A flexible technique for the automatic design of approximate string matching architectures

Tomáš Martínček, Jan Kořenek, Otto Fučík
Faculty of Information Technology
Brno University of Technology
Boželechova 2, Brno, 612 66
Email: martinto, korenek, fucik@fit.vutbr.cz

Matej Lexa
Faculty of Informatics
Masaryk University
Botanická 68a, Brno, 602 00
Email: lexa@muni.cz

Abstract—Systolic array architectures for approximate string matching play a significant role as hardware accelerators in biological applications. However, their wider use is limited by the lack of flexibility required by often variable tasks. In this respect, it is desirable to develop a procedure for automatic design and implementation of such accelerators to reach high performance and efficiency with as little human effort on the side of the designer as possible. This paper proposes the essential element of such procedure, a method for the calculation of generic systolic array parameters with respect to maximal performance and efficient resource utilization.

I. INTRODUCTION

One of the key operations used in many biological applications is approximate matching (AM) between two strings. The typical methods for AM represent Smith-Waterman or Needleman-Wunsch algorithms. Unfortunately, their main disadvantage is the quadratic time complexity of computation. Considering this fact, substantial amount of research has focused on AM methods acceleration using dedicated hardware [1]. The authors usually concentrated on design optimization according to some dedicated task and/or specific target technology. However, as mentioned in [2], for the massive use of hardware accelerators, it’s important to generalize architectures for as wide an area of applications as possible.

The main contribution of this paper is therefore a novel technique for computation of generic systolic array parameters, while the objective is to achieve high performance and efficient resources utilization. Moreover, the performance criterion is based on the average number of active processing elements, which offers a more precise evaluation of real performance. The section II describes our generic systolic array architecture and proposed technique. The results are demonstrated in the section III and the conclusions are given in the section IV.

II. GENERIC SYSTOLIC ARRAY

The architecture of a generic systolic array is shown in the Figure 1. It consists of processing elements, memories for storing reference and test strings, a controller, and other necessary circuitry. The architecture is generic, so the user can define the input character width (in number of bits), width of the score value and the penalization function implemented in processing elements. In order to use the generic systolic array architecture effectively, the number of processing elements and other parameters such as the type of memory blocks etc. should be specified. All of these parameters affect the overall system performance.

The following relations between the systolic array size and the overall number of arrays placed in a single chip can be formulated: (1) In the case of a small number of long systolic arrays, the usage of individual processing elements will not be balanced because a portion of the available processing elements remains unused during the last computation stage. (2) For large number of short systolic arrays, the overhead for each array implementation rises (string memories, auxiliary FIFO, control logic) and thus prevents us from using more processing elements. Our objective therefore is to find the optimal balance between systolic array parameters to reach the highest possible average number of active processing elements. The average number reflects the fact, that not all elements are active in each clock cycle.

The technique for SA parameters calculation is based on the following set of formulas:

1) Amount of resources for memory blocks implementation, Look-Up Tables or BlockRAMs can be utilized: 

\[ SMem_{LUTs} = \left( \frac{h}{LUTSize} \right) \cdot 2 \cdot CharW, \]  

where \( CharW \) is the width of the input character, \( LUTSize \) is the size (in bits) of the memory represented by LUTs and \( h \) represents the maximal number of characters in the string memory.

Fig. 1. Generic systolic array architecture
2) Amount of resources for SA implementation according to number of PE elements ($SA_{size}$).

3) Efficiency of SA for appropriate string’s length $h$:

$$E_{SA} = \frac{h}{\lfloor h/PENum \rfloor \cdot PENum},$$

where $PENum$ is the number of PEs in the SA.

4) Average efficiency for range of input strings’ lengths

$$AvrE_{SA} = \frac{1}{h-l} \sum_{i=l}^{h} \frac{i}{\lfloor i/PENum \rfloor \cdot PENum}$$

where $l$ is minimal and $h$ is maximal string length.

5) And finally, average number of active PEs:

$$AvrTotalActivePEs = AvrE_{SA} \cdot TotalPENum$$

where $TotalPENum$ is total number of PE elements, which can be placed in the appropriate FPGA chip.

III. RESULTS

To demonstrate the advantages of the proposed approach for effective generic systolic array implementation, we chose the example of DNA sequence analysis: The alphabet consists of 4 characters, the target high-level method for sequence analysis in this example is PRIMEX [3] and Virtual PCR. For these reasons, the maximal length of input strings is 50 and the maximal allowed score is 16. To obtain important parameters for the overall array performance computation, the generic architecture of systolic array described in section II has been written in the VHDL language. The processing element and systolic array control logic was synthesized and implemented using Precision Synthesis and Xilinx ISE tools.

In the next step, the number of active PEs according to the systolic array length is calculated using the derived equations. The resulting characteristics for DNA sequences are shown in Figure 2. In figure, several graphs are shown for different implementations of SA memory blocks. Using either Look-Up Tables or BlockRAM memories as building blocks. In the case of DNA sequences, the best results were obtained if the systolic array contained 10 processing elements, the string memories were implemented using LUTs and FIFO was composed of BlockRAMs. It is possible to place up to 63 systolic arrays (630 PEs) in a single FPGA chip (xc2v3000) and to obtain approximately 500 active PEs in each clock cycle as a result.

The comparison with the most significant systems for DNA sequence analysis is shown in Table I. The system performance is measured in billions updates per second (one update represents computation of one SW matrix item). The approach presented in this paper shows better results than the other generic architectures.

IV. CONCLUSIONS

In this paper we propose a technique for the generic systolic array parameters computation. The method considers different memory building blocks for efficient implementations and the performance estimation is based on the average number of active processing elements, which better corresponds to real application performance. This approach shows better results against the other generic architectures.

Acknowledgements

This research has been supported by the Grant Agency of the Czech Republic under contract No. 102/04/0737 "Modern Methods of Digital Systems Synthesis".

REFERENCES

