Computation Core Binding in GTM Mapping on Reconfigurable Computers

Xuejun Liang
Jackson State University
Jackson, MS 39217
xuejun.liang@jsums.edu

ABSTRACT
A combinatorial optimization problem, where the cost function is the FPGA computation time and the constraint is the FPGA board resource, is formulated as a step in mapping generalized template matching operations onto an FPGA board. The problem is then simplified from a multiple FPGA chip case into a single FPGA chip case. Algorithms are proposed to solve the optimization problem. Experimental results are also given to show the efficiency of proposed algorithms.

Categories and Subject Descriptors
B.6.3 [**Logic Design**]: Design Aids – Automatic Synthesis, Optimization. G.2.1 [**Discrete Mathematics**]: Combinatorics – Combinatorial Algorithms.

General Terms
Algorithms, Performance, Design, Experimentation

Keywords
FPGA, Reconfigurable Computing, Template Matching, High-Level Synthesis, Combinatorial Optimization

1. INTRODUCTION
The reconfigurable computer addressed in the paper is a host computer with a co-processor board based on field programmable gate arrays (FPGAs). The target FPGA board may contain multiple FPGA chips, each with an array of homogeneous memory banks. Annapolis's FPGA boards and SRC's MAP processors are such examples.

The generalized template matching (GTM) operations [1] include image-processing algorithms for template matching, 2D digital filtering, morphologic operations, motion estimation, and so on. They all involve moving a "window" (or template) pixel by pixel in a scanned line order.

The overall approach of mapping the GTM operations onto reconfigurable computers consists of three steps. The first two steps enumerate, evaluate, and list enough number of basic GTM building blocks, called region functions (RFs). Each RF contains an FPGA buffer and a pipelined functional unit, called a unit function, which evaluates the window computation at one or more consecutive pixel locations. Different RFs will have different throughputs, occupy different FPGA areas, and require different numbers of memory ports. The third step, called RF binding, is to select one or more RFs for each FPGA chip such that the total FPGA execution time is minimal under the FPGA board resource constraints such as the number of FPGA chips, the size of FPGA chips, and the number of memory ports. RFs on all FPGA chips work independently and in parallel on different image regions and/or, if any, different templates under the control of a host program.

ACM SE'06, March, 10-12, 2006, Melbourne, Florida, USA

2. RF BINDING
In the RF binding process, the selected RFs are assigned to FPGA chips, memory ports, templates, and processing regions (consecutive rows of an image region). The process therefore includes the FPGA chip binding, the memory port binding, the image region partitioning and the processing region binding, and the template binding. For an FPGA chip, i ($1 \le i \le N_{FPGA}$), the region functions $RF_{i,j}$, $1 \le j \le q(i)$, together form the chip design, which has to satisfy an FPGA area constraint and a memory port constraint. As a result, for the FPGA chip binding and the memory port binding, a combinatorial optimization problem can be formulated as follows.

$$(2.1)\begin{cases} \text{To minimize} \\ \quad \max\{Time(RF_{i,j}) \mid 1 \le i \le N_{FPGA}, \text{and } 1 \le j \le q(i)\} \\ \text{Subject to} \\ \quad \begin{cases} \sum_{1 \le j \le q(i)} Area(RF_{i,j}) \le S_{FPGA},\ 1 \le i \le N_{FPGA} \\ \sum_{1 \le j \le q(i)} Port(RF_{i,j}) \le N_{MP},\ 1 \le i \le N_{FPGA} \end{cases} \end{cases}$$

In the above formulation, the objective function is the GTM computation time, N_{FPGA} is the number of FPGA chips on the target board, S_{FPGA} is the size (number of slices) of FPGA chip, and N_{MP} is the number of memory ports connected to each FPGA chip. $Time(RF_{i,j})$ is the $RF_{i,j}$ execution time, $Area(RF_{i,j})$ is the $RF_{i,j}$ FPGA area, and $Port(RF_{i,j})$ is the number of memory ports used by $RF_{i,j}$. The execution time of the GTM design is the maximum execution time of all $RF_{i,j}$ execution times as all $RF_{i,j}$ work independently and in parallel.

To formulate the RF binding problem completely, we define the workload of a GTM operation to be the sum of products of the number of rows of each image region and the number of templates that are applied to the image region.

Similarly, the workload of a RF after the image region partition, the processing region binding, and the template binding is defined to be the sum of products of rows of each assigned processing region and the number of assigned corresponding templates. Therefore, it is clear that the image region partitioning, the processing region binding, and the template binding is a way to partition the GTM workload among the selected RFs.

Therefore, the RF binding problem can be formulated in terms of the workload concept as follows.

$$(2.2)\quad \begin{cases} \text{to minimize} \\ \quad \max\{S(RF_{i,j}) \times WL(RF_{i,j}) \mid \\ \quad 1 \le i \le N_{FPGA},\ \text{and}\ 1 \le j \le q(i)\} \\ \text{subject to} \\ \quad \begin{cases} \sum_{1 \le j \le q(i)} Area(RF_{i,j}) \le S_{FPGA},\ 1 \le i \le N_{FPGA} \\ \sum_{1 \le j \le q(i)} Port(RF_{i,j}) \le N_{MP},\ 1 \le i \le N_{FPGA} \\ \sum_{i=1}^{N_{FPGA}} \sum_{j=1}^{q(i)} WL(RF_{i,j}) = GTM\_workload \end{cases} \end{cases}$$

In the above formulation, $S(RF_{i,j})$ is the computation time of $RF_{i,j}$ for one image row under one template, $WL(RF_{i,j})$ is the workload of $RF_{i,j}$, and the last constraint is for the workload partitioning among selected RFs.

Note that there is a solution to Problem (2.2) in which all FPGA chips contain a common set of RF designs. Therefore, the RF binding problem (2.2) can be simplified to one for a single FPGA chip case as follows.

$$(2.3)\quad \begin{cases} \text{To minimize} \\ \quad \max\{S(FR_j) \times WL(RF_j) \mid 1 \le j \le q\} \\ \text{subject to} \\ \quad \begin{cases} \sum_{1 \le j \le q} Area(RF_j) \le S_{FPGA} \\ \sum_{1 \le j \le q} Port(RF_j) \le N_{MP} \\ \sum_{j=1}^{q} WL(RF_j) = single\_workload \end{cases} \end{cases}$$

where $single\_workload = GTM\_workload / N_{FPGA}$.

For each selected RF_j, let the workload be defined as

$$(2.4)\quad WL(RF_j) = \frac{single\_workload}{S(RF_j) \times (\sum_{j=1}^{q} 1/S(RF_j))}$$

Then, the RF binding problem (2.3) can be further simplified as

$$(2.5)\quad \begin{cases} \text{To maxmize} \\ \quad \sum_{1 \le j \le q} 1/S(RF_j) \\ \text{subject to} \\ \quad \begin{cases} \sum_{1 \le j \le q} Area(RF_j) \le S_{FPGA} \\ \sum_{1 \le j \le q} Port(RF_j) \le N_{MP} \end{cases} \end{cases}$$

A naive method to solve (2.5) can be through enumeration of the solution space and compute the following: For each q ($1 \le q \le N_{MP}$), select q RFs out of the candidate RFs, verify the constraint conditions, and compute the sum of $1/S(RF_j)$.

In order to reduce the search space, the candidate RF designs can be divided into N_{MP} groups, denoted by Cad(i) (i = 1, 2, …, N_{MP}), such that each candidate RF design in Cad(i) requires i memory ports. In this way RF designs can be selected from individual groups instead of from the whole set of candidate RF designs provided that it is known which groups RF designs should be chosen from. This can be achieved by enumerating the solution space of the memory port constraint in Problem (2.5) first. Then each solution provides information about which groups to choose from. Based on this idea, Problem (2.5) can be solved by solving the following two problems.

$$(2.6)\quad \begin{cases} i_1 + i_2 + \ldots + i_q \le N_{MP} \\ 1 \le i_1 \le i_2 \le \ldots \le i_q \\ 1 \le q \le N_{MP} \end{cases}$$

$$(2.7)\quad \begin{cases} \text{To maximize} \\ \quad \sum_{1 \le j \le q} 1/S(RF_{i_j}) \\ \text{subject to} \\ \quad \sum_{1 \le j \le q} Area(RF_{i_j}) \le S_{FPGA} \end{cases}$$

Problem (2.6) is closely related to the integer partition problem. Using the dynamic programming technique can solve this problem efficiently.

Problem (2.7) is a well-known bounded knapsack problem. There exist many classic approaches for solving it. This work provides a new algorithm, called Multi-Dimensional Binary Search, to solve the knapsack program. It uses the divide and conquer method. In each search step, either the search terminates because a search result is found or no result can be found, or the search problem is divided into some smaller size problems.

3. EXPERIMENT RESULTS

From Section 4, there are three methods to solve (2.5). The first one (called the naïve method) is to search the whole design space. The second and the third methods are to solve (2.6) first and then to solve (2.7) for each solution of (2.6). In the second method (called the simple grouping method), (2.7) is solved by searching all the combinations from RF groups. In the third method (called the multi-dimensional binary search), (2.7) is solved by the multi-dimensional binary search algorithm. The three methods are implemented and used for different FPGA sizes. The average search space sizes and the average computation times are listed in Table 1.

Table 1: Search Space and Computation Time

	Space Size	Time (in Second)
Naïve	2880952	148.05
Simple Grouping	9982	0.526
Multi-Dimensional Binary Search	1138	0.1288

4. REFERENCES

[1] X. Liang and J. Jean, Mapping of Generalized Template Mapping onto Reconfigurable Computers, IEEE Trans. on VLSI System, 11(3): 485-498, 2003