Software-hardware Partitioning Strategy Using Hybrid Genetic and Tabu Search

LI Lan-ying
Dept of Computer Science, Harbin University of Science and Technology
Harbin P. R. China
e-mail: lulu08521@sina.com,

SHI Min
Dept of Computer Science, Harbin University of Science and Technology
Harbin P. R. China
e-mail: shiminw@hotmail.com

***Abstract*—One of the most crucial steps in the design of embedded systems is deciding which components of the system should be implemented in software and which ones in hardware. Inspired by genetic algorithm (GA) and tabu search (TS), this paper puts forward a hybrid strategy (GATS) to solve the software-hardware partitioning problem in embedded system. The main frame of GATS is provided by genetic algorithm and the tabu search is taken as the mutation operator. Here the tabu search is used for the solution space in the process of mutation. And the results show that GATS has multiple starting-points, strong mountain-climbing ability and memory function instead of inferior mountain-climbing ability of GA and the single starting-point feature of TS. The experimental results indicate that GATS is superior to the single GA and TS in terms of both required time and system cost, which testify the effectiveness of GATS and produce better portioning results.**

Keywords- embedded system; software-hardware partitioning; genetic algorithm; tabu search; mutation operator.

I. INTRODUCTION

As the development of integrated circuit, SoC (system on a chip) is becoming the main trend of electric system design. A key phase is partitioning the system specification into hardware and software implementations, such that the design constraints, including the time and power constraints are met and at the same time the system cost is minimized.

It is well known that software hardware partitioning problems are *NP*-complete [1] and are therefore intractable. 1992, *Gupta* etc. [2] developed a software-hardware partitioning algorithm to realize the space searching process of the automatic design. Henceforth, partitioning algorithm with various features is developed. The most typical effective one is heuristic algorithm, mountain climbing method, genetic algorithm [3], simulated annealing [4], tabu search [5] etc.

II. PROBLEM DESCRIPTION

A. Function description

The system is formulated to a series of basic schedule blocks (BSB). And then all the blocks are formed into a control data flow graph (CDFG), which is composed of nodes and arcs [6] [7] [8].The nodes denote the BSBs, and the arcs denote the data flows between BSBs. Generally, the CDFG is a directed acyclic graph (DAG). The partitioning algorithm determines which nodes in the DAG are implemented by hardware and which by software. The termination conditions are satisfied, i.e., the solution achieves a good tradeoff of some constraints, such as power, size, and performance. Otherwise, generate a new partition and evaluate again [9].

Every BSB (i.e. node in CDFG) can receive data from its previous nodes, and send data to its next nodes. The BSB is described by a seven-dimension vector:

$$B_i =< as_i, ts_i, ah_i, th_i, r_i, w_i, pc_i >$$

where as_i and ts_i respectively; denote the cost and the executing time implemented by software, ah_i and th_i denote these implemented by hardware. r_i and w_i are two arrays, which, respectively, denote the incoming and outcoming node sets, and, respectively, store the corresponding communication time sets. pc_i denotes that the node i executes pc_i times repeatedly [10].

According to different constraints, there are two basic problems: optimize the time in terms of cost [11] and optimize the cost in terms of time [3]. In this paper, GATS is used to optimize the cost under the time constraint.

B. Partitioning model

The partition model adopted in this paper is shown in *Fig. 1* [6]. "HA1", "HA2",…, "HAm" represent the hardware nodes implemented by ASIC(application specific integrated circuit) or FPGA (field programmable gate array). The software nodes are executed on a programmable processor (denoted by CPU). All the nodes exchange data through a shared bus, and they share the common memory to store the interim data.

The partitioning problem in this paper can be described as follows [3]:

$$\text{Minimize } C \text{ , Subject to } T <= Time\,\text{Re}\,q \qquad (1)$$

978-0-7695-3336-0/08 $25.00 © 2008 IEEE
DOI 10.1109/CSSE.2008.488

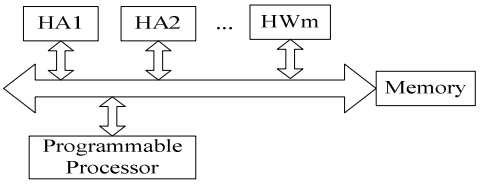

Figure 1. Partitioning model used in this paper

$Time\,\mathrm{Re}\,q$ denotes the time constraint, C denotes the total cost of the system, and T denotes the total executing time. This is a constraint optimization problem. Generally, the required time is given in advance by the designer. The cost of software is usually negligible. So the total cost of the system can be described as

$$C = \sum_{B_i \in HW} ah_i \,, \tag{2}$$

Where $B_i \in HW$ means that node B_i is implemented by hardware.

III. HYBRID STRATEGY OF GA AND TS

Glover, the founder of TS, analyzed and discussed the necessity and feasibility of the hybrid strategy of GA and TS theoretically, which is taken as the theoretical foundation of the hybrid of GA and TS publicly. According to the analysis of GA and TS, a hybrid strategy of GA and TS is proposed in this paper based on the Glover theory, namely GATS, which is used to solve the software-hardware partitioning problem in SoC. TS are taken as GA's mutation operator to improve GA's mountain climbing ability.

A. Fitness function

Construct a general objective function based on the algorithm optimization target, namely the total system executing time T and cost C, and then obtain the fitness function from the general objective function scaling. Turn the executing time as the penalty term in the general objective function where the chosen penalty term should guarantee the constraints satisfaction of the partitioning results, meanwhile the fully utilization of the executing time should also be ensured, there will be no optimum with a too small executing time. The affinity function is similar to the fitness function in Ref. [8]. Two normalization factors σ_c and σ_t are introduced [8]:

$$\sigma_c = CostHw - CostSw \tag{3}$$

$$\sigma_t = \max(TimeSw - Time\,\mathrm{Re}\,q, Time\,\mathrm{Re}\,q - TimeHw) \tag{4}$$

Where *CostHw* denotes the total cost when all nodes are implemented by hardware, and *CostSw* by the software. In general, *CostSw* is negligible. Namely, *CostSw* is equal to 0. *TimeHw* and *TimeSw* respectively denote the corresponding execute time. $Time\,\mathrm{Re}\,q$ denotes the required time constraint. Usually it is a value between *TimeHw* and *TimeSw*. And the affinity function is defined as follows [8]:

$$Fitness = 1/(1 + cobj)\,, \tag{5}$$

where the value of $cobj$ is defined as Eq. (9) [8]:

$$cobj = \alpha \exp\left(\frac{T - Time\mathrm{Req}}{M_i \sigma_t}\right)\left|\frac{T - Time\mathrm{Req}}{\sigma_t}\right| + (1-\alpha)\frac{C}{\sigma_c}\,, \tag{6}$$

In (6), T and C respectively, denote the executing time and the cost of a given partitioning solution. The total executing time T is calculated by the results getting from the program with the data generated by TGFF tools. In Eq. (6), we also adaptively adjust the strength of the penalty item with a factor M_i in ith generation. In experiments, set $M_0 = 1$ and $M_{i+1} = 0.98M_i$ [8]. At the beginning of evolution, M_i is larger in order to keep the diversity of the population. M_i decreases adaptively to give more and more punishment to the offending individuals along with the evolution. The parameter $\alpha = 0.6$ is the coefficient of time and cost for all experiments.

B. GATS algorithm flow

First, give the initial parameters including the max iteration, population size, crossover probability P_c and mutation probability P_m. The initial population is generated randomly by GA, after encoding it using binary, performing selection operation on the population, and then mixing them in a crossover fashion. Finally, TS is taken as the mutation operator. TS part will be executed if the randomly generated probability P is less than P_m. TS involve the concepts like neighbor, tabu list, tabu length, candidate, and aspiration criterion. 0-1 antiposition factor is used to generate the candidate for the mutation operator used in GATS. *Fig. 2* is the flow sheet of GATS algorithm. Here is the algorithm for TS:

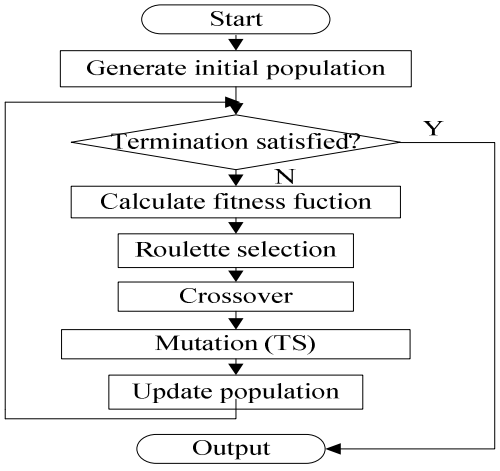

Figure 2. Flow sheet of GATS algorithm

Step1: Choose an initial solution x in S. Set $x^* = x$ and iteration number $i = 0$.

Step2: Set $i = i + 1$ and generate a subset V^* of solution in $N(i,k)$ such that either one of the tabu conditions is violated or at least one of the aspiration conditions hold.

Step3: Choose a best j in V^* and set $x = j$.

Step4: if $f(x) > f(x^*)$ then set $x^* = x$.

Step5: Update tabu and aspiration conditions.

Step6: if the stopping condition is met then stops *else* go to *Step2*.

IV. EXPERIMENT

A. Experiment environment and data

To construct the experimental samples, we generate seven groups of DAGs using TGFF tool (Windows Version 3.1) [12] randomly. There are 30 DAGs in each group, and the average branching factors of the seven DAG groups are 3, 3, 3, 4, 4, 5, and 5. We use the average performance values of the thirty DAG samples as the final performance results.

In our experimentation, we use the following PC configurations: a) Intel Pentium 4 2.80GHz processor, 512M memory. b) Windows XP operating system. c) VC 6.0 programming tools.

TABLE I. DATA OBTAINED FROM TGFF

NumOfNodes	30	40	50	60	70	80	90
TimeHw	4,322	5,608	7,151	8,761	9,845	11,724	13,066
TimeSw	8,764	11,475	14,533	17,684	20,124	23,630	26,417
TimeReq	6,543	8,541	10,842	13,222	14,984	17,677	19,741
CostHw	2,850	3,912	4,532	5,909	6,783	7,446	8,684

B. Experiment results and analysis

In order to make the comparison valid, we use the same controlling parameters for these three types of partitioning algorithms. The initial population is generated randomly, population size of 30, 40… 90 and the corresponding task graphs with node =30, node=40,…,node=90 are used respectively. Other parameters are $P_c = 0.8$, $P_m = 0.01$. In tabu part, the number of neighbors equals to the nodes in task graphs, tabu_length = $\sqrt{n}$, where n is the number of task nodes.

TABLE II. COMPARISON OF GENETIC ALGORITHM (GA), TABU SEARCH (TS) AND GATS

Total DAGs Nodes	Cost			Time		
	GA	*TS*	*GATS*	*GA*	*TS*	*GATS*
30	2,071	2,058	2,056	6,536	6,487	6,457
40	2,845	2,835	2,809	8,541	8,461	8,537
50	3,680	3,573	3,362	10,800	10,675	9,784
60	4,316	4,311	4,310	13,221	13,108	13,056
70	4,944	4,806	4,776	14,933	14,984	14,886
80	5,431	5,248	5,119	17,573	17,673	16,899
90	6,337	6,185	6,042	19,626	19,730	18,189

Under the same experimental condition given above, run the GA, TS and GATS as the software hardware partitioning algorithm using the data from Table 1 separately. The comparison results are shown in Table 2. From Table 2, it is obvious that all the system time are less than the time required, moreover, our algorithm provides smaller system cost and also better fitness value contrast to GA and TS, which is proved in *Fig. 3*.

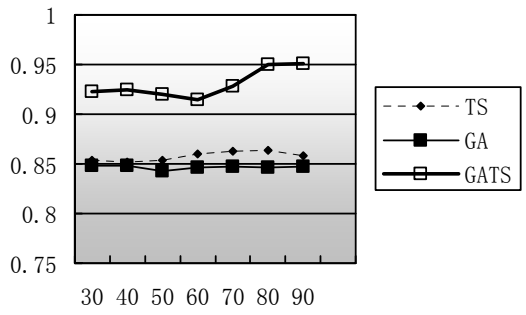

Figure 3. Fitness Value/Nodes Curve

V. CONCLUSION AND FUTURE WORK

Our algorithm takes the advantages of the two algorithms and overcomes their disadvantages. Experiments show our algorithm, namely GATS， excels GA and TS in performance.

This work has only examined the partitioning problem using simulated inputs in the form of directed acyclic task graphs. While this approach allows investigation of our algorithm, GATS, with a variety of problem complexities, our approach needs to be verified on real software-hardware systems. Future work includes the use of realistic benchmarks in evaluating the partitioning algorithm on appropriate hardware platforms (such as transform our algorithm onto the target structure with multiple processor and multiple hardware

components). We also plan to explore other approaches to refine our work.

REFERENCES

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

[2] Gupta R.K, Micheli G.D. System-Level synthesis using re-programmable components. In: Hugo DM, Herman B, eds. Proc. of the European Conf. on Design Automation (EDAC). Brussels: IEEE Computer Society Press, 1992, pp.2-7.

[3] Saha D, Mitra R.S, Basu A. Hardware software partitioning using genetic algorithm. In: Agrawal V, Mahabala HN, eds. Proc. of the 10th Int'l Conf. on VLSI Design. Hyderabad: IEEE Computer Society Press, 1997, pp.155-160.

[4] T. Wiangtong, P. Cheung, and W. Luk. Comparing three heuristic search methods for functional partitioning in hardware-software codesign. Journal of Design Automation for Embedded Systems, 2002, pp.425-449.

[5] Else P, Peng Z, Kuchcinski K, Doboli. A. System level hardware/software partitioning based on simulated annealing and tabu search. Design Automation of Embedded Systems, 1997, 2(1). pp.5-32.

[6] Y. Peng, M. Lin, J. Yang, Hardware–software partitioning research based on resource constraint, J. Circuits Syst. 10, 2005, pp.80-84 (in Chinese).

[7] A. Kalavade, P.A. Subrahmanyam, Hardware/software partitioning for multifunction systems, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 17, 1998, pp.819-837.

[8] Y. Zou, Z. Zhuang, H. Chen, HW–SW partitioning based on genetic algorithm, in: Proceedings of the CEC'04, 2004, pp.628-633.

[9] Yiguo Zhang, Wenjian Luo, Zeming Zhang, Bin Li, Xufa Wang, A hardware/software partitioning algorithm based on artificial immune principles, Applied Soft Computing (Elsevier), January, 2008, 8(1): pp.383-391.

[10] Y. Zou, Z.-Q. Zhang, J.-A. Yang, HW–SW partitioning based on genetic algorithms, J. Univ. Sci. Technol. China 34, 2004, pp.724-731 (in Chinese).

[11] B. Luc, A. Michel, G. Guy, et al., A path analysis based partitioning for time constrained embedded systems, in: Proceedings of the CODES/CASHE'98, 1998, pp.85-89.

[12] R. P. Dick, D. L. Rhodes, and W. Wolf. TGFF: Task graphs for free. Proc. Int. Workshop Hardware/Software Codesign, 1998, pp.97-101.