Architecture Model for Approximate Tandem Repeat Detection

Tomáš Martínek
Faculty of Information Technology
Brno University of Technology
Božetěchova 2, Brno, 612 66, Czech Republic
Email: martinto@fit.vutbr.cz

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

***Abstract*—Algorithms for biological sequence analysis, such as approximate string matching or algorithms for identification of sequence patterns supporting specific structural elements, present good opportunities for hardware acceleration. Implementation of these algorithms often results in architectures based on multidimensional arrays of computing elements. Mapping effectively these computational structures on FPGAs remains one of the challenging problems. This paper focuses on a specific hardware architecture for detection of approximate tandem repeats in sequences. We show how to create a parametrized architecture model combined with an automatic technique that can determine appropriate circuit dimensions with respect to input task parameters and the target platform properties.**

***Keywords*-Approximate tandem repeat; Architecture model; Automated mapping; FPGA;**

I. Introduction

Recently a number of architectures for hardware acceleration of algorithms in bioinformatics have been published. The most important include string matching [1], approximate palindrome detection, repetitive structure detection [2] etc. The FPGA technology turned out to be more than appropriate for this kind of tasks. Thanks to its fine-grained parallelism and possibilities to fine-tune parameters of the overall architecture already at the time of synthesis, it allows to create circuits optimized for specific tasks. Circuits designed in this manner achieve hundred- to thousand-fold acceleration compared to the best purely software-based solutions. An architecture to identify approximate tandem repeats designed in our previous work is one of the examples of such work [3].

Unfortunately, application of these circuits in practical situations is complicated by several factors. First of all, the tasks benefiting from acceleration are quite varied, affecting the dimensions of resulting circuit, which is often composed of variable number of computing elements organized into multidimensional arrays, trees etc. [4].

A promising area in this respect is the field of High Level Synthesis (HLS) which enables users to automatically map circuits or even entire algorithms to ASIC and FPGA chips. The goal of this paper is to build upon our previous work [3] and explore the possibilities for automatic mapping of circuit architecture for approximate tandem repeat detection using the modern approaches of HLS. Our goal is to provide a technique based on a parametrized model of the previously developed hardware architecture that can automatically determine the optimal dimensions of the hardware circuits, given specific details about the sequence analysis task to be solved on the chip and the parameters of the target platform.

The remaining parts of this paper have been divided as follows: Chapter II describes the architecture model for approximate tandem repeat detection. Here we propose an automated method for mapping tasks to architectures. The results of evaluating the proposed model on Virtex5 chips is described in Chapter III, followed by conclusions in Chapter IV.

II. Model of architecture

Prior to creating an architecture model for tandem repeat detection, we first consider the overall system architecture that will host the circuit. Most often the target platform is an FPGA acceleration card with a PCI Express interface or perhaps some alternative interface. The symbols of the analyzed sequence are copied from host RAM to accelerator internal memory via DMA transfers. The main part of the circuit is made of an array of computing cores for tandem repeat detection. Information about identified tandem repeats from individual computing cores are aggregated into a single data stream and transferred back to host RAM via DMA transfers. Here they can be further processed by higher levels of software. Therefore, in addition to basic processing elements, blocks for communication and DMA transfers must be considered as well.

Each core is able to check for the presence of a tandem repeat with up to k errors for one specific tandem period only. To detect all tandem repeats in the input sequence of length n, the computing core must be applied recursively for all periods in the interval $(n/2 + k...n)$. To simplify this task, we consider a fixed number of tested periods in each iteration, from the range $(k...n_{avg})$, where n_{avg} will be the average length of the processed sequence and Q be the overall number of iterations.

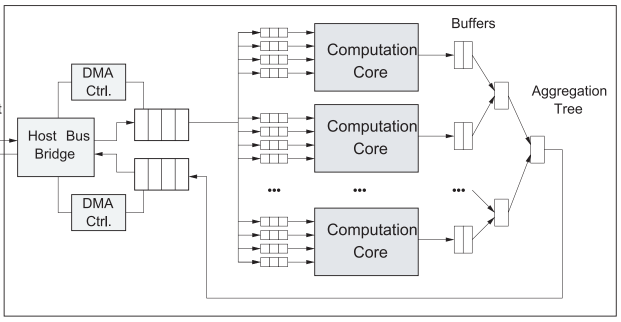

Figure 1. Overall architecture of the system for approximate tandem repeat detection, including PCI communication and DMA operation blocks

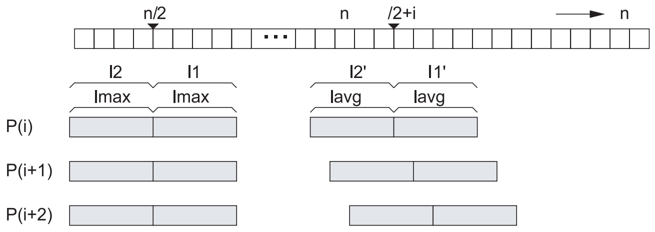

Figure 2. Processing of input string by a systolic array

A. Model parameters

The input parameters of the model will include: the maximal number k of errors allowed in detected tandem repeats, the average length of an input sequence n_{avg}, the overall number of input sequences (iterations) Q and the data-width of input sequence symbols C_{DW}. When constructing the model, we must also consider the parameters of the target platform, including: the number of available resources R_{FPGA}, available input B_{Sin} and output B_{Sout} bandwidth of the system. The number of computing cores N_{SA} is an architecture variable and the main objective of the mapping the architecture to FPGA effectively is to minimize the overall computation time.

B. Computation time

The first important property of the model is the overall computation time in hardware. Before deriving an expression for computation time, we need to express the average time (T_{avg}) necessary to test for the existence of a tandem repeat with a specific period. This elementary operation comprises the calculation of a DP matrix requiring S_{avg} steps and a subsequent evaluation of maximum on a wave, requiring at most $\lceil log_2(k+1)\rceil$ steps (for detailed information see [3]). The average time to process one period is therefore given by equation 1.

$$T_{avg} = \frac{S_{avg} + \lceil log_2(k+1\rceil)}{F} \tag{1}$$

The overall time necessary to complete the entire task is then obtained by summation of computation times for all possible periods and queries distributed over N_{SA} computing cores (see Equation 2).

$$f_T(N_{SA}) = \frac{Q \cdot (n_{avg} - k) \cdot T_{avg}}{N_{SA}} \tag{2}$$

C. Input bandwidth

To ensure all computing cores are utilized without unwanted wait time, it is necessary to supply them with enough input data. This underlines the importance of optimizing input bandwidth of the circuit. Each computing core processes four independent strings: two prefixes l_1 and l'_1 and two suffixes l_2 and l'_2. The way processing is distributed among cores and given that all periods in a single iteration use the same prefix l_1 and suffix l_2, it is not necessary to repeatedly read from host RAM. Also, for each new period $j+1$ we can use most of the symbols already available on the chip in prefix l'_1 and suffix l'_2 of the previously processed period j. The processing of input string on one computing core is shown in Figure 2. We assume the shared prefix l_1 and suffix l_2 will be read in their maximal length l_{max}. Similarly, it is assumed that the shared prefix l'_1 and suffix l'_2 will be read in length l_{avg}, corresponding to the average number of steps S_{avg}. With such assumptions, transition to a new period will on average require an additional reading of just one symbol. Overall, to process all $n_{avg} - k$ periods, the computing core will need $l_{max} + n_{avg}$ symbols (the entire right half of the segment and suffix l_2). The resulting relationship for required input bandwidth B_{in} is given in equation 3.

$$B_{In}(N_{SA}) = N_{SA} \cdot \frac{C_{DW} \cdot (l_{max} + n_{avg})}{(n_{avg} - k) \cdot T_{avg}} \tag{3}$$

D. Output bandwidth

The computing cores not only need to be supplied by data, but the output of processing the input sequences (information about detected tandem repeats) must be transferred back to the host RAM. It is therefore important to be able to secure an sufficient output bandwidth in the modeled architecture. The information transferred to RAM belongs to two classes: (1) a tandem repeat detected or (2) after executing defined number of steps a decision about the presence of a tandem repeat could not be made – i.e. decision will be made in software. In both cases the transfer to RAM includes the query number Q and the period resulting in the event. The resulting relationship expressing the required output bandwidth B_{Out} is given in equation 4, where E_{DW} is the data width of the records transferred to memory and $c_{ke}(n)$ are precalculated characteristics expressing an amount of exported tandems with respect to length of analyzed sequence.

$$B_{Out}(N_{SA}) = N_{SA} \cdot \frac{E_{DW} \cdot (0.0001 + c_{ke}(n_{avg}))}{T_{avg}} \tag{4}$$

E. Amount of resources

An estimation of overall amount of circuit resources with variable number of computing elements can be illustratively

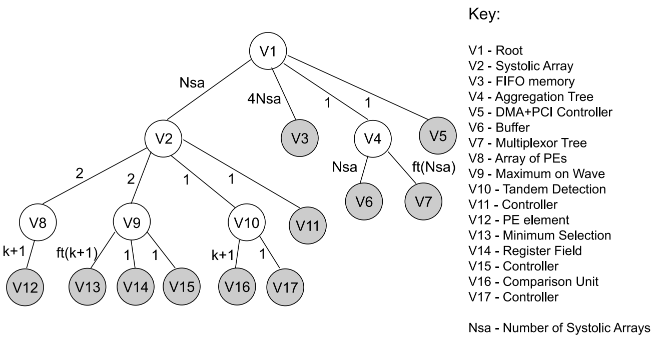

Figure 3. Tree structure of parametrized architecture approximate tandem repeat detection

expressed using tree structures corresponding to architecture hierarchy. Tree nodes represents system components and subcomponents and they are evaluated with amount of resources which consume. Edges represents relation between components and subcomponents and they are evaluated with number of subcomponent instances. An example of tree structure corresponding to architecture for approximate tandem repeat detection is shown on figure 3.

The function f_R estimating the overall amount of resources can be expressed recursively as a sum of all subcomponents resources:

$$\begin{aligned} f_R &= N_{SA} \cdot [4R_{FIFO} + R_{SA}] + \\ &\quad R_{Tree} + R_{DMA} \\ R_{SA} &= 2R_{PEArr} + 2R_{MaxW} + \\ &\quad R_{Det} + R_{SaCtrl} \\ R_{Fifo} &= f_{mem}(l_{max}, C_{DW}) \\ R_{Tree} &= N_{SA} \cdot R_{Buf} + f_{Tree}(N_{SA}) \cdot R_{MX} \end{aligned} \tag{5}$$

where R_{Fifo} is amount of resources for realization of auxiliary input buffers, R_{SA} for computing core, R_{DMA} for DMA blocks, R_{Tree} for aggregation tree, R_{PEArr} for processing element array, R_{MaxW} for unit evaluating maximum on the wave, R_{Det} for tandem detection unit and R_{SaCtrl} computing core controller. For reasons of clarity, individual functions were not divided into m-tuples with respect to different types of resources and level of detail was limited to primary parts of the circuit only.

F. Problem definition and method

The resulting architecture and the specific N_{SA} value respectively is considered valid, if it abides by the system of inequations 6. The first condition represents the fact that it does not make sense to use more computing cores then the number of queries Q. The second one limits the number of cores with respect to the amount of resources available on a chip. The last two conditions limit the number of cores with respect to input B_{Sin} and output B_{Sout} bandwidth.

$$\begin{aligned} Q - N_{SA} &\geq 0 \\ R_{FPGA} - f_R(N_{SA}) &\geq 0 \\ B_{Sout} - B_{Out}(N_{SA}) &\geq 0 \\ B_{Sin} - B_{In}(N_{SA}) &\geq 0 \end{aligned} \tag{6}$$

The objective of a method for automated mapping of the circuit to the FPGA chip is not only to find applicable architecture, but to find an architecture capable of solving the input task in the shortest computation time $f_T(N_{SA})$ as well. Generally, this is a kind of optimization problem in a bounded interval, where the solution primarily depends on properties of utilized functions. If all functions were linear, we would use techniques of linear programming. However, the detailed analysis of the model functions shows that some of them are monotonic (f_T, B_{In}, B_{Out}) and some of them are even linear combinations of monotonic functions (f_R). As the method for finding optimal architecture dimensions, we used a modified bisection method, which is generalized for linear combination of monotonic functions [5].

III. Evaluation and results

The objective of this section is to evaluate the proposed model on a specific set of input tasks and to show how dimensions of the resulting circuits will change on selected chips with Virtex5 LXT technology. The tasks are defined as follows: number of errors k is in the range $1 \ldots 100$, the average length of a run consists of $n_{avg} = 1000$ periods, sequence type is DNA ($C_{DW} = 2$) and the number of queries Q is 100 000 at least. The selected chips contain a huge amount of computational resources ranging from 7200 slices (xc5lx50t) up to 51840 slices (xc5lx330t). Moreover, all these chips contain an embedded IP core for its connection to the PCI Express x8 bus, with maximal theoretical throughput 16Gbps (2GBps) in each direction.

As a first step, all parts of the architecture were implemented. VHDL was used for hardware description and Xilinx ISE tools were used for synthesis. DMA operations and access to PCIe bus were performed using blocks from the NetCOPE platform [6]. Values representing the amount of resources of elementary architecture components were supplemented in the model in the form of constants (specifically in the f_R function).

Using the model we evaluated the requirements of the architecture for input and output bandwidth (without the impact of limited amount of resources). Since all chips achieve the same maximal input/output bandwidth of 16Gbps, the maximal number of computing cores placed inside the chip with respect to the limited input ($N_{PE_{BIN}}$) and output bandwidth ($N_{PE_{BOUT}}$) can be derived directly using equations 3 a 4. The results of this test are summarized in table I, where the maximal numbers of computing cores are listed for different values of k parameter.

Table I
NUMBER OF CORES WITH RESPECT TO LIMITED INPUT/OUTPUT BANDWIDTH

K	1	5	10	20
S_{avg}	8	24	44	84
$c_k(n_{avg})$	$5,4.10^{-4}$	$1,5.10^{-3}$	$2,4.10^{-3}$	$3,4.10^{-3}$
$N_{SA_{BIN}}$	371	1097	1922	3442
$N_{SA_{BOUT}}$	3221	8193	13347	22074
N_{SA_B}	371	1097	1922	3442

In the next part, the number of computing cores was evaluated with respect to the limited amount of resources (without the impact of limited amount of input/output bandwidth). Amount of resources of the whole circuit can be estimated using equation 5. With respect to the nonlinear character of this function, the number of cores N_{SA} can not be derived directly, however this is possible using the modified bisection method described in [5]. The following table II summarizes the maximal number of cores for individual chips with Virtex5 LXT technology and with respect to a variable number of errors k. If we compare these values with the previous table I, we find out that the impact of the limited amount of resources dominates the impact of limited input/output bandwidth for this kind of input task.

Table II
NUMBER OF CORES WITH RESPECT TO LIMITED AMOUNT OF RESOURCES

FPGA	Slices	K=1	K=10	K=50	K=100
xc5vlx50t	7 200	36	9	1	0
xc5vlx110t	17 260	86	21	4	1
xc5vlx220t	34 560	174	43	8	3
xc5vlx330t	51 840	261	65	13	5

Finally, we evaluated the speedup of the resulting circuitry with respect to the best known method implemented in software exploiting a suffix array data structure. The performance characteristics of the software algorithm executed on an Intel Core2 Duo E8400 3GHz processor can be obtained from our previous work [3]. Modified equations for computation time at the level of hardware and software are described in equation 7, where p_{SW} represents software performance in number of analyzed periods per second. The resulting speedup is shown in equation 8. For number of errors $k = 100$ we obtain speedup of 4575, which is a little bit lower in comparison to the previous one (4996).

$$\begin{aligned} T_{SW} &= \frac{Q\cdot(n_{avg}-k)}{p_{SW}} \\ T_{HW} &= \frac{Q\cdot(n_{avg}-k)\cdot T_{avg}}{N_{SA}} + \frac{0.0001Q\cdot(n_{avg}-k)}{p_{SW}} \end{aligned} \qquad (7)$$

$$\alpha = \frac{T_{SW}}{T_{HW}} = \frac{1}{\frac{T_{avg}\cdot p_{SW}}{N_{SA}} + 0.0001} \qquad (8)$$

IV. CONCLUSIONS

In this paper, we designed a novel parametrized model of architecture for approximate tandem repeat detection and a technique for its automated mapping to chips with FPGA technology. The proposed model also accounts for the real environment as close as possible. It includes necessary blocks for communication with the host system and respects a limited input/output bandwidth and limited amount of available resources. Based on this model and input task parameters the proposed method for automated mapping was able to find appropriate dimensions of the architecture on selected chips with Virtex5 technology. The results show that the limiting factor is usually the amount of available resources. A small decrease in the circuit speedup was observed during the comparison of the model with the results published in our previous work [3]. However, it is still in orders of hundreds or thousands in comparison to the best algorithm implemented in software exploiting a suffix array data structure.

ACKNOWLEDGMENT

This research has been partially supported by the Grant Agency of the Czech Republic Research Grant No. 204081560 – In vitro and in silico identification of non-canonical DNA structures in genomic sequences, MSMT Research Grant No.0021630528 – Security-Oriented Research in Information Technology and BUT Grant No. FIT-S-11-1 – Advanced secured, reliable and adaptive IT.

REFERENCES

[1] C. W. Yu, K. H. Kwong, K.-H. Lee, and P. H. W. Leong, “A smith-waterman systolic cell.” in *Field Programmable Logic and Application (FPL 2003)*, Lisbon, Portugal, September 2003, pp. 375–384.

[2] A. A. Conti, T. V. Court, and M. C. Herbordt, “Processing repetitive sequence structures with mismatches at streaming rate,” in *Field Programmable Logic and Application (FPL 2004)*, ser. Lecture Notes in Computer Science. Springer, 2004, pp. 1080–1083.

[3] T. Martnek and M. Lexa, “Hardware acceleration of approximate tandem repeat detection,” in *IEEE Symposium on Field-Programmable Custom Computing Machines*. IEEE Computer Society, 2010, pp. 79–86.

[4] T. VanCourt and M. Herbordt, “Sizing of processing arrays for fpga-based computation,” in *Field Programmable Logic and Application (FPL 2006)*, Madrid, Spain, September 2006, pp. 755–760.

[5] T. Martnek, “Evaluation of biological sequence similarity using fpga technology,” *Information Sciences and Technologies Bulletin of the ACM Slovakia*, vol. 2, no. 2, pp. 93–102, 2010.

[6] T. Martnek and M. Koek, “NetCOPE: Platform for rapid development of network applications,” in *Proc. of 2008 IEEE Design and Diagnostics of Electronic Circuits and Systems Workshop*. IEEE Computer Society, 2008, pp. 219–224.