Automated Incremental Design of Flexible Intrusion Detection Systems on FPGAs[1]

Zachary K. Baker and Viktor K. Prasanna
University of Southern California, Los Angeles, CA, USA
zbaker@halcyon.usc.edu, prasanna@ganges.usc.edu

Abstract

Intrusion detection for network security is a compute-intensive application demanding high system performance. This paper presents a variety of strategies we have developed for the automatic synthesis of highly efficient intrusion detection systems. We create FPGA architectures using a high-level, graph-based partitioning methodology. We provide a library of performance-customized architectures, which, through more efficient communication and extensive reuse of hardware components, provide dramatic increases in area-time performance. This paper addresses a problem of earlier designs, the requirement for complete place-and-route for small changes to the pattern database, through an optimized incremental design strategy.

1 Introduction

The continued discovery of programming errors in network-attached software has driven the introduction of increasingly powerful and devastating attacks [3, 7]. Attacks can cause destruction of data, clogging of network links, and future breaches in security. In order to prevent, or at least mitigate, these attacks, a network administratori can place a firewall or Intrusion Detection System at a network choke-point such as a company's connection to a trunk line (Figure 1). A firewall's function is to filter at the header level; if a connection is attempted to a disallowed port, such as FTP, the connection is refused. This catches many obvious attacks, but in order to detect more subtle attacks, an Intrusion Detection System (IDS) is utilized. The IDS differs from a firewall in that it goes beyond the header, actually searching the packet contents for various patterns that imply an attack is taking place, or that some disallowed content is being transferred across the network. Current IDS pattern databases reach into the thousands of patterns, providing for a difficult computational task.

[1]Supported by the United States National Science Foundation/ITR under award No. ACI-0325409 and in part by an equipment grant from the HP Corporation.

Methods commonly used to protect against security breaches include firewalls with filtering mechanisms to screen out obviously dangerous packets, and intrusion detection systems which use much more sophisticated rules and pattern matching to sense potential malicious packets. These techniques require significant computational resources. However, using automated design strategies for highly-parallel adaptive soft processors, there is potential for dramatic performance improvements. FPGAs provide an attractive platform for hardware implementation of intrusion detection because of the dynamic nature of the ruleset – as new vulnerabilities and attacks are identified, new rules must be added to the database and the device configuration must be regenerated.

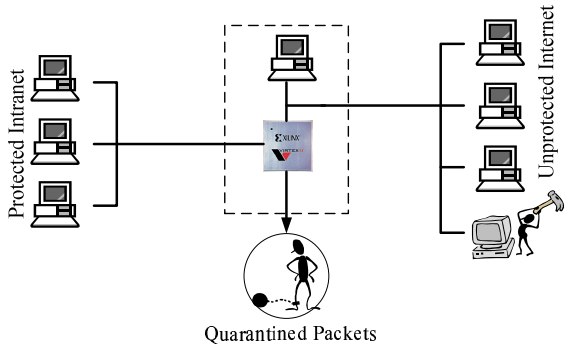

Figure 1. Intrusion detection systems protect networks from external threats. The use of FPGA allows a system to take advantage of massive parallelism in this is a highly computation-intensive task

This paper describes our work in creating Intrusion Detection Systems with *customized performance*, allowing a designer to mix and match from a collection of process steps and a family of architectures we have developed. Some of our results have already been published in [1].

Our basic architecture is a pre-decoded multiple-pipeline shift-and-compare matcher. While this ap-

proach can be considered "brute force" compared to a state machine approach [2, 4, 6] or a hashing approach [5], the simplicity of the units allows for exceptional area and time performance. The basic architecture, as described in detail below, reduces device routing and comparator size by converting incoming characters into many bit lines, each representing the presence of single character.

This basic architecture is extended in various ways. To allow for better area performance, we present a partial tree architecture that allows for significant reduction in redundant comparisons by independently matching prefixes that are shared across a range of patterns. To provide increased throughput performance, we provide a design that replicates a fraction of the hardware to allow for exact matching for k bytes per cycle. To provide high throughput with exceptional area efficiency, we provide an architecture that sacrifices exactness and allows for an increased false positive rate.

The architectures we have developed are only part of the contributions of this paper. To achieve better utilization of these architectures, system-level preprocessing steps are required, serving various functions including partitioning, grouping, and code generation. These steps, by considering the entire set of patterns in lieu of naïve hardware generation.

By intelligently processing an entire ruleset, our tool partitions the pattern collection into multiple pipelines in order to optimize the area and time characteristics of the system. The rule database is first converted into a graph representing the similarity of the ruleset. Depending on the flow, the graph edges are weighted to provide higher connectedness between rules with particular types of similar characters; this allows for increased grouping of prefixes as well as general shared-character grouping. The graph is partitioned based on the weighted graph and then sent to the partitioning routines, which act to reduce the interconnect burden in a given pipeline. Prefixes are then grouped for the tree architecture, if required. Based on this pre-processing, the system is generated from templates. By applying automated graph theory and trie techniques to the problem, the tool more effectively optimizes large ruleset as compared to naïve approaches.

2 Optimized Incremental Design

A problem with recent designs utilizing hard-wired comparator modules is in the requirement for a full place-and-route to make any change, no matter how small, to the design. Because of the exceptional area and time efficiency possible with this customized design paradigm, this issue has been largely ignored.

A portion of this paper covers our solution to the place-and-route problem. For the situation of adding a rule, we utilize the min-cut partitioned graph produced for the initial design. Determining the optimal partition to add a new pattern to is a fairly trivial task, only requiring a consideration of characters already mapped to the partition and pre-existing prefixes. The partition least modified by the addition of the new rule is determined by comparing the pre-decoded bits already within the partition, as well as the potential for using previously mapped prefixes. This VHDL code describing this partition is then modified by the tool. If the new pattern shares a prefix with some other pattern in the partition, the partial result of the previous pattern is mapped to the new pattern, reducing new wiring. The removal of rules is far easier, only the connections to the final result tree are removed. The new partition code is sent to the incremental synthesis and place-and-route functions of Xilinx ISE 6.2. The tool only re-synthesizes the modified modules. Because of the previously defined area constraints, each pipline module is independent of the others. Thus, only the routing in the modified module requires place and route.

Our initial results show that for a change of one pattern in a single partition in system with p partitions, the time for place-and-route is reduced to $1/p$ plus some overhead for reprocessing the guide files. This overhead can be fairly large (approaching 50% of the total PAR time). However,without the use of incremental place and route, the system would require a completely new place-and-route, or p times additional time.

References

[1] Z. K. Baker and V. K. Prasanna. A Methodology for the Synthesis of Effi cient Intrusion Detection Systems on FPGAs. In *The Twelfth Annual IEEE Symposium on Field Programmable Custom Computing Machines 2004 (FCCM '04)*, 2004.

[2] Z. K. Baker and V. K. Prasanna. Time and Area Effi - cient Pattern Matching on FPGAs. In *The Twelfth Annual ACM International Symposium on Field-Programmable Gate Arrays (FPGA '04)*, 2004.

[3] Z. Chen, L. Gao, and K. Kwiat. Modeling the Spread of Active Worms. *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*, April 2003.

[4] C. R. Clark and D. E. Schimmel. Effi cient Reconfi gurable Logic Circuits for Matching Complex Network Intrusion Detection Patterns. In *Proceedings of FPL '03*, 2003.

[5] S. Dharmapurikar, P. Krishnamurthy, T. Sproull, and J. Lockwood. Implementation of a Deep Packet Inspection Circuit using Parallel Bloom Filters in Reconfi gurable Hardware. In *Proceedings of HOTi '03*, 2003.

[6] B. L. Hutchings, R. Franklin, and D. Carver. Assisting Network Intrusion Detection with Reconfi gurable Hardware. In *Proceedings of FCCM '02*, 2002.

[7] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the Slammer Worm. *IEEE Security & Privacy Magazine*, 1(4), July-Aug 2003.