Prefix-based Multi-Pattern Matching on FPGA

Hoang-Gia Vu
Department of Radio-Electronics Engineering
Le Quy Don Technical University
Hanoi, Vietnam
Email: giavh@lqdtu.edu.vn

Yen Hoang Thi
Department of Radio-Electronics Engineering
Le Quy Don Technical University
Hanoi, Vietnam
Email: hoangyenmta@gmail.com

***Abstract*—Multi-pattern matching refers to the search for multiple patterns in a given text at the same time. This matching on FPGA is expected to scale with the number of patterns in hardware consumption. In this paper, we propose a matching architecture that compares the prefixes of multiple patterns with the prefix of the matching window in parallel. The comparison will continue with the body of each pattern if the corresponding prefix is matched. This architecture is called the prefix-based multi-pattern matching architecture. Our implementation on FPGA shows that the proposed matching architecture achieves much higher performance than the implementation on CPU, while the hardware cost is low.**

***Keywords*-multi-pattern matching; FPGA; prefix-based;**

I. Introduction

Multi-pattern matching is an important task in data mining. The multi-pattern matching algorithm is employed to search multiple string patterns in a target text to find the corresponding position of each pattern in the text. This algorithm requires many comparisons for each matching window, thus achieving low performance in sequential-execution processors. Several works proposed algorithms for multi-pattern matching on CPU, such as hash-based multi-pattern matching [1] and multi-string searching based on improved prefix tree [2]. However, these algorithms also require many comparisons, and the execution times also scale with the number of patterns.

We believe that multi-pattern matching can be accelerated on FPGA by executing comparisons in parallel. However, executing all comparisons in hardware consumes a large number of hardware resources when the number of patterns and the length of patterns increase. Yuichiro Utan [3] implemented comparisons on FPGA, but for approximate regular expression matching. Tomas Fukac [4] used a hash-based pre-filter on FPGA to reduce the input traffic before matching in CPU. In this work, we implement the whole exact multi-pattern matching on FPGA. We propose to compare only the prefix of each pattern with the prefix of the matching window. These comparisons are executed in parallel. If a comparison is matched, the rest of the pattern will be compared with the rest of the matching window. Otherwise, the matching window will be shifted to a new string for the next matching.

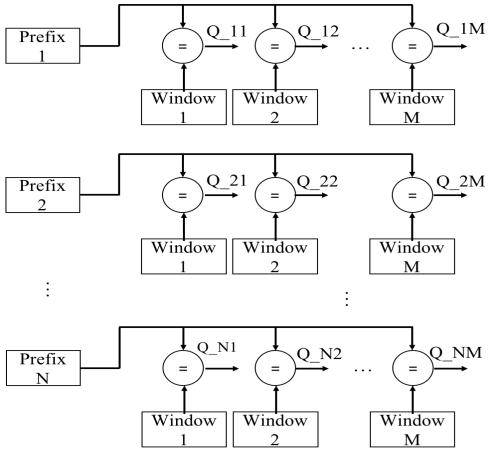

Figure 1. Prefix matching architecture

Our main contributions in this work are the hardware architecture for multi-pattern matching on FPGA and the analysis of its hardware consumption and latency. We also present the experimental results and address future work.

II. Multi-pattern Matching Architecture

The multi-pattern matching architecture is divided into two parts. The first part is called the prefix matching part. The second part is the body matching part. Before presenting the two parts, we define several concepts as follows:

- N is the number of patterns.
- *Prefix* of a string is a set of the first several characters in the string. k is defined as the length of the prefix.
- *Pattern body* is the rest of a pattern after its prefix.
- *Matching window* is the text window in the target text that is matched with patterns. The matching window will slide from the beginning to the end of the text. We assume that all the patterns have the same length of L characters.
- *Prefix window* is the set of the first characters in the matching window that has the same length as the prefixes - k characters.
- *Body window* is the rest of the matching window after the prefix window. The length of the body window is L-k.
- *Input bandwidth* is the number of characters coming per clock cycles, defined as M.

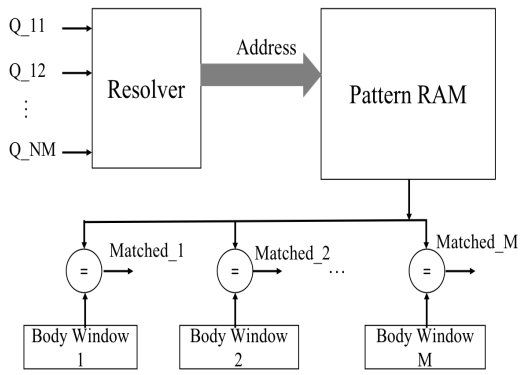

Figure 2. Body matching architecture

A. Prefix Matching Part

The prefix matching part includes circuits comparing each prefix and prefix windows. To meet the input bandwidth of M characters, each prefix should be compared with M prefix windows 1, 2, ..., M. It is noted that prefix window i+1 is prefix window i sliding one character in the target text. The outputs of these comparators lead to a resolver to identify which pattern is accessed.

B. Body Matching Part

The body matching part consists of a resolver, a pattern RAM, and comparators. The resolver is a sequential circuit that has its output as the input address of the pattern RAM. The pattern RAM is a one-port RAM storing N pattern bodies. The output of the pattern RAM is compared with M body windows of the target text. These M body windows are corresponding to M prefix windows in the prefix matching part. A pair of a prefix window and a body window forms a matching window.

III. Analysis

A. Hardware Cost Analysis

Assume that a 2-character comparator consumes c look-up tables (LUTs). Then a comparator in the prefix matching part consumes kc LUTs since a prefix has a k-character length. There are NM comparators in this part. Therefore, this part consumes $NMkc$ LUTs. In the body matching part, a comparator consumes $(L-k)c$ LUTs. There are M comparators in this part. Therefore, this part consumes $M(L-k)c$ LUTs. Totally, the multi-pattern matching architecture consumes $NMkc + M(L-k)c$ LUTs. This formula is reduced to $Mc((L-k) + Nk)$ LUTs. As can be seen from the formula, the number of LUTs depends linearly on the number of patterns N. However, we can choose k small to alleviate the impact of the number of patterns on LUT consumption.

B. Latency Analysis

Since M matching windows are matched simultaneously with patterns, the matching throughput is the same as the input bandwidth (M characters per clock cycle). In case that a prefix window and a prefix are matched, the matching window is stopped sliding for 3 clock cycles. The 3-clock-cycle latency is caused by 2-clock-cycle latency of the resolver and 1-clock-cycle latency of the pattern RAM and comparators.

Table I
Execution time for different platforms

Platform	10 KB	100 KB	1 MB	10 MB	100 MB
Core i5	0.00393	0.02531	0.20580	1.90277	18.95482
Cortex-A9	0.01998	0.19713	3.34643	36.47776	367.99406
FPGA	0.00004	0.00026	0.00263	0.02628	0.26292

IV. Evaluation

We have designed the multi-pattern matching hardware presented in this paper with Verilog HDL and implemented on Xilinx Zedboard using Vivado 2016.4. The matching hardware is combined with 3 direct memory access units (DMAs) and a thread-control unit to form an AXI4-based IP core. In this evaluation, we chose N = 16, k = 4, L = 36, M = 8. The hardware consumption of the IP core consists of 3489 slice registers, 4537 LUTs as logic, 391 LUTs as memory, 3.5 Block RAM tiles. The maximum clock frequency is 160 MHz. The execution time (in seconds) of the matching task when it runs on different platforms, such as CPU Intel Core i5, CPU ARM Cortex-A9, and FPGA (Zedboard), is showed in Table 1. The table shows that multi-pattern matching can achieve much higher performance on FPGA while scaling the text size, compared with the implementation on CPU.

V. Conclusion

We have proposed a prefix-based multi-pattern matching architecture on FPGA. Compared with the software implementation, the hardware architecture shows a much higher efficiency in execution time, while its hardware cost is low. In future work, we will evaluate the architecture while scaling the number of patterns and the length of patterns.

References

[1] Y. Zhou and C. Gao, *Research and Improvement of A Multi-pattern Matching Algorithm Based on Double Hash*, 3rd Int' conf on Computer and Communications, pp. 1772-1776, 2017.

[2] Y. Cheng and T. Zhang, *Design of Fast Multiple String Searching Based on Improved Prefix Tree*, 3rd Int' Conf on Knowledge Discovery and Data Mining, pp. 111-114, 2010.

[3] Y. Utan, Sh. Wakabayashi, and Sh. Nagayama, *An FPGA-based Text Search Engine for Approximate Regular Expression Matching*, 2010 International Conference on Field-Programmable Technology, pp. 184-191, 2010.

[4] T. Fukac and J. Korenek, *Hash-based Pattern Matching for High Speed Networks*, 22nd International Symposium on Design and Diagnostics of Electronic Circuits and Systems, 2019.