Technical Reprot
Systems Control and Automation Group
School of Electro-Mechancial Engineering
Xidian University

Dynamic Low Power Reconfigurations of Embedded Real-Time Systems

Xi Wang[1]
[1]School of Electro-Mechanical Engineering
Xidian University
Xi'an 710071, China
Email: wanggkingg@gmail.com

Mohamed Khalgui[1,2], and Zhiwu Li[1,2]
[2]Institute of Computer Science
Martin Luther University
06120 Halle, Germany
Email: khalgui.mohamed@gmail.com;
zhwli@xidian.edu.cn

Contact Information:
Office Location: A1111, New Sci-Tech Building, North-Campus, Xidian University
Access ID: SCATR201012004
Tel: 86-29-88-20-24-41
Fax: 86-29-88-20-43-38
Website: http://sca.xidian.edu.cn/

2010.12.26

Any comments, suggestions, and criticisms are welcome.

Dynamic Low Power Reconfigurations of Embedded Real-Time Systems

Xi Wang[1]
[1]School of Electro-Mechanical Engineering
Xidian University
Xi'an 710071, China
Email: wanggkingg@gmail.com

Mohamed Khalgui[1,2], and Zhiwu Li[1,2]
[2]Institute of Computer Science
Martin Luther University
06120 Halle, Germany
Email: khalgui.mohamed@gmail.com;
zhwli@xidian.edu.cn

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.

Keywords: Embedded Real-Time System, Reconfiguration, Low Power, Agent-based Architecture.

1 Introduction

Nowadays, the minimization of 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. These systems should provide optimal real-time services with low power consumptions. Several interesting studies have been proposed in recent years for their real-time and low power scheduling [1, 2, 3, 4, 5]. Although these rich and useful studies provide interesting results, they do not address dynamic adaptations with low power consumptions of a system such that real-time solutions are automatically calculated for users when the system's behavior is dynamically changed. The new generations of embedded systems are addressing new criteria such as flexibility and agility. To reduce their cost, they should be changed and adapted to their environment without disturbances. Several interesting academic and industrial efforts have been made in recent years to develop reconfigurable embedded systems [6].

We distinguish in the existing research two reconfiguration policies: static and dynamic reconfigurations. Static reconfigurations are applied off-line to apply changes before a system starts [7], whereas dynamic reconfigurations are dynamically applied at run-time. Two cases exist in the latter: manual reconfigurations applied by users [8] and automatic reconfigurations applied by Intelligent Agents [9, 10]. This research focuses on the dynamic reconfigurations of real-time embedded systems that should meet deadlines defined in user requirements [11].

These systems using CMOS-based processors [12], are implemented by sets of tasks that are assumed independent, periodic and synchronous (i.e., they are simultaneously activated at $t = 0$ time units). Each set is executed when a particular reconfiguration scenario is applied at run-time. According to the work in [13], each task is characterized in this study by a functional priority defining its static priority in the system, a period equal to the deadline, and a Worst Case Execution Time (WCET). We define an automatic reconfiguration as any operation allowing additions-removals or also updates of tasks at run-time. Therefore the system's implementation is dynamically changed and should meet all considered deadlines of the current combination of tasks. In addition, the energy consumption should not be increased but should be stable or decreased after each possible reconfiguration in order to satisfy the battery's capacity.

To reach this goal, an agent-based architecture is defined where an intelligent software agent is proposed to check each dynamic (manual or automatic) reconfiguration scenario to be applied at run-time, and to help users for feasible and low power reconfigurations. If some tasks of the new execution model violate corresponding deadlines, or if the power consumption is increased, the agent proposes new solutions for users in order to re-obtain the system's feasibility with low power consumption.

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 minimization of the energy consumption is computed for each solution. Users should decide in this case which one to apply such that all tasks (new and old) meet deadlines with a low power consumption.

This original work is useful in industry to allow feasible real-time reconfigurable tasks with low power consumptions. Due to the work in [2], it is assumed that the processor utilization in this present paper is proportional to the processor speed. Therefore, it will be useful to keep the energy consumption stable or to minimize it. In order to guarantee feasible low power reconfigurations, users should decide which solution to apply into the real-time embedded system where all tasks (new and old) meet all deadlines and allow the minimization of the energy consumption. A tool was developed at Xidian University, Xi'an, China, to support all the services offered by the agent. The minimization of energy consumption is calculated for each solution to be offered by the agent. These services are useful and precious for users at run-time when reconfiguration scenarios violate deadlines of tasks or increase the energy consumption without any consideration of capacities of batteries.

In the next section, we analyze previous works on low power and real-time scheduling as well as reconfigurations of embedded architectures, before we formalize reconfigurable real-time systems in Section 3 and we evaluate the power consumption of such systems in Section 4. We define in Section 5 the agent-based architecture for low power reconfigurations of real-time embedded systems. This architecture is implemented, simulated and analyzed in Section 6. Finally, we conclude and present our future works in Section 7.

2 Related Work

We present related works dealing with reconfigurations, real-time and low-power scheduling of embedded systems.

2.1 Reconfigurations of Embedded Systems

Nowadays, a fair amount of research has been done to develop reconfigurable embedded systems. The work in [7] proposes reusable tasks to implement a broad range of systems where each task is statically reconfigured without any re-programming. This is accomplished by updating the supporting data structure, i.e., a state transition table, whereas the executable code remains unchanged and may be stored in permanent memory. The state transition table consists of multiple-output binary decision diagrams that represent the next-state mappings of various states and the associated control actions.

Rooker *et al.* propose in [8] a complete methodology based on the human intervention to dynamically reconfigure tasks. They present in addition an interesting experimentation showing the dynamic change of tasks by users without disturbing the whole system. The authors in [14] use Real-time-UML as a meta-model between design models of tasks and their implementation models to support dynamic user-based reconfigurations of control systems. The research in [15] proposes an agent-based reconfiguration approach to save the whole system when faults occur at run-time. Developed in [10] is an ontology-based agent to perform system's reconfigurations that adapt changes in requirements and also in environment. They are interested in studying reconfigurations of control systems when hardware faults occur at run-time.

As far as the authors know, no work is reported to address the problem of dynamic reconfigurations under real-time and low power constraints. We are interested in this original study in feasible low power dynamic reconfigurations of real-time systems where additions and removals of real-time tasks are applied at run-time. In this case, we aim to minimize the energy consumption after any reconfiguration scenario.

2.2 Real-Time Scheduling

Real-time scheduling has been extensively studied in the last three decades [11]. 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. Different classes of scheduling algorithms are followed: i) Clock-driven: primarily used for hard real-time systems where all properties of all jobs are known at design time. ii) Weighted round-robin: primarily used for scheduling a real-time traffic system in high-speed, and iii) Priority-driven: primarily used for more dynamic real-time systems with a mixture of time-based and event-based activities. Among all priority-driven policies, EDF or Least Time to Go (LTG) is a dynamic scheduling algorithm used in real-time operating systems. It places processes in a priority queue. Whenever a scheduling event occurs (a task is finished or a new task is released) the queue will be searched for the process closest to its deadline. This process is the next to be scheduled for execution. EDF is an optimal scheduling algorithm on preemptive uniprocessor 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, can be scheduled by a particular algorithm such that all deadlines are satisfied, then EDF is able to schedule this collection of jobs.

We present the following well-known concepts in the theory of real-time scheduling [13]:

- A periodic task τ_i $(C_i; T_i; D_i)$ is an infinite collection of jobs that have their request times constrained by a regular inter-arrival time T_i, a WCET C_i and a relative deadline D_i,

- A real-time scheduling problem is said feasible if there is at least one scheduling policy able to meet the deadlines of all the considered tasks,

- A set of tasks is schedulable with a given scheduling policy if and only if no jobs in this set miss their deadlines,
- A task is valid with a given scheduling policy if and only if no jobs in this task miss their deadlines,
- An idle time t of a processor is defined as a time where no tasks released before time t are pending at time t. An interval of successive idle times is classically called an idle period,
- A busy period is defined as a time interval $[a, b)$ such that there is no idle time in $[a, b)$ (the processor is fully busy) and such that both a and b are idle times,
- In the case of independent, periodic and synchronous tasks (e.g., simultaneously activated at $t = 0$), the verification of the system's schedulability is possible to be done in a hyper period $[0, LCM]$ where LCM is the Least Common Multiple, and
- $U = \sum_{i=1}^{n} \frac{C_i}{T_i}$ is the processor utilization factor of a system composed of n tasks. In the case of synchronous, independent and periodic tasks such that their deadlines are equal to their periods, $U \leq 1$ is a necessary and sufficient condition for the EDF-based scheduling of real time tasks.

We present an example of preemptive periodic tasks with EDF simulated by Cheddar [17] in Fig. 1, which contains 5 synchronous periodic tasks. All of them release at time equal to zero time unit. The first task, $task_a$, with period/deadline equal to 4 time units and WCET C_a equal to 1 time unit. The second one, $task_b$, with period/deadline equal to 5 time units and C_b equal to 1 time units. The third one, $task_c$, with period/deadline equal to 8 time units and C_c equal to 1 time units. The forth one, $task_d$, with period/deadline equal to 10 time units and C_d equal to 2 time units. The fifth one, $task_e$, with period/deadline equal to 20 time units and C_d equal to 2 time units. In this figure, we can see that $task_d$ and $task_d$ are preemptive, the processor utilization of the tasks set is 0.875, and all tasks respect their deadlines. The LCM is 40 time units.

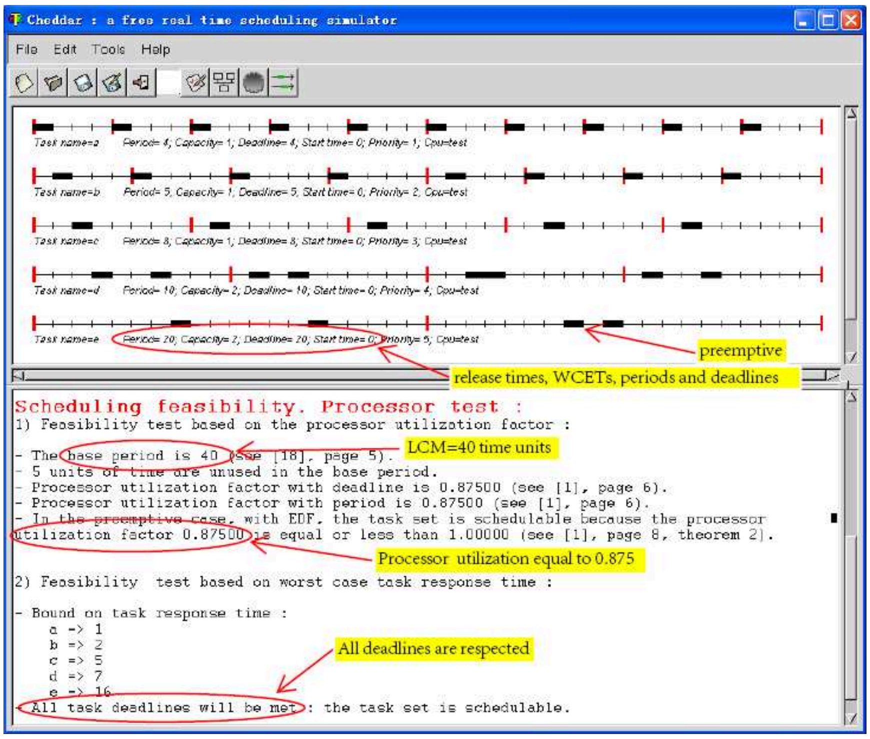
In our current work, it is assumed that dynamic priority of real-time tasks can be reconfigured at run-time. In this case, the energy consumption should be controlled after any reconfiguration scenario adding-removing-updating tasks consider about the limitations of capacity in the used batteries.

2.3 Low Power Scheduling

Several interesting research works have been proposed for low power and real-time scheduling of real-time embedded systems. Under the well-known FPP Policy, Shin and Choi [1] present a simple run-time strategy that reduces the energy consumption. Their work in [2] proposes an optimal algorithm with exponential complexity. Yun and Kim [3] prove that the problem of computing the voltage schedule for real-time tasks under FPP is *NP-hard* and propose an approximate solution to resolve the problem. If the well-known EDF policy is applied, Yao *et al.* in [4] propose an off-line algorithm to find a voltage schedule for independent tasks.

Over the past several years, many methods and techniques for low power consumption of real-time embedded systems have been reported [18, 19, 20]. Power-reduction techniques can be in general classified into two categories [21]: static and dynamic.

Dynamic techniques are generally easy to implement and applied at run-time. Examples of such techniques include those in [1, 22, 23, 24, 25, 26, 27]. We note also that several static power management policies have been investigated [4, 30, 28, 29]. Previous investigations on the voltage scheduling problem have focused mainly on real-time jobs running


Cheddar : a free real time scheduling simulator
File Edit Tools Help
preemptive
release times, WCETs, periods and deadlines
LCM=40 time units
Processor utilization equal to 0.875
All deadlines are respected

```
Task name=a     Period= 4; Capacity= 1; Deadline= 4; Start time= 0; Priority= 1; Cpu=test
Task name=b     Period= 5; Capacity= 1; Deadline= 5; Start time= 0; Priority= 2; Cpu=test
Task name=c     Period= 8; Capacity= 1; Deadline= 8; Start time= 0; Priority= 3; Cpu=test
Task name=d     Period= 10; Capacity= 2; Deadline= 10; Start time= 0; Priority= 4; Cpu=test
Task name=e     Period= 20; Capacity= 2; Deadline= 20; Start time= 0; Priority= 5; Cpu=test
```

```
Scheduling feasibility, Processor test :
1) Feasibility test based on the processor utilization factor :

- The base period is 40 (see [18], page 5).
- 5 units of time are unused in the base period.
- Processor utilization factor with deadline is 0.87500 (see [1], page 6).
- Processor utilization factor with period is 0.87500 (see [1], page 6).
- In the preemptive case, with EDF, the task set is schedulable because the processor
utilization factor 0.87500 is equal or less than 1.00000 (see [1], page 8, theorem 2).

2) Feasibility  test based on worst case task response time :

- Bound on task response time :
    a => 1
    b => 2
    c => 5
    d => 7
    e => 16
- All task deadlines will be met : the task set is schedulable.
```

Figure 1: An example of periodic tasks

under dynamic-priority scheduling algorithms such as the EDF algorithm [33, 28, 31, 32]. Although all these related studies are interesting, our current research addresses low power reconfigurations of real-time embedded systems when dynamic additions-removals-updates of tasks can be applied at run-time to save or improve the performance of the system. An original agent-based architecture is defined where an intelligent agent supervises the system's evolution and provides useful solutions for users after any reconfiguration scenario when deadlines are violated or also when the energy consumption is increased and does not consider the limitation of batteries. The agent proposes modify periods and WCETs or possibly to remove tasks. An original tool is developed that supports all these services which are precious for users at run-time.

3 Formalization of Reconfigurable Real-Time Systems

Recent developments show that dynamic reconfigurations of embedded real-time systems are useful technical solutions to save the whole software/hardware architecture when faults occur at run-time, or also to improve the system's performance under well-defined conditions. A reconfiguration scenario is assumed to be an operation allowing the addition-removal-update of tasks from/to the system. Nevertheless, each scenario should be applied while reducing the energy consumption which is a very important criterion. Indeed, when a scenario is dynamically applied such that new tasks are added to the system, the energy consumption should be stable or decreased. It is assumed in this research work that a real-time embedded system Sys composed of 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 to/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, i.e., all release times are equal to zero time units, and b) the period of each task is equal to the corresponding deadline.

We also assume in the following that the system Sys is dynamically reconfigured at run-time such that its new implementation is $Sys = \{\tau_1, \tau_2, \dots, \tau_n, \tau_{n+1}, \dots, \tau_m\}$. The subset $\{\tau_{n+1}, \dots, \tau_m\}$ is added to the initial implementation $\{\tau_1, \tau_2, \dots, \tau_n\}$. The processor utilization before and after the reconfiguration scenario is as follows:

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

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

4 Power Consumption of Reconfigurable Embedded Real-Time Systems

This section evaluates the new energy consumption after a reconfiguration scenario. In the literature, the power consumption (P) of a processor following the CMOS technology is determined by two components: static and dynamic power consumptions [12]. The static power consumption (P_S) is the product of the device leakage current and the supply voltage. It is assumed to be negligible in this research. The dynamic power consumption (P_D) is composed of two parts: Transient power consumption (P_T) and capacitive-load power consumption (P_L). P, P_T, and P_L can be expressed as follows:

$$P = P_D = P_T + P_L \tag{3}$$

$$P_T = C_{pd} \times V_{CC}^2 \times f_I \times N_{SW} \tag{4}$$

$$P_L = \sum (C_{Ln} \times f_{On}) \times V_{CC}^2 \tag{5}$$

where

- C_{pd} is the power consumption capacitance (F),
- f_I is the input frequency (Hz),
- f_{On} represents the different output frequencies at each output, numbered 1 through n (Hz),
- N_{SW} is the total number of outputs switching,
- V_{CC} is the supply voltage (V), and
- C_{Ln} represents the different load capacitances at each output, numbered 1 through n.

Accordingly, we have

$$P = [(C_{pd} \times f_I \times N_{SW}) + \sum (C_{Ln} \times f_{On})] \times V_{CC}^2 \tag{6}$$

Eq. (6) indicates that the energy consumption (P) of a CMOS-based processor is quadratically dependent on supply voltage V_{CC} and proportional to $(C_{pd} \times f_I \times N_{SW}) + \sum (C_{Ln} \times f_{On})$. In this case, reducing V_{CC} is an effective technique to minimize the energy consumption. In this study, we are interested in the influence of V_{CC} and the influence of $(C_{pd} \times f_I \times N_{SW}) + \sum (C_{Ln} \times f_{On})$ is ignored. We hence have

$$P \propto V_{CC}^2 \tag{7}$$

By [16], the voltage V_{CC} can be modified appropriately according to the processor speed S_p. Similar to [2], it is assumed that S_p is proportional to the V_{CC}, i.e.

$$V_{CC} \propto S_p \tag{8}$$

The work in [1] assumes that the change of the processor speed S_p due to the utilization (to adopt processor computational requirements) can save more energy, keep the processor work at the maximum speed (i.e., under the maximum supply voltage), and then bring it into a power-down mode. If the processor utilization is stable or decreased after any dynamic reconfiguration scenario, then the power will be stable or decreased. The full processor speed is assumed equal to 1. Suppose that S_p corresponds to processor utilization U. We have

$$V_{CC} \propto S_p = U \tag{9}$$

Substituting Eq. (9) into Eq. (7) leads to

$$P = kU^2 \tag{10}$$

Eq. (10) shows that energy consumption is quadratically dependent on processor utilization U. This is interesting and will be used to generate low power reconfigurations of embedded real-time systems.

Proposition 1 *Let U_{aft} and U_{bef} denote the processor utilization after and before a reconfiguration scenario is applied, respectively. If $U_{aft} \leq U_{bef}$, then the processor speed after the reconfiguration scenario can be adjusted to be no greater than that before. In this case the power will be stable or minimized.*

Proof: By Eq. (10), the energy consumption is quadratically dependent on U. If $U_{aft} \leq U_{bef}$, substituting it into Eq. (10) leads to $P_{aft} = kU_{aft}^2 \leq kU_{bef}^2 = P_{bef}$. Then the power will be stable or minimized.

5 Agent-based Architecture for Low Power Reconfigurations of Embedded Systems

An agent-based architecture is defined 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. In this case, it should propose useful functional (remove tasks) or temporal (change their parameters) solutions for the minimization of energy consumption. Three technical solutions are proposed by the agent: i) modification of periods (e.g., deadlines) of tasks, ii) modification of their WCETs, and iii) removal of some tasks. In this case, the users should decide a new low power configuration of the system by these solutions.

5.1 Modification of periods and deadlines

When a reconfiguration scenario is dynamically applied at run-time to add new tasks, the processor utilization of the system will be certainly increased. If the new utilization U_{aft} is greater than 1, then the system is not feasible. 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 not only than 1 but also U_{bef}. For the reconfigured system, we have

$$U_{aft} = \sum_{i=1}^{m} \frac{C_i}{T'_i} \tag{11}$$

where T'_1, T'_2, ..., and T'_m are the modified periods for m tasks. It is assumed that $T'_1 = T'_2 = \ldots = T'_m = T'$.

In order to minimize the energy consumption, based on Eq. (10), the processor utilization should be minimized.

$$U_{aft} = \sum_{i=1}^{m} \frac{C_i}{T'} \leq U_{bef} \tag{12}$$

where

$$T' \geq \frac{\sum\limits_{i=1}^{m} C_i}{U_{bef}} \tag{13}$$

Since each period and deadline should be an integer, Eq. (13) becomes Eq. (14):

$$T' = \lceil \frac{\sum\limits_{i=1}^{m} C_i}{U_{bef}} \rceil \tag{14}$$

In Eq. (14), T' is the modified period of each task. The processor utilization after reconfiguration of the processor is as follows:

$$U_{aft} = \sum_{i=1}^{m} \frac{C_i}{T'} \tag{15}$$

From Eq. (10), the new energy consumption is

$$P_{aft} = kU_{aft}^2 \tag{16}$$

Similarly, the initial energy consumption can be expressed as:

$$P_{bef} = kU_{bef}^2 \tag{17}$$

5.2 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}. Eq. (12) leads to the truth of the following result:

$$U_{aft} = \sum_{i=1}^{m} \frac{C'_i}{T_i} \tag{18}$$

where C'_1, C'_2, ..., and C'_m are the new WCETs of tasks.

Suppose that $C'_1 = C'_2 = \cdots = C'_m = C'$. In order to make $U_{aft} \leq U_{bef}$, we can get:

$$U_{aft} = \sum_{i=1}^{m} \frac{C'}{T_i} \leq U_{bef} \tag{19}$$

where

$$C' \leq \frac{U_{bef}}{\sum\limits_{i=1}^{m} \frac{1}{T_i}} \tag{20}$$

Since each WCET should be an integer, Eq. (20) turns into Eq. (21):

$$C' = \lfloor \frac{U_{bef}}{\sum_{i=1}^{m} \frac{1}{T_i}} \rfloor \tag{21}$$

In Eq. (21), C' is the modified WCET of each task. The actual utilization of the processor is as follows:

$$U_{aft} = C' \times \sum_{i=1}^{m} \frac{1}{T_i} \tag{22}$$

Eq. (16) can be used to calculate the total energy consumption.

5.3 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 removing tasks by their functional priorities or according to their processor utilization. This solution is useful in many scenarios.

5.3.1 First Policy: Functional Priority Criterion.

By defining for each task τ_i a static priority S_i, the agent suggests removing 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

$$U_{aft} = \sum_{F_i=1}^{k} \frac{C_i}{T_i} \leq U_{bef} \tag{23}$$

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. (16).

5.3.2 Second Policy: Processor Utilization Criterion

It is similar to the first. Their difference is that the second policy does not 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.

6 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.

6.1 Implementation of the Reconfiguration Agent

It is presented that the agent's implementation 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 model simulator Cheddar [17] to check its feasibility. This implementation is tested in our research laboratory at Xidian University by assuming several cases of systems.

Eqs. (1), (10), (14), and (21) can be used to calculate U_{bef}, P_{bef}, T', and C', respectively. According to Eqs. (11) and (19), it can be calculated that the utilization U^1_{aft} after the modification of periods (and deadlines), and the utilization U^2_{aft} after the modification of WCETs, respectively. U^3_{aft} and U^4_{aft} correspond to the utilization after tasks are removed by the two fixed policies, as shown in Eq. (23). We use P^1_{aft}, P^2_{aft}, P^3_{aft}, and P^4_{aft} to indicate the power consumption after periods (and deadlines) modification, after WCET modification, after tasks removal by static priorities, and after tasks removal by utilizations, respectively. All of them can be calculated by Eq. (10).

The power reduction PR and power decrease PD^i of each technical solution in Algorithm 1 are as follow:

$$PR_{bef} = 1 - P_{bef} \tag{24}$$

$$PR^i_{aft} = 1 - (P^i_{aft})(for : i = 1, 2, 3, 4) \tag{25}$$

$$PD^i = PR^i_{aft} - PR_{bef}(for : i = 1, 2, 3, 4) \tag{26}$$

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

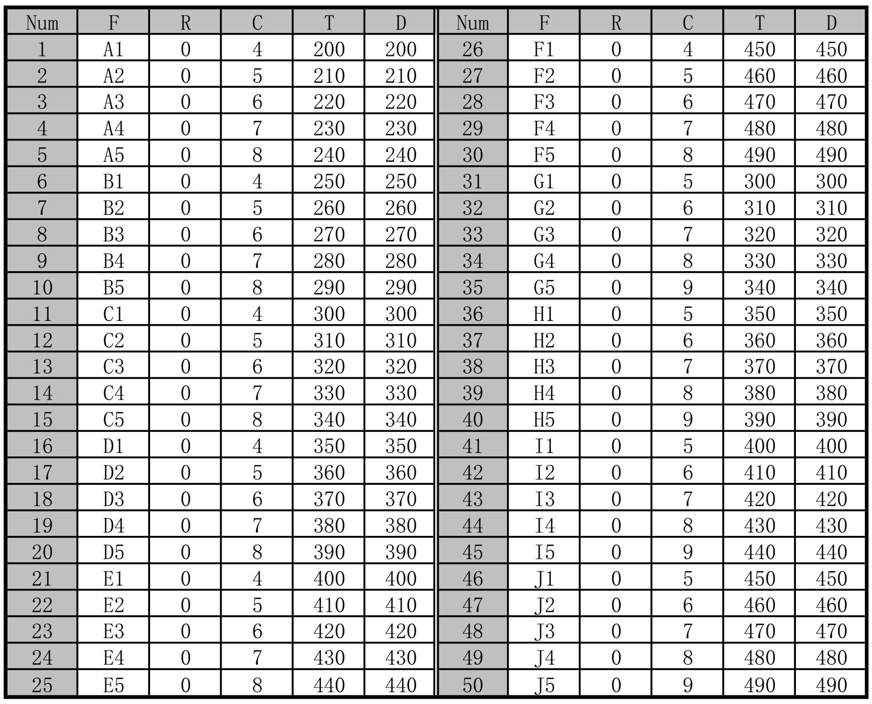
6.2 Simulations

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:** All the following initial tasks of the system are in the file "*system.txt*" (F_i defines the function. R_i, C_i, T_i, and D_i define the temporal parameters of each task, as shown in Fig. 2).
- **Added Tasks:** All the added new tasks are described in the file "*add.txt*" (F_i defines the function. R_i, C_i, T_i, and D_i define the temporal parameters of each task, as shown in Fig. 3).
- **Static priorities:** All the functional priorities of the system's tasks are defined in the file "*priority.txt*" (F_i and S_i define function and static priority of each task, respectively, as shown in Fig. 4).

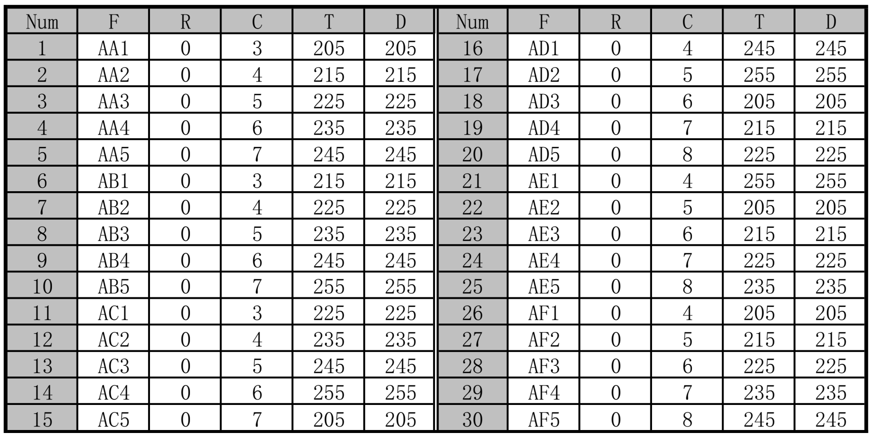
Several simulations are performed to apply the proposed technical solutions for low power reconfigurations of the system. The initial processor utilization U_{bef} is 0.91224. If only the task AA1 is added in the system, the processor utilization U_{aft} becomes 0.926874 which is less than 1. The system is still feasible where all deadlines are satisfied. Nevertheless, the energy consumption is increased ($U_{bef} \leq U_{aft}$) which is not usually accepted. By applying the first solution where periods and deadlines of tasks are modified to be 355, the processor utilization becomes $U_{aft} = 0.909859$. By Eqs. (24) and (25), the power reduction before and after the automatic reconfiguration is as follows:

```
Algorithm 1 Low-Power Reconfigurations
input "system.txt" file;
// the system initial configuration
input "add.txt" file;
// addition of new tasks
input "priority.txt" file;
// functional priorities of tasks
compute ($U_{bef}$);
calculate_period $T'$;
// solution1 to compute the new period of tasks
for ($i = 1, i = size(sys) + size(add), i++$)
// to compute the new utilization when periods are modified
    $U_i = C_i/T'$;
    $\sum U^1_{aft} += U_i$;
endfor;
Evaluate_energy($U_{bef}$, $U^1_{aft}$);
calculate_execution_time $C'$;
// solution2 to compute the new WCET of tasks
for ($i = 1, i = size(sys) + size(add), i++$)
// to compute the new utilization when execution times are modified
    $U_i = C'/T_i$;
    $\sum U^2_{aft} += U_i$;
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 ($Sys_{new1}$);
// solution3 to remove possible tasks (first criterion)
for ($i = 1, i = size(Sys_{new1}), i++$)
// to compute the new utilization when tasks are removed (first criterion)
    $\sum U^3_{aft} += U_i$;
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;
```



Num	F	R	C	T	D	Num	F	R	C	T	D
1	A1	0	4	200	200	26	F1	0	4	450	450
2	A2	0	5	210	210	27	F2	0	5	460	460
3	A3	0	6	220	220	28	F3	0	6	470	470
4	A4	0	7	230	230	29	F4	0	7	480	480
5	A5	0	8	240	240	30	F5	0	8	490	490
6	B1	0	4	250	250	31	G1	0	5	300	300
7	B2	0	5	260	260	32	G2	0	6	310	310
8	B3	0	6	270	270	33	G3	0	7	320	320
9	B4	0	7	280	280	34	G4	0	8	330	330
10	B5	0	8	290	290	35	G5	0	9	340	340
11	C1	0	4	300	300	36	H1	0	5	350	350
12	C2	0	5	310	310	37	H2	0	6	360	360
13	C3	0	6	320	320	38	H3	0	7	370	370
14	C4	0	7	330	330	39	H4	0	8	380	380
15	C5	0	8	340	340	40	H5	0	9	390	390
16	D1	0	4	350	350	41	I1	0	5	400	400
17	D2	0	5	360	360	42	I2	0	6	410	410
18	D3	0	6	370	370	43	I3	0	7	420	420
19	D4	0	7	380	380	44	I4	0	8	430	430
20	D5	0	8	390	390	45	I5	0	9	440	440
21	E1	0	4	400	400	46	J1	0	5	450	450
22	E2	0	5	410	410	47	J2	0	6	460	460
23	E3	0	6	420	420	48	J3	0	7	470	470
24	E4	0	7	430	430	49	J4	0	8	480	480
25	E5	0	8	440	440	50	J5	0	9	490	490

Figure 2: Initial System Tasks



Num	F	R	C	T	D	Num	F	R	C	T	D
1	AA1	0	3	205	205	16	AD1	0	4	245	245
2	AA2	0	4	215	215	17	AD2	0	5	255	255
3	AA3	0	5	225	225	18	AD3	0	6	205	205
4	AA4	0	6	235	235	19	AD4	0	7	215	215
5	AA5	0	7	245	245	20	AD5	0	8	225	225
6	AB1	0	3	215	215	21	AE1	0	4	255	255
7	AB2	0	4	225	225	22	AE2	0	5	205	205
8	AB3	0	5	235	235	23	AE3	0	6	215	215
9	AB4	0	6	245	245	24	AE4	0	7	225	225
10	AB5	0	7	255	255	25	AE5	0	8	235	235
11	AC1	0	3	225	225	26	AF1	0	4	205	205
12	AC2	0	4	235	235	27	AF2	0	5	215	215
13	AC3	0	5	245	245	28	AF3	0	6	225	225
14	AC4	0	6	255	255	29	AF4	0	7	235	235
15	AC5	0	7	205	205	30	AF5	0	8	245	245

Figure 3: Added Tasks

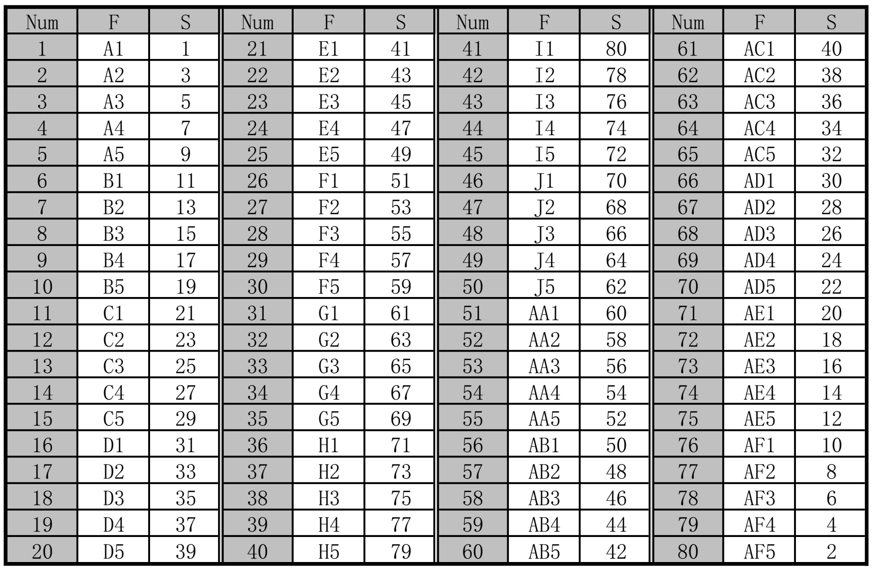
$$PR_{bef} = 1 - 0.91224^2 = 16.7819\% \tag{27}$$

$$PR_{aft} = 1 - 0.909859^2 = 17.2156\% \tag{28}$$

Fig. 5 shows a software package for Algorithm 1 to compute the new periods as well as the new processor utilization. Due to Eq. (26), the decrease of the energy consumption PD is as follows:

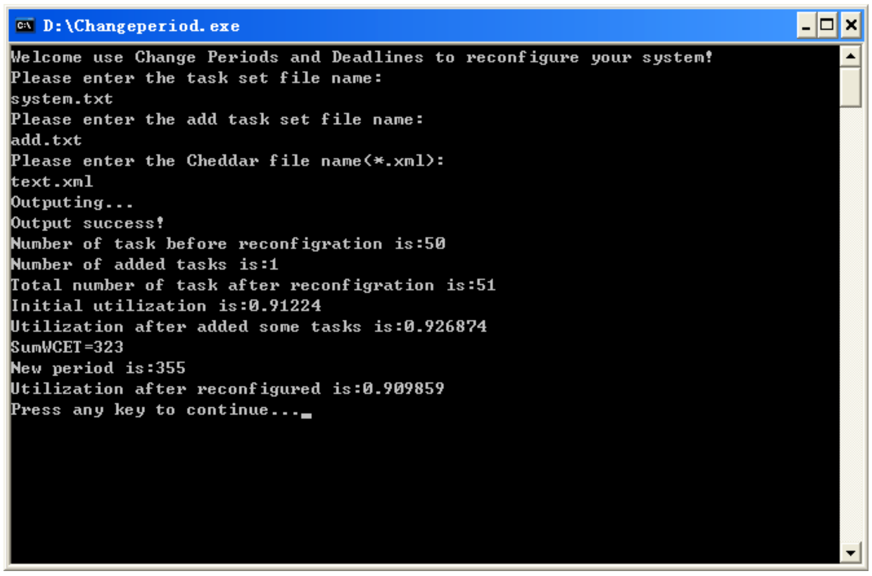
$$PD = PR_{aft} - PR_{bef} = 0.433749\% \tag{29}$$

Similarly, the second and third solutions are applied to decrease the energy consumption of the system. By modifying the execution times, the energy consumption decreases by 2.45583%. It decreases also in the third case by 3.77711% and by 3.37665% when we remove tasks according to their static priorities and their utilizations, respectively.



Num	F	S	Num	F	S	Num	F	S	Num	F	S
1	A1	1	21	E1	41	41	I1	80	61	AC1	40
2	A2	3	22	E2	43	42	I2	78	62	AC2	38
3	A3	5	23	E3	45	43	I3	76	63	AC3	36
4	A4	7	24	E4	47	44	I4	74	64	AC4	34
5	A5	9	25	E5	49	45	I5	72	65	AC5	32
6	B1	11	26	F1	51	46	J1	70	66	AD1	30
7	B2	13	27	F2	53	47	J2	68	67	AD2	28
8	B3	15	28	F3	55	48	J3	66	68	AD3	26
9	B4	17	29	F4	57	49	J4	64	69	AD4	24
10	B5	19	30	F5	59	50	J5	62	70	AD5	22
11	C1	21	31	G1	61	51	AA1	60	71	AE1	20
12	C2	23	32	G2	63	52	AA2	58	72	AE2	18
13	C3	25	33	G3	65	53	AA3	56	73	AE3	16
14	C4	27	34	G4	67	54	AA4	54	74	AE4	14
15	C5	29	35	G5	69	55	AA5	52	75	AE5	12
16	D1	31	36	H1	71	56	AB1	50	76	AF1	10
17	D2	33	37	H2	73	57	AB2	48	77	AF2	8
18	D3	35	38	H3	75	58	AB3	46	78	AF3	6
19	D4	37	39	H4	77	59	AB4	44	79	AF4	4
20	D5	39	40	H5	79	60	AB5	42	80	AF5	2

Figure 4: Static Priorities



```
D:\Changeperiod.exe
Welcome use Change Periods and Deadlines to reconfigure your system!
Please enter the task set file name:
system.txt
Please enter the add task set file name:
add.txt
Please enter the Cheddar file name(*.xml):
text.xml
Outputing...
Output success!
Number of task before reconfigration is:50
Number of added tasks is:1
Total number of task after reconfigration is:51
Initial utilization is:0.91224
Utilization after added some tasks is:0.926874
SumWCET=323
New period is:355
Utilization after reconfigured is:0.909859
Press any key to continue...
```

Figure 5: Developed Software

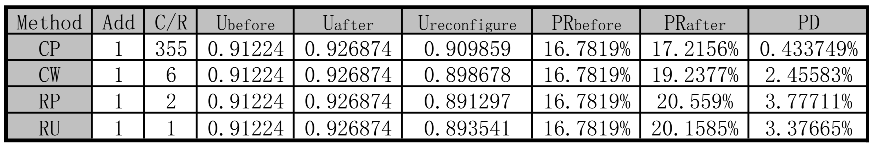
Table. 6 shows the initial utilization U_{bef}, the utilization U_{aft} including the task $AA1$, the utilization after the reconfiguration scenario U_{rec}, the power reduction before the reconfiguration scenario PR_{bef}, the power reduction after such scenario PR_{aft}, and the energy consumption decrease PD. The parameters MP, MW, RP, and RU, correspond to the modification of periods, the modification of WCETs, the removal of tasks according to static priorities, and the removal of tasks according to processor utilizations, respectively.

Table 7 shows the useful solutions for low power reconfigurations of the system when the first 10 tasks are added. If all the 30 tasks are added, the solutions are shown in Table. 8.

All the experimental data of simulations on maintaining or decrease of the energy consumption are shown in the Appendix of this present paper.

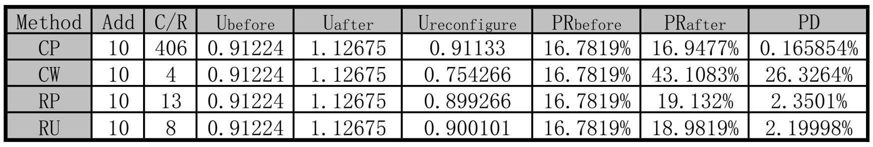
6.3 Analysis

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



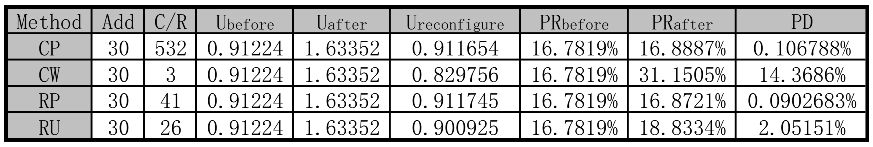
Method	Add	C/R	Ubefore	Uafter	Ureconfigure	PRbefore	PRafter	PD
CP	1	355	0.91224	0.926874	0.909859	16.7819%	17.2156%	0.433749%
CW	1	6	0.91224	0.926874	0.898678	16.7819%	19.2377%	2.45583%
RP	1	2	0.91224	0.926874	0.891297	16.7819%	20.559%	3.77711%
RU	1	1	0.91224	0.926874	0.893541	16.7819%	20.1585%	3.37665%

Figure 6: Low power reconfiguration after the addition of $AA1$



Method	Add	C/R	Ubefore	Uafter	Ureconfigure	PRbefore	PRafter	PD
CP	10	406	0.91224	1.12675	0.91133	16.7819%	16.9477%	0.165854%
CW	10	4	0.91224	1.12675	0.754266	16.7819%	43.1083%	26.3264%
RP	10	13	0.91224	1.12675	0.899266	16.7819%	19.132%	2.3501%
RU	10	8	0.91224	1.12675	0.900101	16.7819%	18.9819%	2.19998%

Figure 7: Low power reconfiguration after the addition of the first ten tasks



Method	Add	C/R	Ubefore	Uafter	Ureconfigure	PRbefore	PRafter	PD
CP	30	532	0.91224	1.63352	0.911654	16.7819%	16.8887%	0.106788%
CW	30	3	0.91224	1.63352	0.829756	16.7819%	31.1505%	14.3686%
RP	30	41	0.91224	1.63352	0.911745	16.7819%	16.8721%	0.0902683%
RU	30	26	0.91224	1.63352	0.900925	16.7819%	18.8334%	2.05151%

Figure 8: Low power reconfiguration after the addition of the thirty new tasks

6.3.1 First Solution

Due to Eqs. (14) and (16), 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 results are shown in Fig. 9 in which we find a decrease of consumption between 0 and 0.45%. The energy reduction is piecewise approximately linear depending on the $\sum C_i = 3, 4, 5, \ldots, 165$.

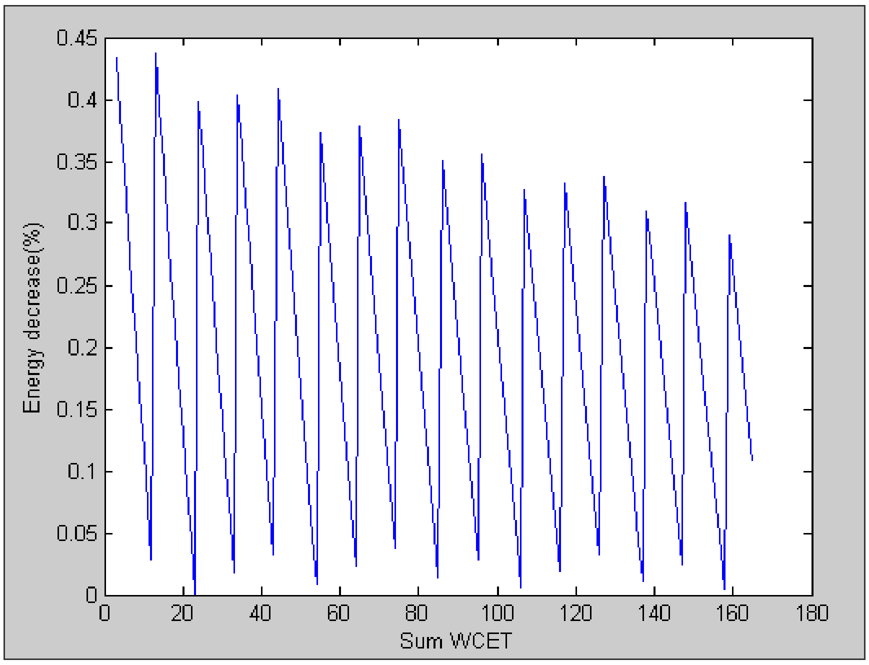


Figure 9: Power save by change periods

6.3.2 Second Solution

From Eqs. (22) and (16), 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. Simulations are made for subsets of tasks with the range of $\sum 1/T_i$ between 0 and 0.131684. The results are shown in Fig. 10 where the minimization of the energy consumption is between 0 and 35%. This simulation result proves the benefits of the second solution more than the first. Nevertheless, sometimes, the modification of periods is more simpler than the minimization of WCET. It is unclear whether the second solution is definitely better than the first.

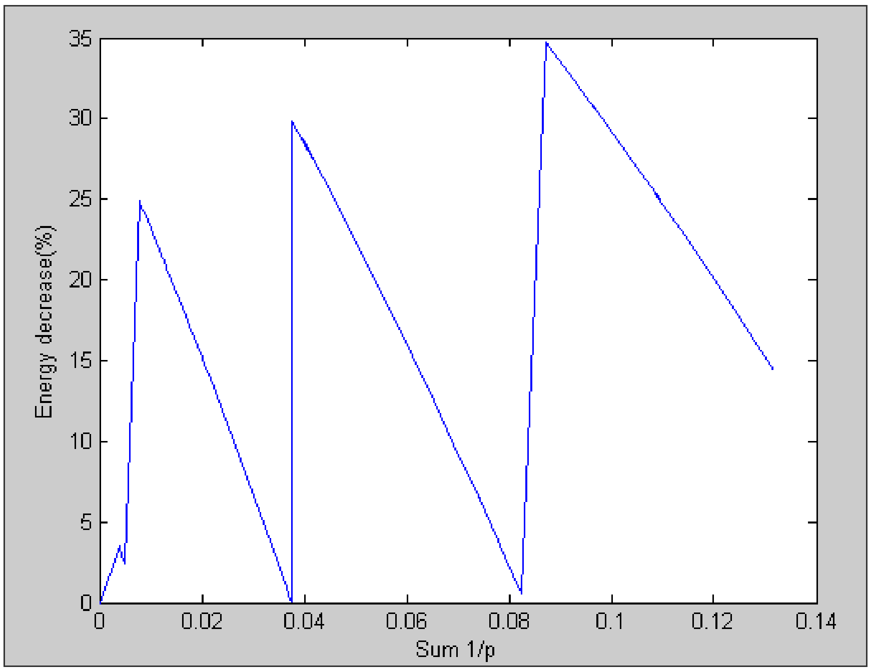


Figure 10: Power save by change WCETs

6.3.3 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. 11, 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 to that added. This second policy is more useful than the first. Fig. 12 shows the minimization of the energy consumption when the first (continuous line) and the second (dotted line) policies are applied. It present that the energy consumption remains to be minimal when we apply the second solution where WCETs of tasks are modified.

7 Conclusion

The 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

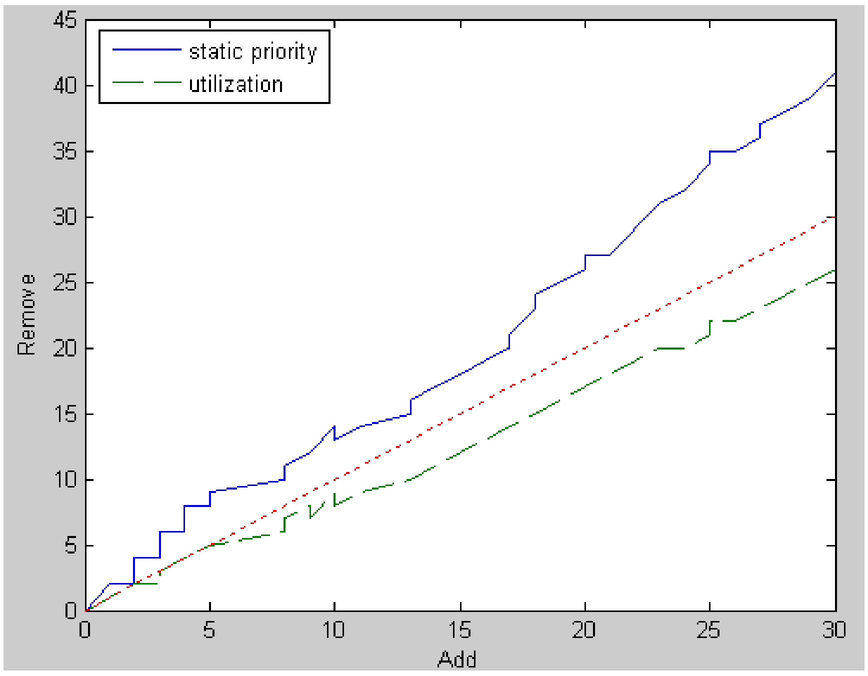


Figure 11: Comparing the number of removed task between two remove strategies

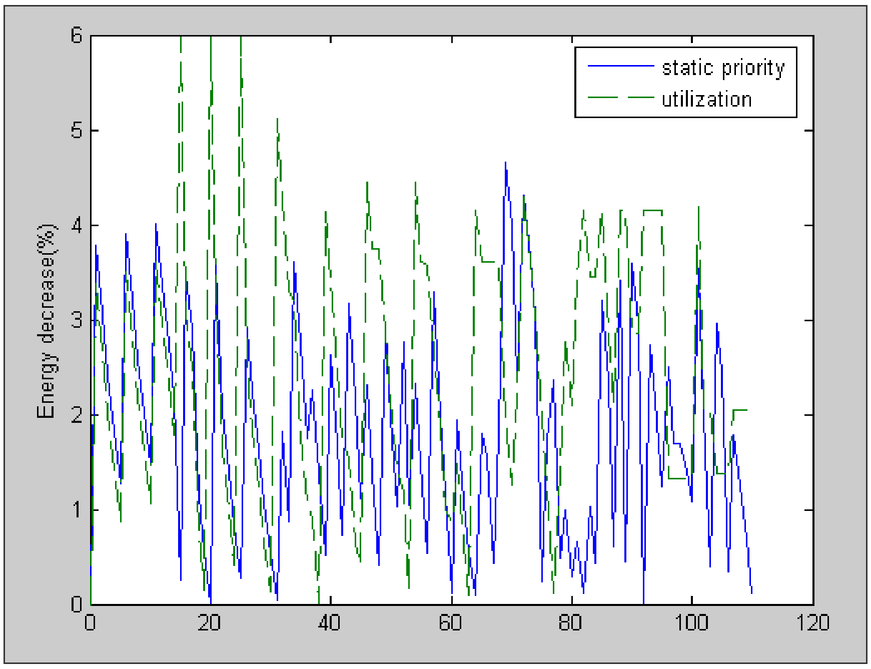


Figure 12: Comparing power save between two remove strategies

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 increase 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.

References

[1] Y. Shin and K. Choi, ”Power conscious fixed priority scheduling for hard real-time systems,” ACM. In: *36th Design Automation Conference*, pp. 134-139, 1999.

[2] G. Quan and X. Hu, ”Minimum energy fixed-priority scheduling for variable voltage processors, Design, Automation and Test,” In: *Europe Conference and Exhibition*, pp. 782-787, 2002.

[3] H. Yun and J. Kim, ”On energy-optimal voltage scheduling for fixed-priority hard real-time systems,” *ACM Transactions on Embedded Computing Systems(TECS)*, vol. 2, no. 3, pp. 393-430, 2003.

[4] F. Yao, A. Demers, and S. Shenker, ”A scheduling model for reduced CPU energy,” In: *Proceedings of IEEE annual foundations of computer science*, pp. 374-382, 1995.

[5] B. Gaujal and N. Navet, ”Dynamic Voltage Scaling under EDF Revisited, Real-Time Systems,” *Springer Verlag, 37(1), Some results are available as research report INRIA RR-5125*, pp.77-97, 2007.

[6] A.-L. Gehin and M. Staroswiecki, ”Reconfiguration Analysis Using Generic Component Models,” *IEEE Transactions on Systems, Machine and Cybernetics*, vol. 38, no. 3, pp. 575-583, 2008.

[7] C. Angelov, K. Sierszecki and N. Marian, ”Design models for reusable and reconfigurable state machines,” In: *L.T. Yang, et al. (Eds.), EUC, LNCS, 3824, International Federation for Information Processing*, pp. 152-163, 2005.

[8] M. N. Rooker, C. Sunder, T. Strasser, A. Zoitl, O. Hummer and G. Ebenhofer, ”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, 2007.

[9] M. Khalgui, O. Mosbahi, ZW. Li and H.-M. Hanisch, ”Reconfigurable Multi-Agent Embedded Control Systems: From Modelling to Implementation,” *IEEE Transactions on Computers*, 2010.

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

[11] S. Baruah, and J. Goossens, ”Scheduling Real-time Tasks: Algorithms and Complexity,” In: *In Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, Joseph Y-T Leung (ed). Chapman Hall/ CRC Press, 2004.

[12] A. Sarvar, $CMOS$ Power Consumption and C_{pd} Calculation, *Texas Insruments*, 1997.

[13] C. L. Liu, and J. W. Layland, ”Scheduling algorithms for multiprogramming in a hard real time environment,” *J. ACM*, vol. 20, pp. 46-61, 1973.

[14] K. Thramboulidis, G. Doukas and A. Frantzis, ”Towards an implementation model for FB-based reconfigurable distributed control applications,” In: *Proceedings of 7th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, pp. 193-200, 2004.

[15] R. W. Brennan, M. Fletcher and D. H. Norrie, "A holonic approach to reconfiguring realtime distributed control systems," In: *Multi-Agent Systems and Applications*, MASA'01, Springer-Verlag, 2001.

[16] T. D. Burd and R. W. Brodersen, "Design issues for dynamic voltage scaling," *ISLPED*, 2000.

[17] F. Singhoff, J. Legrand, L. Nana and L. Marce, "Cheddar: A flexible real time scheduling framework," *Atlanta, GA, United states, Association for Computing Machinery*, pp. 1-8, 2004.

[18] L. Benini, A. Bogliolo and G. Micheli, "A survey of design techniques for system-level dynamic power management,: *IEEE Trans. VLSI Sys.*, vol. 8, no. 3, pp. 299-316, 2000.

[19] G. Micheli and L. and Benini, "System-level low power optimization: Techniques and tools," *Trans. Design Auto. of Electr. Sys.*, vol. 5, no.2, 2000.

[20] M. Pedram, "Power minimization in ic design: principles and applications," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 1, no. 1, pp. 56-66, 1996.

[21] J. Rabaey and M. Pedram, Low Power Design Methodologies. *Kluwer*, 1996.

[22] YH. Lu, EY. Chung, T. Simunic, G. De. Micheli, and L. Benini, "Quantitative comparison of power management algorithms," *Design, Automation and Test in Europe Conference and Exhibition*, pp. 20-26, 2000.

[23] L. Benini, A. Bogliolo, G. Paleologo and G. Micheli, "Policy optimization for dynamic power management," *IEEE Trans. CAD and Sys.*, vol. 18, no. 6, pp. 813-833, 1999.

[24] I. Hong, M. Potkonjak and M. B. Srivastava, "On-line scheduling of hard real-time tasks on variable voltage processor," *Proceedings of ICCAD*, pp. 653-656, 1998.

[25] Q. Qiu, Q. Wu and M. Pedram, "Dynamic power management of complex system using generalized stochastic petrinets," *DAC*, pp. 352-356, 2000.

[26] D. Ramanathan and R. Gupta, "System level online power management algorithms," *DATE*, pp. 606-611, 2000.

[27] M. Srivastava, A. Chandrakasan and R. Brodersen, "Predictive system shutdown and other architectural techniques for energy efficient programmable computation," *IEEE Trans. VLSI Sys.*, vol. 4, pp. 42-55, 1996.

[28] I. Hong, G. Qu, M. Potkonjak and M. B. Srivastava, "Synthesis techniques for low-power hard real-time systems on variable voltage processors," *Proceedings of RTSS*, pp. 178-187, 1998.

[29] T. Ishihara and H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processors," *ISLPED*, pp. 197-202, 1998.

[30] G. Quan and X. S. Hu, "Energy efficient fixed-priority scheduling for real-time systems on voltage variable processors," *Design Automation Conference*, pp. 828-833, 2001.

[31] H. Aydin, R. Melhem, D. Mosse and P. M. Alvarez, "Dynamic and aggressive scheduling techniques for power-aware real-time systems," In: *Proceedings of Real-Time Systems Symposium*, 2001.

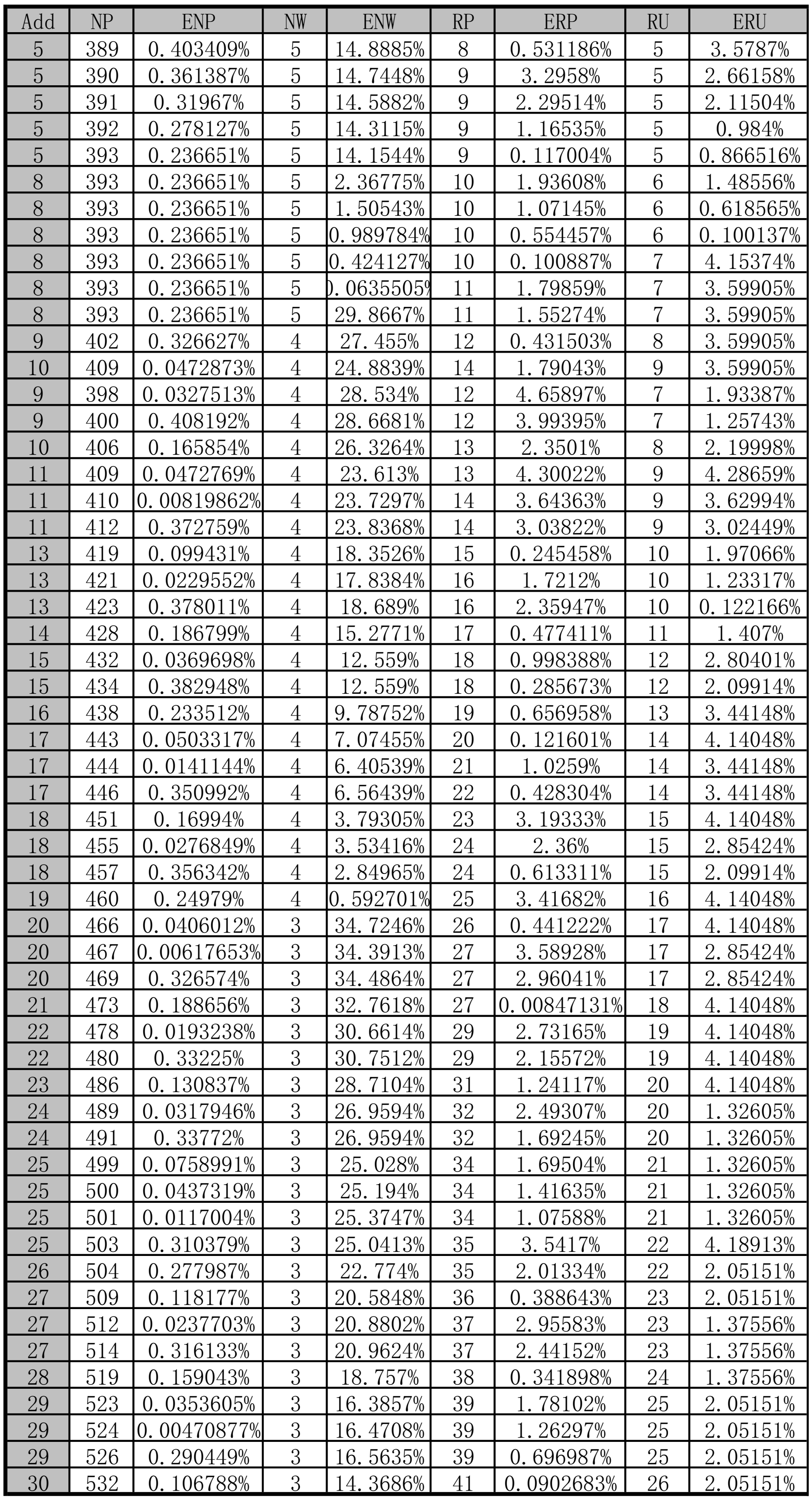
[32] W. Kim, J. Kim and S. L. Min, "A dynamic voltage scaling algorithm for dynamic-priority hard real-time systems using slack time analysis," In: *Proceedings of Design, Automation and Test in Europe*, 2002.

[33] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," In: *Proceedings of ACM Symposium on Operating Systems Principles*, 2001.

Appendix

Add	NP	ENP	NW	ENW	RP	ERP	RU	ERU
1	355	0.433749%	6	0.024558	2	3.77711%	1	3.37665%
1	356	0.3877%	6	2.70032%	2	3.06776%	1	2.66551%
1	357	0.341833%	6	2.92276%	2	2.41871%	1	2.01484%
1	358	0.296225%	6	3.126%	2	1.82261%	1	1.41725%
1	359	0.250866%	6	3.3124%	2	1.27323%	1	0.866516%
1	355	0.433749%	6	2.70032%	2	3.8984%	1	3.49825%
1	356	0.3877%	6	2.92276%	2	3.21574%	1	2.81387%
1	357	0.341833%	6	3.126%	2	2.58863%	1	2.18517%
1	358	0.296225%	6	3.3124%	2	2.01054%	1	1.60564%
1	359	0.250866%	6	3.48401%	2	1.47597%	1	1.06975%
1	355	0.433749%	6	2.92276%	2	4.00884%	1	3.60894%
1	356	0.3877%	6	3.126 %	2	3.35102%	1	2.94948%
1	357	0.341833%	6	3.3124 %	2	2.74452%	1	2.34145%
1	358	0.296225%	6	3.48401%	2	2.18354%	1	1.77908%
1	359	0.250866%	6	2.45583%	2	0.260821%	1	5.97047%
1	356	0.3877%	6	3.3124 %	2	3.47515%	1	3.07392%
1	357	0.341833%	6	3.48401%	2	2.88804%	1	2.48533%
1	358	0.296225%	6	2.45583%	2	1.14703%	1	0.739995%
1	359	0.250866%	6	2.70032%	2	0.549872%	1	0.141355%
1	360	0.205775%	6	2.92276%	2	0.00391454%	1	5.97047%
1	356	0.3877%	6	3.48401%	2	3.58946%	1	3.18851%
1	357	0.341833%	6	2.45583%	2	2.02848%	1	1.62364%
1	358	0.296225%	6	2.70032%	2	1.39349%	1	0.987071%
1	359	0.250866%	6	2.92276%	2	0.812806%	1	0.404927%
1	360	0.205775%	6	3.126 %	2	0.279723%	2	5.97047%
1	356	0.3877%	6	2.45583%	2	2.90519%	1	2.50252%
1	357	0.341833%	6	2.70032%	2	2.23279%	1	1.82845%
1	358	0.296225%	6	2.92276%	2	1.61774%	1	1.21186%
1	359	0.250866%	6	3.126 %	2	1.05299%	1	0.645703%
1	360	0.205775%	6	3.3124 %	2	0.532617%	1	0.124057%
2	361	0.160889%	5	24.8906%	2	0.0516027%	2	5.11393%
2	362	0.116286%	5	24.7683%	3	1.82081%	2	4.24864%
2	363	0.0718445%	5	24.6355%	3	0.862135%	2	3.30433%
2	364	0.0276625%	5	24.4906%	4	3.61288%	2	3.18851%
2	366	0.43799%	5	24.3321%	4	2.74584%	2	1.77908%
2	367	0.393276%	5	24.1579%	4	1.75428%	2	1.06975%
3	368	0.348768%	5	21.8572%	4	2.25422%	2	0.89967%
3	369	0.304507%	5	21.7317%	4	1.37328%	2	0.0114068%
3	370	0.260486%	5	21.6062%	4	0.516692%	3	4.13039%
3	371	0.216712%	5	21.4805%	5	2.61749%	3	3.4072%
3	372	0.173177%	5	21.3439%	5	1.69485%	3	2.48909%
3	373	0.12975%	5	21.2072%	5	0.735447%	3	1.75397%
3	374	0.0866383%	5	21.0704%	6	3.16001%	3	1.53435%
3	375	0.0437438%	5	20.9212%	6	2.15849%	3	0.645703%
3	376	0.00101179%	5	20.7718%	6	1.11649%	3	0.404927%
4	378	0.398494%	5	18.7468%	6	2.30667%	4	4.44325%
4	379	0.355344%	5	18.6183%	6	1.39706%	4	3.74559%
4	380	0.31229%	5	18.4786%	6	0.404288%	4	3.74559%
4	381	0.269519%	5	18.3497%	7	2.88304%	4	2.87278%
4	382	0.226997%	5	18.2208%	7	1.97667%	4	2.14387%
4	383	0.184627%	5	18.0807%	7	1.0189%	4	1.40717%
4	384	0.142507%	5	17.7875%	8	2.76524%	4	1.18709%
4	385	0.100604%	5	17.1528%	8	1.04091%	4	0.124057%
5	386	0.0588551%	5	15.1528%	8	2.32615%	5	4.44325%
5	387	0.0172675%	5	15.0207%	8	1.47455%	5	3.60284%

Figure 13: New Configuration of the System (1)



Add	NP	ENP	NW	ENW	RP	ERP	RU	ERU
5	389	0.403409%	5	14.8885%	8	0.531186%	5	3.5787%
5	390	0.361387%	5	14.7448%	9	3.2958%	5	2.66158%
5	391	0.31967%	5	14.5882%	9	2.29514%	5	2.11504%
5	392	0.278127%	5	14.3115%	9	1.16535%	5	0.984%
5	393	0.236651%	5	14.1544%	9	0.117004%	5	0.866516%
8	393	0.236651%	5	2.36775%	10	1.93608%	6	1.48556%
8	393	0.236651%	5	1.50543%	10	1.07145%	6	0.618565%
8	393	0.236651%	5	0.989784%	10	0.554457%	6	0.100137%
8	393	0.236651%	5	0.424127%	10	0.100887%	7	4.15374%
8	393	0.236651%	5	.0635505	11	1.79859%	7	3.59905%
8	393	0.236651%	5	29.8667%	11	1.55274%	7	3.59905%
9	402	0.326627%	4	27.455%	12	0.431503%	8	3.59905%
10	409	0.0472873%	4	24.8839%	14	1.79043%	9	3.59905%
9	398	0.0327513%	4	28.534%	12	4.65897%	7	1.93387%
9	400	0.408192%	4	28.6681%	12	3.99395%	7	1.25743%
10	406	0.165854%	4	26.3264%	13	2.3501%	8	2.19998%
11	409	0.0472769%	4	23.613%	13	4.30022%	9	4.28659%
11	410	0.00819862%	4	23.7297%	14	3.64363%	9	3.62994%
11	412	0.372759%	4	23.8368%	14	3.03822%	9	3.02449%
13	419	0.099431%	4	18.3526%	15	0.245458%	10	1.97066%
13	421	0.0229552%	4	17.8384%	16	1.7212%	10	1.23317%
13	423	0.378011%	4	18.689%	16	2.35947%	10	0.122166%
14	428	0.186799%	4	15.2771%	17	0.477411%	11	1.407%
15	432	0.0369698%	4	12.559%	18	0.998388%	12	2.80401%
15	434	0.382948%	4	12.559%	18	0.285673%	12	2.09914%
16	438	0.233512%	4	9.78752%	19	0.656958%	13	3.44148%
17	443	0.0503317%	4	7.07455%	20	0.121601%	14	4.14048%
17	444	0.0141144%	4	6.40539%	21	1.0259%	14	3.44148%
17	446	0.350992%	4	6.56439%	22	0.428304%	14	3.44148%
18	451	0.16994%	4	3.79305%	23	3.19333%	15	4.14048%
18	455	0.0276849%	4	3.53416%	24	2.36%	15	2.85424%
18	457	0.356342%	4	2.84965%	24	0.613311%	15	2.09914%
19	460	0.24979%	4	0.592701%	25	3.41682%	16	4.14048%
20	466	0.0406012%	3	34.7246%	26	0.441222%	17	4.14048%
20	467	0.00617653%	3	34.3913%	27	3.58928%	17	2.85424%
20	469	0.326574%	3	34.4864%	27	2.96041%	17	2.85424%
21	473	0.188656%	3	32.7618%	27	0.00847131%	18	4.14048%
22	478	0.0193238%	3	30.6614%	29	2.73165%	19	4.14048%
22	480	0.33225%	3	30.7512%	29	2.15572%	19	4.14048%
23	486	0.130837%	3	28.7104%	31	1.24117%	20	4.14048%
24	489	0.0317946%	3	26.9594%	32	2.49307%	20	1.32605%
24	491	0.33772%	3	26.9594%	32	1.69245%	20	1.32605%
25	499	0.0758991%	3	25.028%	34	1.69504%	21	1.32605%
25	500	0.0437319%	3	25.194%	34	1.41635%	21	1.32605%
25	501	0.0117004%	3	25.3747%	34	1.07588%	21	1.32605%
25	503	0.310379%	3	25.0413%	35	3.5417%	22	4.18913%
26	504	0.277987%	3	22.774%	35	2.01334%	22	2.05151%
27	509	0.118177%	3	20.5848%	36	0.388643%	23	2.05151%
27	512	0.0237703%	3	20.8802%	37	2.95583%	23	1.37556%
27	514	0.316133%	3	20.9624%	37	2.44152%	23	1.37556%
28	519	0.159043%	3	18.757%	38	0.341898%	24	1.37556%
29	523	0.0353605%	3	16.3857%	39	1.78102%	25	2.05151%
29	524	0.00470877%	3	16.4708%	39	1.26297%	25	2.05151%
29	526	0.290449%	3	16.5635%	39	0.696987%	25	2.05151%
30	532	0.106788%	3	14.3686%	41	0.0902683%	26	2.05151%

Figure 14: New Configuration of the System (2)