DYNAMIC LOW POWER RECONFIGURATIONS OF REAL-TIME EMBEDDED SYSTEMS*

Xi Wang[1], Mohamed Khalgui[1,2]
[1]*School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China*

Zhiwu Li[1,2]
[2]*Institute of Computer Science, Martin-Luther Universität, 06120 Halle (Saale), Germany*

Keywords: Embedded system, Reconfiguration, Low power, Real-time scheduling, Agent-based architecture.

Abstract: This paper deals with low power dynamic reconfigurations of real-time embedded systems. A reconfiguration scenario means the addition, removal or update of tasks in order to save the whole system at the occurrence of hardware/software faults, or also to improve its performance at run-time. When such a scenario is applied, the energy consumption can be increased or some real-time constraints can be violated. An agent-based architecture is defined where an intelligent software agent is proposed to check each dynamic reconfiguration scenario and to suggest for users useful technical solutions that minimize the energy consumption. It proposes first of all to modify periods, or to reduce execution times of tasks or finally to remove some of them. Users should choose one of these solutions in order to guarantee a low power consumption satisfying limitations in capacities of used batteries. We developed and tested a tool supporting all these services to be evaluated in the research work.

1 INTRODUCTION

Nowadays, the minimization of the energy consumption is an important criterion for the development of real-time embedded systems due to limitations in the capacity of their batteries, in addition to their tasks which become more and more complex than ever. Several interesting studies have been proposed in recent years for their real-time and low power scheduling (Shin and Choi, 1999; Quan and Hu, 2002; Yun and Kim, 2003; Yao *et al.*, 1995; Gaujal and Navet, 2007). We classically distinguish two reconfiguration policies: static and dynamic reconfigurations. Static reconfigurations are applied off-line to apply changes before the system's cold start (Angelov *et al.*, 2005), whereas dynamic reconfigurations are dynamically applied at run-time. Two cases exist in the latter: manual reconfigurations applied by users (Rooker *et al.*, 2007) and automatic reconfigurations applied by Intelligent Agents (Khalgui *et al.*, 2010; Al-Safi and Vyatkin, 2007). In this paper[2], we assume embedded real-time systems based on CMOS processors (Sarvar, 1997), such that the power consumption (P) of a processor is assumed to be quadratically dependent on supply voltage V_{CC}. Similar to (Shin and Choi, 1999) and (Yao *et al.*, 1995), we assume that the energy consumption is quadratically dependent on the supply voltage V_{CC}, and the processor speed can be adjusted according to the processor utilization U. This idea is interesting and will be used to allow low power reconfigurations of embedded real-time systems to be implemented by sets of independent, periodic and synchronous tasks. To reach this goal, we define an agent-based architecture in which an intelligent software agent is proposed to check each dynamic (manual or automatic) reconfiguration scenario at run-time, and to allow feasible and low power adaptive systems. The agent proposes first of all to modify periods and deadlines of tasks in order to decrease the processor speed. It suggests as a second solution to modify execution times of tasks in order to decrease the processor utilization. Finally it proposes for users to remove some of them according to their static priorities or also according to their processor utilizations.

The next section analyzes previous work on low

*Research Laboratory: http://sca.xidian.edu.cn

[2]Detailed descriptions are available in our website: http://sca.xidian.edu.cn/~wangx/SCATR201012004.pdf.

Wang X., Khalgui M. and Li Z. (2011).
DYNAMIC LOW POWER RECONFIGURATIONS OF REAL-TIME EMBEDDED SYSTEMS.
In *Proceedings of the 1st International Conference on Pervasive and Embedded Computing and Communication Systems*, pages 415-420
DOI: 10.5220/0003308304150420
Copyright © SciTePress

power and real-time scheduling as well as reconfigurations of embedded architectures. Section 3 formalizes reconfigurable real-time systems and their power consumptions. In Section 4, we define an agent-based architecture for the minimization of the energy consumption. This architecture is implemented, simulated and analyzed in Section 5. Section 6 concludes this work.

2 STATE OF THE ART

Although these rich and useful contributions of (Shin and Choi, 1999; Quan and Hu, 2002; Yun and Kim, 2003; Yao *et al.*, 1995; Gaujal and Navet, 2007) provide interesting results, they do not address dynamic reconfigurations with low power consumptions of embedded systems such that real-time solutions are automatically computed for users when the system's behavior is dynamically reconfigured. Real-time scheduling has been extensively studied in the last three decades (Baruah and Goossens, 2004) where several Feasibility Conditions (FC) for the dimensioning of a real-time system are defined to enable a designer to grant that timeliness constraints associated with an application are always met for all possible configurations. Among all scheduling policies, Earliest Deadline First (EDF) is an optimal uniprocessor scheduling algorithm in the following sense: if a collection of independent periodic jobs characterized by arrival times equal to zero and by deadlines equal to corresponding periods are feasible with another scheduling policy, then it is feasible with EDF. Nowadays, several research works have been proposed to develop reconfigurable embedded systems. The work in (Angelov *et al.*, 2005) proposes reusable tasks to implement a broad range of systems where each task is statically reconfigured without any re-programming. Rooker *et al.* propose in (Rooker *et al.*, 2007) a complete methodology based on the human intervention to dynamically reconfigure tasks. The research in (Brennan *et al.*, 2001) proposes an agent-based reconfiguration approach to save the whole system when faults occur at run-time. In (Al-Safi and Vyatkin, 2007), an ontology-based agent is proposed to perform system's reconfigurations that adapt changes in requirements and also in environment. As far as the authors know, no work is reported to address the problem of dynamic reconfigurations of embedded systems under real-time and low-power constraints. Our current research addresses low-power reconfigurations of real-time embedded systems when additions-removals-updates of tasks are dynamically applied at run-time to save the system or improve its performance.

3 FORMALIZATION OF RECONFIGURABLE REAL-TIME SYSTEMS

A reconfiguration scenario is assumed in this paper to be an operation allowing the addition-removal-update of tasks from/into the system. Each scenario should be applied while reducing the energy consumption which is a very important criterion. We define a real-time embedded system *Sys* as a set of tasks that should meet real-time constraints defined in user requirements: $Sys = \{\tau_1, \tau_2, \ldots, \tau_n\}$. When a reconfiguration scenario is applied, a subset of tasks can be added/removed into/from the system. Each task τ_i of *Sys* describes: i) a function F_i defining its function, ii) the static priority S_i among all the system's (new and old) tasks, iii) the release time R_i defining the execution start time of the task, iv) the WCET C_i, v) the period T_i, and vi) the deadline D_i. It is assumed in addition that a) all the system's tasks are periodic and synchronous, and b) the period of each task is equal to the corresponding deadline. The system *Sys* is dynamically reconfigured at run-time such that its new implementation is $Sys = \{\tau_1, \tau_2, \ldots, \tau_n, \tau_{n+1}, \ldots, \tau_m\}$. The subset $\{\tau_{n+1}, \ldots, \tau_m\}$ is added to the initial implementation $\{\tau_1, \tau_2, \ldots, \tau_n\}$. The processor utilization before and after the reconfiguration scenario is as follows:

$$U_{bef} = \sum_{i=1}^{n} \frac{C_i}{T_i} \tag{1}$$

$$U_{aft} = \sum_{i=1}^{m} \frac{C_i}{T_i} \tag{2}$$

The energy consumption is assumed quadratically dependent on processor utilization U (Wang *et al.*, 2004), the equation can be described as:

$$P = kU^2 \tag{3}$$

Proposition 1. *Let U_{aft} and U_{bef} denote the processor utilization after and before a reconfiguration scenario is applied, respectively. If $U_{aft} \le U_{bef}$, then the processor speed after the reconfiguration scenario is not greater than that before. In this case the power consumption is stable or minimized.*

Proof: Due to Eq. (3), the energy consumption is quadratically dependent on U. If $U_{aft} \le U_{bef}$, substituting it into Eq. (3) leads to $P_{aft} = kU_{aft}^2 \le kU_{bef}^2 = P_{bef}$. Then the power will be stable or minimized. $\diamond$

4 AGENT-BASED ARCHITECTURE FOR LOW POWER RECONFIGURATIONS OF EMBEDDED SYSTEMS

We define an agent-based architecture for dynamic low-power reconfigurations of an embedded real-time system. When automatic or manual reconfigurations are applied at run-time to add, remove or update tasks, the agent should check if the power consumption is increased. If a reconfiguration scenario is dynamically applied at run-time to add new tasks, the processor utilization U_{rec} of the system will be certainly increased as follow:

$$U_{rec} = \sum_{i=1}^{m} \frac{C_i}{T_i} \tag{4}$$

4.1 Modification of Periods and Deadlines

The agent proposes as a first technical solution to modify the periods and deadlines of tasks in order to decrease the processor utilization U_{aft} to be lower than U_{bef}. For the reconfigured system, it is assumed that T_1', T_2', …, and T_m' are the modified periods of m tasks, and $T_1' = T_2' = \ldots = T_m' = T'$.

In order to minimize the energy consumption, based on Eq. (3), the processor utilization should be reduced. Proposition 1 leads to:

$$U_{aft} = \sum_{i=1}^{m} \frac{C_i}{T'} \leq U_{bef} \tag{5}$$

where

$$T' = \lceil \frac{\sum_{i=1}^{m} C_i}{U_{bef}} \rceil \tag{6}$$

In Eq. (6), T' is the modified period of each task. The processor utilization after reconfiguration of the processor is as follows:

$$U_{aft} = \sum_{i=1}^{m} \frac{C_i}{T'} \tag{7}$$

4.2 Modification of WCETs

For the low-power reconfiguration of embedded systems, the agent proposes as a second technical solution to reduce the WCETs of tasks in order to decrease the processor utilization U_{aft}.

$$U_{aft} = \sum_{i=1}^{m} \frac{C_i'}{T_i} \tag{8}$$

where C_1', C_2', …, and C_m' are the new WCETs of tasks. Suppose that $C_1' = C_2' = \cdots = C_m' = C'$, Similar to Eq.(5), we can get:

$$U_{aft} = \sum_{i=1}^{m} \frac{C'}{T_i} \leq U_{bef} \tag{9}$$

C' can be solved as:

$$C' = \lfloor \frac{U_{bef}}{\sum_{i=1}^{m} \frac{1}{T_i}} \rfloor \tag{10}$$

The actual utilization of the processor is as follows:

$$U_{aft} = C' \times \sum_{i=1}^{m} \frac{1}{T_i} \tag{11}$$

4.3 Removal of Tasks

The third solution proposed by the agent allows the removal of tasks in order to minimize the energy consumption after any reconfiguration scenario of an embedded system. Two policies are proposed: the agent suggests to remove tasks according to their functional priorities or to their processor utilization.

4.3.1 First Policy: Static Priority Criterion

By defining for each task τ_i a static priority S_i, the agent suggests to remove tasks with lower static priorities because their removal can be useful for a low power reconfiguration of the system. Let *List* be the list of tasks of *Sys* in descending order of static priorities. The most unimportant tasks should be removed to keep the new utilization of the system U_{aft} lower than U_{bef}. Specifically, we have

$$U_{aft} = \sum_{F_i=1}^{k} \frac{C_i}{T_i} \leq U_{bef} \tag{12}$$

where $m-k$ is the number of removed tasks with $k \leq m$. The agent should look for the highest value of U_{aft} such that $U_{aft} \leq U_{bef}$. The total energy consumption can be calculated by Eq. (3).

4.3.2 Second Policy: Processor Utilization Criterion

It is similar to the first policy. Their difference is that the second policy does not to arrange tasks due to their static priorities, but due to their processor utilization. In this case, the *List* is the list of tasks of *Sys* in ascending order of processor utilization of each task τ_i. If the system is not feasible or consumes more energy, the tasks with highest processor utilizations should be removed to keep as many tasks as possible remaining in the system.

5 EXPERIMENTAL STUDIES

This section presents an experimental study applying low power reconfigurations of embedded real-time systems. We present first of all the implementation of the agent-based architecture, before showing the simulations and analysis that are made to evaluate the benefits of our contributions.

5.1 Implementation of the Reconfiguration Agent

In this paper, the agent's goal is to check dynamic (automatic or manual) reconfiguration scenarios, and suggest useful solutions for the minimization of the energy consumption. Each solution is generated as an input file from the agent to the well-known simulator Cheddar (Singhoff *et al.*, 2004) to check its feasibility. This implementation is tested in our research laboratory at Xidian University by assuming several cases of systems. Note that Eqs. (1), (3), (6), and (10) can be used to calculate U_{bef}, P_{bef}, T', and C', respectively. According to Eqs. (4) and (11), the utilization U^1_{aft} is computed after the modification of periods (and deadlines), and the utilization U^2_{aft} after the modification of WCETs, respectively. Note in addition that U^3_{aft} and U^4_{aft} correspond to the utilization after tasks are removed according the two fixed policies, as shown in Eq. (12). We use P^1_{aft}, P^2_{aft}, P^3_{aft}, and P^4_{aft} to indicate the power consumption after the modification of periods (and deadlines), after the modification of WCET, after the removal of tasks by considering their static priorities, and after the removal of tasks by considering their utilizations, respectively. All of them can be calculated by Eq. (3).

In the first two technical solutions, the algorithm's complexity is $O(n)$, and in the third one, it costs $O(n^2)$.

5.2 Simulations

This section presents simulation results by applying low-power reconfigurations of an embedded real-time system that is initially composed of 50 tasks and dynamically reconfigured at run-time to add 30 new ones. We assume the following temporal characteristics of the system under consideration:

- **Initial System's Tasks & Addition Tasks.** All the initial tasks of the system and the addition tasks are in the file "*system.txt*" and "*add.txt*", respectively. (F_i defines the function. R_i, C_i, T_i, and D_i define the temporal parameters of each task.
- **Static Priorities.** All the functional priorities of the system's tasks are defined in the file priority.txt (F_i and S_i define functional and static priorities of each task, respectively.

All the related files and experimental data are shown in the technical report of our laboratory(Wang *et al.*, 2004).

Algorithm 1: Low-Power Reconfigurations.

```
input "system.txt" file;
input "add.txt" file;
input "priority.txt" file;
compute ($U_{bef}$);
$calculate\_period\ T'$;
for ($i = 1, i = size(sys) + size(add), i++$)
    $U_i = C_i/T'$;
    $\sum U^1_{aft} += U_i$;
endfor;
Evaluate_energy($U_{bef}$, $U^1_{aft}$);
$calculate\_execution\_time\ C'$;
for ($i = 1, i = size(sys) + size(add), i++$)
    $U_i = C'/T_i$;
    $\sum U^2_{aft} += U_i$;
endfor;
Evaluate_energy($U_{bef}$, $U^2_{aft}$);
Sort all the tasks by a descending order based on their static priorities;
Loop1 $remove\_tasks\_priority$ ($Sys_{new1}$);
for ($i = 1, i = size(Sys_{new1}), i++$)
    $\sum U^3_{aft} += U_i$;
endfor;
keep_minimal($U^3_{min\_aft}$);
EndLoop1
Evaluate_energy($U_{bef}$, $U^3_{min\_aft}$);
sort all the tasks by a ascending order based on their utilization;
Loop2 $remove\_tasks\_utilization(Sys_{new2})$;//solution 3 to remove possible tasks (second criterion)
for ($i = 1, i = size(Sys_{new2}), i++$)//to compute the new utilization when tasks are removed (second criterion)
    $\sum U^4_{aft} += U_i$;
endfor;
keep_minimal($U^4_{min\_aft}$);
Evaluate_energy($U_{bef}$, $U^4_{min\_aft}$);
EndLoop2
end;
```

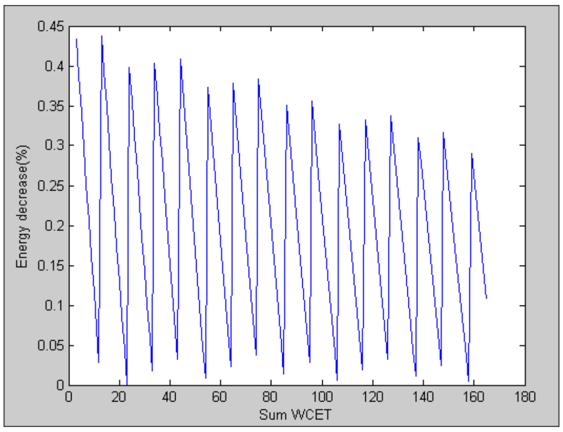

Figure 1: Energy decrease by modify periods.

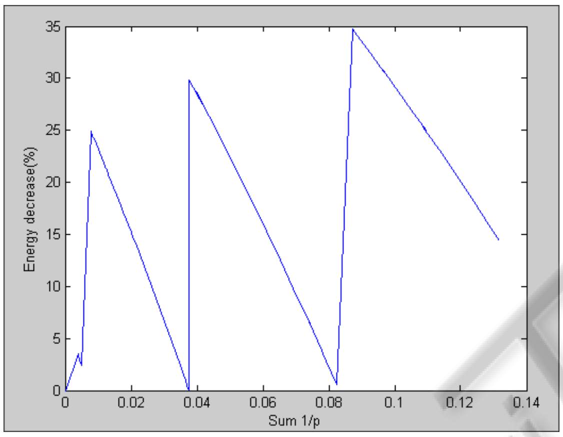

Figure 2: Energy decrease by modify WCETs.

5.3 Analysis

We present some analysis that proves the advantages of the different proposed solutions.

5.3.1 First Solution

Due to Eqs. (6) and (3), the power minimization is proven to be dependent on the WCETs. If all the 30 tasks are added, their WCETs are equal to 165. We made several simulations for $\sum C_i = 3,4,5,\ldots,165$. The result is shown in Fig. 1 in which we find a decrease of consumption between 0 and 0.45%. The energy reduction is piecewise approximately linear depending on the $\sum C_i = 3,4,5,\ldots,165$.

5.3.2 Second Solution

From Eqs. (10) and (3), the power minimization is proven to be dependent on periods. If all the 30 tasks are added, the value of $\sum_{i=1}^{30} 1/T_i$ is equal to 0.131684.

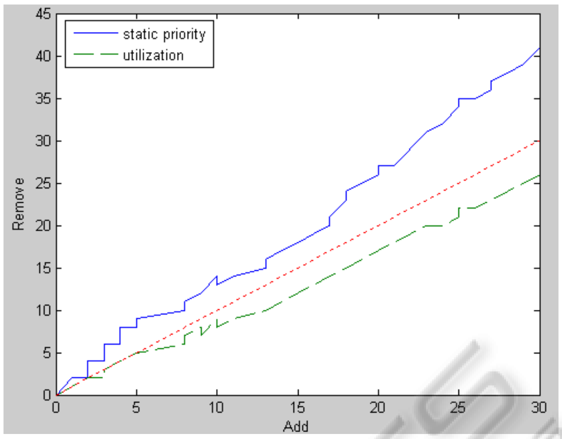

Figure 3: Comparing the number of removed task between two remove strategies.

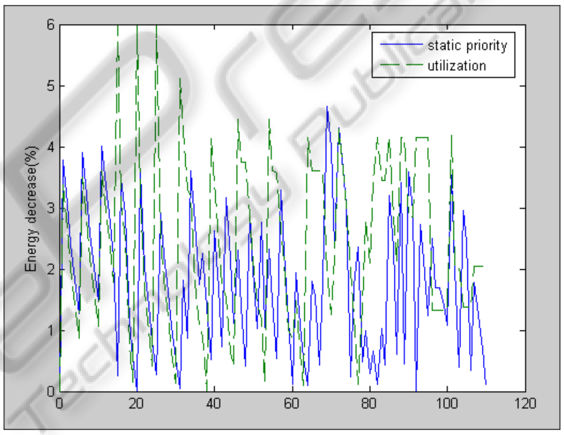

Figure 4: Comparing energy decrease between two remove strategies.

Simulations are made for subsets of tasks with the range of $\sum 1/T_i$ between 0 and 0.131684. The result is shown in Fig. 2 where the minimization of the energy consumption is between 0 and 35%. This simulation result proves the benefits of the second solution than the first. Nevertheless, sometimes, the modification of periods is more simpler that minimization of WCET. It is unclear whether the second solution is definitely better than the first.

5.3.3 Third Solution

In the third solution where tasks are removed at run-time to minimize the energy consumption after reconfiguration scenarios, several simulations are applied to evaluate the benefits. In Fig. 3, the continuous line presents the first policy where the priority function is used as a criterion to remove tasks. In this case, the number of removed tasks is equal to or greater than the added one: the tasks remaining in the system are less than 50. The dotted line corresponds to the removal of tasks due to their processor utilizations. The number of removed tasks is smaller than the addition number. This second policy is more useful than the first. Fig. 4 shows the minimization of the energy consumption when the first (continuous line) and the second (dotted line) policies are applied. It indicates that the energy consumption stills minimal when we apply the second solution where WCETs of tasks are modified.

6 CONCLUSIONS

This paper deals with low power and real-time dynamic reconfigurations of embedded systems to be implemented by sets of tasks that should meet real-time constraints while satisfying limitations in the capacity of batteries. A reconfiguration scenario means the addition, removal or update of tasks in order to save the system when faults occur or to improve its performance. The energy consumption can often be increased or real-time constraints can often be violated when tasks are added. To allow a stable energy consumption before and after the application of each reconfiguration scenario, an agent-based architecture is defined where an intelligent software agent is proposed to check each dynamic reconfiguration scenario and to suggest for users effective solutions in order to minimize the energy consumption. It proposes to modify periods, reduce execution times of tasks or remove some of them. A tool is developed and tested to support all these services. In our future work, we plan to study low power and real-time reconfigurations of asynchronous tasks that can be loaded in a uniprocessor or can be distributed on different calculators.

ACKNOWLEDGEMENTS

This work was supported in part by the Natural Science Foundation of China under Grant No. 60773001 and 61074305, the Fundamental Research Funds for the Central Universities under Grant No. 72103326, the National Research Foundation for the Doctoral Program of Higher Education, the Ministry of Education, P. R. China, under Grant No. 20090203110009, "863" High-tech Research and Development Program of China under Grant No. 2008AA04Z109, the Research Fellowship for International Young Scientists, National Natural Science Foundation of China, and Alexander von Humboldt Foundation.

REFERENCES

Al-Safi, Y. and Vyatkin, V. (2007). An ontology-based reconfiguration agent for intelligent mechatronic systems. In *Third International Conference on Industrial Applications of Holonic and Multi-Agent Systems*. Springer-Verlag.

Angelov, C. Sierszecki, K. and Marian, N. (2005). Design models for reusable and reconfigurable state machines. In *L.T. Yang, et al. (Eds.), EUC, LNCS, 3824*. International Federation for Information Processing.

Baruah, S. and Goossens, J. (2004). Scheduling real-time tasks: Algorithms and complexity. In *In Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. Chapman Hall/ CRC Press.

Brennan, R. W. Fletcher, M. and D. H. Norrie, (2001). A holonic approach to reconfiguring realtime distributed control systems. In *Multi-Agent Systems and Applications*. Springer-Verlag.

Gaujal, B. and Navet, N. (2007). Dynamic voltage scaling under edf revisited, real-time systems. In *Some results are available as research report INRIA RR-5125*. Springer Verlag.

Khalgui, M. Mosbahi, O. Li, Z. W. and Hanisch, H.-M. (2010). Reconfigurable multi-agent embedded control systems: From modelling to implementation. In *IEEE Transactions on Computers*.

Quan, G. and Hu, X. (2002). Minimum energy fixed-priority scheduling for variable voltage processors, design, automation and test. In *Europe Conference and Exhibition*.

Rooker, M. N. Sunder, C. *et al.* (2007). Zero downtime reconfiguration of distributed automation systems: the ε cedac approach. In *Third International Conference on Industrial Applications of Holonic and Multi-Agent Systems*. Springer-Verlag.

Sarvar, A. (1997). *CMOS* power consumption and C_{pd} calculation. Texas Insruments.

Shin, Y. and Choi, K. (1999). Power conscious fixed priority scheduling for hard real-time systems. In *36th Design Automation Conference*.

Singhoff, F. Legrand, J. Nana, L. and Marce, L. (2004). Cheddar: A flexible real time scheduling framework. In *Atlanta, GA, United states, Association for Computing Machinery*.

Wang, X. Khalgui, M. and Li, Z. W. (2004). Dynamic low power reconfigurations of embedded real-time systems. In *Technical Report TR-201012004. Available: http://sca.xidian.edu.cn/~wangx/SCATR201012004.pdf*.

Yao, F. Demers, A. and Shenker, S. (1995). A scheduling model for reduced cpu energy. In *Proceedings of IEEE annual foundations of computer science*.

Yun, H. and Kim, J. (2003). On energy-optimal voltage scheduling for fixed-priority hard real-time systems. In *ACM Transactions on Embedded Computing Systems(TECS)*.