Public Abstract

First Name:Xiang
Middle Name:
Last Name:Wang
Adviser's First Name:Michela
Adviser's Last Name:Becchi
Co-Adviser's First Name:
Co-Adviser's Last Name:

Graduation Term:SS 2014
Department:Computer Engineering
Degree:MS

Title:TECHNIQUES FOR EFFICIENT REGULAR EXPRESSION MATCHING ACROSS HARDWARE ARCHITECTURES

Regular expression matching is a central task for many networking and bioinformatics applications. For example, network intrusion detection systems, which perform deep packet inspection to detect malicious network activities, often encode signatures of malicious traffic through regular expressions. Similarly, several bioinformatics applications perform regular expression matching to find common patterns, called motifs, across multiple gene or protein sequences. Hardware implementations of regular expression matching engines fall into two categories: memory-based and logic-based solutions. In both cases, the design aims to maximize the processing throughput and minimize the resources requirements, either in terms of memory or of logic cells.

Graphical Processing Units (GPUs) offer a highly parallel platform for memory-based implementations, while Field Programmable Gate Arrays (FPGAs) support reconfigurable, logic-based solutions. In addition, Micron Technology has recently announced its Automata Processor, a memory-based, reprogrammable hardware device. From an algorithmic standpoint, regular expression matching engines are based on finite automata, either in their non-deterministic or in their deterministic form (NFA and DFA, respectively). Micron’s Automata Processor is based on a proprietary Automata Network, which extends classical NFA with counters and boolean elements.

In this work, we aim to implement highly parallel memory-based and logic-based regular expression matching solutions. Our contributions are summarized as follows. First, we implemented regular expression matching on GPU. In this process, we explored compression techniques and regular expression clustering algorithms to alleviate the memory pressure of DFA-based GPU implementations. Second, we developed a parser for Automata Networks defined through Micron’s Automata Network Markup Language (ANML), a XML-based high-level language designed to program the Automata Processor. Specifically, our ANML parser first maps the Automata Networks to an internal representation. We then apply NFA optimization techniques designed for other architectures to this internal representation. Finally, we implemented a tool to convert our internal representation to Verilog, thus allowing automatic deployment on FPGA. Our toolchain allows the user to apply existing optimization techniques to Micron’s Automata Processor and to directly compare this new platform with FPGA-based solutions.