Design of Digital Control Systems with Dynamic Voltage Scaling

Hyung Sun Lee and Byung Kook Kim

*Department of Electrical Engineering and Computer Science
Korea Advanced Institute of Science and Technology, Daejeon, Republic of Korea
{hslee@rtcl, bkkim@ee}.kaist.ac.kr*

Abstract

For real-time computer-controlled systems, control performances of tasks as well as energy consumption of overall system must be optimized. A control task does not have a fixed period but a range of periods in which control performance varies. Hence when more than one control tasks are scheduled on a single processor, an optimization problem appears. Furthermore, when an energy saving technique such as dynamic voltage scaling is used, its properties affect the control performance. Using a performance index that involves control performance and energy consumption, a static solution is proposed to obtain optimal processor speed and a set of periods for given control tasks in $O(k)$. Also, a dynamic solution is proposed to utilize system services of real-time operating systems to overcome unavoidable deficiencies of the static solution and to further reduce the energy consumption of the overall system. The performances of proposed solutions are revealed via simulation studies.

1. Introduction

Many embedded systems, such as mobile robots, are real-time computer-controlled systems. One characteristic of such systems is that controllers for various parts are often digitally implemented as control tasks and many research efforts have been put on improving the performance of control tasks. Since embedded systems are often run with limited energy source, another key issue related to such systems is reducing the power consumption.

Often digital control system design is separated into two phases: design of an analog controller and its digital implementation. Design of an analog controller is straightforward. For each target system, one can design an

analog controller of any form, whichever results in desired control performance. For digital implementation of the controller, current states of the system are sampled using analog-to-digital (A/D) converters, control output is computed using microcontrollers or embedded processors, and control output is generated using digital-to-analog (D/A) converters.

Although one may optimally design an analog controller for a control system, performance of its digital implementation may not be as good as expected. Control outputs of analog controllers are continuously updated, but those of digital controllers are periodically updated depending on execution frequencies of implemented control tasks. As control tasks are executed less frequently, outputs of digital controllers deviate from that of analog controllers. Moreover, when there are multiple control tasks implemented to run on a single processor, scheduling issue occurs. Since only one task can occupy processor at one time, execution of one control task inevitably affects execution of other control tasks.

There have been many research works related to implementations of control tasks and their performances. The issue of task design in relation to system performance was addressed by Gerber et al. [1]. Lock and others also studied task scheduling and system performance [2]. Seto et al. studied relationship between task frequency and system performance using performance index (PI) approach [3, 4]. However, it was normally assumed that worst-case execution time of tasks are fixed, which is true for most of systems today with fixed algorithms. But for systems with complex algorithms it is not true. Palopoli et al. studied optimal synthesis of robust digital controllers on a single processor with real-time scheduler, but the goal of optimization was focused on robustness of controllers [19].

Power consumption is another important issue in design of embedded systems and many research efforts has been put on this issue recently. One of the results of such

efforts is dynamic voltage scaling (DVS) technique, which involves dynamically adjusting voltage and frequency of CPU to trade off performance of the processor with amount of power it consumes. Microprocessors, these days, are mostly designed with CMOS transistors and DVS technique uses properties of CMOS transistors. When an inverter is switched on or off, a gate delay occurs, and this delay is inversely proportional to the supply voltage used. With a higher supply voltage, gate delays are reduced and processors can run at higher speed. For an inverter, power consumption is proportional to the square of the supply voltage and to the switching frequency. Hence, when a processor's speed is reduced its performance degrades linearly but power consumption is reduced by powers of three. Pouwelse et al. experimentally demonstrated such behavior on a LART board using SA-1100 processor [10].

Scheduling techniques proposed for DVS systems normally target real-time systems, in which timing constraints are given to tasks, and try to adjust CPU speed without violating their deadline constraints. First DVS scheduling algorithm was a static off-line scheduling algorithm provided by Yao et al. [14]. Only worst-case execution times were considered. Aydin et al. proposed a dynamic on-line scheduling algorithm that uses unused execution times of tasks and an aggressive scheduling algorithm that statistically predicts tasks execution behavior to further reduce the speed of processors [15]. Pillai and Shin proposed a DVS scheduling algorithm and successfully implemented in Linux kernel operating on a notebook computer with AMD K6 processor [16]. There are many other DVS scheduling algorithms proposed for hard real-time environment.

Noting many advantages of DVS, embedded systems are likely to incorporate DVS technique near future. A processor with DVS scheduler has dynamically varying performance which is not notified to scheduled tasks. If control tasks are scheduled in such systems, assumptions of fixed worst-case execution times are no longer true.

In this paper, we formulate a control system design problem considering the control performance as well as energy consumption, and suggest solutions for static and dynamic cases, whose performances are verified via simulation and analysis.

This paper is organized as follows. Section 2 explains more about properties of control systems and DVS technique, and we formulate a problem that occur when these two concepts coexists. A static solution for the suggested problem is proposed in Section 3 and a dynamic solution is proposed in Section 4. Section 5 shows simulation and analysis verifying the proposed solutions. Concluding remarks are drawn in Section 6.

2. Control Systems with DVS

In this section, we describe properties of control systems and identify those properties affected by use of DVS techniques. Then we formulate a task scheduling problem optimizing the performance index with DVS.

2.1. Digital Control System

A prototypical digital control system with feedback is shown in Figure 1. For the given plant, an appropriate control law can be designed which controls the plant satisfying a given constraints. We assume that such analog controller is given.

Figure 1. A typical digital control system.

For each analog controller, the corresponding digital controller can be derived and implemented, as a control task, which is normally composed of three regions: sampling, computation, and actuation. At the beginning of each execution, a control task, τ_i , first samples current states of the plant using A/D converters, computes the value of control output according to the control law, and generates the control output using D/A converters. Assuming that A/D and D/A conversion take negligible amount of time, computation of control output takes up most of the execution time. This time interval between A/D and D/A is called the *computational delay*. Longest possible time needed for execution of the task is called the worst-case execution time and denoted by e_i .

Unlike other real-time tasks, a control task has a range of period in which it can be scheduled. As a control task is scheduled more frequently, its control output will be closer to that of its analog counterpart. But each cycle of execution takes up a finite amount of time. Hence it is clear that even when a single control task is scheduled on a processor, it cannot be scheduled with a period less than its worst-case execution time. For multiple tasks, lower bound of each task's period is determined by the schedulability constraint: Sum of utilization of each control tasks cannot exceed the processor's schedulable utilization bound. Upper bound of each task's period is

determined by the stability constraint of each plant. It is known that the sampling interval of a digital controller affects the control delay of the system [6], and delay reduces phase margin of overall system. Therefore, as the period of each task is increased, its control output diverges from that of analog counterpart and becomes unstable at some point.

Performance of a control system is often expressed with a suitable *performance index* (PI). PI is often a target for optimization from which a control algorithm is derived. For example, a mobile robot following a reference trajectory may have a performance index related to amount of error in its trajectory. For a system that runs on limited energy source, power consumption may be an important part of its performance index. The form of performance index used for the rest of this paper is a standard linear quadratic (LQ) form:

$$J = \frac{1}{2} \int_0^\infty (e^T Q e + u^T R u) dt \quad (1)$$

where Q is a positive semidefinite matrix, R is a positive definite matrix, $e(t) = r(t) - x(t)$ is the state error, and $u(t)$ is the control output. The quadratic term $e^T Q e$ emphasizes minimizing the magnitude of weighted square error, and the other quadratic term $u^T R u$ minimizes the magnitude of weighted square control input.

2.2. DVS Schedulers

A number of processors support dynamic switching of its clock frequencies. Intel's *SpeedStep* technology supports two different frequencies to be used under different situations. Lower speed is used to reduce power consumption when system is battery operated, and faster speed is used to enhance system performance when AC adapter is connected. Transmeta's Crusoe processor has seven different frequencies for which different core voltage can be used [7]. AMD's K6 processor with its *PowerNow!* technology provides eight different choice of core frequency [8].

Processors mentioned above have limitations in voltage scaling. Crusoe processor allows only hardware method for voltage scaling, and K6 processor relies on notebook manufacturers to implement voltage scaling through BIOS functions. First software approach for voltage scaling has been proposed by Pouwelse [10]. He used Intel StrongARM processor (SA-1100) which supports sixteen different choice of core frequency [9], along with a programmable voltage regulator and experimentally verified that necessary core voltage for SA-1100 processor is proportional to core frequency used. One notable

property of processors with DVS support is that all of them support only fixed set of core frequencies. Such property comes from the fact that there is no practical method to have a PLL generating clocks of infinite frequencies.

For systems with DVS, execution time of the same code differs for each processor speed. Therefore, the notion of execution time is replaced with the notion of *execution cycle*. Execution cycle is the number of processor clock cycles needed to execute a portion of code. The execution cycle divided by the processor speed in Hz is the execution time in seconds. Given a task τ_i with period p_i and execution cycle c_i , execution time would be $e_{i,k} = c_i / s_k$ under normal processor speed s_k .

Basic concept of DVS scheduler is that whenever processor is underutilized, system can reduce processor speed just enough to guarantee all timing constraints of scheduled tasks. A simple example is shown in Figure 2. Two tasks, whose task properties (e_i, p_i) are given as (1, 4) and (2, 8), are scheduled with an EDF scheduler. Total utilization of the processor would be $U = 0.5$. Since the maximum allowed utilization for EDF scheduled tasks is $U_{\max} = 1$, a schedule would still be feasible if execution times were doubled. If these tasks were scheduled with a DVS scheduler, remaining utilization of 0.5 would be fully used by decreasing processor speed by half: Execution times of tasks would be doubled, but all tasks would still meet their deadlines. As processor speed is decreased, energy consumption of the system is reduced.

Figure 2. Task schedule under EDF and DVS.

Dynamic DVS schedulers further decrease processor speed by utilizing idle times that appear when actual execution times are less than worst-case execution times. Aggressive DVS schedulers statistically predict actual execution times and utilize any idle times that will appear in the near future. Still all these DVS schedulers follow

the same idea that whenever the processor is underutilized, processor speed is reduced to make $U \cong U_{\max}$.

2.3. Problem Statement

When control systems are implemented on a DVS scheduler, problem of optimizing the performance index with control performance as well as energy consumption occurs.

This problem can be stated as follows: Given

- i) a processor with fixed set of processor speed $S = \{s_0 < s_k < s_K\}$, and power consumption given by a strictly increasing function $g(s)$
- ii) a set of real-time control tasks $T = \{\tau_1, \dots, \tau_n\}$, with ranges of allowable periods $r_{i,\min} \leq r_i \leq r_{i,\max}$, execution cycles c_i , and execution times $e_{i,k} = c_i / s_k$
- iii) a DVS scheduler which schedules tasks with EDF policy

Determine the period of each control task that minimize the overall system performance J given by,

$$J = w_c J_c + w_e J_e \quad (2)$$

$$\text{where } J_c = \frac{1}{2} \int_0^T (e^T Q e + u^T R u) dt \quad (3)$$

$$\text{and } J_e = \int_0^T (C v_{DD}^2 s) dt \quad (4)$$

v_{DD} and s are time varying functions of processor's core voltage and speed, and C is a multiplication factor. w_c and w_e are weighting factors. The lower the performance index, the better the system performance.

3. Static Solution

Assuming static parameters (constant task periods, and constant processor speed), a static optimal solution to the problem in Section 2.3 can be found offline. However, since all N control tasks have independent ranges of allowable periods and K different processor speed exists, a full search using computer simulation has complexity of $O(n^{N+1})$ and may take indefinite amount of time as N is increased. A practical solution with much less complexity is proposed in this section.

Given a set of periodic control tasks, the same task schedule is repeated for each hyperperiod (least common multiples of task periods). The amount of idle times that occur are also the same for every hyperperiod. Hence we

may expect that the processor speed would set to a value that makes the processor utilization close to $U_{\max}$.

The performance index in eq. (2) is a function of periods of each control tasks, as well as a function of processor speed. Note that the control performance term in eq. (3) is a function of periods of each control tasks, and energy consumption term in eq. (4) is a function of processor speed only. Hence the problem of optimizing task periods under DVS can be broken down into two problems: a problem of optimizing periods of control tasks to fully utilize the processor's computing resource (*optimal periods problem*) and a problem of optimizing the overall system performance for different processor speeds a given set of tasks (*optimal processor speed problem*).

Optimal periods problem has been previously studied by Seto et al. [3]. Given execution times and variable periods of tasks, he proposed a method of calculating the optimal periods of tasks in the sense that the overall performance is maximized. The same solution can be used for this problem. For each processor speed s_k where $k = 1, \dots, K$, execution times of each tasks can be found. For each task τ_i , performance index with digital control task J_i can be measured or simulated on $[0, nT]$ as

$$J_i(f) = \sum_{k=0}^{n-1} \int_{mT}^{(m+1)T} [R(u(mT))^2 + (y(t) - r(t))^2] dt \quad (5)$$

where T is the period of its control task, f represents the sampling frequency, which is equal to $1/T$.

The performance index J increases exponentially as the sampling frequency decreases: As the period of digital control task is increased, its control output deviates from ideal form. Using least-square fitting approach, we can obtain the incremental performance index $\Delta J_i^*(f)$ as

$$\Delta J_i^*(f) = |J_i^*(f) - J^*| = \alpha e^{-\beta f} \quad (6)$$

where J^* is the performance of continuous-time control.

Using values of $\alpha_{i,k}$ and $\beta_{i,k}$ obtained for all tasks τ_i for $i = 1, \dots, N$ at processor speed s_k , a unique optimal set of sampling frequencies, $f_{1,k}^*, \dots, f_{n,k}^*$, can be calculated as proposed in [3]. This solution results in the optimal periods of control tasks at processor speed s_k , that fully utilizes computing resource of the processor.

Optimal processor speed problem is straightforward from this point. Now that the optimal periods have been found, a full search using computer simulation has complexity of $O(k)$ only.

In addition to a scheduled set of periodic control tasks,

presence of one or more aperiodic tasks, however, would not affect the stability of control systems or degrade the control performance with the following reasoning. From other researches on scheduling of control systems, we see that sample-to-actuation delay called *latency* also affects the control performance of the system [5, 11]. Control performance degrades with longer latency, but reduction in latency always improves control performance. If aperiodic tasks were added on the system with designed control tasks, DVS scheduler would increase the processor speed to accommodate for increased request for processor resource. An increase in the processor speed results in reduced computation times for each code, which results in reduced sample-to-actuation latency.

4. Dynamic Solution

Static solution in Section 3 uses task parameters in their worst possible situations, e.g. worst-case execution times. Some dynamic DVS schedulers can compensate for such source of inefficiency, but some conservativeness is required.

When digital controllers are designed, they are designed to meet the given requirements at all times, even in the presence of disturbance and error. However, when a target plant does not experience any disturbance and remains in steady-states, one may argue that the optimal task periods are over-designed. Hence there may be a way to further reduce energy consumption without jeopardizing the control performance.

Current DVS schedulers do not collect information about plant states and consider them for scheduling. But most real-time operating systems (RTOSs) provide a way for tasks to change their execution periods at runtime, e.g. RTLinux has `rtl_task_make_periodic` [17] and Real-Time Application Interface (RTAI) has `rt_task_make_periodic` [18]. Therefore we suggest a dynamic solution with a code for each control task with dynamic period as shown in Figure 3.

Each task changes its own period according to the current state of the plant. When the plant is in transient state, the control task reduces its execution period to request more processing resource for its control performance. When the plant is in steady state, the control task relaxes its control period to release processing resource without harming its control performance. By releasing the processing resource, DVS scheduler can further decrease the processor speed and reduce its energy consumption.

For this purpose, each task must be assigned two periods: *fast period*, $r_{i,f}$ and *slow period*, $r_{i,s}$. With the fast period, control task must be able to control the plant

well enough to meet the given requirements. Optimal periods found in Section 3 can well serve as the fast period or one might want to modify weighting factors in eq. (2) to put more weight on control performance. Slow periods will be used during steady states, hence the maximum allowable period $r_{i,max}$ for target plant can be used as the slow period.

```

initialize_variables;
LOOP
    sample_current_states;
    if ( (previous_state==transient)
    && (current_state==steadystate) )
        set_period(slow_period);
    if ( (previous_state==steadystate)
    && (current_state==transient) )
        set_period(fast_period);
    calculate_the_control_law;
    generate_the_control_output;
    wait_until_the_next_period;
END

```

Figure 3. Control task with dynamic period.

5. Simulation and Analysis

5.1. TrueTime Simulator

Two solutions proposed in Sections 3 and 4 were verified using MATLAB along with *TrueTime* simulator. *TrueTime* is a toolbox for real-time and control system developed in Lund Institute of Technology [12, 13]. It enables event-based co-simulation of a multitasking real-time kernel with controller tasks and controlled plants. *TrueTime* supports rate-monotonic (RM), deadline-monotonic (DM), fixed-priority (FP) and earliest deadline first (EDF) policies, but it does not support any DVS schedulers. So a simple DVS scheduler was simulated by calling a subroutine that calculates processor speed and modifies task execution times accordingly at each schedule point.

For target plants, three different inverted pendulums were modeled. Dynamics of three plants are identical, except that their natural frequency have ratio of 1:2:4. An inverted pendulum has two links, their angles θ_1 , θ_2 and angular velocity $\dot{\theta}_1$, $\dot{\theta}_2$ are internal states x . For simplicity of notation, θ_1 is referred to as y . Each plant has one control input u and four output states. Hence 12 outputs in total are connected to inputs of a digital computer for three controllers as shown in Figure 4.

5.2. Simulation Result for Static Solution

Using the static solution proposed in Section 3, the *optimal periods problem* can be solved for simulation environment shown in Figure 4. The optimal period sets obtained for each processor speed s_k is shown in Figure 6. As the processor speed is lowered, execution of each task code takes more time and periods of tasks must be increased to meet the schedulability condition.

To obtain the energy consumption of the processor, a StrongARM SA-1110 processor board with a programmable voltage regulator was implemented. Block diagram of the testbed is shown in Figure 5. Overall layout of this DVS system is similar to the one proposed in [10], except that this system uses SA-1110 processor instead of SA-1100 processor. Voltage-frequency relation of SA-1110 processor was measured as shown in Figure 7. With this information, energy consumption of the processor at each processor speed can be directly calculated with eq. (4). With weighting factors given as $w_C = 100$ and $w_E = 1$, overall system performance index can be obtained using eq. (2). Results are shown in Figure 8.

Figure 4. Overview of simulation environment.

Figure 5. Block diagram of implemented testbed.

Figure 6. Task periods for each processor speed.

Figure 7. Speed-Voltage relation of SA-1110.

Figure 8. Performance index vs. processor speed.

Processor speed at which performance of overall system is optimized appears at 132.7MHz with $J \approx 274$. Note, however, that optimized processor speed would have appeared at lower point, if weighting factor w_E had been increased. Weighing factors w_C and w_E represent importance of control performance and energy consumption in the overall system performance.

5.3. Simulation Result for Dynamic Solution

To demonstrate the effectiveness of online solution proposed in Section 4, impulsive disturbance was applied to each plant at 0.2ms, 0.4ms and 0.6ms respectively as shown in Figure 9. Task periods, $r_{i,f}$ and $r_{i,s}$, used for the simulation are shown in Table 1.

Table 1. Task Parameters for Simulation

	$r_{i,f}$	$r_{i,s}$
Task 1	3ms	5ms
Task 2	5ms	10ms
Task 3	7ms	20ms

Figure 9. Impulse response of three plants.

If the three plants were controlled with fixed periods $r_i = r_{i,f}$ found in Section 5.2, resulting control performance J_C would be 5.61 and energy consumption J_E would be 277.93. By allowing each control task to dynamically switch its execution period between $r_{i,f}$ and $r_{i,s}$ according to state of each plant, control performance J_C deteriorates a little to the value of 6.83, but energy consumption J_E improves significantly as shown in Figure 10. The maximum energy consumption from 0.6ms to 1ms (about 20% of time span) is equal to

that with static processor speed, and about half the energy is consumed during 80% of the time, which greatly contributes in energy saving. If target plants stay in their steady states longer, performance of dynamic solution outmatches that of static solution.

Figure 10. Variation in energy consumption.

Simulation result for dynamic solution in Figure 10 shows a glitch at time ~ 8.5 ms. Such glitch occurs because the control task determines presence of disturbance by comparing current control error with given threshold value. In the worst case, if disturbance near the threshold value is continuously applied, processor frequency may chatter between two values. However, such misbehavior can be safely avoided by use of hysteresis function for determining presence of disturbance.

6. Conclusion

Solutions for static and dynamic cases are suggested for digital control tasks to optimize the performance index that involves control performance and energy consumption under DVS environment.

Assuming that a simple DVS scheduler exists and the proposed performance index is simply a weighted sum of control performance and energy consumption, whenever a number of controllers, as well as weighting factors for performance index are given, the optimal set of periods for the control tasks can be found in $O(k)$ with the proposed static solution, which guarantees the stability of target plants even when aperiodic tasks are added. When the RTOS provides a system call to change a task's own period, which is true for most of RTOSs, we proposed a dynamic solution which can further reduce the energy consumption of the processor, thus improving the overall

performance of the system considerably.

Future work will address on the issue of control system and DVS environment including design of a DVS scheduler that takes into account more properties of control tasks.

References

- [1] R. Gerber, S. Hong, and M. Sabsena, "Guaranteeing end-to-end timing constraints by calibrating intermediate processes," IEEE Real-Time Systems Symposium, 1994
- [2] C. D. Locke. *Best-effort decision making for real-time scheduling*. Ph.D Thesis, Computer Science Department, Carnegie-Mellon University, 1986.
- [3] Danbing Seto, John P. Lehoczky, Lui Sha, "Task Period Selection and Schedulability in Real-Time Systems," IEEE Real-Time Systems Symposium, 1998.
- [4] Danbing Seto, John P. Lehoczky, Lui Sha, "Task Period Selection and Schedulability in Real-Time Systems." IEEE Real-Time Systems Symposium, 1998.
- [5] Byung Kook Kim, "Task Scheduling with Feedback Latency for Real-Time Control Systems," 5th Int. Conf. on Real-Time Computing Systems and Applications, pp. 37-41, Hiroshima, Japan, Oct. 1998.
- [6] G. F. Franklin, J. D. Powell, and M. L. Workman. *Digital Control of Dynamic Systems*, Addison-Wesley, 1998.
- [7] Transmeta Crusoe Processor. <http://www.transmeta.com/crusoe/index.html>
- [8] AMD K6 Processor. <http://www.amd.com/us-en/-Processors/ProductInformation/>
- [9] Intel SA-1100 Processor. http://www.intel.com/-design/pca/applicationsprocessors/1110_brf.htm
- [10] J. Pouwelse, K. Langendoen, and H. Sips, "Dynamic Voltage Scaling on a Low-Power Microprocessor," 7th ACM Int. Conf. on Mobile Computing and Networking (Mobicom), pp. 251-259, Rome, Italy, July 2001.
- [11] Byung Kook Kim, "Control Latency for Task Assignment and Scheduling of Multiprocessor Real-Time Control Systems," International Journal of Systems Science, vol. 30, no. 1, pp. 123-130, Jan. 1999.
- [12] A. Cervin, D. Henriksson, B. Lincoln, J. Eker, K. Arzen, "Analysis and Simulation of Controller Timing," IEEE Control Systems Magazine, vol. 23, no. 3, pp. 16--30, 2003.
- [13] A. Cervin, "The Real-Time Control Systems Simulator Reference Manual", Report ISRN LUTFD2 / TFRT-7592-SE, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, April 2000.
- [14] F. Yao, A. Demers and S. Shenker, "A Scheduling Model for Reduced CPU Energy," IEEE Annual Foundations of Computer Science, pp. 374-382, 1995.
- [15] H. Aydin, R. Melhem, D. Mosse, and P. M. Alvarez, "Dynamic and Aggressive Scheduling Techniques for PowerAware Real-Time Systems," In Proceedings of IEEE RealTime Systems Symposium, December 2001.
- [16] P. Pillai and K. G. Shin, "Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems," In Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP-01), pages 89- 102, Banff, Canada, October 21-24 2001.
- [17] Victor Yodaiken. *The RTLinux Manifesto*. <http://www.fsmlabs.com/articles/archive/rmanifesto.pdf>
- [18] *DIAPM RTAI Programming Guide 1.0*, <http://www.aero.polimi.it/~rtai/documentation/>, 2000.
- [19] L. Palopoli, C. Pinello, A. S. Vincentelli, L. Elghaoui, and A. Bicchi, "Synthesis of Robust Control Systems under Resource Constraints." Lecture Notes in Computer Science, proceedings of the Hybrid Systems: Computation and Control, Stanford USA, March 2002.