Journal of Computer Science 2 (3): 288-291, 2006
ISSN 1549-3636
© 2005 Science Publications

Non-dominated Sorting Genetic Algorithms for Heterogeneous Embedded System Design

[1]A. K. Rath and [2]S. N. Dehuri
[1]Department of Computer Science and Engineering
Krupajal Engineering College, Bhubaneswar, Orissa, India
[2]Department of Information and Communication Technology,
F.M. University, Balasore, Orissa, India

Abstract: The design of complex embedded systems involves the simultaneous optimization of several conflicting and competing objectives. Instead of a single global optimal solution, there exist a set of Pareto optimal solutions. In this study we have used a multi-objective evolutionary optimization algorithms called non-dominated sorting genetic algorithm (NSGA), which will suit to the requirements of designing a complex heterogeneous embedded system. Further, the algorithm is rigorously tested using Video Codec as a case study.

Keywords: Heterogeneous embedded system, Pareto-optimal set, genetic algorithm, NSGA

INTRODUCTION

The general-purpose computers very often do not meet the cost, size and speed requirement for several applications such as control systems in automobiles and air-craft engines, electromechanical systems like elevators and robots, consumer electronic products and numerous home appliances. The computational speed of general-purpose machines is found to be quite inadequate for some applications such as those involving implementation of computation intensive digital signal processing (DSP) algorithms and time-critical reactive systems. Along with the advancement in microelectronics to integrate several millions of transistors, dedicated VLSI chips are developed to fit into the temporal requirement of on-line and real-time applications. In the recent years a dramatic change in trend has been taking place to develop a wide variety of systems comprised of one or more microprocessors, digital signal processors /microcontroller along with application specific integrated circuits (ASICs) or field programmable gate arrays (FPGAs), analog and digital interfaces and software components suitably tailored for a given application, embedded within a larger system. These kinds of systems are called as embedded systems. Embedded systems interact with physical world to perform a single application or a small set of tasks e.g., control, protection and monitoring of processes, machinery, environment and equipments. It is a combination of hardware and software along with necessary interfaces and connections to electromechanical components, sensors and actuators etc. for performing the desired function. Although an embedded system has some software component, it is not itself a general-purpose computer. These systems are embedded usually in a larger system for some specific purpose other than to provide general purpose computing. Embedded systems are often developed with off-the-shelf microprocessors/digital signal processors (DSPs)/microcontrollers and ASICs/ FPGAs for minimizing the cost and development time. In these systems the hardware, software and associated components are optimized for the given application or given set of tasks under the prevailing operating condition keeping the size, cost and performance requirements in view. In the recent years, the design and development of heterogeneous embedded systems has gained tremendous importance and great challenges for its wide and diverse areas of application ranging from home appliances and communication to real-time and distributed control systems in defense/ aerospace missions[1,2].

The design of embedded systems is particularly driven by multiple and conflicting objectives like cost and reliability. Here the user is never satisfied for a trivial solution of both maximal cost and reliability. In contrast, it is more desirable if the user got a solution of minimal cost with high reliability. So this gives rise to a set of optimal solutions, instead of one optimal solution. Since for multiple conflicting objectives no single optimal solution can fulfill the user expectation, we need a tool for optimization. The multi-objective evolutionary algorithms (EA) are the best choice because of its population-based approach. More precisely, EAs maintain a population of structures that evolve according to rules of selection and other operators, which are referred to as genetic operators (such as recombination and mutation).

The objective of this study is to optimize several incomparable and often-competing criterions (cost,

Corresponding Author: A. K. Rath, Department of Computer Science and Engineering, Krupajal Engineering College, Bhubaneswar, Orissa, India

power dissipation, reliability, latency etc.) involved in the design process of complex heterogeneous embedded system.

DESIGN OF HETEROGENEOUS EMBEDDED SYSTEM

The steps involved for the design of heterogeneous embedded systems are as follows:

* System specification
* Software / hardware partitioning
* Software / hardware and interface syntheses
* Validation

The first step of the design process is a specification of the entire system including hardware and software. The hardware and software is partitioned by taking into consideration the desired speed, complexity of the system and flexibility requirements. The hardware, software and interface syntheses is carried out after hardware-software partitioning is completed. These three syntheses are closely and tightly coupled so that if there is any change in one has an instant effect on the other. The validation step is to be carried out after the syntheses step is completed. Figure 1 represents various steps of designing a heterogeneous embedded system[2-6]. Here we have addressed three objectives such as cost, latency and power dissipation for designing a heterogeneous embedded system.

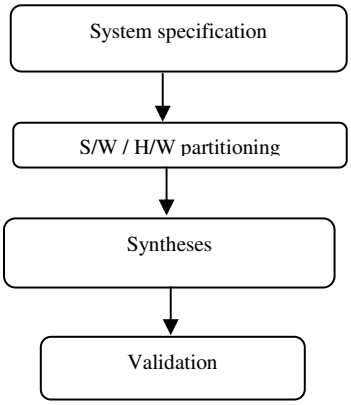

Fig. 1: Flow of heterogeneous embedded system

PARETO OPTIMAL SOLUTIONS FOR MULTIPLE DESIGN CRITERIONS

Consider the design of an embedded system with regard to the two conflicting objectives such as reliability and cost. Both these objectives are very often conflicting to each other and hence, difficult to optimize simultaneously. High reliability architectures substantially increase cost, while cheap architectures usually provide low reliability. In order to optimize the above conflicting objectives we have used multi-objective algorithms. The reason for the optimality of many solutions is that no one can be considered to be better than other with respect to all other objective functions. These optimal solutions have a special name called *Pareto optimal solutions* following the name of an economist Vilfredo Pareto. He stated in 1896 a concept according to his name known as Pareto optimality.

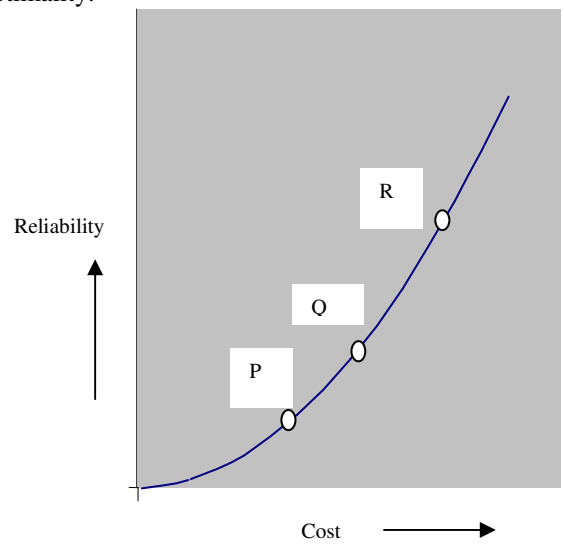

Fig. 2: Pareto optimal solutions

The concept is that the solution to a multiobjective optimization problem is normally not a single value but instead a set of values also called the *Pareto set*. Let us illustrate the *Pareto optimal solution* with the cost and reliability of heterogeneous embedded system architecture.

In Fig. 2 the point 'P' represents a solution, which has both a minimal cost and reliability. On the other hand, the point 'R' represents a solution with high cost and high reliability. Considering both objectives, no solution is optimal. So in this case we cannot say that solution 'P' is better than 'R'. Hence a solution of this type is named as Pareto optimal solution. One cannot sort the solutions, which are belonging to *Pareto optimal set* according to the performance matrices considering both objectives.

OPTIMIZATION THROUGH GENETIC ALGORITHM

Evolutionary algorithm[7] such as GAs are especially suited to this type of problem as they are capable of sampling large and complex search spaces for multiple Pareto optimal solution in parallel. Let us see the working principle of GA.

Since this is a multi-objective optimization problem the simple GA will not work, so in order to avoid that we have used a multi-objective evolutionary algorithms (i.e. called NSGA) proposed by Srinivas and Deb[8]. Non–dominated GAs vary from simple GAs only in the way the selection operator is used.

The crossover and mutation operators remain as usual. For selection, two steps are needed. First, the population is ranked on the basis of an individual's non-domination level and then sharing is used to assign fitness to each individual[9].

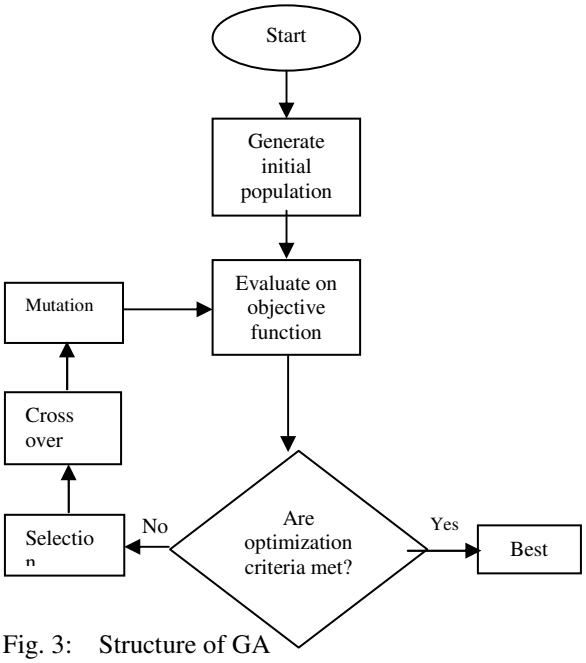

Fig. 3: Structure of GA

Ranking individuals based on non-domination level: Consider a population of size 'n', each having 'm' (> 1) objectives function values. The following algorithm can be used to find the non-dominated set of solutions:

Algorithm

```
for i = 1 to n do
for j = 1 to n do
        if ( j != i) then
        Compare solutions x(i) and x(j) for domination using two
        conditions for all 'm' objectives.
if for any j, x(i) is dominated by x(j), mark x(i) as dominated.
        endif
    endfor
endfor
```

Solutions, which are not marked dominated, are non-dominated solution. All these non-dominated solutions are assumed to constitute the first non-dominated front in the population. In order to find the solutions belonging to the second level of non-domination, we temporarily disregard the solutions of the first level of non-domination and follow the above procedure. The resulting non-dominated solutions are the solutions of the second level of non-domination. This procedure is continued till all solutions are classified into a level of non-domination. It is important to realize that the number of different non-domination levels could vary between 1 to n.

Fitness assignment: The fitness assignment is performed in two stages.

* Assigning same dummy fitness to all the solutions of a particular non-domination level.
* Apply the sharing strategy.

Now we discuss the details of these two stages:

First of all, solutions in the first non-dominated front are assigned a fitness equal to the population size. This becomes the maximum fitness that any solution can have in any population. Based on the sharing strategy, if a solution has many neighboring solutions in the same front, its dummy fitness is reduced by a factor and a shared fitness is computed. The factor depends on the number and proximity of neighboring solutions. Once all solutions in the first front are assigned their fitness values, the smallest shared fitness value is determined.

Thereafter, the individuals in the second non-domination level are all assigned a dummy fitness equal to number smaller than the smallest shared fitness of the previous front. This makes sure that no solution in the second front has a shared fitness better than that of any solution in the first front. This maintains a pressure for the solutions to lead towards the Pareto-optimal region. The sharing method is again used among the individuals of second front and shared fitness of each individual is found. This procedure is continued till all individuals are assigned a shared fitness. After the fitness assignment method, use a stochastic remainder roulette-wheel selection for selecting 'N' individuals. Thereafter apply the crossover and mutation. Shared fitness is calculated as follows:

Given a set of n_k solutions in the k^{th} non-dominated front each having a dummy fitness value f_k, the sharing procedure described in[8] is performed in the following way for each solution i = 1,2,3, .. , n_k:

1 Compute a normalized Euclidean distance measure with another solution 'j' in the k^{th} non-dominated front, as follows:

$$d_{ij} = \sqrt{\sum_{p=1}^{P}\left(\frac{x_p^{(i)} - x_p^{(j)}}{x_p^{(u)} - x_p^{(l)}}\right)^2}$$

Where P is the number of variables in the problem. The parameters $x_p^{(u)}$ and $x_p^{(l)}$ are the upper and lower bounds of variable x_p.

2. This distance d_{ij} is compared with a pre-specified parameter σ_{share} and the following sharing function value is computed:

$$Sh(d_{ij}) = \begin{cases} 1-(d_{ij}/\sigma_{share})^2, if\ d_{ij} \le \sigma_{share}, \\ 0, \qquad\qquad otherwise. \end{cases}$$

3. Increment j. If $j \le n_k$, go to step 1 and calculate $Sh(d_{ij})$. If $j > n_k$, calculate niche count for i^{th} solution as follows:

$$m_i = Sh(d_{ij})$$

4. Degrade the dummy fitness f_k of i^{th} solution in the k^{th} non-domination front to calculate the shared fitness, f_i' as follows:

$$f_i' = f_k / m_i.$$

This procedure is continued for all i= 1,2,.. , n_k and corresponding f_i' is found. Thereafter, the smallest value f_k^{min} of all f_i' in the k^{th} non-dominated front is found for further processing. The dummy fitness of the next non-dominated front is assigned to be $f_{k+1} = f_k^{min} - \epsilon_k$, ϵ_k is a small positive number.

The above sharing procedure requires a pre-specified parameter share, which can be calculated as follows[9,10, 11]:

$\sigma_{share} = 0.5 / p\sqrt{q}$,

Where q is the desired number of distinct Pareto-optimal solutions.

NSGA FOR VIDEO CODEC

Zitzler and Thiele's[12] applied strength Pareto evolutionary algorithms to this problem proposed in does not converge to true Pareto-optimal solutions, because that method uses the fitness assignment procedure, which is very sensitive to concave surface. In order to avoid, we have used NSGA to optimize the conflicts between the three objectives: cost, latency and power consumption. As a case study, we consider the architecture synthesis of a video codec, based on the H.261 standard. The specification of the system including task graph, architecture graph, binding space, communication specifications, etc. can be found in[13]. Figure 4 demonstrates the simulation result. In this study we have used a population size of 100 individuals. The probability of 0.56 and 0.002 is used for recombination and mutation operators. The parameters are optimized after a 50 independent runs. A good compromise solution could be the one represented by the point (350, 40, 154): low power dissipation and good performance at medium cost.

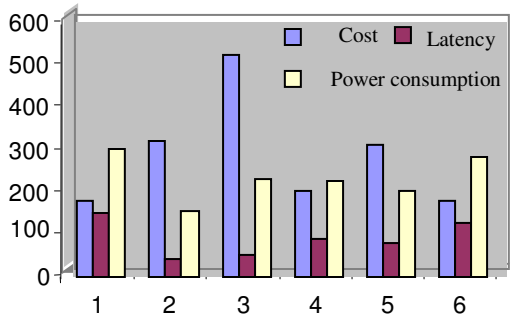

Fig. 4: Simulation result

CONCLUSION AND FUTURE RESEARCH DIRECTION

There are several open questions closely related to the major issues involved in heterogeneous embedded system design. Classically the decisions about the importance of the involved design criteria are made prior to the optimization. In this study, we have done optimization first then design space exploration and finally performed the decision. Even though the number of different Pareto-optimal solutions can be overwhelming in the presence of complex design spaces, the solutions obtained from our simulation results ensure a better distribution of individuals and allows multiple equivalent solutions.

In order to design a complex heterogeneous embedded system the best proposed solution is to take the advantage of a hierarchical approach. For instance, at first the sub-components can be designed and then combine all the sub-components to form the complete design. It would be of major interest to know whether these hierarchical approaches must be reconsidered in the presence of multiple conflicting optimization criterions.

REFERENCES

1. Koopman, P., 1996. Embedded system design issues-The rest of the story. Proc. of the 1996 Intl. Conf. on Computer Design, Austin.
2. Rath, A.K. and P.K. Meher, 2001. Embedded system design: Current issues and perspectives. Computer Sci. and Inform., 31: 8-18.
3. Rath, A.K., 2004. Core-based design of embedded DSP system. Ph. D. Thesis, Utkal University, Bhubaneswar.
4. Balarin, F. *et al*., 1997. Hardware-Software Co-design of Embedded Systems: The Polis Approach. Kluwer Academic Press, Boston.
5. Kalavade, A. and E.A. Lee, 1993. A hardware-software co-design methodology for DSP applications. IEEE Design & Test of Computers, 10: 16-28.
6. Gupta, R.K., 1995. Co-synthesis of Hardware and Software for Digital Embedded Systems. Vol. 329, Kluwer Academic Publishers, Boston.
7. Ghosh, A. and S.N. Dehuri, 2004. Evolutionary algorithms for multicriterion optimization: A survey. Intl. J. Computing and Information Sci., 2: 1.
8. Deb, K., 1999. Multi-objective genetic algorithms: problem difficulties and construction of test problems. Evolutionary Computation J., 7: 205-230.
9. Deb, K. and D.E. Goldberg, 1998. An investigation of niche and species formation in genetic function optimization. Proc. of the Third Intl. Conf. on Genetic Algorithms, pp: 42-50.
10. Goldberg, D.E., 1998. Genetic algorithms for search, optimization and machine learning. Reading, MA: Addision-Wesley.
11. Deb, K., 2002. Multi-objective Optimization using Evolutionary Algorithms. John Wiely & Sons, Ltd.
12. Eisenring, M., L. Thiele and E. Zitzler, 2000. Conflicting criteria in embedded system design. Computer Engineering and Network Laboratory, Swiss Federal Institute of Technology Zurich, Gloriastrasse 35, 8092 Zurich, Switzerland.
13. Blickle, T., J. Teich and L. Thiele, 1998. System-level synthesis using evolutionary algorithms. Design Automation for Embedded Systems, 3: 23-58.