Bulletin of Electrical Engineering and Informatics
Vol. 4, No. 3, September 2015, pp. 217~230
ISSN: 2089-3191

Heuristic Approach for Scheduling Dependent Real-Time Tasks

Medhat Hussein Ahmed Awadalla
Electrical and Computer Engineering Department, Sultan Qaboos University (SQU), Oman
Email: awadalla_medhat@yahoo.co.uk

Abstract

Reducing energy consumption is a critical issue in the design of battery-powered real time systems to prolong battery life. With dynamic voltage scaling (DVS) processors, energy consumption can be reduced efficiently by making appropriate decisions on the processor speed/voltage during the scheduling of real time tasks. Scheduling decision is usually based on parameters which are assumed to be crisp. However, in many circumstances the values of these parameters are vague. The vagueness of parameters suggests that to develop a fuzzy logic approach to reduce energy consumption by determining the appropriate supply-voltage/speed of the processor provided that timing constraints are guaranteed. Intensive simulated experiments and qualitative comparisons with the most related literature have been conducted in the context of dependent real-time tasks. Experimental results have shown that the proposed fuzzy scheduler saves more energy and creates feasible schedules for real time tasks. It also considers tasks priorities which cause higher system utilization and lower deadline miss time.

Keywords*: maximum 5 keywords from paper*

1. Introduction

Real-time systems are vital to industrialized infrastructure such as command and control, process control, flight control, space shuttle avionics, air traffic control systems and also mission critical computations [1]. In all cases, time has an essential role and having the right answer too late is as bad as not having it at all. In the literature, these systems have been defined as: “systems in which the correctness of the system depends not only on the logical results of computation, but also the time at which the results are produced” [1]. Such systems must react to the requests within a fixed amount of time which is called deadline. Scheduling algorithms of these systems may be considered one of the key components of a real-time system, which can either enable the system to thrive or bring it to its knees. Strict timing requirements must often be met within highly dynamic environments which do not lend themselves well to static scheduling algorithms. The level of uncertainty in dynamic, real-time environments is such as to require significant flexibility and adaptivity from real systems. Fuzzy logic contributions in this issue in the form of approximate reasoning, where it provides decision-support and expert systems with powerful reasoning capabilities bound by a minimum number of rules. Theoretically, fuzzy logic is a method for representing analog processes, or natural phenomena that are difficult to model mathematically on a digital computer. Therefore Fuzzy systems fit as scheduling algorithm building into the real-time system flexibility and adaptation to the uncertainty inherent in real-time environments and offer a means to improve several important characteristics of real-time systems.

Since most of real time systems (devices) are battery powered. As the applications on these devices are being complicated, the energy consumption is also effectively increasing. So, minimizing energy consumption is a critical issue in the design of these systems, and techniques that reduce energy consumption have been studied at different levels in details [2].

Dynamic voltage scaling (DVS) is a technique that varies the supply voltage and clock frequency (speed) based on the computation load to provide desired performance with the minimal amount of energy consumption in ubiquitous embedded systems.

The power consumption has two essential components: dynamic and static power. The dynamic power consumption, which is the main component, has a quadratic dependency on supply voltage [3] and can be represented as:

Received February 14, 2015; Revised May 12, 2015; Accepted June 6, 2015

$$P_{dynamic} = C_{ef} \cdot V_{dd}^2 \cdot F \qquad (1)$$

Where C_{ef} is the switched capacitance, V_{dd} is the supply voltage, and F is the processor clock frequency (sometimes referred as speed S) which can be expressed in terms of supply voltage V_{dd} and threshold voltage V_t as following:

$$F = k \cdot (V_{dd} - V_t)^2 / V_{dd} \qquad (2)$$

The static power consumption is primarily occurred due to leakage current (I_{leak}) [3], and the static (leakage) power (P_{leak}) can be expressed as:

$$P_{leak} = I_{leak} \cdot V_{dd} \qquad (3)$$

When the processor is idle, a major portion of the power consumption comes from the leakage. Currently leakage power is rapidly becoming the dominant source of power consumption in circuits and persists whether a computer is active or idle [2], and much work has been done to address this problem [3,4].
So, lowering supply voltage is one of the most effective ways to reduce both dynamic and leakage power consumption. As a result, it reduces energy consumption where the energy consumption is the power dissipated over time:

$$Energy = \int Power\ dt \qquad (4)$$

However, DVS aims at reducing energy consumption by reducing the supply-voltage/speed of the processor provided that timing constraints are guaranteed. In other words, DVS makes use of the fact that there is no benefit of finishing a real time job earlier than its deadline.

DVS processors have two types [4]: ideal and non-ideal. An ideal processor can operate at any speed in the range between its minimum available speed and maximum available speed. A non-ideal processor has only discrete speeds with negligible or non-negligible speed transition overheads. Another classification defines four different types of DVS systems: ideal, feasible, practical, and multiple [4].

In this paper, our motivation is to develop a fuzzy logic approach to reduce energy consumption by determining the appropriate supply-voltage/speed of the processor provided that timing constraints are guaranteed. Fuzzy logic approach is proposed because in a dynamic hard real-time system, not all the characteristics of tasks (e.g., precedence constraints, resource requirements, etc.) are known a priori. For example, the arrival time for the next task is unknown for aperiodic tasks. To be more precise, there is an inherit uncertainty in hard real-time environment which will worsen scheduling problems (e.g. arbitrary arrival time, uncertain computation time and deadline). Characteristics of a task that may be uncertain include expected next arrival time, criticality, or importance of the task, system load and/or predicted load of individual processors, and run time, or more specifically average vs. worst-case run time. Therefore, our goal is to develop an approach for hard real-time scheduling that can be applied to a dynamic environment involving a certain degree of uncertainty. In this paper, we concentrate on a hard real time system on a preemptable uniprocessor system with a set of dependent tasks. These tasks will be characterized by worst-case computation time, blocking time and task deadline.

The rest of the paper is organized as follows: section 2 outlines the related work to the theme of this paper. Section 3 demonstrates the multi-speed algorithm. Section 4 gives an overview about fuzzy inference system. Section 5 shows the proposed fuzzy system. Section 6 presents experiments and discussions. Section 7 concludes the paper.

2. Related Work

Many Researchers have tried to implement fuzzy logic to schedule the processes. There are four main approaches reported in the literature for the fuzzy scheduling problems; fuzzifying directly the classical dispatching rules, using fuzzy ranking, fuzzy dominance relation methods, and solving mathematical models to determine the optimal schedules by heuristic approximation methods [5]. Round robin scheduling using neuro fuzzy approach and Soft real-time fuzzy task scheduling for multiprocessor systems [6]. Fuzzy Better Job First (FBJF) scheduling algorithm logically integrates parameters and uses fuzzy ranking approach to determine the next most worthy job to be executed has been proposed [7]. A fuzzy scheduling approach to arrange real-time periodic and non-periodic tasks with reference to optimal utilization of distributed processors has been proposed [8]. In their paper, an attempt is made to apply fuzzy logic in the design and implementation of a modified scheduling algorithm to overcome the shortcoming of well-known scheduling algorithms. Furthermore, many dynamic and static scheduling algorithms [9-10] have been proposed and applied on uniprocessor systems. Also multiprocessor and distributed systems have been considered [11].

Regarding the energy efficient scheduling, Weiser et al. [12] are considered the pioneers in that field where they expected the DVS technique, then Yao et al. [13] have proposed an optimal static (offline) scheduling algorithm by considering a set of aperiodic jobs on an ideal processor. However, the problem of DVS with dependent tasks because of shared resources has been first addressed in [14]. Jejurikar and Gupta [15] have proposed two algorithms for scheduling fixed priority, Rate Monotonic (RM) scheduler, tasks using priority ceiling protocol (PCP) described in [16] as resource access protocol. They have computed static slowdown factors which guarantee that all tasks will meet their deadlines taking into account the blocking time caused by the task synchronization to access shared resources. In their first algorithm, critical section maximum speed (CSMS), they have let the critical sections (sections deal with shared resources) to be executed at maximum processor speed and they have computed slowdown factors for executing non critical sections. The second algorithm, constant static slowdown (CSS), computes a uniform slowdown factor for all tasks and for all sections (critical and non-critical) saving speed switches occurred in the first algorithm (CSMS).

The same authors [17] have then extended their previous algorithms (CSMS and CSS) to handle dynamic priority, Earliest Deadline First (EDF) scheduler, tasks using dynamic priority ceiling protocol (DPCP) shown in [18]. The dynamic priority ceiling protocol is an extension of original priority ceiling protocol to deal with dynamic priority tasks (EDF scheduling). Jejurikar and Gupta [19] have also proposed a generic algorithm that works with both EDF and RM schedulers, and they have introduced the concept of frequency inheritance in their algorithm. Zhang and Chanson [20] have worked on the same problem (scheduling of dependent tasks) and proposed three algorithms for energy efficient scheduling of dependent tasks with shared resources over EDF scheduler, where they have used stack resource policy (SRP) proposed by Baker [21] as resource access protocol. The SRP can handle static and dynamic priority tasks (EDF and RM schedulers), reduces context switches over PCPs, and is easy implemented. The first algorithm is the same as CSS for EDF scheduler proposed in [22] because they have derived the static slowdown factor directly from the EDF schedulability test with blocking time in [23]:

$$\forall i, 1 \le i \le n, \quad \frac{B_i}{D_i} + \sum_{k=1}^{i} \frac{C_k}{D_k} \le 1 \qquad (5)$$

where C is the computation time (worst case execution time WCET), D is the task relative deadline, n is the number of tasks, and B is the blocking time that can be defined as the maximum time through which a high priority task can be blocked by a low priority task due to exclusive access to shared resource (index i refers to the blocked high priority task). The second algorithm is the dual speed (DS) algorithm. The main concept of this algorithm is using two speeds (L, H) and switching between them. Initially the algorithm operates with the low speed L, and switches to the high speed H as soon as a blocking occurs. The last algorithm is the dual speed dynamic (online) reclaiming algorithm which dynamically collects the residue time from early completed jobs and redistributes it to the other pending jobs to further reduce the processor speed and achieve more energy saving. Then the same authors [24] developed their previous algorithms to achieve more energy saving and also to function with RM scheduler in addition to EDF scheduler. Baruah [25] has taken a closer look at EDF-scheduled systems in which access to shared resources is arbitrated by the SRP. (He has referred to such systems as EDF+SRP scheduled systems), where he has proved that under certain assumptions EDF+SRP is optimal, but he has not taken energy efficiency into account. Lee et al. [26] have developed the dual speed (DS) algorithm proposed by Zhang and Chanson [24] to use multiple speeds instead of two speeds to get their first multi-speed (MS) algorithm. Also they have proposed an enhanced multi-speed (EMS) algorithm that further reduces the energy dissipation by considering only remaining blocking time to compute a lower speed.

3. System Model

A. Task Model

In this paper, for simplicity, real-time periodic tasks are considered. Each task τ is characterized by the following parameters:

- The release time (r): the time when the task first released.
- The period (T): the constant interval between jobs.
- The relative deadline (D): the maximum acceptable delay for task processing.
- The computation time (C): the worst case execution time (WCET) of any job.
- The blocking time (B): the maximum time a task can be blocked by another lower priority task.

In this paper we consider well formed tasks that satisfy the condition: $0 \le C \le D \le T$.
A 3-tuple τ ={C, D, T} represents each task, the relative deadline is assumed to be the same as the period in all illustrative examples.

B. Processor Model

The tasks are scheduled on a single DVS processor that supports variable frequency (speed) and voltage levels continuously, i.e. DVS processors can operate at any speed/voltage in its range (ideal). Of course, practical DVS processors supports discrete speed/voltage levels (non ideal). So, the desired speed/voltage of the ideal DVS processor is rounded to the nearest higher speed/voltage level the practical DVS processor supports. The time (energy) required to change the processor speed is very small compared to that required to complete a task. It is assumed that the voltage change overhead, similar to the context switch overhead, is incorporated in the task computation time. In this paper, it is assumed that the processor's maximum speed is 1 and all other speeds are normalized with respect to the maximum speed.

4. Multi-Speed Algorithm

Multi-speed (MS) algorithm proposed by Lee et al. [25] is a blocking aware scheduling algorithm with non-preemptible critical sections using SRP as resource access protocol.

The MS algorithm can be considered as an extension of dual speed (DS) algorithm [24], where the difference between the two algorithms is that MS algorithm uses many speeds (low speed S_L and multiple high speeds S_m where $1 \le m \le n$) instead of two speeds (low speed L and one high speed H) in DS algorithm. Like DS algorithm, MS algorithm initially starts with the low speed S_L then it switches to one of high speeds as soon as a blocking occurs. The high speed to which the MS algorithm switches is determined according to the blocking task, i.e. each blocking task τ_m has its own high speed S_m, and according to the blocking task, the algorithm switches to the convenient high speed. The low speed S_L, which is exactly the same as low speed L in DS algorithm, is the optimal lowest speed with which all tasks can be scheduled without missing any deadline, and it is derived from the plain EDF schedulability test without shared resources:

$$\sum_{i=1}^{n} \frac{C_i}{D_i} \le S_L \tag{6}$$

The high speed S_m for a blocking task τ_m is derived as in DS algorithm from the EDF schedulability test with shared resources and SRP protocol:

$$\forall k, 1 \le k < m, \sum_{i=1}^{k} \left(\frac{C_i}{D_i} \right) + \frac{B_m}{D_k} \le S_m \tag{7}$$

Where the blocking time B_m here is the maximum time (length of critical section of τ_m) through which a low priority task τ_m can block another high priority task due to exclusive access to shared resource and unlike mentioned before, index m refers to the blocking task (low priority task).
The above mentioned speeds S_L and S_m have to satisfy the condition:

$$S_L \leq S_m \leq 1 \quad (8)$$

MS algorithm ends the high speed interval when the deadline of the blocking task is reached or the processor becomes idle. In some real time tasks [23], MS improves the energy consumption however in others tasks, the timing constraints are not guaranteed.

5. Fuzzy Inference Systems

A fuzzy inference system (FIS) tries to derive answers from a knowledgebase by using a fuzzy inference engine. The inference engine which is considered to be the brain of the expert systems provides the methodologies for reasoning around the information in the knowledgebase and formulating the results. Fuzzy logic is an extension of Boolean logic dealing with the concept of partial truth that denotes the extent to which a proposition is true. Whereas classical logic holds that everything can be expressed in binary terms (0 or 1, black or white, yes or no), fuzzy logic replaces Boolean truth values with the degree of truth. Degree of truth is often employed to capture the imprecise modes of reasoning that play an essential role in the human ability to make decisions in an environment of uncertainty and imprecision. The membership function of a fuzzy set corresponds to the indicator function of the classical sets. It can be expressed in the form of a curve that defines how each point in the input space is mapped to a membership value or a degree of truth between 0 and 1. The most common shape of a membership function is triangular, although trapezoidal and bell curves are also used. The input space is sometimes referred to as the universe of discourse [6]. Fuzzy Inference Systems are conceptually very simple. An FIS consists of an input stage, a processing stage, and an output stage. The input stage maps the inputs, such as deadline, execution time, and so on, to the appropriate membership functions and truth values. The processing stage invokes each appropriate rule and generates a result for each. It then combines the results of the rules. Finally, the output stage converts the combined result back into a specific output value [6]. As discussed earlier, the processing stage, which is called the inference engine, is based on a collection of logic rules in the form of IF-THEN statements, where the IF part is called the "antecedent" and the THEN part is called the "consequent".

Typical fuzzy inference subsystems have dozens of rules. These rules are stored in a knowledgebase. An example of fuzzy IF THEN rules is: IF *deadline* is *early* then *priority* is *high*, in which *deadline* and *priority* are linguistics variables and *priority* and *high* are linguistics terms. The five steps toward a fuzzy inference are as follows:

- fuzzifying inputs
- applying fuzzy operators
- applying implication methods
- aggregating outputs
- defuzzifying results

Below is a quick review of these steps. However, a detailed study is not in the scope of this paper. Fuzzifying the inputs is the act of determining the degree to which they belong to each of the appropriate fuzzy sets via membership functions. Once the inputs have been fuzzified, the degree to which each part of the antecedent has been satisfied for each rule is known. If the antecedent of a given rule has more than one part, the fuzzy operator is applied to obtain one value that represents the result of the antecedent for that rule. The implication function then modifies that output fuzzy set to the degree specified by the antecedent. Since decisions are based on the testing of all of the rules in the Fuzzy Inference Subsystem (FIS), the results from each rule must be combined in order to make the final decision. Aggregation is the process by which the fuzzy sets that represent the outputs of each rule are processes into a single fuzzy set. The input for the defuzzification process is the aggregated output fuzzy set and the output is then a single crisp value [6]. This procedure can be summarized as follows: mapping input characteristics to input membership functions, input membership function to rules, rules to a set of output characteristics, output characteristics to output membership functions, and the output membership function to a single crisp valued output. There are two common inference methods [6]. The first one is called Mamdani's fuzzy inference method proposed by Ebrahim Mamdani [8] and the second one is Takagi-Sugeno-Kang, or simply Sugeno, method of fuzzy inference introduced in [9]. These two methods are the same in many respects, such as the procedure of fuzzifying the inputs and fuzzy operators. The main difference between Mamdani and Sugeno is that the Sugeno's output membership functions are either linear or constant but Mamdani's inference expects the output membership functions to be fuzzy sets.

6. The Proposed Model

In the proposed model, the input stage consists of four input variables i.e. worst execution time, deadline, blocking time and arriving time as shown in FIG. 1. Worst execution time is the actual amount of time a task requires on CPU to get executed, blocking time is how much time a task can wait before getting a chance to get executed, Deadline represents the final time limit before what a task has to get terminated whereas the arriving time is the time at which the job is ready to be assigned to the processor. The combination of four input parameters decides the job priority and the appropriate processor speed to execute it.

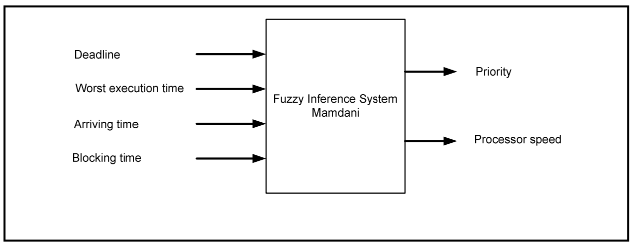


Figure 1. Inference system block diagram

Membership functions describe the degree to which each input parameter represents its association. Linguistic variables are assigned to each input parameter, to represent this association. Worst execution time is categorizes as Low, Medium and High. Similarly blocking time is defined in the same way. However, Deadline and Arriving time are defined as early, medium and late. The output parameters, job priority and processor speed are defined as low, medium and high as depicted in Figure 2. Table 1 depicts the values used for constructing these various fuzzy membership functions.

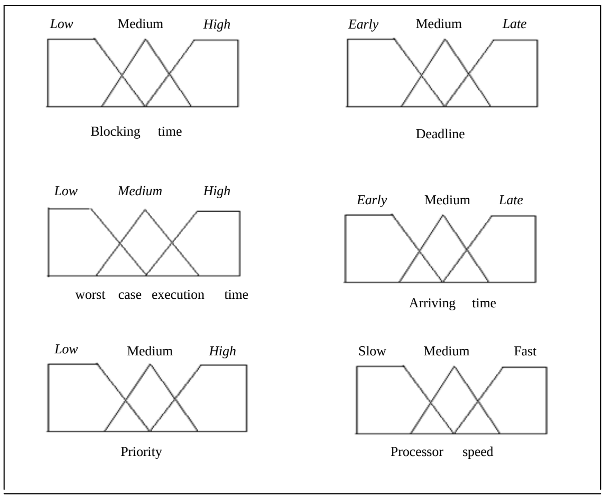


Figure 2. Membership functions of the system parameters

Table 1. Values used for constructing various fuzzy membership Functions.

Variables	Early	Medium	Late
Deadline	1	2.5	5
	2	3	8
	3	6	10

(Parameters for deadline)

Variables	Low	Medium	High
Worst case execution time	1	2.5	5
	2	3	8
	3	6	10

(Parameters for worst case execution time)

Variables	Early	Medium	Late
Arriving time	0.0	0.6	2.0
	0.6	2.0	3.4
	2.0	3.4	4.0

(Parameters for arriving time)

Variables	Low	Medium	High
Blocking time	0.0	0.8	2.0
	0.8	2.0	3.2
	2.0	3.2	4.0

(Parameters for blocking time)

Variables	Low	Medium	High
Priority	1	3.5	6.5
	2.5	5.5	8.5
	4	7	10

(Parameters for priority)

Variables	Slow	Medium	Fast
Processor speed	0.1	0.35	0.65
	0.25	0.55	0.8
	0.4	0.7	1

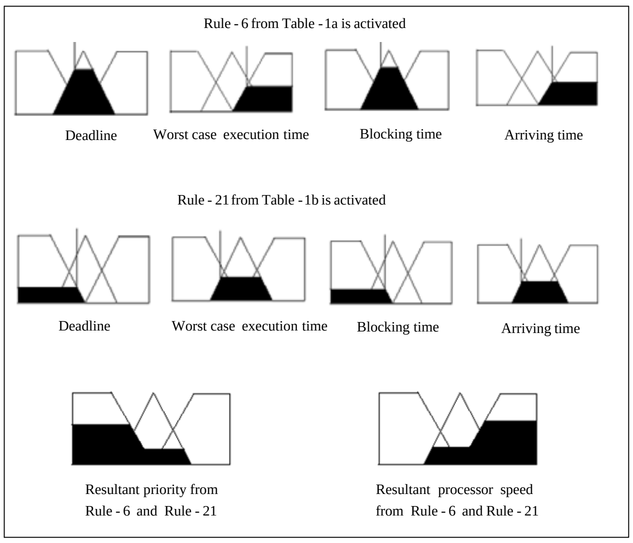
(Parameters for processor speed)

Fuzzy Rule No.	**Deadline**	**Worst case execution time**	**Arriving time**	**Blocking time**	**priority**	**Processor speed**
1	Early	Low	Early	Low	High	Slow
2	Early	Low	Early	Medium	High	Medium
3	Early	Medium	Medium	Low	High	Medium
4	Medium	Medium	Medium	Low	Medium	Medium
5	Medium	Medium	Medium	High	Medium	Fast
6	Medium	High	Medium	High	Medium	Fast
7	Late	High	Late	High	Low	Fast
8	Late	High	Late	Medium	Low	Medium
9	Late	Low	Late	Medium	Low	Medium
10	Early	Low	Late	Medium	High	Medium
11	Early	High	Medium	Low	High	Medium
12	Early	High	Medium	High	High	Fast
13	Medium	Low	Late	High	Medium	Fast
14	Late	Low	Late	High	Low	Fast
15	Medium	Low	Early	Medium	Medium	Medium
16	Medium	Medium	Medium	Medium	Medium	Medium
17	Early	High	Late	Low	High	Low
18	Medium	High	High	Low	Medium	Low
19	Late	High	Early	High	Low	Fast
20	Late	Low	Late	Low	Low	Low
21	Late	Medium	Late	Medium	Low	Medium

Fuzzy rules try to combine these parameters as they are connected in real worlds. Some of these rules are mentioned in table 1:

For instance, rule no-6 and rule no-21 (Table-2) are activated for the control of job priority and processor speed. The resultant priority and processor speed are also given in Figure-3, from which the crisp output values can be determined. In fuzzy inference systems, the number of rules has a direct effect on its time complexity. Therefore, having fewer rules may result in a better system performance.

In our proposed approach, a newly arrived task will be added to the input of job queue. This queue has the remaining tasks from last cycle that has not yet been assigned. The following algorithm will be executed:


Rule - 6 from Table - 1a is activated
Deadline
Worst case execution time
Blocking time
Arriving time
Rule - 21 from Table - 1b is activated
Deadline
Worst case execution time
Blocking time
Arriving time
Resultant priority from Rule - 6 and Rule - 21
Resultant processor speed from Rule - 6 and Rule - 21

Loop
1. For each ready task, feed the deadline, execution time, blocking time and arriving time into the inference engine. Consider the output of inference module as priority of the task and the processor speed.
2. Execute the task with highest priority with the decided speed unless it is blocked by lower priority job until a scheduling event occurs (a running task finishes, a new task arrives).
3. Update the system states.

End Loop

We chose to treat deadline time as the most important principles behind choosing a task for scheduling because the major purpose of hard real-time scheduling is to meet the deadline. After this, worst execution time and then earliest arriving time. However if the lowest priority job is only available job, it will be assigned to the processor till another higher priority job arrives. Since the paper has another objective which is to reduce the power consumption, after deciding which job will be assigned to the processor, the system will decide at which speed the processor should perform. The processor speed is mainly affected by speed of the blocking time of the lower priority tasks.

7. Experiments and Discussion

To illustrate the fuzzy approach and the contribution of this paper, examples implemented in [23] are repeated for the sake of qualitative comparison and other tasks have been performed to address the generalization of applying fuzzy logic as a scheduler approach and its capability to minimize the energy consumption of powered real time systems. The first hard real time system with three tasks is considered as following:

$$\tau_1=\{1 ,4 ,4 \}, \tau_2=\{1.5 ,12 ,12 \}, \tau_3=\{3 ,24 ,24 \}$$

The arrival times and critical sections of the three tasks within the least common multiple (LCM) of periods are shown in figure 1(a).

According to the developed inference system, τ_1 has the highest priority, followed by τ_2 and lastly τ_3. The resultant low speed S_L is equal 0.5, which is the same as calculated based on equation 6 that represents the processor utilization factor $U=\sum C/T$ [26]. There are two blocking tasks in this example: τ_2 that can block higher priority task τ_1 for maximum time $B_2=1.5$ and τ_3 that can block higher priority tasks τ_1 and τ_2 for maximum time $B_3=3$. So, there will be two high speeds (S_2, S_3) according to these two blocking tasks, and these two speeds are $S_2 = 0.6$, and $S_3 = 0.92$. The two speeds (S2, S3) also satisfy the condition (7), where $0.5 \le S_2 \le 1$ and $0.5 \le S_3 \le 1$.

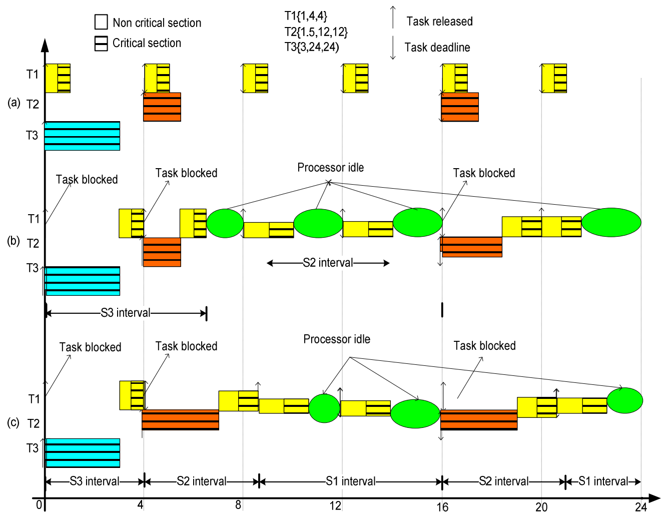


Figure 1. (a) Task set description: arrival times, computation times, and critical sections. (b) MS algorithm. (c) Fuzzy logic.

The rectangles represent the processing of tasks (jobs) by CPU where the vertical dimension represents the processor speed, and the horizontal dimension represents the execution time elapsed for processing tasks according to their WCETs and the processor speed. It is clearly noted that the area of the rectangles of the jobs of the same task is the same due to that a task always takes the same number of execution cycles which equals to processor speed multiplied by elapsed time.

Back to Figure 1(a), it is assumed that τ_3 is released before τ_1 with enough time (ε) to lock the shred resource. When task τ_1 is released, it will be blocked by the lower priority task τ_3 due to exclusive access to shared resource (according to SRP, a task may be blocked when it is released, and as soon as it starts, it can not be blocked). So, the processor will start executing τ_3 with high speed S_3. At time t=4, when the second job of task τ_1 is released, it is also blocked by the lower priority task τ_2 released before it with enough time (ε) to lock the shred resource.

MS algorithm which operates with high speed S_3 switches to the maximum of the two high speeds (S_3,S_2) which is S_3, and MS algorithm ends this high speed interval and switches to the low speed S_L at time t=6.5 when the processor becomes idle.

At time t=16, when the fifth job of task τ_1 is released, it will be blocked by the second job of task τ_2, and MS algorithm switches to the high speed S_2 and ends this high speed interval when the processor becomes idle.

The same scenario is happened in the case of the proposed fuzzy logic approach, as shown in Figure1(c), it starts with high speed S_3, however it switches to S_2 as long as τ_2 showed up. Fuzzy logic ends this high speed interval S_2 and switches to the low speed S_L at time t=9 when τ_1 is released. The processor idle time is reduced from 33% in MS to 23% in fuzzy logic approach which reflects the improvements achieved in the system performance in addition to there is no deadline miss, furthermore the interval of S_3 is reduced which in turn reduce the power consumption.

Another hard real time system with three tasks is addressed:

$$\tau_1 = \{2, 5, 5\},\ \tau_2 = \{2.5, 10, 10\},\ \tau_3 = \{4, 40, 40\}$$

In this example, τ_1 has the highest priority, τ_2 is middle and then $\tau3$ is the lowest one. The resultant low speed S_L is equal 0.7, which is less than the low speed calculated based on equation 6. There are two blocking tasks in this example: τ_2 that can block higher priority task τ_1 for maximum time B_2=2, and τ_3 that can block higher priority tasks τ_1 and τ_2 for maximum time B_3=3. So, there will be two high speeds (S_2, S_3). S_2= 0.8, and S3=1. The two speeds (S_2, S_3) also satisfy the condition (7), where $0.7 \le S_2 \le 1$ and $0.7 \le S_3 \le 1$.

Figure 2(b) shows MS algorithm which ends the high speed interval when processor becomes idle (at times t=10.75, t=19.75, t=29.75, and t=39.75) or the deadline of blocking task is reached, while Figure 2(c) shows the fuzzy logic approach which ends the high speed interval when the blocked task deadline is reached (at time t=6), the processor becomes idle, or a lower or equal priority is selected to run (at times t=10, t=19.125, t=29.125, and t=39.125).

Even though the high speeds and low speed are close to each other in this example, there is again an improvement in the system performance based on fuzzy logic approach compared with MS where the processor idle time is reduced from 10% to 5.8 %. Figure 7 depicts the average consumed power in the proposed approach and most related work, MS and IMS [ ].

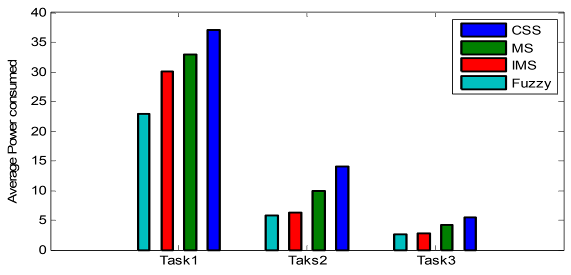
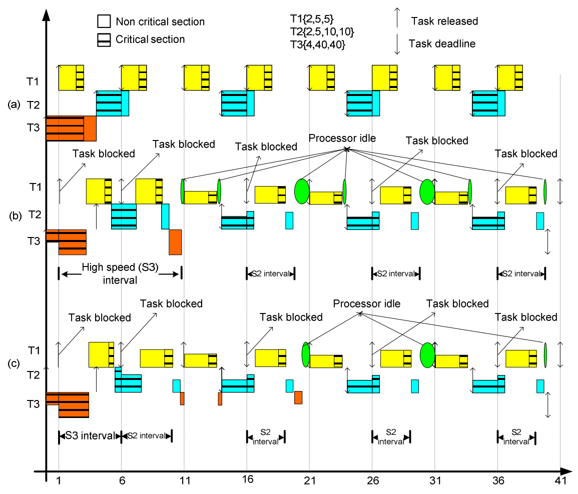


Figure 8. Power consumption in fuzzy, IMS, MS and CSS in the illustrated examples

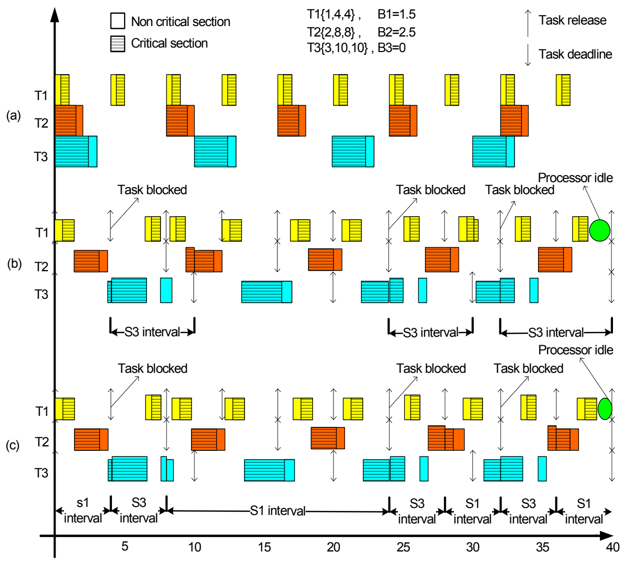


Again another more example, hard real time system with the following three tasks is implemented:

$$\tau_1=\{1, 4, 4\},\ \tau_2=\{2, 8, 8\},\ \tau_3=\{3, 10, 10\}$$

Again the arrival times and critical sections of the three tasks within the least common multiple (LCM) of periods are shown in Figure 3(a). Again τ_1 has the highest priority, τ_2 is middle and then τ_3 is the lowest one. The resultant low speed S_L is equal 0.8, and s2= 0.825 and s3=0.875

There is again an improvement in the system performance based on fuzzy logic compared with MS algorithm where the processor idle time is reduced from 4.28% to 2.6%.



Referring to Figures 2 and 3, reducing the time during which the processor is idle comes from lowering the processor speed for longer time intervals. This, in turn, reduces the energy consumption dramatically due to quadratic dependency between power and processor speed. To verify that, a comparison study has been performed by computing the energy consumed in CSS, DSA, and EDSA using the simplified power model $P=S^2$ used in (Jejurikar and Gupta 2002), where the blocking time B changes from 0 to the highest amount at which the task set is schedulable (when H=1).

Of course, the high speed H changes from H=L (when B=0) to H=1, while the low speed L does not change.

As it is clear from Figure 7, EDSA is the most energy efficient algorithm especially with high blocking times, where the difference between the low and high speeds (L, H) increases significantly.

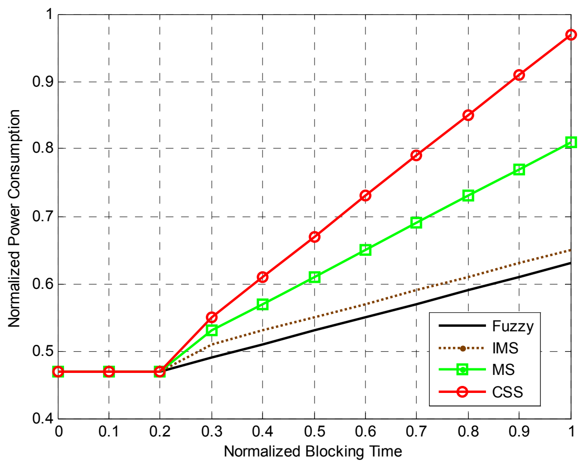


Figure 7. Power Consumption versus Blocking Time Changes in Example1

The comparison is repeated for the second example, it is noticed that, as shown in Figure 8, EDSA exhibits a slight improvement over DSA with the highest blocking time due to the small difference between high and low speeds (H, L).

As a result, when the blocking time is low (the high speed is almost the same as the low speed), the three algorithms exhibit the same performance. When the blocking time increases (the difference between the high and low speeds also increases), EDSA behaves better than the other two algorithms (CSS and DSA) especially when this difference is significant.

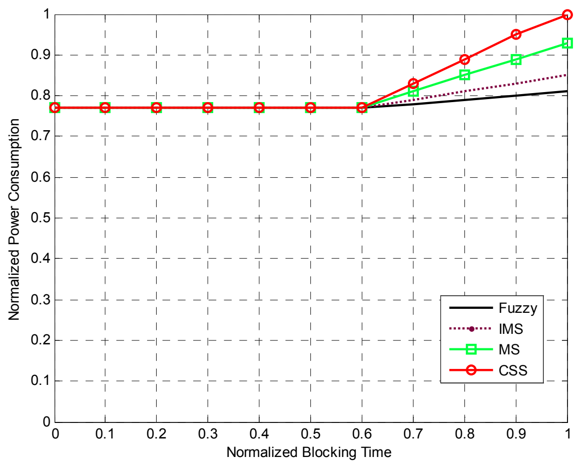


Figure 8. Power Consumption versus Blocking Time Changes in Example2

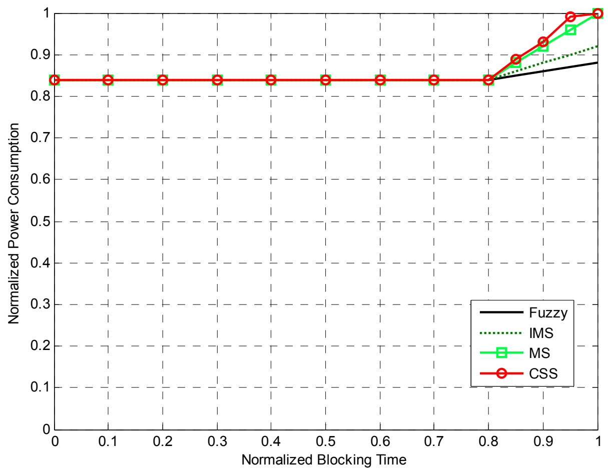


Figure 9. Power Consumption versus Blocking Time Changes in Example3

8. Conclusion

The paper has addressed the problem of real time scheduling of dependent tasks due to exclusive access shared resources taking into account the reducing of energy consumption as a main goal. The paper has proposed fuzzy logic approach to perform multi-speed (MS) scheduling algorithm, where the proposed algorithm has shown more energy saving than traditional MS algorithm and other related approaches.

References

[1] HS Behera, Ratikanta Pattanayak, Priyabrata Mallick. An Improved Fuzzy-Based CPU Scheduling (IFCS) Algorithm for Real Time Systems. *International Journal of Soft Computing and Engineering (IJSCE)* ISSN: 2231-2307. 2012; 2(1).
[2] Z Khani, R Singh and J Alam. Tasks Allocation using Fuzzy Inference in Parallel and Distributed System. *Journal of Information and Operations Management*. ISSN: 0976-7754 & E-ISSN: 0976-7762. 2012; 3(2).
[3] J Chen and C Kuo. “Energy-efficient scheduling for real-time systems on dynamic voltage scaling (DVS) platforms”. *13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2007)*. 2007: 28-38.
[4] V Venkatachalam and M Franz. “Power reduction techniques for microprocessor systems”. *ACM Computing Surveys (CSUR)*. 2005; 37(3): 195-237.
[5] W Liu. “Techniques for leakage power reduction in nanoscale circuits: A survey”. IMM-Technical Report-2007-04, Informatics and Mathematical Modelling, Technical University of Denmark, DTU 2007.
[6] JA Butts and GS Sohi. “*A static power model for architects*”. In Proceedings of the 33rd annual ACM/IEEE international symposium on Microarchitecture, Monterey, CA, USA. 2000: 191-201.
[7] G Qu. “What is the limit of energy saving by dynamic voltage scaling? *IEEE/ACM International Conference on Computer Aided Design, (ICCAD 2001)*. 2001: 560–563.
[8] http://www.intel.com.
[9] http://www.transmeta.com.
[10] M Weiser, B Welch, A Demers and S Shenker. “*Scheduling for reduced cpu energy*”. In Proc. of Symposium on Operating System Design and Implementation (OSDI). 1994: 13–23.
[11] F Yao, A Demers and S Shenker. “*A scheduling model for reduced cpu energy*”. In Proceedings of IEEE Annual Symposium on Foundations of Computer Science. 1995: 374–382.
[12] X Hu, and J Quan. “Fundamentals of power-aware scheduling”. *J. Henkel and S. Parameswaran (eds.), Designing Embedded Processors – A Low Power Perspective*. 2007: 219–229.
[13] X Hu, and J Quan. “Static DVFS scheduling.” *J. Henkel and S. Parameswaran (eds.), Designing Embedded Proc.of A Low Power Perspective*. 2007: 231–242.
[14] P Pillai and KG Shin. “Dynamic DVFS scheduling.” *J. Henkel and S. Parameswaran (eds.), Designing Embedded Processors – A Low Power Perspective*. 2007: 243–258.

[15] D Shin and J Kim. "Dynamic voltage scaling of periodic and aperiodic tasks in priority-driven systems". *ASPDAC*. 2004: 635– 658.

[16] A Andrei, P Eles, Z Peng, M Schmitz, and BM Al-Hashimi. "*Voltage Selection for Time-Constrained Multiprocessor Systems*". In Proc. Of Designing Embedded Processors – a Low Power Perspective. 2007: 259–284.

[17] R Mishra, N Rastogi, D Zhu, D Moss´e, and R Melhem. "Energy aware scheduling for distributed real-time systems". *In International Parallel and Distributed Processing Symposium*. 2003: 21.

[18] R Jejurikar and R Gupta. "*Energy aware task scheduling with task synchronization for embedded real time systems*". In Proc. of International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES). 2002: 164–169.

[19] F Zhang and ST Chanson. "*Processor voltage scheduling for real-time tasks with non-preemptible sections*". In Proc. of the 23rd IEEE Real-Time Systems Symposium (RTSS'02). 2002: 235–245.

[20] L Sha, R Rajkumar, and JP Lehoczky. "Priority inheritance protocols: An approach to real-time synchronization". *IEEE Transactions on Computers*. 1990; 39(9): 1175–1185.

[21] R Jejurikar and R Gupta. "Energy aware edf scheduling with task synchronization for embedded real time operating systems". *in Workshop on Compilers and Operating System for Low Power*. 2002.

[22] M Chen and K Lin. "Dynamic priority ceilings: A concurrency control protocol for real-time systems," *Real Time Systems Journal.* 1990; 2(1): 325–346.

[23] R Jejurikar and R Gupta. "Energy aware task scheduling with task synchronization for embedded real time systems"*. IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*. 2006; 25(6): 1024 – 1037.

[24] TP Baker. "Stack-based scheduling of real time processes". *Journal of Real-Time Systems*. 1991; 3(1): 67– 99.

[25] F Zhang and ST Chanson. "Blocking-aware processor voltage scheduling for real-time tasks". *ACM Transactions in Embedded Computing Systems*. 2004: 307–335.

[26] SK Baruah. "Resource sharing in EDF-scheduled systems: a closer look". *27th IEEE International Real-Time Systems Symposium, (RTSS'06)*. 2006: 379–387.

[27] J Lee, K Koh, and C Lee. "Multi-speed DVS algorithms for periodic tasks with non-preemptible sections". *13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2007).* 2007: 459-468.

[28] F Cottet, J Delacroix, C Kaiser and Z Mammeri. *Scheduling in Real-Time Systems*, John Wiley & Sons Ltd, England. 2002.

[29] SJ Kadhim and KM Al-Aubidy. Design and Evaluation of a Fuzzy-Based CPU Scheduling Algorithm. VV Das et al. (Eds.): BAIP 2010, CCIS 70. Springer-Verlag Berlin Heidelberg. 2010: 45 – 52.

[30] B Alam, MN Doja, R Biswas, M Alam. Fuzzy Priority CPU Scheduling Algorithm. *IJCSI International Journal of Computer Science Issues, ISSN (Online)*: 1694-0814. 2011; 8(1).

[31] D Pandey, Vandana and MK Sharma. Fuzzy Better Job First Scheduling Algorithm. International Journal of Computer Applications (0975 – 8887). 2011; 33(9).

[32] M Hamzeh, SM Fakhraie, and C Lucas. Soft Real-Time Fuzzy Task Scheduling for Multiprocessor Systems. *International Journal of Intelligent Technology*. ISSN 1305-6417. 2007; 2(4).

[33] S Gulati, N Arora and K Deep. *International Journal of Research in Engineering & Applied Sciences*. ISSN: 2249-3905. 2012; 2(2): 1741-1747.