Microprocessors and Microsystems 37 (2013) 1200–1207

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Multi-character cost-effective and high throughput architecture for content scanning [☆]

José M. Bande [a,*], José Hernández Palancar [a], René Cumplido [b]

[a] Advanced Technologies Application Center, 7ma A #21406 e/ 214 y 216, CP: 12200 Havana, Cuba
[b] Instituto Nacional de Astrofísica Optica y Electrónica, Luis E. Erro 1, Sta. Ma. Tonanzintla, Puebla 72840, Mexico

ARTICLE INFO

Article history:
Available online 23 August 2013

Keywords:
NIDS
String matching
Content scanning
FPGA
Unique substrings

ABSTRACT

String matching is a time and resource consuming operation that lies at the core of Network Intrusion Detection Systems. In this paper a method and corresponding hardware architecture for string matching is presented. The proposed method is composed of two main steps. The first step performs a pre-detection of signatures alignment, and in the second step the alignment is corrected and the signatures are detected by a matcher. The compact and efficient architecture is designed to share resources among several modules that perform the detection and correction step needed for the string matching. Implementation results in a FPGA Virtex5 device show that the proposed architecture can perform string matching with a database with more than 400 K characters. And is also capable of achieving speeds of more than 30Gbps, which is much higher that previous works reported in the literature.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

Current data transmission technologies are capable of achieving multi-gigabit rates. For instance, recent standard for optical wire technologies OC-768 [1] present a data transmission rate of 40Gbps. In Network Intrusion Detections Systems (NIDS) one of the main tasks is the scanning of the packets content. Thousands of strings declared as signatures of malicious content must be detected in order to identify imminent attacks. And the speed of the data flow on which this string matching operation should be performed is extremely high. In addition, lowering such speed is not a valid option since this would affect the quality of the service provided by the network.

In the most demanding environments, current sequential machine technologies are unable to meet these requirements [2]. Consider the hypothetical situation where a General Purpose Processors (GPP) could process a single character per clock cycle. It would need to be working at 5 GHz in order to achieve 40 Gbps. Obviously, this is not possible with current technologies. This technological barrier dominated by the operating frequency of the sequential devices can be outperformed with an intensive use of parallelism. Here is where the Field Programmable Gates Arrays devices (FPGAs), have played an important role for NIDS in the last decades. The most important advantages of these devices over traditional GPPs are the reconfigurability and the fine grained parallelism they can provide. Reconfigurability, allows updating the signatures set any number of times, and with the use of parallelism, the signatures can be concurrently detected, while one or more than one character is processed at each clock cycle.

Hardware cost reduction and throughput increment, are key issues in string matching through hardware; being the hardware cost reduction the more widely researched. The reason is that augmenting throughput elevates the hardware cost. Therefore, there will be fewer resources available to detect more strings. That is why the most used practice has been to obtain a low cost architecture and replicate it in order to obtain more throughput.

The throughput of a data flow processing hardware architecture is defined as the amount of bits processed per time unit. Basically, there are three strategies to increase the throughput. The first strategy is focused on the frequency, while the other two concern the amount of data processed. Achieving a high operational frequency depends on several factors such as: the complexity of the placed logic, the critical path signals, and the underlying device technology. Well know techniques as pipelining can be applied in order to reduce timing at the cost of introducing latency in the architecture.

The second strategy is works well when the scanned data flow is divided in streams of data chunks or packets. The data packets are distributed into several identical string matching units working in parallel. The overall throughput is the sum of the throughput provided by each unit. This is commonly called aggregated throughput. Due to the packetized nature of some data flows, this form of processing is not very efficient. This is because it is possible

[☆] This document is a collaborative effort.
[*] Corresponding author.
E-mail addresses: jbande@cenatav.co.cu (J.M. Bande), jpalancar@cenatav.co.cu (J. Hernández Palancar), rcumplido@inaoep.mx (R. Cumplido).

0141-9331/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.micpro.2013.08.001

to have some units with no data to process at all, while there are still data waiting for being processed in other units. This strategy is easy to implement by just replicating as many processing units as possible.

The third strategy consists on augmenting the amount of data symbols processed from the data flow at each clock period. Architectures with this feature are commonly called multi-character architecture. This form is more efficient as it does not underutilize hardware resources and it can be used on either, packetized or continuous data streams. On the other hand, when several characters are processed at each clock cycle, the signatures may appear unaligned regarding the input. The simple approach to face this misalignment problem is a replication of the matching logic linear to the amount of character processed per clock cycle.

The last two naïve strategies to increase the throughput requires resources to invest in hardware replication. In this work we propose a not naïve multi-character approach which reduces considerably the hardware replication. Our architecture solves the misalignment problem in real time. The hardware consumption presented is lower than that obtained in naïve strategies for multi-character architectures. The implementation for four characters per cycle utilizes 63% of a FPGA Virtex5 fx100t device. For a signature set counting more than 84,000 characters the architecture achieves a throughput of 6.4 Gbps. Our strategy is to invest resources in a cost-effective alignment pre-detection and correction phase. In doing so, resources sharing are possible and the replication of the hardware is reduced.

In this work we take leverage of the fact that in a set of strings, each string may have a substring which is unique regarding the rest of the members in the set. The length of the unique substring may be equal or lower than its substring; in case of being equal, the unique substring is the string itself. For the set of signatures included in the most popular Network Intrusion Detection System, Snort, the length of unique substrings tend to be shorter than the length of the container signatures.

The proposed architecture consists of two processing steps. In the first step, the signature unique substrings are matched obtaining the alignment of the candidate signature. A candidate signature indicates a signature likely to exists in the data flow. In the second step the inputs of signature matchers are aligned in correspondence with the information provided by the first step. Since the signature matchers inputs will be aligned with the candidate signature in real time, no hardware replication will be needed in the second step.

We present a method and corresponding hardware architecture for string matching. A previous work that addresses a similar problem was published in [3]. This described an architecture that detects and corrects the signature alignment in the data flow through a unique substring predetection. This work presents a completely new architecture aimed at obtaining a significant reduction in area requirements when compared to the previous reported work. The main contributions of this work are twofold:

- Reduction of area resources of around 40% when compared against the naïve approach for multi-character architecture.
- A new partitioning criterion named security threshold which allows resources sharing and the elimination of ambiguities in matching process due to the misalignment problem.

The rest of the paper is as follows. In Section 2 we expose the works related with special emphasis in multi-character architectures in both, naïve and not naïve approaches. In Section 3, the central ideas and our partitioning scheme are explained in detail. In Section 4, the architecture is presented as well as its functioning. The results of the implementation and tests are discussed in Section 5. Conclusions and future work will be exposed in Section 6.

2. Related work

In general, the most part of hardware-based solutions for string matching fall into the following categories. Brute Force comparators (BF) [4,5,7], Automata based architectures with the Aho-Corassick automaton (AC) as the most implemented [2,3,8,11–13,20]. Content Addressable Memory based (CAM) architectures [6], and Hash based architectures [10]. Naturally, there are combinations of them [22,24,17–19].

Naïve multi-character architectures has been proposed in [4–9]. Sourdis et al. [4] proposed pipelines of discrete characters comparators in order to match an entire string. Their four-character input version was capable of achieving more than 10 Gbps. In a shift-and-compare pipeline, each character line feeds a shift register, by selecting the proper offsets, the character lines are “ANDed” to obtain a final match for a corresponding string. Sourdis et al. [5] proposed a shift-and-compare architecture using SLR16 shift register, while Baker and Prassana in [7] proposed partitioning scheme that allows resource sharing. Sung et al. [6] proposed an algorithm for a CAM memory-based architecture that processes four bytes per cycle, achieving more than 10 Gbps. Clarck and Schiemmel [8] perform a deep analysis of the hardware cost when extending the architecture for processing more bytes per clock cycle. They proposed a NFA logic-based architecture, where they can reach the impressive throughput of 99 Gbps. However, due to the high hardware cost introduced they just match up to 250 characters. Hardware implementation of the shift-or algorithm is proposed in [9]. Processing four character per cycle they achieve 16 Gbps for a 1500 character set.

In [10] the double port feature of modern embedded memory blocks are employed to process 2 bytes per cycle. Their architecture uses hashing in a first step to identify a possible match. Then, in a second step, the string is fetched from memory and is compared one-to-one with the characters in the data flow. Yang et al. [11] present a multi-character logic-based regular expression matching architecture. They propose an algorithm for extending regular expression to multi-character. Additionally, they propose the implementation of character classes operators using embedded RAM. Similarly, in [12] static strings which are part of regular expressions are detected by a classical AC using an off-chip memory, then logic-based NFAs match the regular expression metacharacters. A memory-based multi-character regular expression matching architecture is proposed by Brodie et al. in [13]. They define the concept of Equivalent Class Index, ECI. An ECI represents multiples sequences of characters sharing the same states in a multi-character-extended NFA. Their architecture first encode the data flow into ECI characters, then it uses an state machine to implement the NFA. In addition, they apply several optimizations and codifications for reducing the amount of memory required per NFA state. Yiang et al. [2] Proposed a pipelined and multi-character AC automaton where failure transitions are not needed since each level of the AC trie has its own hardware. A similar approach is proposed in [14], where the firsts states of the AC automaton are implemented using a pipeline of binary search trees.

A not naïve multi-character memory-based architecture is proposed by Cho et al. in [15]. They use the string prefix for pre-detection and alignment correction. Since several strings may have the same prefix, the string set must be partitioned so that each string in a subset possesses a unique prefix. The same strategy is used by in Chang et al. [16] with the difference that instead of memories, and brute force matching, they use logic-based NFA matchers. Serrano et al. [3] proposed the use of unique substrings instead of unique prefixes, since unique substrings tend to be shorter and better suited as partitioning criterion.