Voltage and Frequency Scaling Techniques for Power Aware Scheduling

Shreyas Poyrekar *University of Houston*

***Abstract*—Power and energy are one of the major constraints of real-time embedded systems. Efforts are being made to reduce the power consumed in the real time system, instead of increasing the battery capacity. Power aware scheduling can be the solution to reduce the energy consumption in such cases. This paper provides information about the power performance trade-off in the CMOS devices which forms the basis of the power-aware scheduling techniques. An overview of the voltage and frequency scaling based power-aware scheduling is provided.**

***Index Terms*—Power-Aware scheduling, Static/Dynamic Voltage Frequency Scaling, Earliest Deadline First (EDF), online scheduling.**

I. Introduction

POWER aware Scheduling techniques are needed for the real-time embedded systems which are portable and compact such as smartphones, smart watches, communication devices, transportation machines, entertainment appliances, and medical instruments. The increase in need of more battery capacity and small size, research & development is being done in reducing the power consumption of the devices at the system level. It is widely recognized that decisions made in early design phases at system level are of critical importance in keeping power/energy demands in check[1][4]. To solve this primary issue different power-aware scheduling techniques are being developed[6], [7], [8], [9], [10], [13], [14], [15], [16], [17], [18], [19], [20], [21]. This paper/report aims to introduce to static/dynamic voltage scaling algorithms. The static DVFS & cycle conserving DVFS algorithms are simulated in python. The next section II discusses the power and performance tradeoff in the microprocessor which leads to the fundamentals of power-aware scheduling. Section III classifies the power reduction techniques and discusses the hardware resources needed. Followed by section IV introduces real-time DVFS and compares it with general DVFS algorithms. Section V goes through the static voltage scaling RT-DVFS algorithm discussing the method and efficiency. Section VI discusses the Cycle conserving RT-DVFS algorithm followed by Look ahead RT-DVFS algorithm in section VII both of which are dynamic voltage scaling techniques. In section VIII python simulation is results are shown. Section IX and X suggests about the further reading and implementation and states the conclusion.

II. Performance Trade-Off in CMOS devices

Microprocessors, micro-controllers & digital logic circuits are CMOS based integrated circuits. Power dissipation in a CMOS circuit is static and dynamic power dissipation. Low static power dissipation is an essential characteristic of a CMOS device. Meaning most of the dissolution is only due to the dynamic dissipation in the circuit. Static dissipation is due to the leakage currents in the reverse bias junctions and the subthreshold currents. The leakage currents in a CMOS circuit increases with the scaling of the device. The static power dissipation in terms of leakage current (I_{leak}) and voltage (V) is given as

$$P_{leak} = I_{leak}V, \tag{1}$$

Dynamic dissipation is when in a cycle the load capacitance's are charged and discharged during switching. dynamic power dissipation is expressed as

$$P_{dyn} = \alpha C_L V^2 f, \tag{2}$$

where α is the switching activity factor as most of the gates are not switched at every clock cycle, C_L is the load capacitance, f is the frequency and V is the voltage.[2][3][4]

Observing the equations (1) and (2), we can say that scaling the voltage can effectively reduce both the static and dynamic dissipation in a CMOS circuit. The frequency & voltage in a microprocessor is dependent on each other. If the frequency is scaled, the voltage is reduced. Using these two observations, we can say that scaling the voltage and frequency in a CMOS device can eventually minimize power consumption. But reducing the supply voltage will increase the execution time of the tasks. This is because the circuit delay (τ) is inversely proportional to the maximum working frequency. The circuit delay (τ) is expressed in terms of the supply voltage (V) and threshold voltage (V_{th}) as

$$\tau \propto \frac{V}{(V - V_{th})^2}, \tag{3}$$

Increase in the circuit delay due to decrease in the voltage does not guarantee the timing constraints of the tasks running on the system making it a soft real-time scheduling.[2][4]

III. Power Manageable Hardware Resources

The tasks do not always occupy the processor, i.e. for the significant time of operation the processor is busy only for some time and idle for the rest. During the idle state, the power dissipation is only due to the static dissipation. Running the processor in low leakage mode during the idle state can reduce the leakage power consumption. Shutting down the processor can be considered as running the processor in low leakage mode. This is known as dynamic power management (DPM) mechanism. As for the reduction of power consumption during the execution time by scaling the voltage and frequency is known as dynamic voltage/frequency scaling mechanism. These are the two primary categorized run-time power reduction hardware mechanism[1][2].

The timing overhead for the processor to wake up after shutting down is significant such that performance is degradable and Scaling the voltage/frequency will result in increase of the tasks execution time. Thus use of any power reduction mechanism eventually threatens the timing requirements of the tasks making the system unacceptable. Thus, the challenge of the power-aware scheduling algorithms is to exploit the slack times & obtain an appropriate processor speed at which the timing constraints are met[1][2].

IV. Real Time DVFS

As we know that the processor is not always busy, i.e. the average computation capacity required is most of the time much lower than the peak computation capacity needed to meet the performance requirements[5]. DVFS can provide both low power and high performance in the same system. DVFS is a hardware implementation technique. A programmable DC-DC converter, clock generator, a processor having wide operating ranges are required for DVFS. A general dynamic voltage scaling algorithm cannot be applied to portable, real-time embedded devices like cellular phones. These are based on average computational throughput having a feedback mechanism which detects the idle time of the processor over a period of time and adjusts the frequency & voltage to run the tasks[8][16][21]. This does not guarantee any timing constraints[5].

Real-time DVFS are tightly coupled having an actual real-time scheduler. For real-time DVFS, a set of tasks T_i each having a period of P_i and worst-case computation time C_i are considered. The tasks are released periodically at P_i time units and should be executed before its deadline. The real time scheduler guarantees execution of all tasks if the execution times of T_i does not increase C_i. A real-time DVFS scheduler guarantees that tasks will meet their deadlines given that the task set is schedulable and no task exceeds its specied worst-case computation bound. Real-time DVFS is an integration of DVS mechanisms into real-time schedulers such as Rate Monotonic (RM) and Earliest-Deadline-First (EDF) schedulers. RM is a static priority scheduler which assigns task priority based on their periods. EDF is a dynamic priority scheduler which sorts tasks by their deadlines and gives the priority accordingly. Real-time DVFS assumes that the scheduling and preemption overheads are zero and the tasks are idependent[5].

V. Static Voltage Scaling RT-DVFS

In this mechanism, the lowest operating frequency is selected at which the tasks set are RM or EDF schedulable. The frequency is set statically and is not changed unless the task set is changed. If the frequency is scaled by a factor α, then the worst case computation time is scaled by 1/α. The schedulability test for EDF and RM scheduling is modified by using the scaled values of the worst-case computation time to select the frequency which gives the corresponding voltage at which the tasks is to be executed, keeping the period and the deadline unchanged. For any tasks to be schedulable by EDF, the sum of the worst-case utilization should be less than or equal to 1[12].

$$C_1/P_1 + .. + C_1/P_1 \leq 1, \tag{4}$$

Using the scaled computation time,[5]the EDF schedulability test for static DVFS is given as

$$C_1/P_1 + .. + C_1/P_1 \leq \alpha, \tag{5}$$

The lowest operating frequency selected which satisfies modied schedulability test.The operating frequency's and the corresponding voltages are to available on the hardware on which its supposed to be implemented.[5] Similarly we can write the schedulability test for RM as

$$w_i(t) = \textstyle\sum_{k=1}^{i} C_k \lceil \frac{t}{P_k} \rceil, 0 < t \leq P_i$$

$$w_i(t) \leq \alpha t \tag{6}$$

$$t = kP_j, j = 1, ..., i, k = 1, ..., \lfloor \frac{P_i}{P_j} \rfloor$$

if and only if task T_i is RM-schedulable. If $D_i = P_i$, we replace P_i by $min(di, pi)$ in the above expression. Figure 1

```
EDF_test ($\alpha$):
    if ($C_1/P_1 + \cdots + C_n/P_n \leq \alpha$) return true;
    else return false;

RM_test ($\alpha$):
    if ($\forall T_i \in \{T_1, \ldots, T_n | P_1 \leq \cdots \leq P_n\}$
          $\lceil P_i/P_1 \rceil * C_1 + \cdots + \lceil P_i/P_i \rceil * C_i \leq \alpha * P_i$ )
    return true;
    else return false;

select_frequency:
    use lowest frequency $f_i \in \{f_1, \ldots, f_m | f_1 < \cdots < f_m\}$
    such that RM_test($f_i/f_m$) or EDF_test($f_i/f_m$) is true.
```

Figure 1. Static voltage scaling algorithm for EDF and RM schedulers

shows the pusedo code for the static DVFS. Figure 2 shows the sample worst-case execution for statically-scaled EDF and RM scheduling. The example uses the task set in Figure 3, which indicates each task's period and worst-case computation time, and having discrete frequencies available (0.5, 0.75, and 1.0). Figure also shows that statically scaled RM cannot reduce frequency as done by the EDF version. This whole examples is taken from [5].

As the task set passes the schedulability test with computation time is less than the worst case time this mechanism will timely execute the tasks by their deadlines at a scaled voltage. This algorithm does not realize the full potential of energy savings through frequency and voltage scaling[5]. as the actual computation time is much less than worst-case time.

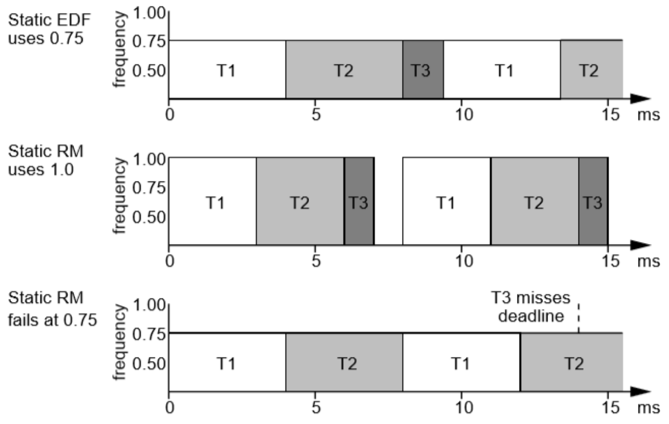


Figure 2. Static voltage scaling example

Task	Computing Time	Period
1	3 ms	8 ms
2	3 ms	10 ms
3	1 ms	14 ms

Figure 3. Example task set, where computing times are specied at the maximum processor frequency

VI. Cycle-conserving RT-DVFS

This mechanism takes advantage of tasks using time less than their worst-case time. During the release of the task, the worst case computation time is considered to calculate the utilization. When the task is completed with the less number of cycles, the processor frequency/voltage is reduced leading to less idle time being wasted. Slack time stealing technique and cycle conserving DVFS are similar in concept but, tasks are run at low frequency instead of completing future tasks.[11]. The schedulability test for this mechanism remains the same as previous. When a task is completed the actual cycles are used to select a lower operating frequency. During the task release, higher frequency is chosen as we don't know the exact computation time so using the worst case time. The pusedo code for Cycle conserving EDF is shown in figure 4 [5].

```
select_frequency():
    use lowest freq. $f_i \in \{f_1, \ldots, f_m | f_1 < \cdots < f_m\}$
    such that $U_1 + \cdots + U_n \leq f_i/f_m$

upon task_release($T_i$):
    set $U_i$ to $C_i/P_i$;
    select_frequency();

upon task_completion($T_i$):
    set $U_i$ to $cc_i/P_i$;
        /* $cc_i$ is the actual cycles used this invocation */
    select_frequency();
```

Figure 4. Cycle-conserving DVS for EDF schedulers

Figure 5 shows the same task set in figure 3 and available frequencies as before, but using actual execution times from figure 6. There are two instances each having different time of execution. We can observe the tasks utilization's mentioned at each task release or completion.

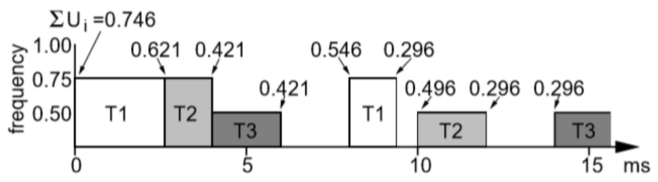


Figure 5. Example of cycle-conserving EDF

Task	Invocation 1	Invocation 2
1	2 ms	1 ms
2	1 ms	1 ms
3	1 ms	1 ms

Figure 6. Actual computation requirements of the example task set (assuming execution at max. frequency)

This algorithm seems to do nothing. However, since multiple tasks are simultaneously put into reduced-utilization state, the total power consumption reduced is signicant. We can use the same method to perform RM based cycle conserving with the schedulibility test shown previously. This agorithm dynamically adjusts the frequency and voltage based on the actual computation time.

VII. Look-Ahead RT-DVFS

Look-ahead RT-DVFS attempts to obtain better energy savings by determining the future computation time needed and deferring the task execution. The cycle-conserving approaches only reduces the operating frequency and voltage based on the actual computation when the task is completed. In contrast, the look-ahead scheme tries to defer as much work as possible, and sets the operating frequency to meet the minimum work that must be done now to ensure all future deadlines are met.In this mechanism the future tasks are forced to run at high frequencies in order to complete all of the deferred task load in time. On the other hand, if tasks tend to use much less than their worst-case computing time allocations, the peak execution rates for deferred work may never be needed, and this heuristic will allow the system to continue operating at a low frequency and voltage while completing all tasks by their deadlines[5].

Continuing with the example used earlier, the working of lookahead RT-DVS EDF algorithm is shown in Figure 7. The goal is to defer work beyond the earliest deadline in the system D_1, so that we can operate at a low frequency at current time. Starting with the task with the latest deadline, T_3.The work is spread between earliest deadline $D+1$ and D_3. This step for T_2, which cannot entirely t between the given epoch after allocating after executing T_3 and reserving capacity for future invocations of T_1.

The operating frequency is determined by all the work allocated before. Once a task has completed, using much less than its specied worst-case execution cycles, the same process is repeated for next tasks to find a suitable lower operating frequency.[5]

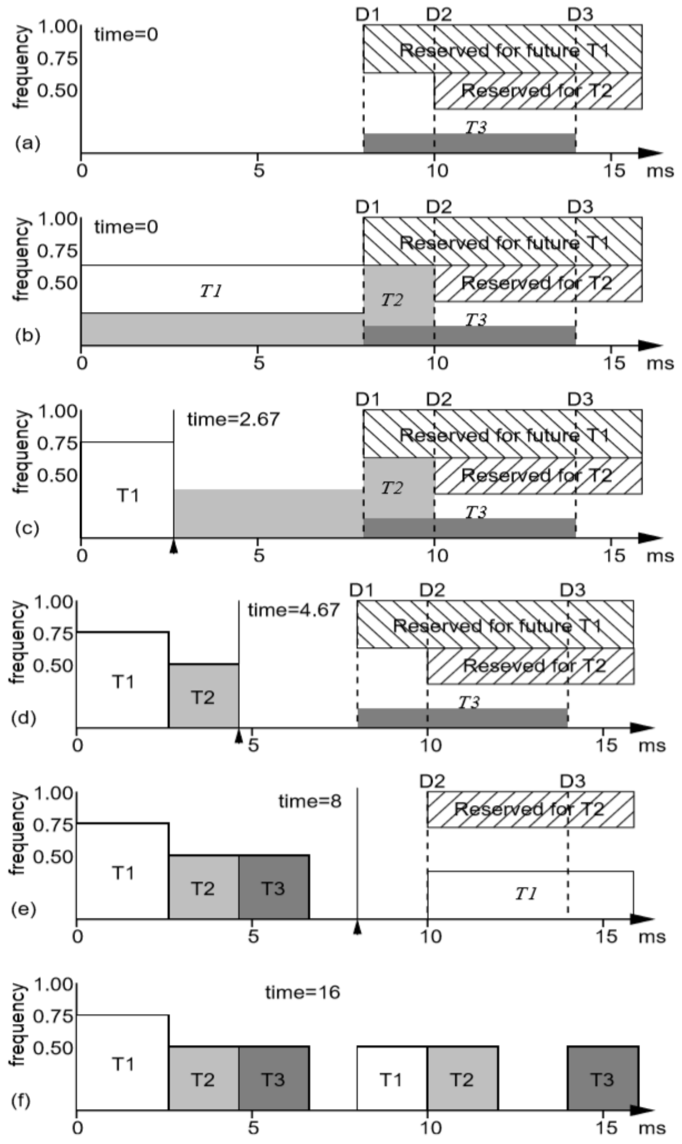


Figure 7. Example of look-ahead EDF: (a) At time 0, plan to defer T3's execution until after D1 (but by its deadline D3, and likewise, try to t T2 between D1 and D2; (b) T1 and the portion of T2 that did not t must execute before D1, requiring use of frequency 0.75; (c) After T1 completes, repeat calculations to nd the new frequency setting, 0.5; (d) Repeating the calculation after T2 completes indicates that we do not need to execute anything by D1, but EDF is work-conserving, so T3 executes at the minimum frequency; (e) This occurs again when T1's next invocation is released; (f) Execution trace through time 16 ms.

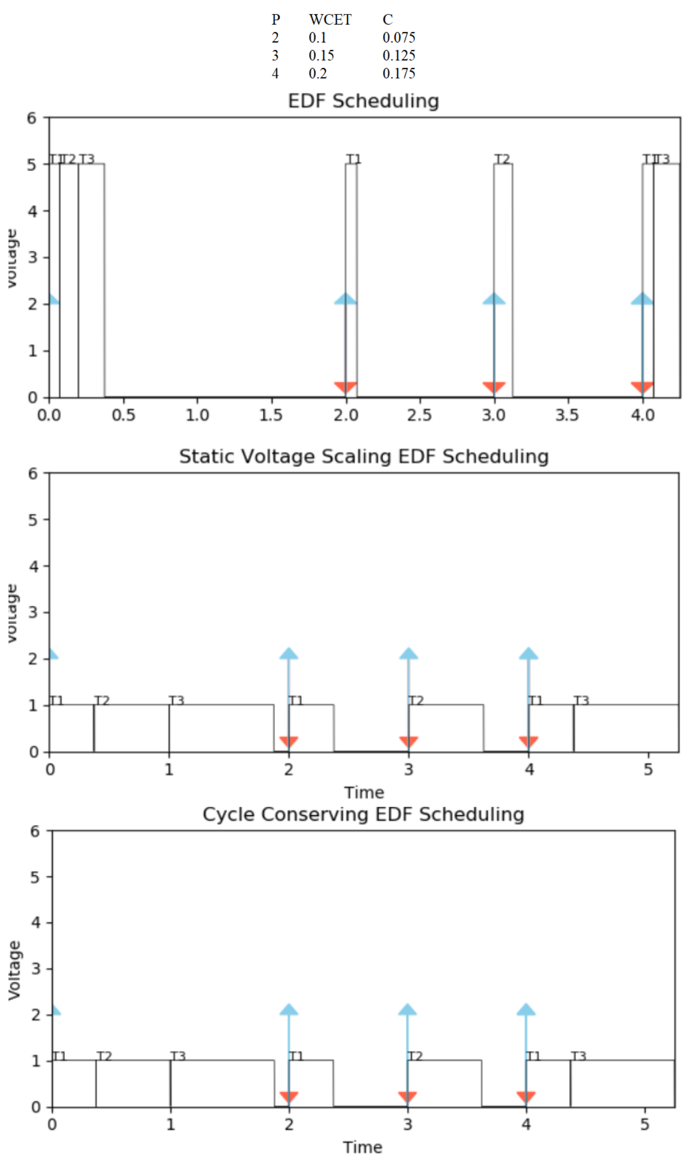
The pesudo code for look-ahead DVFS EDF based algorithm is shown in figure 8. The frequency is selected with same schedulability test for scaled frequency and voltage. When the task is released the work is deferred excluding the task in left and gradually increasing the consideration of task in left while running and finally when the task is completed the all the tasks including in the left are considered for calculating the utilization. Although this algorithm very aggressively reduces processor frequency and voltage, it ensures that there are sufcient cycles available for each task to meet its deadline after reserving worst-case requirements for higher-priority (earlier deadline) tasks[5].

```
select_frequency($x$):
    use lowest freq. $f_i \in \{f_1, \dots, f_m | f_1 < \dots < f_m\}$
    such that $x \leq f_i/f_m$

upon task_release($T_i$):
    set $c\_left_i = C_i$;
    defer();

upon task_completion($T_i$):
    set $c\_left_i = 0$;
    defer();

during task_execution($T_i$):
    decrement $c\_left_i$;

defer():
    set $U = C_1/P_1 + \cdots + C_n/P_n$;
    set $s = 0$;
    for $i = 1$ to $n$,    $T_i \in \{T_1, \dots, T_n | D_1 \geq \cdots \geq D_n\}$
                    /* Note: reverse EDF order of tasks */
        set $U = U - C_i/P_i$;
        set $x = \max(0, c\_left_i - (1 - U)(D_i - D_n))$;
        set $U = U + (c\_left_i - x)/(D_i - D_n)$;
        set $s = s + x$;
    select_frequency ($s/(D_n -$ current_time$)$);
```

Figure 8. Look-Ahead DVS for EDF scheduler)

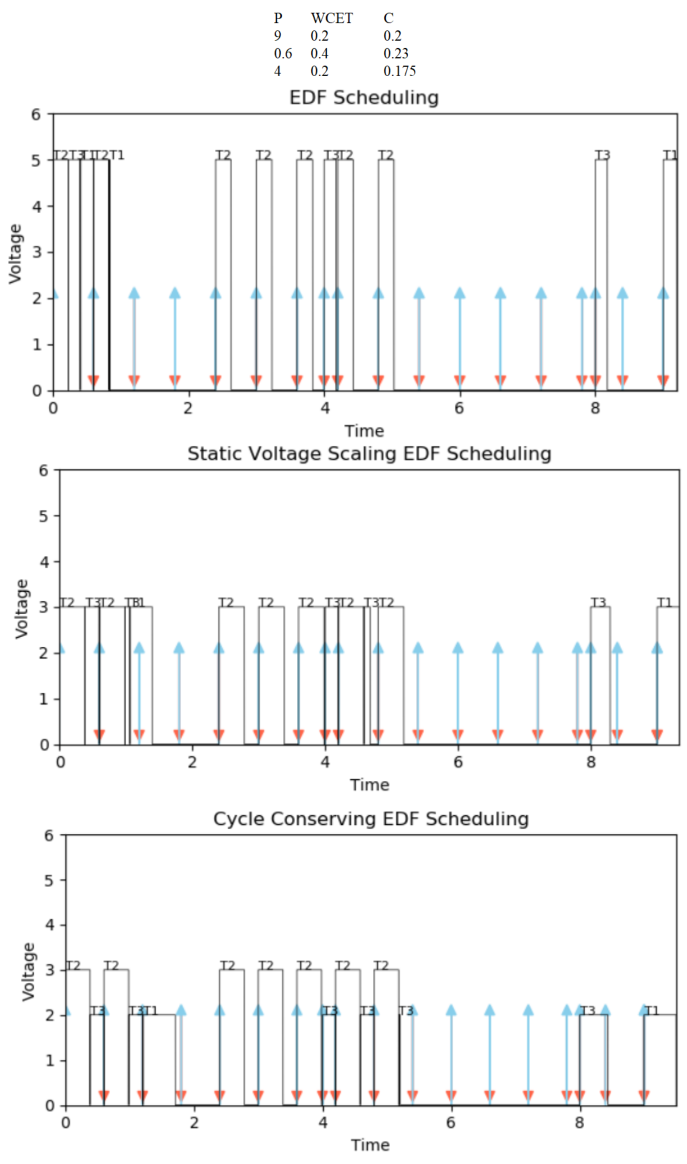
VIII. Simulation

The simulation is done in python. Three algorithms are implemented, EDF scheduling, static voltage scaling EDF based scheduling algorithm, cycle conserving dynamic EDF based scheduling algorithm. The tasks are considered to periodic having period P_i for taskT_i. The input file has inputs specified in format [Period WCET C], where WCET is worst case execution time and C is the actual execution time for tasks.The execution time for all inc-ovations is considered same. Consider the following test case shown in figure 9. We obtain same energy consumption of 0.2 for both staic and cycle conserving method where for normal EDF energy consumed is 1. Example 2 shown in figure 10 has energy consumption of 0.6 for staic and 0.44 cycle conserving method, where for normal EDF energy consumed is 1. The schedule for both the examples are shown below in the figures. The blue arrows indicate the start time of the task and the red indicate the deadlines. There might be some task sets which have energy consumption more for cycle conserving DVFS than the static DVFS implementation when the tasks have more execution time in the 2nd invocation. As the tasks set used are considered having same execution time for each instance no such cases are covered. We can decide on this simulation basis which scheduling algorithm to choose based on the energy consumption of the periodic task sets for online scheduling. The frequency and voltage discrete values used in the simulation are shown in figure 11.



P	WCET	C
2	0.1	0.075
3	0.15	0.125
4	0.2	0.175

Figure 9. Test case 1 where energy consumption for both static and cycle conserving EDF based RT-DVFS are same)



P	WCET	C
9	0.2	0.2
0.6	0.4	0.23
4	0.2	0.175

Figure 10. Test case 2 where energy consumption for static is 0.6 and cycle conserving EDF based RT-DVFS is 0.44)

IX. Further Reading

Further the look-ahead RT-DVFS can be implemented in similar way. Leakage aware algorithms can be studied and implemented. feedback-based mechanisms and control theoretic formulations techniques to predict task execution time and adjust DVFS aggressiveness accordingly[28], [29], [30]. From embedded system architecture perspective, power-aware scheduling techniques can be categorized as uni-processor or multiprocessor scheduling.[1]. Multiprocessor power-aware scheduling techniques focused on distributed systems with precedence relationships among tasks.[22], [23], [24], [25], [26] The adaptive body biasing (ABB) control technique[27].

X. Conclusion

In this report/paper the working and the concept of dynamic voltage scaling techniques such as static, cycel conserving, look-ahead RT-DVFS are studied. The two algorithms are simulated in python and a comparison between the EDF, static EDF and cycle conserving EDF is shown. From the observations we can say that dynamic voltage scaling techniques like cycle conserving and look-ahead are more energy efficient algorithms than the static based DVFS, due to the fact that actual execution time is much less than the worst case execution time, where static uses worst case to compute the operating voltage and dynamic algorithms try to use the actual computation time.

Freq	V
0.36	1
0.56	2
0.72	3
0.89	4
0.97	5

Figure 11. Discrete frequency values corresponding voltage values

References

[1] CPU power dissipation Wikipedia https://en.wikipedia.org/wiki/CPUpowerdissipationClockfrequencies

[2] J. HENKEL and S. PARAMESWARAN DESIGNING EMBEDDED PROCESSORS,A Low Power Perspective p.219-p.290

[3] CMOS Wikipedia https://en.wikipedia.org/wiki/CMOS

[4] Gang Quan Linwei Niu and Xiaobo Sharon Hu Bren Mochocki, Fixed Priority Scheduling for Reducing Overall Energy on Variable Voltage Processors https://www3.nd.edu/ shu/research/papers/rtss04.pdf

[5] Padmanabhan Pillai and Kang G. Shi, Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems http://www.ecs.umass.edu/ece/koren/architecture/RtDVS/rtdvs.pdf

[6] BURD, T. D., AND BRODERSEN, R. W. Energy efcient CMOS microprocessor design. In Proceedings of the 28th Annual Hawaii International Conference on System Sciences. Volume 1: Architecture (Los Alamitos, CA, USA, Jan. 1995), T. N. Mudge and B. D. Shriver, Eds., IEEE Computer Society Press, pp. 288–297.

[7] FLAUTNER, K., REINHARDT, S., AND MUDGE, T. Automatic performance-setting for dynamic voltage scaling. In Proceedings of the 7th Conference on Mobile Computing and Networking MOBICOM'01 (Rome, Italy, July 2001).

[8] GOVIL, K., CHAN, E., AND WASSERMANN, H. Comparing algorithms for dynamic speed-setting of a low-power CPU. In Proceedings of the 1st Conference on Mobile Computing and Networking MOBICOM'95 (Mar. 1995).

[9] GRUIAN, F. Hard real-time scheduling for low energy using stochastic data and DVS processors. In Proceedings of the International Symposium on Low-Power Electronics and Design ISLPED'01 (Huntington Beach, CA, Aug. 2001).

[10] KRISHNA, C. M.,AND LEE, Y.-H. Voltage-clock-scaling techniques for low power in hard real-time systems. In Proceedings of the IEEE Real-Time Technology and Applications Symposium (Washington, D.C., May 2000), pp. 156–165.

[11] LEHOCZKY, J., AND THUEL, S. Algorithms for scheduling hard aperiodic tasks in xed-priority systems using slack stealing. In Proceedings of the IEEE Real-Time Systems Symposium (1994).

[12] LIU, C. L., AND LAYLAND, J. W. Scheduling algorithms for multiprogramming in a hard real-time environment. J. ACM 20, 1 (Jan. 1973), 46–61.

[13] LORCH, J., AND SMITH, A. J. Improving dynamic voltage scaling algorithms with PACE. In Proceedings of the ACM SIGMETRICS 2001 Conference (Cambridge, MA, June 2001), pp. 50–61.

[14] MOSSE, D., AYDIN, H., CHILDERS, B., AND MELHEM, R. Compiler-assisted dynamic power-aware scheduling for real-time applications. In Workshop on Compilers and Operating Systems for Low-Power (COLP'00) (Philadelphia, PA, Oct. 2000).

[15] PERING, T.,AND BRODERSEN, R. Energy efcient voltage scheduling for real-time operating systems. In Proceedings of the 4th IEEE Real-Time Technology and Applications Symposium RTAS'98, Work in Progress Session (Denver, CO, June 1998).

[16] PERING, T., AND BRODERSEN, R. The simulation and evaluation of dynamic voltage scaling algorithms. In Proceedings of the International Symposium on Low-Power Electronics and Design ISLPED'98 (Monterey, CA, Aug. 1998), pp. 76–81.

[17] PERING, T., BURD, T.,AND BRODERSEN, R. Voltage scheduling in the lpARM microprocessor system. In Proceedings of the International Symposium on Low-Power Electronics and Design ISLPED'00 (Rapallo, Italy, July 2000).

[18] POUWELSE, J., LANGENDOEN, K., AND SIPS, H. Dynamic voltage scaling on a low-power microprocessor. In Proceedings of the 7th Conference on Mobile Computing and Networking MOBICOM'01 (Rome, Italy, July 2001).

[19] POUWELSE, J., LANGENDOEN, K., AND SIPS, H. Energy priority scheduling for variable voltage processors. In Proceedings of the International Symposium on Low-Power Electronics and Design ISLPED'01 (Huntington Beach, CA, Aug. 2001).

[20] SWAMINATHAN, V.,AND CHAKRABARTY, K. Real-time task scheduling for energy-aware embedded systems. In Proceedings of the IEEE Real-Time Systems Symp. (Work-in-Progress Session) (Orlando, FL, Nov. 2000).

[21] WEISER, M., WELCH, B., DEMERS, A., AND SHENKER, S. Scheduling for reduced CPU energy. In Proceedings of the First Symposium on Operating Systems Design and Implementation (OSDI) (Monterey, CA, Nov. 1994), pp. 13–23.

[22] F. Gruian and K. Kuchcinski. Lens: Task scheduling for low-energy systemsusingvariablesupplyvoltageprocessors. ASPDAC,pages449–455, 2001.

[23] J. Liu, P.H. Chou, N. Bagherzadeh, and F.J. Kurdahi. Power-aware scheduling under timing constraints for mission-critical embedded systems. DAC, pages 840–845, 2001.

[24] J. Luo and N. Jha. Power-conscious joint scheduling of periodic task graphs and aperiodic tasks in distributed real-time embedded systems. ICCAD, pages 357–364, 2000.

[25] M.T. Schmitz, B.M. Al-Hashimi, and P. Eles. Energy-efcient mapping and scheduling for dvs enabled distributed embedded systems. DATE, pages 321–330, 2002.

[26] Y.Zhang,X.Hu,andD.Chen. Taskschedulingandvoltageselectionfor energy minimization. DAC, pages 183–188, 2002.

[27] Alexandru Andrei1, Petru Eles1, Zebo Peng1, Marcus Schmitz2, and Bashir M. Al-Hashimi2 Voltage Selection for Time-Constrained Multiprocessor Systems on Chip

[28] DUDANI, A., MUELLER, F.,AND ZHU, Y. Energy-conserving feedback EDF scheduling for embedded systems with real-time constraints. InACMSIGPLANJointConferenceLanguages,Compilers,andToolsfor EmbeddedSystems(LCTES'02)andSoftwareandCompilersforEmbedded Systems (SCOPES'02) (June 2002), pp. 213–222.

[29] LU, Z., HEIN, J., HUMPHREY, M., STAN, M., LACH, J.,AND SKADRON, K. Control-theoretic dynamic frequency and voltage scaling for multimedia workloads. In CASES '02: Proceedings of the 2002 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (2002), pp. 156–163.

[30] VARMA,A.,GANESH,B.,SEN,M.,CHOUDHARY,S.R.,SRINIVASAN, L., AND JACOB, B. A control-theoretic approach to dynamic voltage scaling. In Proceedings of International Conference on Compilers, Architectures, and Synthesis for Embedded Systems (CASES 2003) (Oct. 2003), pp. 255–266.