An Energy Aware Scheduling for Reconfigurable Heterogeneous Systems

Ines Ghribi[1], Riadh Ben Abdallah[1] and Mohamed Khalgui[1,2]
[1]*LISI Laboratory, National Institute of Applied Sciences and Technology, University of Carthage, Tunis, Tunisia*
[2]*Faculty of Science of Tunis, University of Tunis el Manar, Tunis, Tunisia*

Keywords: Embedded Systems, Scheduling, Probability, DVFS.

Abstract: One of the major challenges of computer system design is the management and conservation of energy while satisfying QoS requirements. Recently, Dynamic Voltage and Frequency Scaling (DVFS) has been integrated to various embedded processors as a mean to increase the battery life without affecting the responsiveness of tasks. This paper proposes an enhancement for I-codesign methodology [1] optimizing the energy consumption of the designed system.We propose an energy aware real-time scheduling algorithm. This algorithm makes use of the defferable server for the scheduling of aperiodic tasks along with DVFS. Simulation results demonstrate a decrease in the resulting energy consumption compared to the previously published work.

1 INTRODUCTION

Energy consumption is one of the major limiting factors of battery powered real-time systems. In this context, optimizing energy consumption without affecting performance while satisfying real time constraints is of major interest. To meet the timing constraints of the system, a scheduler must coordinate a set of tasks at different states (idle, blocked, running) and asks the run-time system to allocate the required resources to their execution. Many objectives must be considered in the design of a scheduling algorithm: (i) Guarantee that tasks with hard timing constraints will always meet their deadlines, (ii) Attain a high degree of schedulable utilization for hard deadline tasks, (iii) Provide fast average response time for tasks with soft deadlines (aperiodic tasks).

To obtain an energy-efficient design, the Dynamic Voltage and Frequency Scaling (DVFS) feature is widely adopted in modern processors (Horowitz et al., 1994). The basic idea of the DVFS strategy is to reduce a processor's processing frequency, as long as task's timing constraints are not violated. Indeed, the power consumption of the processor is a polynomial of the processing frequency, generally with a degree no less than 2 (Li, 2012), while the overall execution time of a task is just inversely proportional to the processing frequency. DVFS provides the possibility of minimizing energy consumption given a certain performance/timing requirement.

In an early work, we proposed a methodology called I-codesign for reconfigurable co-design (Ghribi et al., 2016a). I-codesign presents an abstract model for hardware/software systems allowing early exploration of hardware/software trade-offs and evaluation of design alternatives. This model supports incremental refinement and evaluation at multiple abstraction levels. Its aim is to lead to an efficient implementation and improve overall system performance. The entry point for I-codesign is a hardware/software specification modeled by a DAG (Directed Acyclic Graph) where nodes are software functions. I-codesign maps this specification into a hardware architecture that is mainly an MPSoC. It also defines a new partitioning and mapping techniques for the proposed hardware/software model: a functional algorithm followed by a constructive algorithm and finally an iterative algorithm for optimization. Based on several design constraints such as inclusion/exclusion, communication costs, energy, memory, real-time feasibility and probabilistic estimations, I-codesign takes decisions of near-optimal placement of software functions into the target hardware units.

In this paper, we investigate the opportunity of reducing power consumption of the system to be designed according to I-codesign methodology and thus by introducing the DVFS feature in the proposed scheduling algorithm. We study the scheduling of a heteregenous task set modeled and partitioned according to the I-codesign methodology. An energy aware real-time scheduling algorithm for probabilistic heterogeneous task set is introduced in order to enhance the I-codesign power consumption and response time metrics. DVFS is applied on periodic

Ghribi, I., Abdallah, R. and Khalgui, M.
An Energy Aware Scheduling for Reconfigurable Heterogeneous Systems.
DOI: 10.5220/0006460701710177
In *Proceedings of the 12th International Conference on Software Technologies (ICSOFT 2017)*, pages 171-177
ISBN: 978-989-758-262-2

tasks in order to reduce energy consumption without compromising periodic function deadlines and aperiodic functions responsiveness. In order to apply the proposed scheduling algorithm periodic functions priorities have been redefined according to two constraints : (i) the edge probability connecting the function to their predecessors in the task DAG, (ii) Function hierarchy which refers to the level of the corresponding function on the task DAG representation. The originality of this work resides in including the probabilistic estimation of the task's execution not only in the mapping process but also in the scheduling algorithm. The consideration of the precedence constraint through the proposed DAG hierarchy rule does cooperate and improve the overall system scheduling performance.

The paper proceeds as follows. The next Section describes useful background. Section III presents the I-codesign methodology. In Section IV, the system formalization and the notations used in this paper are developed. Section V exposes the proposed algorithm. Section VI shows simulation results of scheduling of real-time tasks and finally we conclude this paper in Section IV.

2 RELATED WORK

This section reviews the main approaches for scheduling a mixture of aperiodic tasks and periodic hard real-time tasks. The easiest way to prevent aperiodic tasks from interfering with periodic hard real-time tasks is to schedule them as background tasks executing only at times when there is no periodic task ready for execution. Although this method guarantees the schedulability of a periodic task, the execution of aperiodic tasks may be delayed and their response times are prolonged unnecessarily. The polling server is a periodic task with a period Ts, a capacity Cs and the highest priority (Li-yong et al., 2010). Every server's activation, it checks if there are any pending aperiodic tasks, if there are, the server uses its capacity to service them until either the task is finished or the server's capacity is depleted. However, if there is no pending aperiodic task, the server remains idle until its next activation which means that even if an aperiodic request occurs in the middle of the server's servicing time, the request will not be treated until the next period as the server will already be inactive (Liyong et al., 2010). The Priority Exchange (PE) and Deferrable Server (DS) algorithms, introduced by Strosnider in (Strosnider et al., 1995), overcome the drawbacks associated with polling and background servicing of aperiodic requests. As with polling, the PE and DS algorithms create a periodic task (usually of a high priority) for servicing aperiodic requests. However, unlike polling, these algorithms will preserve the execution time allocated for aperiodic service if, upon the invocation of the server task, no aperiodic requests are pending. These algorithms can yield improved average response times for aperiodic requests because of their ability to provide immediate service for aperiodic tasks. The DS algorithm maintains its aperiodic execution time for the duration of the server's period. Thus, aperiodic requests can be serviced at the server's high priority at anytime as long as the server's execution time for the current period has not been exhausted. At the beginning of the DS's period, the server's high priority execution time is replenished to its full capacity. Unlike the DS algorithm, the PE algorithm preserves its high priority execution time by exchanging it for the execution time of a lower priority periodic task (Desokey et al., 2006). The DS algorithm can provide better aperiodic responsiveness than polling because it preserves its execution time until it is needed by an aperiodic task. The DS algorithm is a simple algorithm to implement than the PE algorithm, because the DS algorithm always maintains its high priority execution time at its original priority level and never exchanges its execution time with lower priority levels as does the PE algorithm. It also requires less memory space than the PE and much lower computational complexity.

During the past two decades, tremendous works have been done regarding energy-aware scheduling on DVFS-enabled platforms. The application of DVFS algorithm to periodic task set is a well known research area (Tchamgoue et al., 2012; Ansari et al., 2013). However, few works in literature focuses on DVFS applied to heterogeneous task set comprising of periodic and aperiodic tasks (Dongkun and Jihong, 2004; Shin and Kim, 2006). The DVFS algorithm focuses on the usage and distribution of available slack time. The total time required by a task to run completely i.e. the actual execution time (aet) is always less than its worst case execution time (wcet). The difference that exists is the slack and it in turn, is utilized for reducing the voltage and frequency dynamically.

This paper proposes a new scheduling algorithm with original evaluation metrics for priority calculation along with the defferable server for the scheduling algorithm. In this work, DVFS is incorporated into the scheduling process in order to dynamically redefine the scheduled element's periodicity and reduce the energy consumption.

3 I-CODESIGN METHODOLOGY

The goal of I-codesign is to achieve a concurrent hardware/software system design. It acts on a probabilistic task model to a hardware architecture in a manner that fulfills all the system requirements and respects the design constraints. I-codesign deals with a set of models and transformations. The main idea behind I-codesign is the use of the *probabilistic* task model in mapping which embeds useful data for the mapping and further optimization steps. Figure 1 presents the flow diagram of the I-codesign methodology.

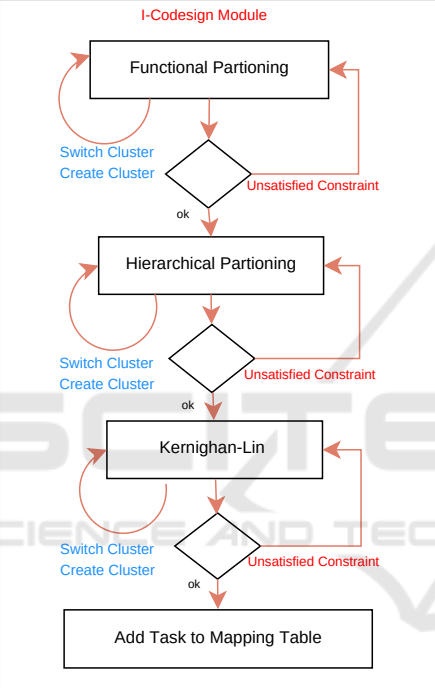

Figure 1: I-codesign methodology Flow Diagram.

The first step is the functional partitioning algorithm. It evaluates the inclusion/exclusion constraints between task functions and creates clusters depending on this constraint. Couples that are concerned with inclusion or exclusion constraints are placed in either the same or different clusters. Once all the inclusions and exclusions are evaluated, a feasibility analysis is performed. If all clustered functions sets on the created clusters are schedulable on one of the available processors then the schedulability test is validated. Otherwise, the functional partitioning is applied again to create new clusters with schedulable function sets. Since any inclusion/exclusion constraint is hard, the clustered tasks are locked and cannot be moved any more. The second phase is the hierarchical partitioning algorithm. It clusters the remaining functions that have no inclusion/exclusion constraints. The functions are evaluated by their connecting edge's probabilities and high probability values are treated first. The available memory space is evaluated at each iteration. Once all the remaining functions are placed into clusters a feasibility analysis is performed. If all the functions sets on the created clusters are schedulable on one of the available processors then the schedulability test is validated. Otherwise, the hierarchical clustering is applied again to generate clusters with schedulable function sets. The last phase is the kernighan-Lin optimization algorithm. This step evaluates both probability and communication cost on the edges connecting functions by gain calculation. If the gain is positive, then the function is moved to another cluster if its energy consumption on the other cluster is less or equal to its energy consumption on the original cluster. Otherwise it is left on the original cluster.

4 SYSTEM MODEL AND PROBLEM DEFINITION

4.1 Task Model

The software model comprises a set of tasks T_i , i $\in$ [1..N] represented by a directed acyclic graph T_i= (V_i, E_i), where (i) V_i is a set of nodes that correspond to functions, and (ii) E_i is a set of arcs which describe connection between functions. The edges are weighted with a couple $\prec Pr, Cc \succ$ where Pr is the probability of executing this edge and Cc is the communication cost of data transfer between the two nodes connected with the edge. A task T_i is a set of n periodic functions F= $\{F_1, F_2, .., F_n\}$. Each function F_i is described by quadruplet F_i= (R_i, C_i, P_i, D_i) where R_i is the phase, C_i is the *wcet*, P_i is the period and D_i is the deadline of the i^{th} periodic function. There are m aperiodic functions A= $\{A_1, A_2, .., A_m\}$. Each function A_i is described by doublet A_i= (Ar_i, C_i) where Ar_i is the arrival time and C_i is the execution time of the i^{th} aperiodic function A_i. Figure 2 presents an example of the task model composed of six periodic functions and two aperiodic functions. The following notations are used in the rest of the paper:

- EdgeProba(F_i) returns the highest probability of the edges connecting F_i to its predecessors; For example EdgeProba(F_5) returns 1,
- Level(F_i) returns the level of F_i on the DAG; For example level(F_3) returns 2.

We also defined inclusion/exclusion constraint. It is used to impose at a couple of functions and/or behaviors to be executed either on the same computing unit or on different ones. The exclusion constraint

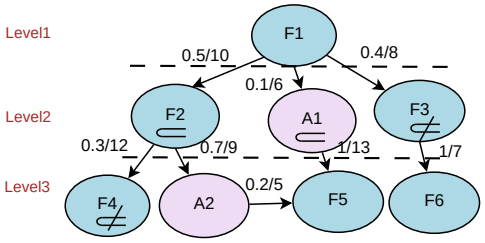

Figure 2: A task graph.

is modeled within the task representation by marking the symbol $\not\subset$ on the function F_i which means that F_i must not be executed with its predecessor on the same computing unit. The inclusion constraint is modeled by marking the symbol $\subset$ on F_i which means that F_i must be executed with its predecessor on the same computing unit.

4.2 Problem Definition

In a previous work (Ghribi et al., 2016a), I-codesign is described in detail. For a given task set, I-codesign is applied and as a result we get an optimized mapping of the system tasks into the hardware processing elements. The resulting mapping allows the execution of all possible reconfigurable scenarios of the designed system. It reduces the inter-PEs communications and guarantees the schedulability of tasks.
I-codesign has been developed with the assumption that all functions are periodic. In this work we address the scheduling of heterogeneous task set scheduling while reducing the overall energy consumption of the system.

5 PROPOSED ALGORITHM AND EXAMPLE

In this section we present a scheduling algorithm for heterogeneous task set comprising periodic and aperiodic tasks. This algorithm relies on a deffrebale server for aperiodic tasks. Aperiodic functions will be executed at the maximum frequency and priority in order to achieve lowest response time whereas utilization of periodic functions will be updated according to the DVFS algorithm. Thus, we defined a defferable server T_{DS} having a period denoted P_{DS} and a capacity denoted C_{DS}. For periodic functions, a priority definition is proposed as follows:

1. $\forall$ F_i, $F_j \in T_i$, if Level(F_i) $>$ Level(F_j), then priority(F_i) $>$ priority(F_j),
2. $\forall$ F_i, $F_j \in T_i$, if Level(F_i)=Level(F_j) and EdgeProba(F_i) $>$ EdgeProba(F_j), then priority(F_i) $>$ priority(F_j),

5.1 Scheduling Algorithm

The inputs of the proposed algorithm are the following : (i) H is the hyper-period of all the periodic tasks including T_{DS}, (ii) PQ is the periodic queue, (iii) AQ is the aperiodic queue. In our algorithm, we make use of the following methods: (i) *PQ.Sort()* is a function that sorts the periodic queue according the the DAG hierarchy and the Edge probability of the periodic functions, (ii) *AQ.Sort()* is a function that sorts the aperiodic queue according to the arrival time, (iii) *System.Run(F, T_{exe}, freq, Voltage)* is a function that executes a function F during a time T_{exe} and at the specified frequency *freq* and voltage *vol*, (iv) *System.Preempt()* is a function that preempts a function F and (v) *AQ.Aperiodic Request()* is a system call notifying the scheduler about the arrival of a new aperiodic function in AQ. The output of this algorithm is the next function to be executed with its estimated scaled frequency f_{scaled} and voltage V_{scaled}. An ERROR message is generated upon missing deadlines of any periodic task. Our scheduling algorithm is depicted in the following:

Algorithm 1.

```
1: procedure I-CODESIGN SCHEDULING(VAR PQ: TAB-PERIODIC, AQ: TAB-APERIODIC, TASKDAG: TREE, T: TIME
2:     for t = 0 to 2H do
3:         if (t == 0) or (t mod P_PES) == 0 then
4:             C_PES = MaxValue;
5:         end if
6:         if (AQ.Head != null) then
7:             System.Run(AQ.Head, C_PES, f_max, V_max)
8:         else FrequencyScaling(PQ, AQ, f_scaled)
9:             System.Run(PQ.Head, ET_PQ.Head, f_scaled, V_scaled)
10:        end if
11:        if AQ.AperiodicRequest() then
12:            System.Preempt()
13:            System.Run(AQ.Head, C_PES)
14:        end if
15:    end for
16: end procedure
```

Algorithm 2 implements the frequency scaling for periodic tasks execution as follows: (i) U_p is the utilization of all periodic functions, and (ii) U_{DS} is the utilization of aperiodic functions in the aperiodic queue.

Algorithm 2.

1: **procedure** FREQUENCY SCALING(PQ: TAB-PERIODIC, AQ: TAB-APERIODIC, VAR f_{scaled}: FREQUENCY)
2: $U_p = 0$
3: **for** $i = 0 \rightarrow length(PQ)$ **do**
4: $U_p = U_p + PQ[i].C_i/PQ[i].P_i$
5: **end for**
6: **if** $AQ! = null$ **then**
7: $U_{DS} = C_{DS}/P_{DS}$
8: **else** $U_{DS} = 0$
9: **end if**
10: $U_T = U_p + U_{DS}$
11: $f_{scaled} = min(f_1, f_2, .., f_{max})$
12: $U_T \leq f_{scaled}/f_{max}$
13: **end procedure**

The operating frequency selected is the lowest one for which the modified schedulability test succeeds. The voltage, of course, is changed to match the operating frequency. This algorithm is called by Algorithm 1 when a periodic function is specified for execution.

5.2 Example

We propose to apply the proposed algorithm on task T_1 presented in figure 3. The task is composed of a set of periodic function F= $\{F_1, F_2, .., F_7\}$ and aperiodic functions A= $\{A_1, A_2, A_3\}$. The output of I-codesign methodology applied to T_1 are two partitioning clusters. The operating frequencies are f= $\{0.25, 0.5, 0.75, 1\}$ Ghz.

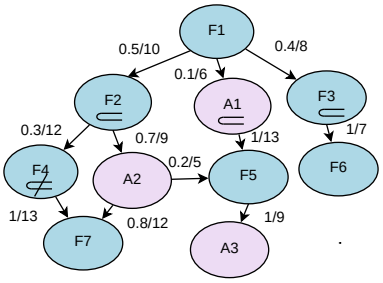

Figure 3: Task graph T_1.

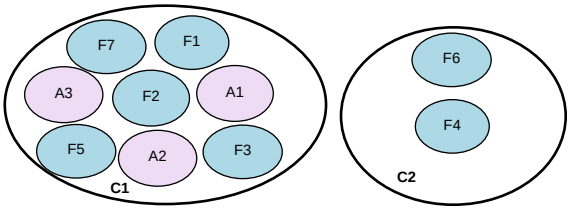

Figure 4: Resulted Clusters after applying I-codesign to T_1.

The final clusters are presented in figure 4. We propose to study the scheduling of cluster C1. The real-time parameters of the periodic and aperiodic functions are presented in Table-I and Table II.

Table-I describes the aperiodic task set with arrival time and execution time. Table II describes the periodic tasks proprieties including the defferable server.

Table 1: Aperiodic functions Set.

Function ID	Arrival Time Ar_i	Execution Time C_i
A_1	5	2
A_2	7	2
A_3	21	3

Table 2: Periodic functions Set.

Function ID	C_i	P_i	D_i	Priority
F_1	5	20	20	1
F_2	3	22	22	2
F_3	6	25	25	3
F_4	3	38	38	4
F_5	2	35	35	5
F_6	2	32	32	6
F_7	5	50	50	7
T_{PES}	2	5	5	1

In order to schedule the functions associated to the cluster C1, the PQ and AQ are populated according to the proposed priority definition and arrival time. For this example, a high priority server is created with an execution time of 2 time units and a period of 5 time units. At time= 0, the server's execution time is brought to its full capacity. This capacity is preserved until the first aperiodic request occurs at time = 5 since there is no pending aperiodic function. Hence, F_1 is executed. The frequency is scaled at the value f= 0.75 Ghz. At time= 5, the periodic request occurs to serve A_1 along with F_2 in the head of the periodic queue. Clearly, A_1 is serviced at the maximum frequency f= 1 Ghz until time= 7. F_2 and F_3 belong to the same DAG level, hence the probability on the edges connecting these functions with F_1 is assessed in order to determine the next function to be executed at time= 7. F_2 is serviced at a scaled frequency eaqual to 0.75 Ghz since EdgeProba (F_2) > EdgeProba (F_3). At time= 10, the server's execution time at priority 1 is brought to its full capacity and is used to provide immediate service for A_2 at the maximum frequency f= 1 Ghz. At time= 12, F_3 is serviced since there are no periodic function with higher edge probability at its DAG level at the frequency f= 0.75 Ghz followed by F_5 at time= 18 at a scaled frequency of 0.5 Ghz. At

time= 20, F_1 is serviced until time= 21 at a frequency f= 0.5 Ghz when it is preempted in order to serve A_3 immediately at the maximum frequency f= 1 Ghz. at time= 23, F_1 continues its execution at f= 0.75 Ghz. Figure 5 illustrates the time-line scheduling of the example as described above.

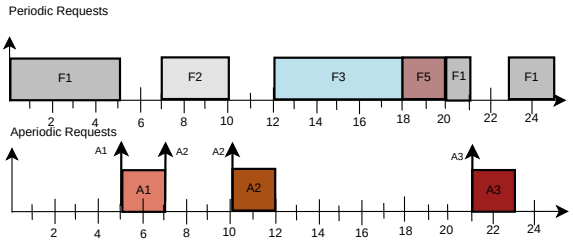

Figure 5: Scheduling time-line of the cluster C1.

Table III shows the resulting schedule of the given example task in table I and II.

Table 3: Defferable Server Schedule.

Time	Released	Pending	f_{scaled}	C_{PES}
0	F_1	–	0.75	2
5	A_1	–	1	0
7	F_2	A_2	0.75	0
10	A_2	–	1	0
12	F_3	–	0.75	0
18	F_5	–	0.5	2
20	F_1	–	0.5	2
21	A_3	F_1	1	0
21	F_1	–	0.75	0

6 SIMULATION RESULTS

In a previous work (Ghribi et al., 2016b), we developed a co-design execution environment called SPEX. It provides a toolbox that allows the creation of a hardware/software system description according to the proposed design models and that implements the I-codesign algorithms. It proposes a flexible task set generator for different scenarios and purposes. The tool places the software specification following several proposed design constraints as inclusion/exclusion parameters, probabilistic execution of the software tasks, available memory and energy on the hardware units and real-time parameters. To evaluate the new scheduling algorithm several task sets of different dimensions are generated. The generated tasks are passed through SPEX and we obtain mapping scheme of the task set. We developed a new simulation module that implements our scheduling algorithm based on defferable server along with DVFS technique. The simulator populates the periodic and aperidoic queue, runs the specification according to function characteristics and generates estimations of the total execution time and consumed energy.

In order to evaluate the proposed scheduling algorithms, various random task sets are generated according to the I-codesign modeling for probabilistic reconfigurable task sets. These tasks are decomposed into elementary functions and then characterized with the different co-design constraints (probability, communication costs, inclusion/exclusion). After applying the I-codesign algorithms, the resulting mapping is passed through the scheduling simulator. The scheduling results are compared to Earliest Deadline First (EDF) algorithm and the Rate Monotonic algorithm. Figures 6 and 7 present the performance results of our scheduling algorithm applied to different task sets along with those of EDF and RM. The comparison between the evaluated approaches has demonstrated that the new I-codesign scheduling algorithm offers better performance results particularly with large utilization factors and high number of nodes on the specification DAGs. These enhancements are due to probabilistic estimation of the communicated functions/behaviors that store dependent tasks with high chances to be executed successively on same PEs. Simulation results show that this contribution has many benefits: (i) the energy consumed during the system execution has been noticeably reduced and (ii) the global execution time has been minimized. Another advantage of I-codesign is its validation tests that includes real-time feasibility which result in avoiding any system fail due to a lack of resources.

7 CONCLUSIONS

In this work, an energy aware real-time scheduling algorithm with Dynamic Voltage and Frequency Scaling based on the Defferable Server has been proposed and implemented for mixed task set. This new scheduling algorithm is developed in order to enhance the I-codesign methodology. It considers the trade-offs between the energy consumption and the response time. It relies on simple constraints: the DAG hierarchy and the probability of execution for periodic functions. It makes use of DVFS technique in order to reduce the energy consumption of the system. Extensive simulation is carried out on our tasks sets. The results showed that our proposed energy efficient algorithm succeeds in reducing noticeably the energy consumption with no degradation in responsiveness of aperiodic tasks.

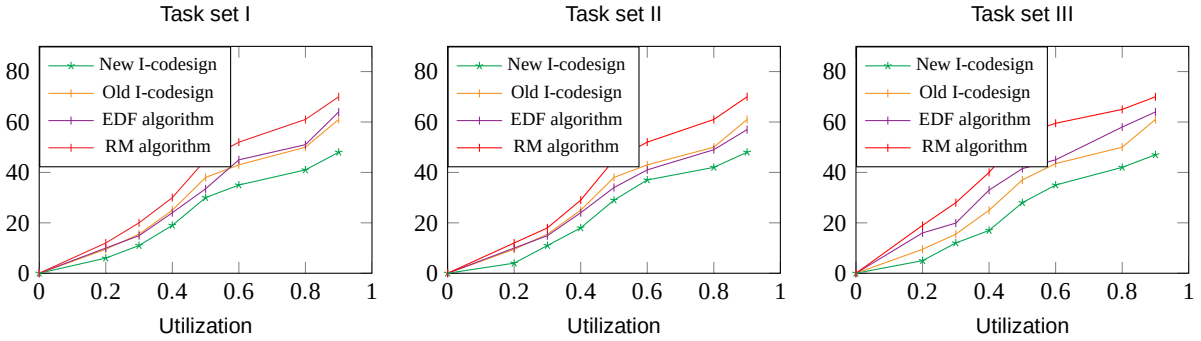

Figure 6: Simulation Results for Energy Consumption.

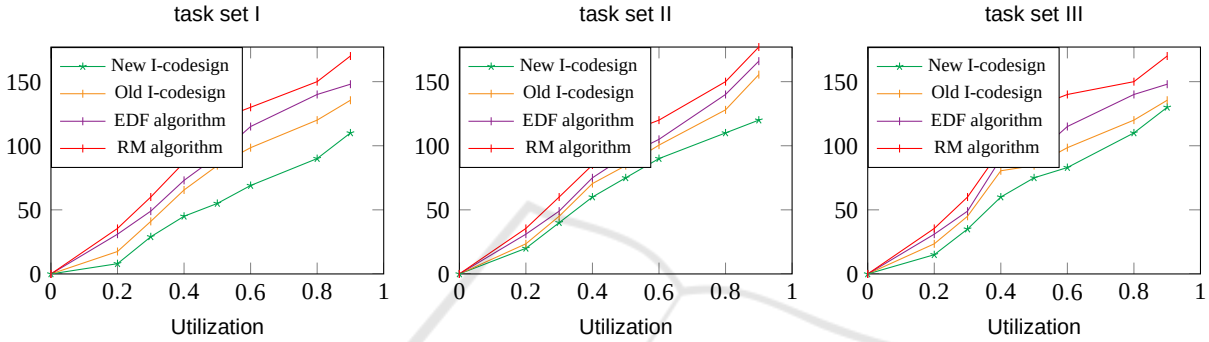

Figure 7: Simulation Results for Execution Time.

REFERENCES

Ansari, K. H., Chitra, P., and Sonaiyakarthick, P. (2013). Power-aware scheduling of fixed priority tasks in soft real-time multicore systems. In *2013 IEEE International Conference ON Emerging Trends in Computing, Communication and Nanotechnology (ICECCN)*.

Desokey, A. E. E., Gawad, A. A. E., Sarhan, A., and Moawed, S. (2006). Improving the performance of the deferrable server based garbage collection scheduling strategy. In *2006 ITI 4th International Conference on Information Communications Technology*, pages 1–2.

Dongkun, S. and Jihong, K. (2004). Dynamic voltage scaling of periodic and aperiodic tasks in priority-driven systems. In *Proceedings of the 2004 Asia and South Pacific Design Automation Conference*, pages 653–658, Yokohama, Japan. IEEE Press.

Ghribi, I., Abdallah, R. B., Khalgui, M., and Platzner, M. (2016a). New co-design methodology for real-time embedded systems. In *Proceedings of the 11th International Joint Conference on Software Technologies (ICSOFT 2016)*, pages 353–364, Lisbon, Portugal.

Ghribi, I., Abdallah, R. B., Khalgui, M., and Platzner, M. (2016b). Rco-design: New visual environment for reconfigurable embedded systems. In *Proceedings of 30th European Simulation and Modelling Conference - ESM'2016*, pages 217–224.

Horowitz, M., Indermaur, T., and Gonzalez, R. (1994). Low-power digital design. In *Proceedings of 1994 IEEE Symposium on Low Power Electronics*, pages 8–11.

Li, K. (2012). Scheduling precedence constrained tasks with reduced processor energy on multiprocessor computers. *IEEE Transactions on Computers*, 61(12):1668–1681.

Li-yong, B., Dong-feng, Z., and Yi-fan, Z. (2010). A priority-based polling scheduling algorithm in web cluster servers. In *Proceedings of 3rd International Conference on Computer Science and Information Technology*, volume 8, pages 501–505.

Liyong, B., Dongfeng, Z., and Yifan, Z. (2010). A priority-based polling scheduling algorithm in web cluster servers. In *2010 3rd International Conference on Computer Science and Information Technology*, volume 8, pages 501–505.

Shin, D. and Kim, J. (2006). Dynamic voltage scaling of mixed task sets in priority-driven systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(3):438–453.

Strosnider, J. K., Lehoczky, J. P., and Sha, L. (1995). The deferrable server algorithm for enhanced aperiodic responsiveness in hard real-time environments. *IEEE Transactions on Computers*, 44(1):73–91.

Tchamgoue, G. M., Kim, K. H., and Jun, Y. K. (2012). Dynamic voltage scaling for power-aware hierarchical real-time scheduling framework. In *2012 IEEE 15th International Conference on Computational Science and Engineering*, pages 540–547.