Combinatorial Optimization in Mapping Generalized Template Matching onto Reconfigurable Computers

Xuejun Liang* and Qutaibah Malluhi**
*Department of Computer Science, Jackson State University, Jackson, MS, USA 39217
xuejun.liang@jsums.edu
**Computer Science and Engineering Department, Qatar University, Doha, Qatar
qmalluhi@qu.edu.qa

ABSTRACT: A brief review of mapping generalized template matching operations onto reconfigurable computers is given. A combinatorial optimization process, where the objective is to minimize the FPGA computation time and the constraint is the FPGA board resources, is a key step of the whole mapping process. This paper presents algorithms for solving the optimization problem and experimental results.

Keywords: FPGA, Reconfigurable Computing, Template Matching, Combinatorial Optimization

1. Introduction

The generalized template matching (GTM) operations 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 scanning line order. GTM operations offer ample parallelism opportunities and can be accelerated by reconfigurable computers based on FPGA coprocessor boards.

Previously the first author and other researchers at Wright State University in Ohio developed a tool to automatically explore the GTM FPGA design space and produce VHDL files and C++ host program [1-3]. The constraints of FPGA chip area and memory ports connected with each FPGA were considered by the tool. Our approach is different from many compiler-based design environment efforts on reconfigurable computers [4-5]. By restricting the target applications, our approach can explore the design space and provide design solutions that satisfy FPGA board resource constraints. This is done by evaluating different buffer structures and changing the amount of parallelism and hardware sharing.

The overall approach of mapping 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. These two steps were reported in [1]. The third step, called RF binding, selects one or more RFs for each FPGA chip such that the total FPGA execution time is minimal under the FPGA board resource constraints. 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. This paper will present RF binding problem, which is formulated as a combinatorial optimization problem, algorithms for solving the problem, as well as some experimental results.

2. RF Binding Problem

In the RF binding process, the selected RFs are needed to bind to FPGA chips, memory ports, templates, and processing regions (consecutive rows of an image region).

The workload of a GTM operation is defined 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 selected RF is defined to be the sum of products of rows of each assigned processing region and the number of assigned corresponding templates. It is clear that the workloads of all selected RFs must construct a partition of the overall GTM workload. It can be proved that if the GTM workload is evenly partitioned among FPGA chips, then only one FPGA chip is needed to consider in the RF binding process.

The execution time of a selected RF working on the workload *WL(RF)* can be approximately computed by

$$(2.1) \qquad Time(RF) = S(RF) \times WL(RF)$$

where S(RF) is the computation time of RF computing along one image row with one template.

Now, assume that RF_1, RF_2, …, and RF_q are selected for an FPGA design. Then, the FPGA execution time is equal to max{Time(RF_j) : $1 \le j \le q$} because all RF_j ($j = 1, 2, \ldots, q$) work independently and in parallel. Note that if the workload of RF_j ($j = 1, 2, \ldots, q$) is assigned according to the following formula:

$$(2.2) \quad WL(RF_j) = \frac{Total\_workload}{S(RF_j) \times (\sum_{1 \le j \le q} 1/S(RF_j))}$$

then, WL(RF_1), WL(RF_2), …, WL(RF_q) construct a partition of the overall GTM workload, and the FPGA execution time is equal to

$$(2.3) \quad \frac{Total\_workload}{\sum_{1 \le j \le k} 1/S(RF_j)}$$

Also note that to minimize the FPGA execution time (2.3) is equivalent to maximize sum of 1/S(RF_j) over j (j=1, 2, …, q). Therefore, the RF binding problem can be formulated as follows:

$$(2.4) \quad \begin{cases} \text{To maxmize} \\ \qquad \sum_{1 \le j \le q} 1/S(RF_j) \\ \text{subject to} \\ \quad \begin{cases} \sum_{1 \le j \le q} Area(RF_j) \le S_{FPGA} \\ \sum_{1 \le j \le q} Port(RF_j) \le N_{MP} \end{cases} \end{cases}$$

In the above formulation, the objective is to minimize the FPGA computation time, S_{FPGA} is the size (number of slices) of an FPGA chip, and N_{MP} is the number of memory ports connected to each FPGA chip, Area(RF_j) is the RF_j FPGA area, and Port(RF_j) is the number of memory ports used by RF_j, j = 1, 2, …, q. The first constraint is the FPGA area constraint; the second constraint is the memory port constraint. Note that different RFs will not share memory ports because all RFs need to access onboard memory every clock cycle.

Note that after one gets a set of RFs from (2.4), one can simply bind them to all templates, and then bind them to appropriate processing regions such that (2.2) holds. Also note that when (2.2) is not an integer, a truncation should be used.

3. RF Binding Algorithms

A naive method to solve Problem (2.4) is to go through the entire search space and to compute the following: For each q ($1 \le q \le N_{MP}$), select all possible q RFs out of the candidate RFs, verify the constraint conditions, and compute and compare sums of 1/S(RF_j) over j (j=1, 2, …, q). However, the complexity of this naïve method is equal to $\theta(N^{N_{MP}})$, where N is the number of all candidate RF designs.

3.1 Grouping

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 group each RF design should be chosen from. Based on this idea, Problem (2.4) can be solved by solving the following problem:

$$(3.1) \quad \begin{cases} i_1 + i_2 + \ldots + i_q \le N_{MP} \\ 1 \le i_1 \le i_2 \le \ldots \le i_q \\ 1 \le q \le N_{MP} \end{cases}$$

first, and then for each solution (i_1, i_2, …, i_q) to (3.1), solving the following problem:

$$(3.2) \quad \begin{cases} \text{To maximize} \\ \qquad \sum_{1 \le j \le q} 1/S(RF_{i_j}) \\ \text{subject to} \\ \qquad \sum_{1 \le j \le q} Area(RF_{i_j}) \le S_{FPGA} \end{cases}$$

where $RF_{i_j} \in Cad(i_j)$. Therefore, the solution to (3.2) for (i_1, i_2, …, i_q) also satisfies the FPGA area constraint in (2.4), and then is a feasible solution to (2.4).

3.2 Integer Partition Algorithm

Problem (3.1) is closely related to the following integer partition problem:

$$(3.3) \quad \begin{cases} i_1 + i_2 + \ldots + i_q = n \\ 1 \le i_1 \le i_2 \le \ldots \le i_q \end{cases}$$

The solution to the above problem is a set of q-vectors denoted by $S(q,n)$. For example, if n=8 and q=4, then $S(q, n)$ = { (1,1,1,5), (1,1,2,4), (1,1,3,3), (1,2,2,3), (2,2,2,2)}. Thus the solution space to (3.1) is the union of $S(q, n)$ over all q and n satisfying $1 \le q \le n \le N_{MP}$.

Define an operation: Θ: I×{q-vectors} → {(q+1)-vectors} such that $a \ \Theta \ (a_1, a_2, \ldots, a_q) = (a, a_1+a-1, a_2+a-1, \ldots, a_q+a-1)$. Define a notation $a\Theta S(q,n)$ = $\{a\Theta(a_1, a_2, \ldots, a_q) : (a_1, a_2, \ldots, a_q) \in S(q,n)\}$. Then the solution to the above integer partition problem can be expressed by the following recursive equation.

$$S(q,n) = \begin{cases} \{(n)\} & if \quad q=1 \\ \{(1,1,\ldots,1)\} & if \quad q=n \\ \bigcup_{b=1}^{k} b\Theta S(q-1,(n-b)-(b-1)*(q-1)) & otherwise \end{cases}$$

where $k=\lfloor n/q \rfloor$.

In fact, $S(q,n)$ needs to be computed for all q and n that satisfy $1 \le q \le n \le N_{MP}$. If each $S(q,n)$ is computed by the above recursive equation, there would be a lot of repeated computations. To avoid that, the dynamic programming technique should be used. First, compute all $S(i,i)$, i = 1, 2, …, N_{MP}, and all $S(1,j)$, j = 2, 3, …, N_{MP}. Then, for $k > 1$, compute all $S(k,j)$, $j = k+1$, …, N_{MP}, based on the previously computed value of $S(k\text{-}1, j)$, $j = k$, …, N_{MP}.

3.3 Multi-Dimensional Binary Search

This section presents an efficient algorithm, called Multi-Dimensional Binary Search, to solve Problem (3.2). 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 smaller size problems.

Initially, each candidate RF group $Cad(i) = \{ RF_{i,j} \mid j=1,2,\ldots,m(i)\}$, (i = 1, 2, …, N_{MP}), can be sorted in the decreasing order of the RF speeds (i.e. 1/S(RF)). So it is assumed that

(3.4) $S(RF_{i,1}) < S(RF_{i,2}) < \ldots < S(RF_{i,m(i)})$

It can be also assumed that

(3.5) $Area(RF_{i,1}) > Area(RF_{i,2}) > \ldots > Area(RF_{i,m(i)})$

This assumption is reasonable because if an RF in Cad(i) uses larger FPGA area and has slower speed than another RF in Cad(i), it should be removed from the candidate RF group.

Given a solution (i_1, i_2, …, i_q) to (3.1), compute sum of the largest RF area (SLA) and sum of the smallest RF area (SSA), from Cad(i_1), …, Cad(i_q), respectively:

(3.6) $SLA = \sum_{1\le j\le q} Area(RF_{i_j,1})$

(3.7) $SSA = \sum_{1\le j\le q} Area(RF_{i_j,m(j)})$

If $SSA > S_{FPGA}$, then no $\{ RF_{i_j} \mid 1 \le j \le q\}$ will satisfy the FPGA area constraint. So there is no solution to (3.2) for (i_1, i_2, …, i_q). On the other hand, if $SLA < S_{FPGA}$, then the set $\{ RF_{i_j,1} \mid 1 \le j \le q\}$ is a feasible solution to (3.2) for (i_1, i_2,…, i_q).

Otherwise, compute sum of the median RF area (SMA) from Cad(i_1), …, Cad(i_q):

(3.8) $SMA = \sum_{1\le j\le q} Area(RF_{i_j,m(j)/2})$

Note that when $m(j)$ is not divisible by 2, the ceiling of $m(j)/2$ is used. For example, 1/2 = 1 and 3/2 = 2.

If $SMA > S_{FPGA}$, then Cad(i_j) ($1 \le j \le q$) can be divided into two parts: $Cad(i_j,0) = \{RF_{i_j,k} \mid k = 1,\ldots,m(i_j)/2\}$ and $Cad(i_j,1) = \{RF_{i_j,k} \mid k = m(i_j)/2+1,\ldots,m(i_j)\}$. There are 2^q combinations in picking q RFs from these $2\times q$ sets. One combination Cad(i_1,0),…, Cad(i_q,0) can be ruled out right away. So the search can be divided into 2^q-1 sub-problems. Otherwise, if $SMA \le S_{FPGA}$, then Cad(i_j) ($1 \le j \le q$) can be divided into two parts: $Cad(i_j,0) = \{RF_{i_j,k} \mid k = 1,\ldots,m(i_j)/2-1\}$ and $Cad(i_j,1) = \{RF_{i_j,k} \mid k = m(i_j)/2,\ldots,m(i_j)\}$. Picking q RFs from these 2×q sets gives 2^q combinations. Then the search can be divided into 2^q-1 sub-problems by ruling out one combination Cad(i_1,1), …, Cad(i_q,1). In addition, the set $\{ RF_{i_j,m(i_j)/2} \mid 1 \le j \le q\}$ is a feasible solution of (3.2) for (i_1, i_2,…, i_q).

Assume that each Cad(i) contains the same number of RFs. The worst-case complexity of this algorithm will be

$$\begin{cases} \theta(M^{\log_2(2^q-1)}) & \text{if } q>1 \\ \theta(\log_2 M) & \text{if } q=1 \end{cases}$$

where M=N/N_{MP} and N is the number of all candidate RFs. Note that Problem (3.2) can also be solved by an exhaustive search, i.e. searching all the combinations from RF groups. Under the same assumption, the worst-case complexity of the exhaustive search will be $\theta\left(M^q\right)$.

It can be noticed that when q is large, the worst-case complexity of multi-dimensional binary search is almost equal to $\theta\left(M^q\right)$. But in an average case, it is much smaller.

The following example illustrates how the multi-dimensional binary search algorithm works. Assume S_{FPGA}=37 and (1, 2) is a solution to (3.1). So we need to find one RF from Cad(1) and one RF from Cad(2). Assume there are six RFs in Cad(1) and five RFs in Cad(2) and their areas are listed in the following table.

j	1	2	3	4	5	6
Area($RF_{1,j}$)	24	20	16	14	13	12
Area($RF_{2,j}$)	35	29	26	22	18	NA

First, SLA = 24+35 = 59 > S_{FPGA} and SSA=12+18 = 30 < S_{FPGA}. As SMA = 16+26 = 42 > S_{FPGA}, three sub-problems are needed to consider. The groupings of the three sub-problems are shown in the following three tables.

Sub(1): j	1	2	3	4	5	6
Area($RF_{1,j}$)	NA	NA	NA	14	13	12
Area($RF_{2,j}$)	NA	NA	NA	22	18	NA

Sub(2): j	1	2	3	4	5	6
Area($RF_{1,j}$)	NA	NA	NA	14	13	12
Area($RF_{2,j}$)	35	29	26	NA	NA	NA

Sub(3): j	1	2	3	4	5	6
Area($RF_{1,j}$)	24	20	16	NA	NA	NA
Area($RF_{2,j}$)	NA	NA	NA	22	18	NA

For the sub-problem (1), since *SLA* = 14+22 = 36 < S_{FPGA}, ($RF_{1,4}$, $RF_{2,4}$) is a feasible solution to (3.2). For the sub-problem (2), since *SSA* = 12+26 = 38 > S_{FPGA}, there is no solution. For the sub-problem (3), *SLA* = 46 and *SSA* = 34. As *SMA* = 42 > S_{FPGA}, it splits into three sub-problems again as shown below. Sub-problem (3-1) provides another feasible solution ($RF_{1,3}$, $RF_{2,5}$), and Sub-problems (3-2) and (3-3) have no solution.

Sub(3-1): J	1	2	3	4	5	6
Area($RF_{1,j}$)	NA	NA	16	NA	NA	NA
Area($RF_{2,j}$)	NA	NA	NA	NA	18	NA

Sub(3-2): J	1	2	3	4	5	6
Area($RF_{1,j}$)	NA	NA	16	NA	NA	NA
Area($RF_{2,j}$)	NA	NA	NA	22	NA	NA

Sub(3-3): J	1	2	3	4	5	6
Area($RF_{1,j}$)	24	20	NA	NA	NA	NA
Area($RF_{2,j}$)	NA	NA	NA	NA	18	NA

The final solution to (3.2) for (1, 2) then will be one of the two feasible solutions that has maximum sum of speeds.

4. Experimental Results and Analysis

From Section 3, Problem (2.4) can be solved by a naïve method or by solving Problem (3.1) and Problem (3.2). Further, Problem (3.2) can be solved by either the exhaustive search or the multi-dimensional binary search. These three methods of solving (2.4), called the naïve method, the exhaustive search, and the multi-dimensional binary search, respectively, were all implemented in a GTM design tool for testing their performance. In the experiment, assume N_{MP}=4, for a GTM operation with a 3×4 template, 43 candidate RFs were obtained first by the process of enumerating basic GTM building blocks. Then the RF binding program is executed with each of the above three methods for different FPGA Slice counts.

In average, the search space of the naïve approach is about 288 times larger than that of the exhaustive search; the search space of the exhaustive search is about 9 times larger then that of the multi-dimensional binary search. The search space sizes of the exhaustive search and the multi-dimensional binary search are depicted in Figure 1. (The diagram for the computation times has a similar shape with that in Figure 1 and is omitted.)

When the FPGA Slice count (area constraint) is larger than a value, the search space size of the exhaustive search are constant. This is because when the FPGA size is large enough, the number of candidate RFs is fixed. When FPGA size is small, RF designs whose area is bigger than FPGA size will be not considered. On the other hand, the search space size of the multi-dimensional binary search has a normal distribution over the FPGA Slice counts. When the FPGA area is relatively small or large enough, the search space size is small, because the multi-dimensional binary search will need less numbers of iterations.

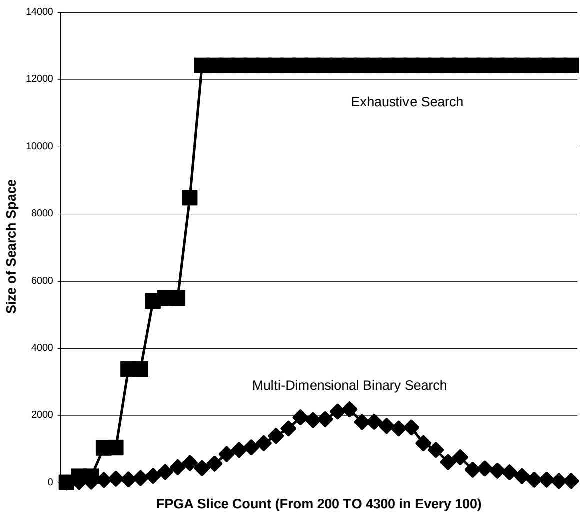

Figure 1: Sizes of the Search Spaces

5. 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

[2] J. Jean, X. Liang, X. Guo, H. Zhang, and F. Wang, Initial Results of GOM (GTM Optimal Mapping), Proceedings of International Conference on Engineering of Reconfigurable Systems and Algorithms", Las Vegas, Nevada, USA, pp. 146-152, June 2002

[3] X. Liang and J. Jean, Memory Access Pattern Enumeration in GTM Mapping on Reconfigurable Computers, Proceedings of International Conference on Engineering of Reconfigurable Systems and Algorithms, pp. 8-14, June 2001

[4] W. Bohm, J. Hammes, B. Draper, M. Chawathe, C. Ross, R. Rinker, and W. Najjar, Mapping a Single Assignment Programming Language to Reconfigurable systems, Supercomputing, 21:117-130, 2002

[5] D. C. Cronquist, etc, Specifying and Compiling Applications for RaPiD, in IEEE Symposium on FPGA Custom Computing Machines, pp. 116-125, April 1998.