A Graph Covering Method for Template Based System Partition

Muhua HAN*, Leibo LIU, Shaojun WEI
Tsinghua National Laboratory for Information Science and Technology,
Institute of Microelectronics,
Tsinghua University, Beijing 100084, China
*hanmuhua00@mails.tsinghua.edu.cn

Abstract—**Template based system partition method by regularity extraction can be applied to hardware and software architecture generation of complex systems, such as multi-core system and ASIP (Application Specific Instruction-set Processor) containing reconfigurable computing. In the previous work, the search of optimal graph cover, which is the key of this method, was just a greedy way and adopted simple criterions for optimization, thereby sub-optimal and limited. This paper presents a graph covering method by adapted genetic algorithms for template based system partition. Different criterions can be set as evaluation models of various applications, and the optimal covers will always be found by the genetic iterative. Experiments show that the genetic algorithms are effective, even to the time constrained optimization problems including allocation, binding and scheduling.**

I. INTRODUCTION

Nowadays, ASIP like architecture with multiple computing units, even containing reconfigurable resources, are promoted for high performance requirement of embedded multimedia applications. Furthermore, the multimedia applications are known to have a good amount of intrinsic regularity in them. Extracting regularity can be used as an effective way to hardware and software architecture construction of such ASIP.

Template based system partition methods, which consists of template matching and graph covering, have been applied to circuit design, such as area and performance optimization of data-path in high-level and gate-level [1], micro cell generation in RTL [2, 3], low power design [4], instruction generation of reconfigurable processor [5], and system partition in behavior level [6]. Graph covering by multiple templates is the most important problem in the template based partition method, and is known to be NP-Complete[7]. Previous works solved the graph covering problem by a greedy way and did not explore the solution space sufficiently, thus resulted in sub-optimal solutions. [1, 5] selected sub-graphs to cover the systems by the priorities of the templates, but the greedy method was not optimal due to the simple definition of priorities. [2-4] constructed the cover to reduce the total execution delay and used an iterative way to select better matches. However the way to avoid being trapped to local minimum and the allocation of resources were not considered. [6] used conflict graphs to select the matches of templates to cover the specification. It was a list based algorithm and the critical path was optimized. However it was also a greedy way, in which the binding and scheduling of the sub-graphs were neglected, and thus sub-optimal. And all the previous works did not consider the covering optimization problem with constraints, as well as the communication model of the architecture.

A novel graph covering method is presented in this paper by adapted genetic algorithms, which can avoid trapping into the local minimum for the reason of the probabilistic nature of the algorithm. Moreover various applications, such as instruction generation, system partition with constraints, can be solved by just changing the evaluation model of the procedures. A two-dimension coding of the gene is adopted to disperse the selections of different templates and their matches. To remove the invalid solutions, which were generated by crossover and mutation, a parallel solution transformation procedure is constructed. The optimal covers can be found in most cases, if computation time is not critical. Examples of this method are described in the latter of this paper.

II. TEMPLATE BASED SYSTEM PARTITION AND GRAPH COVERING PROBLEM

Template based system partition method by regularity extraction can be employed to solve many problems in circuit designs. Specification of design can be represented as DAG (Directed Acyclic Graph), denoted as $G(V, E)$, where the node in V corresponds to a task and the edge in E corresponds to interconnection or communication between tasks. And template is defined as the common extraction of similar sub-graphs of DAG. Design can be partitioned by several templates, and then be represented as groups of occurrences of the templates called matches. Template based system partition method includes two

Supported by NNSF of China grant 60506007 & 60676012

important problems: a) template generation and sub-graph matches, b) graph covering by multiple templates. This paper is focusing on the latter problem.

Graph covering problem: for s task graph $G(V,E)$, sub-graph $S_i(A_i,P_i)\subseteq G(V,E)$ is a match by $T(V^T,E^T)$, denoted as $S_i(A_i,P_i)\cong T(V^T,E^T)$. Let $\{S_i^l(A_i^l,P_i^l)\}_{i=1,2,..,m_l}$ be the set of all matches in $G(V,E)$ by $T(V^T,E^T)$, such that for each $S_i^l(A_i^l,P_i^l)\subseteq G(V,E)$, $S_i^l(A_i^l,P_i^l)\cong T_l(V_l^T,E_l^T)$, $i=1,2,..,m_l$. Then graph covering problem is defined as to find a group of sets $\{S_i^1(A_i^1,P_i^1)\}_{i=1,2,..,m_1}$, $\{S_i^2(A_i^2,P_i^2)\}_{i=1,2,..,m_2}$, ..., $\{S_i^L(A_i^L,P_i^L)\}_{i=1,2,..,m_L}$, which satisfies equations (1)-(4).

$$\forall i,j\in\{1,2,..,m_l\},\text{and } i\neq j: S_i^l(A_i^l,P_i^l)\bigcap S_j^l(A_j^l,P_j^l)=\varnothing, l=1,2,..,L \quad (1)$$

$$\forall l',l''\in\{1,2,..,L\},\text{and } l'\neq l'', i\in\{1,2,..,m_{l'}\}, \\ j\in\{1,2,..,m_{l''}\}: S_i^{l'}(A_i^{l'},P_i^{l'})\bigcap S_j^{l''}(A_j^{l''},P_j^{l''})=\varnothing \quad (2)$$

$$\bigcup_{l\in\{1,2,..,L\}}\Big(\bigcup_{i\in\{1,2,..,m_l\}} S_i^l(A_i^l,P_i^l)\Big)=G(V,E) \quad (3)$$

$$\forall i\in\{1,2,...,m_l\}, S_i^l(A_i^l,P_i^l)\cong T_l(V_l^T,E_l^T) \quad (4)$$

Graph covering procedure is described in Fig.1. Since the number of templates and their matches can be very large in some cases, the search of optimal graph cover is therefore a time consuming problem, and must be solved by heuristic algorithm. The covering procedure is in an iterative manner. Evaluation model may be varied according to various applications. And hyper-graph execution model with the template information is used to synthesize and estimate the cost of the cover.

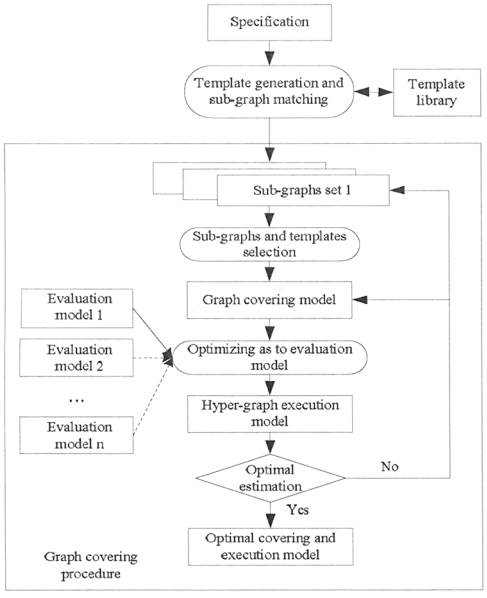

Fig.1 Graph covering procedure

III. GENETIC ALGORITHMS FOR COVERING OPTIMIZATION

Holland [8] originated the concept of genetic algorithms to mimic the process of natural evolution. A standard genetic algorithm consists of reproduction, crossover, and mutation operators. This is first paper to use genetic algorithm to solve graph covering problem. Genetic algorithm here is adapted by adding a transformation procedure to remove the invalid solutions after mutation operators.

A. Graph Covering Procedure by Multiple Templates

A graph cover consists of several matches by various templates which do not conflict to each other. Graph cover is encoded as gene, and different combinations of the matches are explored by evolution iterative. It starts with an initial population of potential solutions. Then it enters an iterative procedure to search the optimal solutions by application of genetic operators: crossover, mutation, transformation, and evaluation/rank, as described in Fig.2. Crossover is a reproduction technique that takes two parent chromosomes and mates them to produce two child chromosomes. Mutation is used to explore new regions of design space and for population diversity. Transformation is then performed, which is different from traditional genetic algorithms, to transfer the solutions which have conflicted matches in them to a valid cover. Another distinct operator is evaluation/rank, which can switch to different evaluation models for different applications, and then has corresponding fitness functions to define optimal solutions.

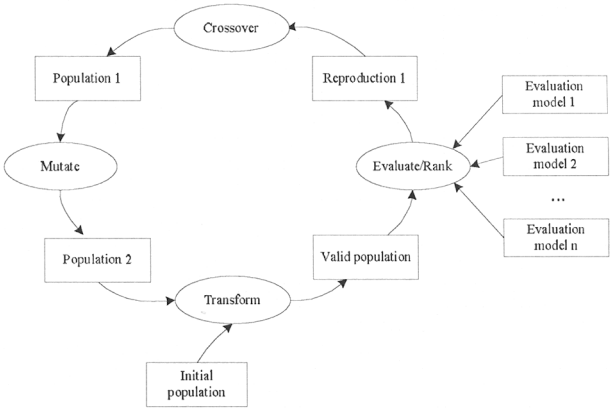

Fig.2 Genetic algorithms for graph covering

B. Encoding

For the graph covering problem, solution or cover is obtained by two procedures: the selection of templates to partition the graph, and the selection of sub-graphs that matches the template to cover the design. Then the chromosome, which encodes the solution, has a two dimension structure, as described in Fig. 3. In the first

dimension an non-zero vector I_i denotes the selection of the template T_i. And in the second dimension binary code 1 in bit j of vector I_i denotes the selection of sub-graph S_j^i of template T_i, and 0 otherwise. Two dimension encoding has the merit of maintaining population diversity after crossover, as the independence of the selection of template and its matches.

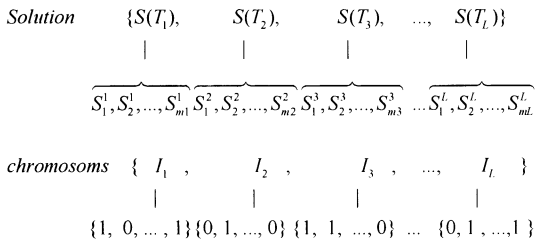

Fig.3 Chromosome encoding

C. *Transformation*

Invalid solution which has conflicted matches will be produced after crossover and mutation in the genetic algorithms. Transformation is used to remove the conflicted matches by an iterative algorithm called Tabu Search. It is a parallel procedure and all individuals in the population will be transferred to valid solutions at a time. The inputs of Tabu Search are K individuals of population and the constraint graph, which describes conflicted matches. The output is valid solutions that contain $(K+w)$ un-conflicted sets. Cost function f_1 is defined as the number of conflicted edges in the solution. And the neighbor of a solution is obtained by moving or reproducing a match from one set to another in the solution. Tabu Search is presented in Fig.4.

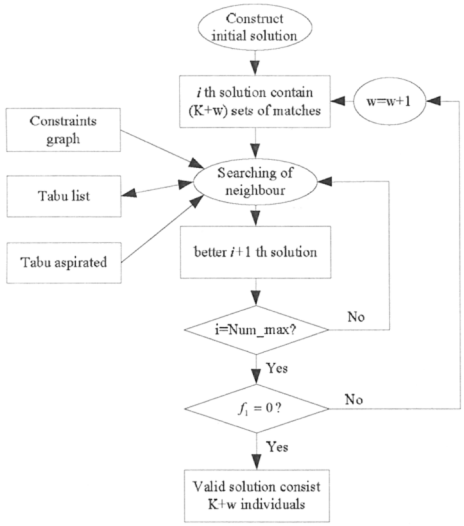

Fig 4. Transformation using Tabu Search

D. *Evaluation Model and Fitness Function*

Evaluation model and fitness function may be varied as to different applications. For simple instruction generation, the number of templates and the number of matches used in a cover are expected to as few as possible. While for system partition, communication model should be considered and matches are scheduled to template resources, and schedule length and area of a cover may be objects. Here are two examples of evaluation models and their fitness functions.

- Evaluation model I: instruction generation: cover the system graph by as fewer templates and their matches as possible.
  Fitness function I: *fitness*=(*number_template* * *number_matches*)
- Evaluation model II: multi-core system partition with time constraint: covering with communication model and time constrained optimization, including allocation of template resources, binding and scheduling of matches, should be considered. Communication interface is added to template resource, and communication time between adjacent matches is inserted. The number of resources and scheduling should be optimized according to time constraint and area object.
  Fitness function II: *fitness*= (area of template and communication resources, under timing constraint).

IV. EXAMPLES

The algorithms of graph covering have been coded by C++. The inputs contain all possible templates, the corresponding matches, evaluation model, and the constraint graph of the matches. The outputs are optimal covers according to the evaluation model. If evaluation model II, as in section III, is used, outputs also consist of the allocation of templates, the binding of matches to template resources, and the scheduling of matches of the best covers.

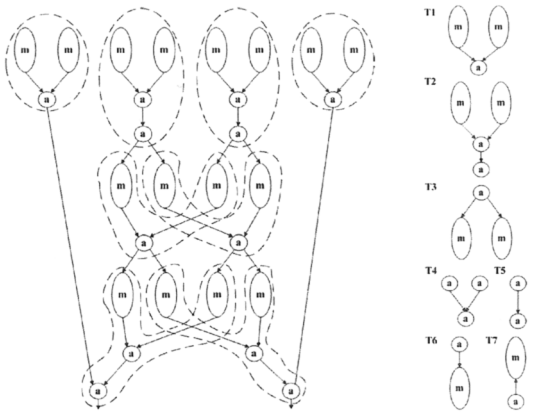

Fig 5. Partition of AR filter

A sample is AR filter which has 26 task nodes as described in Fig.5. The genetic procedures have an initial population size of about 20 individuals, crossover probability rate of 0.6, and mutation probability rate of 0.1. The length of Tabu list in transformation is set at about 4-6. Transformation of initial population takes about 30% total computing time, as chromosomes in the initial population have lots of conflicted pairs of matches. But computing time of transformation in the later iterations is much reduced, as few conflicted relations are induced by crossover and mutation. Two best individuals are propagated directly to next generation without crossover and mutation. And the population converges to the optimal solutions within 100-300 generations. With evaluation model I, the algorithms can get optimal covers as in Table 1, which are the same with the results in [6]. Moreover, if evaluation model II is considered, which can be divided into the communication model and the non-communication model, with time constraint or without time constraint, best covers can also be achieved by the same procedures, just with different fitness functions. The results for procedures using evaluation model II are presented in Table 2. As can be viewed that the selection of templates and their matches are changed due to different models or constraints.

Evaluation objects	Cover	Nodes left
Min(n_T*n_S+ n_left/n_num)	T1: 8 matches	4 add
Min(n_T+ n_left/n_num)	T1: 8 matches	4 add
Min(n_S+ n_left/n_num)	T1: 4 matches, T2: 4 matches	0

Table 1. Covers with evaluation model I

V. CONCLUSION

This paper proposed an iterative graph covering method by genetic algorithms for template based system partition. The algorithms can be applied to different applications, such as multiple processing unit partition in SOC as well as hardware and software instruction generation for ASIP, by just using the different fitness functions or evaluation models. The experiments show that the genetic algorithms are effective to search the optimal solution space and the best solution can always be found if computing time is not critical. And this is the first time to consider optimal graph covering for time constrained system partition with communication models, which consists of template allocation, sub-graph binding and scheduling.

Comm.	Time constraint	Best cover (min(area))	Resources	Area/ Scheduling legth**
NCOMM*	Infinity	T1: 8 matches	1-T1_r	20/33
COMM	Infinity	T1: 8 matches	1-T1_r,1-Comm_r	28/34
NCOMM	14	T1: 4 matches T2: 4 matches	2-T1_r, 2-T2_r	52/14
		T1: 4 matches T3: 4 matches T5: 2 matches	2-T1_r, 2-T3_r, 2-T5_r	52/14
	17	T1: 8 matches	2-T1_r	32/17
COMM	16	T1: 4 matches T2: 4 matches	2-T1_r, 2-T2_r, 4-Comm_r	68/16
	17	T1: 4 matches T3: 4 matches T5: 2 matches	2-T1_r, 2-T3_r, 2-T5_r, 4-Comm_r	67/17
	19	T1: 8 matches	2-T1_r, 4-Comm_r	50/19
	22	T1: 8 matches	2-T1_r, 1-Comm_r	41/22

* NCOMM: Non-Communication-Model. COMM: Communication-Model.
**Area: add 2, mul. 5 comm. 3, comm. interface 1, time: add 1, mul 3, comm. 1

Table 2. Covers with evaluation model II

REFERENCES

[1] A. Chowdhary, S. Kale, P. K. Saripella, N. K. Sehgal, and R. K. Gupta, "Extraction of functional regularity in datapath circuits," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 18, pp. 1279-1296, 1999.

[2] M. R. Corazao, M. A. Khalaf, L. M. Guerra, M. Potkonjak, and J. M. Rabaey, "Performance optimization using template mapping for datapath-intensive high-level synthesis," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 15, pp. 877-888, 1996.

[3] S. Park and K. Choi, "Performance-driven high-level synthesis with bit-level chaining and clock selection," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 20: Institute of Electrical and Electronics Engineers Inc., pp. 199-212, 2001.

[4] G. Lakshminarayana and N. K. Jha, "High-level synthesis of power-optimized and area-optimized circuits from hierarchical data-flow intensive behaviors," Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 18, pp. 265-281, 1999.

[5] R. Kastner, S. Ogrenci-Memik, E. Bozorgzadeh, and M. Sarrafzadeh, "Instruction generation for hybrid reconfigurable systems," IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers. San Jose, CA: Institute of Electrical and Electronics Engineers Computer Society, pp. 127-130, 2001.

[6] D. Sreenivasa Rao and F. J. Kurdahi, "On clustering for maximal regularity extraction," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 12, pp. 1198-1208, 1993.

[7] M. R. Garey. and D. S. Johnson., Computers and Intractability: A Guide to the Theory of NP-Completeness. NY: W. H. Freeman and Company, 1979.

[8] J. H. Holland, "Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence," Cambridge, Mass. : MIT Press, 1992.