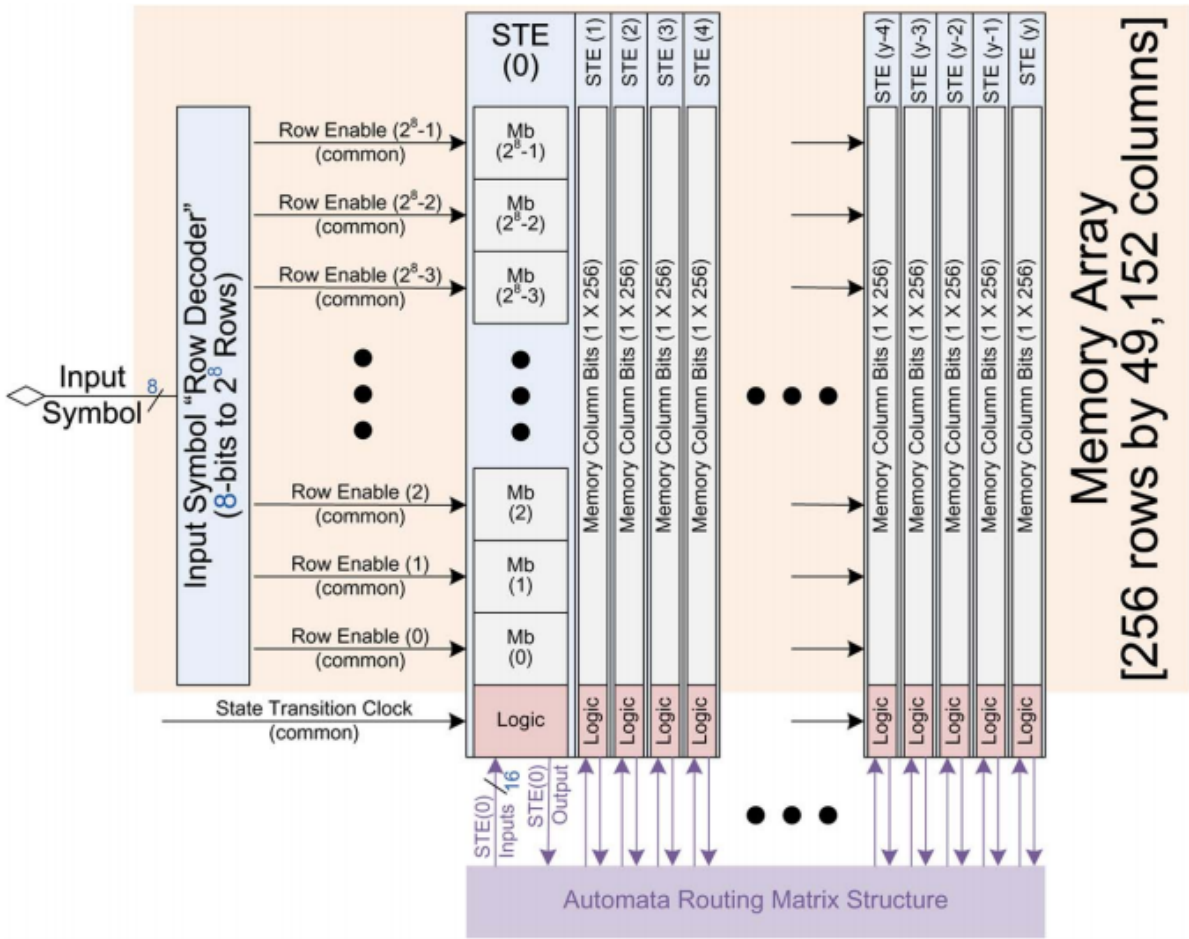
Automata Processor Memory Array



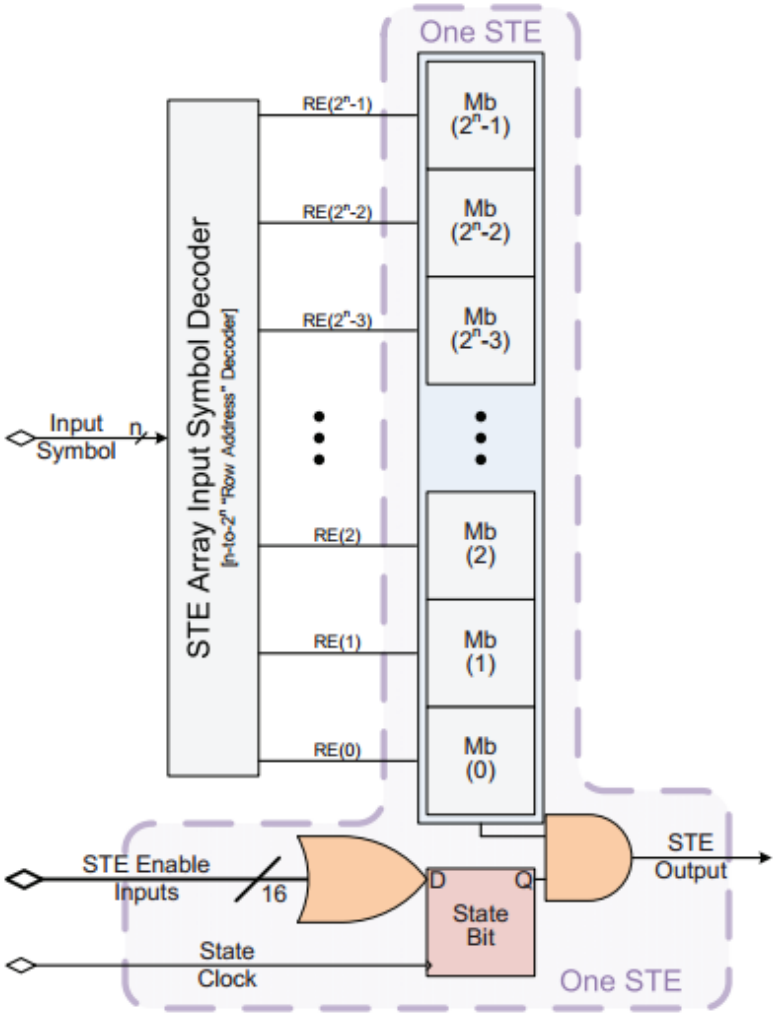
Automata DIMM format[1]

- Is implemented in Microns standard DDR3 SDRAM technology
- Processes 8 bit input symbols at 1 Gbps
- Input symbol is given as row address
- Invokes operations with State Transition Elements (STE) and the Routing Matrix

AP Memory Array [2]



State Transition Element (STE)



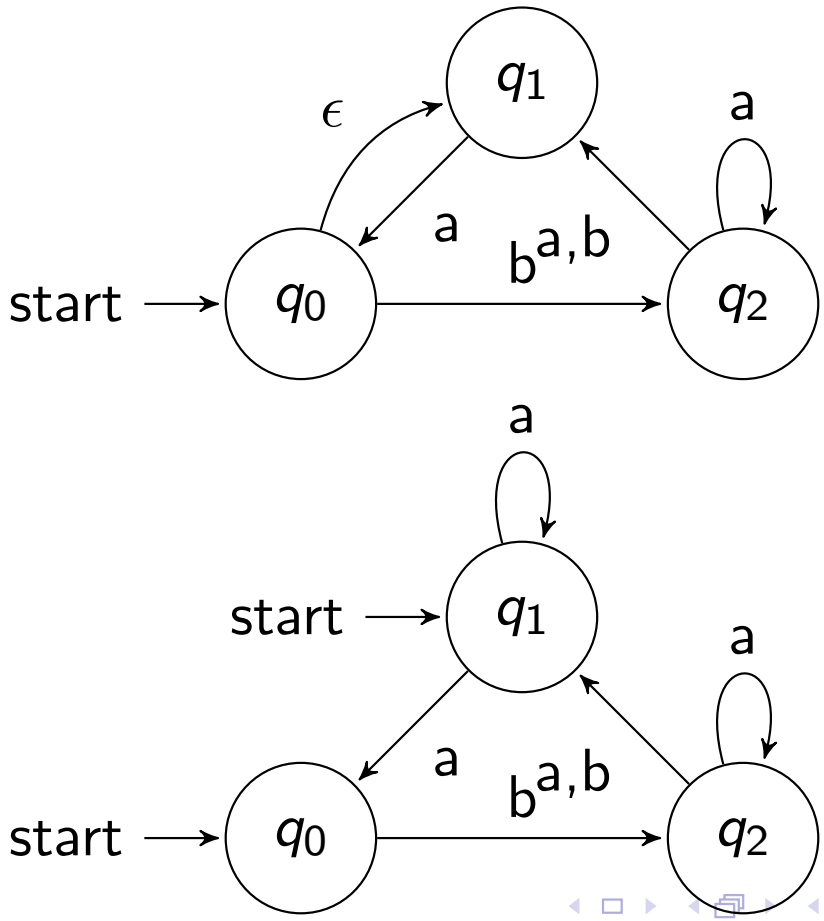
STE [2]

Consists of:

- State memory
- State transition logic, state bit

State Transition Element (STE)

- State memory contains matching character or character class
- Each state bit can be programmed multiple start states



Counter

- 12 bit counter
- To simplify automaton
- Possible to implement a subset of pushdown automaton
- Turing complete

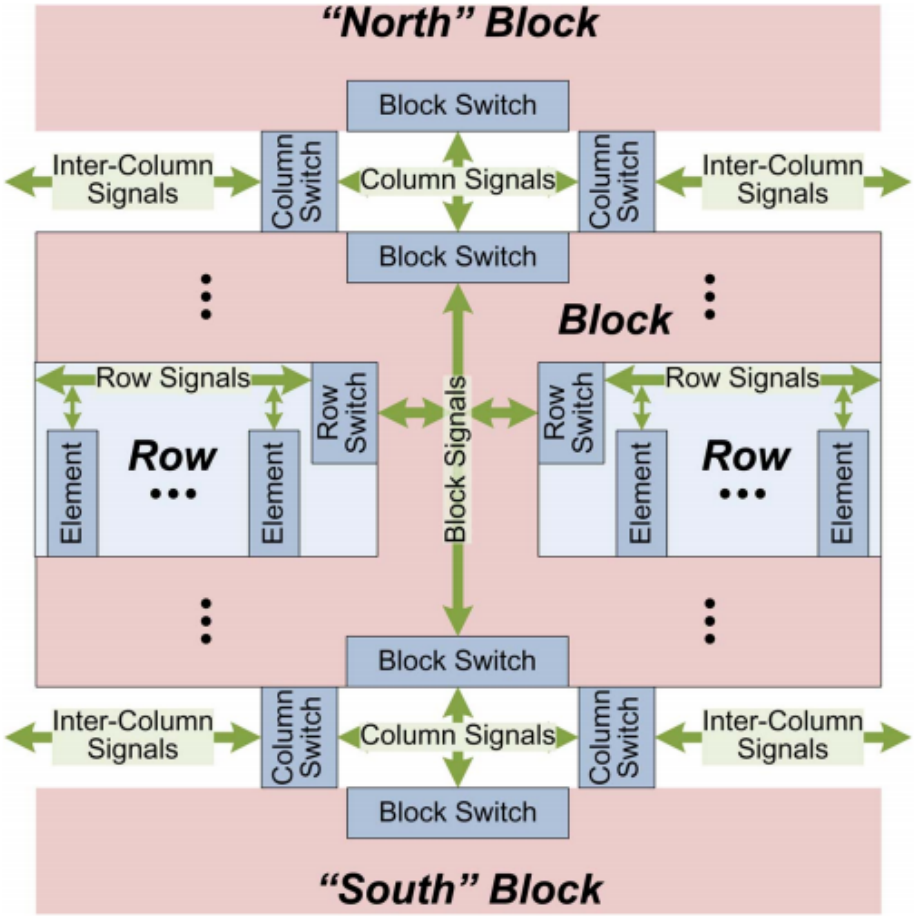
Boolean Element

- Configurable to
  - OR
  - AND
  - NAND
  - NOR
  - sum of products
  - product of sums
- No state
- Also to simplify automaton

Routing Matrix

- Programmable routing matrix to connect the STEs
- Theoretically every state must be able to connect with every other
- Implemented hierarchically
- Grouped in row $\rightarrow$ rows grouped in blocks $\rightarrow$ grid of blocks

Routing Matrix



Hierarchical routing scheme [2]

Reconfigurability and Inter Rank Bus

The AP is:

- Partially reconfigurable
- There is an inter rank bus to distribute input symbols over APs

Programming the AP

- Perl Compatible Regular Expressions (PCRE)
- Automaton Network Markup Language (ANML)

Description via PCRE

$$/a(b|c) * [de]/i$$

PCRE example

- For some regular expressions software post-processing is needed
- Some restrictions to assure hardware performance
- Regular Expressions can be defined within C/C++
- AP SDK is used to compile them

C Example

```
#include <ap_compile.h>

ap_automaton_t CreateAutomaton(void)
{
    ap_automaton_t amton;
    ap_exprdb_t db;

    db = AP_CreateExpressionDB();

    AP_AddExpression(db, NULL, "/a(b|c)*[de]/i", 1,
        0, AP_GRAMMAR_PCRE_DELIMITED);

    AP_Compile(db, &amton, 0, NULL, 0,
        DEVICE_FAMILY_FRIO);

    AP_DestroyExpressionDB(db);

    return amton;
}
```

Description via ANML

- XML-based language
- Directly describes automata
- ANML can be produced from C/C++
- ANML can be graphically developed from AP Workbench

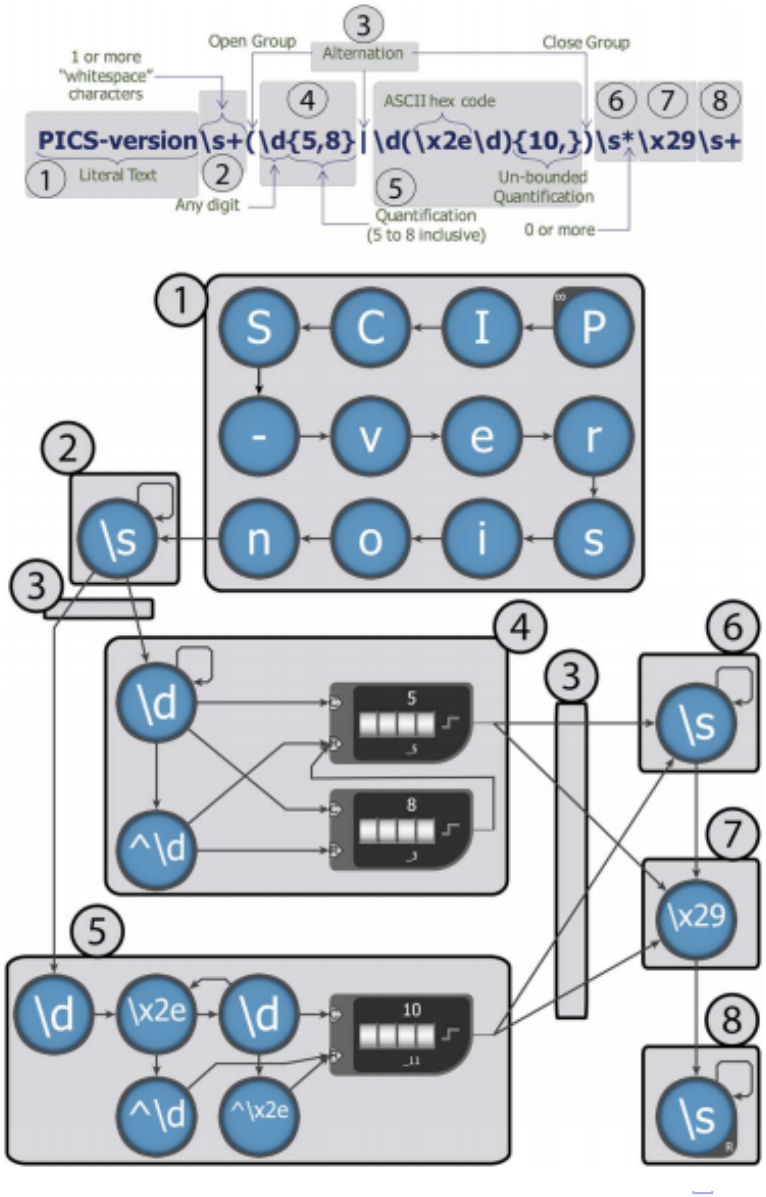
Application fields for the AP

- Bioinformatics
- Network Security
- Sigint and Crypto
- Finance applications

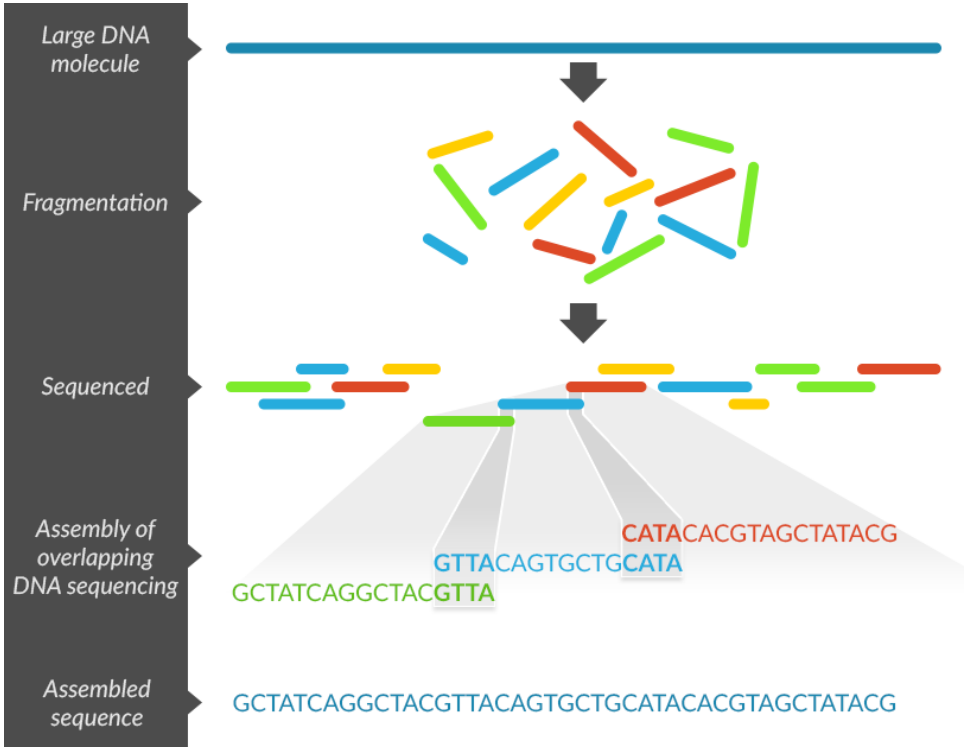
Network Security

- Signature based detection of known malware patterns
- AP is good for complex RegEx
- Allows generic signature matching

Network Security [2]



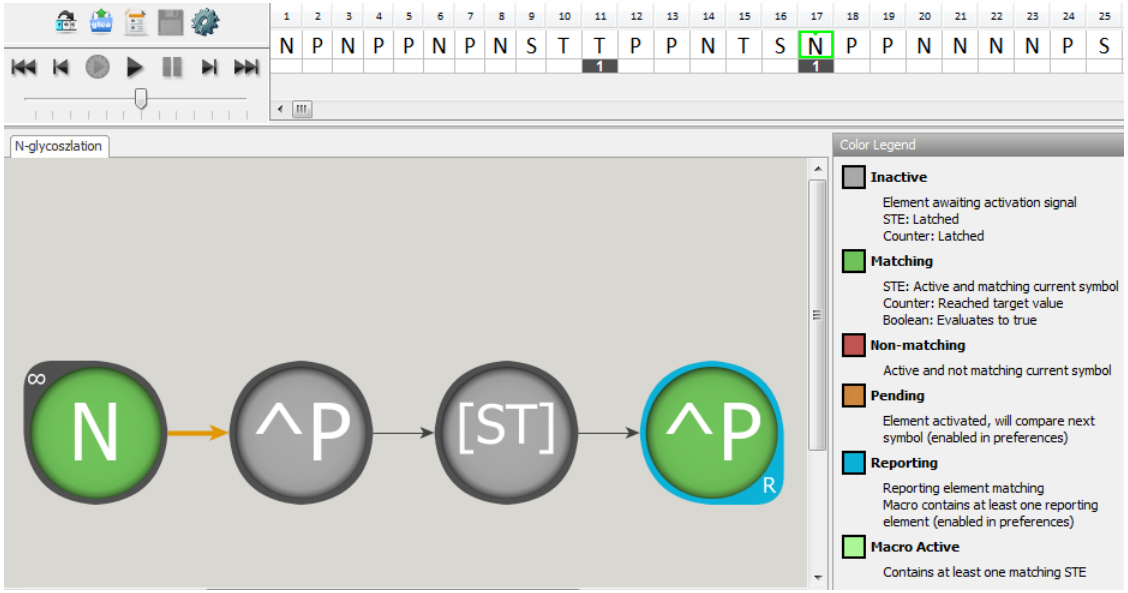
Bioinformatics



Modern DNA Sequencing workfow [3]

- Modern DNA Sequencing
- Need the assembly of overlapping sequences
- Another task, the search for motifs

Motif matching



Conclusion

- Very specific applications
- Difficult to determine if the AP fits for a problem
- Defined language to describe automaton (ANML)
- Logic implemented in DDR3 SDRAM technology
- Fits good into current accelerators

References

- [1] Micron. Automata http://www.micronautomata.com/
- [2] Paul Glendenning Michael Leventhal Harold Noyes Paul Dlugosch, Dave Brown. An efficient and scalabe semiconductor architecture for parallel automata processing. IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS 25:11, 2014.
- [3] Micron. Automata applications. http://www.micronautomata.com/#applications

Questions?

Additional/New Information

Planted Motif Search Problem	Automata Processor	UCONN - BECAT Hornet Cluster
Processors	48 (PCIe Board)+CPU	48 CPU (Cluster/OpenMPI)
Power	245W-315W[1]	>2,000W[1]
Cost	TBD	~$20,000[1]
Performance (25,10)	12.26 minutes[2]	20.5 minutes
Performance (26,11)	13.96 minutes[2]	46.9 hours
Performance (36,16)	36.22 minutes[2]	Unsolved

Performance table from Micron AP Briefing