Optimization of Function Partition between Hardware and Software Based on United Evolutionary Algorithm

Qiaoling Tong, Xuecheng Zou
Huazhong University of Science & Technology
Department of Electronic Science & Technology

Hengqing Tong, Fei Gao
Wuhan University of Technology
Department of Mathematics

Abstract

Partition between hardware and software is a key problem in hardware-software co-design, and global optimums detection of the objective function is of vital importance in hardware/software partitioning. Though stochastic optimization strategies simulating evolution process are proved to be valuable tools, the balance between exploitation and exploration of which is difficult to be maintained. In this paper, the model of the embedded system was constructed by directed acyclic graph to obtain the objective function. Then some established techniques to improve the performance of evolutionary computation are discussed, such as uniform design, deflection and stretching the objective function, and space contraction. A novel scheme of evolutionary algorithms is proposed to solve the optimization problems through adding evolution operations to the searching space contracted regularly with these techniques. A typical evolutionary algorithm differential evolution is chosen to exhibit the performance of new scheme. The improved algorithm can avoid local optimal solution efficiently and be conveniently implemented in the field of hardware/software partitioning.

1. Introduction

Embedded system requires hardware/software co-design, and hardware/software partitioning is a crucial technique of the hardware/software co-design. Embedded system is composed of one or more microprocessors, memory and other components. With the enhancement of function, the system structure has become increasingly sophisticated, and the integrity is getting higher and higher. The hardware/software co-design has overcome the traditional time and cost consuming problems, considered and weighed the hardware and software at the same time in each step, and optimized system architecture [1]. From hardware and software system design space, in accordance with the definition of system functions, the hardware/software partitioning tends to meet the optimal realization of time, cost, and power consumption [2] [3]. This creates an optimization problem.

Many optimization algorithms have been applied to the hardware/software partitioning, such as climbing method, Evolutionary Algorithm(EA), Genetic Algorithm, Ant Algorithm, simulated annealing, chaos optimization algorithm, and so on [4][5]. These methods have their own characteristics in the actual use. In this paper, a Novel United Evolutionary Algorithm Scheme is proposed to achieve the optimized computation of hardware/software partitioning.

2. Model of hardware/software partitioning

In the hardware/software co-design, the function of embedded system is described by the C language or other process languages at first. Then extract the system call graph in accordance with the requirements. Call Graph normally use directed acyclic graph (DAG), as Figure 1 shows below. In DAG, the *T (T1, T2,⋯, T9)* is the node that represents the task; *E (e12, e23, ⋯ , e89)* is the directed edge between the nodes, and it denotes the relationship between the nodes. There are data access and transfer relations between the two linked nodes.

The current study is targeted mainly at dual division, that is, there is only one embedded processor in the system (operative software) and ASIC, FPGA, or other hardware (operative hardware). Embedded processors and hardware modules communicate through shared memory or exclusively channel. In addition, we assume that the communication time between the hardware and software has been included in the respective task time.

DAG nodes in each task have three attributes: a. tsi, time required for software implementation of task nodes; b. thi, time required for hardware implementation of task nodes; c. chi, hardware costs required for hardware implementation of task nodes. Each border of DAG has an attribute: the times of task node i called by his father node j number, cij. The total times that node i is called is the sum of all father nodes call the node i in each possible path, so the number of calls is:

$$C_i = \sum_{k=1}^{E_k} \sum_{j=1}^{V_i^j} c_{ij} \qquad (1)$$

where T_i^j is the set of all the father nodes of node i; Ek calls for all possible paths set. Tsi and Thi represent the total time of software and hardware implementation in node i respectively. Under the assumption that there is no parallelism of hardware and software, we have $T_{si} = t_{si} \cdot C_i$ and $T_{hi} = t_{hi} \cdot C_i$. Then the total implementation time T_{total}^p and total hardware cost C^p are:

$$T_{total}^p = \sum_{i=1}^{V_s^p} T_{si} + \sum_{j=1}^{V_h^p} T_{hj} \quad (2)$$

$$C^p = \sum_{i=1}^{V_h^p} C_{hi} \quad (3)$$

where T_s^p and T_h^p are the set of software and hardware nodes under p partitioning.

The optimal partitioning discussed in this paper is, in a certain price for the hardware constraints, we obtain optimal system performance.

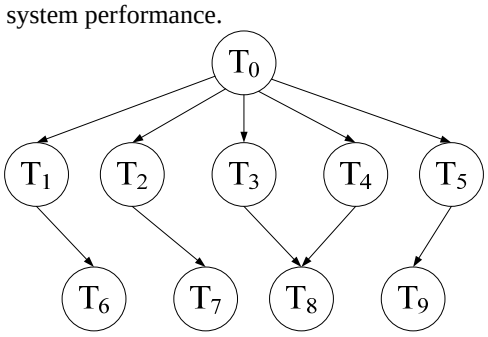

Figure 1. 10-node system task directed acyclic graph

3. Some Established Techniques of EA

In this section we discuss some established techniques of EA such as uniform design, deflection and stretching the objective function, and space contraction[6].

We uses uniform design method to generate the initial population in feasible field so as to have the property of convergence in large scale without better approximation of the unknown parameter as iterative initial point.

Suppose u_{ij} is the element of uniform design table $U_n(N^n)$, $a_{ij} = (2u_{ij} - 1)/2n$, $j = 1, \cdots, N$, then set $P_M = \{a_k = (a_{k1}, \cdots, a_{kN}), k = 1, \cdots, M\}$ contains M points uniformly distributed in $[0,1]^N$.

It is known that set generated by uniform design method is better than by random method statistically in reflecting the distribution property of the objective function just as Figure 2 shows.

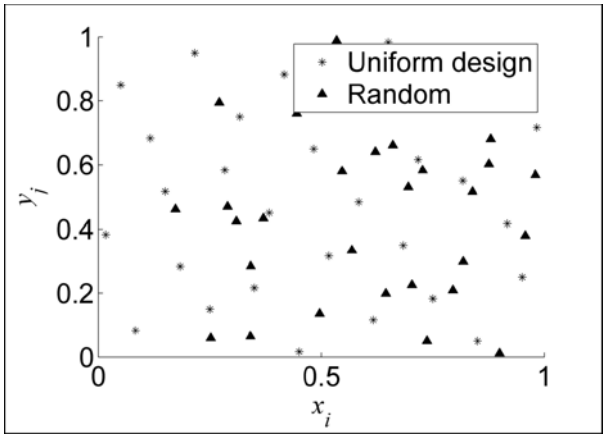

Figure 2. Sets generated by uniform design and random method

Deflection and Stretching

We restrain the normal EA's local convergence limitation virtually through deflection and stretching of objective function.

If the objective function $f(x)$ is full of local optimums and more than one minimizer is needed, we choose another established techniques to guarantee the detection of a different minimizer, such as deflection and stretching are introduced. Suppose objective function is $f(x)$, we use deflection technique as below to generate the new objective function $F(x)$:

$$F(x) = \prod_{i=1}^{k} [\tanh(\lambda_i \| X - x_i^* \|)]^{-1} f(x) \quad (4)$$

where $x_i^* (i = 1, \cdots, k)$ are k minimizes founded, $\lambda_i \in (0,1)$.

We also introduce stretching technique to generate the new objective functions $G(x)$ and $H(x)$ as new objective functions: .

$$G(x) = f(x) + \beta_1 \| x - x_i^* \| [1 + \mathrm{sgn}(f(x) - f(x_i^*))] \quad (5)$$

$$H(x) = H(x) + \beta_2 \frac{1 + \mathrm{sgn}(f(x) - f(x_i^*))}{\tanh(\delta(G(x) - G(x_i^*)))} \quad (6)$$

where $\beta_1 > 0$, $\beta_2 > 0$, $\delta > 0$.

Fig.3 shows deflection and stretching effects on $f(x) = \cos x$ at $x = \pi$. In this way, we see that the searching algorithms will not locate $x = \pi$.

Space Contraction

To avoid exploitation excessively in redundant space and searching efficiency in the whole feasible space, make EA with relative fewer generations as a step, we put a novel technique through a technique space contraction simulating the idea of sequential number theoretic optimization (SNTO) as below.

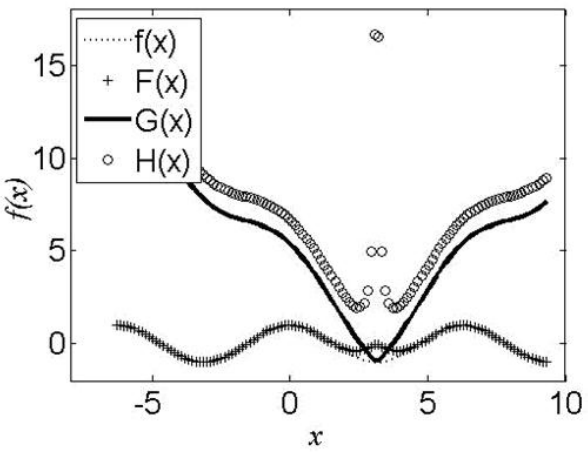

Figure 3. Deflection and stretching effects on f(x)

If a local optimum Q_g is found, we define a new searching space $D^{(t+1)}=[a^{(t+1)},b^{(t+1)}]$ centering Q_g from the current searching space $D^{(t)}=[a^{(t)},b^{(t)}]$ as below:

$$\begin{cases} a^{(t+1)} = \max(x_i^{(t)} - \gamma c_i^{(t)},\ a_i^{(t)}) \\ b^{(t+1)} = \max(x_i^{(t)} - \gamma c_i^{(t)},\ b_i^{(t)}) \end{cases} \quad i = 1,2,\cdots,s \qquad (7)$$

where γ in $(0,1)$ is pre-given contraction ratio. Then we use EA to search in the new space to get a new optimum Q'_g, save the better one in Q'_g and Q_g.

4. A Novel United Evolutionary Algorithm Scheme

With these techniques, we can put a novel United Evolutionary Algorithms Scheme (UEAS) to progress the EAs.

Algorithm 1. United Evolutionary Algorithms Scheme (U EAS)

Step0: Initialization. $t=0, D(0)=D, a(0)=a, b(0)=b,$ $\gamma \in (0,1)$.

Step1: Outer cycle Termination condition Judging. If the optimums wanted are found, output them and terminate otherwise deal with the objective function with the deflection and stretching techniques.

Step2: Generate the initial set $A(t)$ on $[a(t),b(t)]$ by uniform design method, evaluate the fitness and note the best one Q_g.

Step3: Use EA to get a current optimum; update the best one Q_g.

Step4: Inner cycle Termination condition Judging. Given $\delta > 0$ enough small, $c(t)=(b(t)-a(t))/2$, if $\max c(t) < \delta$, then $x(t), M(t)$ are accept, go to Step1, otherwise go to Step5.

Step5: Space contraction. Define a new region $D^{(t+1)}=[a^{(t+1)},b^{(t+1)}]$ by (7) with $\gamma = 0.5$, $t=t+1$, go to Step2.

As many experiment results reported suggest that too many generations do not bring the optimum better than the local one, thus the EA in Step3 of Algorithm1 has relatively fewer generations contrast to the normal EAs, usually 30 to 1000.

Through Algorithm 1, firstly UEAS can detect the objective function's character as far as possible with the initial set by uniform design method. Secondly UEAS can get more optimums and avoid premature through the deflection and stretching techniques. Thirdly UEAS can avoid exploitation excessively in redundant space and search in the most prospective space of the feasible field, so it can jump the local optimum easier. Fourthly with the help of Outer cycle Termination condition Judging UEAS can find all the optimums sequentially. And lastly if EA in Step3 is valid enough, UEAS will not contract the searching space, and in this sense Eas are the special cases of UEAS, then UEAS can combine most of current stochastic optimization strategies such as such as Genetic Algorithms, Evolutionary Programming, PSO, DE algorithm, BOA et al.

Now we choose a typical stochastic optimization strategies DE[7] as the Step3 of UEAS to show its advantages.

DE algorithm grew out of Price's attempts to solve the Chebyshev Polynomial fitting Problem that had been posed to him by Storn[8]. It utilizes $^{M\,n}$ - dimensional vectors as a population for each iteration, called a generation, of the algorithm. At each generation, two operators, namely mutation and crossover (recombination), are applied on each individual, thus producing the new population. Then, a selection phase takes place, where each individual of the new population is compared to the corresponding individual of the old population, and the best between them is selected as a member of the population in the next generation. The details of the DE are given as below.

Algorithm 2. Differential Evolution (DE) Algorithm

Step1: Initialization. Random generate M individuals in feasible region S, $G=0$, crossover constant $CR>0$, mutation constant $CF=0.5$, $G_{\max}$, define a fitness function $f(x)$, value the population and label the best individual in current population as Q.

Step2: DE Evolution. $g=g+1$, for each $x_{i1}, x_{i2}, \cdots, x_{in}$.

(1) Mutation. Random choose four mutually different individuals x_a, x_b, x_c, x_d, in the current population to get a

vector $D_{abcd} = (x_a - x_b) + (x_c - x_d)$, use it to generate new vector as below:

$$\xi_i = Q + CF \times D_{abcd} \quad (8)$$

(2)Crossover.

To get a new testing vector $U_i = (u_{i1}, \cdots, u_{in})$ with ξ_i:

$$u_{ij} = \begin{cases} \xi_{ij}, & if\ (randb(j) \le CR)\ or\ (j = rnbr(i)) \\ x_{ij}, & if\ (randb(j) > CR)\ and\ (j \ne rnbr(i)) \end{cases} \quad (9)$$

where $randb(j)$ is $j-th$ random real in [0,1], $j = 1, \cdots, n$, and $rnbr(i)$ is random integer in $\{1, \cdots, n\}$.

(3) Replacement. Remain the better one between x_i and U_i:

$$x_i = \begin{cases} U_i, if\ f(U_i) < f(x_i) \\ x_i, if\ f(U_i) \ge f(x_i) \end{cases} \quad (10)$$

Step3: Updating. Find the current best Q' and remain the better between Q and Q' as the new Q.

Step4: Termination. If $g > G_{\max}$, then export the Q, else go back to Step2.

Hardware/software partitioning

In the hardware/software partitioning, the optimal partition P can be expressed by a sequences, as $\{X_1(k), X_2(k), \ldots, X_M(k)\}$. $Xi(k)$ denotes the software or hardware partitioned state of the node. This sequence is an optimization variable.

Based on the above analysis, we propose the improved PSO algorithm on hardware/software partitioning as follows:

Step 1: Initialization. Use the uniform design method to generate M individuals $X_i(k)$. Then design the objective function $f(x)$. $f(x)$ can be optimized by above methods based on equation (2) and (3).

Step 2: Calculate the adaptive value and mark best Q_g.

Step 3: Use EA to get a current optimum; update the best one Q_g.

Step4: Inner cycle Termination condition Judging.

Step5: Space contraction. Define a new region $D^{(t+1)} = [a^{(t+1)}, b^{(t+1)}]$. Then go to step 2.

5. Conclusion

In this paper the directed acyclic graph was introduced to model the embedded system and get the objective function for hardware/software partitioning. Then we discussed a Novel United Evolutionary Scheme that can be used in hardware/software partitioning. The algorithm can avoid local optimal solution efficiently by the introduction of translation objective function and stretching objective function. The algorithm also has fast convergence speed. Our further exploration will focus on the impact of parameters on algorithm efficiency. Because our algorithm can be easily implemented by MATLAB, more detailed data examples are omitted here.

The project is supported by National Science Foundation of China (30570611, 60773210).

References

[1] R.Daniel, S.Peter, S.Paul, “A detailed cost model for concurrent use with hardware/software co-design”, Proceedings of the 39th Conference on Design Automation, 2002, pp. 269-274.

[2] Staunstrup J., Wolf W., Hardware/Software Co-Design: Principles and Practice, Kluwer Academic Publishers, Boston, 1997.

[3] Dick P. R., Multiobjective synthesis of low-power real-time distributed embedded systems, Princeton University, Princeton, 2002.

[4] Henkel J., Ernst R., “An approach to automated hardware/software partitioning using a flexible granularity that is driven by high-level estimation techniques”, Very Large Scale Integration Systems, 2001, Vol. 9, No. 2, pp. 273-289.

[5] Saha D., Mitra RS., Basu, “A hardware software partitioning using genetic algorithm”, Proceedings of the International Conference on VLSI Design, 1997, pp. 155-160.

[6] Parsopoulos, K. E., Vrahatis, M. N., “On the Computation of All Global Minimizers through Particle Swarm Optimization”, IEEE Tran. on evolutionary computation, 2004, Vol. 8, No.3, pp. 211–224.

[7] Storn, R., Price, K., “Differential evolution-a simple and efficient adaptive schemefor global optimization over continuous spaces”, Journal of Global Optimization, 1997, Vol. 11, pp. 341 – 359.

[8] Price, K., Storn, R., Lampinen, J. Differential Evolution, “A Practical Approach to Global Optimization”, Natural Computing Series, Springer, 2005.

Qiaoling Tong received his B.Sc. and the M.S. from the department of Electronic Science and Technology in Huazhong University of Science and Technology, China, in 2003 and 2005, respectively. He is currently a Ph.D. student at Huazhong University of Science and Technology. Now he attends in advanced studies in University of California Irv e, USA. His research interests include VLSI and systems, neural networks and intelligent computing.

Xuecheng Zou received his B.Sc in received B.S, M.Sc and Ph.D degrees from Dept. of Electronic Science & Technology in Huazhong University of Science and Technology in 1985, 1988 and 1995, respectively. He is currently a professor and the head of Department of Electronic Science and Technology, Huazhong University of Science and Technology, Wuhan, China. His research interests include Microelectronics and high-speed I/O circuit design.

Hengqing Tong received the B.Sc. in Mathematics from Wuhan University of Technology, China, in1982, the M.S. in Statistics from Huazhong University of Science and Technology, China, in 1987, and the Ph.D. in Statistics from Shanghai Financial & Economic University, China, in 1997. Dr. Tong is currently a professor in the Department of Mathematics at Wuhan University of Technology. His research interests include applied statistics, econometrics, statistical computing, and neural networks and applications.

Fei Gao received the B.Sc. and M.S. in Mathematics from Wuhan University, China, in1999 and 2002 respectively. Dr. Tong is currently an associate professor in the Department of Mathematics at Wuhan University of Technology. Currently he works in a multi-disciplinary environment involving Optimization theories & methods, Computational Intelligence, Bio-informatics, Chaos control & chaos synchronization, and their inter-applied to various real world problems.