FAST SUPPORT FOR UNSTRUCTURED DATA PROCESSING: THE UNIFIED AUTOMATA PROCESSOR

Yuanwei Fang, Tung T. Hoang, Michela Becchi, Andrew A. Chien

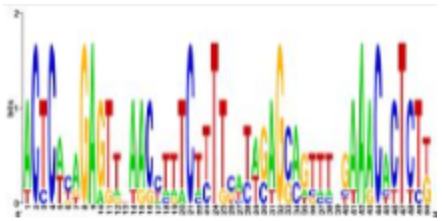
Unstructured Data Applications

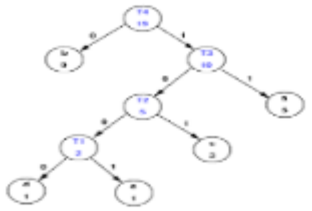
Voice
Assistant

Web Search

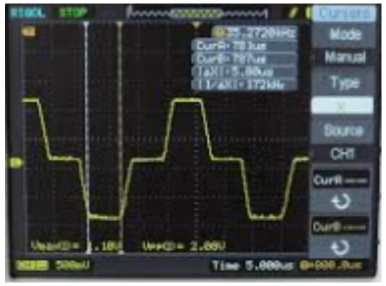
Intrusion
Detection

XML Mining/
Parsing


Motif
Searching

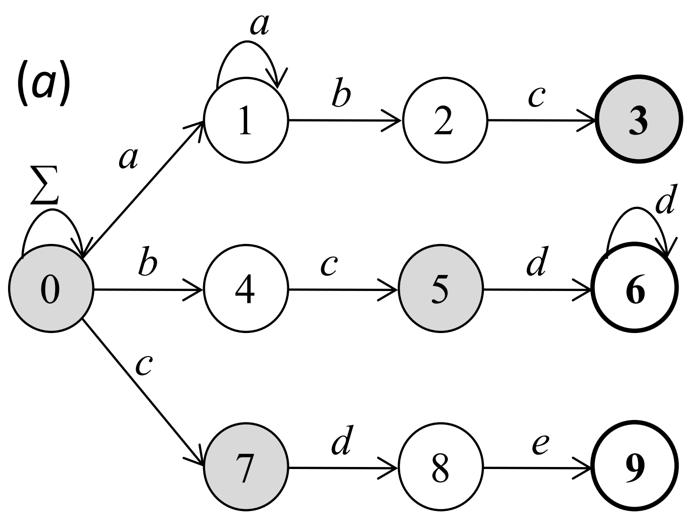

Compress/
Decompress

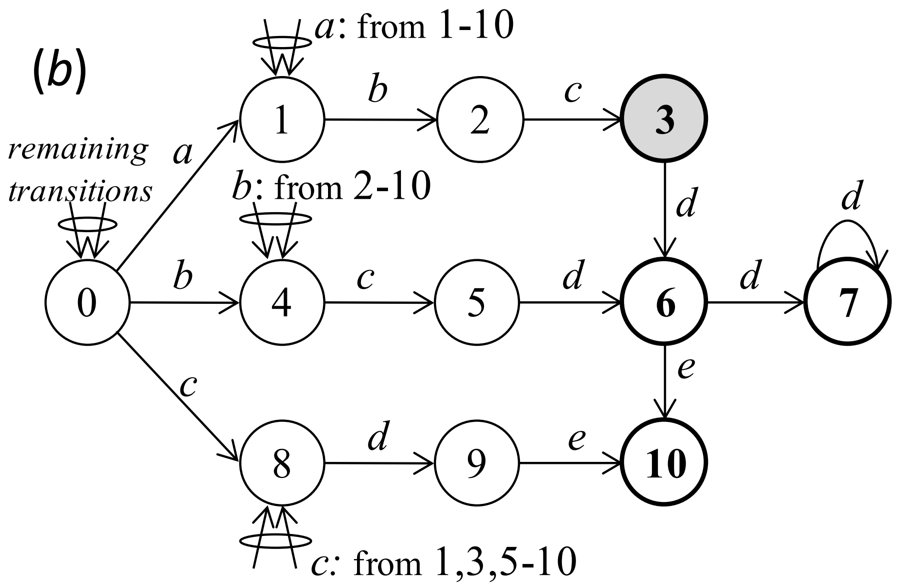
Text Data Mining


Signal
Triggering

- Unstructured Data: No pre-defined data model, text-intensive.
- Growing applications of "Big Data"; Dataset scaling in size, and sophistication
- Increasingly "real time"

Finite Automata: a Powerful Tool for Pattern Matching



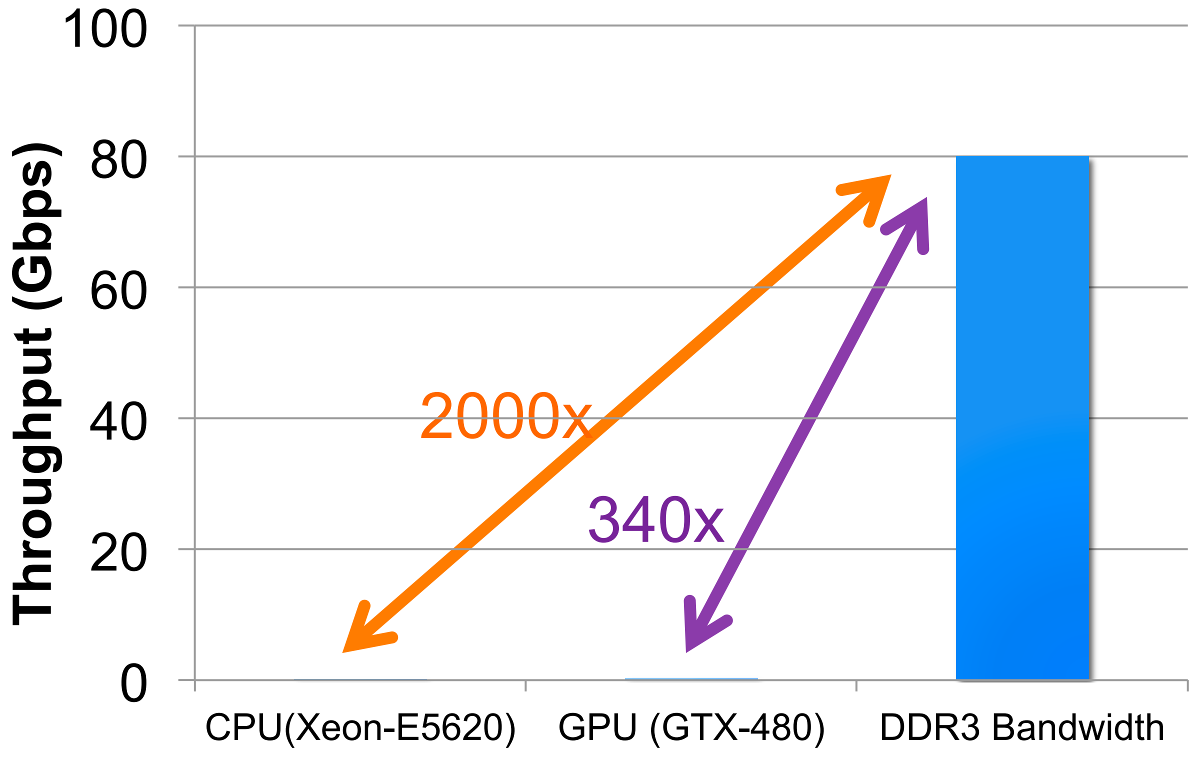


NFA **DFA**

Search Pattern: a+bc, bcd+, cde

Why Automata Processing has Poor Performance on CPU/GPU?

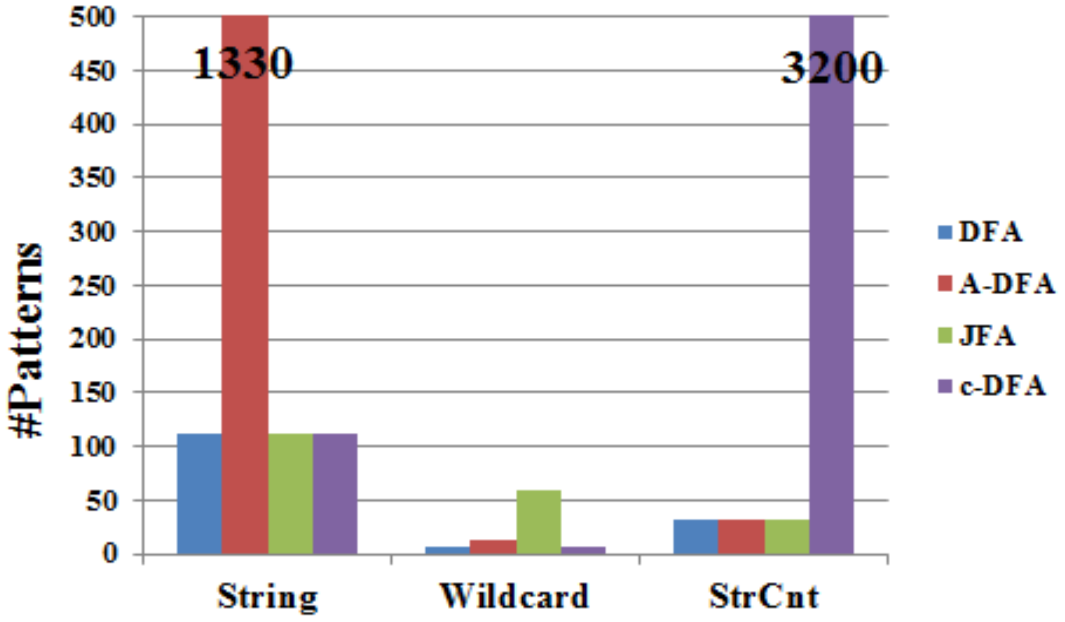
- Worst case for modern processors
  - Sequential memory indirections
  - Unpredictable accesses / branches
  - Short data items
  - Memory hierarchy



Importance of General-purpose FA Processing

FA Model Innovation

- DFA, NFA (classical)
- A-DFA, delay-DFA [SIGCOMM'06]
- Hybrid-FA [CoNEXT'07]
- XFA [SIGCOMM'08]
- Counting-FA [CoNEXT'08]
- SFA [INFOCOM'11]
- Dual-FA [ToC'13]
- JFA [JSAC'14]
- ……



To date, no such general-purpose architecture exists!

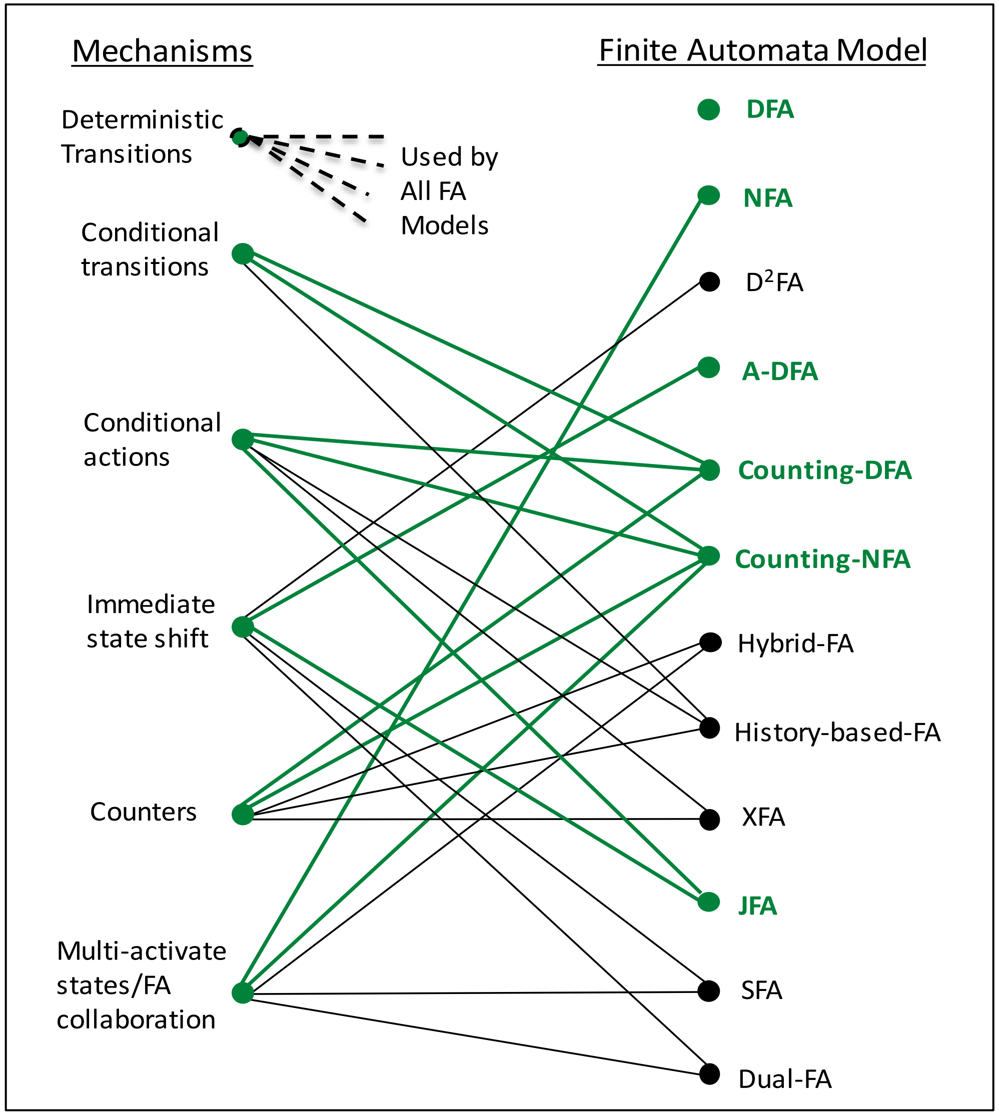
Goals for a Unified Automata Processor (UAP)

- General-purpose:
  - support all finite-automata models
- High Performance:
  - fast enough to match the data movement speed in the memory system
  - low enough power to fit anywhere in the system
- Ease of Use:
  - same programming model as CPU
  - same way as people use regex library

General-purpose and Flexibility

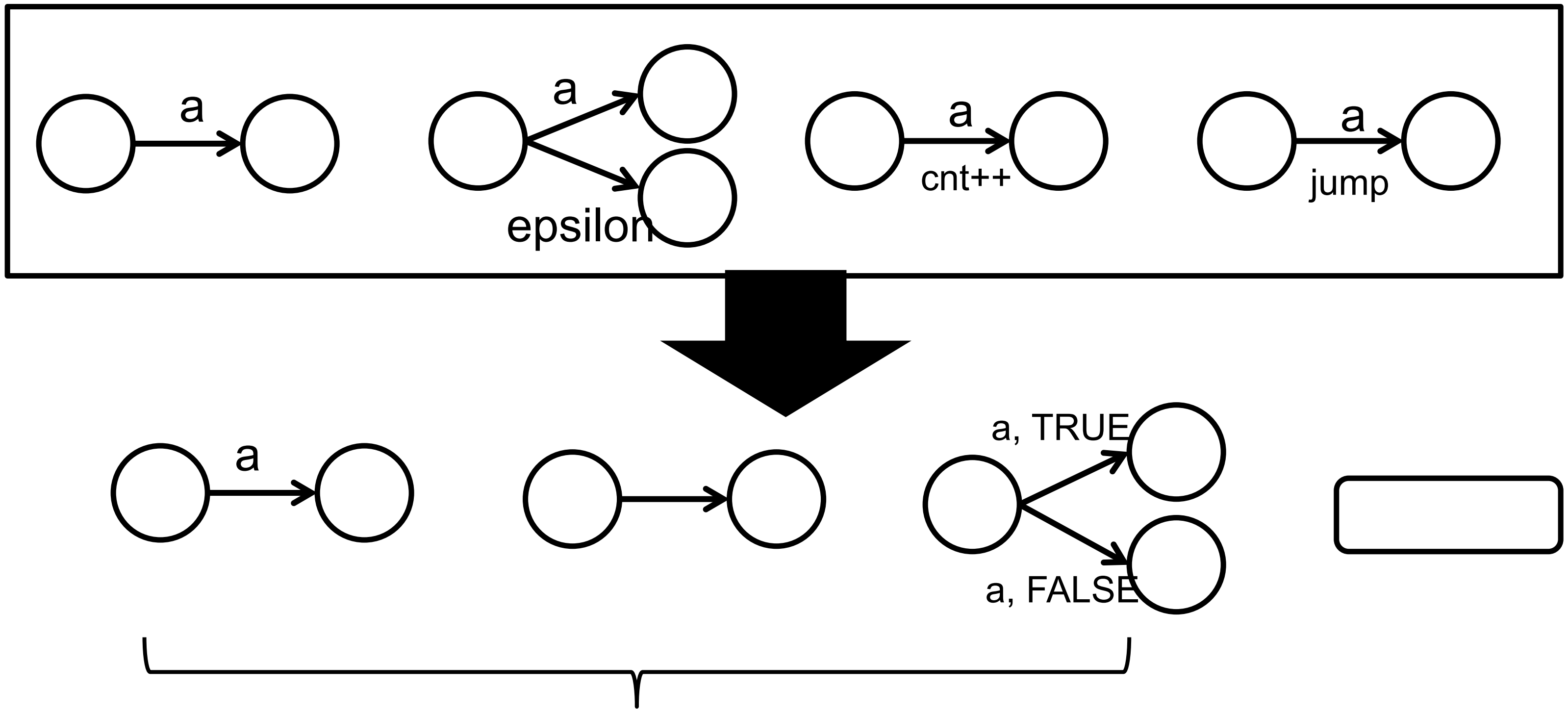
- Detailed Analysis of Dozens of Models
  - identify key data representations
  - Identify common mechanism requirements

- Transition variety is the key
- Transition types define different FA models



Support Transition Diversity

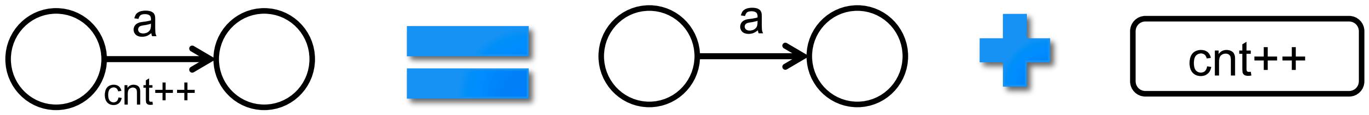
Approach: Transition Decomposition



Transition Primitive

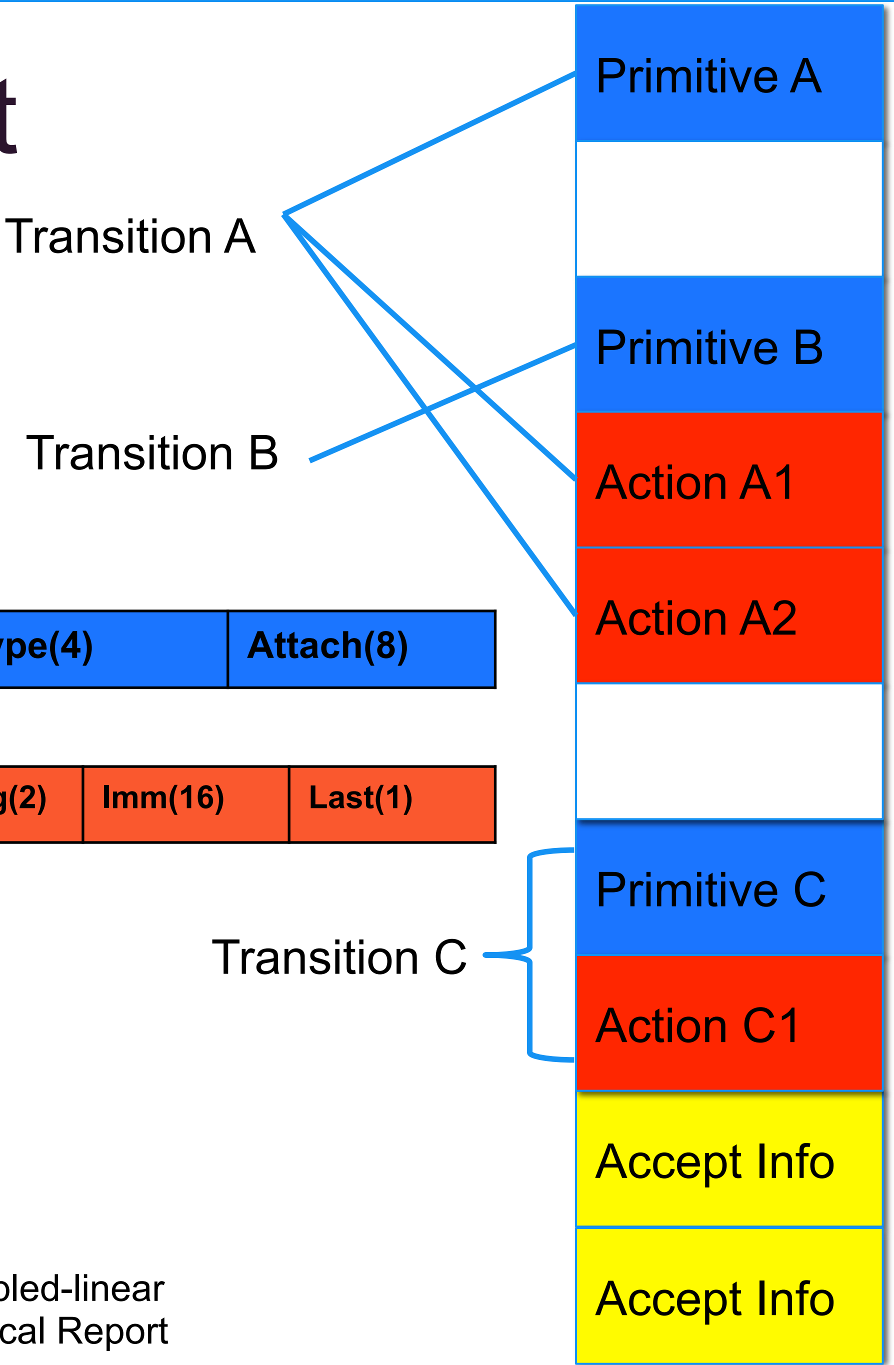
Action

Example:



UAP Memory Layout

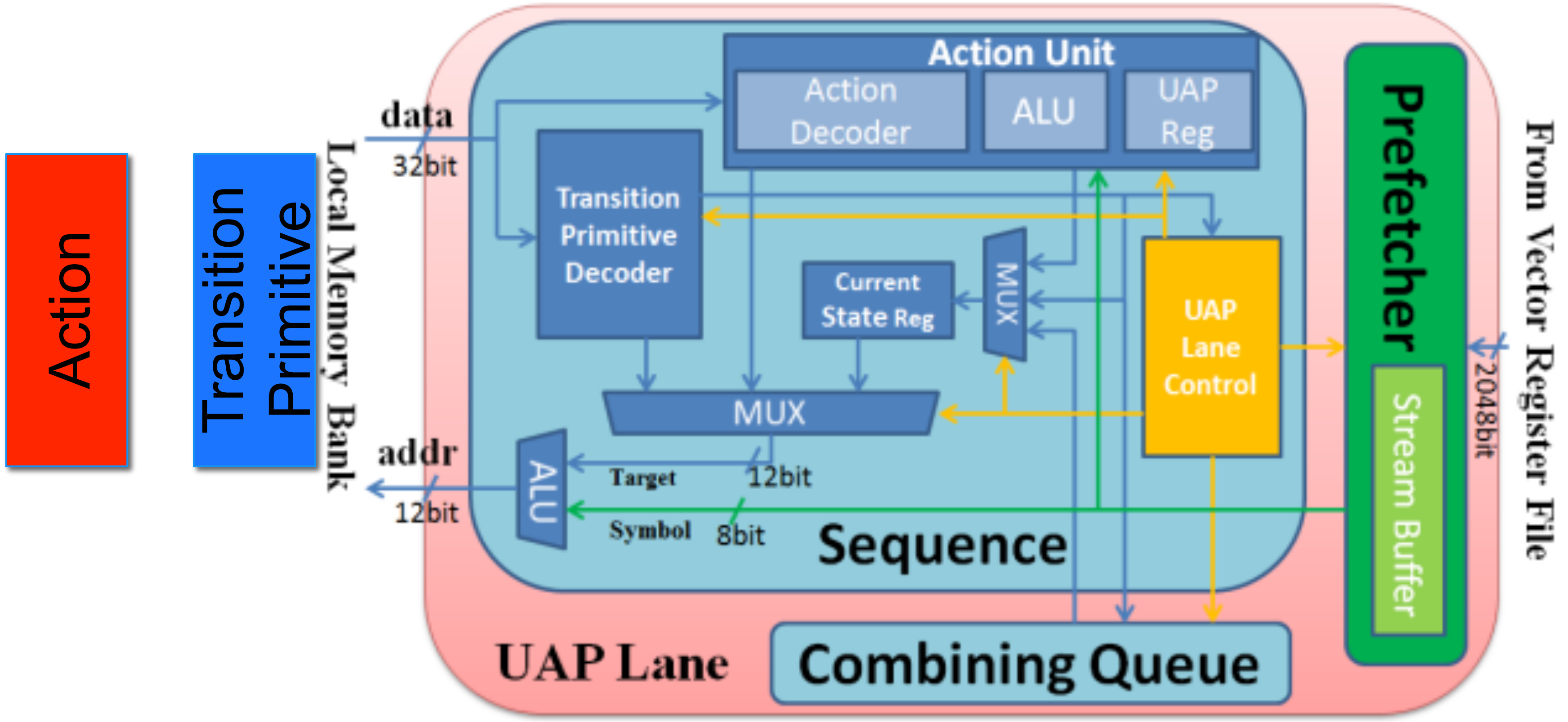
- Using EffCLiP algorithm[1] to pack transitions and actions



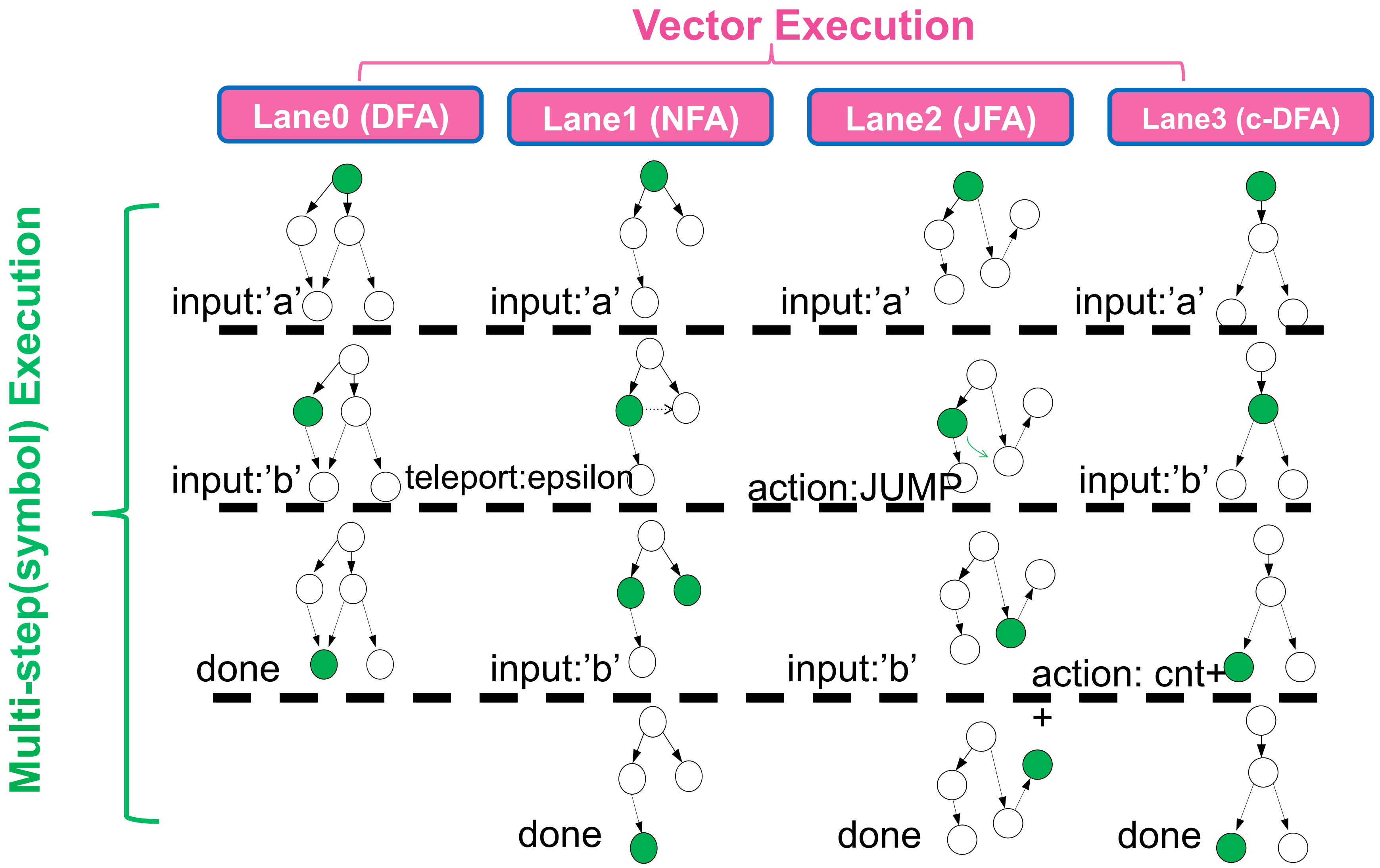
Transition Primitive	Signature(8)	Target(12)	Type(4)	Attach(8)		
Action	Unused(7)	Opcode(4)	DstReg(2)	SrcReg(2)	Imm(16)	Last(1)

1. Y. Fang, A. Lehane, A. A. Chien, "EffCLiP: Efficient Coupled-linear Packing for Finite Automata", University of Chicago Technical Report

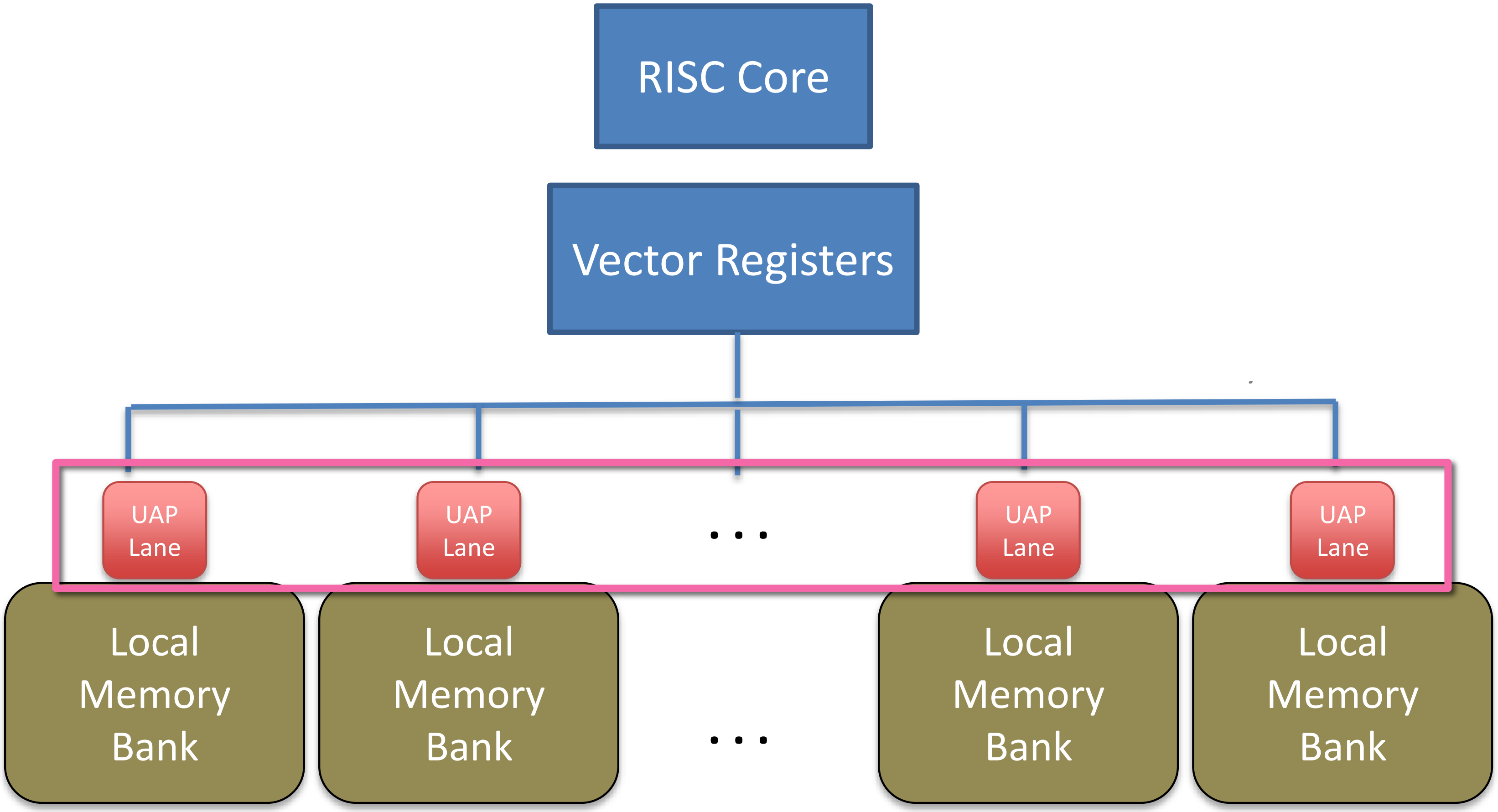
Basic UAP Engine



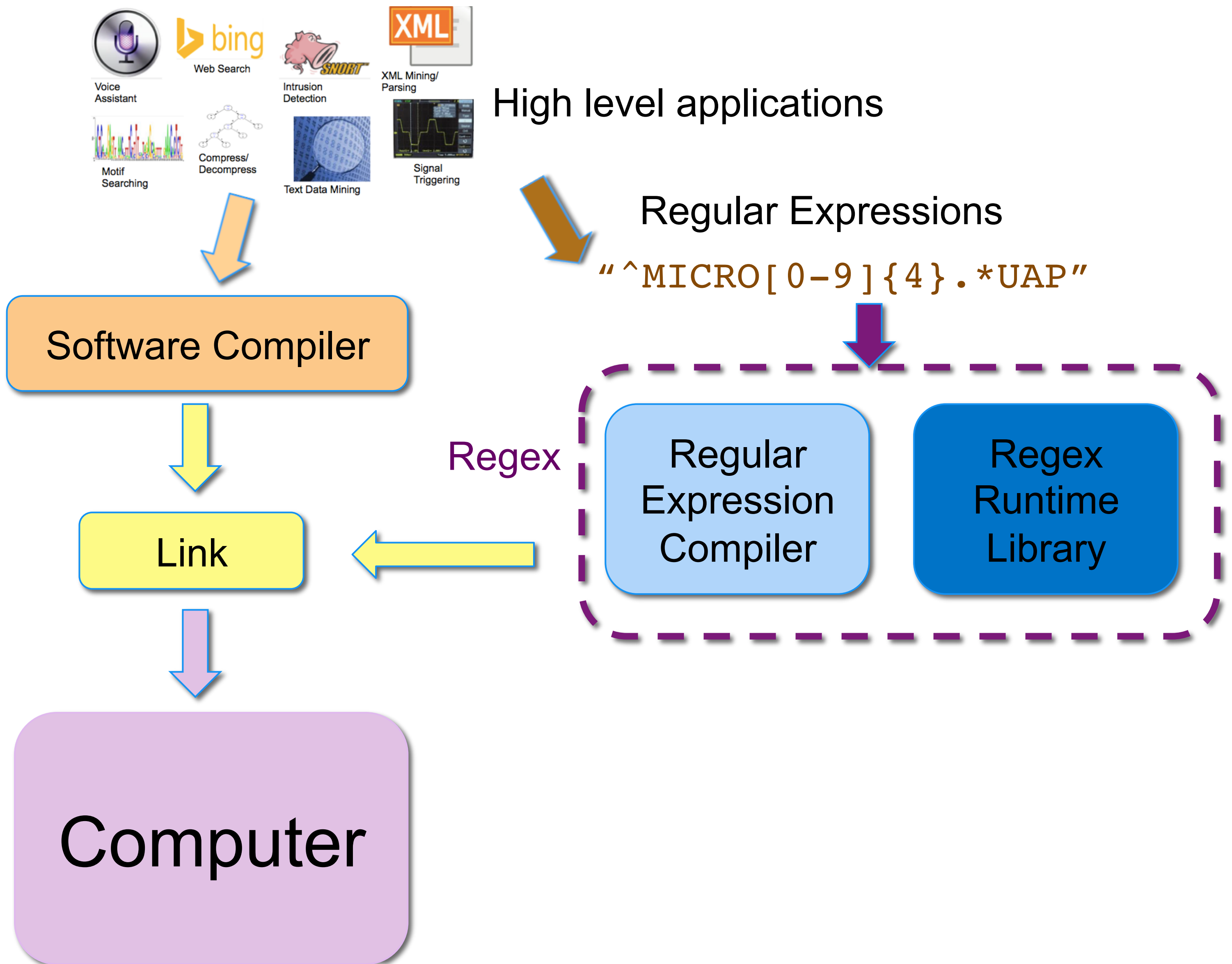
Vector Parallelism and Multi-step Execution



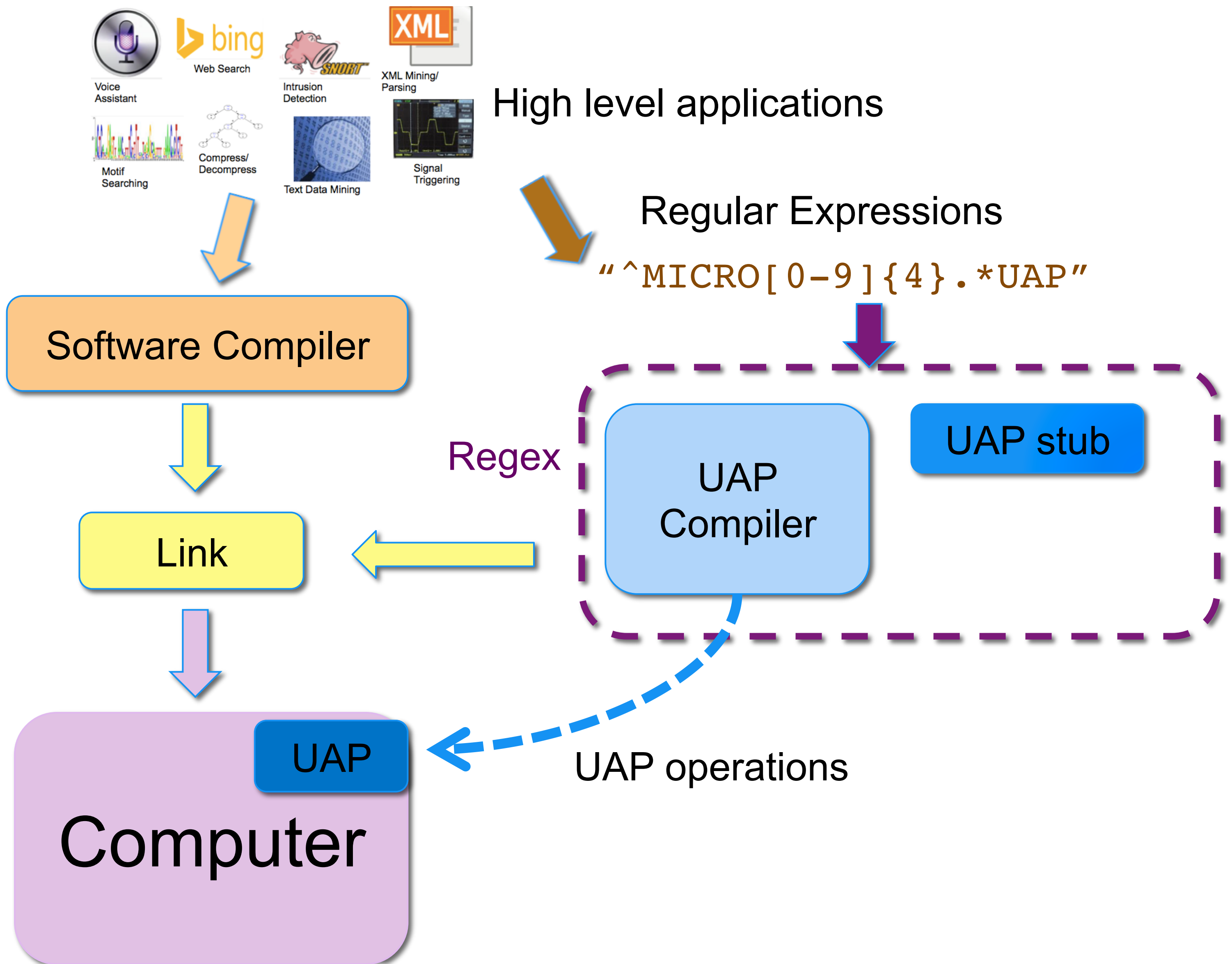
UAP System Architecture



Use Regex Library for FA Processing



Use UAP for FA Processing



Code to do FA Processing

```
#include <regex.h>
... Your Program’s Code ...
regex_t regex;
int ret;
regcomp(&regex, ”^MICRO[0-9]{4}.*UAP", 0);
ret = regexec(&regex, “abcdefghijklmn”, 0, NULL, 0);
if (!ret) {
    puts("Match");
}
else if (ret == REG_NOMATCH) {
    puts("No match");
}
... Your Program’s Code ...
```

Exploiting UAP in Software System

```
#include <regex.h>
... Your Program’s Code ...
regex_t regex;
int ret;
regcomp(&regex, ”^MICRO[0-9]{4}.*UAP", 0);
ret = regexec(&regex, “abcdefghijklmn”, 0, NULL, 0);
if (!ret) {
    puts("Match");
}
else if (ret == REG_NOMATCH) {
    puts("No match");
}
... Your Program’s Code ...
```

Exploiting UAP in Software System

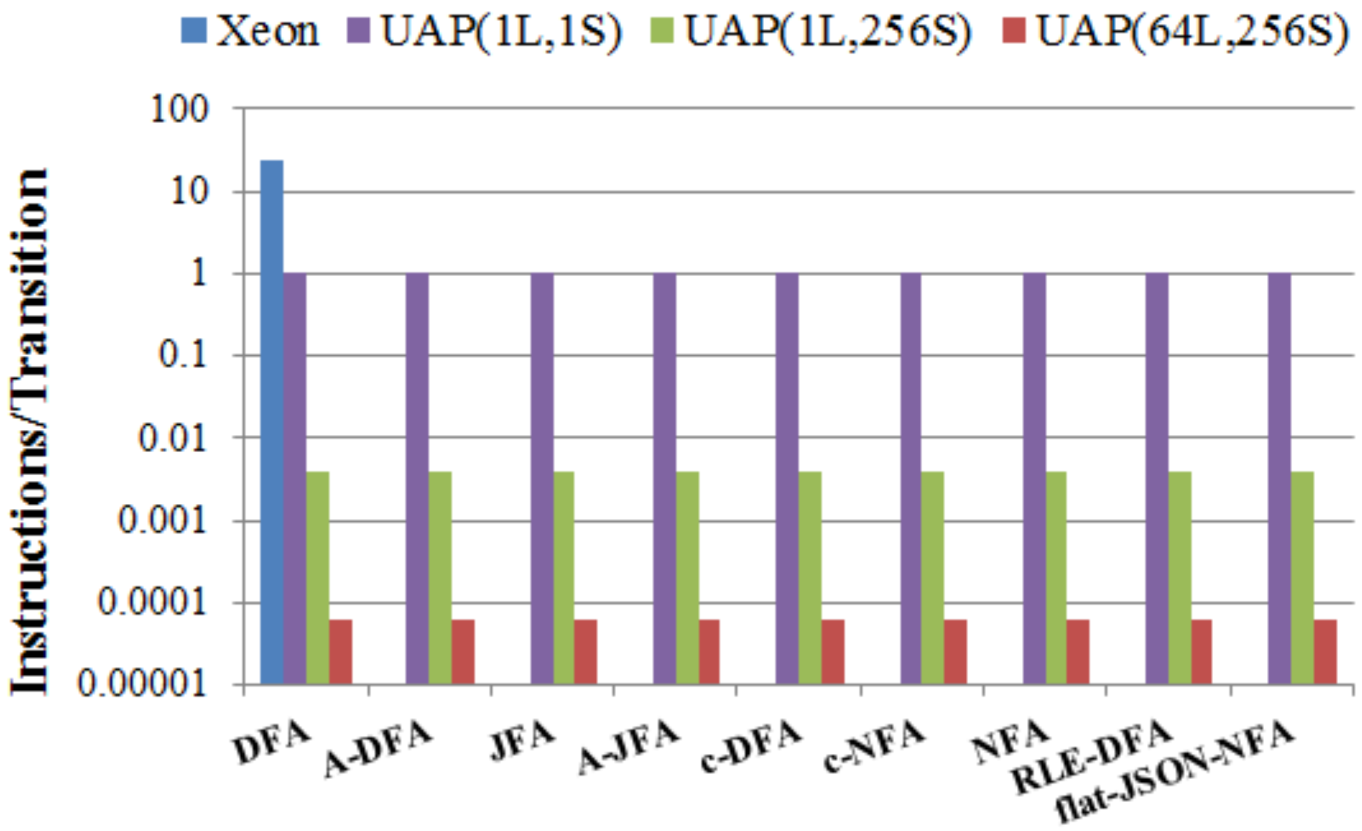
```
#include <regex.h>
... Your Program’s Code ...
regex_t regex;
int ret;
regcomp(&regex, ”^MICRO[0-9]{4}.*UAP", 0);
ret = regexec(&regex, “abcdefghijklmn”, 0, NULL, 0);
if (!ret) {
```

```
… load Compiled FA descriptions …
//  Initialize UAP, Clear acceptance flags
 write_state_regs(initial states, memory offsets)
 vec64 = 0;
// Looping consuming 256 symbols per iteration
while (all_zeros(vec64)){
     … load stream data into vector registers …
     traverse(0, 256, vec64);
     complete_traverse();
 }
```

Evaluation

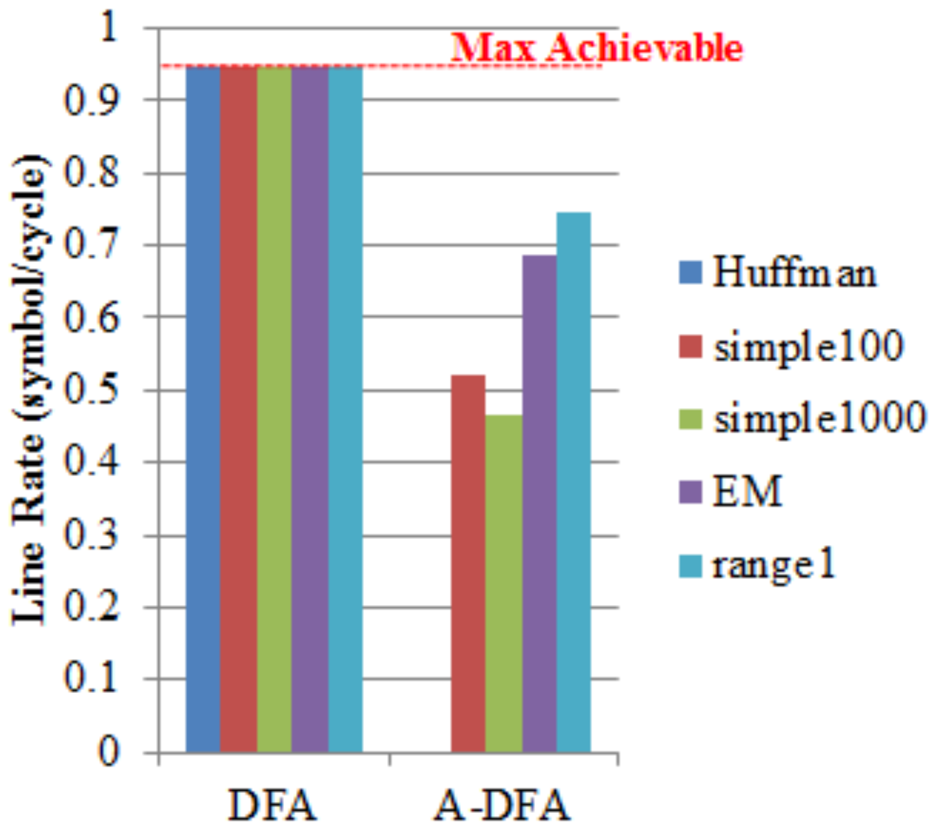
- Workload Datasets
  - Network Intrusion Detection (DFA, A-DFA, JFA, A-JFA, NFA)
  - Compression/Decompression (DFA, RLE-DFA)
  - Signal Triggering (c-DFA, c-NFA)
  - Database Query (flat-JSON-NFA)
- Simulation
  - Cycle-accurate Logic, Local Memory, Cache, DRAM
  - Detailed 32nm TSMC design, UAP runs @1.2GHz
  - Energy-accurate modeling of each element

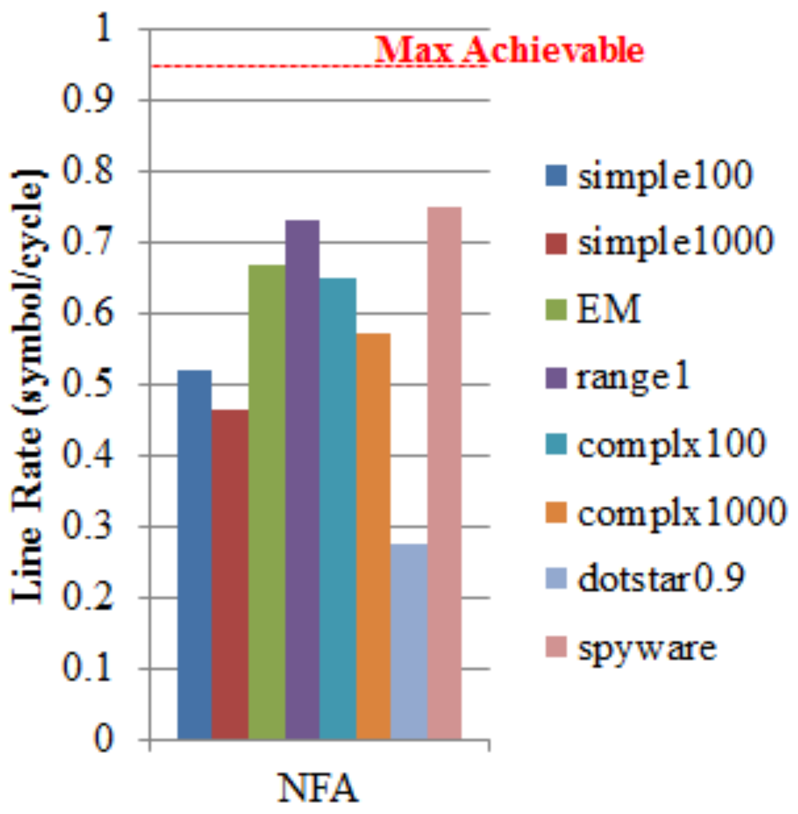
Instruction Count



Vector Parallelism and Multi-step Execution significantly reduce instruction counts for ALL FA MODELS

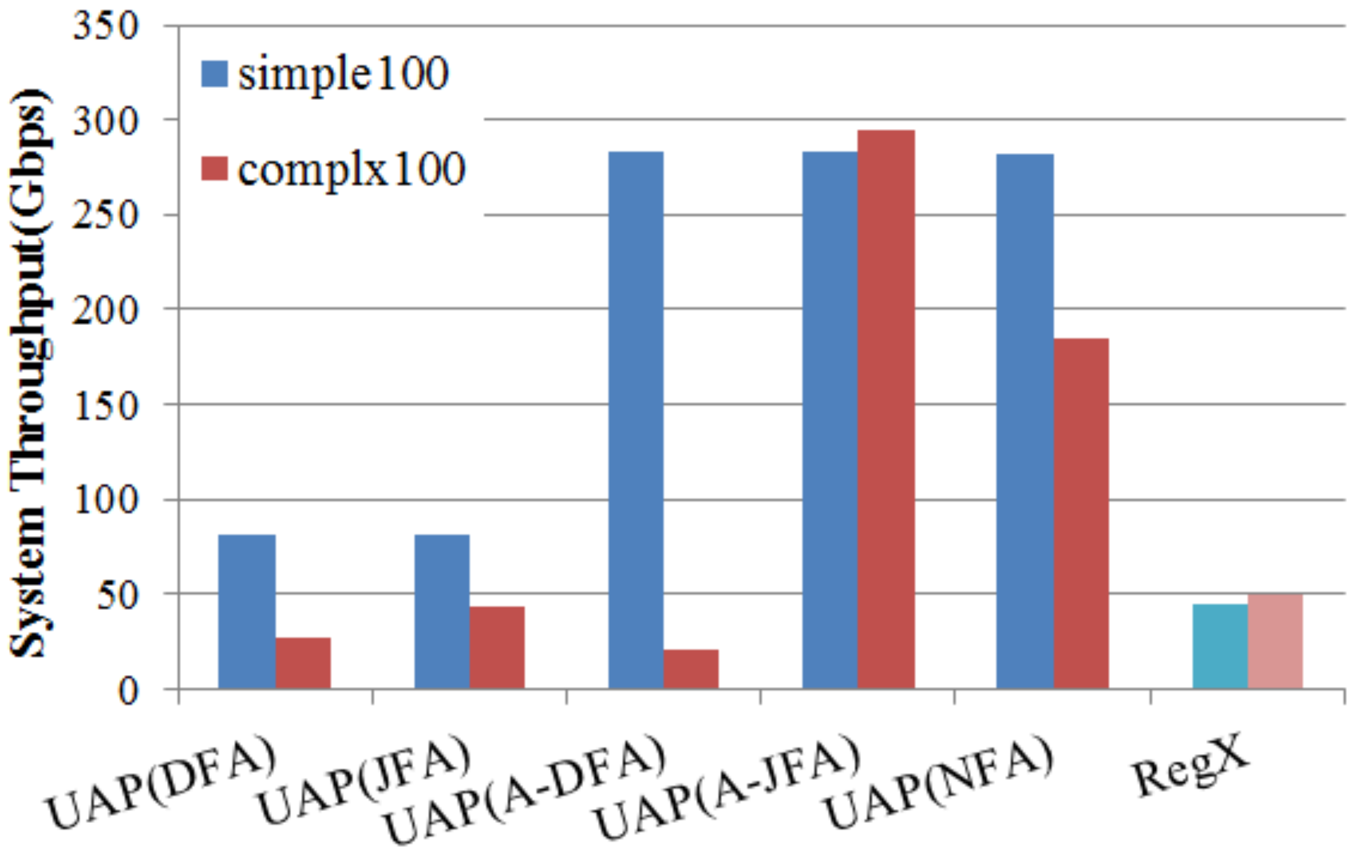
Line Rate (single stream performance)





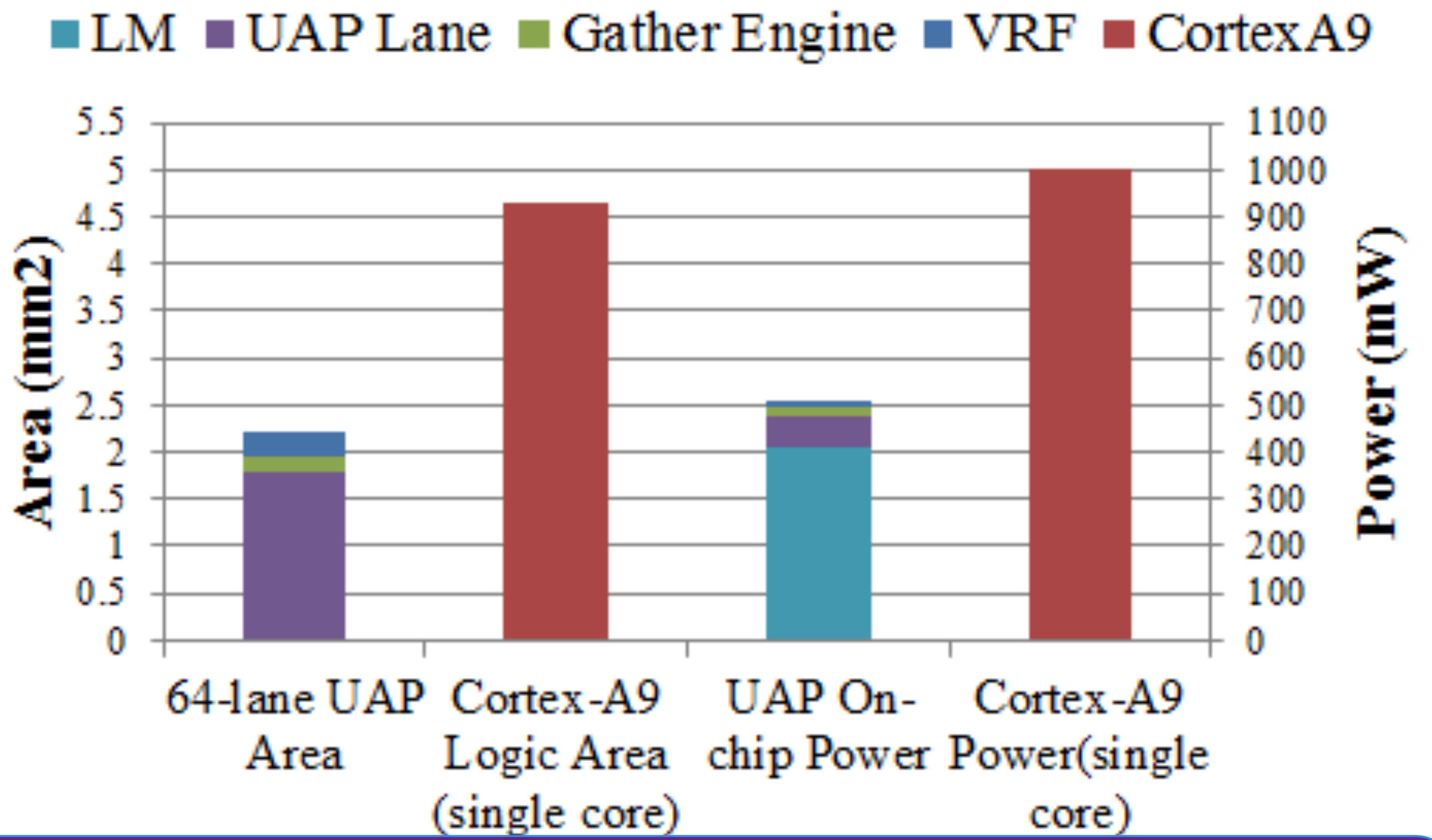
UAP@1.2GHz (DFA) ~ 9.1Gbps
227x/70x best published CPU/GPU result on realistic dataset!

Power of General Purpose FA Computation



General-purpose FA processing enables UAP to outperform IBM RegX by 5x and CPU/GPU by 100x .

Area and On-chip Power Breakdown



64-lane UAP 2.2 mm^2, 507 mW
Half size of a single Cortex A-9 core logic
Half power of a single Cortex A-9 core

Related Work

- Software Implementation
  - CPU: [Becchi@ANCS09, Mytkowicz@ASPLOS14, Zhao@ASPLOS15, Cameron@PACT14], best throughput 3Gbps (small datasets)
  - GPU: [Yu@CF13, Zu@PPoPP12], best throughput 13Gbps (small datasets)
- Customized Hardware
  - Single model
    - DFA: ASIC [Lunteren@MICRO12, Brodie@ISCA06, Tan@ISCA05], TCAM [Liu@USENIX Security10], best line rate 9.2 Gbps
    - NFA: MicronAP [Dlugosch@TPDS14], line rate 1 Gbps
  - Multiple models
    - FPGA [Yang@ToC12], line rate 1~4 Gbps
- UAP combines general purpose FA computing with top performance
  - DFA line rate 9.1Gbps
  - Significant throughput DFA (450Gbps), NFA (250 Gbps)

Summary

- UAP is a general-purpose FA processing architecture
- UAP increases performance by >100x vs CPUs and GPUs; Matches customized ASIC and FPGA implementations
- UAP is small and low-power, so it can be easily integrated across future computing systems
- UAP is easy to use. It makes software processing of unstructured data using FA's efficient and pervasively usable

Funding Acknowledgement

- DARPA HR0011-13-2-0014
- NSF CCF-1421765 and CNS-1319748
- Keysight Research
- Synopsys Academic program