ENERGY EFFICIENT REAL TIME SCHEDULING OF DEPENDENT TASKS SHARING RESOURCES

Abdullah M. Elewi, Medhat H. A. Awadalla, and Mohamed I. Eladawy
Computer Engineering Department
Helwan University
11421,Cairo, Egypt
E-mail:{ abdullahelewi@gmail.com, awadalla_medhat@yahoo.co.uk }

KEYWORDS
Energy efficient scheduling, dynamic voltage scaling, real time, embedded systems, dependent tasks.

ABSTRACT

In the design of embedded systems, it is very important to reduce energy consumption and thus prolong battery life in everywhere existed battery-powered embedded systems. Dynamic voltage scaling (DVS) processors, which support many operating voltages and speeds, can efficiently reduce energy consumption 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 where the tasks are dependent due to exclusive access shared resources. Furthermore, the paper proposes enhancements over the existing dual speed switching algorithm (DSA) where the proposed algorithm achieves more energy saving and has the capability to function with both SRP and DPCP protocols.

INTRODUCTION

Recently, embedded systems are seen in everywhere especially the portable ones such as cell phones, pocket PCs, multimedia devices, PDAs (personal digital assistants), wireless sensors, 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 embedded systems, and techniques that reduce energy consumption have been studied at different levels in details (Chen and Kuo 2007).

Dynamic Voltage Scaling (DVS) is a new technology used to reduce power consumption in real time embedded systems, where the power consumption has two essential components: dynamic and static power. The dynamic power consumption, which is the main component (Gruian 2002), has a quadratic dependency on supply voltage and can be represented as:

$$P_d = C_{ef} \cdot V^2_{dd} \cdot F \quad (1)$$

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:

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

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

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

When the processor is idle, a major portion of the power consumption comes from the leakage. Currently the leakage power significantly increases with the new generations of processors, and much work has been done to address this problem (De and Borkar 1999, Butts and Sohi 2000).

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:

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

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.

A good review has been introduced in (Chen and Kuo 2007) about types and current trends of energy efficient real time scheduling techniques on dynamic voltage scaling (DVS) platforms.

DVS processors have two types: *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.

Well-known examples of DVS processors are Intel StrongARM SA1100 processor which supports 12 voltage/speed levels, the Intel XScale which supports 4 voltage/speed levels, the Transmeta TM5400 processor which supports 6 voltage/speed levels, and the 1.6 GHz

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

Intel Pentium M processor which supports 6 voltage/speed levels.

RELATED WORK

During the last decade much work has been done in the field of energy efficient real time scheduling, but the authors of (Weiser et al. 1994) are considered the pioneers in this field, where they expected the amazing DVS technique, then Yao et al. (Yao et al. 1995) 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 (Hu and Quan 2007, Pillai and Shin 2007, Shin and Kim 2004) have been proposed and applied on uniprocessor systems. Also multiprocessor and distributed systems have been considered (Andrei et al. 2007, Mishra et al. 2003). However, the problem of DVS with dependent tasks because of shared resources has been first addressed in (Zhang and Chanson 2002, Jejurikar and Gupta 2002a).
Jejurikar and Gupta (Jejurikar and Gupta 2002a) have proposed two algorithms for scheduling fixed priority (RM scheduler) tasks using priority ceiling protocol (PCP) described in (Sha et al. 1990) 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 exclusive 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).
Then the same authors (Jejurikar and Gupta 2002b) have extended their previous algorithms (CSMS and CSS) to handle dynamic priority (EDF scheduler) tasks using dynamic priority ceiling protocol (DPCP) shown in (Chen and Lin 1990). The dynamic priority ceiling protocol is an extension of original priority ceiling protocol to deal with dynamic priority tasks (EDF scheduling).
Jejurikar and Gupta (Jejurikar and Gupta 2006) 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 (Zhang and Chanson 2002) have worked on the same problem, and proposed three algorithms for energy efficient scheduling of dependent tasks with shared resources, but they made use of stack resource policy (SRP) proposed by Baker (Baker 1991) 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 (Jejurikar and Gupta 2002b) because they all have derived the static slowdown factor directly from the EDF schedulability test with blocking time:

$$\overset{\forall i}{i=1,\ldots,n} \quad \frac{B_i}{D_i}+\sum_{k=1}^{i}\frac{C_k}{D_k}\le 1 \tag{5}$$

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 within which a high priority task can be blocked by a low priority task due to mutual exclusion shared resource.
The second algorithm, which is the interest of this paper, is the dual speed switching algorithm. The main concept of this algorithm is using two speeds (H, L) and switching between them. Initially the algorithm operates at 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 for further reducing of the processor speed and achieving more energy saving.

SYSTEM MODEL

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 (Cottet et al. 2002):

- 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.

Processor Model

The tasks are scheduled on a single DVS processor that 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 required to change the processor voltage/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.

DUAL SPEED SWITCHING ALGORITHM

Dual speed algorithm (DSA) proposed in (Zhang and Chanson 2002) is a blocking aware scheduling algorithm with non-preemptible critical sections using SRP as resource access protocol.
The authors have noted that the static speed in the constant static slowdown (CSS) algorithm is higher than the required, and CSS consumes a lot of energy, but it is extremely effective when a task is blocked to avoid deadline miss. So, if it is possible to switch between two speeds: high H and low L (where $L \leq H \leq 1$), the algorithm will be more energy efficient, and this is the concept behind DSA.
The high speed is the same as the static speed in CSS, and it is the speed that satisfies all tasks to meet their deadlines when a blocking occurs. The high speed is derived directly from the EDF schedulability test with shared resources and SRP protocol:

$$\forall i \quad i = 1,\ldots,n \qquad \frac{B_i}{D_i} + \sum_{k=1}^{i} \frac{C_k}{D_k} \leq H \tag{6}$$

The low speed is the optimal lowest speed with which all tasks can be scheduled without missing any deadline (no blocking occurs), and it is derived from the plain EDF schedulability test without shared resources:

$$\sum_{i=1}^{n} \frac{C_i}{D_i} \leq L \tag{7}$$

Initially, DSA starts with the low speed L, then it switches to the high speed H as soon as a task is blocked.

An Illustrative Example

An example presented in (Jejurikar and Gupta 2006) is implemented to illustrate the dual speed algorithm and the contribution of this paper. A hard real time system with two tasks is considered as following:

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

The arrival times and critical sections of the two tasks within the least common multiple (LCM) of periods are shown in Figure 1(a). τ_1 may be blocked by the critical section of the lower priority τ_2, so the blocking time of τ_1 is the length of τ_2's critical section, i.e. $B_1=3$, and in the same manner $B_2=0$ because there are no lower priority tasks to block τ_2.
According to (6) the static (high) speed is H= max (2/5+3/5, 2/5+4/40) =1 which is used to schedule the tasks in CSS algorithm as shown in Figure 1(b), then using (7) let us calculate the low speed L= 2/5+4/40=0.5 to use it with the high (static) speed calculated earlier in the scheduling of the two tasks (τ_1,τ_2) using DSA as shown in Figure 1(c).
The rectangles represent the processing of tasks, 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 that represent the jobs of the same task is the same because these jobs always take the same number of execution cycles which equals to processor speed multiplied by elapsed time.

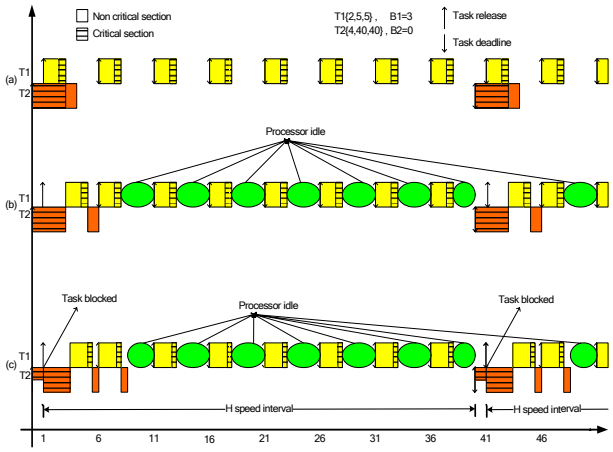

Figure 1: (a) Task Set Description: Arrival Times, Computation Times, and Critical Sections. (b) CSS. (c) DSA.

It is clear in this example that the CSS does not save any energy because it has used the maximum speed. Due to such situations DSA has been developed to achieve more energy savings.

Enhanced Dual Speed Algorithm

It is obvious in Figure 1 that DSA has used the low speed just for one second, and then it has switched to the high speed and continued until the blocking task (the low priority task) deadline which is mostly further than the blocked task (the high priority task) deadline. This highlights the key idea of the first contribution of the enhanced DSA (EDSA) proposed in this paper to end the high speed interval by the blocked task deadline which is mostly earlier.
To test the performance of EDSA, the previous example has been performed using DSA and EDSA. As shown in Figure2, the processor idle time is reduced from 49% in DSA to 12.5% in EDSA which reflects the improvements achieved in the system performance.

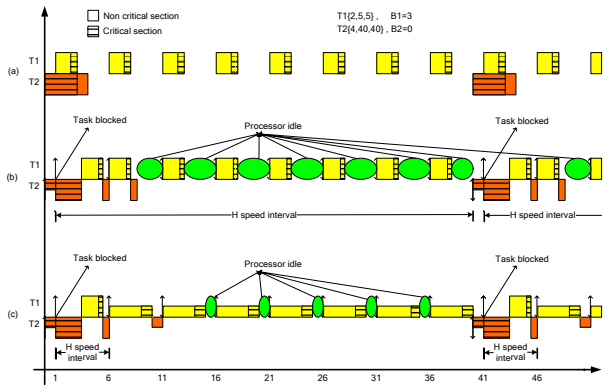

Figure 2: (a) Task Set Description. (b) DSA. (c) EDSA.

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

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

The arrival times and critical sections of the three tasks within the least common multiple (LCM) of periods are shown in Figure 3(a). τ_1 may be blocked by the critical section of the lower priority τ_2 or τ_3, so the blocking time of τ_1 is the length of longest critical section in the worst case, i.e. B_1= max (1.5, 2.5) =2.5, and in the same manner B_2=2.5 because τ_2 can be blocked by τ_3, while B_3=0 because there are no lower priority tasks to block τ_3.

According to (6, 7), the high speed H=max (1/4+2.5/4, 1/4+2/8+2.5/8, 1/4+2/8+3/10) =0.875, and the low speed L =1/4+2/8+3/10=0.8.

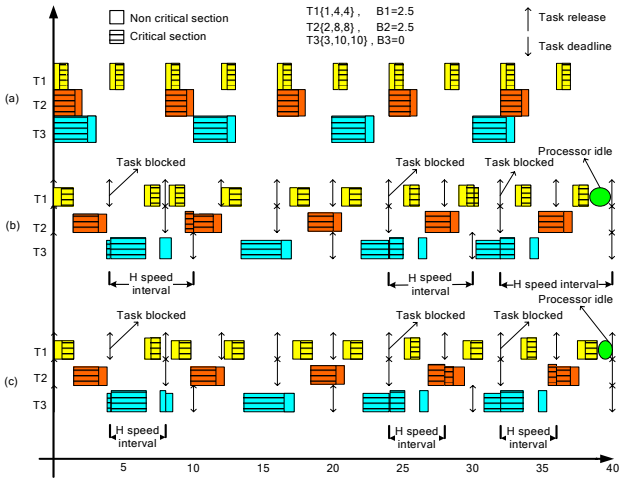

Figure 3: (a) Task Set Description. (b) DSA. (c) EDSA.

Even though the high speed and low speed are close to each other in this example, there is again an improvement in the system performance based on EDSA compared with DSA where the processor idle time is reduced from 4.28% to 2.8%.

DSA and Priority Ceiling Protocols

Jejurikar and Gupta (Jejurikar and Gupta 2006) have shown that the success of DSA requires that the critical sections be non-preemptible, and DSA 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. However, the blocking occurs at the moment when the high priority task needs to access the resource, and the switching to high speed H is delayed until this moment causing deadline miss as shown in Figure 4.

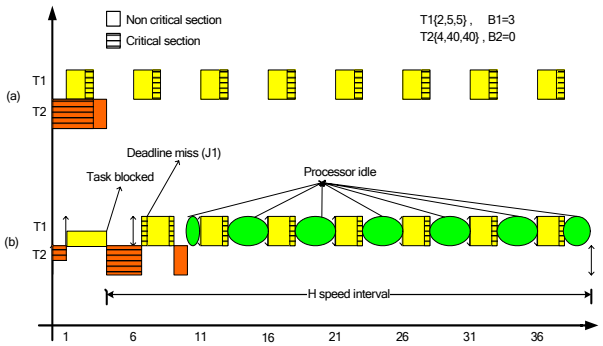

Figure 4: (a) Task Set Description. (b) Deadline Miss When DSA Is Used With DPCP.

At the moment t=1, task τ_1 is released, and task τ_2 that executes its critical section is preempted by the high priority task τ_1. Task τ_1 is just blocked when it needs to access the shared resource, i.e. at the moment t=4, where the high speed interval 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 4(b).

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

Figure5 shows the adaptation of EDSA with DPCP as resource access protocol to avoid deadline misses and achieve energy saving as done with SRP, where the switching to the 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 2(c) and 5, and this is the key advantage of SRP over PCPs as mentioned earlier.

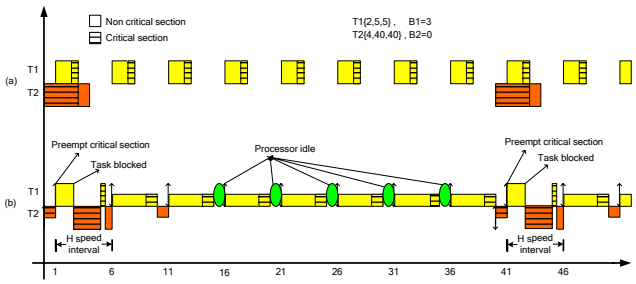

Figure 5: (a) Task Set Description. (b) Deadline Meeting When EDSA Is Used With DPCP.

Figure6 shows the proposed algorithm EDSA which achieves more energy saving and works with SRP and DPCP as resource access protocols.

/* Initially the processor speed is L. End_H indicates the end of the high speed interval. If the system isn't in a high speed interval, End_H = -1. Initially End_H = -1 */

When job $J_{i,j}$ arrives:
 if Priority($J_{i,j}$) > Priority(current job)
  if Preempt_Current_Job( ) is successful
   if Preempt_Critical_Section( ) is successful
    Set_Speed(H); /* Set the processor speed at H*/
    End_H = max(End_H, $d_{i,j}$);
     /* d is the deadline of the job */
   end if
   Execute $J_{i,j}$;
  else /* $J_{i,j}$ is blocked */
    Set_Speed(H); /* Set the processor speed at H*/
    End_H = max(End_H, $d_{i,j}$);
  end if
 end if

when the end of high speed interval is reached:
  End_H = -1;
  Set_Speed(L); /* Set the processor speed at L */

Figure 6: The Enhanced Dual Speed Algorithm (EDSA)

RESULTS AND DISCUSSION

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 Figure7, 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.

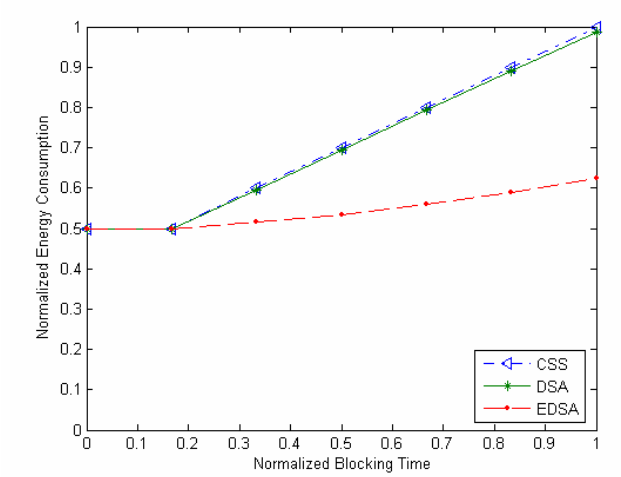

Figure 7: Energy Consumption Versus Blocking Time Changes In Example1

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.

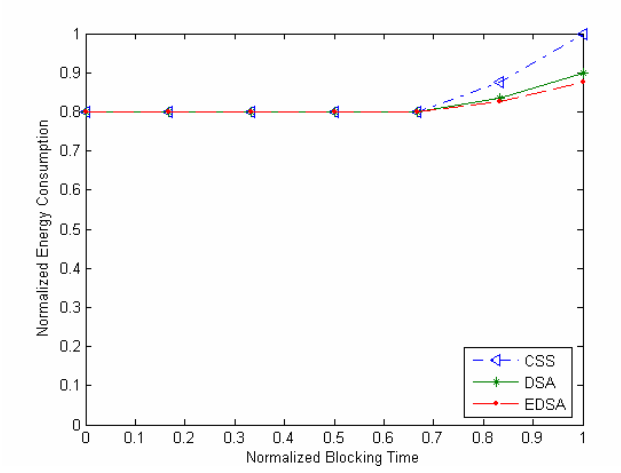

Figure 8: Energy Consumption Versus Blocking Time Changes In Example2

Furthermore, EDSA has the advantage over DSA that it can work with SRP and DPCP as resource access protocols as shown in Figures 2 and 5.

CONCLUSIONS

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 dual speed switching algorithm (DSA), where the proposed algorithm, EDSA, has shown more energy saving than DSA. furthermore, it has adaptation to work not only with SRP but also with DPCP as resource access protocols.

REFERENCES

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

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

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

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

Chen, J. and C. Kuo. 2007. "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).

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

De, V. and S. Borkar. 1999. "Technology and Design Challenges for Low Power and High Performance". In Proceedings of the International Symposium on Low Power Electronics and Design, San Diego, CA, USA, 163-168.

Gruian, F. 2002. Energy-Centric Scheduling for real time systems, PhD thesis , Lund Institute of Technology, Lund University,

Hu, X. and J. Quan. 2007. "Fundamentals of Power-Aware Scheduling". J. Henkel and S. Parameswaran (eds.), Designing Embedded Processors – A Low Power Perspective, 219–229..

Hu, X., J. Quan. 2007. "Static DVFS Scheduling." J. Henkel and S. Parameswaran (eds.), Designing Embedded Proc.of A Low Power Perspective, 231–242.

Jejurikar, R. and R. Gupta. 2002. "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), 164–169.

Jejurikar, R. and R. Gupta. 2002. "Energy aware edf scheduling with task synchronization for embedded real time operating systems," Technical report #02-24. Department of Information and Computer Science, University of California at Irvine, (Aug).

Jejurikar, R. and R. Gupta. 2006 "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, 1024 – 1037

Mishra, R., N. Rastogi, D. Zhu, D. Moss´e, and R. Melhem. 2003. "Energy aware scheduling for distributed real-time systems". In International Parallel and Distributed Processing Symposium, 21-30.

Pillai, P. and K. G. Shin. 2007. "Dynamic DVFS Scheduling." J. Henkel and S. Parameswaran (eds.), Designing Embedded Processors – A Low Power Perspective, 243–258.

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

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

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

Yao, F., A. Demers and S. Shenker. 1995. "A scheduling model for reduced cpu energy". In Proceedings of IEEE Annual Symposium on Foundations of Computer Science, 374–382.

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

AUTHOR BIOGRAPHIES

ABDULLAH M. ELEWI graduated from the Electrical and Electronic Engineering Faculty of Aleppo University, Syria, where he studied computer engineering and obtained his Bachelor degree in 2005 and graduate studies Diploma in 2006, then he moved to Egypt to complete his graduate studies. He is currently a master student at the computer engineering department at Helwan University. His research interests include energy efficient real time scheduling techniques, embedded systems and real time operating systems. His e-mail address is: abdullahelewi@gmail.com

MEDHAT H. A. AWADALLA has obtained his B.Sc. from Helwan University, 1991 in electronics and communications department and got his M.Sc in the field of reconfigurable computer architecture in 1996 from Helwan university and His PhD from Cardiff university, UK in the filed of mobile robots in 2005. He has a post Doc. at Cardiff university in 2006 and currently he is working as a lecturer in local and private universities in Egypt as Helwan university, Misr international university, and Misr university for science and technology. His research interests include real time systems, multi-core processors, computing grid, mobile robots, and sensor networks. His e-mail address is : awadalla_medhat@yahoo.co.uk

MOHAMED I. ELADAWY graduated from the Department of Electrical Engineering, Faculty of Engineering of Assiut University in May 1974; M.Sc. from Cairo University in May 1979; Ph.D. from Connecticut State University, School of Engineering, in May 1984. He worked as an Instructor at the Faculty of Engineering, Helwan University since 1974. Currently he is a Professor at the Department of Communication and Electronics Engineering and the Vice Dean for Student Affairs in the same faculty. He was working for the general organization for technical and vocational training for 6 years from 1989 to 1995 in Saudi Arabia. His main research interests include signal processing and its medical applications, real time systems, and image processing.