# AUTOMATED EFFICIENT METHOD FOR STRING SET MATCHING USING FPGA

**S.M. Vidanagamachchi[1], S.D. Dewasurendra[1], R.G. Ragel[1] and M. Niranjan[2]**

[1] *Department of Computer Engineering, Faculty of Engineering, University of Peradeniya*

[2]*School of Electronics and Computer Science, University of Southampton, UK*

## Introduction

String matching is a classical problem and it is fundamental to many applications that need processing of text data or some sequence data. String matching has been widely studied in the past three decades. Several string matching algorithms are used to find and locate one or several string patterns that are found within a larger string/text. String matching algorithms are used not only in applications such as text editors, word processing and bibliographic search, but also in comparing biological sequences in bioinformatics with the remarkable increase in the number of DNA and protein sequences been identified.

The objectives of this paper are to give a brief introduction of string matching algorithms with Field Programmable Gate Arrays (FPGA) and to describe the process of automating the implementation of string matching algorithms on an FPGA.

In computational biology exact string matching is commonly required. Pair wise sequence matching, multiple sequence matching, global alignment and local alignment are some types of string matching which are used in bioinformatics and computational biology. Needleman-Wunsch global alignment algorithm and Smith Waterman's local alignment algorithm use dynamic programming to obtain optimal alignments between a pair of sequences. The exponential growth of biological data resulted in these algorithms being infeasible, therefore the need to move into heuristic algorithms such as the one used in BLAST.

String matching algorithms can be basically classified into three groups; single pattern matching algorithms (search a single pattern within the text), algorithms that uses finite number of patterns (search finite number of patterns within the text) and algorithms that uses infinite number of patterns (search infinite number of patterns within the text such as regular expressions). Single pattern matching algorithms include Rabin-Karp, finite state automation based search, Knuth-Morris-Pratt and Boyer- Moore. Rabin-Karp algorithm can be used for both single pattern matching and for matching a finite number of patterns. For Rabin-Karp, in a string of length n, if p patterns of combined length m are to be matched, the best and worst case running time are O (n+m) and O (nm), respectively. Finite state automata based search is expensive to construct (power-set construction), but very easy to use. Knuth-Morris-Pratt (KMP) algorithm turns the search string into a finite state machine, and then runs the machine with the string to be searched as the input string. Running time of this algorithm is O (n+m) (Akl et al., 2006). Algorithms used for finite number of patterns (multi patterns) include Aho-Corasick, Commentz-Water, Rabin-Karp and Wu-Manber. Aho-Corasick is a dictionary matching algorithm and matches all patterns at once. Aho-Corasick algorithm has asymptotic worst-time complexity O (n+m) in space O (m).

FPGA technology is an emerging technology for providing better hardware solutions for sequence comparison, protein/molecular structure comparison and large scale clustering than other Programmable Logic Devices (PLDs). While maintaining its performance in par with other hardware solutions (such as ASICs), it is programmable and reliable.

**Problem Definition**

During last few decades, advances in life sciences have lead to the generation of a huge amount of biological data. Therefore there is a pressing need for efficient computational methods to cope with them. However a significant bottleneck exists in the analysis of such data. According to the predictions of Moore's Law the number of transistors that can be placed on an integrated circuit has doubled approximately every two years, but computational demand for analyzing huge amount of biological data is growing faster than the increase in processing power of computers. Many attempts have been made by several research groups to develop efficient algorithms as well as dedicated hardware/software solutions to deal with this explosion.

Manual implementation of Aho-Corasick algorithm on an FPGA is difficult due to its time consuming nature. As there is no details of automating Aho-Corasick Algorithm implementation on FPGAs in the literature (Yoginder et al., 2008, Akl et al., 2006), we give some information of the automation we performed.

**Methodology**

Since Aho-Corasick algorithm is the best and the widest used multiple pattern matching algorithm which searches all occurrences of any of a finite number of keywords in a text string, Yoginder et al. (2008) have used this algorithm for hardware acceleration of peptides pattern matching for the 1st chromosome of human genome. This algorithm consists of two phases; constructing a finite state machine from keywords and then using the state machine locating the keywords by processing the text string in a single pass. They have used bit split implementation of Aho-Corasick to reduce storage space. Here each amino acid is represented by 5 bits and then 5 Finite State Machines (FSM) are generated for the given set of peptides. An FSM can be constructed as a graph or a keyword tree, which consists of several states. FSM creation starts with the initial state '0' and then according to the given peptide (keyword) input it goes to next states (for new incoming characters) and finally it creates the machine deterministically. Final state of a given peptide represents the matching state. This algorithm can be used only with exact string matching applications and cannot be used with approximate string matching. We have automated the implementation of the hardware system using C++.

Our software can generate a full FSM as well as five bit-split FSMs automatically for a given set of peptides. This system outputs each bit-split FSM representing a bit of each amino acid (since there are 20 total numbers of amino acids each amino acid can be encoded into 5 bits) as VHDL models. The software makes use of tables for each and every FSM, which contain the states. Columns of these tables represent possible amino acids and rows represent all possible states (therefore data in the table represent next state according to the relevant input amino acid). This automation software also creates graphs (using graphviz software http://www.graphviz.org/) for each finite state machine indicating every state.

**Data collection, pre-processing and results**

We have used protein data from GenBank database and PeptideMass software which is available in Expasy Proteomics Server to generate peptides. PeptideMass creates possible peptides for a given protein. Then these peptides are input in the software system (implemented in C++) to generate the VHDL (Very-high-speed integrated circuit Hardware Description Language) implementation. For example consider a tile which has the average length of 4 amino acids, maximum length of 8 amino acids and minimum length of 3 amino acids of peptides. For this tile manually writing a VHDL model for hardware to match 20 peptides takes about 2 hours for an experienced hardware designer. However, our automated system takes around 150- 200 milliseconds to do the same and 450 – 475 milliseconds to generate all FSMs, tables and graphs in an Intel (R) Core (TM)2 Duo CPU 2 GHz with 2GB RAM and 32-bit operating system.

**Discussion**

It is time consuming and tedious to manually write a VHDL model that matches a large number of peptides. Therefore this automation gives an efficient and convenient way of implementing hardware in VHDL by just specifying the set of peptides.

**References**

Akl, P., Davor, C. and Ivan, M. (2006). SNIDS: FPGA-based Implementation of a Simple Network Intrusion Detection System. Retrieved from http://www.eecg.toronto.edu/~imatos/ECE532_Report_PatrickDavorIvan.pdf.

Yoginder, S.D., Shane, C.B., Mark, L. and Susan, M.B. (2008). Accelerating String Set Matching in FPGA Hardware for Bioinformatics Research, BMC Bioinformatics 2008 Journal, 9, 197. doi:10.1186/1471-2105-9-197