RAPID: Accelerating Pattern Search Applications with Reconfigurable Hardware

Kevin Angstadt  Jack Wadden  Xiaoping Huang†
Mohamed El-Hadedy‡  Westley Weimer  Kevin Skadron

University of Virginia, †Northwestern Polytechnical University,
‡University of Illinois at Urbana-Champaign

{angstadt, wadden, weimer, skadron}@virginia.edu,
huangxp@nwpu.edu.cn, hadedy@illinois.edu
Finding Needles in a Haystack

- Researchers and companies are collecting increasing amounts of data
- 44x data production in 2020 than in 2009†
- Demand for real-time analysis of collected data‡

† Computer Sciences Corporation. Big data universe beginning to explode. 2012
‡ Capgemini. Big & fast data: The rise of insight-driven business. 2015.
What is the common theme?

- Locate the most probable location for a DNA fragment in the human genome
- Identify consumer sentiment based off of social media posts
- Find products that are most commonly purchased together
- Search for Higgs events based off on paths of subatomic particles

Pattern Search Problems
Parallel searches

At CGCGATATCGA...

Key

Active Searches

Target Pattern

Incoming Data

CGCGCAT

Kevin Angstadt
Parallel searches

Key

Active Searches

Target Pattern

Incoming Data

Active Search triggers report

G C T G A C C A T

ATCGA

CGGCAT

Kevin Angstadt
Parallel Searches: Goals

- Fast processing
- Concise, maintainable representation
- Efficient compilation
  - High throughput
  - Low compilation time

Specialized Hardware + VASim

RAPID Programming Language
Specialized Hardware

Micron Automata Processor
- Memory-derived hardware implementation of non-deterministic finite automata
- Accelerates identification of patterns in input data stream using massive parallelism

FPGAs
- Logic-based reconfigurable fabric of LUTs and Memory
- Allow custom implementation of applications for high-speed processing
A researcher should spend his or her time designing an algorithm to find the important data, not building a machine that will obey said algorithm.
The Remainder of this Talk

• Automata Processing
  – Current Programming Models
• RAPID Programming Language
  – Language Overview
• VASim: Virtual Automata Simulator
  – Synthesizable Verilog Generation
• Experimental Evaluation
• Conclusions and Future Directions
The Remainder of this Talk

• Automata Processing
  – Current Programming Models
• RAPID Programming Language
  – Language Overview
• VASim: Virtual Automata Simulator
  – Synthesizable Verilog Generation
• Experimental Evaluation
• Conclusions and Future Directions
Finite Automata

• Useful for filtering data based on patterns
• Equivalent in representative power to Regular Expressions

Start STE

Reporting STE

.*[Dd](o|ough)nut
Programming Challenges

• Finite automata development akin to *assembly programming*
  – Requires knowledge of automata theory **and** hardware properties
  – Tedious and error-prone development process
  – In many areas, specification of FA is automated!

• Regular expressions challenging to implement
  – Often exhaustive enumerations
  – Similarly error-prone (high rates of runtime exceptions)
The Remainder of this Talk

• Automata Processing
  – Current Programming Models

• RAPID Programming Language
  – Language Overview

• VASim: Virtual Automata Simulator
  – Synthesizable Verilog Generation

• Experimental Evaluation

• Conclusions and Future Directions
RAPID at a Glance

• Provides concise, clear, maintainable, and efficient representations for pattern-identification algorithms
• Conventional, C-style language
• Domain-specific parallel control structures
• Provides suitable data structures for pattern search problems
• Recursive algorithm to transform RAPID program into a finite automaton for execution
Example RAPID Program

Association Rule Mining
Identify items from a database that frequently occur together
Example RAPID Program

```cpp
macro frequent (String set, Counter cnt) {
    foreach(char c : set) {
        while(input() != c);
    }
    cnt.count();
}

network (String[] set) {
    some(String s : set) {
        Counter cnt;
        whenever(START_OF_INPUT == input())
            frequent(s, cnt);
        if (cnt > 128)
            report;
    }
}
```

Spawn parallel computation for each item set
Example RAPID Program

```
macro frequent (String set, Counter cnt) {
    foreach(char c : set) {
        while(input() != c);
    }
    cnt.count();
}

network (String[] set) {
    some(String s : set) {
        Counter cnt;
        whenever(START_OF_INPUT == input())
            frequent(s,cnt);
        if (cnt > 128)
            report;
    }
}
```

Spawn parallel computation for each item set

Sliding window search calls frequent on every input
Example RAPID Program

```cpp
macro frequent (String set, Counter cnt) {
    foreach(char c : set) {
        while(input() != c);
    }
    cnt.count();
}

network (String[] set) {
    some(String s : set) {
        Counter cnt;
        whenever(START_OF_INPUT == input())
            frequent(s,cnt);
        if (cnt > 128)
            report;
    }
}
```

- If all symbols in item set match, increment counter
- Spawn parallel computation for each item set
- Sliding window search calls `frequent` on every input

Kevin Angstadt
Example RAPID Program

```
macro frequent (String set, Counter cnt) {
    foreach(char c : set) {
        while(input() != c);
    }
    cnt.count();
}

network (String[] set) {
    some(String s : set) {
        Counter cnt;
        whenever(START_OF_INPUT == input())
            frequent(s, cnt);
        if (cnt > 128)
            report;
    }
}
```

- If all symbols in item set match, increment counter
- Spawn parallel computation for each item set
- Sliding window search calls `frequent` on every input
- Trigger `report` if threshold reached
System Overview

RAPID Program

RAPID Compiler

Driver Code

AP Binary

FPGA Engine

APcompile

ANML

Verilog

Xilinx PAR

VASim

Kevin Angstadt
The Remainder of this Talk

• Automata Processing
  – Current Programming Models
• RAPID Programming Language
  – Language Overview
• VASim: Virtual Automata Simulator
  – Synthesizable Verilog Generation
• Experimental Evaluation
• Conclusions and Future Directions
VASim: Open-Source Automata Processing Platform

• Standard platform for automata application and architecture research

• **Highly flexible**: Can be arbitrarily extended with hypothetical functionality

• **Common Algorithm Repository**: standard location for both old (DFA subset construction, prefix merging) and new (hybrid finite automata) automata optimizations

• **High-Performance**: on-par with industrial quality regex engines like RE2/HyperScan
Generating Verilog

- **Inputs:** clock, reset, 8-bit input symbol
- **Outputs:** report events

- Update activations every clock cycle
- State activations stored in registers
- Activate state if
  - State accepts input symbol
  - State with incident edge is active
The Remainder of this Talk

• Automata Processing
  – Current Programming Models
• RAPID Programming Language
  – Language Overview
• VASim: Virtual Automata Simulator
  – Synthesizable Verilog Generation
• Experimental Evaluation
• Conclusions and Future Directions
Parallel Searches: Goals

• Fast processing
• Concise, maintainable representation
• Efficient compilation
  – High throughput
  – Low compilation time

Specialized Hardware + VASim

RAPID Programming Language
Description of Benchmarks

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Description</th>
<th>Domain</th>
<th>Baseline Generation Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM</td>
<td>Association Rule Mining</td>
<td>ML</td>
<td>Meta Program</td>
</tr>
<tr>
<td>Brill</td>
<td>Brill Part of Speech Tagging</td>
<td>NLP</td>
<td>Meta Program</td>
</tr>
<tr>
<td>Exact</td>
<td>Exact DNA Alignment</td>
<td>Bioinformatics</td>
<td>ANML</td>
</tr>
<tr>
<td>Gappy</td>
<td>DNA Alignment with Gaps</td>
<td>Bioinformatics</td>
<td>ANML</td>
</tr>
<tr>
<td>MOTOMATA</td>
<td>Planted Motif Search</td>
<td>Bioinformatics</td>
<td>ANML</td>
</tr>
</tbody>
</table>
RAPID Lines of Code

Percent Reduction

- ARM
- Brill
- Exact
- Gappy
- MOTOMATA

Kevin Angstadt
Generated STEs (Automata Processor)

Percent Reduction

-40% -20% 0% 20% 40% 60% 80%

- ARM - Brill - Exact - Gappy - MOTOMATA
Generated LUTs and Registers (FPGA)

Xilinx Kintex UltraScale XCKU060

Kevin Angstadt
The Remainder of this Talk

• Automata Processing
  – Current Programming Models
• RAPID Programming Language
  – Language Overview
  – Code Generation
• VASim: Virtual Automata Simulator
  – Synthesizable Verilog Generation
• Experimental Evaluation
• Conclusions and Future Directions
Hardware-Agnostic RAPID

• Does RAPID provide true hardware-agnostic representation of pattern search?
• Full timing evaluation on FPGA
  – Implementation of custom reporting architecture
• Evaluation with CPU and GPGPU engines
Tech Transfer

• Industry collaborators
  – Contacts with Micron and Xilinx
  – Center for Automata Processing brings together researchers and industry experts

• Publications/presentations
  – Associated work presented at ASPLOS 2016 and Supercomputing Frontiers 2016
  – Weekly/semiweekly teleconferences with Micron and Xilinx to present research

• Tools will be released open source (BSD)
Conclusions

• RAPID is a **concise, maintainable, and efficient** high-level language for pattern-search algorithms

• VASim is an **extensible and general** framework for automata application and architecture research

• Combination of these tools allows for efficient execution using the Automata Processor, FPGAs

This work was supported in part by the Center for Future Architectures Research (C-FAR), one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.
Programming Challenges

• Implement **single instance** of a problem
  – Each instance of a problem requires a brand new design
  – Need for meta-programs to generate final design

• Current programming models place unnecessary burden on developer
Parallel searches

Maximize number of parallel active searches by reducing STE usage

Key

Incoming Data

ATCGA

Small footprint increases throughput

CGGCAT

Active Searches

Target Pattern

Kevin Angstadt