International Journal of Electronics and Electrical Engineering

Volume 1 | Issue 1 Article 4

July 2012

Dynamic Voltage Scaling With Reduced Frequency Switching And Preemptions

Arya Lekshmi Mohan
Dept. of Electrical and Electronics Engineering Amrita School of Engineering, Coimbatore, India, aryalekshmim@yahoo.co.in

Anju S. Pillai
Dept. of Electrical and Electronics Engineering Amrita School of Engineering, Coimbatore, India, s_anju@cb.amrita.edu

Recommended Citation
Mohan, Arya Lekshmi and Pillai, Anju S. (2012) "Dynamic Voltage Scaling With Reduced Frequency Switching And Preemptions," *International Journal of Electronics and Electical Engineering*: Vol. 1 : Iss. 1 , Article 4.
DOI: 10.47893/IJEEE.2012.1003
Available at: https://www.interscience.in/ijeee/vol1/iss1/4

Dynamic Voltage Scaling With Reduced Frequency Switching And Preemptions

Arya Lekshmi Mohan, Anju S.Pillai
Dept. of Electrical and Electronics Engineering Amrita School of Engineering
Coimbatore, India
E-mail - aryalekshmim@yahoo.co.in, s_anju@cb.amrita.edu

Abstract*— *Dynamic Voltage Scaling is an innovative technique for reducing the power consumption of a processor by utilizing its hardware functionality. Dynamic Voltage Scaling processors are mainly focusing on power management. Such processors can be switch between discrete frequency and voltage levels. The main challenges of Dynamic Voltage Scaling are increased number of preemptions and frequency switching. A part of dynamic energy as well as CPU time is lost due to these processes. To limit such processes, an algorithm is proposed which reduces both unwanted frequency switching and preemptions.

Keywords-preemption; frequency switches; dynamic voltage scaling; Open Media Application Platform;

I. INTRODUCTION

Power consumption reduction has got great importance in real-time embedded systems. In an electronic system, major part of power is drawn by the processor.

The most commonly using microprocessors for embedded applications are CMOS processors. The main characteristics of a CMOS processor are its immunity to noise and low static power dissipation. The dynamic power taken by a CMOS chip is given by

$$P=c_{eff}\, v^2 f. \qquad (1)$$

Where P is the power consumed by the processor, c_{eff} is the load capacitance, v is the operating voltage and f is the operating frequency. Power consumption of the processor can be reduced by decreasing the operating voltage as well as the frequency. A significant amount of power can be reduced by lowering the voltage, since it has a quadratic relation to power. Now, in all DVS processors voltage reduces along with frequency.

XScale, Transmeta Crusoe, OMAP are some of the DVS processors which has been evolved in last decade. Among these processors OMAP has got many power management characteristics like DVS, AVS etc. These processors have discrete number of operating points to switch in between. As number of voltage level increases, the performance of the system also increases, since the tasks will get more choices to select the frequency and can run without loss of much energy.

Transition time is the time taken by a task to switch from one frequency to another. Transition time is an overhead for the real-time systems, since it will keep the processor idle for some time. Switching from one frequency to another may cause power loss due to charging and discharging of gate capacitors. Preemptions are another overhead to Dynamic Voltage Scaling; hence it requires energy for context switching. The energy required for context switching is given as 0.20 mJ [1]. Transition latency of processor is approximately 300 micro seconds.

The following section of this paper describes the similar works done in this area. Phase 3 describes the background for this work. In phase 4 is given with algorithm for reducing task switching and frequency switching. Hardware implementations details are given in section 5 Experimental results are given in section 6.

II. RELATED WORKS

One of the disadvantages of Dynamic Voltage scaling is increasing the number of preemptions. To reduce that overhead, Preemption-Aware DVS for hard Real-Time Systems is implemented. Two preemption control Methods are proposed in paper [2] they are accelerated-completion based technique and delayed preemption based technique. In that work the preemption is reduced by postponing the high priority tasks and forces low priority tasks to complete the execution. Integration of preemption threshold scheduling and dynamic voltage scaling is given by Jejurikar et al. In paper [1], an algorithm to compute threshold preemption levels for tasks with given static slowdown factors is presented. The proposed algorithm improves upon known algorithms in terms of time complexity. The experimental result shows that preemption threshold scheduling reduces on an average 90% context switches, even in the presence of task slowdown. There are two methods for frequency transition elimination is given in the paper [3], they are offline and online. Extra energy will be consumed during a voltage transition because, a voltage transition itself consumes energy and voltage transitions take a fixed amount of time, within which no jobs can be executed.

Another effort for reducing the frequency switching is done by Muhammad et al. This algorithm is applied only when the number of ready task is one. By this algorithm an average of 30% of the frequency switching is reduced. [4].

The first implementation of RT-DVS algorithm was done by Pillai et al [5]. In that paper they had given three

DVS algorithms named Static Voltage Scaling, Cycle–conserving DVS and Look-ahead DVS. Among these algorithms Look-ahead gives a better result. In this paper task preemptions are taken as negligible. Experiments show that these algorithms reduce approximately 20-40% of energy consumption. PLG with look-ahead RT-DVS is another algorithm. Usually for a processor a set of voltage-frequency pairs will be given for scaling. But this voltage levels may not be acceptable to the peripherals. The peripherals might have problems running at voltages below a certain threshold. To overcome this problem a set of intermediate frequencies are required, these frequencies are called pseudo frequencies. These frequencies are obtained by taking linear combinations of supporting frequencies [6].

III. SYSTEM MODEL

A. Task Model

The task set used in this paper include periodic tasks. Let T (r_i, c_i, d_i) represents a periodic task with release time r_i, worst case computation time as c_i and relative deadline as d_i. The assumption is taken here is that all tasks are independent.

B. Task scheduling algorithm

Task scheduling algorithm used in this work is Earliest Deadline First. This is a dynamic scheduling policy and it is used for uniprocessor with the assumption that task period is equal to deadline. EDF scheduling policy is based on the relative deadline of ready tasks. The condition for a task to be scheduled by EDF is that total utilization of the system should be less than or equal to 1. The utilization can be calculated from the given equation 2, where C_i is the computation time and D_i is the deadline of i th task and U is the total utilization.

$$U=\sum_{i=1:n} C_i/D_i \quad (2)$$

C. Task Set Generation

Task set required for the experiment is obtained by UUnifast algorithm [7]. This algorithm will generate individual utilizations for the desired system utilization. From these utilizations, execution time and deadlines are generated.

D. Processor Model

Processor model used for the experiment is a DVS processor with n number of frequency levels and its corresponding voltage levels. Let denote f_1, f_2, $f_3 \ldots f_n$ are the frequencies and v_1, v_2, $v_3 \ldots v_n$ are the corresponding voltages. Based on the load at a particular point in time, one of these operating points can be selected, with assumption that any of the tasks will never miss its deadline when it is executed with its worst case execution time.

IV. MODIFIED DYNAMIC VOLTAGE SCALING ALGORITHM

Dynamic voltage scaling algorithm used in this work is Look-ahead RT DVS [5] with modifications. In look-ahead, the high priority task tries to execute with low frequencies, and in order to meet deadlines, the low priority tasks run with higher frequencies. Look-ahead algorithm defers as much work as possible, and sets the operating frequency such that the minimum work that must be done now to ensure all future deadlines are met [5]. If much of slack is available then the scheme tries to execute all the tasks with lower frequency. The assumption taken here is that the task will take less time to execute i.e. Actual execution times of the tasks are less than its worst case execution times. Look-ahead algorithm does not deal with the energy loss due to unnecessary frequency switching and preemption.

For reducing the number of frequency switches, the algorithm will check whether any task is there in ready queue with same frequency as the frequency of previous task.

Let $T_1, T_2, T_3 \ldots T_N$ are the periodic tasks in the system, and $C_1, C_2, C_3 \ldots C_N$ are the computation time of those tasks respectively. Let $f_1, f_2, ..f_N$ are the frequency of the periodic tasks and $d_1, d_2, ..d_N$ are the deadline of the tasks. Consider T_P as previously completed task and T_C as current task ready to run. Consider the computation time of current task ready to run as C_c, the frequency of previously run task as f_p, and frequency of currently selected task for running as f_c. Let the deadline of current task as d_c and current tick as t_c, Ti is the ready tasks at t=tc, C_i is the computation time of ready tasks and fi is the frequency of ready tasks. N is the number of ready task at t=tc.

Pseudo code for new algorithm is given in the next subsection. The Defer function [5] given here is used for computing the frequency of the ready tasks. The flow chart for reducing the frequency switching is given in figure 1.

A. Pseudo code

```
Upon task release(Ti)
    Set $C_{left\_i}$= Ci
Upon task completion
    Set $C_{left\_i}$=0
During task execution
    Decrement $C_{left\_i}$
    Defer()

Preempt_Freq_switch():

If curnt_Task $\neq$ Prev_Task
    For i=1 to N
        if freq_i = freq_previous
            if $C_{left\_current} + C_{left\_i} <=$ (deadline_curnt_Task –- current_time)
                Current_task=i
                break
end
```

```
            end
            end
        if $C_{left\_previous} \neq 0$
if $C_{left\_current} + C_{left\_previous} <= $ (deadline_current_task –- current_time)
        Current_task=Previous_task
            end
        end;end
```

When preemption occurs, the algorithm checks whether the sum of remaining execution time of preempted task and computation time of current task is less than the difference between the deadline and current tick. If this condition is satisfied then the current task will continue other vice that preemption cannot be avoided using this method.

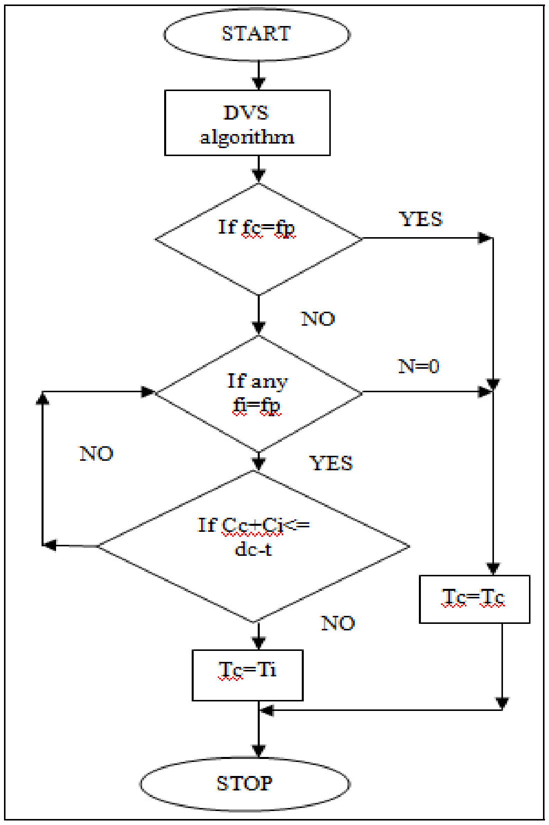


Figure 1. Flow chart for reducing the frequency switching

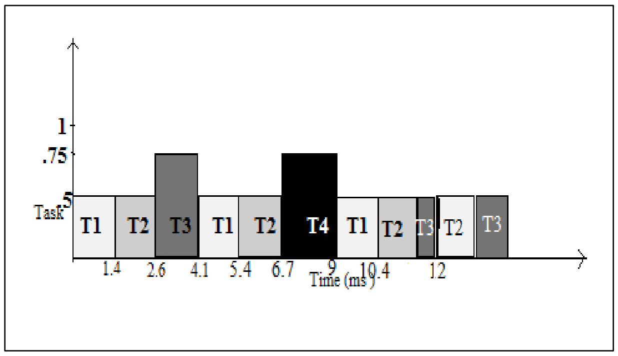
B. Example

Table 1. Example task set

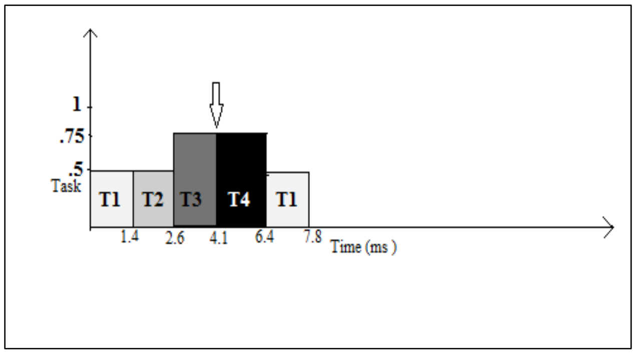
TASK NO	EXECUTION TIME	DEADLINE
1	1	4
2	1	5
3	2	10
4	4	12

An example task set is given in the table 1. Figure 2 shows the graph that obtained before applying the algorithm. Figure 3 shows the result after applying the frequency switching. The graph obtained after avoiding the preemption is given in figure 4.

In the given example of the task set, first instance of Task1 and Task2 run with normalized frequency .5, and Task 3 runs with frequency of .75. At time t=4.1 ms the task T1 is ready for running with a frequency of .5 and there is another task, Task4 having same frequency as the previous task that



had already run (T3), so the algorithm checks whether the Task4 can run without missing the deadline of Task1. The



algorithm selects Task 4 and it executes till time, t=6.4ms.

Figure2. Example trace before applying the algorithm

Figure 3. Trace showing output of frequency switching reduction algorithm

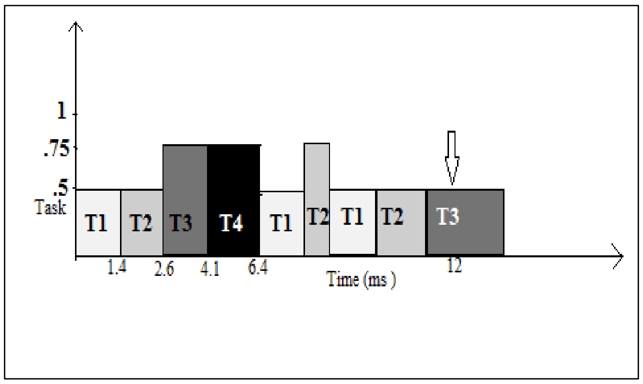


Figure 4. Trace Showing Avoided Preemption

Figure 3.Shows the trace, after applying the algorithm on example task set. Before applying the algorithm there are 3 frequencies switched. But after applying the algorithm it reduced to two frequency switching. The point at which frequency switching avoided is shown by the arrow.

The same algorithm also checks for pre-emption. According to DVS algorithm, at t=12 Task 3 is pre-empted by Task 1; trace is given in figure 2. New algorithm verified and found that preemption can be avoided, and continues with execution of Task3. the trace after avoiding the preemption is given in figure 4.

V. HARDWARE IMPLEMENTATION

Hardware platform used for this work is beagle board with OMAP processor. OMAP is a DVS processor and it can support 5 discrete frequency and voltage pairs. The specialized companion IC called TPS 65950 helps to perform the power management functions.TPS 65950 consists of Low Drop Out regulators (LDO) and DC-DC converters. The cpufreq frame work in Linux supports the power management of OMAP processor. The experiment is conducted on Linux Angstro**m version, which provides the cpufreq power management** frame work. Governors are present in cpufreq, which decides what frequency range should be used. The cpufreq provides the scaling_available_governors, which consists of conservative, ondemand, power save, userspace and performance. In this work, the governor used is userspace. Userspace enables the user program to choose the frequency level.

The default frequency supported by OMAP processor is 720000Hz.The transition latency for one frequency switch in OMAP processor is 300 micro seconds. The frequency required to execute the current task is chosen by the algorithm. The frequency of the processor can be changed by using the command:

System (“echo 600000 > /sys/devices/ system /cpu/cpu0/cpufreq/scaling_setspeed”)

The operating performance points (OPP) supported by OMAP processor is given in Table 2.

Table 2
Operating points of OMAP processor

Frequency(MHz)	Voltage(v)
125000	0.9
250000	1.0
500000	1.2
550000	1.27
600000	1.35

After running the tasks; the time at which the processor remains in each frequency state can be obtained from the system command:

System ("cat /sys/devices /system/cpu /cpu0 /cpufreq/stats/time_in_state")

For hardware implementation, two LED task are defined and those tasks run with reduced frequency and total energy taken by these tasks are calculated and obtained as 0.1 J.

VI. SIMULATION RESULTS

The figure 5 gives the plot for normalized energy Vs. utilization. In that figure, the comparison of power consumed by the processor without DVS algorithm, with DVS algorithm and with modified algorithm are given. From the figure it is clear that about 2-8% of energy can be further reduced by reducing the number of preemptions and frequency switching. After applying the modified algorithm an average of 50 % of energy is saved.

Fig.6 provides the plot for showing the frequency switching reduction. This simulation is conducted with 10 different task set. By this method an average of 35% frequency switching is reduced.

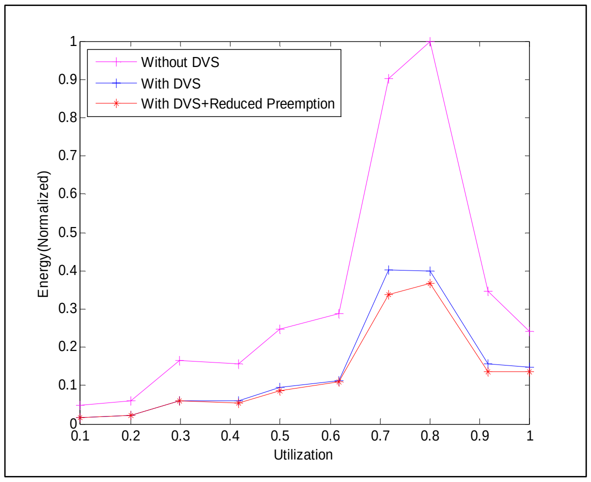


Figure 5 . Normalized Energy Vs Utilization

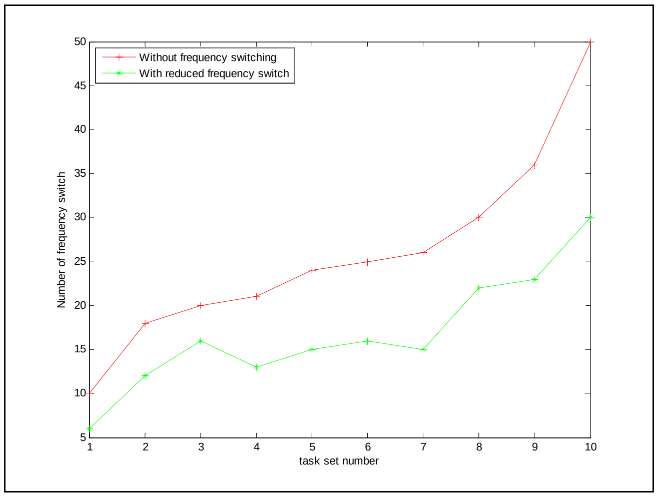


Figure 6. Frequency Switching Reduction

CONCLUSION

Dynamic voltage scaling is used to reduce the power consumption in processors. As the number of operating points increases the chance of occurring the preemption and frequency switching is more. An effective algorithm is applied to tasks in order to overcome these problems. After reducing the pre-emption and frequency switching, energy is reduced by 2-5% in the modified algorithm, and about 35% frequency switching is reduced by this algorithm. Therefore, a total of 50% of the average power consumption reduction is obtained after applying the modified algorithm. Dynamic Voltage Scaling algorithm is implemented in an OMAP processor.

ACKNOWLEDGMENT

I would like to thank God Almighty for giving me this opportunity to complete this work. I would like to extend my gratefulness to Dr. T.N Padmanabhan Nambiar for providing the valuable support and help in this work.

REFERENCES

[1] Ravindra Jejurikar and Rajesh Gupta,"Integrating Preemption Threshold Scheduling and Dynamic Voltage Scaling for Energy Efficient Real-Time Systems," *Proceedings of the Ninth International Conference on Real-Time Computing Systems and Applications*, 2004.

[2] Woonseok Kim, Jihong Kim, Sang Lyul Min," Preemption-Aware Dynamic Voltage Scaling in Hard Real-Time Systems," In *Proceedings of the 2004 international symposium on Low power electronics and design* (ISLPED '04). ACM, USA, pp. 393-398, 2004.

[3] Bren Mochocki, Xiaobo Sharon Hu, Gang Quan, "Transition-Overhead-Aware Voltage Scheduling for Fixed-Priority Real-Time Systems," *ACM Transactions on Design Automation of Electronic Systems*, Vol. 12,No. 2, Article 11, April 2007..

[4] Farooq Muhammad, Bhatti M. Khurram, Fabrice Muller, Cecile Belleudy, Michel Auguin, "Precognitive DVFS: Minimizing Switching Points to Further Reduce the Energy Consumption," *Proceedings of the 14th Real-Time and Embedded Technology and Applications Symposium*, 2008..

[5] Padmanabhan Pillai and Kang G. Shin,"Real-Time Dynamic Voltage Scaling for Low Power Embedded Operating Systems," *Symposium on Operating Systems Principles'01,* pp. 89-102, October,*2001*.

[6] Venkat Rao, Gaurav Singhal, Anshul Kumar," Real Time Dynamic Voltage Scaling for Embedded Systems", 17th International Conference on VLSI Design, Mumbai, India, 2005.

[7] Enricobini, Giorgioc Buttazzo, "Biasing Effects in Schedulability Measures," In *Proceedings of the 16th Euromicro Conference on Real-Time Systems* (ECRTS '04). IEEE Computer Society, USA, pp.196-203,2004.