Energy-Efficient Multi-Speed Algorithm for Scheduling Dependent Real-Time Tasks

A. M. Elewi, M. H. A. Awadalla, and M. I. Eladawy
Communication, Electronics, and Computer Engineering Department
Faculty of Engineering, Helwan University
11421, Cairo, Egypt

Abstract- **Reducing energy consumption is a critical issue in the design of battery-powered embedded 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. This paper addresses the problem of energy efficient real-time task scheduling over earliest deadline first (EDF) scheduling policy where the tasks are dependent due to shared resources. Furthermore, the paper proposes enhancements over the existing multi-speed (MS) algorithm where the proposed algorithm achieves more energy saving and has the capability to function with both stack resource policy (SRP) and dynamic priority ceiling protocol (DPCP) as resource access protocols.**

I. INTRODUCTION

Currently, embedded systems are involved in most details of our life such as cell phones, pocket PCs, multimedia devices, PDAs (personal digital assistants), digital cameras, and medical implants, which all 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 embedded systems, and techniques that reduce energy consumption have been studied at different levels in details [1].

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 [1] and can be represented as:

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

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 currents (I_{leak}) [2], 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 [1]: *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* [5].

Some well-known examples of DVS processors are Intel StrongARM SA1100 processor which supports 12 voltage/speed levels, the Intel PXA250 XScale which supports 4 voltage/speed levels [6], the Transmeta TM5400 processor which supports 6 voltage/speed levels [7], and the 1.6 GHz Intel Pentium M processor which supports 6 voltage/speed levels [6].

II. RELATED WORK

During the last decade much work has been done in the field of energy efficient scheduling, but Weiser et al. [8] are considered the pioneers in that field where they expected the DVS technique, then Yao et al. [9] have proposed an optimal static (offline) scheduling algorithm by considering a set of aperiodic jobs on an ideal processor. Furthermore, many dynamic and static scheduling algorithms [10,11,12,13] have been proposed and applied on uniprocessor systems. Also multiprocessor and distributed systems have been considered [14,15]. However, the problem of DVS with dependent tasks because of shared resources has been first addressed in [16,17].

Jejurikar and Gupta [16] have proposed two algorithms for scheduling fixed priority, rate monotonic (RM) scheduler, tasks using priority ceiling protocol (PCP) described in [18] 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 [19] 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 [20]. The dynamic priority ceiling protocol is an extension of original priority ceiling protocol to deal with dynamic priority tasks (EDF scheduling).

Jejurikar and Gupta [21] 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 [17] 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[22] 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 [19] because they all have derived the static slowdown factor directly from the EDF schedulability test with blocking time for scheduling *n* tasks ordered by their increasing relative deadlines [22]:

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

Where C is the computation time (worst case execution time WCET), D is the task relative deadline, 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 [23] developed their previous algorithms to achieve more energy saving and also to function with RM scheduler in addition to EDF scheduler.

Baruah [24] 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. [25] have developed the dual speed (DS) algorithm proposed by Zhang and Chanson [17] to use multiple speeds instead of two speeds to get their first multi-speed (MS) algorithm which is the interest of this paper. 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.

III. System Model

A. Task Model

In this paper, real-time periodic tasks are considered. A periodic task is a sequence of jobs released at constant intervals (called the period). Each task τ is characterized by the following parameters [26]:

- 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 $\tau = \{C, D, T\}$ 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 speed/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.

IV. 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 proposed by Zhang and Chanson [17], 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 blocking time due to shared resources:

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

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 following condition:

$$S_L \le S_m \le 1 \tag{8}$$

MS algorithm ends the high speed interval when the deadline of the blocking task is reached or the processor becomes idle.

A. *An Illustrative Example*

An example is implemented to illustrate the MS algorithm and the contribution of this paper. A hard real time system with three tasks is considered as following:

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

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 (6) the low speed is $S_L=(1/4+1.5/12+3/24)=0.5$ which 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 computed according to (7), where $S_2=(1/4+1.5/4)=0.625$, and $S_3=\max(1/4+3/4, 1/4+1.5/12+3/12)=1$. The two speeds (S2,S3) also satisfy the condition (7), where $0.5 \le S_2 \le 1$ and $0.5 \le S_3 \le 1$.

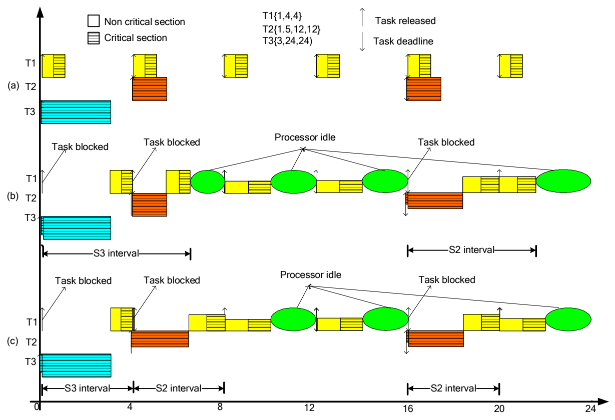

Figure 1. (a) Task set description: arrival times, computation times, and critical sections. (b) MS algorithm. (c) IMS algorithm.

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(b), 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, MS algorithm switches to the high speed S_3 because the blocking task is τ_3. At time t=4, when the second job of task τ_1, 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.

B. *Improved Multi-Speed Algorithm*

MS algorithm ends the high speed interval when the deadline of the blocking task is reached or the processor becomes idle causing more energy consumption. This highlights the key idea of the first contribution of the improved MS (IMS) algorithm proposed in this paper to end the high speed interval by the blocked (high priority) task deadline which is mostly earlier than the blocking(low priority) task deadline. Also the high speed interval has to be ended when the processor becomes idle or a lower or equal priority task (job) is selected to execute as in [23].

To test the performance of IMS algorithm, the previous example has been performed using IMS algorithm as shown in Figure1(c). The processor idle time is reduced from 33% in MS to 25% in IMS which reflects the improvements achieved in the system performance.

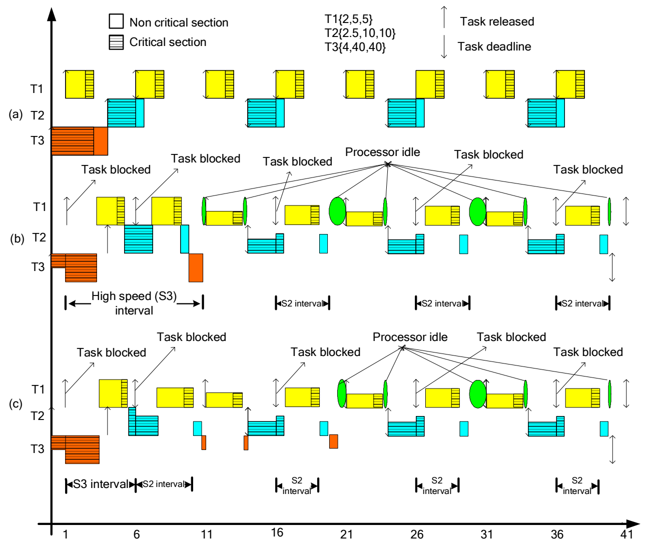

Figure 2. (a) Task set description: arrival times, computation times, and critical sections. (b) MS algorithm. (c) IMS algorithm.

To validate the effect of IMS, another hard real time system with three tasks is addressed:

τ_1= {2, 5, 5}, τ_2= {2.5, 10, 10}, and τ_3= {4, 40, 40}.

The arrival times and critical sections of the three tasks within the least common multiple (LCM) of periods are shown in Figure 2(a).

According to (6) the low speed is S_L=(2/5+2.5/10+4/40)=0.75 which represents the processor utilization factor U.

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) according to these two blocking tasks, and these two speeds are computed according to (7), where S_2=(2/5+2/5)=0.8, and S_3=max(2/5+3/5,2/5+2.5/10+3/10)=1. The two speeds (S2,S3) also satisfy the condition (7), where $0.75 \le S_2 \le 1$ and $0.75 \le S_3 \le 1$.

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 IMS algorithm 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 IMS compared with MS where the processor idle time is reduced from 10% to 6.5%.

C. MS and Dynamic Priority Ceiling Protocol

The success of MS algorithm requires that the critical sections be non-preemptible, and MS algorithm may cause deadline miss when it is used with dynamic priority ceiling protocol (DPCP) because a high priority task can preempt a critical section of another low priority task if the high priority task does not need to use the shared resource.

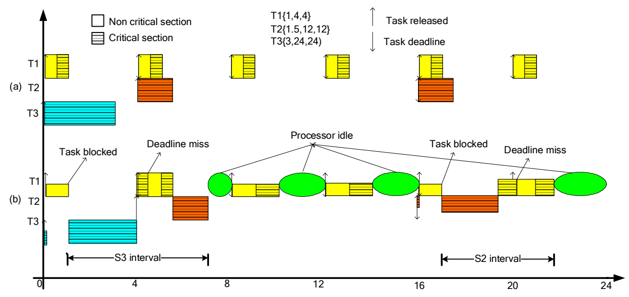

Figure 3. (a) Task set description. (b) Deadline miss when MS algorithm is used with DPCP.

However, the blocking occurs at the moment when the high priority task needs to access the resource, and the switching to one of high speeds S_m is delayed until this moment causing deadline miss as shown in Figure 3.

Figure 3 shows the same example in Figure 1, but it uses DPCP instead of SRP as resource access protocol. Task τ_3 that began to execute its critical section is preempted by the released high priority task τ_1 and task τ_1 executes its non-critical section. Task τ_1 is just blocked when it needs to access the shared resource, i.e. at time t=1, where the high speed interval S_3 starts, and τ_2 resume executing until it ends its critical section to be preempted again by τ_1. Then, task τ_1 resume executing, but it can not meet the deadline, where the first job (j_1) misses its deadline because of delaying the start of the high speed interval (blocking instant) due to using DPCP as resource access protocol as shown in Figure 3(b).

It is clear that to avoid the occurrence of deadline miss, the switching to the high speed should occur earlier as in using SRP as resource access protocol, and this is the second contribution of IMS algorithm. To achieve that, IMS algorithm proposes that the switching to a high speed should occur when a critical section is preempted or a blocking takes place.

Figure 4 shows the adaptation of IMS algorithm with DPCP as resource access protocol to avoid deadline misses and achieve energy saving as done with SRP, i.e., the switching to a high speed occurs not only when the high priority task is blocked, but also when a critical section is preempted. Unlike SRP, DPCP increases the context switches as it is obvious in Figures 1(c) and 4, and this is the key advantage of SRP over PCPs as mentioned earlier.

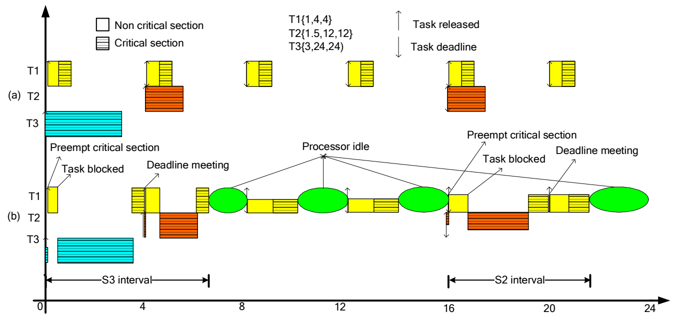

Figure 4. (a) Task set description. (b) Deadline meeting when IMS algorithm is used with DPCP.

```
// END_H is the end of $S_m$ interval
// If the system isn't in a high speed $S_m$ interval, End_H = -1
// $J_m$ is the current job.
// $S_{curr}$ is the current processor speed.
Initialization:
    END_H = -1
    Calculate $S_L$                          // using (6)
    Set Speed ( $S_L$ )
When a task joins or leaves the system:
    Calculate $S_L$
    for m=1 to number of tasks
        Calculate $S_m$                      // using (7) and (8)
When job $J_{i,k}$ arrives:
    if Priority( $J_{i,k}$ ) > Priority( $J_m$ )
      if Preempt_ $J_m$ ( ) is successful
        if Preempt_Critical_Section( ) is successful
          if $S_m > S_{curr}$
            Set Speed( $S_m$ )
            END_H=max(END_H, deadline of $J_{i,k}$)
          end if
        end if
        Execute $J_{i,k}$
      else if $S_m > S_{curr}$               // $J_{i,k}$ is blocked
        Set Speed( $S_m$ )
        END_H=max(END_H, deadline of $J_{i,k}$)
      end if
    end if
When END_H is reached, the processor is idle, or a lower (equal)
priority task is selected to run:
    END_H = -1
    Set Speed ( $S_L$ )
```

Figure 5. Improved multi-speed (IMS) algorithm

Figure 5 shows the proposed algorithm IMS which achieves more energy saving and works with SRP and DPCP as resource access protocols.

As shown in Figure 5, IMS algorithm recalculates the low speed S_L and the high speeds S_m (one high speed for each blocking task) when a task joins or leave the system. The algorithm initially starts with the low speed S_L and switches to one of high speeds according to the blocking (running) task if a high priority task is blocked or a critical section is preempted. Then, it ends the high speed interval if the deadline of blocked (high priority) task is reached, processor is idle, or task, whose priority is equal or lower than the priority of the blocking task, is selected to run.

V. Results and Discussion

Referring to Figures 1 and 2, 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, MS, and IMS using the simplified power model $P=S^2$ used in [16], where the blocking time B changes from 0 to the highest amount at which the task set is schedulable (when a high speed S_m=1). Of course, the high speeds S_m changes from S_m=S_L (when B=0) to S_m=1, while the low speed S_L does not change.

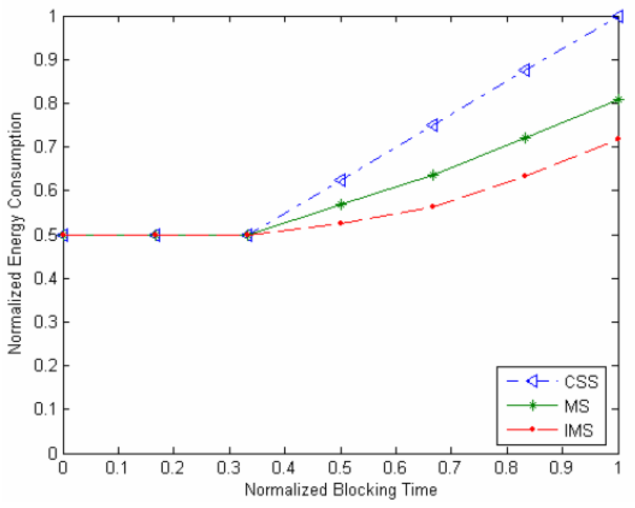

Figure 6. Energy consumption versus blocking time changes when processor utilization factor U= 0.5.

As it is clear from Figure 6, IMS is the most energy efficient algorithm especially with high blocking times, where the difference between the low and high speeds (S_L, S_m) increases significantly.

The comparison is repeated for a higher utilization factor, it is noticed that, as shown in Figure 7, IMS exhibits a slight improvement over MS with the highest blocking time because of the small difference between high and low speeds.

As a result, when the blocking time is low (the high speeds are 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), IMS behaves better than the other two algorithms (CSS and MS) especially when this difference is significant, i.e., when the processor utilization factor (low speed) is low enough.

Figure 8 shows the effects of utilization factor changes on energy consumption, where the blocking time is at the highest amount with which the task set is schedulable.

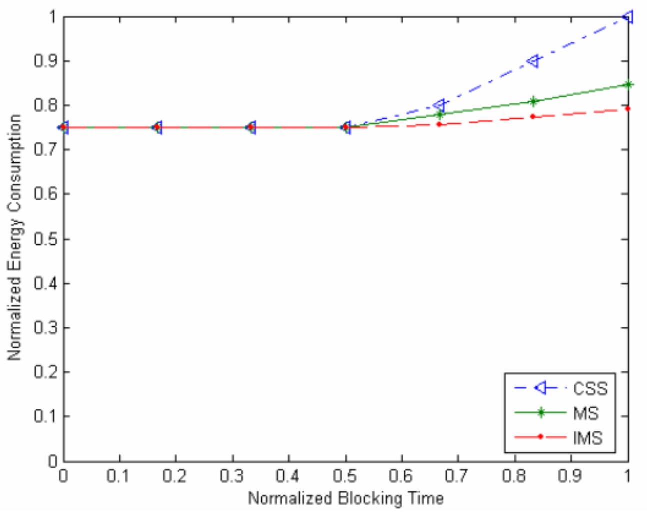

Figure 7. Energy consumption versus blocking time changes when processor utilization factor U=0.75.

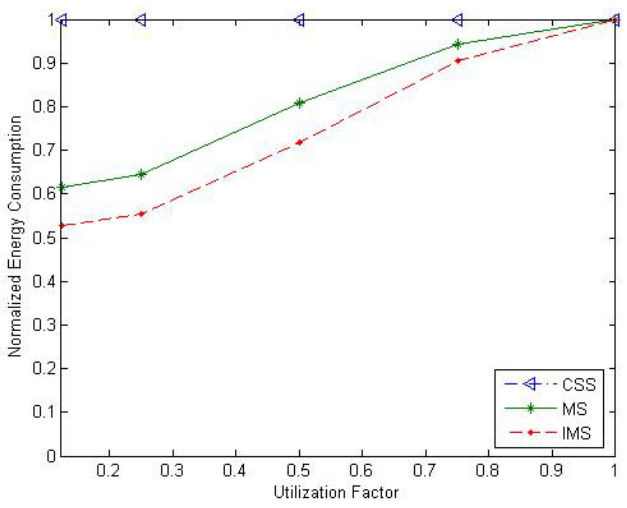

Figure 8. Energy consumption versus utilization factor changes with maximum blocking time.

It is clear that IMS is the most energy efficient algorithm especially with low utilization factors. As the utilization factor increases, the difference between low and high speeds decreases, and thus the energy consumption increases accordingly. In other words IMS shows its best performance with low utilization factors and high blocking times.

Furthermore, IMS algorithm has the advantage over MS algorithm that it can work with SRP and DPCP as resource access protocols as shown in Figures 1 and 4.

VI. 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 improvements over the existing multi-speed (MS) algorithm, where the proposed algorithm, IMS, has shown more energy saving than MS. Furthermore, it has adaptation to work not only with SRP but also with DPCP as resource access protocols.

References

[1] 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)*.pp. 28-38, 2007.

[2] V. Venkatachalam, and M. Franz "Power reduction techniques for microprocessor systems." *ACM Computing Surveys (CSUR)*,Volume 37 , Issue 3, pp: 195-237, 2005.

[3] 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.

[4] J.A. Butts, and G.S. Sohi. "A static power model for architects." *In Proceedings of the 33rd annual ACM/IEEE international symposium on Microarchitecture, Monterey, CA, USA*, pp. 191-201, 2000.

[5] G. Qu. "What is the limit of energy saving by dynamic voltage scaling? " *IEEE/ACM International Conference on Computer Aided Design, (ICCAD 2001)*, pp. 560–563, 2001.

[6] http://www.intel.com.

[7] http://www.transmeta.com.

[8] M. Weiser, B. Welch, A. Demers and S. Shenker. "Scheduling for reduced cpu energy." *In Proceedings of Symposium on Operating system Design and Implementation (OSDI),* pp. 13–23, 1994.

[9] 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,* pp. 374–382, 1995.

[10] X. Hu, and J. Quan. "Fundamentals of power-aware scheduling." In *Designing Embedded Processors – A Low Power Perspective*. J. Henkel and S. Parameswaran (eds.), pp. 219–229, Springer Netherlands, 2007.

[11] X. Hu, and J. Quan. "Static DVFS scheduling." In *Designing Embedded Proc.of A Low Power Perspective.* J. Henkel and S. Parameswaran (eds.), pp. 231–242, Springer Netherlands, 2007.

[12] P. Pillai and K. G. Shin. "Dynamic DVFS scheduling." In *Designing Embedded Processors – A Low Power Perspective.* J. Henkel and S. Parameswaran (eds.), pp.243–258, Springer Netherlands, 2007.

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

[14] A. Andrei, P. Eles, Z. Peng, M. Schmitz, and B. M. Al-Hashimi. "Voltage Selection for Time-Constrained Multiprocessor Systems." In *Designing Embedded Processors – a Low Power Perspective.* J. Henkel and S. Parameswaran (eds.), pp. 259–284, Springer Netherlands, 2007.

[15] 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,* pp. 21, 2003.

[16] 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)*, pp. 164–169, 2002.

[17] F. Zhang and S. T. Chanson. "Processor voltage scheduling for real-time tasks with non-preemptible sections." *In Proceedings of the 23rd IEEE Real-Time Systems Symposium (RTSS'02)*, pp. 235–245, 2002.

[18] L. Sha, R. Rajkumar, and J. P. Lehoczky, "Priority inheritance protocols: An approach to real-time synchronization." *IEEE Transactions on Computers*, Vol. 39, No. 9, pp. 1175–1185, 1990.

[19] 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.

[20] M. Chen and K. Lin, "Dynamic priority ceilings: A concurrency control protocol for real-time systems." *Real Time Systems Journal*, Vol. 2, No. 1, pp. 325–346, 1990.

[21] 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*, Vol. 25, No. 6, pp. 1024 – 1037, 2006.

[22] T. P. Baker, "Stack-based scheduling of real time processes." *Journal of Real-Time Systems*, Vol. 3, No. 1, pp. 67– 99, 1991.

[23] F. Zhang and S. T. Chanson. "Blocking-aware processor voltage scheduling for real-time tasks." *ACM Transactions in Embedded Computing Systems*, pp. 307–335, 2004.

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

[25] 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).* pp. 459-468, 2007.

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