International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 7, July - 2014

A Differential Evolutionary Approach to Solve the Hardware Software Partitioning Problem

Dolly Das
Computer Science and Engg. Deptt
Raipur Institute of Technology
Raipur [C.G.]

Mr. Love Verma
Assistant Professor
Information Technology Deptt
Raipur Institute of Technology
Raipur [C.G.]

Anita Das
Lecturer
Computer Science and Engg. Deptt
Columbia Institute of Technology
Raipur [C.G.]

***Abstract*— The partitioning problem rose out to be one of the crucial problems in the design of an embedded system as the overall cost and delay of the system depends sturdily on this issue. The paper presented here focuses on an approach to solve the hardware software partitioning problem using differential evolutionary algorithm. The novelty of this approach is its enhanced efficiency and the quality of the result in a given constraint. The results are also compared with particle swarm optimisation algorithm and the results of the experiment show the proposed algorithm's effectiveness and efficiency in finding the near optimal solution even for large number of tasks.**

Keywords—Hardware software partitioning, differential evolution, particle swarm optimization

1. INTRODUCTION

An embedded system is a computer system with a dedicated function within a larger mechanical or electrical system, often with real-time computing constraints. . It is *embedded* as part of a complete device often includes hardware and mechanical parts. It is a specialized computer system that is part of a larger system or machine. Typically, an embedded system is housed on a single microprocessor board with the programs stored in ROM. Some embedded systems include an operating system, but many are so specialized that the entire logic can be implemented as a single program.

Modern embedded systems are often based on microcontrollers i.e. CPUs with integrated memory and/or peripheral interfaces but ordinary microprocessors are also still common, especially in more complex systems For example embedded signal processing application uses both hardware specific accelerator circuit as well as general-purpose programmable unit running specific software

Such a combination is very useful as hardware specific accelerator circuit named as hardware works much faster as compared to general-purpose programmable unit running specific software named as software but the cost required for the hardware is much more than that of the software. The amount required to create a software solution and its maintenance is low but the problem with the software is that it work at a slower rate as compared to the hardware

Thus those components which are restricted by performance are realized by hardware , and software is used to realize non critical components. Hence, a good composition between cost and performance can be attained. This has led to the field of hardware software co-design. The problem consists in the discovery of intend of a structure that meets the performance and cost requirements.

The prime part of hardware software codesign [l, 2, 3, 41 is to partition the system into specific hardware and software parts so as to meet the design and real time constraints as well as to minimize the system cost. Thus the real challenge depends on to choose which components of a system should be realized in hardware and which ones in software.

In partitioning, the system's operation is divided in small functional tasks finding the best possible cost and performance trade off with a set of constraints in mind. Since the systems design have turn out to be more and more difficult, studies have been undertaken to automate partitioning as much as possible and hardware software codesign together with partitioning is becoming a rising solution to modern embedded system [10].

With the increment of tasks, finding the best partitioning is becoming more difficult. Therefore use of search and optimization methods has been an appropriate approach. One set of well-liked and influential approaches for search and optimization problems are Evolutionary Algorithms (EAs) [6], [7]. These days EAs are widely used in many engineering applications. A very successful solution for solving the global continuous optimization problem is the Differential Evolution (DE) algorithm. It uses distance and direction information from current population for further search guidance. This paper proposes a DE algorithm technique for hardware software partitioning application.

The remaining sections of this paper are organized as follows: Section 2 gives a review about related work in this area. Section 3 gives an idea of DE algorithm. Section 4 explains chosen hardware software partitioning algorithm and opted partitioning methodology. The achieved results are explained in Section 5. Finally, Section 6 gives the conclusions the paper and also explains the future works.

2. RELATED WORK

In earlier times partitioning was carried out manually. But due to the increase in complexity of the systems, automation of partitioning gained researchers interest. The joint problem of partitioning and scheduling is resolved in Ref [9]. It includes two local search heuristics for partitioning and scheduling differently. Both algorithms operate on the similar graph at the same time. The minimization of the worst case latency of the task graph subject to the area constraints on the architecture is the objective of this technique.

Another solution for hardware software partitioning problem is mixed Integer Linear Program (ILP) method. This is a two-phase heuristic optimization scheme which aims at fast and improved timing estimations by means of continual scheduling phases, as well as using the estimates in the partitioning phases. This approach was sluggish and just realistic for small problems [11], [12], [13]. Arato et al[17] offered a ILP-based approach that worked well for relatively large systems.

Recently Saha et al. [15] modeled partitioning problem as a Constraint Satisfaction Problem (CSP), and have obtained a Genetic Algorithm (GA) based approach to solve the CSP. A. Bhattacharya [5] formulated hardware software partitioning problem as an optimization problem, and multi-agent search algorithm called Particle Swarm Optimization (PSO) can be invoked to discover most favorable result to the partitioning problem.

A. Farmahin and M. Kamal [20] gave another solution for this problem using Discrete Particle Swarm Optimization algorithm. This discrete PSO explores the search spaces step by step and found optimal or near optimal solution, if the algorithm is given enough time.

3. DIFFERENTIAL EVOLUTION ALGORITHM

Differential Evolution (DE) introduced in 1995 has been a competitive stochastic real parameter optimization algorithm. As a standard Evolutionary Algorithm (EA), DE also possesses computational steps. DE perturbs the population members with the scaled differences of distinct population members. Hence, a step-size parameter used in algorithms such as evolutionary programming and evolution strategy is not required to be specified. The strong consistent performance of DE has drawn the interest of numerous researchers.

DE uses the distance and direction information i.e. differential information from the present population to direct its new search. But DE has no method to extort and exploit global information regarding the search space. This algorithm maintains a population of N points in every generation, where each point is a possible solution and N is a control parameter. The algorithm evolves and improves the population iteratively. In each generation, a new population is generated based on the current population. To generate offsprings for the new population, the algorithm extracts distance and direction information from the current population members and adds random deviation for diversity. If an offspring has a lower objective function value than a predetermined population member, it will replace this population member. This evolution process continues until a stopping criteria is met i.e. current best objective function value is smaller than a given value or the number of generations is equal to a given maximum value.

In generation k, we denote the population members by

x_1^k, x_2^k …………… x_n^k.

The DE algorithm is given as follows

DE Algorithm

Step 1: Set k ; = 0 and randomly generate N points x_1^0, x_2^0 …………… x_n^0 from X to form an initial population.

Step2: For each point x_1^k ($1 \leq i \leq N$), execute the DE offspring generation scheme to generate an offspring x_1^{k+1}.

Step 3: If the given stop criteria are not met, set k := k+1, goto Step 2.

The DE offspring generation scheme for generating x_1^{k+1} are as follows :

Step1: Choose one point x_d randomly such that $f(x_d) \leq f(x_i^k)$

Step 2: Generate a trail point $u = (u_1, u_2, \ldots\ldots\ldots u_n)$ as follows

Step 2.1: DE Mutation

Generate a temporary point z as follows:

$$z = (F + 0.5) * x_d + (F - 0.5) * x_i + F * (x_b - x_c) \quad (1)$$

where F is the control factor

Step 2.2: DE Crossover

For j € S, u_j is chosen to be z_j : for j not an element of S, u_j is chosen to be $(x_i^k)_j$

Step 3: If $f(u) \leq f(x_i^k)$, set $x_i^{k+1} := x_i^k$.

S is determined by the probability of crossover which is a parameter of the DE algorithm

The point z is generated by combining the current point x_i with a better point x_d which is randomly selected from the current population and a randomly sampled vectors differentials $(x_b - x_c)$

The advantages of the utilization of sampled vector differential are as follows:

i. Since the mean of the distribution of differentials is always zero there is no sampling bias. This is helpful in maintaining the population diversity

ii. The standard deviation of the differentials could change along with the size and shape of the population in the search space which is valuable for problems whose parameters exhibit vastly in different ranges and sensitivities

iii. The scheme works well as a local optimizer since the differentials in a converging population will eventually tend to zero.

Because of this ability of maintaining the diversity and to do the local search, the De algorithm performs better than other EA's. But DE algorithm has no mechanism to directly use global information about the search space to steer the population towards potential areas.

4. HARDWARE SOFTWARE PARTITIONING

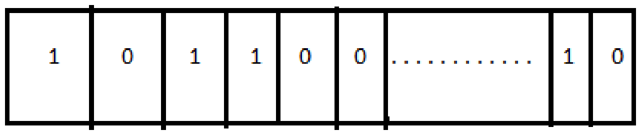
The increase in the usage of embedded system in the market attracted researchers and gained interest in the field of hardware software partitioning

Problem Definition

The formulization of hardware software partitioning problem is given by Arato et al. [8]. The problem is represented by an undirected graph G = (V, E). The set of vertices, represented by V refer to tasks and set of edges, denoted by E refers to the communication between the selected pair of vertices. Attempts were taken to partition the set of vertices into V_H and V_S. Here V_H represents the task to be realized on a hardware and V_S represents the task to realized on software $V_H \cap V_S = \emptyset$ and $V_H \cup V_S$ The partitioning problem can be formally defined as a set of tasks together represent a design, and the problem of finding a partition of hardware and software such that combining them create an equivalent system with a minimal cost , while fulfilling performance constraints. Hence determining which task should be implemented in hardware and which on software in a manner that the whole design meets cost and time constraints.

Problem Representation

Representation of the problem is an important step in DE The method of partitioning is explained as a string of bits for each particle. The number of tasks denotes the length of a particle. . A task must be implemented in hardware or whether it should be developed by software is defined by a bit. Bit 1 represents hardware and bit 0 represents software. An example of a particle is illustrated in fig.



A matrix representing number of solution and number of task is formed which is randomly generated using Matlab code.

Partitioning Methodology

To evaluate the fitness function there is a need to provide some information in the specified constraint. The population size which represents the number of solutions and the dimension which represents the number of tasks along with the number of iterations is provided as input. In the beginning initial population and the trail point are randomly generated. In each generation (iteration), fitness of each particle and an estimation of cost and time is achieved. Then new offspring is generated using (1). This process is continued until termination condition is not reached. Termination conditions are generation number or number of iterations or convergence of algorithm to a predefined fitness value.

Fitness Function

In order to evaluate an obtained solution, we need to know the goodness of a partition which is measured using metrics. The possible metrics are economical cost, performance time, power consumption, silicon area, number of pins, memory size, lines of code, or communications cost. Generally combination of these metrics are combined and a fitness function (co design cost and time estimator) is obtained to guide the algorithm's optimization process.

The focus in our partition is on performance (execution time), area costs and communication costs between software and hardware blocks. Minimizing the system cost while maintaining the constraint requirement for worst execution time is the main objective of our solution. Meeting the time constraint as well as having minimal cost will be the optimal solution for this system.

The fitness of solution is calculated by adding all the four constraint values i.e. hardware cost, hardware execution time, software cost and software execution time. The minimum value is selected as fitness function.

5. EXPIERIMENTAL RESULTS

The partitioning algorithm of the proposed system is implemented using PSO and DE and its performance is evaluated by comparing both the results. Matlab is used for the implementation. Table 1 show the PSO and DE parameters which used in the experiment.

Each algorithm has been run for 1000 generations and the Table 1 and Table 2 reports the outcomes. There are four variables i.e. population size, dimension, PSO execution result and DE execution result. in Table 1 dimension is kept constant and Table 3 population size is invariable the population and the dimension depicts here the inputs of the above described problem.

Population Size	Dimension	PSO	DE
10	30	15574	12676
15	30	16341	13422.2
20	30	16181	13196.8
25	30	16281	11999
30	30	16320	11593.8

Table 1

Population Size	Dimension	PSO	DE
10	10	5249	2794
10	15	4154	2804
10	20	6850	4118.4
10	25	6338	3626.2
10	30	5123	3193.8

Table 2

The result shows that the proposed system gives best result for DE algorithm. In both the above tables readings show that the result derived from DE is always best in compared to PSO. The results are diagrammatically represented in figure 1 and figure 2

The problem of hardware software partitioning give an optimum result when accompanied with differential evolutionary algorithm. the graphical representation shows the outperformance of proposed algorithm

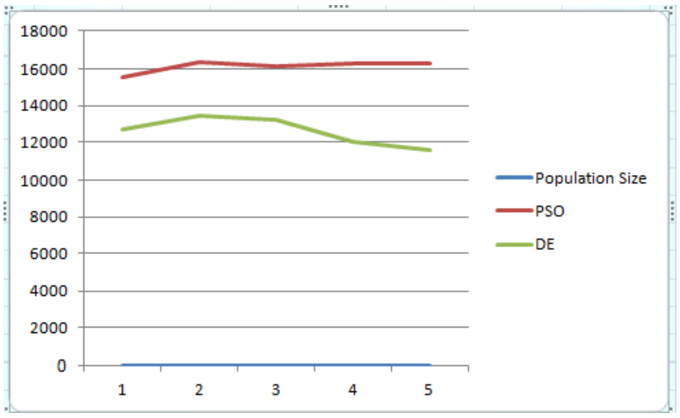
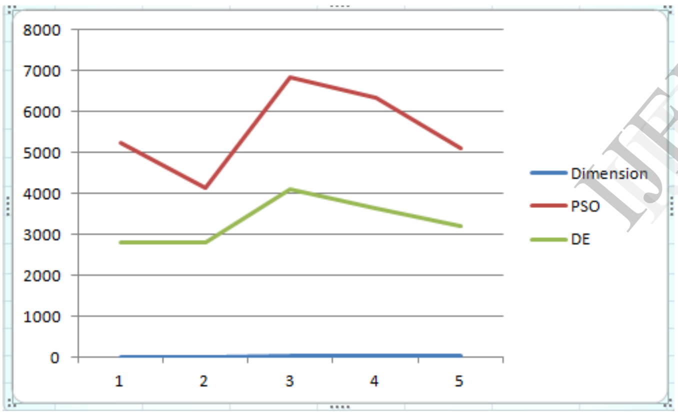


Figure1.When dimension is constant



Figur2. When population size is constant

7. CONCLUSION

In this paper an approach is proposed for hardware software partitioning problem using differential evolutionary algorithm. The performance of differential evolutionary algorithm is competitive to PSO and is able to find optimum solution for the above described problem. But there is still scope in futher enhancements in this project and still research opportunities in this field and more optimum results.

8. REFERENCES

[l] R. K. Gupta and G. D. Micheli,“ Hardware Software cosynthesis for digital systems,” IEEE Design and Test, pp. 29-41, Sept, 1993

[2] S. Kumar, J. H. Aylor, B. J. Johnson, and W. A. Wulf, (‘ A framework for hardware software codesign ,” IEEE Computer, pp. 39-45, Dec 1993

[3] D. E. Thomas, J. K. Adams, and H. Schmit, “A model and methodology for hardware software codesign,” IEEE Design and Test, pp. 6-15, Sept, 1993

[4] R. Ernst, J. Henkel, and T. Benner, “ Hardware soft- ware cosynthesis for micro controller^,^^ IEEE Design and Test, pp. 64-75, Dec, 1993

[5] A. Bhattacharya , A. Konar 1, S. Das 1, Crina Grosan and A. Abraham Hardware Software Partitioning Problem in Embedded System Design Using Particle Swarm Optimization Algorithm

[6] A. E. Eiben, and J. E. Smith, Introduction to evolutionary computing, Berlin Heildberg: Springer-Verlag, 2003.

[7] T. Mitchell, Machine learning, New York, McGraw-Hill,

[8] J¨urgen Teich. Digitale Hardware/Software Systeme. Springer Verlag, 1997.

[9] K. S. Chatha and R. Vemuri, “MAGELLAN: multiway hardware software partitioning and scheduling for latency minimization of hierarchical control-dataflow task graphs,” in Proc. Intl. Conf. Hardware-Software Codesign and System Synthesis, 2001.

[10] R. Ernst, “Codesign of embedded systems: status and trends,” in Proc. IEEE Design & Test of Computers, 1998, pp.45-54.

[11] W. Wolf, “A decade of hardware/software codesign,” in Computer, pp. 38-43, Apr. 2003.

[12] R. Niemann and P. Marwedel, “An algorithm for hardware/software partitioning using mixed integer linear programming,” in Proc. Design Automation for Embedded Systems, special issue: Partitioning Methods for Embedded Systems, vol. 2, Mar. 1997, pp. 165-193.

[13] R. Niemann, “Hardware/software codesign for data flow dominated embedded systems,” Kluwer Academic Publishers, 1998.

[14] R. A. Wildman, J. I. Kramer, D. S. Weile, and P. Christie, “Multi-objective optimization of interconnect geometry,” in IEEE Trans. On Very Large Scale Integration Syst., pp. 15- 23, Feb. 2003.

[15] K. C. Tan, T. H. Lee, and E. F. Khor, “Evolutionary algorithms for multi-objective optimization: performance assessments and comparisons,” in Proc. Cong. Evo1utionary Computation, May 2001, pp. 979-986.

[16]. F. Glover, E. Taillard and D. de Werra. A user’s guide to tabu search. Annals of Operations Research 41: 3–28, 1993.

[17] P. Arato, S. Juhasz, Z. A. Mann, A. A. Orban, and D. Papp, “Hardware software partitioning in embedded system design,” in Proc. Intelligent Signal Processing, Sept. 2003.

[18] R. C. Eberhart and J. Kennedy, “A new optimizer using particle swarm theory,” in Proc. 6 th Symp. Micro Machine and Human Science , Nagoya, Japan, 1995, pp. 39-43.

[19] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi. Optimization by simulated annealing. Science 220(4598): 671–680, 1983.

[20] HW/SW Partitioning Using Discrete Particle Swarm by A. Farmahini, M. Kamal and S. Mehdi Fakhraie