RAPID: Accelerating Pattern Search Applications with Reconfigurable Hardware

Kevin Angstadt, Jack Wadden, Xiaoping Huang, Mohamed El-Hededy, Wesley Weimer, Kevin Skadron
University of Virginia, Northwestern Polytechnical University, University of Illinois at Urbana-Champaign

Problem

- Many big data applications reduce to highly parallel pattern search problems against a single stream of data.
- Hardware accelerators, such as the Automata Processor and FPGAs, support these searches, but programming can be challenging.
- Goal: A concise, maintainable programming model that maximizes the number of parallel searches for higher throughput.

Background: Reconfigurable Hardware

Micron’s Automata Processor
- Memory-based hardware implementation of non-deterministic finite automata
- States stored in memory array
- Connections made with hierarchical, reconfigurable routing matrix
- Accelerates identification of patterns in data stream using massive parallelism

Field Programmable Gate Arrays
- Logic-based reconfigurable fabric of LUTs and Memory
- Allows custom implementation of applications for high-speed processing
- Use: Xilinx Kintex UltraScale XCKU060

Manipulating Automata: VASim

Static Thread Spawning
- Example: Search for ‘and’ or ‘or’ in parallel
  ```
  either {
    'a' == input();
    'g' == input();
    'w' == input();
    'h' == input();
  } 
  ```
Dynamic Thread Spawning
- Example: Search for all strings within a Hamming distance of 5 from any string in the dna array
  ```
  word(String s, int d) {
    hamming_distance(s,5);
  }
  ```

Parallel Control Structures

- Sliding Window Searches
- Example: Find the word ‘rapid’ anywhere in the input stream
  ```
  whenever ( ALL_INPUT == input() ) {
    for ( char c : " rapid ") {
      c == input();
      report;
    }
  }
  ```

Acknowledgments

We acknowledge the partial support of the NSF (CCF-1059424, CCF-116299, CNS-1124833), Air Force (FA8750-15-2-0075); Virginia Commonwealth Fellowship; Jefferson Scholars Foundation; the ARCS Foundation; and C-Falk, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA. We would like to thank Micron Technology for their support with programming the AP.

Our Approach: RAPID

- We developed RAPID, a high-level programming language for pattern-search algorithms
- C-style language with added domain-specific parallel control structures
- Suitable data structures for pattern search problems
- Synchronized access to input data stream across active computation
- RAPID programs are compiled to finite automata via a recursive algorithm
- Finite automata are then synthesized for reconfigurable hardware (e.g., AP, FPGA)

Manipulating Automata: VASim

- Open-source, highly-flexible platform for automata engine and architecture research
- Common repository for automata optimizations, transformations, and static and dynamic analyses
- Multi-threaded, high-performance CPU simulation core
- Emits synthesizable Verilog for executing automata on FPGAs
- Update activations every clock cycle (activations stored in registers)
- Activate state if: (1) the state accepts the input symbol and (2) a state with an incident edge is active

Conclusions

- RAPID reduces program text by 92%-98%
- RAPID programs are just as fast as their hand-crafted counterparts
- RAPID programs require no modifications to execute efficiently on the AP and FPGAs

Technology Transfer

- Associated work published and presented at ASPLOS 2016 and Supercomputing Frontiers 2016
- Weekly/monthly teleconferences with Micron and Xilinx to present research

Experimental Results

Benchmarks: Association Rule Mining (ARM), Brill POS Tagging, Exact DNA Alignment, Gappy DNA Alignment, Planted Motif Search (MOTOMATA)

Percent Reduction over Handcrafted

<table>
<thead>
<tr>
<th>Number of Resources</th>
<th>Handcrafted LUTs</th>
<th>Handcrafted Registers</th>
<th>RAPID LUTs</th>
<th>RAPID Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAPID Program</td>
<td>ARM</td>
<td>Exact</td>
<td>Gappy</td>
<td>Motomata</td>
</tr>
<tr>
<td>RAPID Compiler</td>
<td>ARM</td>
<td>Exact</td>
<td>Gappy</td>
<td>Motomata</td>
</tr>
<tr>
<td>ANML</td>
<td>ARM</td>
<td>Exact</td>
<td>Gappy</td>
<td>Motomata</td>
</tr>
</tbody>
</table>

Lines of Code

<table>
<thead>
<tr>
<th>Tool Pipeline</th>
<th>RAPID Program</th>
<th>RAPID Compiler</th>
<th>ANML</th>
<th>ARM</th>
<th>Exact</th>
<th>Gappy</th>
<th>Motomata</th>
</tr>
</thead>
</table>

Acknowledgments

Kevin Angstadt, Jack Wadden, Xiaoping Huang, Mohamed El-Hededy, Wesley Weimer, Kevin Skadron
University of Virginia, Northwestern Polytechnical University, University of Illinois at Urbana-Champaign

Acknowledgments

We acknowledge the partial support of the NSF (CCF-1059424, CCF-116299, CNS-1124833), Air Force (FA8750-15-2-0075); Virginia Commonwealth Fellowship; Jefferson Scholars Foundation; the ARCS Foundation; and C-Falk, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA. We would like to thank Micron Technology for their support with programming the AP.

Our Approach: RAPID

- We developed RAPID, a high-level programming language for pattern-search algorithms
- C-style language with added domain-specific parallel control structures
- Suitable data structures for pattern search problems
- Synchronized access to input data stream across active computation
- RAPID programs are compiled to finite automata via a recursive algorithm
- Finite automata are then synthesized for reconfigurable hardware (e.g., AP, FPGA)

Manipulating Automata: VASim

- Static Thread Spawning
- Example: Search for ‘and’ or ‘or’ in parallel
  ```
  either {
    'a' == input();
    'g' == input();
    'w' == input();
    'h' == input();
  } 
  ```
Dynamic Thread Spawning
- Example: Search for all strings within a Hamming distance of 5 from any string in the dna array
  ```
  word(String s, int d) {
    hamming_distance(s,5);
  }
  ```

Parallel Control Structures

- Sliding Window Searches
- Example: Find the word ‘rapid’ anywhere in the input stream
  ```
  whenever ( ALL_INPUT == input() ) {
    for ( char c : " rapid ") {
      c == input();
      report;
    }
  }
  ```

Acknowledgments

We acknowledge the partial support of the NSF (CCF-1059424, CCF-116299, CNS-1124833), Air Force (FA8750-15-2-0075); Virginia Commonwealth Fellowship; Jefferson Scholars Foundation; the ARCS Foundation; and C-Falk, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA. We would like to thank Micron Technology for their support with programming the AP.

Our Approach: RAPID

- We developed RAPID, a high-level programming language for pattern-search algorithms
- C-style language with added domain-specific parallel control structures
- Suitable data structures for pattern search problems
- Synchronized access to input data stream across active computation
- RAPID programs are compiled to finite automata via a recursive algorithm
- Finite automata are then synthesized for reconfigurable hardware (e.g., AP, FPGA)

Manipulating Automata: VASim

- Static Thread Spawning
- Example: Search for ‘and’ or ‘or’ in parallel
  ```
  either {
    'a' == input();
    'g' == input();
    'w' == input();
    'h' == input();
  } 
  ```
Dynamic Thread Spawning
- Example: Search for all strings within a Hamming distance of 5 from any string in the dna array
  ```
  word(String s, int d) {
    hamming_distance(s,5);
  }
  ```

Parallel Control Structures

- Sliding Window Searches
- Example: Find the word ‘rapid’ anywhere in the input stream
  ```
  whenever ( ALL_INPUT == input() ) {
    for ( char c : " rapid ") {
      c == input();
      report;
    }
  }
  ```

Acknowledgments

We acknowledge the partial support of the NSF (CCF-1059424, CCF-116299, CNS-1124833), Air Force (FA8750-15-2-0075); Virginia Commonwealth Fellowship; Jefferson Scholars Foundation; the ARCS Foundation; and C-Falk, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA. We would like to thank Micron Technology for their support with programming the AP.

Our Approach: RAPID

- We developed RAPID, a high-level programming language for pattern-search algorithms
- C-style language with added domain-specific parallel control structures
- Suitable data structures for pattern search problems
- Synchronized access to input data stream across active computation
- RAPID programs are compiled to finite automata via a recursive algorithm
- Finite automata are then synthesized for reconfigurable hardware (e.g., AP, FPGA)