A Multi-objective Task Scheduling Method for Embedded System Design

Mohaddaseh Nikseresht
School of Electrical & Computer Engineering
Shiraz University
Shiraz, Iran
m.nikseresht@cse.shirazu.ac.ir

Mohsen Raji
School of Electrical & Computer Engineering
Shiraz University
Shiraz, Iran
mraji@shirazu.ac.ir

***Abstract*— efficient embedded system design requires considering several design parameters during the task scheduling step. In this paper, a new multi-objective task scheduling method based on genetic algorithm is proposed for embedded systems. In this method, the architecture platform and the tasks in the form of task graphs are given as the inputs of the algorithm. The objective functions in the proposed multi-objective task scheduling include reliability in addition to execution time and energy consumption. The experimental results show that, the proposed algorithm provides better solutions (i.e. scheduled tasks) in terms of all objectives. Moreover, in order to verify the optimization provided by the proposed algorithm, it is shown that the algorithm achieves better solutions in terms of each objective when compared to the solutions obtained by the greedy method. Furthermore, the efficacy of the proposed method is shown in comparison to some well-known single objective heuristic scheduling algorithms where the performance of the proposed method is 29.5% and 21% higher in terms of metrics of scheduling length ratio (SLR) and speeds up, respectively.**

Keywords—Embedded Systems, Task Scheduling, Multi-Objective Optimization, Genetic Algorithm.

I. INTRODUCTION

Embedded Systems are designed and used to control a large and distinct system that is not necessarily computerized. Today, embedded systems are variously used in industrial, military and commercial services and products. On the other hand, the increasing complexity of these systems leads to turn the design of embedded systems to a critical engineering challenge.

One of the main parts of this challenge includes the problem of mapping and scheduling of tasks during the design time of embedded systems. In the scheduling and mapping problem, the functionality of the embedded system is partitioned into a series of software tasks modelled as a task graph. The order of the executing tasks should be determined based on the existing hardware infrastructure. On the other hand, embedded systems are faced several constraints such as energy consumption, reliability, and performance which may be conflicting in some cases. Hence, the scheduling problem should be solved by simultaneously considering these parameters in the form of a multi-objective optimization problem. In this way, a system designer can apply the desired balance between various design options during the mapping and scheduling phase. The task scheduling problem is an NP-hard problem [1]; i.e. by increasing the dimensions of the problem, it is not possible to reach the best answer at a reasonable time. Hence, it is required to use the heuristic algorithms to find the solutions that are as close as possible to the global optimum solution.

Genetic algorithm, inspired by the evolution of humans, is capable of solving the optimization problem with the help of applying genetic operators after modelling the problem in the form of chromosomes. Due to its efficiency and yet the simplicity of its implementation, the genetic algorithm has been widely used to solve various multi-objective optimization problems. Several works based on genetic algorithm have been used to solve the problem of mapping and task scheduling during hardware/software design of embedded systems. In [2], a multi-objective genetic algorithm for the synthesis of hardware/software of distributed embedded systems is presented for the first time. In this work, the system is synthesized considering the energy consumption and the execution time of the tasks. However, the reliability parameter is neglected in this work. In [3], an approach for mapping and scheduling an embedded system based on a genetic algorithm is presented. In this work, the run-time and cost are considered. Additionally, to improve the reliability of the design, the scheduling problem is applied to the modified task graph in which redundant tasks are added to the task graph. This approach does not compute the reliability of the original task graph in the scheduling problem. However, instead, the scheduling is done on the modified graph by imposing a high cost. In [4], an integrated framework for the synthesis of embedded systems is presented at the system level. In this framework, the genetic algorithm applied to improve the multi-objective mapping parameters and various architectural. However, this framework does not consider the design space concerning parameters such as energy consumption and reliability. In another category of works, different genetic algorithms are proposed to improve the execution time of tasks during scheduling for embedded systems. In these works, important parameters of the embedded systems, including energy consumption and reliability, are not considered. In [5], the problem of scheduling is solved to reduce energy consumption. However, the execution time of the tasks and reliability are not considered. Previous works have severe limitations; i.e. some previous works only deal with the problem of mapping tasks. On the other hand, in task scheduling works, the parameters of the execution time, energy

978-1-7281-5937-9/20/$31.00 ©2020 IEEE
Authorized licensed use limited to: KU Leuven Libraries. Downloaded on September 30,2024 at 17:23:47 UTC from IEEE Xplore. Restrictions apply.

consumption, and reliability are not considered simultaneously during the problem of task scheduling in embedded systems.

In this paper, a new multi-objective task scheduling method based on genetic algorithm is presented for embedded system design. In the proposed method, tasks which are modelled as a task graph are mapped and scheduled according to the existing hardware architecture platform. To this end, task mapping and scheduling solutions are modelled in the form of chromosomes while each chromosome is evaluated using different objective functions for the parameters of execution time, energy consumption, and reliability. Pareto solutions are selected during the optimization flow to have a multi-objective optimization. The experimental results show that the final solutions (i.e. scheduled tasks) provided the proposed method is improved in term of all objectives. Additionally, as a baseline, a greedy algorithm-based task scheduling is compared with the proposed method considering the best solution of each objective. The solutions found by the greedy method are dominant by the one provided by the proposed method for each objective (i.e. execution time, energy, and reliability). Moreover, the performance of the proposed method in finding improved solutions in comparison to some well-known single- and multi-objective heuristic scheduling algorithms is evaluated in term of scheduling length ratio (SLR) and speeds up. It is observed that the SLR and speed up of the final solution provided by the proposed method are respectively 29.5% and 21% betted than the ones obtained by the other similar methods.

The rest of the paper is organized as follows: In Section II, the proposed methodology is explained In Section III, experimental results are presented and, finally, the paper is concluded in Section IV.

II. THE PROPOSED METHODOLOGY OF MULTI-OBJECTIVE DESIGN BASED ON GENETIC ALGORITHM

In this section, our multi-objective design methodology based on genetic algorithms is presented in two-part. At first, tasks and hardware architecture model are presented and, in the second part, the design space exploration using the genetic algorithm is considered.

A. Modeling tasks and hardware architecture

For modelling tasks, the model introduced in [6] has been used. This model is based on the Kahn Process Network (KPN), a very well-known computational model for modelling embedded systems. KPN is a method for representing tasks in which tasks are indicated in the form of a directed graph. The KPN is represented by the $G_{AP}(V_{AP}, E_{AP})$ and for each vertex $V_i \epsilon \{1,\ldots,\vee V_{AP} \vee\}$ represents a task or process from the G_{AP} graph. For each vertex V_i the set of tasks associated with this vertex shown in the form of $B_i=\{e_j \epsilon E_{AP}\}$. When Vi is assigned to a hardware component for scheduling, ht_i represents the execute time on this hardware component. If a task can be run on multiple cores, its execution time is shown as the set of $ht_i=\{ht_{i1}, ht_{i2}, \ldots, ht_{iU}\}$. Where U represents the number of hardware cores, a specific task on which can be executed. When a vertex is assigned to a software component for scheduling, st_i represents the execute time on that software component. When that task can run on multiple software components, the execution time is shown as the set of $St_i=\{st_{i1}, st_{i2}, \ldots, st_{iV}\}$. Where V represents the number of software components in which this task can be run.

Each edge $e_j \in \{1,\ldots,\vee E_{AP} \vee\}$ represents the relationship between two tasks of the G_{AP}graph. If this link is allocated to memory,mt_j represents the memory access time. If during scheduling, it can be allocated to multiple memories. The access time is represented as to the set of $mt_j=\{mt_{j1}, mt_{j2,}, \ldots, mt_{jW}\}$, where W denotes the number of memories that the task can be assigned during the scheduling. Also, the architecture model is represented as the graph$G_{AR}(V_{AR}, E_{AR})$ in which the symbols V_{AR} and E_{AR} represent respectively the architectural components and the communication links between them. A set of architectural components consists of two independent sets: a memory set (M), as well as a set of processors (P) that includes hardware and software components$V_{AR}=P \cup M$. A delay between communication links is indicated by$¿_{pq}$, where $p,q \in \{1\ldots|E_{AR}|\}$. The energy consumption is considered during the run for processor p, $w_{pe}$and, $w_{me}$for memory m and communication link$w_{¿}$. In this paper, it is assumed that the architecture platform is available because the main goal is not to solve the problem of determining the ideal architecture, but to propose a method for solving the problem of mapping and scheduling in embedded systems.

The problem of mapping and scheduling is finding the optimal assignment of tasks and communications between them on the hardware architecture platform. An example of mapping and scheduling tasks on the hardware platform is shown in Figure (1). As it can be seen in this figure, each vertex/edge of the graph represents a task/a relationship between two tasks. Figure 1.a shows a mapping process explicitly, for executing a task graph, processors from the hardware architecture platform are allocated to tasks of the graph and communication links map to memories; also two or more tasks may be assigned to the same processor for executing. Following that, fiqure1.b illustrate the scheduling part, the order of processors execution is determined.

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on September 30,2024 at 17:23:47 UTC from IEEE Xplore. Restrictions apply.

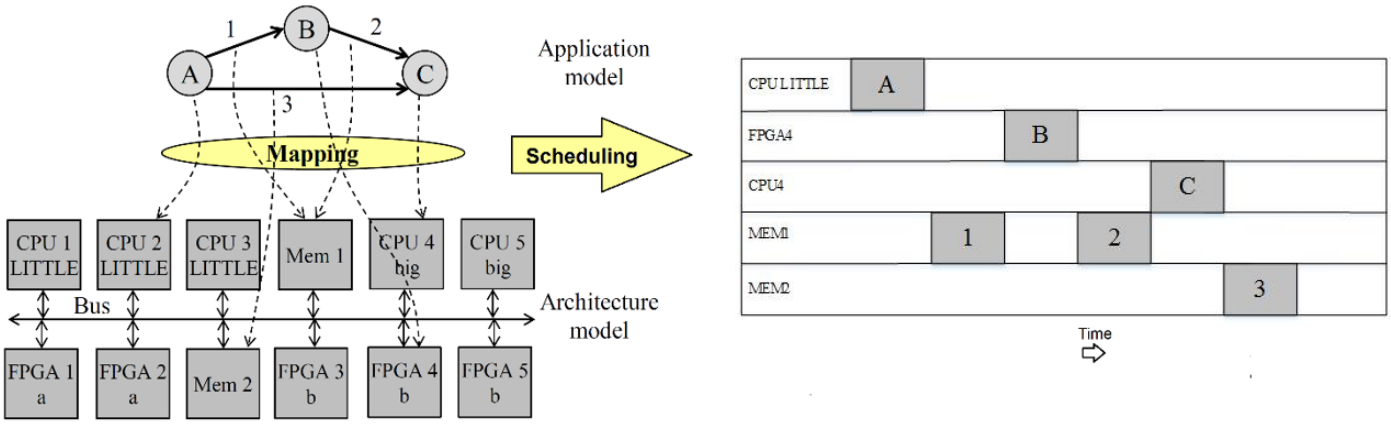

Figure 1. The Problem of mapping [5] and scheduling in the design of Embedded Systems

In this paper, a multi-objective optimization algorithm based on genetic algorithm has been introduced for exploring the best mapping and scheduling concerning the essential objectives of the embedded systems, including execution time, energy consumption and reliability. In the second part, our approach toward finding optimum scheduling has been explained.

B. Design space exploration using the genetic algorithm

The genetic algorithm is used to find the best mapping and scheduling of tasks in the design space. Due to the complexity and limitations in the design of embedded systems, as it said before, the search for the design space is a challenging process, and it is known as an NP-hard problem. In a multi-objective optimization problem, optimization goals are usually in conflict with one another. The scheduling and mapping decisions of each task of application graph are defined, and their community forms a solution of the problem in term of a chromosome to solve our explained multi-objective problem. Meanwhile, each solution should meet all given constraints of the application defined by the designer. Our chromosome representation and genetic-based operations have been explained in the following.

1) Chromosome Representation

In each chromosome, the scheduling model on processors and communication edges must be specified. First of all, based on the segmentation approach described in [4], we divide the scheduling into several segments while each segment includes independent tasks which can be simultaneously executed on different processors considering the dependency graph. However, we assumed the task order to be constant as indexes of each segment and the values inside each segment show the processors selected for executing each task. This different viewpoint makes it possible to eliminate all dependencies in chromosome and convert the in-ordered list problem to disordered list problem, which is a more convenient problem to solve. That is because, by this segmentation approach, all the tasks in each segment can be it simultaneously scheduled without making any difference in the final results. A sample chromosome (for the task and architecture model mentioned in Figure 1) is shown in Figure 2.

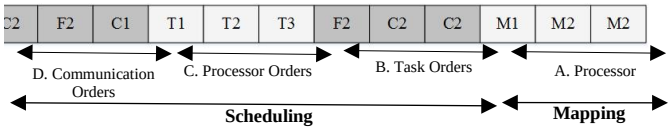

Figure 2.. The Chromosome Representation has been shown.

2) Crossover

As one of the genetic operators, crossover used to combine the genetic information of two parents to generate a new offspring. At first, one of the parents selected randomly, to prevent local optimums, we have developed a local crossover mechanism. According to this method, chromosomes are classified base on NSGA II Pareto set classification [7]. The other parent randomly selected from the same class as the first one. After parents determination, with probability one child is generated by following Figure 3. In this algorithm, for each task of a child, a random integer number between 0 and 10 is generated. If the number was lower than 5, the mother's processor is selected to execute the child's task. Otherwise, the father's processor is chosen. We have marked crossover as in figure 3.

	T8	T9	T12	T6	T7	T0	T1
Father	C3	F1	C4	C5	C2	C0	C0
Mother	C7	C5	C4	C3	C1	F2	C1
Child	C3	C5	C4	C5	C2	F2	C0

Figure 3. Crossover operation

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on September 30,2024 at 17:23:47 UTC from IEEE Xplore. Restrictions apply.

3) Mutation

A mutation is a genetic operator used to maintain genetic diversity from one generation of a population of genetic algorithm chromosomes to the next with probability P_m. For each task with a random distribution, we decide to whatever the child keep the parent processor or choose another one from the processors Range. A sample Mutation operation is shown in Figure 4.

	T8	T9	T12	T6	T7	T0	T1
Parent	C7	C5	C4	C3	C1	F2	C1
	T8	T9	T12	T6	T7	T0	T1
Child	C2	C5	C4	F1	C4	F2	F1

Figure 4. Mutation operation

4) Selection

The selection part is done based on NSGAII [7] algorithm. The pseudo-code of NSGAII is shown in Figure 5.

A. Fitness functions for the objectives

5) Objective one: Execution Time

The following function model is used to minimize processing time on a critical path, the path that includes the maximum executing tasks. The first and second part of the equation represents the effect of hardware processors and software processors on the execution time, respectfully. The last part of the equation shows the effect of links delay due to memory access time if the task communication j is assigned to a memory core. The parameter mt_{jw}, illustrates the memory access time for w$\in\{1,\ldots,M\}$ in which M is the maximum number of memory cores. Additionally, the other parameter, $¿_{kl}$illustrates the latency of the Processor architecture k and l with$k,l\in\{1,\ldots,\vee V_{AR}\vee\}$, where V_{AR} is the maximum value of latency. Finally the variable $¿_{mn}$ represents the communication delay among two cores m and n with m, n $\in\{1,\ldots,\vee V_{AR}\vee\}$.

$$min ¿ \qquad (1)$$

To model which task is assigned to each processor, the decision variables are used. In more details, whatever a task i is assigned to the hardware processor u or software processor v, the decision variables x_{iu} and x_{iv} are used, respectfully. On the other hand to show if a communication link j is linked to a memory core w, or a communication channel is included within a core, the decision variables x_{jw} and x_j have been applied. For example if two communicating tasks are assigned to the same processor x_j value will be zero.

6) O bjective two: energy *consumption*

There is an energy modelling function, the second target function to minimizing energy consumption.

$$min\left\{\sum_{i\in v_{AR}}^{\square} t_p^e w_{pe}+\sum_{j\in E_{AR}}^{\square}\left(t_l^c w_{¿}+t_m w_{me}\right)\right\} \qquad (2)$$

In the first part of the equation t_p^e shows the time passed on the CPU processor and W_{pe} illustrates the power consumption per unit of this processor. On the other hand, in the second part of the equation t_p^e andt_m determine the time elapsed on the Communication link and memory core respectfully. Additionally, the parameters $w_{¿}$ and w_{me} indicate energy consumption by communication link and memory core in order.

7) Objective 3: Reliability

The last objective function is the system's reliability model adopted from methods introduced in [8] and [9]. This function is modeled base on Markov chain technique (DTMC). To compute reliability amount, the DTMC model is taken from the system architecture. The aim is to maximizing this value.

$$R_n \text{R} = \text{S(1,n)} \qquad (3)$$

In the above equation, The S is the base matrix of the DTMC, S (i, j) shows the number of expected times to visit j from the base i where the parameter n is the number of states in the model. To align this objective function with previous ones, it has been rewrite in following fashion:

$$\min \{1\text{-R}\} \qquad (4)$$

Solving the multi-objective optimization problem

After modelling the target functions, in this section, the general form of the multi-objective optimization function written in the form of the following equation:

$$\left(f_1(x),f_2(x),f_3(x)\right)^T = \text{Min(x)} \;\; \text{z=f(x)} \qquad x\in X \;\; .\text{s.t} \qquad (5)$$

In the following equation, x represents an accurate answer from the set of acceptable answers X. In this case, an appropriate response to schedule depends entirely on how application tasks are assigned to the processes of the architecture platform. In a particular scheduling solution, the target functionsf_1, f_2 and $f_3$calculate the expressions from equations (1), (2), and (4). The final function of optimizing $z=f(X)_{\square}$ translate a solution x from the decision space defined by the decision variable to a point in the objective space defined by the three objective or cost functions. In our example, the target space is three dimensional and has the same weights. Which, is obtained using the three functions of equations (1), (2) and (4).

Since the multi-objective optimization problems usually do not have a single best answer to be able to improve all the objectives at the same time, we are interested in finding the set of answers that form the so-called Pareto frontiers. The answers that shape the Pareto solution set are sets of responses that non-dominated by any other solution point among all solutions from the possible set. We use evolutionary algorithms to solve the Genetic Algorithm and generate Pareto solutions due to the

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on September 30,2024 at 17:23:47 UTC from IEEE Xplore. Restrictions apply.

capability of this algorithm simultaneously optimizing several independent objective functions. More specifically, in this paper, the NSGAII algorithm [7] is used. Because this algorithm has shown that, in contrast to other evolutionary algorithms, it has significant advantages such as ease of implementation and low computing energy [7]. The pseudocode of this algorithm is shown in Fig. 5. The genetic algorithm generates repetitively new children in the population from the previous parent solution population using two Crossover and Mutation agents that are embedded in the algorithm. Initially, the algorithm requires the production of a first population set, initial population, which at the beginning of the work we created it for simplicity in a random way.

Input: N population size, M max number of generation
Output: Pareto frontier, non-dominated solutions in P_M
P_0 = GenerateInitialPopulation(); // *Size N*
$Q_0 = \emptyset$; // *Start with children set empty*
EvaluateObjectiveFunction(P_0); *//calculate fitness*
RankPopulation(P_0); *//Done according to fitness values*
For (i=0 to M-1) **do**
// Create children population
Q_i = ¿ SelectionCrossoverMutation(P_i);
// Calculate children fitness
EvaluateObjectiveFunction (Q_i);
P_{i+1} = ¿CombineParentsAndChildren(P_i , Q_i);
RankPopulation(P_{i+1});
//Elitism: Keep non-dominated:
P_{i+1} = ¿SelectNIndividuals(P_{i+1});
End for

Figure 5. Design Space Exploration based on NSGAII[7]

III. Experimental Results

The proposed mapping and scheduling method (MOGATS) is implemented in C programming language and run on a LINUX machine with Intel Core i3 with 2.30 GHz clock frequency and 2 GB RAM. For simulation, we used two test cases, both of which are in the domain of automotive software; i.e. Anti-lock Brake System (ABS) and Adapter Controller System (ACC). These two test cases are adapted from [10]. Since the reliability, energy consumption and the execution time of tasks are shown as objective functions, the only constraint that we used in our problem formulation consists of the architecture platform being given and the HW/SW partitioning of the given application.

In the experiments, the hardware platform is assumed to consist of twelve parts including ten processors and two communication units. The communication arcs in the graph are assumed to be implemented via memory mapping; that is, the source task writes into a memory component, and the destination tasks read from the memory component. In our assumed architecture platform that is similar to the work done in [5] has shown in Figure 1.

The problem of mapping and scheduling, in the form of software scheduling programming on the hardware platform, is considered. Pareto solutions taking into account reliability parameters in the form of unreliability (1 - R), the energy consumption and the execution time in the objective space are considered. The sample of solutions obtained for the two experimental test cases is shown in Fig. 6. The initialization populations and final population respectively have been shown in a circle (red) and triangle (blue). The final result is given after executing 9 million rounds of the algorithm. As can be seen in this figure, there has been a noticeable improvement over its original state. The outcomes are re-displayed using two-dimensional graphs to study the results better. For example, Fig. 7.a compares the results of the two parameters of energy consumption and execution-time in the initialize population and the final population. As you can see in Figure 7, the Pareto set of the final population always covered the initialization population.

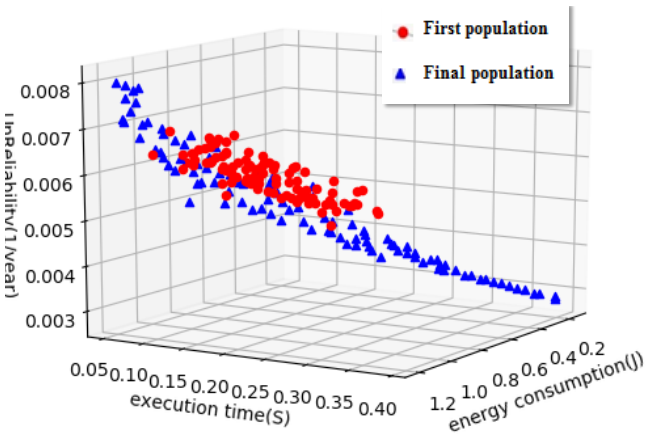

Figure 6. Comparison of initialization Population (circle - red) and Final Population (triangle - blue) using the proposed Multi-objective Genetic Algorithm

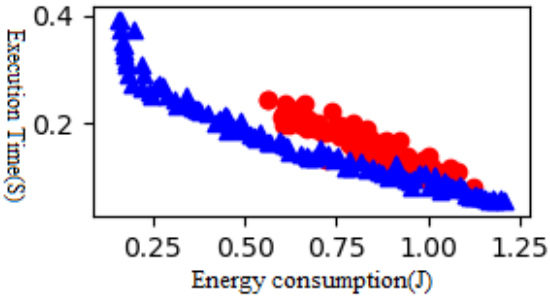

Figure 7.a. Energy consumption – Execution Time Graph

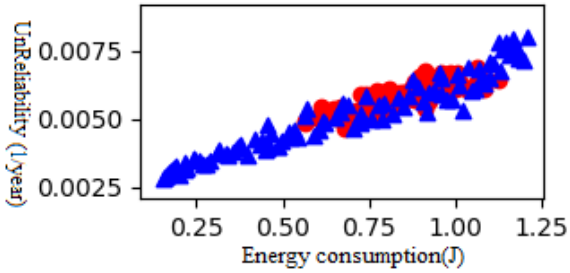

Figure 7.b. Energy consumption – Unreliability Graph

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on September 30,2024 at 17:23:47 UTC from IEEE Xplore. Restrictions apply.

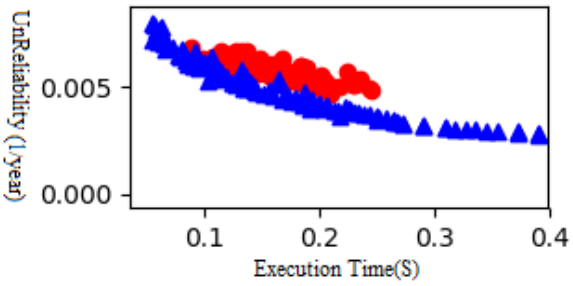

Figure 7.c. Execution Time – Unreliability Graph

Figure 7. Determining the results in the form of two-dimensional charts for better viewing- the initialization Population and Final Population has .been shown respectively in circle and triangle

Table 1 summarizes the results of the experiments comparing the best performance of the proposed method to the greedy-based strategy. The greedy algorithm is an algorithm model that follows the problem-solving heuristic of making the locally optimal choice at each stage with the intent of finding a global optimum. This experiment, ABS and ACC benchmarks were used [12]. To have a fair comparison with our multi-objective proposed method, for each objective function (i.e. execution time, energy consumption, and reliability) the greedy function is executed on the data set individually. In Table 2.a, both methods find the best Execution time, but Greedy method required less time to get to the result. However, in comparison to Reliability and energy, which are shown in table2.b and table2.c, the output obtained by the greedy method is dominant by the proposed method. The best unreliability and energy results from the greedy algorithm are 0.0758 and 0.1764, but the proposed method improves the results up to 0.0140 and 0.1563, respectively. This is the result of exploring the search space in a higher accuracy by the proposed method.

Table 1.a. Comparison of execution time of proposed method with greedy method

	Multi Objective	Greed
Execution Time(s)	0.0545	0.054
Energy consumption (J)	0.8087	0.816
Unreliability	0.6581	0.759

Table 1.b. Comparison of unreliability of proposed method with greedy method

	Multi Objective	Gree
Execution Time(s)	0.1954	0.27
Energy consumption (J)	0.3909	0.21
Unreliability	0.0140	0.07

Table 1.c. Comparison of energy of proposed method with greedy method

	Multi Objective	Gree
Execution Time(s)	0.3909	0.272
Energy consumption (J)	0.1563	0.176
Unreliability	0.4304	0.795

The most critical factor in measuring system performance is calculating the resultant graph length from the output of the algorithm. The Evaluation Metrics are based on two standards, SLR and Speedup.

- *Scheduling length Ratio(SLR):*

The length of the scheduled graph from the output graph is an essential factor in comparing different scheduling algorithms. We need to synchronize these graphs to get the shortest possible length in each graph. To obtain a suitable rate for comparing them. This quantity, according to the algorithm presented in [11], represented in the following equation:

$$SLR = \frac{makespan}{\sum_{T_i \in CP_{min}} min\ p_k \in p(W(T_i, P_k))} \quad (6)$$

Where makespan or completion time is the total time taken to process a set of tasks for its complete execution, the critical path is determined by CP_{min} and the execution time T_i task works on the P_k processor represents by$W$ ¿).

- *Speedup parameter:*

Speedup metric is resulted from dividing the speed of task continues execution on the fastest processor by total completion time of tasks, which is computed as:

$$Speedup = \frac{min\ P_k \in p(\sum_{T_i \in T} W(T_i, P_k))}{makesspan} \quad (7)$$

In order to show the efficacy of the proposed algorithm, we compare the proposed method with well-known single- and multi-objective heuristic scheduling algorithms (e.g. EGA-TS [4], HEFT [11] and, BGA [12]) in terms of performance standards (SLR and Speedup). To this end, 100 random graphs with size 20 are created using the generating graph processing software. The number of nodes in each level is uniformly distributed, and the mean value α × √n is generated randomly. In a random graph, the cost of communication for all the nodes is considered to be the same. The Table.1.a and Table.1.b show the results of comparisons; as the tables show, the proposed method outperforms EGA-TS [4], HEFT [11] and, BGA [12] in SLR and speedup by an average of 29.5% and 21%, respectively.

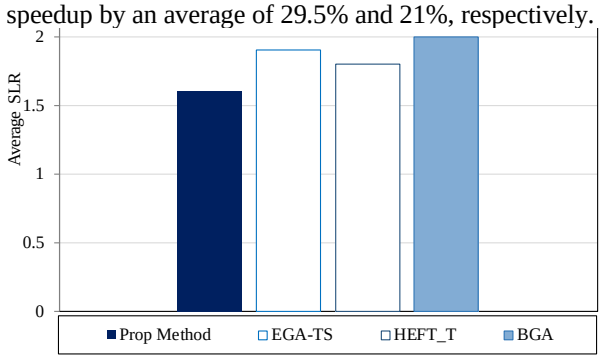

Figure 8.a. Indicates SLR in the proposed method compared to other methods

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on September 30,2024 at 17:23:47 UTC from IEEE Xplore. Restrictions apply.

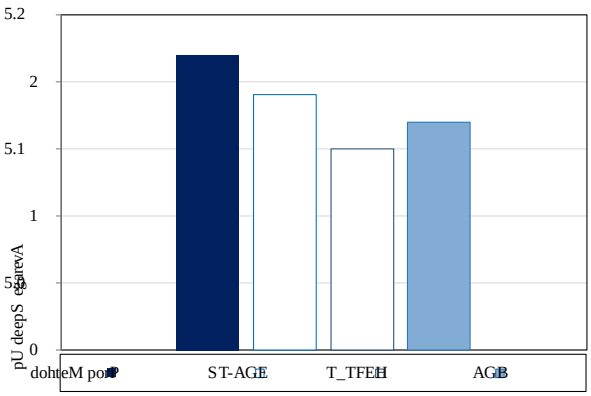

Figure 8.b. Indicates the acceleration rate of the proposed method compared to other methods

IV. CONCLUSION

In this paper, a genetic algorithm-based multi-objective task scheduling method is proposed for designing embedded systems. In this method, tasks which are modelled as a task graph are mapped and scheduled on a given hardware architecture. The proposed method for multi-objective optimization addresses the reliability of the scheduled tasks in addition to their execution time and energy consumption. The experimental results show the capability of the proposed algorithm in multi-objective task scheduling by comparing it to the greedy method. Furthermore, the superiority of the proposed method is shown as it achieves 29.5% better SLR and 21% better speedup in comparison to the well-known task scheduling algorithms. The use of a multi-objective optimization strategy allows the system designer to balance the various parameters of the embedded system (hardware/software) during the task scheduling design step.

REFERENCES

[1] R. Cheung, "A User-Oriented Software Reliability Model", IEEE Transactions on Software Engineering, vol. -6, no. 2, pp. 118-125, 1980.

[2] [3]H. Nikolov, T. Stefanov and E. Deprettere, "Systematic and Automated Multiprocessor System Design, Programming, and Implementation", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 27, no. 3, pp. 542-555, 2008.

[3] K. Sigdel, C. Galuzzi, K. Bertels, M. Thompson and A. Pimentel, "Evaluation of Runtime Task Mapping Using the rSesame Framework", International Journal of Reconfigurable Computing, vol. 2012, pp. 1-17, 2012.

[4] M. Akbari, H. Rashidi and S. Alizadeh, "An enhanced genetic algorithm with new operators for task scheduling in heterogeneous computing systems", Engineering Applications of Artificial Intelligence, vol. 61, pp. 35-46, 2017.

[5] W. Guan, M. Moghaddam and C. Ababei, “Uncertainty aware mapping of embedded systems for reliability, performance, and energy”, 19th International Symposium on Quality Electronic Design (ISQED), 2018.

[6] A. Pimentel, C. Erbas and S. Polstra, "A systematic approach to exploring embedded system architectures at multiple abstraction levels", IEEE Transactions on Computers, vol. 55, no. 2, pp. 99-112, 2006.

[7] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, “A fast and elitist multiojective genetic algorithm: NSGA-II,” *IEEE Trans. On Evolutionary Computation, 2002.*

[8] Y. Xu, K. Li, J. Hu and K. Li, "A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues", Information Sciences, vol. 270, pp. 255-287, 2014.

[9] I. Meedenyia, “Architecture Optimization of Embedded Systems under Uncertainty in Probabilistic Reliability Evaluation Model Parameters,” Ph.D. Thesis, Faculty of Science, Aamsterdam University, 2006.

[10] I. Meedeniya, A. Aleti, and L. Grunske, “Architecture-driven reliability optimization with uncertain model parameters”, *J.of Systems and Software, 2012*

[11] H. Topcuoglu, S. Hariri and Min-You Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing", IEEE Transactions on Parallel and Distributed Systems, vol. 13, no. 3, pp. 260-274, 2002.

[12] S. Gupta, V. Kumar and G. Agarwal, “Task Scheduling in Multiprocessor System Using Genetic Algorithm”, 2010 Second International Conference on Machine Learning and Computing, 2010.

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on September 30,2024 at 17:23:47 UTC from IEEE Xplore. Restrictions apply.