Hardware - Software Functional Partitioning Of Embedded Systems Using Genetic Algorithms

M.Jagadeeswari[1] M.C.Bhuvaneswari[2]

Abstract

The crucial step in the design of embedded systems is the hardware/software partitioning as it is made at the beginning of the design cycle. Functional partitioning is preferred to structural partitioning because software solutions as well as hardware implementations can be achieved. This paper analyses the cost function for hardware/software partitioning of embedded systems using one of the most suitable heuristic algorithm for partitioning (i.e.) Genetic Algorithm The algorithm operates on functional blocks for designs represented as Directed Acyclic Graph (DAG) with the objective to obtain a Hardware or Software implementation that meets performance requirements with a reduced design cost. Results show that on increasing the timing constraint, the cost of the system reduces to minimum with performance tradeoff.

Key Words: Embedded systems, hardware/software Functional partitioning, cost function, genetic algorithms

1. Introduction

Hardware/software partitioning is a key problem in the codesign of embedded systems (i.e.) the decision to partition an application into hardware and software execution units. Application of embedded systems includes telecommunications, multimedia systems, consumer products, robotics, DSP processors and automotive control systems. Application specific integrated circuits (ASIC) are used whenever the performance cannot be met by general purpose processors [1]. But the completely application specific systems are too expensive in terms of design cost and time. Hence they can be thought of as software working on a hardware platform [2].

Genetic algorithm has the capability to converge to global optimum by using initialization, crossover, mutation and objective function (fitness function) even for larger systems. Specialty of genetic algorithm is that non-valid individuals are also allowed but are punished by the fitness function.

The input to the system is a task graph made up of functional units. The functionality is broken into functional portions of some granularity. In any embedded application the internal model can be described as a Directed Acyclic Graph (DAG) or a task graph [1], G (V, E), where V is the set of tasks (nodes) and the edge E, representing the data dependency between the two tasks with $V \in \{v_1, v_2, \ldots v_i\}$ and $E \in \{e_1, e_2, \ldots e_i\}$, which describes the data exchange between the nodes. These portions along with the additional information such as data shared between portions make up the internal model. This is mapped to either software or hardware by partitioning algorithms that search large number of solutions. These algorithms are guided by estimators that evaluate cost function for each partitioning. The cost function is a function which returns a natural number

[1]AP/ ECE Sri Ramakrishna Engineering College, Coimbatore.

[2]AP/EEE, P.S.G./College of Technology, Coimbatore.

Email: jagadee_raj@rediffmail.com

that summarizes the overall goodness of a given partition. The smaller the value of the function, the better is the implementation. The output is a set of functional portions to be mapped to software and others to be mapped to hardware with minimum cost satisfying the performance constraint.

However the feasibility of these methods depends exclusively on the accuracy of the estimators (fitness function) selected for optimizing execution time, hardware size, power consumption etc. Fig 1 shows the configuration of a partitioning system [3]. The crucial step in the design is to identify which task should be implemented in hardware and which in software by using genetic algorithm.

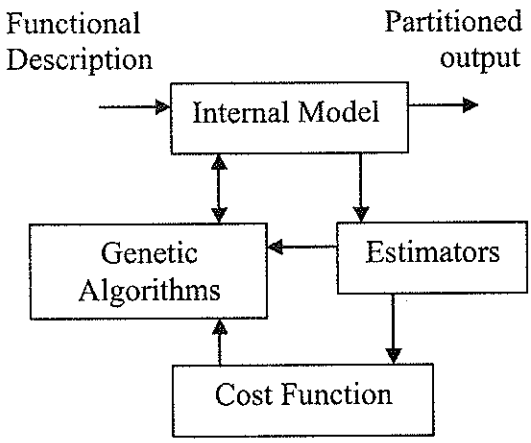


Figure 1: Basic Partitioning System

2. Related Works

In [1] the essential criteria needed for partitioning was described with no specific algorithms for partitioning. In [3] certain partitioning issues without specific algorithm are discussed. Presently only a few algorithms are available for hardware / software partitioning. Ernst in [2] presented a software oriented approach. Simulated annealing was used to find out the optimized combination of hardware-realized and software-realized functions to get maximum performance under the given system cost constraint yielding a long run time. In [4] a hardware software algorithm was proposed which finds a feasible hardware software/software partition which leads to the minimum hardware cost design. Firstly whole functions were implemented by software then gradually moved to hardware to find a reduced system hardware cost.

In [5] a binary-constraint search algorithm is used for minimizing the hardware. Hardware Software codesign for DSP applications are discussed in [6]. In [7] an algorithm was described which starts with most functionality in hardware and which then moves portions to software as long as such a move improves the design. In [8] an approach was described which starts with all functionality in software and then moves portions into hardware using an iterative improvement algorithm such as simulated annealing. Problem solving using high level metrics are discussed in [9].

In this paper a genetic algorithm based approach to minimize the cost while meeting performance constraint for a simple embedded system is discussed. When constraint is imposed on the system, the best cost of the system reduces. Analysis of the system with respect to final cost and final performance was done and is also presented in the graphical format.

The outline of this paper is as follows. In section III the problem is defined. In section IV the heuristic algorithm used for the partitioning approach is explained. Analysis and the experimental results are presented in section V and the conclusion is provided at section VI.

3. Partitioning Problem

An embedded system represented in the form of a Directed Acyclic Graph is taken for analysis. Each Vertex (node) in the graph G is a block or component of an architecture associated with four non-negative numbers with t_{Hi} as Hardware-performance of i^{th} block, t_{Si} as the Software-performance of i^{th} block, c_{Hi} as the Hardware cost of i^{th} block, and Communication Costs [4]. There are tools to

transform an algorithm expressed in a high-level language to data flow graph, example CodeSync, SUIF, RAISE etc. Communication costs are considered only when the components of the system have different implementations (Hardware or Software). The software cost is neglected as it is assumed that there is sufficient memory for programming. Design constraints are selected suitably so that the constraint is higher than the performance time of hardware and software.

To assert the applicability of the above procedure, an industry designed UMTS receiver component, the Delay Profile estimator (DPE) has been used for analysis from [9]. The DPE consists of Nine functional blocks as shown in figure 2.

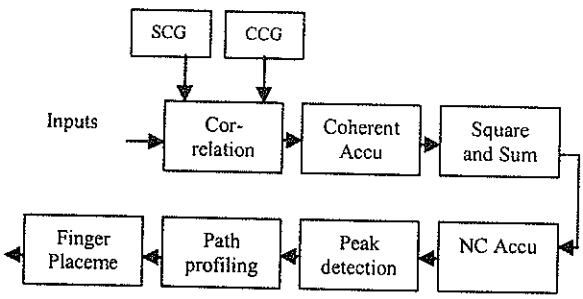


Figure 2: Block Diagram of DPF

Two blocks generate the scrambling (SCG) and Channelisation codes (CCG). The received input signal is correlated with the outputs of the generators and accumulated, squared and summed. The non-coherent accumulator (NC Accu) block removes spurious peaks out of the data stream. After that peaks are detected by checking against a threshold. The results are fed into the path profiling and the correct placement of the finger is established. (Finger Placeme) as shown in the block diagram. The algorithm decides which of these blocks to be implemented in hardware and which in hardware by imposing hardware performance constraint and obtaining an overall minimal cost system.

The study is made by taking the circuit from jam1.chs obtained from [4]. The circuit as such is not used for analysis but its graphical representation and its associated input file is used as shown in Fig.3.

Estimation values for the circuit in Fig.3 are shown in Table 1. The data exchange between nodes is indicated in Table 2.

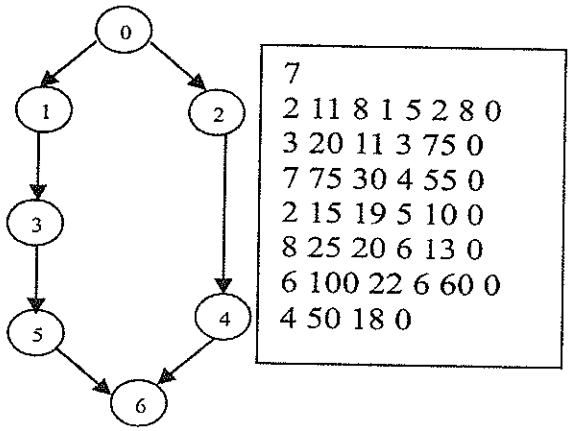


Figure 3: Data Flow Graph & Input File

Table 1 : Estimation Values

Block	T_H	C_H	T_S
0	2	11	8
1	3	20	11
2	7	75	30
3	2	15	19
4	8	25	20
5	6	100	22
6	4	50	18

Table 2 : Data Exchange (Communication Costs)

	B0	B1	B2	B3	B4	B5	B6
B0	0	5	8	0	0	0	0
B1	0	0	0	75	0	0	0
B2	0	0	0	0	55	0	0
B3	0	0	0	0	0	10	0
B4	0	0	0	0	0	0	13
B5	0	0	0	0	0	0	60
B6	0	0	0	0	0	0	0

4. Functional Partitioning Based On Genetic Algorithm

Genetic algorithm is based on Darwinian natural evaluation and selection. This algorithm has four main operations: evaluation, selection, crossover and mutation. The steps used for solving the partitioning problem using genetic algorithm is clearly discussed. The goal of this algorithm is to determine and implement a pattern of software or hardware of each block (node).

a) Encoding, Creation of initial population, population size and Termination condition.

Binary encoding scheme is employed. The *chromosome* which characterizes an individual is defined as $\{b_1, b_2, b_3, b_4 \ldots\ldots b_n\}$, bi€ $\{1, 0\}$, I € $\{1, 2 \ldots .n\}$, where n is the number of blocks in the embedded system. If $b_i = 1$ the corresponding block can be implemented in hardware and if $b_i = 0$ the corresponding block can be implemented in software as in figure 4.

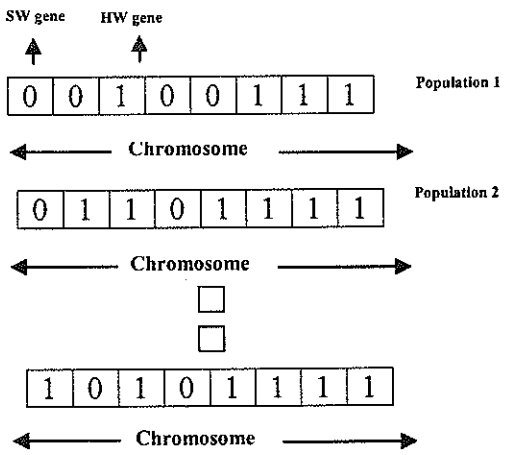


Figure 4: GA Chromosomes for Partitioning Problem

Chromosomes are made up of units called *genes*. A population is formed by individuals. The individual is characterized by a chromosome with an amount of genes equal to the number of functional blocks, with each gene representing a block in the system. The initial population is created randomly to make the algorithm efficient. Population size can be any number greater than the number of blocks in the system. The algorithm is terminated after a certain number of generations (100, 200, etc.) or after achieving minimum cost. In this algorithm the algorithm is terminated after 1000 generations.

b) Fitness Function

An objective function (Fitness Function) is used to guide the algorithm through the optimization process. The value of performance A_i and cost C_i for their inclusion on an objective function is as follows.

- $A = (\Sigma t_{Hi} * \text{gene} + \Sigma t_{Si} * \text{gene})$
- $C = (\Sigma C_{Hi} * \text{gene} + \Sigma C_{si} * \text{gene} +$ Communication Costs)

The objective function is defined as

$$F_i = \begin{cases} K(A\text{-}Tr) + C & \text{if } A >= T_r \\ C & \text{if } A < T_r \end{cases}$$

Where Fi = Fitness Function of member i in the population, A is the total performance of the system, C is the total cost of the system, K, T_r are constants. The software cost of the system is neglected considering that there is enough memory for programming.

In order to ensure that the hardware and software performance does not exceed the time constraint Tr, *K* must be chosen to be significantly larger than *Á*. These cost functions are adapted from those found in [4]. The objective of the optimization procedure is to maximize the fitness values and to minimize the total cost.

For a performance satisfying solution, K = 1000 is assumed. T_r is the Timing constraint imposed on the system. The algorithm uses this function to look for optimal solution. It is a classical objective function that uses a weighted sum of two terms. An optimal solution is a solution with minimal cost.

c) Selection and Evolution Strategy

The selection process is based on Roulette Wheel selection and elite reserving. That is better individuals have larger selection probabilities. Elite strategy is

preserved by adding the highly ranked in the original population to the new population.

Crossover and mutation operations influence the efficiency of the algorithm very much. Two children are reproduced from two parents by first randomly dividing each parent's chromosome into two sets of genes, and then mixing them in a crossover fashion based on the cross over probability *Pc*. This process, known as *crossover*, helps to transmit the good features of parents to the next generation. The efficiency of GA in solving a given problem depends heavily on the representation of desirable characteristics in the chromosome, a good choice of the fitness function, and the appropriate use of the crossover mechanism. One-point crossover and two-point crossover was used with two schemes for choosing the individuals for crossover. In the first case all the individuals are chosen with the same probability. In the second case the probability of choosing the individual depends on the fitness function with a selection probability given by

$$P = F_i / \text{Cum}\,F_i$$

Where $\text{Cum}\,F_i = \sum_{j=1}^{i} F_j$

The better individuals are the individuals with higher probability which are selected for cross over operation. According to the analysis, the best results were obtained when two-point crossover was used with the probability of selection based on the fitness function. Finally, mutation is used to avoid the searching process being trapped in local minima. This is done by occasionally modifying a gene (i.e. mutated) with a small probability value of *Pmu* (e.g. less than 0.1). Fig 5 depicts the general process of Genetic algorithm used for any partitioning application.

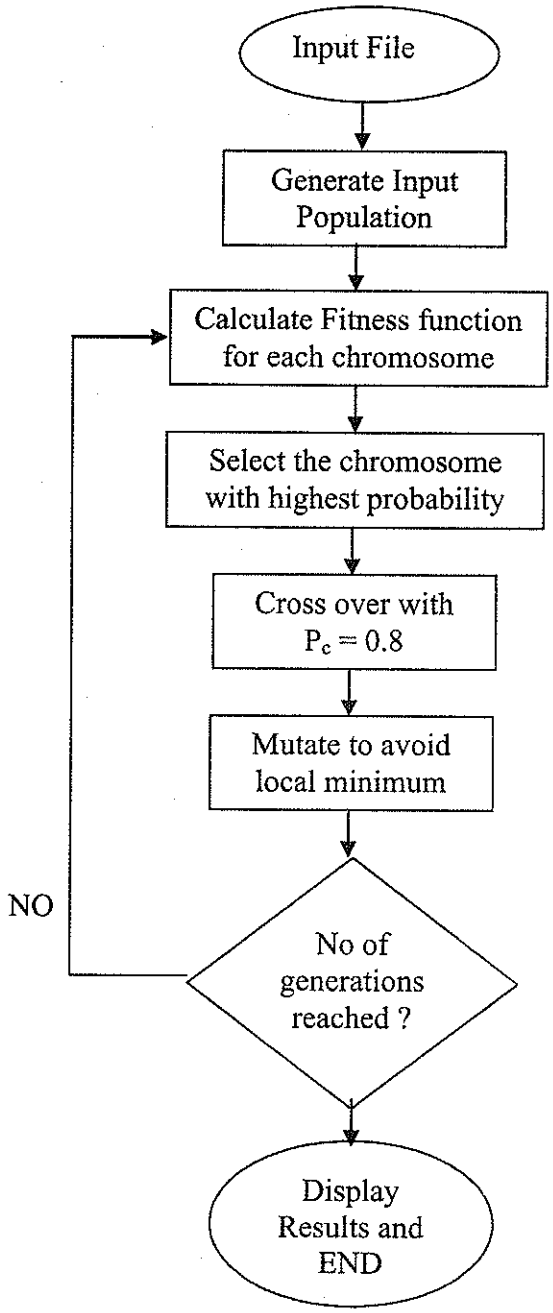


Figure 5: Genetic Algorithm for Partitioning

5. Experimental Result

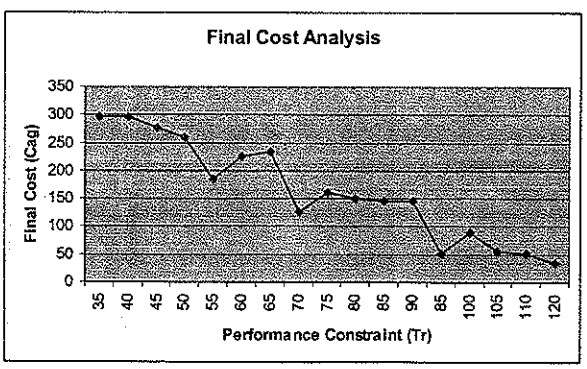
C programming language was used for the implementation of this algorithm. Using the estimated values [4] from table 1 and 2 the analysis was carried out. Table 3 shows the results in terms of Final cost (Cga) and Final

performance (Tga) for several values of performance constraint (Tr).

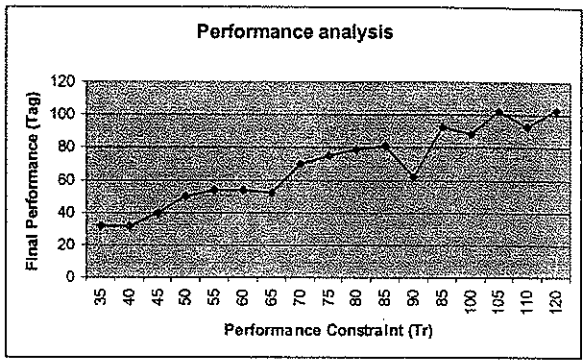
Table 3 : Cost and Time Results

Tr	Cga	Tga	Individuals
35	296	32	1111111
40	296	32	1111111
45	276	40	1011111
50	260	50	0111011
55	185	54	0111101
60	226	54	1011110
65	235	52	0111110
70	126	70	1011100
75	160	75	0010101
80	150	79	0010101
85	146	81	1101010
90	146	62	1111100
95	51	93	1001100
100	90	88	0011000
105	56	102	1100100
110	51	93	1001100
120	35	103	0101000

The results in the table 3 show that the cost of the system is high for all hardware implementations and it reduces on switching to software. On the other hand the Execution time of the system is less for all hardware implementations and is higher when the system is implemented by software.

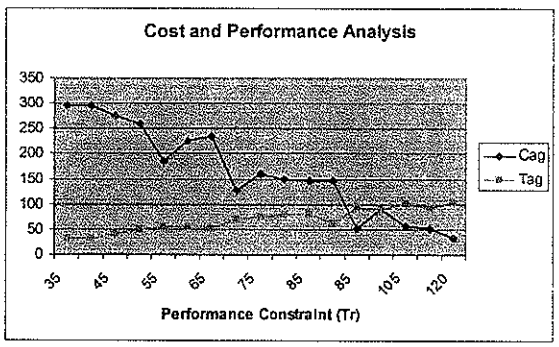


Graph 1 : Final Cost Analysis



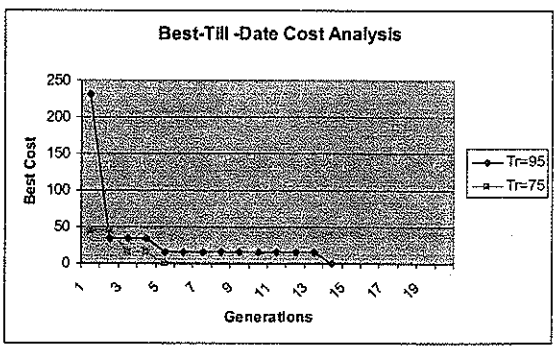
Graph 2 : Final Time Analysis

Graphs 1 and 2 shows that the final cost is improved compared to the initial cost and the final time is slightly higher than the initial time with all the blocks implemented in hardware.



Graph 3 : Cost and Performance Analysis

Graph 3 shows the improvement of cost drastically for smaller variation in performance.



Graph 4 : Best Cost Curve

The best-till-date cost is solution achieved till that particular generation. Graph 4 shows the variation for two different values of performance constraint. It is also

noted that the best solution in each generation is not always better than the earlier generation, but after a few generations, if a new solution is obtained that is better than the previous global best solution, it becomes the best-till –date solution.

6. Conclusion

To improve the efficiency or enhance the outcome of the partitioning problem, genetic algorithm method is used. On the analysis of HW-SW functional partitioning of the embedded systems using genetic algorithm, an implementation with minimum cost was obtained for a circuit with seven functional blocks. Experimental results reveal the algorithms effectiveness in reducing the Final cost of the system satisfying the timing Constraint. Multi-objective optimization strategy of using the weighted sum of various parameters is followed in this paper. The effective ness can also be verified by increasing the population size of the individuals used in genetic algorithm.

The algorithm can also be used for Directed Acyclic graphs with larger nodes and yields better reduction in cost.

References

[1] D.Thomas, J.Adams and H.Schmitt, *"A Model and Methodology for Hardware/software Codesign"*, IEEE design and test of computers, PP. 6-15, 1993.

[2] J.Ernst, J.Henkel and T.Benner, *"Hardware Software Cosynthesis for Microcontrollers"*, IEEE Design and Test of Computers, PP. 64-75, 1992.

[3] Sung Joo Yoo, Jinhwan Jeon, Seang Soo Hong, Kiyoung Choi, *"Hardware– Software Codesign of Resource-constrained Real-time Systems"*, Proceedings of the IEEE, 1996.

[4] J. I. Hidalgo and J. Lanchares, *"Functional partitioning for hardware-software codesign using genetic algorithms"*, Proceedings of the 23rd EUROMICRO Conference on New Frontiers of Information Technology, PP. 631–638, 1997.

[5] Frank Vahid, jie Gong and Daniel D.Gajski, *"A Binary Constraint Search Algorithm for Minimizing Hardware during Hardware/Software Partitioning"*, ACM transactions, PP. 214-219, 1994.

[6] A Kalavade and E.Lee, *"A hardware/Software Codesign Methodology for DSP applications"*, IEEE design and test of computers, 1993.

[7] R.Gupta and G.De Micheli, *"System Level Synthesis using Re-Programmable Components"*, European Design Automation Conference, PP. 2-7, 1992.

[8] R.Ernst and J.Henkel, *"Hardware/Software Codesign of Embedded Controllers based on Hardware Extraction"*, International Workshop on Hardware-Software Codesign, 1992.

[9] B.Knerr, M.Holzer and M.Rupp, *"HW/SW Partitioning Using High Level Metrics"*, International Symposium of Signals, Systems and Electronics (ISSSE), August 2004.

Author's Biography

M.Jagadeeswari received her B.E Electronics and Communication Engineering from Government College of Technology, Coimbatore and ME (Applied Electronics) from P.S.G College of Technology, Coimbatore in the year 1992 and 1999 respectively. She is presently working as Assistant Professor in the department of Electronics and Communication Engineering at Sri Ramakrishna Engineering College, Coimbatore. She is currently pursuing her Ph.D from Anna University, Chennai and is a research scholar at P.S.G Tech Coimbatore. She has published 5 research papers in the National & International Journals/ Conferences. Her research

interests are VLSI design, Hardware Software co-design, Computer architecture and Genetic algorithms.

Dr.M.C.Bhuvaneswari is a faculty in the department of Electrical and Electronics Engineering, P.S.G College of Technology, Coimbatore. She has completed her B.E in Electronics and Communication Engineering in 1986, from Government College of Technology, Coimbatore. She has obtained her Doctoral degree in the area of VLSI Design and Testing, from P.S.G College of Technology, affiliated to Bharathiar University in 2002. She has published 26 research papers in National & International Journals / Conferences. Her areas of interest are VLSI Design and Testing, Computer Architecture, Genetic Algorithms and Fuzzy Logic.