VOLTAGE-CLOCK SCALING AND SCHEDULING FOR ENERGY-CONSTRAINED REAL-TIME SYSTEMS

By

YOONMEE DOH

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2003

To my son Minjae, my husband Daeyoung, and my families, with love

ACKNOWLEDGMENTS

I would like to thank many individuals for the care and support they have given me during my doctoral study. First of all, I would like to express my gratitude to Professor Jih-Kwon Peir. As my advisor, he has provided me with guidance, insightful comments, and suggestions. I wish to extend my thanks to my main advisor, Professor Yann-Hang Lee at the CSE Department of Arizona State University, not only for his invaluable academic guidance but also for his strong support, encouragement, and his caring ways through the years he has supervised my research. I would also like to thank Professor C. Mani Krishna at University of Massachusetts for his priceless advice. My appreciation also goes to the professors on my supervisory committee, Randy Y. Chow, Douglas D. Dankel, Richard Newman, and Paul W. Chun, for their valuable and constructive advice.

I would also like to thank former and current members of the Real-Time Systems Research Group at the University of Florida and Arizona State University for their invaluable help, friendship, and discussion. Thanks also go to the CSE Department of Arizona State University for giving me an opportunity to visit and to work as a visiting research scholar and for providing administrative support during my stay.

I am also deeply grateful to my middle school teacher, Jongyee Choo at Sungwha High School. Without his continued encouragement and moral support, I could never be what I am. My thanks should also go to the former director of ETRI, Professor Munkee Choi, at Information and Communications University and the current members of KCISE at the University of Florida for their help with all the details needed at UF to complete a degree during my stay at ASU.

Finally, I would like to give special thanks to the people who are most dear to me, my dear mother, Seeok Han, my brothers Jinwoong, Youngjin, Younghyo, and my sister, Sunmee, to

whom I owe everything. Their blessing, guidance, and love made me what I am today. I am also grateful to my parents, my sisters, and my brothers-in-law since they have always encouraged me in my studies and believed in me. Many thanks go to my various friends near and far, for hanging in there with me. Especially, I thank Wendy Tiedemann and Okehee Goh for their endless support. My final acknowledgements go to my wonderful husband, Daeyoung, and my lovely son, Minjae, for their continued love and support given to me until now.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

VOLTAGE-CLOCK SCALING AND SCHEDULING FOR ENERGY-CONSTRAINED REAL-TIME SYSTEMS

By

Yoonmee Doh

May 2003

Chairman: Jih-Kwon Peir
Major Department: Computer and Information Science and Engineering

Nowadays, exponentially increasing demands for portable devices and more sophisticated and intelligent functionalities in embedded applications make processors much power-hungrier. Due to a limited energy budget, low-power computing at a relatively low level of power consumption or power-aware computing with the knowledge of power source plays important role in extending systems' lifetime. In the dissertation, we study how to reduce power consumption using a low power voltage-clock scaling scheme in real-time systems that require strict time and energy constraints. The property of power-delay tradeoff has significant impact on the schedulability as well as on the performance of the systems.

First, we focus on dynamic reclaiming of early released resources in earliest deadline first (EDF) scheduling using voltage-clock scaling. In addition to a static voltage assignment, we propose a new dynamic-mode assignment, which has a flexible voltage mode setting at run-time, enabling much larger energy savings. Using simulation results and exploiting the interplay among power supply voltage, frequency, and circuit delay in CMOS technology, we find the optimal two-level voltage settings that minimize energy consumption. Next, we study battery-driven real-

time systems that jointly schedule hard periodic tasks and aperiodic (sporadic) tasks, whose power sources are bounded in a feasible range decided by a set of tasks. Therefore, reducing power consumption is not the only objective of task scheduling in this case. To make the most of the available energy budget, an effective energy-sharing scheme is proposed using two-mode voltage-clock scaling-EDF scheduling.

Based on the proposed energy allocation model for energy-constrained real-time systems, a dual-policy dynamic scheduling method is proposed not only for faster responsiveness but also for reducing power consumption. The feature of the approach is an intermixing of two schedules; the one consumes minimum energy and the other the worst-case energy, respectively, leading to much longer lifetime under a bounded energy budget. Lastly, we build a scaling mechanism that optimizes energy saving and responsiveness to the aim of the system's performance. Fully utilizing the property in sharing constrained-energy and processor's capacity in total bandwidth server, we adjust total power consumption and average response time of aperiodic tasks by simply controlling the utilization of periodic tasks.

CHAPTER 1
INTRODUCTION

1.1 Low-power and Power-aware Real-Time Systems

In recent years, the rapidly increasing popularity of portable battery-operated computing and communication devices have motivated research efforts toward low power and energy consumption. Battery life is the primary constraint for energy-constrained systems such as personal mobile phones, laptop computers, digital cameras, and personal digital assistants (PDA's) [1]. On the other hand, limiting power is also important in the field of general or high-performance systems. In the report of examination of electricity usages in U.S., total power use by office equipment and network equipment is about 74 TWh per year, which is about only 2% of total electricity use in the U.S., and power management currently saves 23 TWh/year [2]. An exponential growth of Internet usage also requires power-efficient server farms, which are warehouse-sized buildings filled with servers of Internet service providers (ISP). For example, an ISP with 8,000 servers needs 2 megawatt of power [3]. The power demanded by information technology (IT) will grow exponentially to overpass all other power uses combined in the near future. Therefore, it is obvious that both energy-constrained electronics and servers need processors that consume less energy.

Due to the confined battery capacity in portable systems and power efficiency in high-end systems, to maintain adequate computing performance levels without useless drainage is the main purpose in power-aware computing. Therefore there has been tremendous interest in power management for portable and mobile embedded systems, showing evident advances in the fields of circuit designs, processor architecture, and operating systems, and moving toward system-level or high-levels in design architecture.

From research during the past several years, we can easily catch two major shifts in the field of power-aware computing. First, power-efficient designs, i.e., low power designs, are no longer only for portable computing systems. Power dissipation has rapidly become a first-order design constraint in almost every type of computing system including hand-held devices, desktop computers, and even high performance computing servers. The second is that computing efforts based on the state of power capacity and the power estimation of system components or whole system level have been introduced as power-aware computing.

For the power-efficient designs, significant reduction in power consumption is possible from techniques ranging from low-power CMOS circuit design to power management software tools. Initial power management efforts focused on putting the system in a low-power/low-performance sleep mode when it was idle. With the advent of the advance configuration power interface (ACPI) standard, such power management has been successfully employed in real-life systems. However, such approaches depend for their efficacy on efficient ways to decide when and which device should be shut down and woken up.

An energy characteristic called voltage-clock scaling (VCS) in CMOS technology allows the designers to obtain revolutionary power reduction. Power is proportional to the square of the operating voltage and proportional to clock frequency. Modern processor cores can operate at different voltage ranges to achieve different levels of energy efficiency showing a property of operating voltage-frequency tradeoff. In other words, lowering the voltage means lowering the frequency and vice versa. Scaling down both operating voltage and clock frequency allows the low power design to save energy consumption and to be studied actively. For instance an ARM7D processor can run at 33MHz and 5V as well as at 20MHz and 3.3V. The energy-performance measures at these two operation modes are 185 MIPS/WATT and 579 MIPS/WATT, and the MIPS measures are 30.6 and 19.1, respectively. Another example is Motorola's PowerPC 860 processor that can be operated in a high-performance mode at 50MHz and with a supply voltage of 3.3V, or a low-power mode at 25MHz and with an internal voltage of 2.4V. The power

consumption in the high-performance mode is 1.3 Watts, as compared to 241mW in the low-power mode.

Advances in low power circuit design and power management have introduced a new research field of low power computing, which emphasizes running at a low energy consumption level, if possible, while conserving the performance of an application. In addition to low power designs, power-aware computing that reflects the state of battery capacity and the characteristics of batteries is widely studied. But, in general, the timeliness of computing is not greatly considered in power-aware computing. Real-time systems are often needed in specific and hard environments, such as stand-alone or isolated locations, so that power consumption becomes a critical design issue. More importantly, they have strict requirements of temporal correctness as well as functional correctness. Missing timeliness in real-time systems is considered as a failure and may have catastrophic consequences. For instance, a missed deadline in the control loop of the auto landing system may cause a crash. Even for non-time-critical tasks that are with soft or firm deadlines, a delay in task completion greatly reduces the value of their computation results. Thus, we can generalize system failures to include non-responsiveness or tardy outputs of the system.

Therefore, to get effective power-savings requirements in real-time systems without violating time constraints of real-time applications, we propose a new concept of power-aware real-time computing and its system model. It supports an integration of hard and soft real-time tasks and multi-task soft real-time applications like multimedia application currently attracting much attention.

In this dissertation, first we investigate the issues related to the low-power real-time scheduling and significantly improve energy-efficiency in real-time systems by combining the power reduction characteristics with real-time scheduling. For hard real-time scheduling for a single processor, we propose VCS-EDF scheduling focused on dynamic reclaiming of early released resources in earliest deadline first scheduling using voltage clock scaling. In addition to a

static voltage assignment, we propose a new dynamic-mode assignment, which has a flexible voltage mode setting at run-time enabling much larger energy savings. Using simulation results in VCS-EDF scheduling and exploiting the interplay among power supply voltage, frequency, and circuit delay in CMOS technology, we find the optimal two-level voltage settings that minimize energy consumption.

Secondly, we study battery-driven real-time systems, which jointly schedule hard periodic tasks and aperiodic (sporadic) tasks, whose available energy is bounded in the range possibly given by a set of tasks. Therefore, reducing power consumption is not the only objective of task scheduling in this case since response time may be unrestricted when it concerns only concerning total energy reduction. To make the best of an available energy budget constrained for specific time period, an effective energy-sharing scheme should be provided for the combined objectives. We model and solve these problems for a real-time system based on the tasks' requirements, their characteristics of energy and utilization demands, and the power-utilization property in VCS-EDF scheduling. We first profile energy and utilization usages of periodic and aperiodic tasks focusing on EDF scheduling using static VCS. Then for the sharing of the bounded energy budget in a real-time system, an energy budget allocation method is proposed. We also propose a static scheduling algorithm to determine the optimal two-level voltage settings of all tasks under bounded energy consumption, while guaranteeing that no deadline of periodic task is missed and average response time of aperiodic tasks is minimized. The algorithm selects the voltage settings that have the minimum average response time among the schedulable ones within a given energy consumption. To schedule aperiodic tasks, we adopt Total Bandwidth Server proposed by Spuri and Buttazzo, which can be expressed by bandwidth preserving algorithm that guarantees the isolation of bandwidth between periodic and aperiodic tasks.

Thirdly, we expand our energy budget allocation model to a dual-policy dynamic scheduling that significantly improves energy-efficiency and extends the lifetime of battery-driven real-time systems. The proposed dynamic scheduling switches its scheduling policies

between two schedules utilizing an explicit pattern of event occurrence at run-time, exploiting the interplay between utilization and energy consumption in VCS-EDF and devising a new view of total bandwidth server algorithm in terms of sharing constrained energy. One scheduling policy is a set of worst-case voltage settings from the proper allocations of energy budget. The other is for a given periodic task set to consume minimum energy under allocated energy consumption. During the intervals having only periodic tasks, the switched mode execution from high-speed in the worst-case schedule to low-speed enables the Dual-Policy Dynamic Scheduling to outperform in the reduction of energy consumption, compared with the scheduling method that sticks to the running modes by the worst-case.

Finally, in the proposed Dual-Policy Dynamic Scheduling, we build a scaling mechanism that can optimize energy consumption and responsiveness to the aim performance of a system. Fully utilizing the property of energy and utilization sharing in total bandwidth server having a restriction of limitation in energy consumption, we characterize the performance of the Dual-Policy Dynamic Scheduling and show scaling up/down total power consumption and average response time by controlling the utilization of periodic tasks in their plausible range.

1.2 Research Background and Related Works

1.2.1 Dynamic Power Management

An approach to low power is to utilize power down features to minimize the power consumption of unused hardware. For instance, turning off (or dimming) the screen while a laptop computer is idle and shutting down hard disks while it is not accessed conserves the energy without affecting any functionality. Packing message transmission delays state change from the listening mode to the transmit mode of radio part such that it extends the battery life for mobile devices in the wireless communication environment. But, re-activation of hardware can take some time, affecting performance like response times.

The simple-power-down-when-idle techniques can significantly reduce the system's power consumption [1]. Dynamic power management (DPM) reduces power consumption by

selectively putting idle system components to low-power states. Thus there are several power states for each component in a system and the state transitions are controlled by commands issued by a power manager (PM) that decides when and how to force the power state transitions based on its observations and/or assumptions on the workload. The power manager makes state transition decisions according to the power management policy [4]. Because shutting down an active device and waking up a sleeping device take time and extra energy, changing power states has delay and/or energy overhead and the policies trade off the performance for the power consumption.

Microsoft proposed OnNow Initiative that controls power/performance through the software layer [5]. Intel, Microsoft, and Toshiba proposed the open industrial standard called advance configuration power interface (ACPI) such that system-level power management has been successfully settled down in real-life [6]. Linux also supports ACPI from kernel 2.4 [7]. But because the policies depend on the workload predictions and/or assumptions, such approaches should be more refined for the systems to be optimally power managed.

1.2.2 Low Power vs. Power-aware Computing

Limiting power consumption presents a critical issue in computing, particularly for portable and mobile devices. The field of computer architecture has recently seen a rapidly expanding interest in power reduction at the architectural and software level. Low power computing has traditionally been the primary focus of designers of portable and battery-powered computing systems and has in the past largely been considered a low-level circuit design issue.

There are many facets in modern computers where the system designer can get energy saving because most computing systems generally do not consume all of their resources all of the time. Slack time can be exploited to either slow down computation or save energy by shutting off components while not in use. And much previous research on low power design techniques tries to minimize average power consumption either by transition to a low power consumption state when it is idle [8, 9, 10, 11] by reducing the active current drawn by a circuit keeping the supply

voltage fixed, or by scaling the supply voltage statically or dynamically. But, from the viewpoints of power-aware computing, the goal targets not only saving energy without sacrificing performance in intended applications to get longer battery life, but also making the best use of the available power for power-confined devices. For example, power-aware computing/ communication (PAC/C) is carried in DARPA project [12], in which promising areas such as power-aware algorithm designs/libraries, power-aware compilation, power-aware operating systems as well as a novel integrated software/hardware technology suite incorporating innovative individual power reduction technologies are being pioneered.

In power-aware computing, the remaining power capacity and the consumption management of power source are very important such that a system can effectively use the available power. Battery-driven systems like handheld devices have more stringent power consumption limits. In such systems, battery management is also highly emphasized on an extent of mission duration and quality of services due to the increasing requirement of miniaturization and complicated functionalities. Therefore, a lot of works have been carried to understand the behavior and properties of the power source in power-aware computing [13, 14, 15, 16, 17].

Sometimes, to prolong the lifetime of the power source, the reduction of workload to be done is required to fit the power consumption into the available level, but leads to unnoticeably degraded performance in applications. Available energy should be well allocated among applications in the system. Regarding proper allocation, components in a system including hardware and software need to be redefined as power-aware ones such that they enable the applications to become acceptable for the user at some power level. Power-awareness of components requires that the energy consumption be quantified in component basis. Much work has been done to measure and abstract the quantities in every system level, from circuits to applications. One of the well-known frameworks of energy consumption estimation is the advanced configuration and power interface that provides system-wide power states. For low-power software development Simulators such as Wattch [18] and SimplePower [19] estimate the

power consumption of microprocessor-based systems in an instructional- or architecture-level. As a profiling application energy usage, PowerScope quantifies what fraction of the total energy is consumed for specific processes in the system [20]. For system-on-chip (SOC) components such as memory hierarchies and system buses, power estimation and optimization techniques have been proposed [21]. X. Fan *et al.* investigate the memory's influence on selecting the appropriate voltage/operating frequency and extract the factors that affect the clock scaling decision to meet the system's energy/performance goals [22]. Furthermore, a circuit level energy-monitoring tool is proposed for the embedded systems, in which it collects power consumption data even in a cycle-by-cycle resolution [23].

1.2.3 Dynamic Voltage-Clock Scaling

The basic concept of power reduction in the variable voltage processors is a technique called *voltage-clock scaling* or *dynamic-voltage-scaling* in CMOS circuit technology. CMOS circuits have both dynamic and static power consumption. Static power consumption is caused by bias and leakage currents and is insignificant in most designs that consume more than 1mW. The dominant power consumption for CMOS microprocessors is the dynamic power. Every transition of digital circuit consumes power, because every charge or discharge of the digital circuit's capacitance drains power. The dynamic power consumption is equal to

$$P_{cmos} = C_L N_{sw} V_{DD}^2 f, \tag{1-1}$$

where C_L is the output capacitance, N_{sw} the number of switches per clock, and f the clock frequency. It is clear that power consumption in CMOS digital circuits is proportional to the square of the supply voltage (V_{DD}) and the clock frequency such that lowering V_{DD} is the most effective mean to reduce the power consumption. Lowering V_{DD}, however, causes the problem of increased circuit delay. The circuit delay t_d is given by the following equation [24]:

$$t_d = k \frac{V_{DD}}{(V_{DD} - V_T)^2}, \tag{1-2}$$

where k is a constant depending on the output the gate size and the output capacitance, and V_T is the threshold voltage. As the clock frequency is inversely proportional to the circuit delay, it is expressed using t_d and the logic depth of a critical path, L_d [25]

$$f = \frac{1}{L_d t_d}. \tag{1-3}$$

Obviously, from the equations, we can see that there is a fundamental trade-off between circuit delay and supply voltage. Processor can operate at a lower supply voltage, but only if the clock frequency is reduced to tolerate the increased propagation delay.

Kuroda *et al*. use voltage scaling in the design of a processor core, in which they can adjust internal supply voltages to the minimum automatically according to its operating frequency [26]. More recently, there are a number of variable voltage processors that allow the supply voltage to be changed. Based on the VCS technique, most of today's processor cores have been designed to operate at different voltage ranges to achieve different levels of energy efficiency as shown in Table 1-1.

Table 1-1 Microprocessors that allow the core to operate at different voltages and frequencies

Processors	Voltage	Speed (MHz)	Power Consumption (Watt)	Features
StrongARM SA-2	1.30 0.75	600 150	0.45 0.04	12-fold energy reduction
Pentium-III	1.60 1.35	650 500	22 9	SpeedStep Technology - 2 modes
Crusoe (TM5400)	1.65 1.10	700 200	2 1	16 levels in steps of 33MHz
XScale	0.75 1.6	150 800	0.04 0.9	185 MIPS 1000 MIPS
ARM7D	5.0 3.3	33 20	0.165 0.033	185 MIPS/W 579 MIPS/W
PowerPC860	3.3 2.4	50 25	1.3 0.241	2 modes

The StrongARM SA-2 processor, designed by Intel, is estimated to dissipate 500 mW at 600 MHz, but only 40 mW when running at 150 MHz, showing a 12-fold energy reduction for a 4-fold performance reduction [27] and the Pentium-III processor with SpeedStep technology dissipates 22W at 650MHz but 9W at 500 MHz [28]. Likewise, AMD has added clock and

voltage scaling to the AMD Mobile K6 Plus processor family and Transmeta has also announced TM5400 or Crusoe processor [29] that actually supports voltage scaling and 16 levels ranging from 1.65V at 700 MHz to 1.1 V at 200 MHz in steps of 33 MHz. For wireless and networking services, Intel also introduces XScale micro-architecture that features on-the-fly-scaling of voltage and frequency in the range from 40mW at 150MHz up to 0.9W at 800MHz [30]. Pouwelse *et al*. show several levels of the clock frequency versus supply voltage for the Transmeta Crusoe processor and carried an actual implementation in a wearable computer for low-power computing [31].

Digital CMOS circuits are very amenable to implementing DVS because their performance and energy consumption scale together over a wide range of supply voltage. Although the maximum supply voltage drops with improved process technology, reducing the range, DVS will continue to be a viable technique for future process technologies.

1.2.4 Fundamental Scheduling Theories

1.2.4.1 RMA and EDF Scheduling

Rate monotonic analysis (RMA) is a collection of quantitative methods and algorithms with which we can specify, understand, analyze, and predict the timing behavior of real-time system designs. RMA grew out of the theory of fixed priority scheduling. A theoretical treatment of the problem of scheduling periodic tasks was first discussed by Serlin in 1972 [32] and then more comprehensively discussed by Liu and Layland in 1973 [33]. They studied an idealized situation, in which all tasks are periodic, do not synchronize with one another, do not suspend themselves during execution, can be instantly preempted by higher priority tasks, and have deadlines at the end of their periods. The term "rate monotonic" originated as a name for the optimal task priority assignment in which higher priorities are accorded to tasks that execute at higher rates (that is, as a monotonic function of rate). Rate monotonic scheduling is a term used in reference to fixed priority task scheduling that uses a rate monotonic prioritization. Although a

fixed priority scheduler is used widely due to the feature of analysis for a system's feasibility, RMA cannot commit to the feasibility of a system that uses more than in worst case 69% [33] or in an average case 88% [34] of available CPU time. And the worst problem with fixed priority scheduling is that it is oblivious to deadlines, that is, blindness to hard real-time.

Compared to rate monotonic scheduling, which is a static priority driven scheduling method, earliest deadline first is a dynamic priority driven scheduling method. In EDF, the task with the earliest deadline is always executed first and fairly easy to implement. The execution queue is sorted by the time of the next deadline. When a task becomes active it must be inserted in the execution queue, or if its deadline is before the deadline of the currently executing task, it must preempt the currently executing task. EDF has only one serious drawback: the theory is optimal only for loads that can be scheduled. It does not address overloads. EDF degrades ungracefully when it is overloaded. Another big point is that dynamic priority scheduling tends to be complex and sometimes uses heuristics comparing to the guaranteed feasibility based on an analysis in fixed priority scheduling. But a processor can be fully utilized by dynamic priority scheduling.

1.2.4.2 Aperiodic task scheduling

Contrary to the guarantee of meeting deadlines in the hard task scheduling, there is no form of guarantee for aperiodic tasks. They are usually served in background with respect to hard tasks in order not to jeopardize the schedulability of hard tasks or served by an aperiodic server to improve responsiveness. Substantial research has been done in the scheduling of aperiodic tasks in both fixed and dynamic priority systems. The sporadic server algorithm [35], the deferrable server algorithm [36], and the slack stealer algorithm [37] are for the fixed priority environment. A number of algorithms that solve the aperiodic task scheduling in dynamic priority systems using EDF scheduler can be found in literatures; deferrable server [38], slack stealing [39], or sporadic server [39, 40]. To improve the response time of aperiodic requests, an efficient

approach called total bandwidth server (TBS) has been proposed by Spuri and Buttazzo [40, 41]. Yielding much faster responsiveness under sharing resources among periodic and aperiodic tasks, the algorithm has been extended to reflect in the environment of resource sharing [42, 43]. The TBS allocates feasible priority by assigning suitable deadlines to arrived aperiodic tasks while guaranteeing deadlines of other periodic tasks and schedules them as hard periodic tasks. By handling aperiodic tasks like periodic tasks within the reserved bandwidth, the TBS can produce better responsiveness than other aperiodic mechanisms, sporadic server, slack stealing, and so on, based on the idea of using idle time of periodic scheduling or "stealing" all the possible processing time from the periodic tasks, without causing their deadlines to be missed.

1.2.5 Low Power Real-Time Computing

Low energy consumption has already shown up as a critical feature in high-end systems as well as power confined real-time systems. Dynamic voltage scaling and frequency scaling in variable voltage processors are outstanding techniques because the power consumed by a component implemented in CMOS technology is quadratically proportional to voltage and linearly proportional to frequency. In other words, operation takes even longer, while greater energy reductions can be achieved with slower clock and lower voltage. The longer execution time affects performance degradation in application or meeting strict time constraints in real-time systems. Therefore there have been a number of techniques that combine low-power techniques and scheduling algorithms in systems.

The benefits of reducing the supply voltage and clock speed in a processor to save energy have been primarily studied based on simulations. Weiser *et al*. showed the potentials of voltage scaling for general-purpose processors using traces gathered from UNIX-based applications [44]. Their scheduling algorithms are interval-based voltage schedulers, which set clock speed depending on processor utilization in previous intervals. By employing the same traces, Govil *et al*. extend the work of Weiser *et al*. and study several dynamic speed-setting policies that slow the processor when it is mostly idle and speed it up when it is mostly active [45]. Also using interval-

based clock scheduling Pering *et al*. achieve considerable reduction of energy consumption comparing to full speed running [46] and later they present a heuristic based on EDF scheduling [47]. Unlike previous studies in voltage scaling that rely on simulations, there are several raw measurements of the power-delay trade-off in voltage scaling. Grunwald *et al*. have investigated the power consumption of the Itsy Pocket Computer [48] for previously proposed interval-schedulers and found that frequent voltage and clock changes incur unnecessary overhead. Throughout a wireless multimedia terminal called InfoPad, Truman *et al*. measure the power consumption in the portable system and DVS scheme to reduce the power consumption [49]. With the importance of the prediction of appropriate performance for the next interval, a workload prediction strategy using adaptive filtering is proposed [50]. However, since the gain is extremely dependent on the interval length, the optimal setting varies per application, and even the individual requirements of running tasks, e.g., deadlines and computation estimates, were not considered, interval-based voltage schedulers result in poor behavior for more complex workloads such as burst applications.

In DVS approaches for real-time systems, scheduling is done at the task level using the task execution times based on worst execution times (WCET) in general and exploiting slack times from static analysis and runtime workload variation.

The concept of real-time scheduling is firstly applied to dynamic speed setting by Pering and Broderson [51]. Non-preemptive EDF scheduling based on the ILP method is also presented [52]. The authors controlled two different operating speeds/voltages through OS system calls to minimize the energy consumed by a periodic task set. A minimum-energy scheduler for dynamic speed-setting based on the EDF scheduling policy is proposed [53] and they compare some on-line algorithms to an off-line algorithm executed by assigning the optimal processor speed setting to a critical interval that requires maximum processing. Similarly, Hong *et al*. consider a low energy heuristic for non-preemptive scheduling with a dynamically variable range of [0.8, 3.3] V [54]. They also present a heuristic even for preemptive model with a limited speed changes [55].

Quan and Hu study the optimal voltage setting for fixed-priority scheduling [56]. All these approaches require that the task release times must be known *a priori*. Hong *et al*. describe an on-line scheduling algorithm for hard real-time tasks assuming the release times of jobs are not known *a priori* [57]. For the periodic task model and rate-monotonic scheduling, two on-line voltage-scaling methods [58] were proposed, which change voltage levels at the execution stage from the initially assigned levels as such changes become necessary. Using integer linear programming (ILP), Ishihara and Yasuura present a very interesting result regarding the impact on energy of the number of available distinct voltage levels [59]. They pointed out that at the most two voltage levels are usually enough to minimize energy consumption. In the presence of task synchronization, R. Jejurikar and Gupta use slowdown factors based on DVS for energy saving of EDF scheduling [60].

There are also several hybrid approaches exploiting both DPM and DVS schemes. Transmeta has combined both schemes in a processor called Crusoe [61] targeting mobile computing, which is able to dynamically adjust clock frequency in 33 MHz steps. By running applications at several frequencies even in the *active* mode of DPM, they can get larger power savings in addition to the one obtained by DPM. Based on the studies of energy efficient design for portable devices [62], another hybrid method has been studied on MPEG applications on top of a Crusoe processor based system [63]. Also Kim and Ha observe trade-offs between DPM and DVS schemes and propose a hybrid method that exploits both workload-variation slack time and shutting off unused components on a timeslot-by-timeslot basis [64].

In this dissertation, we apply VCS scheme to EDF scheduling for hard real-time tasks, reclaiming resources based on the property that actual execution times of tasks in real-world embedded systems change as much as 87% relative to the measured WCET [65].

1.2.6 Power-aware Real-Time Computing

Maximizing the utilization time of energy that can be delivered by a battery have become one of the most important design considerations for a power-limited system, since it directly

impacts the extent of the system life. Instead of focusing on reducing power consumption alone, researchers have begun to study the battery behavior and the effect of the battery discharge pattern on battery capacity as well [66]. Against the result of previous interval-based dynamic voltage scaling scheduling, in the experimental research by using the Itsy hand-held computer [67], Martin and Sieworek show that reducing power consumption is not always the best strategy when taking battery effectiveness into account because most of the batteries are non-ideal [68, 69]. He presents a revised Weiser's PAST algorithm to account for the non-ideal properties of batteries and the non-linear relationship between system power and clock frequency. Based on the non-linearity in battery discharge, a system-level power estimation methodology is proposed [70], where they have shown the formula of energy-consumed processor, memory and L2 cache, interconnects and pins, DC-DC Converter, and a battery capacity model. Pedram and Wu study tradeoffs between energy dissipation and delay in battery-powered digital CMOS design [71, 71]. Benini *et al*. propose several DPM policies to take into account battery's charge state study [73]. In particular, they introduce a class of closed-loop policies, whose decision rules used to control the system operation state are based on the observation of both system workload and output voltage of the battery.

Power-aware real-time scheduling must carry out two key features, not only being aware of domain-specific knowledge such as the power source, battery status, and other operating conditions, but also guaranteeing requirements of real-time tasks, assuring the best performance of real-time applications, and even efficient utilization of power constraints under the power-awareness.

Much research has been done to introduce the power-awareness into real-time computing, from the properties of power source to instructional level by an assistance of compiler. For distributed real-time embedded systems having voltage-scalable capability, Luo and Jha suggest two schemes to optimize the battery lifetime by modeling a battery [74]. The one is a battery-aware scheduling scheme based on heuristics of minimization of the peak power consumption

and reduction of the variance of the discharge current profile to get the battery efficiency. The other is a variable-voltage scheduling scheme via efficient slack-time re-allocation. Motivated by the task requirements and power source characteristics of Mars rover having two power sources: a solar panel and non-rechargeable battery, Liu *et al*. propose a power-aware real-time scheduling [75]. All prior studies try to reduce total energy consumption exploiting the property of scaling scheme or power source.

Besides, real-time embedded systems having a constrained energy budget lower than the maximum, which the tasks can consume, demand an efficient scheme to share a limited energy budget among competing real-time and non-real-time tasks as well as reduction of total energy consumption. For systems consisting of soft aperiodic tasks, the objective of minimizing power consumption using a popular energy reduction technique, voltage-clock scaling, will result in slow execution. On the other hand, in many cases, the battery capacity can be replenished or there is a finite mission lifetime. Minimizing power consumption that doesn't utilize all available energy may not lead to optimal system performance. A better power control strategy in such cases is to minimize the response times of soft real-time tasks, providing that the deadlines of hard real-time tasks are met and the average power consumption is bounded.

In spite of widespread recognition of the importance of energy, there is no convenient abstraction of the power consumption and sharing in real-time scheduling. One of goals in this dissertation is to abstract energy usage and utilization required by real-time tasks and to allocate the confined energy budget as a first criterion in real-time scheduling. On top of the quantitative analysis for the system's power-awareness such as system component or instructional level power estimation/monitoring, we abstract energy and utilization usages on task basis by exploiting the property of voltage-clock scaling. Also, we build a constrained energy budget allocation model that enables tasks to share the available energy concerning the requirements of real-time tasks and present a static analysis based on voltage-clock scaling. Furthermore, we propose an energy efficient real-time dynamic scheduling, which switches two scheduling policies to get energy

saving, and construct an algorithm to optimize energy consumption and responsiveness according to real-time application's requirement.

1.3 Main Contributions

The objectives of the dissertation are to design a low-power scheduling reducing total power consumption for hard-real-time applications and to build power-aware real-time systems supporting energy efficiency for real-time application under bounded energy budget. The main contributions of the dissertation are as follows:

A low-power scheduling for hard real-time systems. We devise comprehensive scheduling algorithms for integrated real-time systems. They include

(1) a fundamental two-mode scheduling theory,

(2) reduction of power consumption using power-aware voltage-clock scaling,

(3) dynamic reclaiming of early released resources, and

(4) optimal voltage setting in VCS-EDF scheduling.

Power-aware scheduling for mixed real-time tasks under a constrained energy budget. In order to develop a power-aware static analysis for real-time embedded system, the relationship of energy and workload demands is investigated. We develop an energy budget allocation model for battery-driven real-time systems, including not only small appliances being widely used, but also sensor network nodes for special purposes. The model concerns

(1) integrated scheduling of both hard and soft real-time tasks in a system,

(2) profiling of energy and processor's capacity usages on the basis of task

(3) efficient share of a bounded energy budget for battery-driven real-time systems, and

(4) better performance in non-real-time applications.

Dynamic scheduling for mixed real-time tasks considering power efficiency. We demonstrate that more sophisticated and effective power-aware dynamic scheduling may enhance the overall power consumption for real-time systems. They include

(1) a fundamental dual-policy dynamic scheduling theory,

(2) static analysis using the energy budget allocation model,

(3) modified Total Bandwidth Server for scheduling aperiodic tasks, and

(4) reduction of total energy consumption.

Optimizing energy efficiency and responsiveness. Given the energy-utilization tradeoffs in power-aware real-time scheduling, we show that the adjustment of the periodic tasks' workload within schedulable range can lead to an effective utilization of the limited energy budget from the understanding of the energy and performance implications. The tuning of periodic tasks' workload implies

(1) making the best use of a constraint energy budget,

(2) appropriate sharing of energy and processor's computation capacity in run-time scheduling,

(3) proper adjustment of energy consumption and responsiveness to the system's performance, and

(4) reduction of total energy consumption pursuing system's target performance.

1.4 Organization of the Dissertation

The rest of the dissertation is organized as follows. Chapter 2 presents a solution for reduction of total energy consumption in hard real-time systems. The two-mode Voltage-Clock Scaling EDF scheduler is proposed, which uses a simple setting arrangement that the processor in a real-time system can be dynamically configured in one of two modes named *low-voltage* (*L) - mode* and *high-voltage* (*H)-mode*. In *L -mode,* the processor is supplied with a low voltage (V_L) and runs at a slow clock rate. And, the processor can be set in *H-mode,* i.e. be supplied with a high voltage (V_H) and run at a fast clock rate.

Chapter 3 investigates the VCS problem in sharing a limited energy in battery-driven real-time embedded systems. To provide real-time guarantees, the delay penalty in VCS needs to be carefully considered in real-time scheduling. In addition to real-time requirements, the systems may contain non-real-time tasks whose response time should be minimized. Thus, a combination of optimization objectives should be addressed when we establish a scheduling policy under a

power consumption constraint. In this dissertation, we devise the VCS-EDF scheduling to get proper allocation of energy budget that is limited to mixed hard and soft real-time tasks. Based on schedulability of VCS-EDF, we first analyze the characteristics of energy and utilization demand of periodic and aperiodic tasks, and then profile energy and utilization usages in the viewpoint of real-time scheduling. The profiling enables developers to build an energy-efficient or power-aware real-time schedule and is directly related to how definitely it assigns a bounded energy cost to tasks for specific real-time application. Then, we build a model for sharing constrained energy budget to real-time tasks and propose a static real-time scheduling to assign operation voltage/frequency to tasks, satisfying the requirements of hard and soft real-time applications. To get a better performance for aperiodic tasks, voltage settings resulting in the fastest responsiveness are selected among the schedulable ones under a given energy budget.

Chapter 4 proposes an efficient energy saving scheme for run-time scheduling called dual-policy dynamic real-time scheduling. It is for the real-time embedded systems, which jointly schedule both hard and soft tasks, and whose energy usage is bounded. By the constrained energy allocation scheme proposed in chapter 3, a set of voltage settings that make the best use of assigned energy and gives better performance for aperiodic tasks is decided. In the chapter, based on the voltage settings determined in the constrained energy allocation, we propose a dynamic scheduling to ensure an extended lifetime of the system under the given energy budget. The proposed approach is featuring an intermixing of two elementary schedules with the following observations. As total bandwidth server algorithm is applied to mixed tasks under a constrained energy budget, we notice the share of the processor's capacity is directly related to the energy sharing among them. Also, we observe a change of pattern in a joint scheduling of both periodic and aperiodic tasks, which consists of two intervals assorted by the existence of any aperiodic event. As for energy consumption in two-mode voltage-clock scaling EDF scheduler based on total bandwidth server algorithm, the voltage settings for only periodic tasks consumes less energy than the ones for mixed tasks. In the intervals having only periodic tasks, applying worst-

case voltage settings determined for the mixed tasks leads to more energy consumption to the voltage settings assigned for periodic tasks only. Therefore, the proposed algorithm switches running modes (scheduling policies) along with the change of pattern in event occurrences. For the intervals having mixed events of periodic and aperiodic tasks, the worst-case schedule is taken. For the other intervals having only events of periodic tasks, the voltage settings only for periodic tasks are allowed for energy saving.

Chapter 5 considers a scaling mechanism in total energy saving and responsiveness of aperiodic tasks in the dual-policy dynamic real-time scheduling and proposes a method to adjust energy consumption and response times to the aimed performance. When we introduce a set of scheduling policies as a set of voltage settings/running speeds for periodic tasks in chapter 4, they fall upon extremes in feasible ranges of their energy consumption and utilization. Due to the difference in energy consumption between the two extreme schedules, the dual-policy dynamic scheduling can save energy consumption. In-between the extreme schedules for periodic tasks, there exist other combinations of voltage settings/running speeds, which have different energy consumption and utilization. If a set of voltage settings other than extremes is used for the intervals consist of only periodic tasks, the amount of energy consumption and average response time that the scheduler can acquire is changed accordingly. Thus, we introduce a new scaling factor, which can control the energy consumption/responsiveness, such that the dual-policy dynamic scheduling optimize the system's performance to the requirement of the real-time system under a specific energy budget.

Chapter 6 summarizes the main contributions of the dissertation, and discusses future research directions.

CHAPTER 2
TWO-MODE VCS-EDF SCHEDULING FOR HARD REAL-TIME SYSTEMS

2.1 Introduction

In recent years, the rapidly increasing popularity of portable battery-operated computing and communication devices have motivated research efforts toward low power and energy consumption. Battery life is the primary constraint for energy-constrained systems such as personal mobile phones, laptop computers, and PDA's. Significant reduction in power consumption is possible from techniques ranging from low-power CMOS circuit design to power management software tools.

Initial power management efforts focused on putting the system in a low-power/low-performance sleep mode when it was idle. With the advent of the Advance Configuration Power Interface (ACPI) standard, such power management has been successfully employed in real-life systems. However, such approaches depend for their efficacy on efficient ways to decide when and which device should be shut down and woken up [4, 5, 6].

Processor cores can operate at different voltage ranges to achieve different levels of energy efficiency. For instance an ARM7D processor can run at 33MHz and 5V as well as at 20MHz and 3.3V. The energy-performance measures at these two operation modes are 185 MIPS/WATT and 579 MIPS/WATT, and the MIPS measures are 30.6 and 19.1, respectively [76]. Another example is Motorola's PowerPC 860 processor, which can be operated in a high-performance mode at 50MHz and with a supply voltage of 3.3V, or a low-power mode at 25MHz and with an internal voltage of 2.4V [77]. The power consumption in the high-performance mode is 1.3 Watts, as compared to 241mW in the low-power mode.

The basic concept of power reduction in the variable voltage processors is a technique called Voltage-Clock-Scaling in CMOS circuit technology. The power consumed for transitions in digital circuit is equal to $P_{cmos} = C_L N_{sw} V_{DD}^2 f$ (2-1). Due to the quadratic relationship between the supply voltage (V_{DD}) and the clock frequency, a small reduction in voltage can produce a significant reduction in power consumption. However, lowering V_{DD} increases the circuit delay by the amount of the equation $t_d = k V_{DD} / (V_{DD} - V_T)^2$ (2-2). Obviously, the longer execution time affects performance degradation in application or meeting strict time constraints in real-time systems. From equation (2-1), lowering supply voltage will reduce power consumption. However, it also slows down the logic.

Based on these equations, a voltage-scaling scheme was introduced in many studies by dynamically adjusting clock speed, allowing a system to operate at a lower voltage level to save energy without missing task deadlines. Clearly, if low energy consumption is a desirable feature in real-time systems, voltage-clock scaling must cooperate with the task scheduling algorithms since the power-delay tradeoff property in low power design affects meeting the strict time-constraints of real-time systems. For instance, the execution of a high-priority at a low voltage and slow clock rate may cause a low-priority task to miss the deadline due to the additional delay from the execution of the high priority task.

In this dissertation we extend the resource reclaiming [78] with EDF scheduling algorithms. Energy saving is made possible by running tasks in low-voltage mode during reclaimed slack periods that are released by tasks that do not consume their entire worst-case execution times. The algorithm proposed in this paper makes the slack time reclamation possible even if the task arrival instances or phases are not known *a priori.* We also propose a new dynamic voltage assignment which can result in much greater energy saving. We then discuss and demonstrate how the voltage setting should be chosen so that the dynamic algorithm and slack period reclamation can be most effective.

The chapter is organized as follows. In Section 2.2, we outline the preliminary system model. In Section 2.3, we describe the detail algorithms of voltage assignment and slack period reclamation. To illustrate the effectiveness of the proposed algorithms, we evaluate the energy saving performance in Section 2.4. In the following section, we present how to get the optimal voltage settings. Finally, a short conclusion is provided in Section 2.6.

2.2 System Model

In real-time systems, tasks may arrive periodically or sporadically and have individual deadlines. For Earliest Deadline First (EDF) scheduling, a task τ_i is modeled as a cyclic processor activity characterized by two parameters, T_i and C_i, where T_i is minimum inter-arrival time between two consecutive instances and C_i the worst-case execution time of task τ_i.

The EDF scheduling algorithm always executes the earliest-deadline task awaiting service. To apply voltage-clock scaling under EDF scheduling, we make several assumptions as follows:

A1. Voltage switching consumes negligible overhead. This is analogous to the assumption, made in classical real-time scheduling theory, that preemption costs are negligible. Voltage switching typically takes a few microseconds. In fact, a bound of the total overhead can be calculated by considering the number of task arrivals and departures since voltage switches are only done at task-dispatch moments.

A2. Tasks are independent: no task depends on the output of any other task.

A3. The worst-case execution demand of each task τ_i, C_i, is known. The actual execution demand is not known a priori and may vary from one arrival instance to the other.

A4. The overhead of the scheduling algorithm is negligible when compared to the execution time of the application.

A5. The system operates at two different voltage levels. Ideally, a variable voltage processor that has continuous voltage and clock setting in the operational range is available.

The first four assumptions are analogous to assumptions made in real-time scheduling theory. As for A5, we assume a simple setting arrangement that the processor in a real-time system can be dynamically configured in one of two modes: *low-voltage* (*L) -mode* and *high-voltage* (*H)-mode*. In *L -mode,* the processor is supplied with a low voltage (V_L) and runs at a

slow clock rate. Thus, task execution may be prolonged but the processor consumes less energy. On the other hand, the processor can be set in *H-mode,* i.e. be supplied with a high voltage (V_H) and run at a fast clock rate, in order to complete tasks sooner at the expense of more energy consumption. The processing speeds at *L-mode* and *H-mode* are denoted as α_L and α_H, respectively, in terms of some unit of computational work. Depending upon the voltage setting for task τ_i, the worst-case execution time (WCET) is C_i/α_L or C_i/α_H.

Assume that in the system there are n tasks, $\tau_1, \tau_2, \ldots,$ and τ_n, that are numbered in decreasing priority order. That is, $D_1 \leq D_2 \leq \ldots \leq D_n$ where D_i is the deadline of task τ_i. Thus, the total utilization demand by all tasks is $\rho = \sum_{i=1}^{n} C_i/T_i$. Also, upon each invocation, the task τ_i must be completed before its deadline period D_i. This is the real-time requirement.

To minimize energy consumption in real-time systems, the voltage-clock scaling problem is imposed when $\alpha_L < \rho \leq \alpha_H$. The task set is schedulable if the processor is entirely run in *H-mode*, and misses at least one deadline if running in *L-mode* completely. Keeping the processor entirely in *H-mode* to meet the deadlines of all the tasks results in extra energy consumption. Therefore, an algorithm is called for to determine the optimal voltage settings and to minimize *H-mode* execution, while guaranteeing that no deadline is missed

2.3 Voltage-Clock Scaling under EDF Scheduling

The VCS-EDF scheduling is composed of mode assignment and resource reclaiming phases. Assuming tasks run to their WCETs on EDF scheduling, *mode assignment* picks the voltage setting for each task, i.e. *H-* or *L-mode*, that will minimize the total energy consumed and ensure schedulability under EDF. The reclaiming phase dynamically switches voltage settings taking into account the resources released by tasks that complete their work ahead of WCET. The difference between WCET of a task and the execution time it consumes is called a *slack period*.

The optimization problem can be stated as follows: Pick H and L such that

- $H \cup L = \{\tau_1, \tau_2, \ldots, \tau_n\}$

- $H \cap L = \varnothing$
- $\sum_{i \in H} C_i/T_i$ is minimized

subject to the well-known sufficient condition[1] for the schedulability of periodic tasks under EDF, i.e.,

$$\frac{1}{\alpha_H} \sum_{i \in H} \frac{C_i}{\min(T_i, D_i)} + \frac{1}{\alpha_L} \sum_{i \in L} \frac{C_i}{\min(T_i, D_i)} \leq 1. \tag{2-3}$$

This optimization problem can be treated equivalently as the decision problem of the subset sum, which is *NP*-complete. Consequently, efficient search techniques e.g., branch-and-bound algorithms, should be employed to find a solution if n is large.

For dynamic mode assignment, the subsets H and L are determined for each busy cycle during which the scheduler is busy continuously without any idle intervals. We assume that $H = \{\tau_1, \tau_2, ..., \tau_n\}$ and $L = \varnothing$ at the beginning of a busy cycle. At the first arrival instance of task τ_j, the subsets can be modified to $H\text{-}\{\tau_j\}$ and $L \cup \{\tau_j\}$, respectively, if

$$\frac{1}{\alpha_H} \sum_{i \in H - \{\tau_j\}} \frac{C_i}{\min(T_i, D_i)} + \frac{1}{\alpha_L} \sum_{i \in L \cup \{\tau_j\}} \frac{C_i}{\min(T_i, D_i)} \leq 1. \tag{2-4}$$

In other words, the tasks will be assigned to the low voltage mode subject to the schedulability constraint and the assignment is done in the order of the tasks' first arrival instances during each busy cycle. While the dynamic assignment is not aimed at the minimization of the execution period in *H-mode*, it takes advantage of short busy cycles where no *H-mode* execution should be invoked.

Given the voltage assignments defined in H and L, tasks can be dispatched and run in the assigned mode. The sufficient condition guarantees that, if every task takes up to its WCET, there is enough system capacity for each task to be completed before its deadline. When a task consumes less execution time than its WCET, there is a slack period to indicate the remaining budget that is no longer needed before the task's deadline. Since the slack period will not exist

[1] The condition is also necessary if $D_i \geq T_i$ for all i.

any more after the task's deadline, we denote this deadline as the expiration time of the slack period. Thus, a waiting task that has a deadline greater than the expiration time of a slack period can use up this slack period and run in low-voltage mode. This reclamation algorithm is given in Figure 2-1 where the following definitions are used:

ST-Q: the slack-time queue to track the slack periods given from the tasks that don't use up the whole WCETs. Each element in the *ST-Q* indicates an amount of slack period of a completed task. A slack period has an expiration time that is the deadline of the task and can be consumed by any other tasks that have a deadline earlier than the expiration time.

TK-Q: the normal EDF task queue for all tasks in the system.

ct_i : the computing time that task τ_i has consumed for its scheduled period of the EDF schedule. It doesn't include the running time during any available slack periods of other tasks.

st_i : the slack time of task τ_i. The initial slack time is equal to C_i/α - ct_i at the completion instance of τ_i where α is the speed factor of the assigned voltage mode.

enQueue(element, Q): insert an element into the queue of *Q*.

deQueue(Q): delete the first element from the queue of *Q*.

head(Q): the element at the head of queue *Q*.

SL-decrement: the flag to indicate the slack period in the head of *ST-Q* must be decremented as time progresses. The flag will be on when a task is using up the slack period or when the system is idle.

To track the slack periods for all tasks and to combine online EDF scheduling and voltage-clock scaling, we employ two queues, i.e., *TK-Q* and *ST-Q*, in which tasks and slack periods are ordered according to their deadlines and expiration times, respectively. A task's slack period is computed when it completes the actual computation. The slack period is inserted into *ST-Q* if it is greater than zero. At the dispatch instance of task τ_i, the task is executed at low voltage if there is a slack period can be reclaimed (i.e. the expiration time of the slack period is less than the task's deadline). Otherwise, it is executed at its assigned voltage mode.

```
Algorithm Slack-time Reclamation:
  at the arrival instance av_i of task τ_i {
      if (TK-Q != empty) adjust-ct-st;
      ct_i = 0;
      set task deadline to av_i+D_i;
      enQueue(τ_i, TK-Q);  // enqueue the arriving task
      dispatch(TK-Q, ST-Q);
     }
  at the completion instance of a task {
      adjust-ct-st;
      τ_i = deQueue(TK-Q);
      if (τ_i ∈ H)  st_i = C_i/α_H – ct_i      // set slack period
      else  st_i = C_i/α_L – ct_i
      set st_i's expiration time to the deadline of task τ_i
      if (st_i > 0 ) enQueue(st_i, ST-Q);
      dispatch(TK-Q, ST-Q);
     }
  at the end of a slack period {
      st_i = deQueue(ST-Q);
      dispatch(TK-Q, ST-Q);
    }
  Procedure adjust-ct-st {// adjust accumulated execution

          // time and available slack periods
      τ_j = head(TK-Q);
      exec_time = the execution time
                        since the last dispatch instance;
      if (SL-decrement==off)          ct_j = ct_j + exec_time;
      else {          st_k = head(ST-Q);
                      st_k = st_k - exec_time;
      }
    }
  Procedure dispatch(TK-Q, ST-Q)  { // dispatch a task
          // and a slack period, assign execution mode
      if (TK-Q != empty) {
          τ_j = head(TK-Q);
          SL-decrement = off;              // assume no slack time
          if (ST-Q != empty ) {
                      st_k = head(ST-Q);
                      if (deadline_j >= expiration_time_k)  {
                                  SL-decrement = on;          // reclaim
                                  set voltage_mode to L;      // L- mode
                                  set the end of slack period to
                                              (current instance + st_k );
                      }
                      if (SL-decrement == off) {// no reclamation,
                                  // run the  task with the assigned voltage
                                  if (τ_j ∈ H) set voltage-mode to H;
                                  else set voltage_mode to L;
                      }
                  }
      }
      else        {       // the system is idle,
          st_k = head(ST-Q);              // slack period must be
                                  SL-decrement = on;          // decremented
                  set the end of slack period to
                                  (current instance + st_k );
      }
  }
```

Figure 2-1. On-line reclamation algorithm for voltage-clock scaling

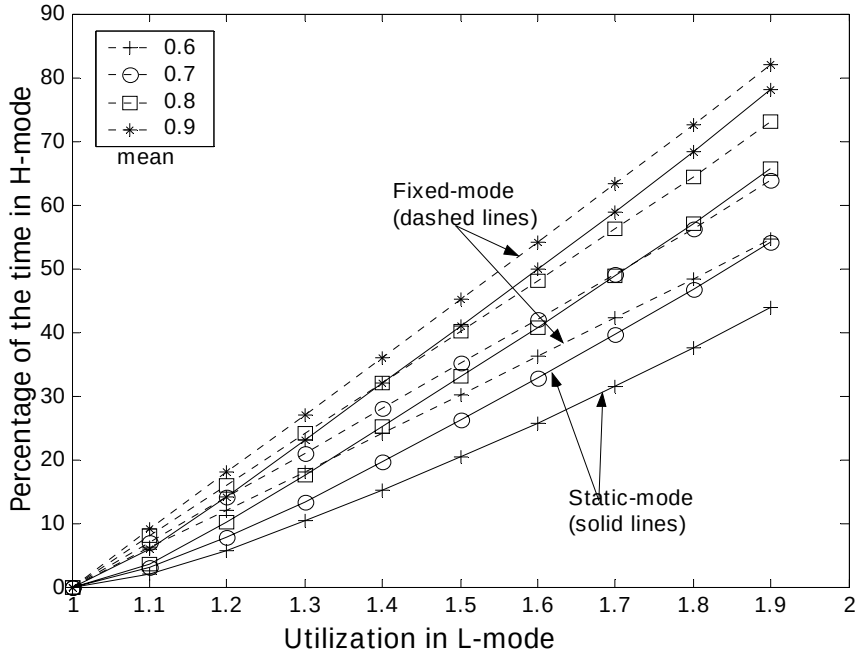
When a slack period is reclaimed by a task or when the system is idle, the slack period at the head of *ST-Q* is decremented as time progresses and is removed from *ST-Q* once it is exhausted. On the other hand, a running task needs to accumulate its computation time when it doesn't use up any slack periods. This implies that the execution is using up other tasks' budgets and, when the task is done, an additional slack period can be accumulated due to its reclamation and possible early completion.

2.4 Simulation Evaluation of VCS-EDF Algorithm

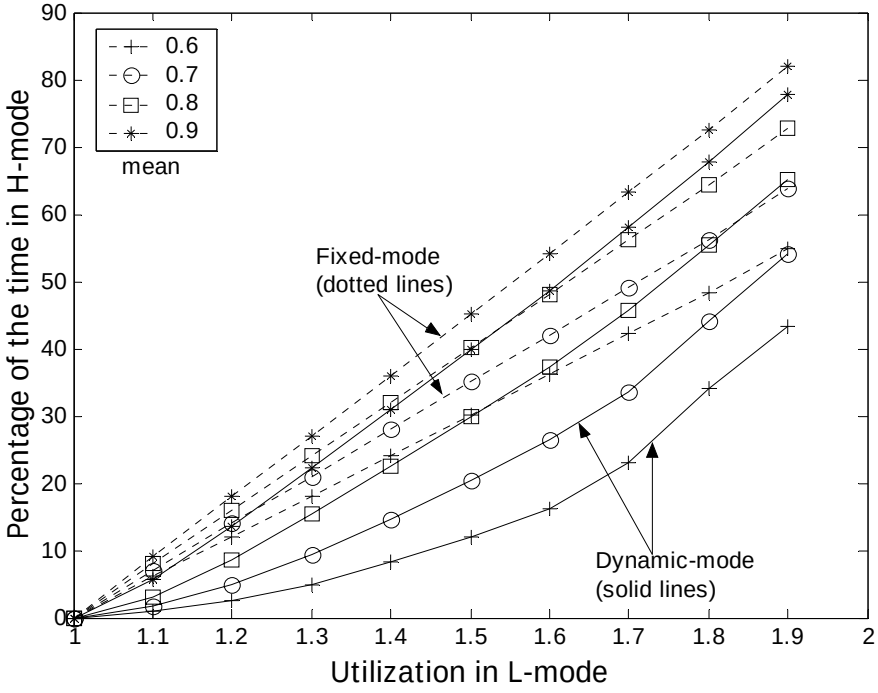
As described in Section 2.2.2, the static and dynamic voltage assignment can guarantee all tasks to meet their deadlines. Depending upon task characteristics, such as periods, WCETs, and arrival phases, the amount of time that the processor is running in *H-mode* may vary a lot. In this section, we evaluate their average energy consumption based on random task parameters in a system of 10 real-time tasks. For each test case, we measure the percentages of the time that the processor is in *H-mode* and *L-mode,* and is idle.

In addition to the schemes of static and dynamic voltage assignments with slack period reclamation, we also considered a fixed voltage assignment which follow the static assignment but with no slack period reclamation. Note that the fixed assignment minimizes the energy consumption when all tasks invoke the worst-case execution demands.

In our simulation, we first generate 10 random task periods in the range of 100 to 1000, and assign the task deadlines equal to their periods. We assume that V_H and V_L are fixed and the execution speeds are set to $\alpha_L=1.0$ and $\alpha_H=2.0$. $\alpha_L=1.0$ represents the normalized processing speed. Then, relatively the execution speed at $\alpha_H=2.0$ is double the speed of *L-mode*. The worst-case execution demands of the tasks are chosen in the range of α_L to α_H. The real task execution demand is then selected randomly such that the mean is in the range of 0.6 to 1.0 of the worst-case execution demand.



(a) Percentages of the time in H-mode for fixed and static voltage assignments



(b) Percentages of the time in H-mode for fixed and dynamic voltage assignments

Figure 2-2. The percentage of execution time in *H-mode*

In Figure 2-2(a) and (b), we show the evaluation results for execution demands ranging from *1* to *2*. The curves with dashed lines are the percentage measures from the fixed voltage assignment, whereas the solid lines are based on the static and dynamic voltage assignments. With the fixed mode assignment, the measured percentage in *H-mode* is approximately proportional to the average of the actual workload.

The curves in Figure 2-2(a) and (b) show some interesting trends. The difference of the measured percentage in *H-mode* execution between the fixed-mode and static-mode assignments increases with the workload. This indicates the effectiveness of slack period reclamation in a heavily loaded system. Also, the variance of task execution time plays an important factor, as an increased number of slack periods can be useful for a subsequent task execution. Furthermore, the dynamic voltage assignment is more effective when the workload is moderate (neither very low nor very high). This is due to two facts. The first is that the probability of finding short busy cycles diminishes as we increase the workload. On the other hand, when the workload is low, most tasks are started in *L-mode* in the fixed assignment. The voltage assignment made by the dynamic approach at the beginning of each busy cycle would not be substantially different from the one under the fixed approach.

We study the improvement of the proposed VCS-EDF scheduling in Figure 2-3 where the percentage reductions of the *H-mode* execution times over the fixed assignment approach are depicted. For instance, with $\alpha_L=1.0$, $\alpha_H=2.0$, $\rho=1.5$, and the average execution time is *0.7*of C_i, the percentages of the time in *H-mode* under the fixed-, static- and dynamic-mode assignments are 35.04%, 26.08%, and 20.28%, respectively. These percentages indicate that, compared to the fixed approach, the reductions of the time in *H-mode* reach (0.35-0.26)/0.35=0.26(26%) and (0.35-0.20)/0.35=0.43(43%) for static and dynamic approaches, respectively. The curves in the figure show that the dynamic voltage scaling can almost completely eliminate *H-mode* execution when the workload is low and the real task execution time is much smaller than the WCET. This significant result is due to (1) no pre-assignment of *H-mode* execution, and (2) effective slack

time reclamation. As we increment the workload, the percentage reductions shrink as the *H-mode* become necessary to meet deadline requirement, even though the reduction of the *H-mode* execution is substantial as shown in Figure 2-2. Figure 2-3 also reveals the difference between the static and dynamic approaches. As long as the workload is not high, there will be many short busy cycles and few long ones. The dynamic approach of making a greedy decision at the beginning of each busy cycle is justified given that there is no knowledge of subsequent task arrivals.

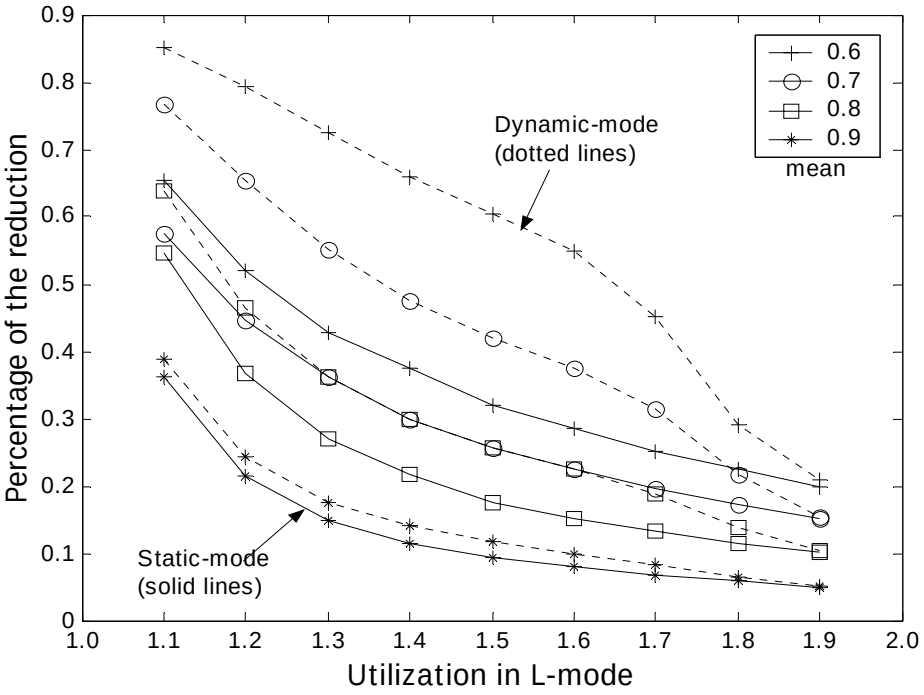


Figure 2-3. The percentage of the reduction for the time in *H-mode*

Using the percentage of the time spent in *H-mode, L-mode,* and in the idle state, we can measure the savings in power consumption. Assume that $V_T=0$ and that there is no power consumption in the idle state (in practice, idle power consumptions are extremely low, which justifies this assumption). The ratio of the average power consumption under the static approach to that of the fixed approach is

$$\frac{P_{CMOS}(\,static\,)}{P_{CMOS}(\,fixed\,)} = \frac{PT_H(\,static\,)V_H^2 + PT_L(\,static\,)V_L^2}{PT_H(\,fixed\,)V_H^2 + PT_L(\,fixed\,)V_L^2}$$
$$= \frac{PT_H(\,static\,) + PT_L(\,static\,)(\frac{\alpha_L}{\alpha_H})^2}{PT_H(\,fixed\,) + PT_L(\,fixed\,)(\frac{\alpha_L}{\alpha_H})^2} \qquad (2\text{-}5)$$

where $PT_x(y)$ is the percentage of execution time in *x-mode* under *y* approach. Similarly, we can compute the ratio $P_{CMOS}(dynamic)/P_{CMOS}(fixed)$. The ratios are plotted in Figure 2-4 for various workloads. In the extreme case, more than 25% of the energy can be saved by the dynamic approach over the fixed assignment. Note that the fixed assignment can outperform any global voltage setting, as it uses the optimal voltage assignment for each task, but does not incur any on-line adjustment.

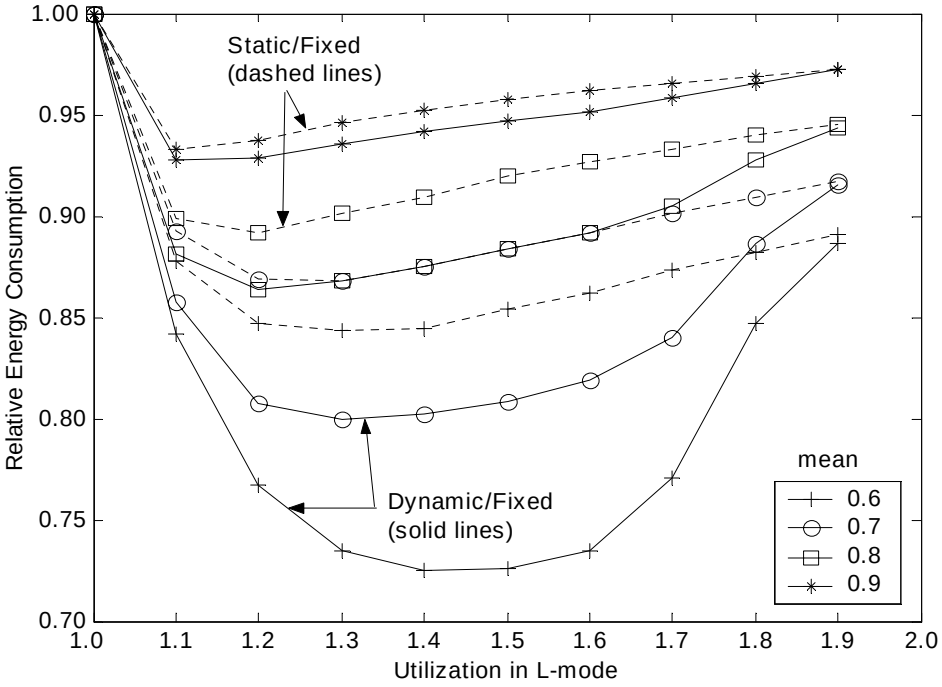


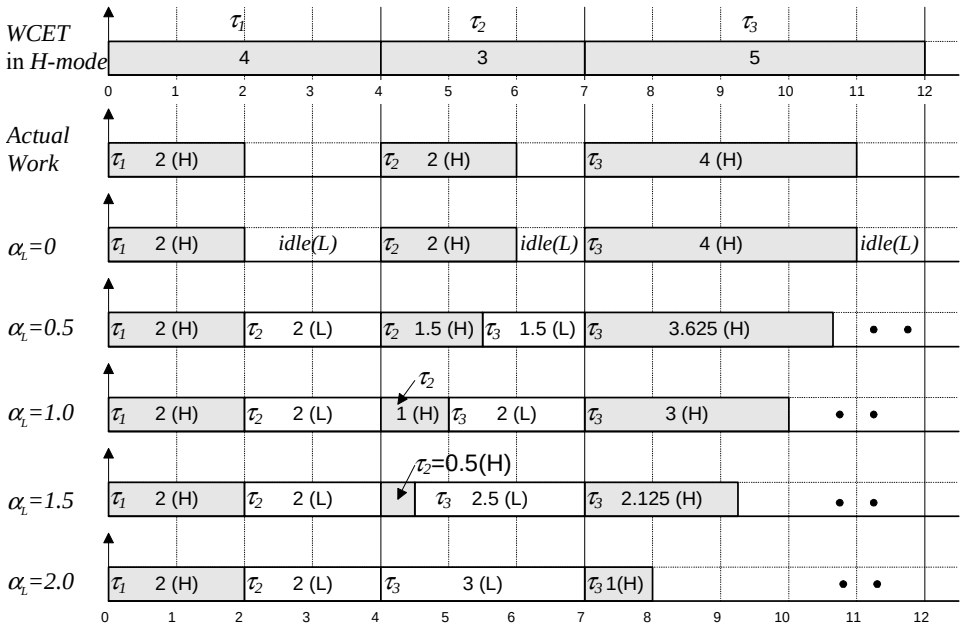
Figure 2-4. The relative energy consumption of fixed, static, and dynamic approaches

2.5 Optimal Voltage Setting in VCS-EDF scheduling

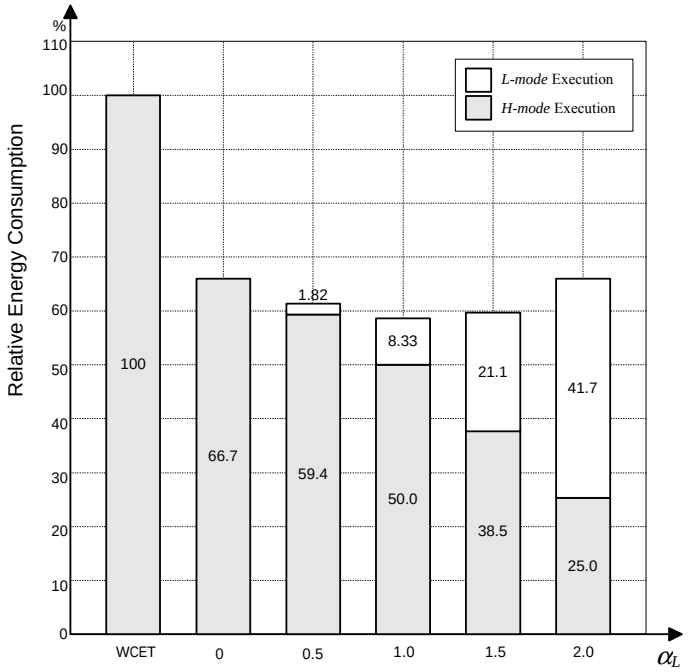
In the simulation experiments of Section 2.2.3, we assume fixed α_L and α_H (i.e. fixed V_L and V_H). Here, we address the problem of determining the optimal voltage settings given the characteristics of the given task set.

If the task execution demands are constant and every instance of task τ_i requires a fixed demand of C_i, then V_H should be set to V_H^* which leads to an execution speed α_H^* and $\rho/\alpha_H^* = 1$. The processor will have a full utilization of 1. There is no need to have a different $V_L < V_H^*$ for voltage scaling and any V_H less than V_H^* can result in task deadline violations. Also, any V_H higher than V_H^* will cause additional power consumption as the power consumption is a concave and increasing function of the supply voltage.

When the real task execution demands are random and are bounded by the worst-case demand, we still need to set $V_H = V_H^*$ to guarantee schedulability in the worst-case scenario. On the other hand, any settings of V_L are feasible under the VCS-EDF scheduling and lead to variant execution times in *H-mode* and *L-mode* as well as the average power consumption. In Figure 2-5, we have an example of variant V_L (α_L) vs. execution time and power consumption. We first adjust V_H such that the processor is fully utilized under the worst-case scenario. With the actual execution times, we can utilize the slack periods and run the tasks in *L-mode*. The execution sequences for different α_L's are shown in Figure 2-5 (a). With the assumptions that $V_T = 0$ and no power consumption at idle state, we can compute the relative energy consumption for a special case where the real workload is 66.7% of the worst-case execution demand. With a slow execution speed and low power consumption in *L-mode*, the processor must stay in *H-mode* for a substantial amount of time. Thus, there is only a marginal energy saving. On the other hand, if we set a high α_L and spend less time in *H-mode*, the total energy saving is again diminished as a high V_L must then be applied. As illustrated in Figure 2-5 (b), there exists an optimal setting of V_L, which balances the invocation of *H-mode* execution and the power consumption in *L-mode*.



(a) Task executions according to the speeds of two modes



(b) Relative energy consumptions for different α_L

Figure 2-5. An example of finding the optimal α_L

In Figure 2-6, we present simulation results of the optimal voltage settings using the DC characteristics of Motorola's MPC860 processor. At V_H=3.3V, the processor runs at an operating

frequency of 50 MHz and has a power consumption of 1.3 Watts. In addition, we assume that the processing speed is proportional to the operating frequency and that $V_T=0.7$ which is the typical threshold voltage in most CMOS circuits with 0.25-0.35μm technology. In our experiments, the worst-case execution demands of 10 tasks are chosen such that a full utilization is achieved at *H-mode.* Then, the supply voltage at *L-mode,* V_L, is varying from 0.7 to 3.3, and the real task execution demands are generated as a random number such that the mean demand is a proportion of the worst-case demand. Using the simulation results of the dynamic VCS-EDF scheduling, i.e. the percentages of time in *H*-mode and *L*-mode execution, we can compute the average energy consumption for various V_L from the equations (2-1), (2-2) and (2-3). The curves in Figure 2-6 indicate the optimal V_L and the possible power savings when a good candidate of V_L is chosen

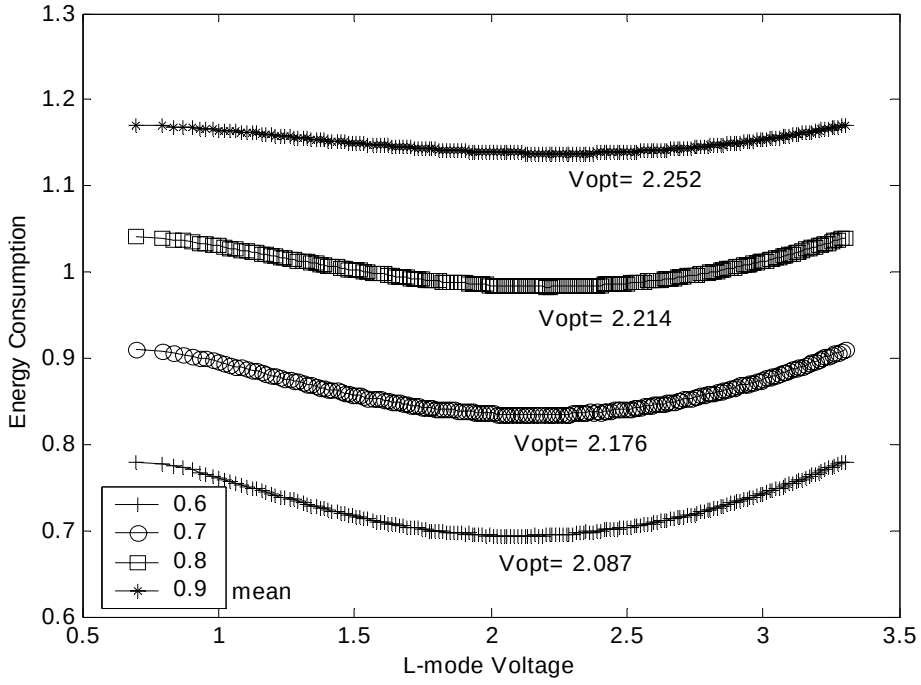


Figure 2-6. The optimal *L-mode* voltages V_L (V_H= 3.3 V, 1.3 Watts, and 50 MHz)

2.6 Conclusion

In this dissertation, we investigate the voltage-clock scaling problem in real-time embedded systems. Under EDF scheduling, we propose two scaling methods to reduce power consumption. Both methods utilize *L-mode* execution during task slack periods that come from the discrepancy between the worst-case and the real computation times. The static approach assigns a fixed voltage setting for each task in the offline phase such that the average execution time in *H-mode* is minimized and the schedulability condition is satisfied. At the online (run-time) stage, the tasks are dispatched according to the EDF algorithm and run at the assigned voltages except during the slack periods. The other method, a dynamic approach, does voltage assignment at the beginning of each busy cycle. Subject to the schedulability constraint, it assigns tasks to *L-mode* execution when the first instance of each task arrives in the busy cycle.

Our simulation results demonstrate that the proposed two approaches are quite effective compared to a fixed optimal task-base voltage assignment that does not perform any online adjustment. The dynamic approach can outperform the static one if no *H-mode* execution is invoked during short busy cycles. Finally, we discuss the selection of optimal voltage settings for given workloads.

CHAPTER 3
CONSTRAINED ENERGY BUDGET ALLOCATION

3.1 Introduction

Recently, the growth of mobile computing and communication devices such as laptop computers, cellular phones, and personal digital assistants (PDA's) has been explosive and the demands for embedded applications on those devices has become more sophisticated and intelligent along with the advanced technology in microprocessors. With accelerated miniaturization, portability, and complex functionality of the handheld devices (i.e. multimedia in portable phone), batteries having smaller size but longer-lifetime are indispensable. Processors are becoming increasingly power-hungry. Still, there is a big gap between the pace of current battery technology and energy demands of microprocessor systems [79]. For this reason, the field of power-aware computing has gained increasing attention over the past decade.

Simple techniques, such as turning off (or dimming) the screen while a system is idle and shutting down hard disks while it is not accessed is now commonly adopted in most portable device designs. However, in many cases, re-activation of hardware can take some time, and affect response time. Also, deciding when and which device should be shut down and woken up are often far from trivial. Another effective approach to power reduction is a technique called *Voltage-Clock Scaling* or *Dynamic-Voltage-Scaling* in CMOS circuit technology. Due to the quadratic relationship between supply voltage (V_{DD}) and clock frequency, a small reduction in voltage can produce a significant reduction in power consumption [24, 25]. However, increase in the circuit delay by lowering V_{DD} provides longer execution time and this leads to performance degradation in application response time and a failure to meet real-time deadlines.

If low energy consumption is a desirable feature in real-time embedded systems, voltage-clock scaling must cooperate with the task scheduling algorithms since the power-delay tradeoff property in low power design affects meeting the strict time-constraints of real-time systems. The execution of a high-priority at a low voltage and slow clock rate may cause a low-priority task to miss the deadline due to the additional delay from the execution of the high priority task.

While VCS has been a well-populated research area, power-aware system design has generally focused on minimizing total power consumption. For systems consisting of soft aperiodic tasks, the objective of minimizing power consumption will result in slow execution. On the other hand, in many cases, the battery capacity can be replenished or there is a finite mission lifetime. Minimizing power consumption that doesn't utilize all available energy may not lead to optimal system performance. A better power control strategy in such cases is to minimize the response times of soft real-time tasks, providing that the deadlines of hard real-time tasks are met and the average power consumption is bounded.

In this dissertation, we target battery-driven real-time systems, jointly scheduling hard periodic tasks and soft aperiodic tasks, whose battery capacity is bounded in the feasible range given by a set of tasks. The scheduling should guarantee meeting the task deadlines of hard real-time periodic tasks and achieve average response time of aperiodic tasks that are as low as possible. Under the constraints of a bounded energy budget, finding an optimal schedule for a task set should aim to satisfy both optimal power consumption and strict timing constraints simultaneously.

We first investigate the characteristics of energy demands of periodic and aperiodic tasks focusing an EDF scheduling exploiting the feature of VCS. Based on the energy requirement of mixed real-time tasks, we also propose a static scheduling for energy budget allocation, which determines the optimal two-level voltage settings of all tasks under bounded energy consumption, while guaranteeing that no deadline of any periodic task is missed and that the average response time of aperiodic tasks is minimized. The algorithm selects the voltage settings that have the

minimum average response time among the schedulable ones within a given energy consumption. To schedule aperiodic tasks, we adopt the Total Bandwidth Server, which was proposed by Spuri and Buttazzo and handles aperiodic tasks like periodic tasks within the reserved bandwidth such that it outperforms other mechanisms in responsiveness.

The rest of this chapter is structured as follows. Main concepts for designing a constrained energy budget allocation model are explained in section 3.2. In section 3.3, we outline the preliminary model with several assumptions. Then, we discuss profiling of energy consumption and processor utilization under bounded energy budget in section 3.4. Considering the characteristics described in section 3.4, energy allocation methods and an algorithm of voltage assignment are described in section 3.5. To illustrate the effectiveness of the proposed algorithm, we evaluate the performance in section 3.6. A short conclusion then follows in Section 3.7.

3.2 Design Concepts

As a key implementation vehicle of the real-time systems having mixed tasks under bounded energy consumption, the energy budget allocation model has the following design concepts at its core:

Scheduling hard and soft real-time tasks with power-awareness. An integration of hard and soft real-time tasks and multi-task soft real-time applications like multimedia application are currently attracting much attention.

Sharing a bounded energy budget. In many occasions, the battery capacity can be replenished or there is a targeted mission lifetime for a system that is confined by its batteries. In such cases, reducing power consumption cannot be the only objective of task scheduling. If batteries will be charged at a predictable instant, the tasks should be scheduled to make the best of the available energy as better performance as they can achieve. And if the application of the system is composed of mixed real-time tasks, the bounded energy budget should be shared in a way that the requirements are satisfied for both sets of tasks.

Switching voltage levels by Voltage-Clock Scaling. Based on the VCS technique, most of today's processor cores have been designed to operate at different voltage ranges to achieve different levels of energy efficiency. In this dissertation, we assume simple voltage settings that the processor in a real-time system can be dynamically configured in one of two modes: low-power-low-frequency mode and high-power-high frequency mode.

Profiling energy and bandwidth usages of tasks. To allocate a limited energy budget to tasks effectively such that the budget is fully utilized for better performance, the energy usage should be outlined at each task level in a system. In the dissertation, we have devised the VCS-EDF scheduling to profile energy demands. The processor's operation at different energy performance levels leads the energy usage to be profiled according to real-time tasks' execution requirements. In addition to profiling the energy usage, the processor utilization must be also profiled to guarantee the schedulability of tasks. Due to the property of power-delay tradeoff in the voltage-clock scaling, processor requires prolonged execution time in low-power-low-frequency mode, comparing to the one in high-power-high-frequency mode. Obviously, the variation of a given execution time according to the determination of operating modes (frequencies) affects processor utilization and the schedulability of EDF scheduling.

Guaranteeing no missed deadline for hard periodic tasks. In real-time systems that have mixed tasks, aperiodic tasks are usually served in the background with respect to hard periodic tasks in order not to jeopardize the schedulability of hard tasks and served by an aperiodic server to improve responsiveness. In this dissertation, the schedulibility of hard periodic tasks is guaranteed by the isolation of bandwidth between periodic and aperiodic tasks.

Ensuring fast responsiveness for aperiodic tasks under a given energy budget. If systems consist of only soft aperiodic tasks, slow execution for these tasks to minimize power consumption may result in unrestrained response time. On the contrary, if the objective of minimizing aperiodic tasks' response time is given for scheduling mixed tasks under a limited energy budget, an appropriate amount of energy should be adjusted to aperiodic tasks to ensure

responsiveness as fast as they can get within the available energy budget for them. Due to the inverse-relationship between energy consumption and utilization, the responsiveness can be also affected by sharing of the total energy budget to periodic tasks. The increased energy allocation to aperiodic tasks leaves periodic tasks less energy budget and this causes a delayed response time in return for the increased utilization of periodic tasks.

Expanding the life span of a battery-driven real-time embedded system. Since reduced power consumption promises the longer lifetime of the system, developing an effective scheduling scheme that consumes much less power is also preferred to schedule mixed tasks under a power consumption constraint.

3.3 System Model for Energy Sharing

For the targeted real-time systems, tasks may arrive periodically and have individual deadlines that must be met. Or they can be aperiodic and can accrue computation values, which are inversely proportionate to their response times. Under a given bound on energy consumption, we build a system model and make several assumptions as follows.

3.3.1 Schedule for Periodic Tasks

For Earliest Deadline First (EDF) scheduling, a periodic task τ_i is modeled as a cyclic processor activity characterized by two parameters, T_i and C_i, where T_i is minimum inter-arrival time between two consecutive instances and C_i the worst-case execution time (WCET) of task τ_i. The EDF scheduling algorithm always serves a task that has the earliest-deadline among waiting tasks. The following assumptions are analogous to assumptions made in real-time scheduling theory.

- Tasks are independent: no task depends on the output of any other task.
- The worst-case execution demand of each task τ_i, i.e., C_i, is known. The actual execution demand is not known a priori and may vary from one arrival instance to the other.
- The overhead of the scheduling algorithm is negligible when compared to the execution time of the application.

3.3.2 Schedule for Aperiodic Tasks

Infinite number of soft aperiodic tasks $\{J_i \mid i=0,1,2,\ldots\}$ are modeled as sporadic processor activities represented by two parameters, λ and μ, where λ is average inter-arrival time between two consecutive aperiodic instances and μ the average worst-case execution time of task J_i.

Aperiodic tasks are scheduled by *total bandwidth server* algorithm that makes fictitious but feasible deadline assignment based on the available processor utilization guaranteed by the isolation of bandwidth between periodic and aperiodic tasks. In the TBS algorithm, the *k*-th aperiodic request arrives at time $t = r_k$, a task deadline

$$d_k = \max(r_k, d_{k-1}) + \frac{C_k}{U_A}, \tag{3-1}$$

is assigned, where C_k is the execution time of the request and U_A the allocated processor utilization for aperiodic tasks. By definition $d_0=0$. The request is then inserted into the ready queue of the system and scheduled by the EDF algorithm, as any other periodic instances or aperiodic requests already present in the system. In Figure 3-1, an example of deadline assignment by Total Bandwidth Server is shown. Given utilization U_A= 0.25 for aperiodic task of τ_0, the deadlines of the instance J_1, J_2, and J_3, that are requesting execution time of 1, 2, 1, are assigned as d_1= 4, d_2= 8, d_3= 7, respectively, by the equation (3-1) and scheduled with periodic tasks, τ_1 and τ_2.

Note that the assignment of deadlines is such that in each interval of time, the processor utilization of the aperiodic tasks is at most U_A. Hence, a set of periodic tasks with utilization factor $U_P = \sum_{i=1}^{n} C_i / T_i$ and a TBS with a bandwidth U_A is schedulable by EDF if and only if $U_A + U_P \leq 1$. Comparing to the other scheduling algorithms for aperiodic tasks, the TBS algorithm has a very simple implementation complexity and shows very good performance in average response time [40].

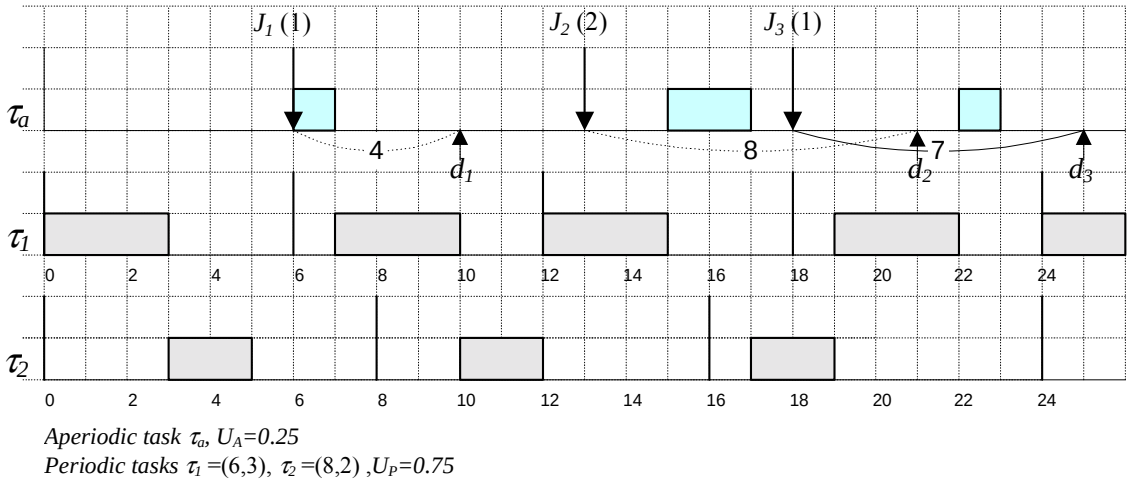


Figure 3-1 An example of the deadline assigning algorithm in Total Bandwidth Server

3.3.3 Voltage-Clock Scaling

3.3.3.1 Voltage Switching

We assume voltage switching consumes negligible overhead. This is also analogous to the assumption made in classical real-time scheduling theory, that preemption costs are negligible. Voltage switching typically takes a few microseconds. In fact, a bound of the total overhead can be calculated by considering the number of task arrivals and departures since voltage switches are only done at task-dispatch moments.

3.3.3.2 Two Voltage Levels

The system operates at two different voltage levels. Ideally, a variable voltage processor that has continuous voltage and clock setting in the operational range is available as explained in Table 1-1. We assume a simple setting arrangement that the processor in a real-time system can be dynamically configured in one of two modes: *low-voltage* (*L) -mode* and *high-voltage* (*H)-mode*. In *L-mode,* the processor is supplied with a low voltage (V_L) and runs at a slow clock rate. Thus, task execution may be prolonged but the processor consumes less energy. On the other hand, the processor can be set in *H-mode,* i.e. be supplied with a high voltage (V_H) and run at a fast clock rate, in order to complete tasks sooner at the expense of more energy consumption. The

operating speeds at *L-mode* and *H-mode* are denoted as α_L and α_H, respectively, in terms of some unit of computational work. Depending upon the voltage setting for task τ_i, the worst-case execution time is C_i/α_L or C_i/α_H.

3.3.4 Bounded Energy Consumption

In battery-powered embedded systems, it is often equally important to control power consumption to extend the lifetime of the battery and to enhance system performance. Given that the battery can be replenished or the mission lifetime is limited, we may assume that the available capacity can safely be consumed during a predefined interval of operation. Thus, an average power consumption rate or energy budget can be set to the ratio of available capacity to the target operation interval. Also, it is possible to communicate with the battery such that the system and its scheduler can know the current status of the battery capacity. One of the mechanisms for doing this is the *smart battery system* (SBS), which has been now actively standardized and introduced to battery-driven systems [79]. In this dissertation, for the information of bounded energy budgets, we assume battery-driven real-time systems have such kind of mechanism. And, we assume the embedded system, whose processor is the major factor of the energy consumption

3.4 Profiling Energy and Utilization Usages in a Real-time Embedded System

For all real-time tasks, the available energy consumption is confined to a given energy budget called E_C, which has to be shared among periodic and aperiodic tasks. Let E_P and E_A are the energy budget allocated to periodic tasks and aperiodic tasks, respectively. The voltage-clock scaling problem is to find voltage settings for both periodic and aperiodic tasks such that

- all periodic tasks are completed before their deadlines and have energy consumption less than E_P.
- all aperiodic tasks can attain the minimal response times while consuming energy less that E_A.
- $E_P + E_A \leq E_C$.

3.4.1 Periodic Tasks

Assume that, for periodic task τ_i, m_i is the voltage setting determined between the two possible modes, i.e. *L-mode* and *H-mode* and $\alpha_i(m_i)$ is the speed of task τ_i at mode m_i. Given m_i for all of periodic task τ_i, the energy demand for periodic task of E_p is

$$E_P(m_i) = \sum \frac{1}{\alpha_i(m_i)} \frac{\overline{C_i}}{T_i} p(m_i) \tag{3-2}$$

where $p(m_i)$ is the power consumption at mode m_i, $\overline{C_i}$ the average execution time of task τ_i In addition, the worst-case utilization is given by

$$U_P(m_i) = \sum \frac{1}{\alpha_i(m_i)} \frac{C_i}{T_i} \tag{3-3}$$

If m_i is *H-mode* for all periodic tasks, the processor runs at a fast clock rate all the time, thereby minimizing the utilization. The maximum energy demand for the tasks is represented as

$$max\, E_P = \sum \frac{1}{\alpha_H} \frac{\overline{C_i}}{T_i} p_H \tag{3-4}$$

and its utilization becomes

$$min U_P = \sum \frac{1}{\alpha_H} \frac{C_i}{T_i} \tag{3-5}$$

On the contrary, if m_i is *L-mode* for every periodic task τ_i, the processor runs at a slow clock rate all the time such that the utilization is maximized, but consumes the minimum possible energy. For the sake of schedulability, the tasks should be scheduled in such a way that the utilization is less than unity Therefore, we define *min* E_P as an energy demand when there exists a set of $\{m_i\}$ so that the worst-case utilization

$$U_P(m_i) = \sum \frac{1}{\alpha(m_i)} \frac{C_i}{T_i} \le 1 \text{ and } E_P(m_i) = \sum \frac{1}{\alpha(m_i)} \frac{\overline{C_i}}{T_i} p(m_i) \text{ is minimized.} \tag{3-6}$$

In Figure 3-2, we describe the relationship between energy consumption and utilization for a set of periodic tasks. The maxima and minima are denoted as *max* E_P and *min* E_P for the

energy and *max* U_P and *min* U_P for the utilization, respectively. Again, *min* E_P follows equation (3-6). Regarding the feasibility of the energy constraint and the worst-case utilization, E_C must be greater than *min* E_P and *min* U_P should be no greater than 1. By definition, if *min* U_P is greater than unity with all *H-mode* executions, it is impossible to find voltage settings to ensure that all tasks meet their deadlines. If *max* U_P is less than 1, the tasks are schedulable with all *L-mode* assignments and energy consumption can never be less than *min* E_P. In the case, *max* U_P becomes

$U_P(L) = \sum \frac{1}{\alpha_L} \frac{C_i}{T_i}$ and *min* E_P does $E_P(L) = \sum \frac{1}{\alpha_L} \frac{\overline{C_i}}{T_i} p_L$.

If energy budget E_P is given in the range from *min* E_P to *max* E_P, $U_P^{available}$ is the available utilization corresponding to the allocated energy budget E_P. And, by searching a set of voltage settings meeting the given energy budget and schedulability, energy demand and utilization for periodic tasks are determined as $E_P(m_i) \le E_P$ and $U_P(m_i) \ge U_P^{available}$, respectively.

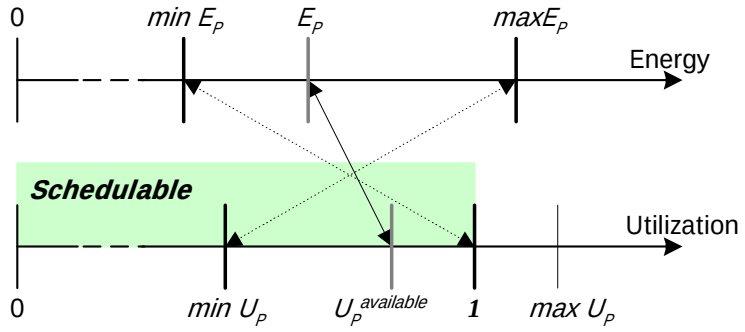


Figure 3-2 The relationship between power consumption and utilization for a set of periodic tasks

3.4.2 Aperiodic Tasks

Denote by m_A the voltage setting determined between the two possible modes for aperiodic tasks, which have the average inter-arrival time of λ and the average worst-case execution time of μ. If all of them are assigned in mode m_A and the power consumption at mode m_A is $p(m_A)$, the energy consumption and utilization of them are

$$E_A(m_A) = \frac{1}{\alpha(m_A)} \frac{\mu}{\lambda} p(m_A), \tag{3-7}$$

$$U_A(m_A) = \frac{1}{\alpha(m_A)} \frac{\mu}{\lambda}, \tag{3-8}$$

respectively.

Also, if all of them are assigned in *L-mode* or *H-mode*, they demand minimum energy *min* E_A or *max* E_A given by the following equations

$$min\, E_A = \frac{1}{\alpha_L} \frac{\mu}{\lambda} p_L \text{ and } max\, E_A = \frac{1}{\alpha_H} \frac{\mu}{\lambda} p_H \tag{3-9}$$

having the utilization

$$min U_A = \frac{1}{\alpha_L} \frac{\mu}{\lambda} \text{ and } max U_A = \frac{1}{\alpha_H} \frac{\mu}{\lambda}. \tag{3-10}$$

3.4.3 Energy Budget Allocation

While the constraint $E_P + E_A \leq E_C$ must be satisfied, we can decide how processor utilization, task scheduling, and task response time are affected. From the viewpoint of utilization, the more utilization is available for aperiodic tasks, the shorter the deadlines that are assigned to them by the deadline assignment of equation (3-1). This assigns higher priorities to them in EDF scheduling such that they can get a faster response. To give more utilization to aperiodic tasks, the utilization of periodic tasks must be shrunk and it can be done by assigning more tasks to *H-mode*, but requires more energy consumption. Since the total energy budget is bounded, the energy budget left to aperiodic tasks will be reduced. As a result, the aperiodic tasks must be run in a low voltage mode and their response times will be extended.

Likewise, from the viewpoint of the energy budget, the portion assigned in *H-mode* for aperiodic tasks should be maximized within an assigned energy budget E_A to get faster responsiveness. But, as before, if the energy demand of aperiodic tasks is increased, the energy available for periodic tasks will be decreased. Consequently, the available utilization for aperiodic

tasks will be decreased due to the increased execution time of periodic tasks, which may result in degradation in responsiveness.

Eventually, to get both schedulability and fast responsiveness under a bounded energy budget, an effective scheduling and energy allocation scheme is needed for jointly scheduling hard periodic and soft aperiodic tasks. The scheduling should address the concern of the trade-off between utilization and energy consumption as shown in Figure 3-2.

3.5 Constrained Energy Allocation using VCS-EDF Scheduling

In this section, we describe an energy allocation scheme, which allocates bounded energy budget to periodic and aperiodic tasks based on VCS-EDF scheduling, meeting the requirements of real-time tasks, i.e. to meet deadlines of periodic tasks and to get faster average response time for aperiodic tasks. Given an energy budget E_C, considering the feasible range of energy demand determined by tasks, it finds voltage settings for periodic tasks and the execution portion in *H-mode* and *L-mode* to the worst-case execution time for aperiodic tasks under a bounded energy budget.

3.5.1 Energy Allocation Factors

Suppose that E_P and E_A can be allocated in the range of $[min\ E_P, max\ E_P]$ and $[min\ E_A, max\ E_A]$, respectively, $E_{max} = max\ E_P + max\ E_A$, and $E_{min} = min\ E_P + min\ E_A$. If the bounded energy consumption budget is given as E_C, E_C must fall into the range $E_{min} \le E_C \le E_{max}$ where $min\ E_P$, $min\ E_A$, $max\ E_P$, and $max\ E_A$ are as defined in equations (3-4), (3-5), and (3-9) to a given set of tasks. Then, voltage settings must be determined such that the energy consumption satisfies the constraint of $E_P + E_A \le E_C$, while guaranteeing the schedulability of periodic tasks and minimizing average response time for aperiodic tasks. For ease of explanation, we define $E_{diff} = E_{max} - E_{min}$.

Let β and γ be energy allocation factors of aperiodic and periodic tasks, given by $0 \leq \beta \leq 1$ and $0 \leq \gamma \leq 1$, respectively. Then, the energy budgets E_P and E_A allocated to them are represented as

$$E_P = min\ E_P + \beta(max\ E_P - min\ E_P), \tag{3-11}$$

$$E_A = min\ E_A + \gamma(max\ E_A - min\ E_A), \tag{3-12}$$

respectively.

Suppose $\Delta a = (\max E_A - \min E_A)$, $\Delta p = (\max E_P - \min E_P)$, and $\Delta c = E_C - (\min E_A + \min E_P)$, respectively, then the inequality $(E_P + E_A \leq E_C)$ becomes

$$\gamma\Delta a + \beta\Delta p \leq \Delta c.$$

Hence, β and γ are determined by

$$0 \leq \beta = \frac{\Delta c - \gamma\Delta a}{\Delta p} \leq 1, \qquad \frac{\Delta c - \Delta p}{\Delta a} \leq \gamma \leq \frac{\Delta c}{\Delta a}. \tag{3-13}$$

respectively. The choice of γ determines β, and *vice versa*, and also determines E_A and E_P by equations (3-11) and (3-12).

If $\gamma=0$ and $\gamma=1$, energy $min\ E_A$ and $max\ E_A$ are assigned to aperiodic tasks, i.e. assigning all aperiodic tasks in *L-mode* and *H-mode*, respectively. If γ is 0.6, energy assigned for aperiodic tasks becomes $(min\ E_A + 0.6\Delta a)$. Unlike voltage settings for periodic tasks, which are decided on the basis of a task, the running mode for aperiodic tasks are determined by the fraction in *H-mode* and *L-mode*. If the fraction assigned to *H-mode* is x_H, then that assigned to *L-mode* becomes $(1-x_H)$. The energy consumption needs to be bounded by the budget, and so

$$\frac{x}{\alpha_H}\frac{\mu}{\lambda}p_H + \frac{(1-x)}{\alpha_L}\frac{\mu}{\lambda}p_L \leq E_A. \tag{3-14}$$

Similarly, the execution time of an aperiodic task is determined according to the voltage modes and the deadline assigned in equation (3-1) is adjusted. As for responsiveness, the greater the fraction of the processor utilization that is given to aperiodic tasks, the better is the

responsiveness expected under the TBS algorithm, because shorter deadlines are assigned to them. Under energy budgets of E_C and E_P, the utilization for aperiodic tasks will be increased if the voltage settings are determined to allocate more *H-mode* to periodic tasks within the energy budget such that it can minimize the utilization $U_P\ (m_i)$ and make an increase in $U_A^{available}$. We therefore have a constrained optimization problem to determine the optimal voltage settings, maximizing *H-mode* execution, within the constraint of budget E_P and guaranteeing that no deadline of any periodic task is missed.

The optimization problem to find voltage settings for periodic tasks can be stated as follows: Pick the task subsets *H* and *L* for voltage settings of *H-mode* and *L-mode* such that

- $H \cup L = \{\tau_1, \tau_2, ..., \tau_n\}$
- $H \cap L = \varnothing$
- $U_P(m_i) = \sum \frac{1}{\alpha(m_i)} \frac{C_i}{T_i}$ is minimized

subject to the well-known sufficient condition[1] for the schedulability of periodic tasks under EDF, i.e.,

$$\frac{1}{\alpha_H} \sum_{i \in H} \frac{C_i}{\min(T_i, D_i)} + \frac{1}{\alpha_L} \sum_{i \in L} \frac{C_i}{\min(T_i, D_i)} \le 1 \qquad (3\text{-}15)$$

and the energy consumption constraint of

$$\frac{p_H}{\alpha_H} \sum_{i \in H} \frac{\overline{C_i}}{\min(T_i, D_i)} + \frac{p_L}{\alpha_L} \sum_{i \in L} \frac{\overline{C_i}}{\min(T_i, D_i)} \le E_P .$$

This optimization problem can be treated equivalently to the decision problem of the subset sum, which is *NP*-complete. Consequently, efficient search heuristics, e.g., branch-and-bound algorithms, should be employed to find a solution if *n* is large.

[1] The condition is also necessary if $D_i \ge T_i$ for all *i*.

3.5.2 Algorithm for Energy Budget Allocation

We describe here the algorithm for the bounded energy allocation explained in the previous section. The algorithm outputs the energy allocation factors, β and γ, voltage settings for periodic tasks, $\{m_i\}$, and the percentage of *H-mode* assignment for aperiodic tasks, x_H.

The Algorithm of VCS-EDF Scheduling Under Bounded Energy Consumption E_C

1. Compute *min* E_P, *max* E_P, *min* E_A, *max* E_A, E_{max}, and E_{min}.
2. If E_C is less than E_{min} = *(min* E_P + *min* E_A*), there is not enough energy to execute the workload.*
3. If E_C is in the range of $E_{min} \le E_C \le E_{max}$, compute the range of γ and β, E_A and E_P, accordingly.
4. For each γ in the range of $0 \le \gamma \le 1$, execute the following steps
   - (4a) Compute β, E_A and E_P,
   - (4b) Find $\{m_i\}$, which satisfies $\sum \frac{\overline{C_i}}{T_i} \cdot \frac{1}{\alpha(m_i)} \cdot p(m_i) \le E_P$ and that $U_P(m_i) = \sum \frac{C_i}{T_i} \cdot \frac{1}{\alpha(m_i)}$ is minimized, where m_i is voltage setting either in *H-mode* or *L-mode* for periodic task τ_i, using simple search or branch-and-bound algorithms.
   - (4c) Compute $U_A^{available} = 1 - U_P(m_i)$.
   - (4d) Given E_A, find x_H, the fraction of execution in *H-mode* for aperiodic tasks, and $(1-x_H)$ the fraction in *L-mode*.
   - (4e) Applying the TBS algorithm for the deadline assignment $U_P(m_i)$ and $U_A^{available}$ computed in step (4b) and (4c), respectively, run VCS-EDF scheduling in voltage settings $\{m_i\}$ for periodic tasks and x_H and $(1-x_H)$ for aperiodic tasks.
5. Find γ having the minimum average response time from the result of the scheduling in step 4.
6. The value of γ determined in step 5 is selected for energy allocation, which gives the best performance for aperiodic tasks, x_H for running the aperiodic tasks in *H-mode* and $\{m_i\}$ for voltage settings of the periodic tasks are determined accordingly.

Figure 3-3 The algorithm for constrained energy budget allocation

3.6 Simulation Evaluation

We analyze here the properties of sharing the bounded energy budgets between periodic and aperiodic tasks based on VCS approach and evaluate the VCS-EDF scheme to schedule mixed real-time tasks. For the power consumption and speed settings, Motorola's PowerPC 860 processor is used for our simulation, which can be operated in a high-performance mode at 50MHz and with a supply voltage of 3.3V, or a low-power mode at 25MHz and with an internal voltage of 2.4V such that V_H and V_L are fixed to V_H=*3.3* and V_L=*2.4*. The power consumption in the high-performance mode is 1.3 Watts (p_H), as compared to 241mW (p_L) in the low-power mode. The clock rate at high voltage is 100% greater than at low voltage: α_H=*2.0* and α_L=*1.0*.

A simulation study is performed to address the improvement of task execution time with extra available energy. In other words, the system is assumed to possess enough energy to complete the tasks and meet the deadline requirements. In addition, there is extra energy that can be allocated to improve the response time of aperiodic tasks. Our immediate objective of the simulation study is to see how the response time can be reduced through a proper voltage setting. Furthermore, this extra energy can be allocated to periodic tasks such that the processor utilization reserved for periodic tasks is reduced. This leads to a reduction of deadline assignment in the total-bandwidth scheduling scheme. On the other hand, the extra energy can be consumed by aperiodic tasks that can result in a first-order effect in the reduction of response time.

In our simulation, we first generate 10 random task periods in the range of 100 to 1000 and set the task deadlines equal to their respective periods. The worst-case execution demands of the tasks are randomly chosen such that, for each simulation case, no deadlines need be missed and the resultant utilization is set to U_p *(L)*=0.8, 1.0, or 1.2, respectively. For aperiodic tasks, we adopt the exponentially distributed execution time with an average μ equal to 45. Then we let the inter-arrival time be exponentially distributed with mean of between 450 (10% workload, i.e. U_A

(L)=0.1) and 112.5 (40%, i.e. U_A *(L)*=0.4). The energy budget E_C is set at each of several energy levels in the range from (E_{min}+0.6E_{diff}) to (E_{min}+ E_{diff}).

To get fast responsiveness, how much energy budget can be allowed to periodic and aperiodic tasks, respectively? Over various γ's and constraint energy budgets, we obtain the average response times of aperiodic tasks from the simulation and plot them in Figure 3-4. Regardless of increase in the energy budget, Figure 3-4 reveals a trend of reduction in average response time of aperiodic tasks as γ increases. The average response time does not show always a monotonic decrease with an increase in γ. In some regions, it has an abrupt increase or is flat over increasing γ. This occurs especially when E_C =(E_{min}+0.6E_{diff}) or E_C =(E_{min}+0.7E_{diff}).

Note that when we increase γ, aperiodic tasks are invoked more in high-voltage high-speed execution. This results in a reduced CPU utilization, i.e. the utilization required by aperiodic tasks under the voltage setting. On the other hand, as β is reduced, the energy allocated to the periodic tasks decreases which leads to an increase in U_P (m_i) and a decrease in $U_A^{available}$. The two reductions, one on the demand to complete aperiodic tasks and the other one on the available utilization for aperiodic tasks, can have a profound impact on the response times. Let the CPU utilization required be denoted as U_A^{real} and we show the ratio of $U_A^{available}$ to U_A^{real} in Figure 3-5. For instance, with E_C =(E_{min}+0.6E_{diff}) and γ=1.0 in Figure 3-5 (a), there still exists extra energy to be assigned to periodic tasks (β >0) and an optimal voltage setting is obtained which leads to U_P (m_i)=0.55 and $U_A^{available}$ = 0.45. On the other hand, U_A^{real} is reduced to 0.15 as we increase γto1.0. A ratio of 3 is then obtained and plotted in the Figure.

It is interesting to observe that, whenever the ratio is flat in Figure 3-5, the average response times have uneven decreases in Figure 3-5. . In fact, as long as the ratio of $U_A^{available}$ to U_A^{real} continues to increase, the processor possesses greater capacity to complete aperiodic tasks and the response time drops. In contrast, there would be a monotonic decrease in response time if the ratio were flat as we increase γ.

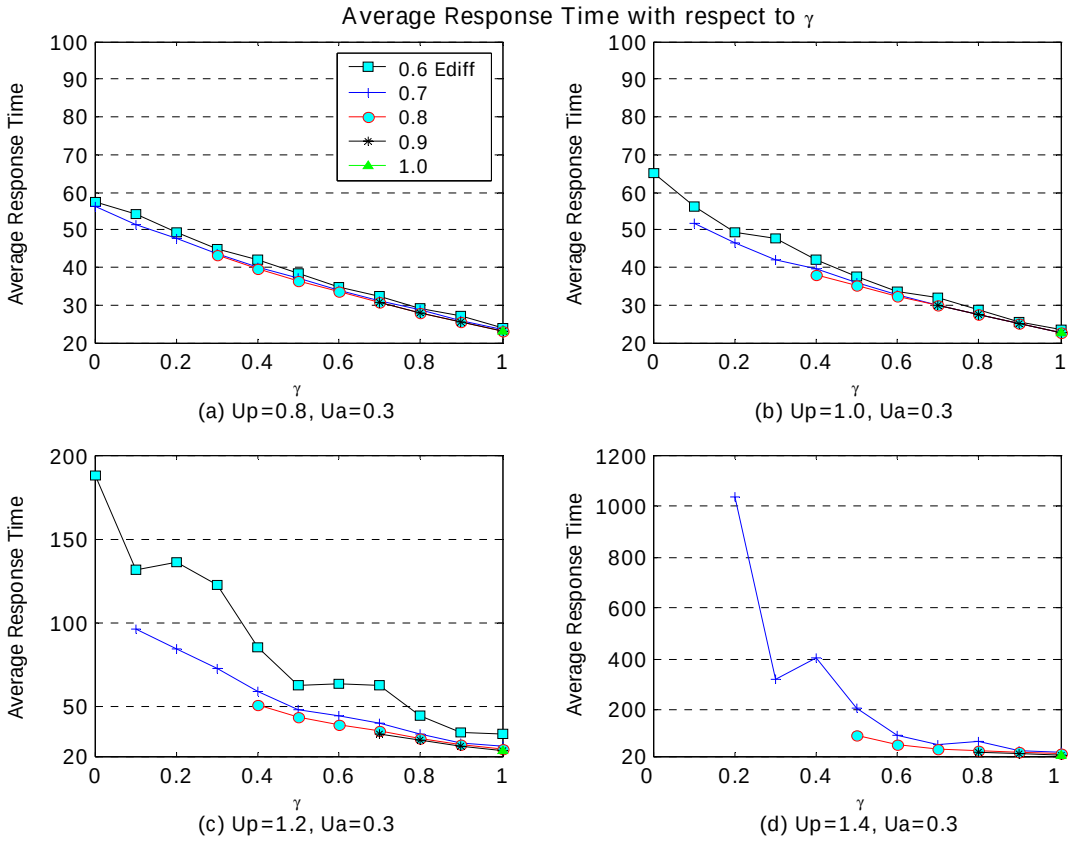


Figure 3-4 The responsiveness to the energy allocation of aperiodic tasks

The other interesting observation in Figure 3-5 is that utilization ratios are not available for all of γvalues. It indicates that the possible choices of γonly exist in the range where the plots are shown. This is also evidenced in equation (3-14) and is originated from the definition of γ, in which the minimum value of γmeans the percentage of energy available for aperiodic tasks after periodic tasks take energy budget as much as they can.

From these results, to get fast responsiveness of aperiodic tasks, the greater portion of the energy budget should be allocated to aperiodic tasks, and then voltage settings of periodic tasks need to be determined within the energy budget remaining for them. Note that the way we formulate the minimal energy budget is based on the schedulability for periodic tasks and ensuring no CPU starvation for aperiodic tasks. If the energy budget is below this minimum, aperiodic tasks will incur much longer response times.

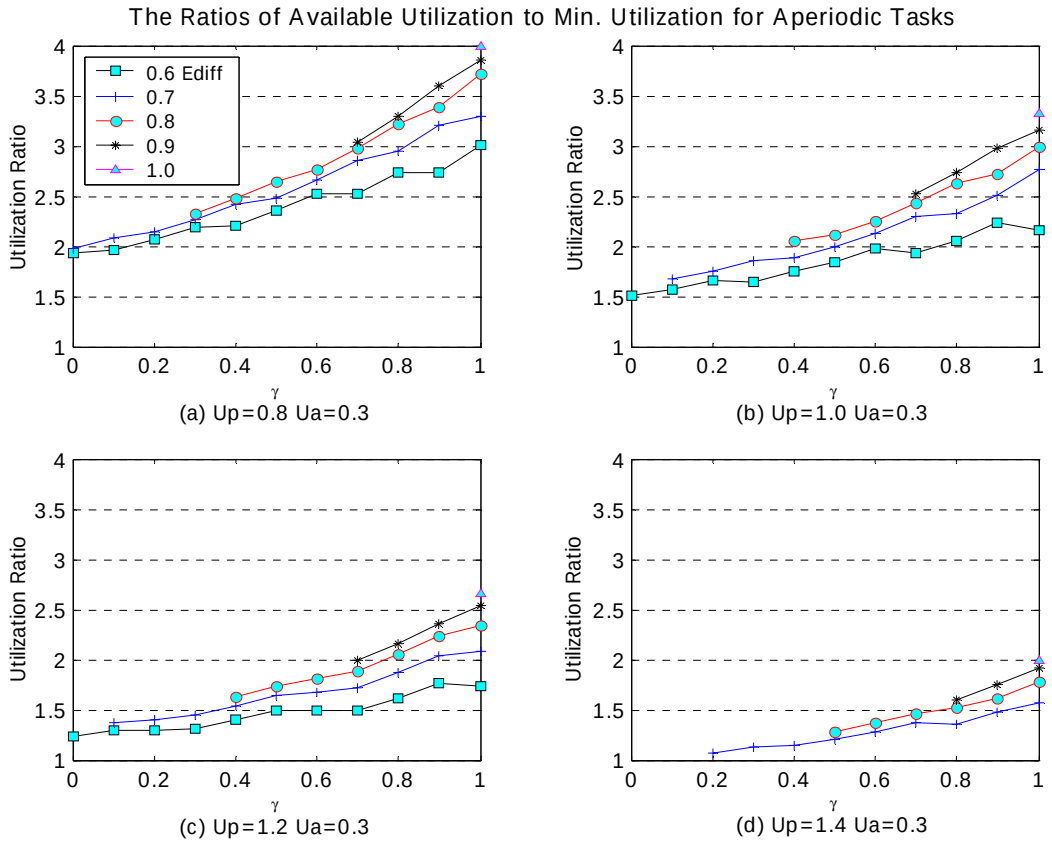


Figure 3-5 The ratios of available utilization to the minimum utilization for aperiodic tasks

To reveal the causes that lead to the flat regions in Figure 3-5, we now investigate how the energy budget is allocated to periodic and aperiodic tasks, respectively. In Figure 3-6, we show the energy sharing as percentages of allocated energy E_P for periodic and E_A for aperiodic tasks to the maximum energy demand, E_{max}, that is the maximal energy consumption by a given task set. The plots in Figure 3-6 (a)~(c) cover the case when E_C is bounded to (E_{min}+0.6E_{diff}). But in Figure 3-6 (d), we plot the energy percentages under E_C = (E_{min}+0.7E_{diff}) unlike the ones for other periodic workloads. The reason is the energy budget (E_{min}+0.6E_{diff}) is too low to select proper voltage settings making the given set of tasks schedulable under the periodic workload of U_P =1.4.

When a set of periodic tasks can make the most of the given energy budget E_P, i.e. E_P $(m_i) \leq E_P$, E_P (m_i) is determined by the chosen set of voltage settings, m_i, in the VCS-EDF

algorithm subject to requirements imposed by the need to maintain schedulability. Thus, there is a small discrepancy in energy consumption between E_P and E_P (m_i).

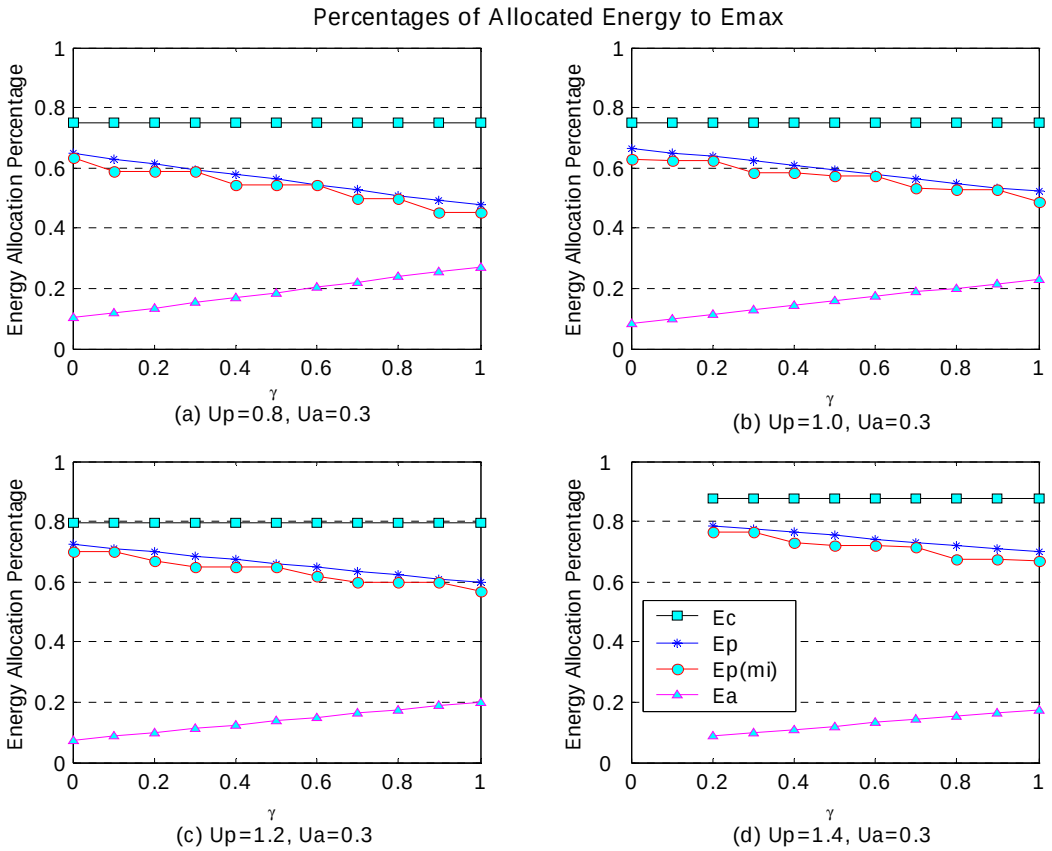


Figure 3-6 Energy allocation percentage to the maximum energy demand

Over several regions of γ, E_P (m_i) is kept at the same level even if E_P is decreasing, while being less than E_P. In other words, the same voltage settings are selected for different E_P's. For all the possible combinations of voltage settings, if we sort them in descending/ascending order according to energy demands, a discontinuity in energy demands exists between any two sets of voltage settings adjacent in the sorted list. Let the discontinuity in energy demand be an energy gap. Then, even if there is a small amount of change in energy budget E_P, it cannot change voltage settings unless it jumps up/down any energy gap between adjacent energy levels. However, if the number of periodic tasks is getting bigger, the flatness in Figure 3-6 will be

reduced because of the fine energy gap between adjacent energy levels of discontinuous voltage settings.

It should be noticed the big drops in the response times of aperiodic tasks occur when the voltage settings of periodic tasks result in a energy allocation E_P (m_i) that is very close to the available budget E_P. For instance, at γ=0.1 and 0.5 of Figure 3-6 (c), the settings lead to a little reduction of $U_A^{available}$ which, combining with the decrease of U_A^{real}, bring about a considerable decrease in the task response time of Figure 3-4 (c).

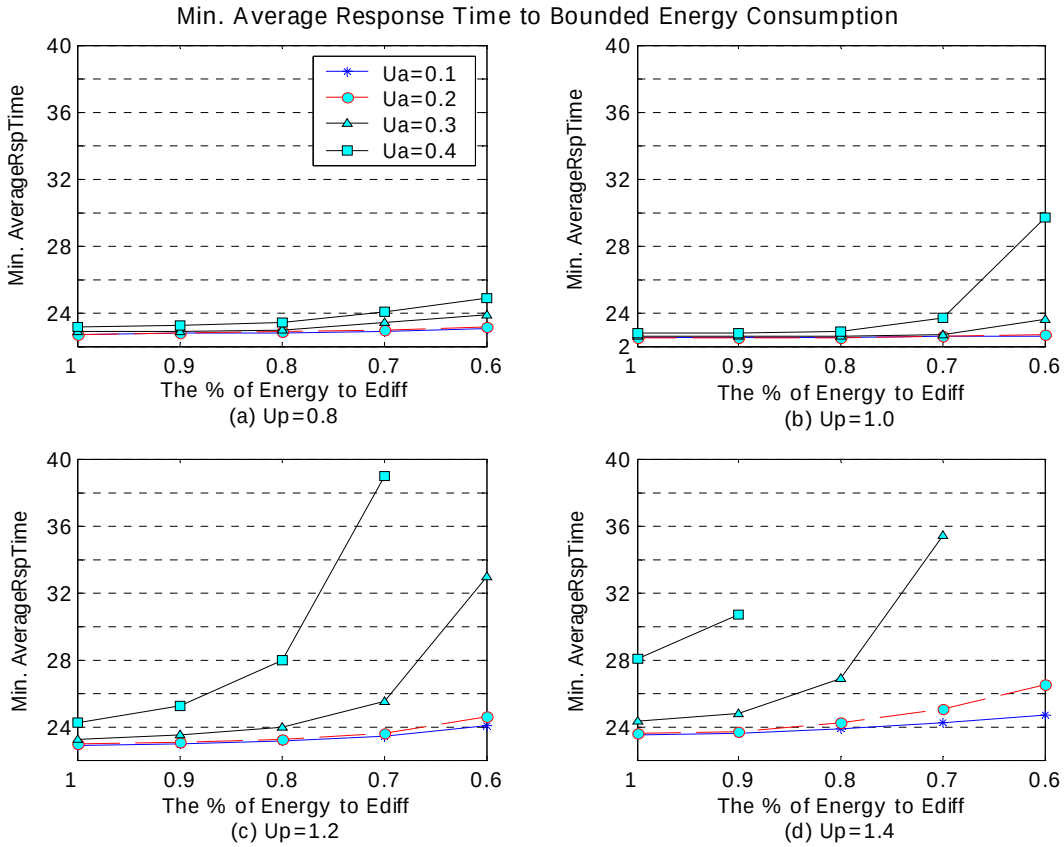


Figure 3-7 Minimum average response time with respect to the bounded energy budget

We now consider how much improvement we can obtain from an increased energy budget. In Figure 3-7, we show the evaluation results for the minimum average response time to the constraint energy ranging from 0.6 to E_{diff}. The responsiveness of aperiodic tasks for U_A=0.1 and 0.2 is not much affected by the periodic tasks' workload U_P and the constraint energy budget

E_C. Since every aperiodic task is assigned to *H-mode* (i.e. γ=1.0 to ensure minimal response time) and is allocated with the maximal energy budget, the available energy budget for periodic tasks decreases as U_A increases. As a consequence, the increased workload in periodic tasks increases the average response time for the case of U_A=0.3 and 0.4 as $U_A^{available}$ is limited and the deadlines assigned to the aperiodic tasks are extended.

3.7 Conclusion

In this paper, we have presented an algorithm to carry out voltage clock scaling in workloads consisting of periodic hard and soft real-time tasks. The aim is to keep within a predefined energy budget. The objective of the scheduling scheme is to minimize the response time of aperiodic tasks while all deadlines of periodic tasks are met and the total energy consumption is bounded by the energy budget. As we apply total bandwidth scheduling for aperiodic tasks, we notice two conflicting factors in energy budget allocation. When an extra budget is assigned to aperiodic tasks, their execution can be done in high-voltage and high-speed mode. This leads to a reduced response time. On the other hand, the extra energy budget allocated to periodic tasks can result in a lowering of the CPU utilization reserved for periodic tasks. This, in turn, leaves more available CPU utilization for aperiodic tasks and cause shorter deadlines as defined in the total bandwidth scheduling scheme.

Our simulation study assumes that there the energy budget is enough to meet the hard real-time periodic tasks and to complete the aperiodic tasks. In addition, here there is extra energy that can be allocated to either periodic or aperiodic tasks. Our results demonstrate that the VCS-EDF scheduling gets the fastest responsiveness when the extra energy budget is allocated to aperiodic tasks at their maximum energy demand such that all of them can run in *H-mode*. Given the requirement of responsiveness and any energy budget, the proposed scheduling method can decide the voltage settings for periodic tasks so that real-time tasks can share the bounded energy budget effectively. Therefore, the work provides the battery-driven embedded real-time system designer with a general view, which allows scheduling real-time tasks considering their general

characteristics of energy demands and processor utilization, given a constraint of bounded energy availability.

CHAPTER 4
DUAL-POLICY DYNAMIC SCHEDULING

4.1 Introduction

Due to the accelerated miniaturization, portability, and complex functionality for handheld devices (i.e. multimedia in portable phone), mobile computing and communication devices such as laptop computers, personal digital assistants (PDA's) are demanding more power. But, the big gap between the pace of current battery technology and energy demands of microprocessor systems cannot satisfy the demands such that power management or power efficiency become the critical design issue of battery-powered embedded system [79].

An effective approach to power reduction is a technique called *Voltage-Clock Scaling* or *Dynamic-Voltage-Scaling* in CMOS circuit technology. Due to the quadratic relationship between supply voltage and clock frequency, a small reduction in voltage produces a significant reduction in power consumption. However, since lowering the supply voltage accompanies a reduction in the clock frequency from the intrinsic relationship, obviously, the objective of minimizing power consumption will result in performance degradation of response time or a failure to meet real-time deadlines without concerning the power-delay tradeoff in low power design.

Despite the fact that there have been a lot of studies in low-power and power aware computing with widely recognized importance of energy consumption in systems, energy efficiency of real-time system under a bounded energy budget, for instance, battery-powered real-time systems, is relatively undefined.

One of the keys to utilize a constrained energy budget more effectively is the ability to allocate the energy consumption to specific components. For an appropriate energy allocation, there have been many studies to measure energy impact at various levels of system such as

instructional, functional, or system components levels [81, 82]. Besides, in real-time embedded systems, the relationship of energy and workload demands must be well understood on top of the energy profiling since energy saving can be achieved by conformity to the conventions of sacrificing system performance in tolerable level.

For aperiodic tasks that have no form of guarantees, they are usually served in the background with respect to hard tasks in order not to jeopardize the schedulability of hard tasks or served by an aperiodic server to improve responsiveness. The objective of minimizing power consumption will result in slow execution for these tasks and the response time could be unrestrained. On the other hand, reduction of energy consumption that doesn't utilize all available energy may not lead to optimal system performance. A better power control strategy in such cases is to minimize the response times of soft real-time tasks, providing that the deadlines of hard real-time tasks are met and the average power consumption is bounded. Fulfilling all these requirements, an energy allocation model is proposed to provide energy profiling and energy sharing scheme at task level [83] for real-time embedded systems that have mixed hard and soft tasks. The model is also reflecting the tradeoff relationship between energy consumption and utilization in VCS-EDF scheduling.

In this chapter, based on the constrained energy allocation model, we propose a *Dual-Policy Dynamic Scheduling* method not only to get better performance for aperiodic tasks but also to extend the lifetime of a system by lowering energy consumption under a confined energy budget. The approach is featuring an intermixing of two elementary schedules, which originated from a pattern of event occurrences in a joint scheduling of periodic and aperiodic tasks. According to traditional scheduling schemes, both kinds of tasks are scheduled under the worst-case scenario, satisfying energy consumption and real-time schedulability. Triggered by the existence of aperiodic tasks' events, explicitly two kinds of intervals are observed in the runtime scheduling for mixed tasks. The one is *P* (Periodic) interval that has only periodic task events and the other *PAP* (Period and APeriodic) interval that has both hard and soft task events. By the

relationship between energy consumption and utilization in VCS-EDF scheduling, the voltage settings for periodic tasks demand a greater energy budget in case of scheduling both kinds of tasks than scheduling only periodic task set from the viewpoint of abiding schedulability. Therefore, in the intervals of *P*, the proposed scheduling algorithm applies an elementary scheduling policy that consumes minimum energy for periodic task set rather than the worst-case scheduling policy determined for scheduling both kinds of tasks. The energy efficiency is obtained from how more tasks are executed in low-speed mode instead of high-speed mode for the intervals of *P*.

Also, executing periodic tasks in different running speed from worst-case in the intervals of *P* and *PAP* increase remaining workload at the switching point such that the backlogged workload to next scheduling policy may cause at least one periodic task to miss its deadline. Since there are slack times brought by the difference of WCET and real execution demand, the backlogging can be alleviated by the slack times. But for the cases the slack times are not enough to nullify the backlogging, two transitory scheduling policies are introduced in-between the intervals of *P* and *PAP*. By the transitory scheduling policies, when the backlogged workload becomes equal to or less than zero, the next targeting elementary schedule can be activated.

According to the switching scheduling policies along with the intervals, we modify total bandwidth server (TBS) algorithm to apply to aperiodic tasks. If there's no backlogged workload to *PAP* intervals, the modified TBS assigns proper deadline algorithm to aperiodic task like the intrinsic total bandwidth server. Otherwise, an infinite deadline is assigned and it is maintained till the backlogged workload becomes no greater than zero.

The rest of this paper is structured as follows. In section 4.2, we discuss design concepts of a dynamic scheduling for energy saving and fast responsiveness under a bounded energy budget. Then, we propose the model of the dual-policy dynamic scheduling in section 4.3. In section 4.4, the proposed dynamic scheduling is explained in detail, including elementary schedules that constitute the dynamic scheduling. The proposed dynamic scheduling algorithm is

explained in section 4.5. To illustrate the effectiveness of the proposed scheduling algorithm, we evaluate the performance through simulations in section 4.6. In section 4.7, a short conclusion is given.

4.2 Design Concepts

In this section, we propose a dynamic real-time scheduling model for mixed periodic and aperiodic tasks. The goals are not only for sharing a bounded energy budget, but also for reducing power consumption under a given energy budget. For the targeted real-time systems, tasks may arrive periodically or sporadically, where the periodic tasks have a constraint whose individual deadlines should be met and the performance of aperiodic tasks is measured in average response time. Under limited energy consumption, we build a dynamic dual-policy scheduling model based on the profiles of energy and utilization demands for both periodic and aperiodic tasks and the energy budget allocation model as proposed in Chapter 3.

4.2.1 Simple Two Voltage Settings

The system operates at two different voltage levels. Ideally, a variable voltage processor that has continuous voltage and clock setting in the operational range is available. We assume a simple setting arrangement that the processor in a real-time system can be dynamically configured in one of two modes: *low-voltage* (*L) -mode* and *high-voltage* (*H)-mode*. In *L-mode,* the processor is supplied with a low voltage (V_L) and runs at a slow clock rate. Thus, task execution may be prolonged but the processor consumes less energy. On the other hand, the processor can be set in *H-mode,* i.e. be supplied with a high voltage (V_H) and run at a fast clock rate, in order to complete tasks sooner at the expense of more energy consumption. The operating speeds at *L-mode* and *H-mode* are denoted as α_L and α_H, respectively, in terms of some unit of computational work. Depending upon the voltage setting for task τ_i, the worst-case execution time is C_i/α_L or C_i/α_H.

4.2.2 Total Bandwidth Server for Aperiodic Tasks

In traditional scheduling in real-time systems that have both periodically and sporadically arriving tasks, aperiodic tasks are usually served in the background with respect to hard tasks in order not to violate the schedulability of hard tasks or served by an aperiodic server to improve responsiveness. Among substantial research works in the scheduling of aperiodic tasks in both fixed and dynamic priority systems, the TBS can produce better responsiveness than other aperiodic mechanisms, sporadic server, and slack stealing, and so on.

Infinite number of soft aperiodic tasks $\{J_i \mid i=0,1,2,\ldots\}$ are modeled as aperiodic computation activities represented by two parameters, λ and μ, where λ is the average inter-arrival time between two consecutive aperiodic instances and μ the average worst-case execution time of all aperiodic tasks .

Aperiodic tasks are scheduled by total bandwidth server algorithm that makes fictitious but feasible deadline assignment based on the available processor utilization guaranteed by the isolation of bandwidth between periodic and aperiodic tasks. In the TBS algorithm, the k-th aperiodic request arriving at time $t = r_k$, a task deadline

$$d_k = max(r_k, d_{k-1}) + \frac{C_k}{U_A} \tag{4-1}$$

is assigned, where C_k is the execution time of the request and U_A the allocated processor utilization for aperiodic tasks. By definition $d_0=0$. The request is then inserted into the ready queue of the system and scheduled by the EDF, as any other periodic instance or aperiodic request already present in the system.

Note that the assignment of the deadlines is such that in each interval of time the processor utilization of the aperiodic tasks is at most U_A. Hence, a set of periodic tasks with utilization factor $U_P = \sum_{i=1}^{n} C_i/T_i$ and a TBS with a bandwidth U_A is schedulable by EDF if and only if $U_A+U_P \leq 1$. The definition and the formal analysis of this algorithm are proved [40]

4.2.3 Energy Budget Allocations using Two-mode VCS-EDF Scheme

While the constraint $E_P + E_A \leq E_C$ must be satisfied, there is a profound intervention on how processor utilization, task scheduling, and task response time are affected. From the viewpoint of utilization, the more utilization is available for aperiodic tasks, the shorter deadlines are assigned to them by the deadline assignment of equation (4-1). This brings higher priorities to them in EDF scheduling such that they can get faster response times. To give more utilization to aperiodic tasks, the utilization of periodic tasks must be shrunken and it can be done by assigning more tasks to *H-mode*, but requires more energy budget. Since total energy budget given to a system is bounded, the energy budget left to aperiodic tasks will be reduced. As a result, the aperiodic tasks must be run in a low voltage mode and their response times will be extended.

Likewise, from the viewpoint of energy budget, the portion assigned in *H-mode* for aperiodic tasks should be maximized within an assigned energy budget E_A to get faster responsiveness. But, under bounded energy budget in a system, if the energy demand of aperiodic tasks is increased, the energy left to periodic tasks will be decreased. Consequently, available utilization for aperiodic tasks will be decreased due to the increased execution time of periodic tasks, which may result in degradation in responsiveness.

Eventually, to get both the schedulability and fast responsiveness under a bounded energy budget, an effective scheduling and energy allocation scheme is needed for jointly scheduling of hard periodic and soft aperiodic tasks. The scheduling should address the concern of the trade-off between utilization and energy consumption.

4.2.4 Elementary Schedules

In a sequence of events of a real-time scheduling, idle and busy cycles appear in turn for irregular duration of time so that the scheduling sequence consists of a continuum of idle and busy cycles. For an idle cycle, the scheduler does not have any jobs to do such that neither task's arrival nor completion of arrived tasks is happened. For a busy cycle the scheduler is working continuously with no idle moments. Begun with the arrival of either a periodic or aperiodic task

and followed by the arrivals of other tasks, a busy cycle is mainly composed of arrivals, executions, and completions of tasks. The execution is computations based on a scheduling policy and execution demand. A completion represents that the computation has been completed and the executed task leaves the scheduler. When there are no more tasks waiting to being executed, another idle cycle starts, ending the busy cycle.

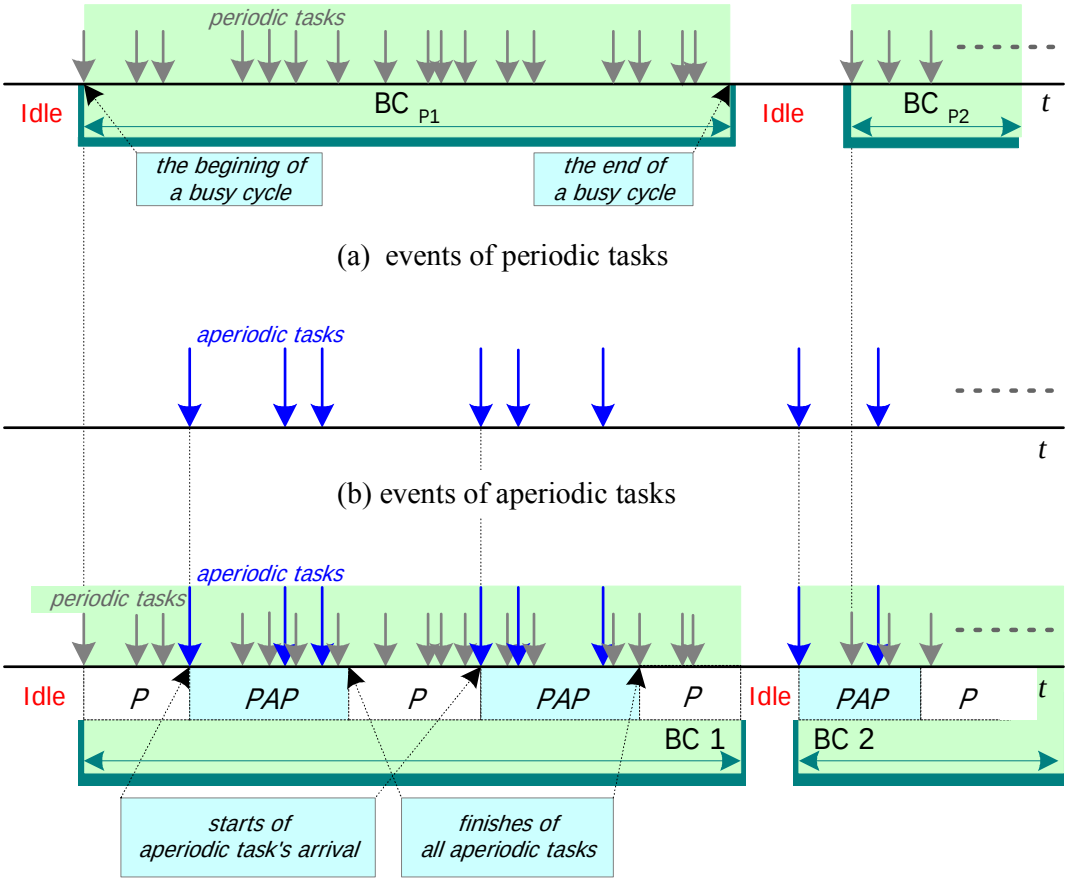


Figure 4-1 An example of scheduling mixed real-time tasks and two explicit intervals in the event pattern

An example of event sequences is illustrated in Figure 4-1. An arrival sequence of mixed tasks, periodic and aperiodic tasks in Figure 4-1 (c), is separately arranged into the arrivals of

only periodic tasks and the other of only aperiodic tasks as shown in Figure 4-1 (a) and (b), respectively. In Figure 4-1 (a), the periodic tasks are scheduled, busy and idle cycles appears in turns, i.e. BC_{P1}, idle, $BC_{P2,}$ idle…, depending on the periods, deadlines, and execution times/WCETs of the task set. To minimize energy consumption, the voltage-clock scaling problem is imposed when task set is schedulable if a processor is entirely run in *H-mode*, and misses at least one deadline if running in *L-mode* completely. Keeping the processor entirely in *H-mode* to meet all tasks' deadlines results in extra energy consumption. Therefore, the algorithm for hard real-time tasks, VCS-EDF scheduling, is called for to determine the optimal voltage settings and to minimize *H-mode* execution, while guaranteeing that no deadline is missed, such that it will play its role for energy efficiency as shown in [58].

If the systems consist of only soft aperiodic tasks as shown in Figure 4-1 (b), the objective of minimizing power consumption will result in slow execution for these tasks and the response time could be unrestrained. This is not a desirable strategy for scheduling aperiodic tasks. When aperiodic tasks are scheduled with the same set of periodic tasks by TBS algorithm, the processor utilization occupied by periodic tasks must be shrunk to make sufficient room for the aperiodic tasks without violating the schedulability. It means more periodic tasks must be assigned in *H-mode* to accommodate aperiodic tasks at the cost of higher energy consumption. Thus, clearly, the energy consumption for scheduling aperiodic tasks with periodic tasks is higher than the case of scheduling only for the set of periodic tasks.

In a joint scheduling of periodic tasks with aperiodic tasks, the arrival and completion of periodic and aperiodic tasks are intermixed with each other and show an explicit pattern as shown in Figure 4-1 (c). Following after *idle cycles*, any arrival of a periodic or aperodic task invokes busy cycles, *BC1* and *BC2*, and finishes of all arrived tasks end the busy cycles. A hard periodic task opens the first busy cycle (*BC1*) and is followed by several arrivals of periodic and aperiodic tasks. The second (*BC2*) begins with an aperiodic task. Suppose *P* (*Periodic*) is the interval, which has no event for aperiodic tasks, but only periodic tasks and *PAP* (*PAP: Periodic and*

APeriodic), which has mixed arrivals of both tasks. Then *BC2* starts with *PAP*, as does *BC1* with *P*. Likewise, an arrival of aperiodic task closes *P* and opens *PAP*. And the completion of all aperiodic tasks awaiting execution ends *PAP* interval and the busy cycle encounters an interval *P* again.

Our dual policy scheduling for mixed tasks is motivated by the fact that the energy consumption required by periodic tasks is less than the one required by both periodic and aperiodic tasks under the same amount of energy allocation. If voltage settings that consume minimum energy for periodic tasks are selected to schedule periodic tasks for *P* intervals, less energy consumption is expected compared to when the worst-case voltage settings for mixed tasks throughout scheduling regardless of the pattern in event occurrences. Consequently, whenever *P* interval is coming up, switching running modes from the set for mixed tasks to the set for only periodic tasks can reduce energy consumption.

In an extreme case, a busy cycle may consist of only periodic tasks without any aperiodic task, i.e. *P* intervals. If periodic tasks for *P* intervals are scheduled by the voltage settings that use up minimum energy budget, the energy consumption will be remarkably reduced comparing the running modes for mixed tasks, which are used for periodic tasks. Thus, to exploit the benefit in energy consumption in switching from worst-case to minimum energy consuming schedule for the intervals of *P*'s, we build elementary schedules named *S1* and *S2* for the two intervals, respectively, and develop a dynamic scheduling model based on the elementary schedules, in which every first and last existence of aperiodic events in a sequence of scheduling events is the turning point between the elementary schedules.

4.3 Dynamic Dual-Policy Scheduling Model

In this section, we consider VCS-EDF scheduling to improve energy efficiency in real-time embedded systems having mixed periodic and aperiodic tasks. An energy efficient on-line scheduling under a bounded energy budget can extend the service time of energy-confined system much more. Since power consumption and execution time are contradictory to each other in

voltage-clock scaling scheme, there exists a profound intervention in utilization, schedulability, and response time and they should be well considered in real-time systems. In addition, the bounded energy budget must be properly shared among tasks to satisfy the time constraints for periodic tasks and fast response time for aperiodic tasks in performance criterion that real-time system pursues. Understanding the relationship between energy consumption and utilization of tasks based on VCS-EDF scheme, we use a task-based profiling of energy and utilization demands for both periodic and aperiodic tasks and adopt the energy budget allocation model [83].

Based on the event pattern occurring in a scheduling of mixed periodic and aperiodic tasks and the relationship between energy and processor utilization demands, we develop an on-line scheduling named *dual-policy dynamic scheduling*, which can reduce energy consumption by switching the scheduling policies along with the events of aperiodic tasks. Triggered by new arrivals after *P* intervals and last execution of aperiodic task in *PAP* intervals, the event pattern of a busy cycle is repeated by two regions of interval in turns, one consists of only periodic and the other of mixed tasks. Corresponding to the pattern by the two intervals in scheduling, we build elementary schedules named *S1* and *S2*, which can be applied for intervals of *P* and *PAP*, respectively.

We assume switching between elementary schedules consumes a negligible overhead like context switching or voltage switching between tasks. This is also analogous to the assumption made in classical real-time scheduling theory, that preemption costs are negligible.

4.3.1 Schedule for Only Periodic Tasks

Energy is consumed only for periodic tasks. The voltage settings are determined to minimize energy consumption by periodic tasks, while meeting their schedulability within the allocated energy budget. Given a set of periodic tasks, a set of voltage settings, *S1*: $\{m_i\}$, satisfying the conditions uniquely exists for them. Since the voltage settings demand the minimum energy consumption under the allocated energy budget, let them be defined as a *best-case schedule* for periodic tasks regarding energy saving.

For the sake of schedulability in EDF scheduling, the tasks should be scheduled in such a way that the utilization is less than unity. Therefore, we define *min* E_P as an energy demand and *max* U_P as a worst-case utilization when there exist a set of voltage settings *S1*: $\{m_i\}$ so that the worst-case utilization

$$U_P(m_i) = \sum \frac{1}{\alpha(m_i)} \frac{C_i}{T_i} \le 1 \quad \text{and} \quad E_P(m_i) = \sum \frac{1}{\alpha(m_i)} \frac{\overline{e_i}}{T_i} p(m_i) \text{ is minimized.} \quad (4\text{-}2)$$

Regarding the feasibility of energy constraint, E_C must be greater than *min* E_P.

4.3.2 Schedule for Both Periodic and Aperiodic Tasks

Periodic tasks share a constrained energy budget with aperiodic tasks, taking account of fast responsiveness for aperiodic tasks. The voltage settings are decided to achieve minimal response times for all aperiodic tasks, guaranteeing to meet deadlines of periodic tasks. Each aperiodic task J_k gets a finite deadline d_k according to deadline assignment algorithm in total bandwidth server. According to the *constrained energy allocation model* proposed in Chapter 3, voltage settings, *S2*: $\{m_i\}$, are determined. As the voltage settings demand the worst-case energy consumption under the allocated energy budget, let them be defined as *worst-case schedule* for periodic tasks.

Voltage settings must be determined such that the energy consumption satisfies the constraint of $E_P + E_A \le E_C$, while guaranteeing the schedulability of periodic tasks and minimizing average response time for aperiodic tasks. Using the algorithm developed in Chapter 3, the energy allocation factors, β and γ, voltage settings for periodic tasks, *S2*: $\{m_i\}$, and the percentage of *H-mode* assignment for aperiodic tasks, x_H are determined under the bounded energy consumption E_C.

The optimization problem to find voltage settings for periodic tasks can be stated as follows: Pick the task subsets *H* and *L* for voltage settings of *H-mode* and *L-mode* such that

- $H \cup L = \{\tau_1, \tau_2, ..., \tau_n\}$
- $H \cap L = \varnothing$

- $U_P(m_i) = \sum \frac{1}{\alpha(m_i)} \frac{C_i}{T_i}$ is minimized

subject to the well-known sufficient condition[1] for the schedulability of periodic tasks under EDF, i.e.,

$$\frac{1}{\alpha_H} \sum_{i \in H} \frac{C_i}{\min(T_i, D_i)} + \frac{1}{\alpha_L} \sum_{i \in L} \frac{C_i}{\min(T_i, D_i)} \le 1 \tag{4-3}$$

and the energy consumption constraint of

$$\frac{p_H}{\alpha_H} \sum_{i \in H} \frac{\overline{C_i}}{\min(T_i, D_i)} + \frac{p_L}{\alpha_L} \sum_{i \in L} \frac{\overline{C_i}}{\min(T_i, D_i)} \le E_P .$$

This optimization problem can be treated equivalently to the decision problem of the subset sum, which is *NP*-complete. Consequently, efficient search heuristics, e.g., branch-and-bound algorithms, should be employed to find a solution if n is large.

As for scheduling aperiodic tasks, the total-bandwidth algorithm is used based on $\gamma=1$, which is decided to give best responsiveness within the range of the bounded energy budget. According to the schedulable condition of $U_A+U_P \le 1$ in the total bandwidth server, the utilization of aperiodic tasks U_A is determined by $U_A = 1-U_P$, where U_P is determined by the voltage settings in the equation (4-3).

4.3.3 Schedules for Transition between Elementary Schedules

Since the constitutions of voltage settings are different from each other for two sets of elementary schedules, *S1*: $\{m_i\}$ and *S2*: $\{m_i\}$, a transition between different schedules, such as from *S1* to *S2* or from *S2* to *S1*, cannot be simply accomplished by applying different voltage settings to the tasks. The remaining workload at the switching decision points may cause the targeting schedule to violate some times later. To prevent the violation, in this section, two

[1] The condition is also necessary if $D_i \ge T_i$ for all i.

transitory schedules, *S12* for the switching from *S1* to *S2* schedule and *S21* from *S2* to *S1* schedule, are proposed.

4.3.3.1 Energy Saving

For *P* intervals, selecting the running mode for periodic tasks from *S1* instead of *S2* reduces energy consumption by executing tasks in *L-mode*, which is assigned in *H-mode* in *S2* schedule. Compared to the energy consumption by the worst-case schedule *S2*, energy saving is achieved from the difference in energy consumption existing between the two different energy levels for a task in two schedules. The amount of saved energy is proportional to the difference in amount of executed time by switched running mode. However, the remaining workload increased by running periodic tasks in low speed by *S1*: $\{m_i\}$ instead of in high speed assigned in *S2*: $\{m_i\}$ is expected to affect not only the schedulability of scheduling in *S2*, but also the performance in average response time with an arrival of aperiodic task.

4.3.3.2 Backlogged Workloads at Switching Points

High-speed execution finishes given amount of execution demand, which consists of a serial of instructions, earlier than in low-speed execution due to the higher operating frequency. Thus, as of workload accomplished after a certain passage of time, low-speed execution of a task leaves more workload to execute than high-speed execution. The difference in remaining workload is also proportional to the difference in operating frequencies of two execution modes.

For a set of periodic tasks in the proposed dual-policy scheduling, voltage settings are determined as two sets of mixed *H-mode* and *L-mode* for both elementary schedules, *S1*: $\{m_i\}$ and *S2*: $\{m_i\}$. It is obvious that workload assigned to *H-mode* in *S1* schedule is less than in *S2* schedule since periodic tasks in *S2* schedule are more likely to be assigned to *H-mode* due to the share of processing capacity with aperiodic tasks under total bandwidth server algorithm, $U_P\ (S2) \leq 1 - U_A\ (S2)$. In other words, when any periodic task that is assigned to *H-mode* for *S2* is executed in *L-mode* for an interval *P* (by *S1* schedule), the low-speed execution of the task may not finish

the execution demand and may cause the unfinished portion to be accumulated to be a backlogged workload to the next policy. During intervals of *P*, the more periodic tasks are executed in *L-mode* of *S1* schedule instead of *H-mode* of *S2* schedule, the more workload is remained without being executed. On the other hand, any periodic task assigned to *L-mode* for *S2* is executed in *H-mode* for interval *P* (by *S1* schedule), the faster execution of the task will finish the execution demand earlier at the cost of more energy consumption. By the definition of schedules *S1* and *S2*, the difference of workload by the *P* intervals is varied along with the sum of execution times of tasks in different voltage settings among tasks encountered in the intervals.

We define *WD (t)* as a difference in workloads at time *t* between current scheduling policy and targeting worst-case scheduling policy, to which the scheduler is going to switch.

4.3.3.3 Schedulability Analysis at Switching Points of Scheduling Policies

At the moment of an arrival of aperiodic task after the interval *P* when the dynamic scheduler changes its policy with *S2* schedule, there can be more workload than *S2* schedule can afford to schedule mixed periodic and aperiodic tasks, abiding by its schedulability. So, if workload at switching point is more than targeting worst-case schedule to which the scheduler switches, at least one of the periodic tasks will miss its deadline. The unfinished workload by running periodic tasks in *S1* instead of *S2* schedule for *P* intervals should be completed before changing to *S2* schedule not to lead an unschedulable state. Like switching from *S1* to *S2* schedules at the end of *P* interval, for switching of scheduling policies from *S2* to *S1* at the end of *PAP* intervals, there may also exist a difference in remained workload due to the different voltage settings for two schedules.

Therefore, for the sake of real-time schedulability, workload handed over from previous schedule at the points of switching scheduling policies must not be higher than the workload affordable by targeting worst-case schedule. If so, there must be an effective scheme to prevent from being unschedulable. In case that the remained workload for the current schedule is less than

the one affordable for the worst case of the schedule to take, the switching of schedules does not need any further action to make them schedulable.

Before proposing transitory schedules between two elementary schedules, we explain how the non-zero workload difference between two scheduling policies endangers the schedulability of VCS-EDF scheduling. Based on the concept of processor demand for hard real-time tasks [80], we extend the schedulability of a periodic task set whose utilization U_P is defined in the range of $\alpha_L \le U_P \le \alpha_H$ in *L-mode*.

- ***Feasibility Analysis of EDF scheduling***

Spuri *et al*. define two concepts to analyze the feasibility of real-time task sets; the processor demand as a focused measure of how much computation is requested, with respect to timing constraints, in a given interval of time and the loading factor as the maximum fraction of processor time possibly demanded by the task set in any interval of time. Given a set of real-time jobs and an interval of time $[t_1, t_2)$, the processor demand and loading factor of the job set on the interval $[t_1, t_2)$ are respectively defined as $h_{[t_1,t_2)} = \sum_{t_1 \le r_i, t_2 \le d_i} C_i$ and $u_{[t_1,t_2)} = \frac{h_{[t_1,t_2)}}{t_2 - t_1}$ [80].

Theorem 4.1 (Spuri) Each set of real-time tasks is feasibly scheduled by EDF if and only if $u \le 1$ [80].

By showing that the loading factor u of the task set is equal to U, Spuri proves the feasibility analysis of a task set under EDF scheduling addressed by Liu and Layland [33] as follows.

Corollary 4.1 (Liu and Layland) Any set of n synchronous periodic tasks with processor utilization $U = \sum_{i=1}^{n} \frac{C_i}{T_i}$ is feasibly scheduled by EDF if and only if $U \le 1$.

Proof. By Theorem 4.1, the thesis then follows.

For any interval $[t_1, t_2)$: $\sum_{t_1 \le r_k, t_2 \le d_i} C_i \le \sum_{i=1}^{n} \left\lfloor \frac{t_2 - t_1}{T_i} \right\rfloor C_i \le \sum_{i=1}^{n} \frac{t_2 - t_1}{T_i} C_i = (t_2 - t_1) \sum_{i=1}^{n} \frac{C_i}{T_i}$,

that is, $u_{[t_1, t_2)} \le U$.

Now, let $t_1=0$ and t_2= lcm ($T_1, \ldots, T_n$):

$u_{[t_1, t_2)} = \frac{\sum_{i=1}^{n} \frac{t_2}{T_i} C_i}{t_2} = U$. And it follows $u=U$. ■

- ***Feasibility Analysis of Two-mode VCS-EDF Scheduling***

In the same way as explained above, in the VCS-EDF scheduling given by equations (4-2) and (4-3), the processor demand and the loading factor for the task set assigned in either *H-mode* or *L-mode* on the interval $[t_1, t_2)$ is respectively defined as

$$h_{[t_1, t_2)} = \frac{1}{\alpha_H} \sum_{\substack{t_1 \le r_k, t_2 \le D_k \\ k \in H}} C_k + \frac{1}{\alpha_L} \sum_{\substack{t_1 \le r_k, t_2 \le D_k \\ k \in L}} C_k \tag{4-4}$$

and

$$u_{[t_1, t_2)} = \frac{h_{[t_1, t_2)}}{t_2 - t_1} .$$

Corresponding to the Corollary 4.1, the feasibility of VCS-EDF scheduling is defined by the following Corollary.

Corollary 4.2 Any set of n synchronous periodic tasks with processor utilization $U = \frac{1}{\alpha_H} \sum_{i \in H} \frac{C_i}{T_i} + \frac{1}{\alpha_L} \sum_{i \in L} \frac{C_i}{T_i}$ is feasibly scheduled by EDF if and only if $U \le 1$.

Proof. By Theorem 4.1, the thesis then follows.

For any interval $[t_1, t_2)$:

$$h_{[t_1, t_2)} = \frac{1}{\alpha_H} \sum_{\substack{t_1 \le r_k, t_2 \le D_i \\ i \in H}} C_i + \frac{1}{\alpha_L} \sum_{\substack{t_1 \le r_k, t_2 \le D_i \\ i \in L}} C_i$$

$$\le \frac{1}{\alpha_H}\sum_{i\in H}^{n}\left\lfloor \frac{t_2-t_1}{T_i}\right\rfloor C_i + \frac{1}{\alpha_L}\sum_{i\in H}^{n}\left\lfloor \frac{t_2-t_1}{T_i}\right\rfloor C_i$$

$$\le \frac{1}{\alpha_H}\sum_{i\in H}^{n}\frac{t_2-t_1}{T_i}C_i + \frac{1}{\alpha_L}\sum_{i\in H}^{n}\frac{t_2-t_1}{T_i}C_i$$

$$= (t_2-t_1)\left(\frac{1}{\alpha_H}\sum_{i\in H}^{n}\frac{C_i}{T_i} + \frac{1}{\alpha_L}\sum_{i\in L}^{n}\frac{C_i}{T_i}\right)$$

Now, let t_1=0 and t_2= lcm (T_1, …, T_n):

$\frac{h_{[0,t_2)}}{t_2} = \frac{1}{\alpha_H}\sum_{i\in H}^{n}\frac{C_i}{T_i} + \frac{1}{\alpha_L}\sum_{i\in L}^{n}\frac{C_i}{T_i} = U$. And it follows *u*=*U*. ■

- ***Feasibility Analysis of Dual-Policy Dynamic Scheduling***

By the characteristics of the processor demand, it is also corresponded to the workload in any interval of time. Then, let $WD_T(t)$ be the workload difference, backlogged workload to policy *T*, at time *t* defined as $W^C(t) - W^T(t) = h^C_{[0,t)} - h^T_{[0,t)}$, where *C* represents for current scheduling policy and so does *T* for targeted one.

Theorem 4.2 Any set of *n* synchronous periodic tasks with processor utilization $U_P^{S1} = \frac{1}{\alpha_H}\sum_{i\in H_{S1}}\frac{C_i}{T_i} + \frac{1}{\alpha_L}\sum_{i\in L_{S1}}\frac{C_i}{T_i}$ and $U_P^{S2} = \frac{1}{\alpha_H}\sum_{i\in H_{S2}}\frac{C_i}{T_i} + \frac{1}{\alpha_L}\sum_{i\in L_{S2}}\frac{C_i}{T_i}$, where $U_P^{S2} < U_P^{S1}$, $U_A = 1 - U_P^{S2}$, and periodic tasks are scheduled by *S1* by the time *t* with real execution demand e_j for task τ_j, is not feasibly scheduled by *S2* after time *t* if $WD_2(t) > 0$.

Proof. Suppose that for the interval of $P = [t_1, t_2)$ having only periodic events, there is at least an arrival of a periodic task, τ_j, assigned to *H-mode* in *S2* schedule and *L-mode* in *S1* schedule. Then the processor demand for *S1* schedule at t_2 is

$$h^{S1}_{[t_1,t_2)} = \frac{1}{\alpha_H}\sum_{\substack{t_1\le r_i, t_2\le D_i \\ k\in H_1}} C_i + \frac{1}{\alpha_L}\sum_{\substack{t_1\le r_i, t_2\le D_i \\ k\in L_1}} C_i + \frac{1}{\alpha_L}e_j$$

$$= (t_2 - t_1)\left(\frac{1}{\alpha_H} \sum_{i \in H_1}^{n} \frac{C_i}{T_i} + \frac{1}{\alpha_L} \frac{e_j}{T_j} + \frac{1}{\alpha_L} \sum_{i \in L_1, i \neq j}^{n} \frac{C_i}{T_i} \right)$$

$$= (t_2 - t_1)\left(\frac{1}{\alpha_H} \left(\sum_{i \in H_1, i \neq j}^{n} \frac{C_i}{T_i} + \frac{C_j}{T_j} \right) - \frac{1}{\alpha_H} \frac{C_j}{T_j} + \frac{1}{\alpha_L} \frac{e_j}{T_j} + \frac{1}{\alpha_L} \sum_{i \in L_2}^{n} \frac{C_i}{T_i} \right)$$

$$= (t_2 - t_1)\left(\frac{1}{\alpha_H} \sum_{i \in H_2}^{n} \frac{C_i}{T_i} + \frac{\alpha_H e_j - \alpha_L C_j}{\alpha_H \alpha_L} \frac{1}{T_j} + \frac{1}{\alpha_L} \sum_{i \in L_2}^{n} \frac{C_i}{T_i} \right)$$

$$= (t_2 - t_1)\left(\frac{1}{\alpha_H} \sum_{i \in H_2}^{n} \frac{C_i}{T_i} + \frac{1}{\alpha_L} \sum_{i \in L_2}^{n} \frac{C_i}{T_i} \right) + (t_2 - t_1) \frac{\alpha_H e_j - \alpha_L C_j}{\alpha_H \alpha_L T_j}$$

$$WD_2(t_2) = h^{S1}_{[t_1, t_2)} - h^{S2}_{[t_1, t_2)} = (t_2 - t_1) \frac{\alpha_H e_j - \alpha_L C_j}{\alpha_H \alpha_L T_j}$$

Now, let $t_1 = 0$ and $t_2 = \text{lcm}(T_1, \ldots, T_n)$:

If $WD_2(t_2) > 0$, $\alpha_H e_j - \alpha_L C_j > 0$

$$\frac{h^{S1}_{[0, t_2)}}{t_2} = \frac{h^{S2}_{[0, t_2)}}{t_2} + \frac{\alpha_H e_j - \alpha_L C_j}{\alpha_H \alpha_L T_j}$$

$$U_P^{S1} = U_P^{S2} + \frac{\alpha_H e_j - \alpha_L C_j}{\alpha_H \alpha_L T_j} > U_P^{S2}. \tag{4-5}$$

Otherwise, $\alpha_H e_j - \alpha_L C_j \leq 0$

$$U_P^{S1} = U_P^{S2} + \frac{\alpha_H e_j - \alpha_L C_j}{\alpha_H \alpha_L T_j} \leq U_P^{S2}.$$

At time $t = t_2$, the workload by *S1* schedule handed over *S2* schedule make the utilization by *S2* schedule higher than U_P^{S2} and the scheduling by *S2* not feasible from the time. If task τ_j, which is assigned to *H-mode* in *S2*, is executed in *L-mode* for intervals of *P* by *S1* schedule before any arrival of aperiodic task, there is a non-zero

difference as much as $\left(\frac{\alpha_H e_j - \alpha_L C_j}{\alpha_H \alpha_L T_j} \right)$ between utilizations of two schedules. From the schedulability point of view when *S2* schedule is taken with an arrival of aperiodic task, it makes *S2* schedule *not* schedulable such that at least one of the tasks can miss its deadline. ■

Then, for the targeting schedule's schedulability, how the workload handed over at time *t* can be effectively reduced before switching to the targeting *S1* or *S2* schedule. There is a major factor to reduce the *WD (t)* to zero or less than zero. Considering that a periodic task consumes less execution time than the WCET, the *WD (t)* in-between the interval *P* and *PAP* will not be so high. Thus, if the slack times coming from the difference between WCET and actual execution demand nullify the *WD (t)* or make it less than zero, the switching can be performed easily without going through further processes to maintain schedulability. Otherwise, an efficient scheme is required to deal with the reduction of the workload difference. The scheme should not only guarantee schedulability in transitory intervals between schedules, *S1* and *S2*, but also lead it to the schedulable dynamic scheduling by the target policy. Thus, we introduce *transitory schedules* for the in-between elementary schedules.

Making use of slack times to lower *WD (t)*, if tasks assigned at low-speed mode in targeting schedule are executed in high-speed mode, the accelerated executions also reduce the handed-over workload. And running tasks for the transitory intervals in high-speed mode is schedulable at the cost of more energy consumption, as the workload of the tasks set is given in the range of $\alpha_L \le U_P \le \alpha_H$ and the utilization for the case become *min* U_P, which is $U_P = \frac{1}{\alpha_H} \sum_i \frac{C_i}{T_i}$ satisfying schedulability, since $U_P^{S1 or S2} = \frac{1}{\alpha_H} \sum_{i \in H_1} \frac{C_i}{T_i} + \frac{1}{\alpha_L} \sum_{i \in L} \frac{C_i}{T_i} \le 1$, where $min U_P \le U_P^{S1 or S2}$.

4.3.3.4 Modified TBS in Dual-Policy Dynamic Scheduling

For the switching policies from *S1* to *S2* schedule with any arrival of aperiodic task, we modify the TBS algorithm to guarantee the schedulability. If *WD2 (t) > 0* from the equations (4-4) and (4-5), a TBS with a bandwidth U_A becomes unschedulable by EDF. Thus, when *WD2 (t) > 0*, *Modified TBS (MTBS)* overcome the unschedulable state by assigning infinite deadlines to aperiodic tasks at the cost of increased response time, making utilization of U_A zero. By the state of *WD2 (t)* in Dual-Policy Dynamic Scheduling, the *k*-th aperiodic request arriving at time $t = r_k$, MTBS flexibly assigns a task deadline

$$d_k = max(r_k, d_{k-1}) + \frac{C_k}{U_A}, \quad \text{when } WD2\ (t) \le 0 \tag{4-6}$$

$$d_k = \infty, \quad \text{when } WD2\ (t) > 0. \tag{4-7}$$

The assignment of infinite deadlines to aperiodic tasks represents that the scheduler regards the arrived tasks as not ready to be executed. At the moment when the schedulability for the target elementary schedule is guaranteed, aperiodic tasks that arrived but waiting in the scheduler are activated. Due to the waiting time from arrival to the activation, the responsiveness of aperiodic tasks is directly affected by the postponed executions of arrived aperiodic tasks.

4.3.3.5 Transitory Schedule S12

With any arrival of aperiodic task after a *P* interval, scheduling policy need to be switched to *S2* schedule. Given workload difference *WD2 (t)* between current scheduling and S2 scheduling, since *WD2 (t)* is bigger than zero, the targeting scheduling policy cannot replace S1 scheduling. Instead, transitory schedule *S12* is selected such that all periodic tasks are executed in *H-mode* and the deadlines of aperiodic tasks are assigned as infinite until *WD2 (t)* becomes less than or equal to zero. But the work amount increased by running some periodic tasks in *L-mode* instead of in *H-mode* affect the performance in average response time with an arrival of aperiodic

task. At the point where the work difference becomes zero, tasks including aperiodic tasks are finally scheduled by targeting schedule of *S2*.

4.3.3.6 Transitory Schedule S21

After finishing an aperiodic task, schedule *S1* can be chosen for the events of periodic tasks if the workload is affordable for the schedule. Because staying in schedule *S2* after finishing an aperiodic task consumes more energy than scheduling tasks using *S1*, it is more efficient in the energy consumption point of view to switch scheduling policy from *S2* to *S1*. Like transitory schedule *S12*, all tasks are executed in *H-mode* until *WD1* (t) becomes less than or equal to zero.

4.4 Dual-Policy Dynamic Scheduling Using VCS-EDF

In Dual-Policy Dynamic Scheduling model, the switching instants are basically at the boundaries, where intervals *P* and *PAP* are crossed. They are the moments of the first arrival of aperiodic task after an idle cycle or after the intervals *P* and the last completion of aperiodic task in the scheduler. Since the voltage settings are different in two scheduling policies, for the assurance of schedulability, the workload differences at switching points needs to be checked, making the workload difference *WD (t)* the main criterion to judge the schedulability at switching point *t*. *WD1 (t)* is the workload difference between real executions and worst-case executions when *S1* schedule is taken as targeting schedule from the instant *t*. Likewise, *WD2 (t)* is the one when *S2* schedule is taken at *t*.

4.4.1 Terms and Conditions

Suppose t_B, t_A, t_T, t_P, and t_F be the instants when a scheduling policy switching is determined based the events of aperiodic tasks and the state of workload difference. A busy cycle is closed by any instant except for t_B. Also T_{idle}, T_{BA}, T_{AT}, T_{TP}, and T_{PA} are the intervals divided by the instants in a busy cycle.

- $\boldsymbol{t_B}$: the instant of a busy cycle begun with any arrival of either periodic or aperiodic task, idle cycle is closed.

- t_A: the instant when the aperiodic task J_k has arrived so that the event pattern includes both periodic and aperiodic tasks

- t_T: the instant when *WD2* (t_T) becomes less than or equal to zero so the scheduling policy can be switched to *S2* schedule.

- t_F: the instant when all aperiodic tasks in scheduling queue are finished so no more aperiodic tasks are waiting in scheduler.

- t_P: the instant when *WD1* (t_P) becomes less than or equal to zero so that the scheduling policy can be switched to *S1* schedule.

- T_{idle}: the interval there's no task's arrival from either t_A, t_T, t_P, or t_F to t_B.

- T_{BA}: the interval from time t_B to t_A.

- T_{AT}: the interval from time t_A to t_T.

- T_{TF}: the interval from time t_T to t_F.

- T_{FP}: the interval from time t_F to t_P.

- T_{PA}: the interval from time t_P to t_A.

If the bounded energy consumption budget is given as E_C, E_C must fall into the range $E_{min} \le E_C \le E_{max}$ where *min* E_P, *min* E_A, *max* E_P, and *max* E_A are defined as follows.

- E_{max} : Energy consumption when all periodic tasks are run in *H-mode* so the processor runs at a fast clock rate all the time. E_{max} = *(max* E_P + *max* E_A*)*, where $max\,E_P = \sum \frac{1}{\alpha_H} \frac{\overline{C_i}}{T_i} p_H$ and $max\,E_A = \frac{1}{\alpha_H} \frac{\mu}{\lambda} p_H$.

- E_{min} : E_{min} = *(min* E_P + *min* E_A*)*, where *min* E_P is determined by (4-3) when the sum of utilizations of periodic and aperiodic tasks takes any value in the range of α_L to α_H. As for *min* E_A, it is the minimum energy consumption when all aperiodic tasks are run in *L-mode, i.e.* $min\,E_A = \frac{1}{\alpha_L} \frac{\mu}{\lambda} p_L$.

- E_{diff} : The difference in maximum and minimum energy consumptions by mixed tasks such that E_{diff} = *(*$E_{max} - E_{min}$*)*.

4.4.2 Switching Scheduling Policies

In Figure 4-2, we illustrate an example of switching scheduling policies in a busy cycle starting with an arrival of a periodic task followed by aperiodic and periodic tasks. It also shows when all of the schedules in dual-policy dynamic scheduling are activated in the busy cycle

according to switching criteria. Basically, switching decisions between scheduling policies are committed at the moment of the first arrival of aperiodic task in the interval *PAP* within the busy cycle (t_A) and finishes all of the arrived aperiodic tasks (t_F) with the status of workload differences by real executions and worst-case demands, *WD1 (t)* and *WD2 (t)*. As the work differences do not satisfy the switching condition, transitory schedules *S21* and *S12* are chosen. Finally, switches to the targeting elementary schedules are committed at both t_T and t_P. All of the intervals except for T_{idle} and T_{BA} are circulated in turns along upcoming events by mixed tasks in a busy cycle. And another round of intervals is started after T_{idle}.

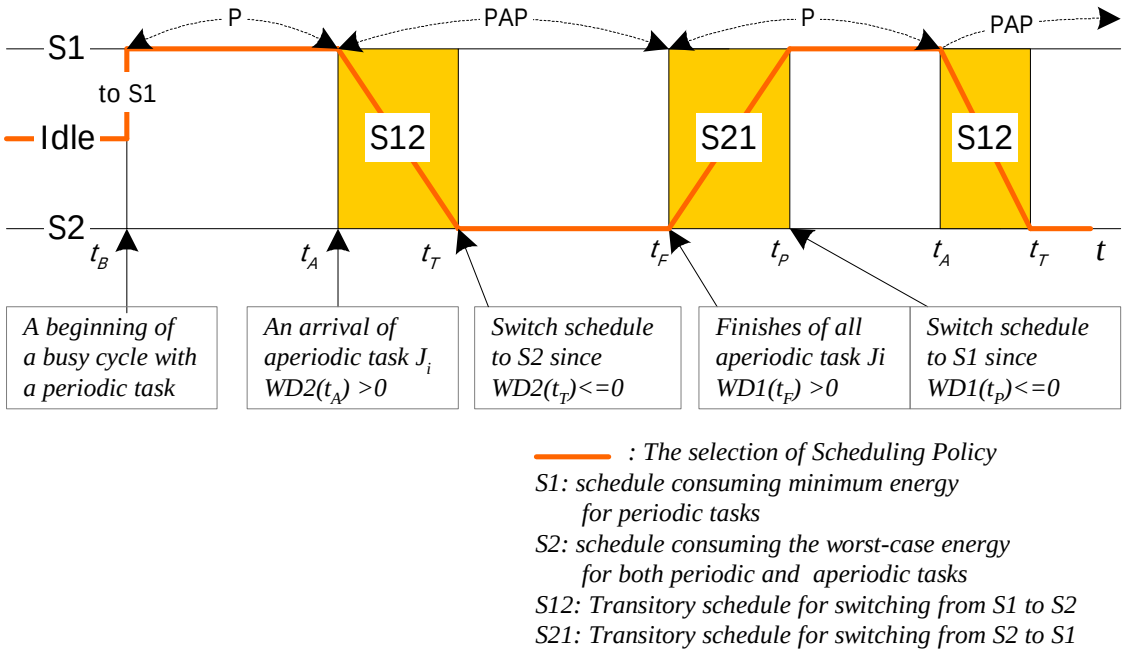


Figure 4-2 Switching schedules for a busy cycle starting with a periodic task and non-zero *WD*

The detailed functions for each instant and interval in Dual-Policy Dynamic Scheduling when a periodic task opens a busy cycle are as follows:

At time t_B: An arrival of periodic task starts *P* interval and initiates scheduling by *S1*schedule since there's no aperiodic task to execute. An aperiodic task's arrival at t_B directly

starts *PAP* interval without going through T_{BA} and *WD2* (t_B) = *WD2* (t_A) and is shown in Figure 4-3.

For interval T_{BA}: As there's no aperiodic task, periodic tasks are executed in the running modes consuming minimum energy without violating their schedulability, *S1*: $\{m_i\}$, instead of in the modes of worst-case schedule, *S2*: $\{m_i\}$.

At time t_A: An aperiodic task J_k arrives. It's time to return back to the worst-case schedule *S2* from *S1* schedule such that the target schedule is *S2* now. The state of *WD2* (t_A) should be checked at the instant. The arrived aperiodic tasks can get either finite deadline based on the *MTBS* algorithm when *WD2* (t_A) ≤ 0 or infinite deadline when *WD2* (t_A) > 0. And if *WD2* (t_A) ≤ 0, the instant becomes exactly the same as the instant of t_T. If *WD2* (t_A) > 0, which means the workload left on the next target schedule policy of *S2* is more than it can afford to execute, the transitory schedule *S12* is chosen till the schedulability condition is satisfied to make the tasks schedulable by *S2* schedule and the *MTBS* assigns an infinite deadline to the aperiodic task. Otherwise, the workload left behind is less than or equal to the one is schedulable by *S2*, the instant of t_A is merged directly to t_P without going through transitory schedule *S12*.

For interval T_{AT}: Since *WD2* (t_A) is bigger than zero at time t_A, the scheduler cannot jump directly to *S2* schedule, transitory schedule *S12* is chosen. Running all tasks in *H-mode* reduces the workload remaining at t_A by *S1* schedule for the interval T_{BA}, i.e. non-zero workload difference between *S2* and *S1*, and finally makes it at the same level that *S2* schedule can afford. *WD2 (t)* is still the work difference between worst-case based on *S2* scheduling and all *H-mode* real execution. Catching up the workload level of *S2* by executing all tasks in *H-mode* consumes more energy than executing tasks in the modes by the worst-case schedule *S2*. As long as *WD2 (t)* is greater than zero, the deadlines of aperiodic tasks that arrive for this interval are assigned as infinite. When *WD2 (t)* is less than or equal to zero, the moment is transferred to time t_T, allowing the scheduler to select *S2* schedule from the instant of t_T as scheduling policy.

At time t_T: Because *WD2* (t_T) is less than or equal to zero at the instant, *S2* schedule is chosen as scheduling policy. Thus, all of aperiodic tasks that arrived before time t_T but got infinite deadlines for the interval T_{AT} are activated from at t_T by getting finite deadlines. The deadlines are determined based on their execution demands and deadline assigning equation (4-6) in *MTBS* algorithm. From the instant t_T, all aperiodic tasks that arrive at the scheduler get finite deadlines.

For interval T_{TF}: The *MTBS* assigns finite deadlines according to the execution demands of incoming aperiodic tasks till no more aperiodic tasks are available in the scheduler. For the interval T_{TF}, if any task under the voltage settings of the worst-case schedule *S2* makes the real execution workload greater than the worst case one, i.e. non-zero *WD2 (t)*, the schedulability can not be guaranteed. Therefore, *WD2 (t)* must be checked whether it becomes greater than zero by the different running modes. All tasks are run in high-speed to make the scheduler keep schedulable if *WD2 (t)* become greater than zero.

At time t_F: All of the aperiodic tasks in the scheduler have been completed. *S1* schedule that consumes lower power becomes the target scheduling policy from the instant. But the possible non-zero workload difference transferred by taking *S2* as scheduling policy previously may cause at least one of the tasks to miss its deadline. Therefore, tasks' schedulability is checked again at the instant before setting running modes to the target schedule. Since the current target scheduling policy is *S1* schedule, *WD1* (t_F) is checked out whether the scheduler can directly select *S1* schedule or should go through *S21* schedule. If *WD1* (t_F) ≤ 0, the instant becomes exactly the same as the instant of t_P and *S1* schedule is selected, guaranteeing the schedulability. Otherwise, *S21* schedule that executes all tasks in *H-mode* is chosen.

For interval T_{FP}: Since *WD1* (t_F) is greater than zero at time t_F, every task is executed in *H-mode* to reduce the excessive workload for *S1* schedule, consuming more energy by executing in *H* instead of in *L-mode* in the schedule of *S1*.

At time t_P : Due to the speed-up execution in *H-mode*, *WD1 (t)* becomes less than or equals to zero at time t_P. So, *S1* schedule is selected at time t_P.

For interval T_{PA}: *S1* schedule is applied to periodic tasks till another first aperiodic task arrives in the busy cycle, getting more execution time in *L-mode* than in *H-mode* and bringing energy saving to the system.

Figure 4-3 shows an example, in which a busy cycle also starts with a periodic task like shown in Figure 4-2. But now it satisfies the scheduability of elementary schedules at time t_A and time t_F, at which the next scheduling policy, not being all work differences higher than zero, i.e. *WD1 (t)<=0* and *WD2 (t) <=0*. Thus the scheduler can directly take elementary schedules of *S1* and *S2* without going through the transitory schedules *S12* and *S21*. Thus, the time of t_A is the same with time t_T by *WD2* (t_A) *<=0* and so is the time of t_F with t_P by *WD1* (t_F) *<=0*. The energy saving is expected to be maximized if busy cycles are composed of the events meeting schedulability, *WD2* (t_A) *<=0* and *WD1* (t_F) *<=0*, making all the moments of t_A and t_F to t_T and t_P, since there is no taking of transitory schedules such that no speed-up execution running all tasks in *H-mode* by *S12* or *S21*.

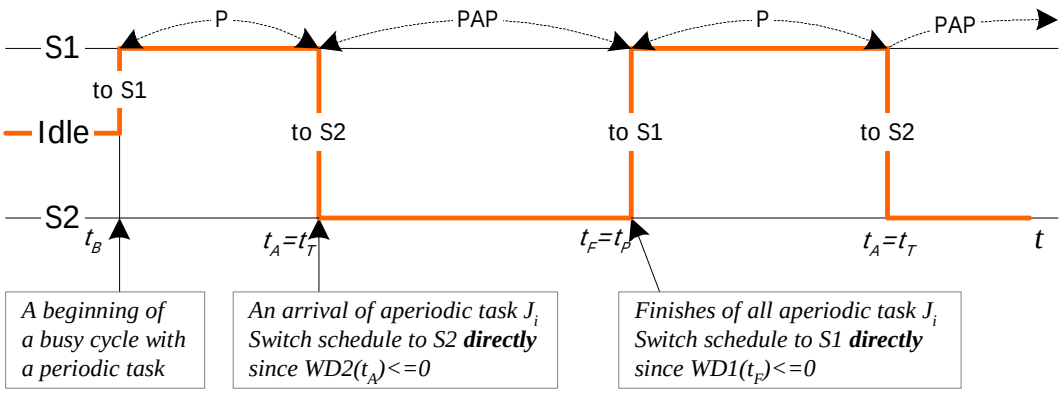


Figure 4-3 Switching schedules for a busy cycle starting with a periodic task and all *WD (t)≤0*

In Figure 4-4, the first arrival in a busy cycle is done by an aperiodic task J_k, making the workload difference *WD2* (t_B) at time t_B equal to zero since no task is waiting in dynamic scheduler except for the just arrived aperiodic task at t_B. The zero workload difference lets the elementary schedule *S2* be selected to schedule mixed tasks. And the real execution demands for arriving periodic tasks are less than the *WCET*'s of *S2* schedule, *WD2 (t)* remains equal to or less

than zero till *S1* schedule is activated with the completion of all waiting aperiodic tasks and it makes the time t_T be matched with t_B and t_A. As soon as the last aperiodic task in dynamic scheduler is completed at time t_F, *WD1* (t_F) is checked whether the voltage settings of tasks can be switched to the energy-saving mode schedule *S1*. From the time t_F, the scheduling policy selection follows the same procedures as shown in Figure 4-2.

To show the workload difference at the moment of switching scheduling policies and how to change scheduling policies, we show examples of the switching moments in Figure 4-5 and Figure 4-6, which have transitory schedules from *S1* to *S2* schedule and from *S2* to *S1* schedule, respectively. In both figures, $W_2(t)$, $W_1(t)$, and $W_0(t)$ represent the workload demands of worst-case schedule *S2* for mixed tasks, of worst-case schedule *S1* for only aperiodic tasks (for energy saving), and of real execution for both periodic and aperiodic tasks, respectively.

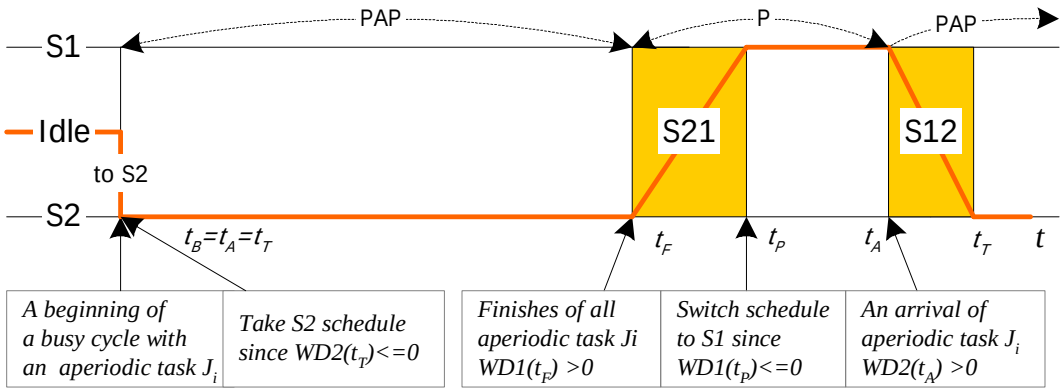


Figure 4-4 Switching schedules for a busy cycle starting with an aperiodic task and *WD (t) >0*

In Figure 4-5, before the arrival of aperiodic task, voltage settings of *S1* schedule are chosen for the periodic tasks and $W_0(t)$ shows a decreasing slope by execution in low speed (*L-mode*) along the time *t*. $W_2(t)$ also represents the decrement in workload by execution in high-speed (*H-mode*). The discrepancy in execution speeds of tasks between *S1*and *S2* schedules appears as non-zero workload difference *WD2 (t)* at time t_A and this makes at least one of the periodic tasks miss its deadline after t_A from the viewpoint of schedulability in target scheduling

policy *S2* at t_A for both periodic and aperiodic tasks. With the arrival of aperiodic task J_k at time t_A, both $W_2(t_A)$ and $W_0(t_A)$ jump up by the amount of execution demand, but there's no change in $WD2(t_A)$. Before taking *S2* schedule as scheduling policy from time t_A, *WD2 (t)* is reduced to be equal to or less than zero.

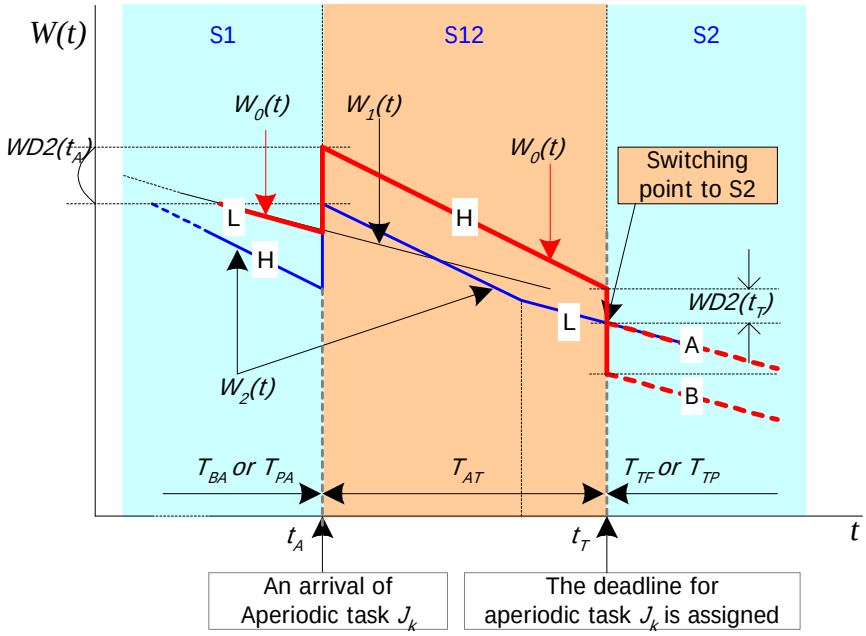


Figure 4-5 Switching from *S1* to *S2* schedules

So, all periodic tasks are executed in high-speed and aperiodic tasks get infinite deadlines until the *WD2 (t)* becomes zero or less than zero. Applying *S12* schedule, the execution in the same speed as *S2* schedule is useless to decrease *WD2 (t)*. *WD2 (t)* can be reduced only when the tasks are executed in *H*-mode, but assigned in *L*-mode for worst-case *S2* schedule. In addition, even with the same speed execution in high-speed with *S2* schedule, *WD2 (t)* can be lowered for every completion of periodic task by the slack time from the difference of WCET and real execution demand of periodic tasks. In Figure 4-5, execution in *H*-mode for *S12* schedule, but assigned in *L*-mode for *S2* schedule makes $WD2(t_T) \leq WD2(t_A)$. And, both workload lines A and B represent that the last periodic task in interval T_{AT} brought to non-zero slack time. Line B is for the case having bigger slack time than line A. From time t_T, with zero or less than zero $WD_2(t_T)$

aperiodic tasks can finally get their finite deadlines based on the utilization given by the equation (4-2) in modified total bandwidth algorithm.

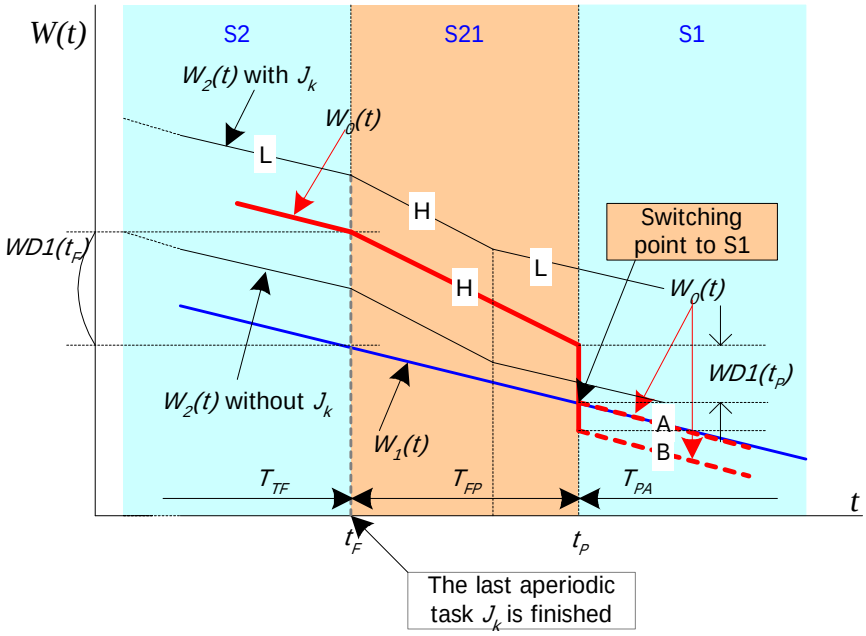


Figure 4-6 Switching from *S2* to *S1* schedules

Figure 4-6 shows the trajectory of workloads when the dynamic scheduler has finished all aperiodic tasks with the last completion of aperiodic task J_k and switches schedules based on the status of workload differences. By the instant t_F since there is aperiodic tasks to be scheduled and workload $W_0(t)$ is less than the one of $W_2(t)$ thanks to the slack times of periodic tasks, the worst-case schedule *S2* is chosen. At the time t_F, *S1* schedule becomes the target scheduling policy under the purpose of lowering energy consumption. However, the non-zero workload difference $WD1(t_F)$ may put the scheduling at the risk of violating any deadline of periodic tasks. Thus, a transitory schedule *S21* is selected to reduce $WD1(t_F)$ to schedulable level. Likewise the case of transit interval from *S21* to *S2*, the slack times from the difference between WCET in *S1* schedule and real execution demand of a periodic task lower the workload difference at time t_F, $WD1(t_F)$, to be zero or less than zero such that the scheduler can switch its voltage settings of from *S21* to *S1* schedule at time t_P. Also, executing tasks in all *H-mode* regardless of *L-mode* assigning in *S1* schedule reduce the difference in workload. The stepwise decrease to either the line A or B at

time t_P in Figure 4-6 represents different non-zero slack times by the last periodic tasks in the interval T_{FP}. From the time t_P, the change scheduling policy from *S2* schedule to *S1* schedule may reduce the total energy consumption.

4.5 Algorithm for Dual-Policy Dynamic Scheduling

Given the voltage assignments defined in *H* and *L*, tasks can be dispatched and run in the assigned mode. Let the scheduling be called as *Single-Policy Dynamic Scheduling*. The voltage settings are determined by allocating a constrained energy budget to periodic and aperiodic tasks, with the objective of better performance for aperiodic tasks and guaranteeing schedulability for periodic tasks. SPDP schedules tasks by the worst-case schedule *S2* with the total bandwidth sever.

Instead of maintaining the worst-case schedule determined for single-policy dynamic scheduling, *Dual-Policy Dynamic Scheduling* schedules tasks by intermixing the worst-case schedule with energy-saving schedule for periodic tasks such that energy consumption can be reduced as much as is available by the effect of increasing execution time in *L-mode* instead of in *H-mode* from worst-case assignment.

In Table 4-1, we summarize the flags to signal the status of *WD (t)* and aperiodic task's existence in the scheduler. If *WD (t)* becomes greater than zero at time *t*, scheduler set the *TransitFlag* to ON. Otherwise, it sets the flag to OFF. And, if there's any aperiodic task in the scheduler, scheduler set the *ApFlag* to ON. Otherwise, it sets the flag to OFF.

Table 4-1 Switching Scheduling Policies with respect to the Interval Changes

Flags			Scheduling Policy		Energy Consumption
TransitFlag	ApFlag	Interval	Dual	Single	
OFF	OFF	T_{BA} or T_{PA}	S1	S2	Energy Saving
ON	ON	T_{AT}	S12	S2	At most worst-case
	OFF	T_{FP}	S21	S2	
OFF	ON	T_{TF} or T_{TP}	S2	S2	At most worst-Case

During the intervals, not having aperiodic tasks and having both flags with the state 'OFF', energy saving is achieved by changing running mode in *S1* schedule instead of *S2* schedule.

The detail algorithm for the Dual-Policy Dynamic Scheduling is given in Figure 4-, where the following definitions are used, including the flags shown in Table 4-1.

TK–Q: the normal EDF task queue for all of periodic and aperiodic tasks in the system.

S1–Q: the normal EDF task queue only for all periodic tasks in the system. It is used for following currently how much workload remained by energy-saving schedule *S1*.

S2–Q: another normal EDF task queue for all of periodic and aperiodic tasks in the system. It is used for following currently how much workload remained by worst-case schedule *S2* for mixed tasks.

enQueue(element, Q): insert an element into the queue of *Q*.

enQueue(element, Q): insert an element into the queue of *Q*.

head(Q): the element at the head of queue *Q*.

compute-exeTime () : adjust the status of queues by the executed time since the last dispatch instance based on execution time demands

Timer (period): if it is *on* with the amount of *period*, a timer is set to the required time *period*. Otherwise it is off, the timer is stopped and maintaining the remaining amount at the stopped time.

WD_ DNTO_ZERO and *WD_ UPTO_ZERO*: parameter to indicate the time amount that *WD (t)*'s are reduced and increased, respectively, by currently dispatching tasks in both *TK_Q* and either *S1_Q* or *S2_Q* according to the status of flags. *WD_ DNTO_ZERO* is used for the checking whether *WD1 (t)* or *WD2 (t)* hits the zero when transitory schedules are used for the intervals of T_{AT} and T_{FP}. *WD_ UPTO_ZERO* is also used for the checking whether

WD1 (t) or *WD2 (t)* hits the zero to keep track of the schedulability outside of the transient intervals of T_{AT} and T_{FP}. Both parameters are watched at every event occurrence.

To track the worst-case workload and to compute workload difference using combined EDF scheduling with voltage-clock scaling, we employ three queues, i.e., *TK-Q*, *S1-Q* and *S2-Q*, in which tasks are ordered according to their respective deadlines and expiration times. *S1-Q* and *S2-Q* are for worst-case workload but *TK-Q* for actual workload. *TK-Q* and *S2-Q* take both periodic and aperiodic tasks, whereas *S1-Q* does only periodic tasks. By the remaining workload between *TK-Q* and either *S1-Q* or *S2-Q*, the workload differences, *WD1 (t)* and *WD2 (t)*, are computed to check the schedulability by the targeting worst-case schedule. In addition to events of arrival and completion, the moments of when *WD1 (t)* or *WD2 (t)* become zero or less than zero are the scheduling-policy decision points, being tracked by the timer values of *WD_ DNTO_ZERO* and *WD_ UPTO_ZERO*.

When any first aperiodic task arrives after interval *P*, *ApFlag* is set to ON and *WD2 (t)* is compared with zero. The *ApFlag* is maintained as ON till the last aperiodic instance is finished. If *WD2 (t)* is equal to or less than zero, *TransitFlag* is set to OFF; the deadline of the aperiodic instance is computed based on equation (4-6) and it is inserted into both *S2-Q* and *TK-Q* according to the target scheduling policy *S2*. Otherwise, *TransitFlag* is set to ON; the aperiodic task is inserted into *S2-Q* with the finite deadline by the equation (4-6) but into *TK-Q* with infinite deadline by the equation (4-7), complying with the transitory schedule *S12*. As long as *WD2 (t)* is greater than zero, *TransitFlag* is maintained as ON, arriving aperiodic instances are inserted into *TK-Q* with infinite deadline and not dispatched. When *WD2 (t)* becomes zero or less than zero, i.e. the *S2* schedule afford to meet schedulability, *TransitFlag* is set to OFF; all un-dispatched aperiodic instances are assigned by their finite deadlines by the equation (4-6), switching to *S2* schedule. Once the scheduler has chosen the targeting schedule *S2*, arriving aperiodic instances are considered as periodic tasks by assigning their deadlines and inserted into both *S2-Q* and *TK-Q*.

As soon as the last aperiodic instance that arrived has been completed, *ApFlag* is set to OFF and *WD1 (t)* is compared with zero. The *ApFlag* is maintained as OFF until any new aperiodic instances arrive. If *WD1 (t)* is equal to or less than zero, *TransitFlag* is set to OFF; scheduling policy is switched from *S2* to *S1* schedules. Otherwise, *TransitFlag* is set to ON; it goes through transitory schedule *S21* till *WD1 (t)* is equal to or less than zero, at which *S1* schedule is selected to turn on the energy saving scheduling policy.

With a new arrival of aperiodic instance, the whole process is repeated.

Algorithm for Dual-Policy Dynamic Scheduling

at the arrival instance av_i of task τ_i {

```
Timer()= off;
if ((TK-Q != empty) or (S2-Q != empty) or (S1-Q != empty))
    compute-exeTime;

    compute-wd;              // compute WD (t)
    check_flags;

    if ( TK-Q == empty ) {
        BusyFlag= on;

    if (τi == aperiodic task) {
        ApFlag = on;
        compute task deadline Di based on TBS algorithm;
        set task deadline to avi+Di;
    }
    else {
        set task deadline to avi+Di;
        enQueue(τi, S1-Q);
    }
    enQueue(τi, TK-Q);   // enqueue the arriving task with deadline Dj
    enQueue(τi, S2-Q);
}
else {
    if (τi == aperiodic task) {
        compute task deadline Di based on TBS algorithm;
        set task deadline to avi+Di;
        enQueue(τi, S2-Q);

        if (TransitFlag==on) set task deadline to ∞;
        else                 set task deadline to avi+Di;
        enQueue(τi, TK-Q);   // enqueue the arriving task with deadline ∞
    }
    else ….
```

Figure 4-7 The algorithm for Dual-Policy Dynamic Scheduling

```
        else {
            set task deadline to av_i+D_i;
            enQueue(τ_i, TK-Q);  // enqueue the arriving task
            enQueue(τ_i, S2-Q);
            enQueue(τ_i, S1-Q);
        }
    }
    compute-wd;          // compute WD (t)
    dispatch;
}

at the completion instance of a task {

    Timer()= off;
    if (the task from TK-Q) {
        τ_i = deQueue(TK-Q);
        compute-exeTime;

        compute-wd;          // compute WD (t)
        check_flags;
        if ( τ_i == aperiodic task) {
            compute-responseTime;
            if (the last aperiodic task) { // Prepare Transition mode
                if (WD1(t) <=0)
                    TransitFlag = off;
                else   TransitFlag = on;
                ApFlag = off;
            }
        }
        else
            check_meetDeadline;

        if (TK-Q==empty) {
            BusyFlag= off;   // idle period starts
            TransitFlag = off;
            ApFlag = off;
        }
    }
    else if (the task from S1-Q){
        τ_i = deQueue(S1-Q);
        compute-exeTime;
        compute-wd;          // compute WD (t)
        check_flags;
    }
    else {
        τ_i = deQueue(S2-Q);
        compute-exeTime;
        compute-wd;          // compute WD (t)
        check_flags;
    }
    dispatch;
}
```

Figure 4-7 (continued.)

```
at the expiration of Timer {    //

    if (TransitFlag==on) {
        TransitFlag = off;
        if (ApFlag==on) {
            for all aperiodic task τj in TK-Q {
                compute task deadline Dj based on TBS algorithm
                set task deadline to avj+Dj;
            }
        }
        dispatch;
    }
    else { // to make the scheduler schedulable
        τj = head(TK-Q);
        set voltage-mode to H;
    }
}

Procedure compute-wd {        // adjust accumulated execution time
    W0(t) = added-up workload to be executed in TK-Q;
    W1(t) = added-up workload to be executed in S1-Q;
    W2(t) = added-up workload to be executed in S2-Q;

    WD1(t) = W0(t) – W1(t);
    WD2(t) = W0(t) – W2(t);
}

Procedure check_flags () {  // change flags
    if(ApFlag ==on)
            WD (t)= WD2 (t);
    else
            WD (t)= WD1 (t);

    if (WD (t) <=0)
            TransitFlag =off;
    else
            TransitFlag =on;
}

Procedure dispatch ()  { // dispatch a task
    activate_S1S2 ();
    dispatch_S1S2 ();

    if (TransitFlag == on)
        dispatch_ transit ();
    else {
        if (ApFlag == on)
            dispatch_S2mode();
        else
            dispatch_S1mode();
    }
}
```

Figure 4-7 (continued.)

```
Procedure dispatch_S1mode ()  {       // assign execution mode
          // to low-energy consumption mode,  S1 voltage settings
   if (TK-Q!= empty) {
      τj = head(TK-Q);
      τk = head(S1-Q);

      if ( τj ∈ LS1 ) {
         set voltage-mode to L;
         if ( τk ∈ HS1 )
               compute WD_UPTO_ZERO from τj and τk;
         else   WD_ UPTO _ZERO=0;

         if (WD_ UPTO_ZERO> abs(WD1(t)) Timer(WD1(t))= on;
      }
      else    set voltage-mode to H;
   }
}

Procedure dispatch_S2mode ()  {  // assign execution mode
                                 // to worst-case mode
   if (TK-Q!= empty) {
      τj = head(TK-Q);
      τk = head(S2-Q);

      if ( τj ∈ LS2 ) {
         set voltage-mode to L;

         if ( τk ∈ HS2 )
            compute WD_UPTO_ZERO from τj and τk;
         else WD_ UPTO _ZERO=0;

         if (WD_ UPTO_ZERO> abs(WD2(t))
            Timer(WD2(t))= on;
      }
      else    set voltage-mode to H;
   }
}

Procedure dispatch_S1S2 ()  { // assign execution mode
                              // to follow worst cases of S1 and S2's
   if (S1-Q!= empty) {
      τj = head(S1-Q);
      // S1 voltage settings
      if (τj ∈ HS1)       set voltage-mode to H;
      else                set voltage-mode to L;
   }
   if (S2-Q!= empty) {
      τj = head(S2-Q);
      // S2 voltage settings
      if (τj ∈ HS2)       set voltage-mode to H;
      else                set voltage-mode to L;
   }
}
```

Figure 4-7 (continued.)

```
Procedure dispatch_transit ()  { // assign execution mode
                                 // to High-mode
    if (ApFlag ==on)          WD (t)= WD2 (t);
    else                      WD (t)= WD1 (t);

    τj = head(TK-Q);
    compute WD_DNTO_ZERO by assuming τj in H-mode;

    if (WD_ DNTO_ZERO>WD (t))
        Timer(WD (t))= on;
    set voltage-mode to H;
}
```

Figure 4-7 (continued.)

4.6 Performance Evaluation

We analyze here the responsiveness and energy saving in Dual-Policy Dynamic Scheduling, sharing the bounded energy budgets between periodic and aperiodic tasks based on VCS approach and evaluate the VCS-EDF scheme to schedule mixed real-time tasks. Over Single-Policy Dynamic Scheduling, whose voltage settings are found in static scheduling, the superiority of DPDS, the dynamic scheduling scheme of switching scheduling policies to the pattern changes of task events, is gauged by energy saving and responsiveness along the various energy constraints available by the task set. For the power consumption and speed settings, Motorola's PowerPC 860 processor is used for our simulation, which can be operated in a high-performance mode at 50MHz and with a supply voltage of 3.3V, or a low-power mode at 25MHz and with an internal voltage of 2.4V [77] such that V_H and V_L are fixed to V_H=*3.3* and V_L=*2.4*. The power consumption in the high-performance mode is 1.3 Watts (p_H), as compared to 241mW (p_L) in the low-power mode. The clock rate at high voltage is 100% faster than at low voltage: α_H=*2.0* and α_L=*1.0*. The worst-case execution demands of the tasks are chosen in the range of α_L to α_H. The real task execution demand is then selected randomly such that the mean is in the range of 0.6 to 1.0 of the worst-case execution demand.

In our simulation, we use a set of 10 periodic tasks having random task periods in the range of 100 to 1000 and set the task deadlines equal to their respective periods. The worst-case

execution demands of the tasks are randomly chosen such that, for each simulation case, no deadlines should be missed and the resultant utilization is set to U_P *(L)* from 0.8 to 1.4. The sum of worst-case execution demands of both periodic and aperiodic tasks is also chosen in the range of α_L to α_H. The real task execution demand is then selected randomly such that the mean is in the range of 0.6 to 1.0 of the worst-case execution demand. For aperiodic tasks, we adopt the exponentially distributed execution time with an average μ equal to 45. Then we let the inter-arrival time be exponentially distributed with mean of between 450 (10% workload, i.e. U_A *(L)*=0.1) and 64.3 (70%, i.e. U_A *(L)*=0.7). The energy budget E_C is set at each of several energy levels in the range from (E_{min}+0.6E_{diff}) to (E_{min}+ E_{diff}) by the increment of 0.1E_{diff}.

The improvement of energy saving and average response time is addressed by a simulation study performed under the assumption of enough energy to complete the tasks and meet the deadline requirements. In addition, the simulation study follows the objective in the constrained energy allocation model gives best responsiveness when aperiodic tasks are assigned in high-voltage and high-speed mode. Throughout the performance analyses, the dual-policy dynamic scheduling is compared to single-policy dynamic scheduling, which schedules tasks only in the voltage settings assigned in static scheduling. In the circulated changes of the intervals T_{idle}, T_{BA}, T_{AT}, T_{TF}, T_{FP}, and T_{PA}, periodic tasks are executed in the other speed and voltage from assigned ones of the worst-case. For efficient comparisons, we introduce several parameters, *HtoL*, *LtoH*, and *T-LH* to accumulate executed times under the following conditions. All of them will be compared after being normalized to the total simulation time.

- *HtoL*: the accumulated time for which some tasks assigned as *H-mode* in the worst-case schedule *S2* are executed in *L-mode* by *S1* schedule during the intervals of T_{BA} and T_{PA}.
- *LtoH*: the accumulated time for which some of tasks assigned as *L-mode* in both schedules of *S1* and *S2* are executed in *H-mode* during the intervals of T_{AT}, T_{TF}, T_{FP}, and T_{PA}.
- *T-LH*: the accumulated time for which some of tasks assigned as *L-mode* in *S2* schedule is executed in *H-mode* by *S12* schedule during the transitory intervals of T_{AT}. The amount is less than *LtoH*.

More energy is saved from the difference in energy consumption between two-speed modes. Thus, *HtoL* stands for how much energy can be saved. And *T-LH*'s is added up by the workload difference generated from the switched computation from low to high-speed mode when *S12* schedule is selected. In the DPDS, the workload difference is decreased by two factors. First, it is slack periods available whenever a periodic task completes. So, the average of actual workload of periodic tasks tells how fast the workload difference is reduced. The other one is the switched mode execution from low to high speed. If non-zero workload is still remained at switching point, doubled execution speed from low to high speed contributes to clearing the workload difference up, advancing the moment when the aperiodic tasks are assigned their finite deadlines. From the switching point, if the execution times in doubled speed, *T-LH*'s, are accumulated, it can be a measure for judging the responsiveness for aperiodic tasks.

4.6.1 Energy Consumption

How much increase in low-speed execution times is obtained for the dual-policy dynamic scheduling? Over several energy budgets controlled by E_{diff}, we obtain the percentage of executed times in *H-* and *L-mode* to the total execution time from the simulation and compare them in where the occupations by all components of periodic, aperiodic tasks and idling are represented in stacks of bar. '*H for P*', '*L for P*', '*H for AP*', and '*L for AP*' are representing how much *H-mode* (*H:*high-speed) and *L-mode* (*L:*low-speed) executions for periodic (*P*) and aperiodic (*AP*) tasks take the processor's computing capacity. '*Idle*' represents the ratio of processor's idling times without computation to the total execution time. Here, the more processor capacity is used for specific-speed execution of tasks, the longer bar it takes in the stack of bars.

Under a fixed utilization of aperiodic tasks, the percentages by the worst-case execution demands are measured and shown in Figure 4-8 (a), (d) and Figure 4-9 (a), (d), respectively, increasing the utilization of periodic tasks from 0.8 to 1.4 by 0.2. Under the real execution time demand of 0.6 for periodic tasks, the percentages of executed times for the single-policy are shown in Figure 4-8 (b), (e) and Figure 4-9 (b), (e), and the percentages for dual-policy in Figure

4-8 (c), (f) and Figure 4-9 (c), (f), respectively. As for the percentage of aperiodic tasks, since the fast responsiveness is obtained when a constrained energy budget is allocated to execute them in high-speed and high-voltage mode in the study [83], the utilization by aperiodic tasks becomes 0.15 when the utilization of 0.3 for aperiodic tasks in low-speed. Thus, regardless of increase in the energy budget, the lengths of bar '*H for AP*' are the same as 0.15 and '*L for AP*' is zero. With the increase of energy budget allocated to periodic tasks, the portion of '*H for P*' is increased from 0.183 to the maximum 0.4 when E_C is given as E_{max}= $maxE_A$ + $maxE_P$ = (E_{min}+ E_{diff}). '*L for P*' is decreased from 0.434 to zero, causing '*H for P*' to be maximum and the total utilization by periodic tasks, the sum of '*H for P*' and '*L for P*', to be minimum at E_C = E_{max}. The remaining utilization is allocated to aperiodic tasks and it gives decreased deadlines to them by total bandwidth algorithm. This leads to fast responsiveness.

With the increase of U_P as shown in Figure 4-8 (a), (d) and Figure 4-9 (a), (d), note that the ratios of *idle* are decreased as the total utilization by periodic tasks is increased. According to the TBS algorithm, it directly implies the increment in response times due to the lower assignment of utilization for aperiodic tasks, given by the equation $U_A(S2) \leq 1- U_P(S2)$. In Figure 4-9 (d), (e), and (f), interestingly, execution times are not available at energy allocation of E_C =(E_{min}+ 0.6 E_{diff}), since the energy assignment is not good enough to schedule under U_P=1.4 and U_A=0.3.The stacks of bar in Figure 4-8 (b), (e) and Figure 4-9 (b), (e), shows the percentages of each elements by the real execution demands of 0.6 for periodic tasks and U_A=0.3. The bars are showing the same characteristics as shown in static scheduling except that periodic tasks have exactly 60% of the worst-case execution demands. The exact amount of generated real execution demands are executed as shown in the length of the bars for periodic tasks in single-policy.

Comparing (b) to (c) and (e) to (f) in Figure 4-8 and, there are big differences in the lengths of '*H for P*' and '*L for P*' over all energy budgets. The same amounts of '*H for P*' as the single-policy dynamic scheduling are tremendously reduced and show a much slower increasing slope than single-policy with the increase of energy budgets. On the contrary, the same amounts

of '*L for P*' as the single-policy dynamic scheduling are tremendously increased. But they also show a much slower increasing slope with the increase of energy budgets like '*H for P*'. The reduction of '*H for P*' and the increase of '*L for P*' are coming from the switching from worst-case schedule *S2* to energy-saving schedule *S1* whenever the event pattern having only periodic tasks is available.

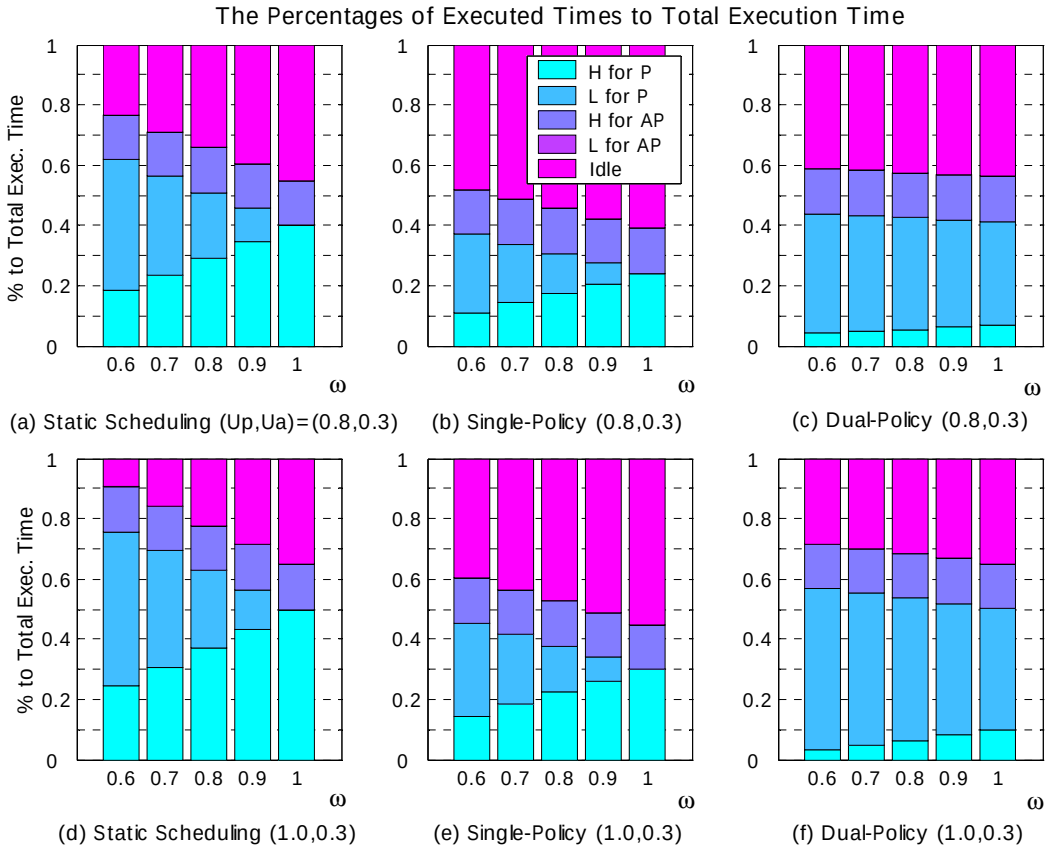


Figure 4-8 The percentages of executed times to the total execution times over a various energy budget in cases of U_P=0.8 and 1.0 for a fixed U_A=0.3
(the average of real workload for periodic tasks is 0.6)

The use of two scheduling policies decreases the chances of execution in *H-mode* and increases execution in *L-mode*. The switched mode execution in the ratios of components to total executed time are positively related to energy saving. This indicates the effectiveness of switching scheduling policies in a system for mixed task. As we increase the workload demand of periodic tasks U_P as shown in Figure 4-8 (f), and Figure 4-9 (c), (f), the increase of '*L for P*' is

slow down. This indicates that, as the utilization of periodic tasks getting higher, more processor processing capacity should be dedicated to high-speed execution to make sure about the schedulability.

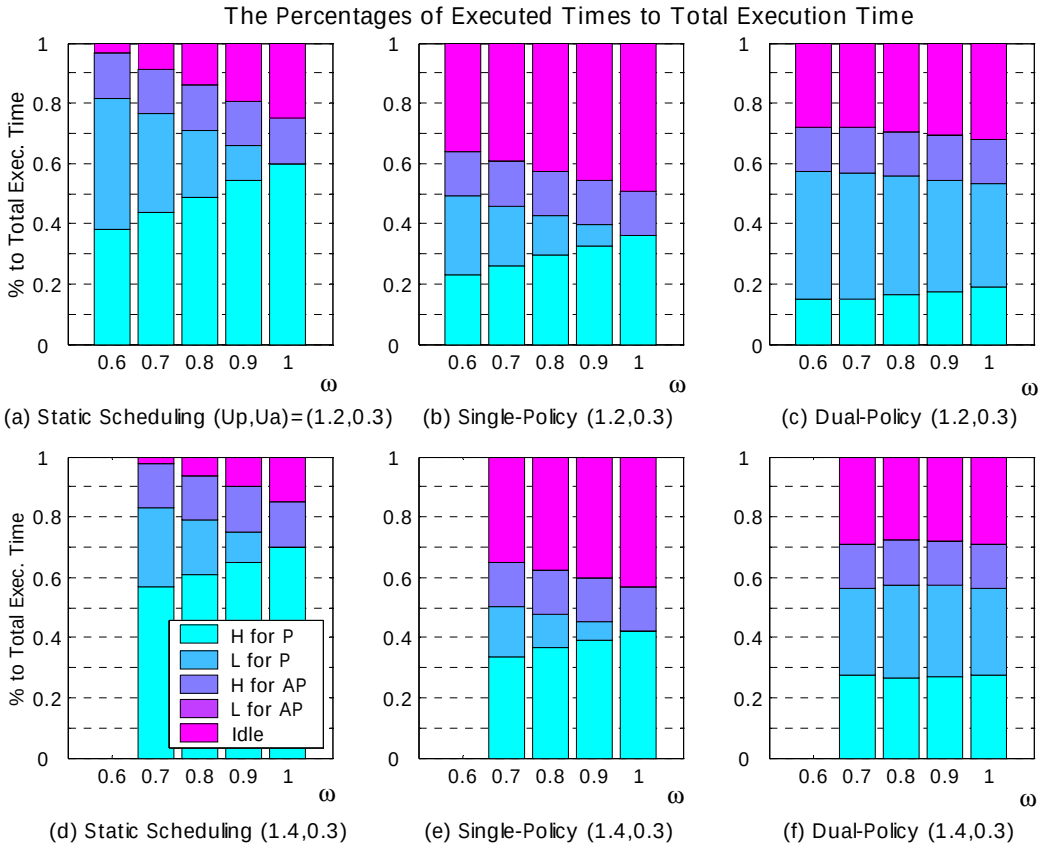


Figure 4-9 The percentages of executed times to the total execution times over a various energy budget in cases of U_P=1.2 and 1.4 for a fixed U_A=0.3
(the average of real workload for periodic tasks is 0.6)

Interestingly, the amounts of increased '*L for P*' become maxima when U_P= 1.0 and diminish with respect to the decrease of U_P from 1.0. When the utilization of periodic tasks is less than 1.0, the portion of *L-mode* assigned tasks in *S1* schedule is absolutely smaller than U_P= 1.0. Thus, the lower *L-mode* assignments in *S1* schedule, the lower possibility in *L-mode* execution instead of *H-mode* by *S2* schedule. To see the relationship between the increment of '*L for P*' by dual-policy and the portion of *L-mode* assignment in *S1* schedule, we compare '*L for P*', '*H for P*', and '*Idle*' for both scheduling schemes including *S1* schedule over execution demands

ranging from 0.8 to 1.4 when $E_C = E_{min} + 0.7\,E_{diff}$ and U_A= 0.149 (0.15) in Table 2. In the column of '*Increase in L*', the increments of total *L-mode* execution times from single-policy to dual-policy are shown as U_P is augmented. When U_P= 1.0, the increment of *L-mode* execution time is maximum and this is due to the highest utilization for assigned *L-mode* in *S1* schedule as shown in the column of '*L*' under '*S1*'.

In addition, Table 4-2 shows the source of the increments in measured low-speed execution times is from that the reduced amount of high-speed execution turns into the increment of low-speed execution time by the switched policies from *S2* to *S1* schedule. For example, when U_P= 1.0, the increased amount is coming from $(H\ in\ SP - H\ in\ DP)\times\alpha_H = (0.185 - 0.047)\times 2 = 0.276$.

Table 4-2 An example of increased *L-mode* execution times for periodic tasks when U_A= 0.3 (Mean of real execution demands = 0.6, $E_C = E_{min} + 0.7\,E_{diff}$, and aperiodic tasks are run in all *H-mode*)

	Single-Policy (SP)			Dual-Policy (DP)			Increase in L	S1		
U_P	H	L	Idle	H	L	Idle	(DP_L–SP_L)	H	L	Idle
0.8	0.142	0.195	0.513	0.047	0.386	0.418	0.191	0.008	0.785	0.207
0.9	0.163	0.214	0.474	0.050	0.439	0.362	0.225	0.007	0.886	0.107
1.0	0.185	0.231	0.435	0.047	0.505	0.298	0.274	0.009	0.982	0.009
1.1	0.222	0.216	0.413	0.096	0.467	0.287	0.251	0.101	0.898	0.001
1.2	0.261	0.198	0.392	0.151	0.418	0.282	0.220	0.200	0.799	0.001
1.3	0.300	0.181	0.370	0.206	0.367	0.277	0.186	0.300	0.699	0.001
1.4	0.338	0.163	0.349	0.276	0.287	0.228	0.124	0.400	0.599	0.001

As we increase the utilization of periodic task U_A under a fixed U_P as 1.2, the percentages of high-speed execution time are measured along the increase of both energy budgets and real execution demands and plotted in Figure 4-10. The curves with the solid lines are the percentage measures from the single-policy dynamic scheduling, whereas are dashed lines based on the dual-policy dynamic scheduling. The percentages increase as we augment the energy budgets for both of the scheduling schemes. The solid lines show slower slopes than the dashed line along with the

raise in energy budgets. No available data in Figure 4-10 (c) and (d) also represent the fact that the energy budgets given to tasks at the regions are not enough to schedule them.

For the increment of the real execution demands from 0.6 to 0.9, the single-policy dynamic scheduling shows equal increments in the percentages, revealing parallels. But, the increasing slopes for the percentages for the dual-policy dynamic scheduling are gradually higher with the augment of the actual workload. For the case of U_A = 0.4 and 0.5 and the real execution demands of 0.6, the percentage measures are slightly convex down at ω = 0.8 and ω = 0.9, respectively. Thus, the more low-speed execution times are available under the conditions.

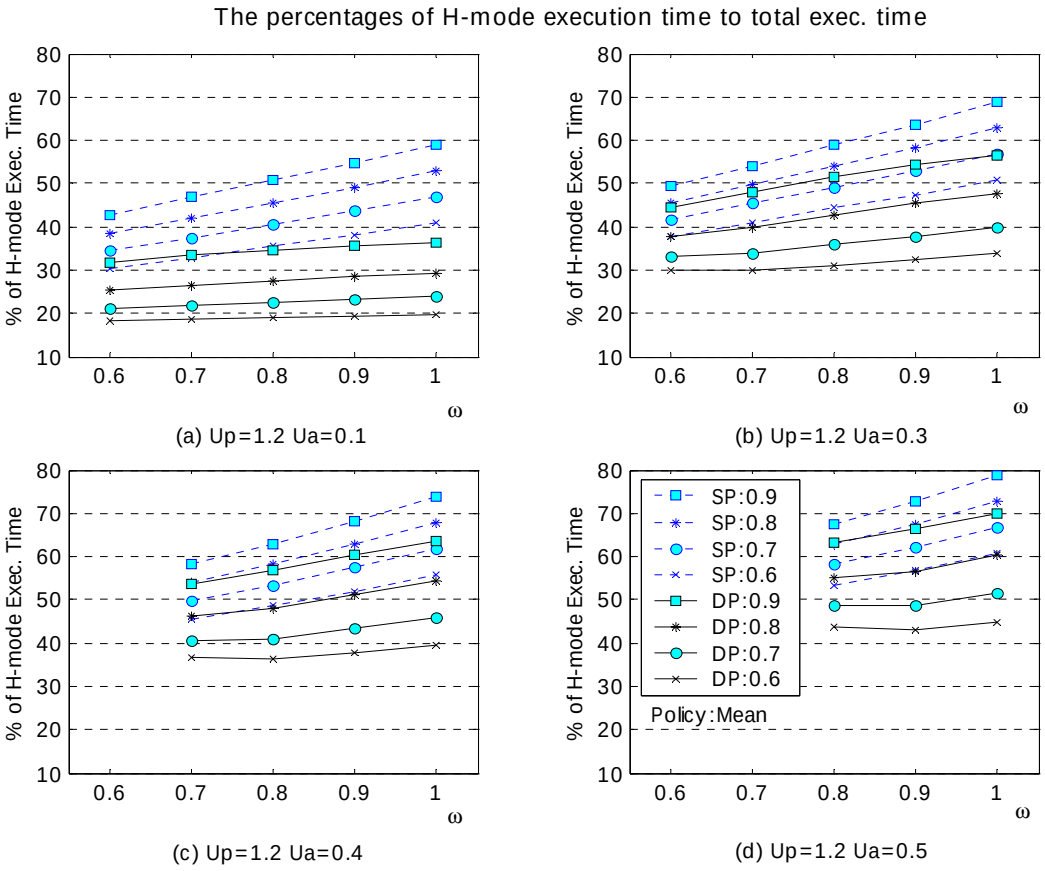


Figure 4-10 The percentages of time executed in high-speed mode to the total execution times over a various energy budget in cases of U_P=1.2
(Solid and dashed lines stand for dual-policy and single-policy, respectively, and the values in the legend for the averages of actual workload for periodic tasks)

The effects of switched mode execution to low-speed from high-speed assignment in worst-case schedule appear as prolonged execution times in *L-mode* and shorten execution times in *H-mode*. And this is under the same simulation given in Figure 4-10, the percentages of low-speed execution time are measured and plotted in Figure 4-11.

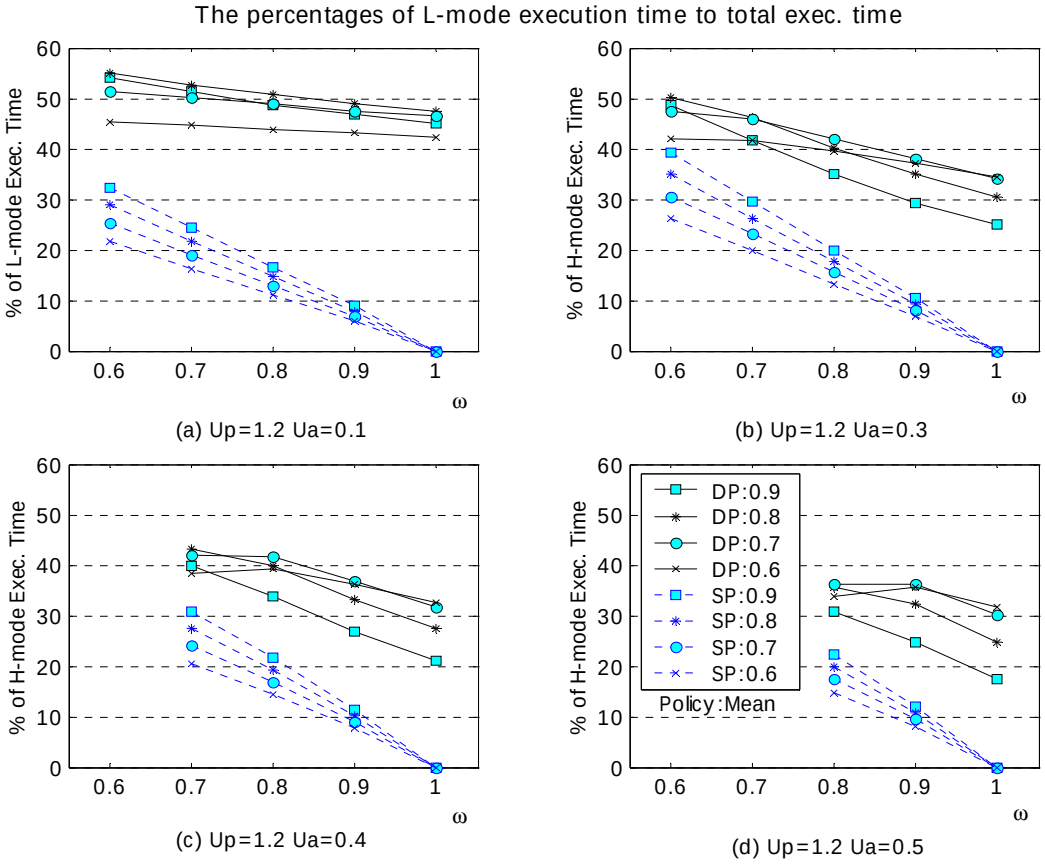


Figure 4-11 The percentages of time executed in low-speed mode to the total execution times over a various energy budget in cases of U_P=1.2
(Solid and dashed lines stand for dual-policy and single-policy, respectively, and the values in the legend for the averages of actual workload for periodic tasks)

The percentage measures for the single-policy dynamic scheduling diminish to zero at the maximum energy budget along the raise in the energy budgets and they also increase along with the augment of actual workloads. But the curves with dashed lines for the dual-policy dynamic scheduling show some interesting non-monotonic variations along with the raises in both energy budgets and actual workloads. As we know in the analyses of the percentage measures in high-speed execution and energy consumption, the down-convexities for the case of U_A = 0.4 and 0.5

and the average actual workloads of 0.6, can affect to the energy consumption when the energy budgets are given as $\omega = 0.8$ and $\omega = 0.9$, respectively.

How much can the dual-policy scheduling reduce energy consumption comparing to the single-policy one? Using the power consumption of Motorola's PowerPC 860 processor, which requires 1.3 Watts and 241mW for the high-performance mode (p_H) and the low-power mode (p_L), the normalized energy consumptions are calculated based on the percentage measures in high- and low-speed execution.

First, to study how the increased low-speed execution times in the dual-policy dynamic scheduling impacts to lowering energy consumption, we calculate the normalized energy consumptions and present them in Table 4-3, based on the percentages of low- and high-speed execution times to the total execution time in Table 4-2. The saved energy becomes maximum when the workload demand U_P= 1.0 due to the more execution times in low-speed than any other workload demands. Compared to the static scheduling and the single-policy dynamic scheduling, the dual-policy scheduling saves (0.686–0.377)/0.686= 45% and (0.490–0.0.377)/0.490= 23.1%, respectively. Given energy budget E_{static}, i.e., E_P, the enhancement of energy saving to the single-policy scheduling is (0.296–0.183)/0.686=16.5%. And as the workload demand is increased, the energy saving is reduced.

For the comparisons between two dynamic scheduling schemes over the variation of average workload of periodic tasks, energy budget, and aperiodic tasks' utilization, the ratios of each scheme to the static scheduling are plotted in Figure 4-12 as the solid lines for the dual-policy dynamic scheduling and the dashed lines for the single-policy dynamic scheduling. As we increase the average of actual workloads, both schemes show monotonic increases. And as the utilization of aperiodic tasks is increased from 0.1 to 0.5 under a fixed utilization of periodic tasks U_P=1.2**,** the ratios gradually move up due to the increased energy consumption by aperiodic tasks.

For the single-policy dynamic scheduling, the lines are gathered around the similar value of ratios regardless of the augments in energy budget. This is due to the monotonic increase in the percentages of high-speed execution times and the monotonic decrease in percentage of low-speed execution time as shown in Figure 4-10 and Figure 4-11. As for the dual-policy dynamic scheduling, they show parallel line-ups with the increase of energy budget. This comes from that as more energy budget is given, the percentage of low-speed execution time is higher, whereas lower of high-speed execution time as shown in Figure 4-10 and Figure 4-11.

Table 4-3 An example of increased *L-mode* execution times and energy saving for periodic tasks when U_A= 0.3
(Mean of actual execution demands = 0.6, $E_C = E_{min} + 0.7\ E_{diff}$, and aperiodic tasks are run in all *H-mode* (1.3 Watts) with energy consumption 0.194)

	Single-Policy (SP)			Dual-Policy (DP)			Energy Saving	Static	Saving Ratio
U_P	H	L	E_P	H	L	E_P	ΔE_P	E_{Static}	$\Delta E_P / E_{Static}$
0.8	0.142	0.195	0.232	0.047	0.386	0.154	0.078	0.590	13.2%
0.9	0.163	0.214	0.263	0.050	0.439	0.171	0.092	0.633	14.5%
1.0	0.185	0.231	0.296	0.047	0.505	0.183	0.113	0.686	16.5%
1.1	0.222	0.216	0.341	0.096	0.467	0.237	0.104	0.762	13.6%
1.2	0.261	0.198	0.387	0.151	0.418	0.490	0.103	0.839	12.3%
1.3	0.300	0.181	0.434	0.206	0.367	0.356	0.078	0.917	8.5%
1.4	0.338	0.163	0.479	0.276	0.287	0.428	0.051	0.997	5.1%

Also, little convexes in both high- and low-execution times form Figure 4-10 and Figure 4-11 draw slight more energy consumption at the average of actual workload 0.6 at the lowest energy budgets in Figure 4-12 (b), (c), and (d). Note that the energy consumptions of the two schemes are gradually closer to each other along with the increase of aperiodic tasks' utilization mainly due to the increase of energy consumption in the dual-policy dynamic scheduling. The reason is explained this way. As utilization of aperiodic tasks gets higher, the energy budget allocation to periodic tasks is reduced by the increased allocation to aperiodic tasks. The reduced energy budget make the *S2* schedule closer to the energy consumption of *S1* schedule, whose voltage settings consume the lowest energy in the schedulable range, leading the dual-policy

dynamic scheduling to lower chance of the switched task execution from high-speed to low-speed. Conclusively, under a fixed workload demand of periodic tasks, the lower workload demand of aperiodic tasks or the higher energy budget is allocated to the system, the more energy saving it can get.

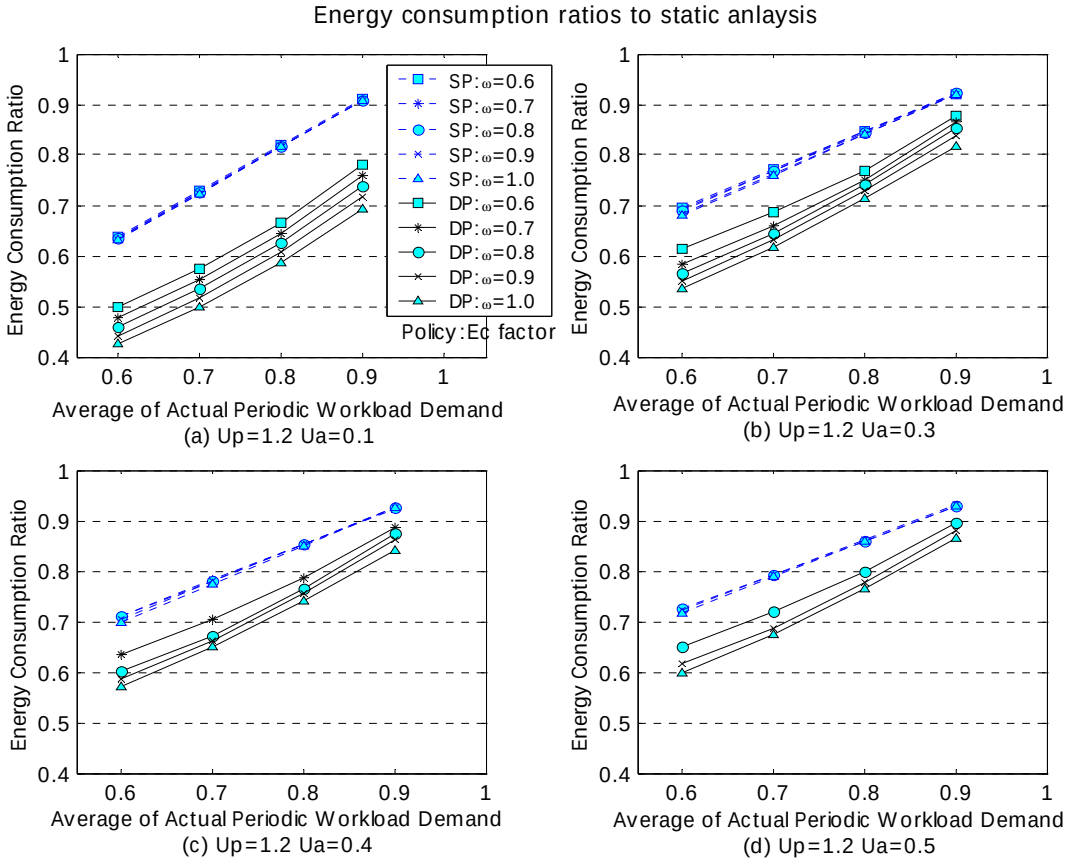


Figure 4-12 Energy consumption ratios of dual and single-policy dynamic scheduling to static analysis for the variation of the real execution time demands of periodic task when U_P=1.2
(Solid and dashed lines stand for dual-policy and single-policy, respectively, and so do the values in the legend for the constrained energy budget $E_C = E_{min} + \varpi E_{diff}$ in the range of $0.6 \le \varpi \le 1.0$)

So far, we have evaluated how the dual-policy dynamic scheduling gets more energy saving over the single-policy dynamic scheduling through the analyses of simulation results. Then, how fast responsiveness does it get?

4.6.2 Average Response Time

Over several real execution demands of periodic tasks and constraint energy budgets, we obtain the average response times of aperiodic tasks from the simulation having a fixed periodic workload demand of 1.2 and plot them in Figure 4-13. Regardless of increase in the energy budget, Figure 4-13 reveals a trend of increments in average response time of aperiodic tasks as actual workloads of both periodic and aperiodic tasks increase although there are some differences of the increasing intensity.

In Figure 4-13 (a), where the aperiodic execution demand is as low as 0.1, we notice that the increasing changes in both energy budget and the average of actual workload do not much affect the responsiveness of the single-policy dynamic scheduling such that the average response times remain in the small range of variation. However, at the available lowest energy budget for each simulation case, the dashed curves are getting closer to solid curves as the workload demand for aperiodic tasks is intensified in Figure 4-13 (b), (c), and (d). This is due to the delayed execution by increased deadline assignment on the TBS algorithm. The increasing workload demand of aperiodic tasks enables periodic tasks to get lower energy budget. The voltage mode assignment for periodic tasks under a lowered energy budget cause periodic tasks to take more processor's capacity by low-speed execution and then the utilization left over to aperiodic tasks lengthens the deadlines of aperiodic tasks.

As for the dual-policy dynamic scheduling in Figure 4-13 (a), roughly they show monotonic increases along with the variation in both the actual workload of periodic tasks and energy budget. First, as we increase the workload, the variance of execution time is reduced, making slack periods shorter. The decreased slack periods extend the time instant when the workload difference can meet the schedulability. Therefore, when the switching from *S2* to *S1* schedule is needed, the delayed switching time causes the scheduler to hold the arrived tasks longer and affects to response times. Furthermore, in spite of the same actual workload of periodic tasks the responsiveness is degraded, as the constrained energy gets higher. At any

energy budget, the more tasks have high-speed mode assignments for *S2* schedule, the more switched mode executions from *H-mode* to *L-mode* in *S1* schedule are available. Thus, the more workload differences are generated during intervals having only periodic tasks.

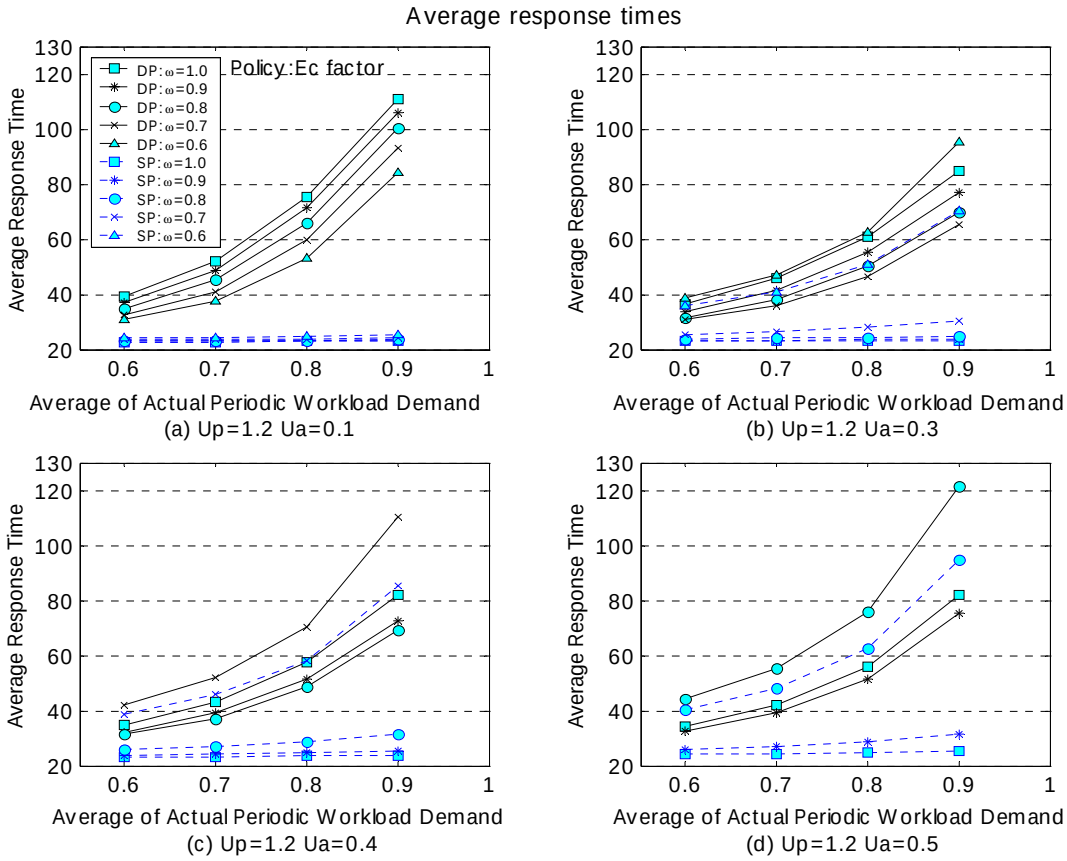


Figure 4-13 Average response times in regard to the actual workload demands of periodic tasks when U_P=1.2
(Solid and dashed lines stand for dual-policy and single-policy, respectively, and so do the values in the legend for the constrained energy budget $E_C = E_{min} + \varpi E_{diff}$ in the range of $0.6 \le \varpi \le 1.0$)

Then, from the end of intervals, in addition to the slack periods the more switched mode execution from *L-mode* in *S2* schedule to *H-mode* in *S12* schedule is demanded for the transitory schedule *S12* to reduce workload difference as soon as possible, if it has. But, if most tasks are assigned in *H-mode* in S2 schedule, the reduction of workload difference is prolonged and mostly carried by the slack periods. Thus, to get the best responsiveness under a confined energy budget,

some extent of mixed combination of voltage settings is needed for *S2* schedule to accelerate the reduction of workload difference.

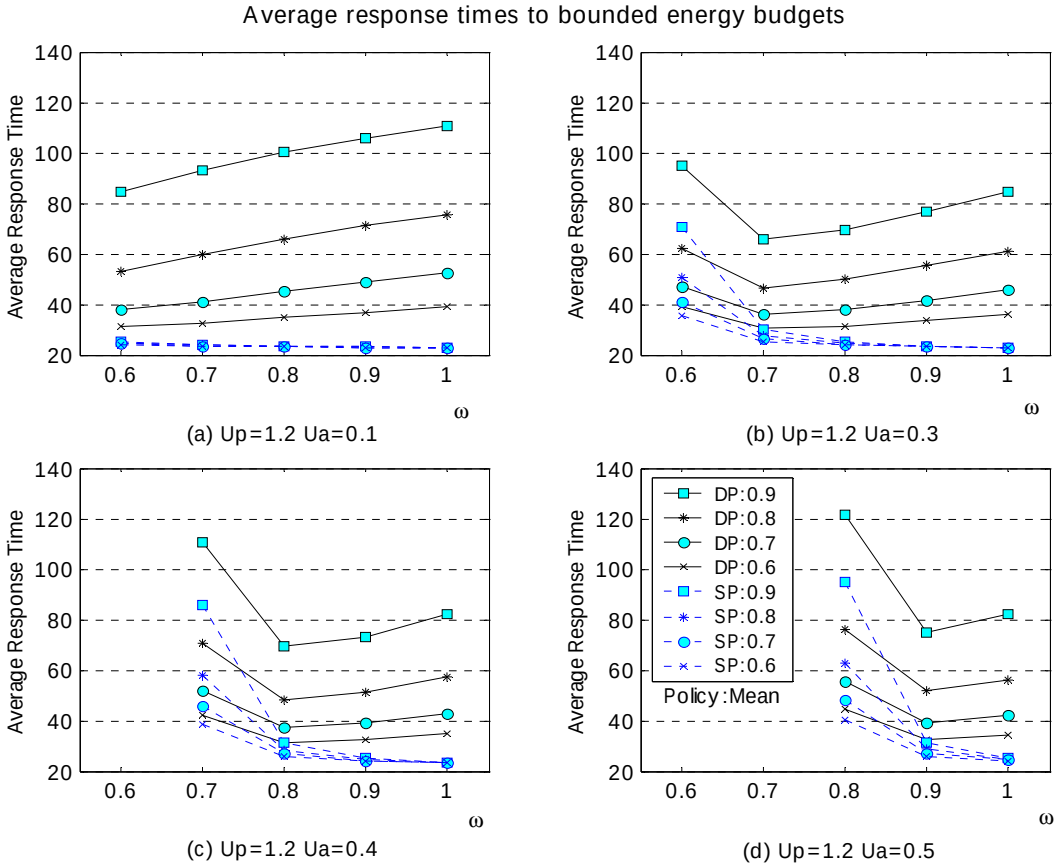


Figure 4-14 The responsiveness with respect to the bounded energy budgets when U_P=1.2 (Solid and dashed lines are for dual-policy and single-policy, respectively, and the values in the legend for the averages of actual workload for periodic tasks)

Unlike in Figure 4-13 (a), interesting results from Figure 4-13 (b), (c), and (d) are that the dual-policy scheduling reveals abrupt big decrements in the measure of average response times as the energy budget is increased. When the energy budgets are given as ω = 0.7, ω = 0.8, and ω = 0.9for the utilization of aperiodic tasks of 0.3, 0.4, and 0.5, respectively, the average response times show the minima over the change of the actual workload demand. In each case, to the augments of energy budget, they increase but lower than the average response times for the lowest energy budget. The results for another workload demand of periodic tasks also catch up with the trend that shows minima from a certain workload demand of aperiodic tasks. As the

workload demand of periodic tasks is raised, the convex down phenomenon appears at lower workload demand of aperiodic tasks.

To get better comparisons in a different view, the average response times are plotted in Figure 4-14 with the same conditions as in Figure 4-13, but with respect to the energy budgets. In Figure 4-14 (a), the average response times for both dynamic scheduling schemes show slight monotonic increase or maintain almost flat to the increase of energy budget. In Figure 4-14 (b), (c), and (d), there are rapid improvements in the responsiveness for both scheduling schemes from the lowest to the next increased level of energy budgets.

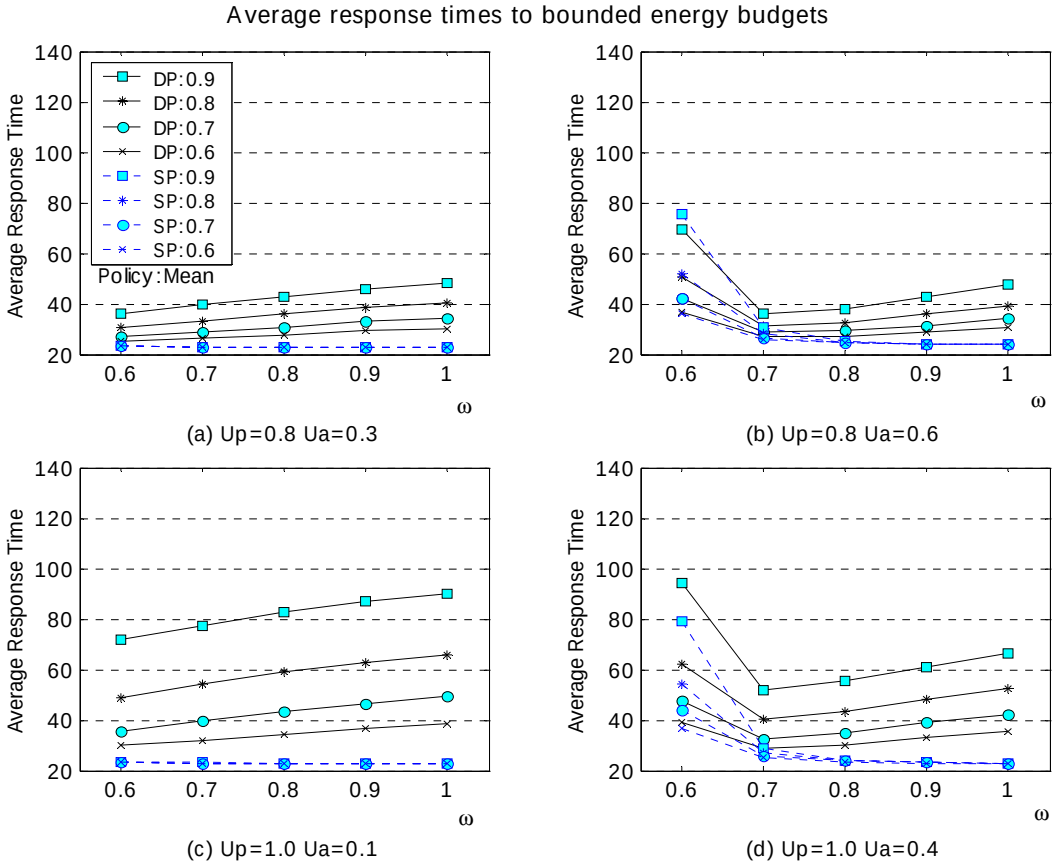


Figure 4-15 The responsiveness with respect to the bounded energy budgets when U_P=0.8 and 1.0
(Solid and dashed lines are for dual-policy and single-policy, respectively, and the values in the legend for the averages of actual workload for periodic tasks)

But, as we increase the energy budget more, the two schemes develop in opposite directions. The solid lines for the dual-policy scheduling show some extent of degradation in the

responsiveness and the minima at $\omega = 0.7$, $\omega = 0.8$, and $\omega = 0.9$ as the results shown in Figure 4-13 (b), (c), and (d), while the dashed lines for the single-policy dynamic scheduling continue to get improvements but weaker than the first increase of energy budget. As shown in Figure 4-15 and Figure 4-16, for the case of U_P= 0.8, 1.0, 1.2, and 1.4, the minima appear when U_A= 0.6, 0.4, 0.3, and 0.2, respectively.

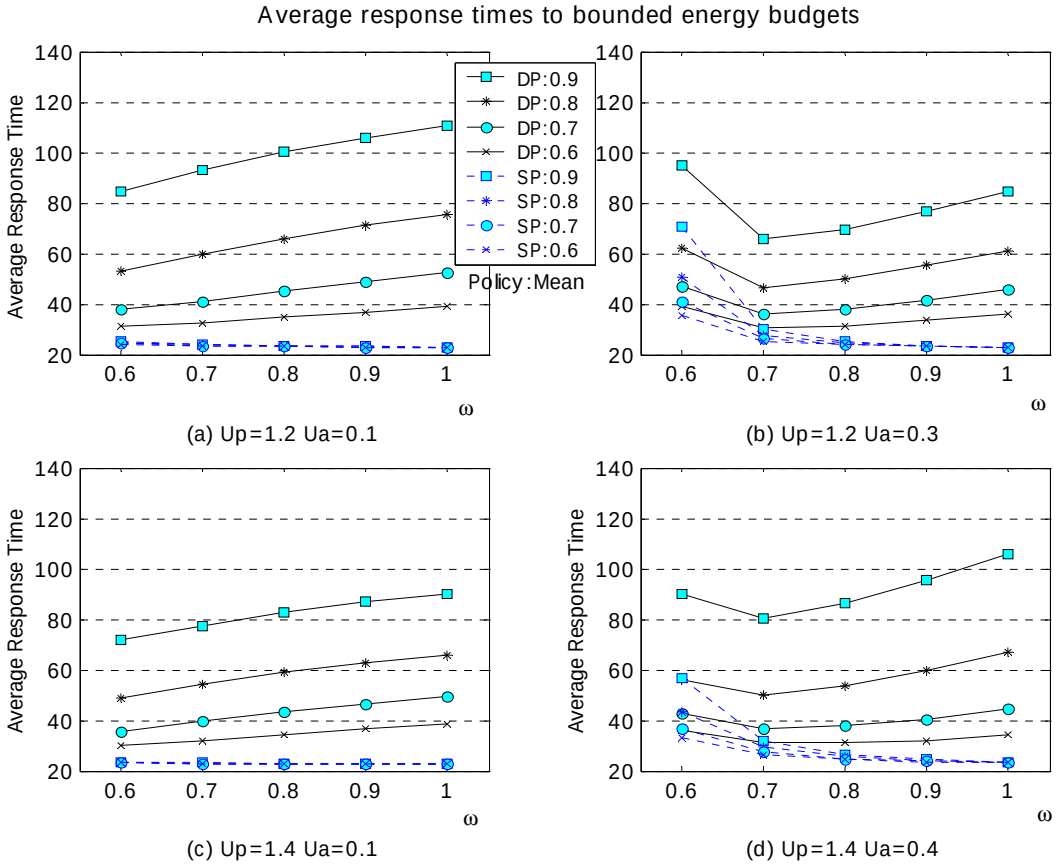


Figure 4-16 The responsiveness with respect to the bounded energy budgets when U_P=1.2 and 1.4
(Solid and dashed lines are for dual-policy and single-policy, respectively, and the values in the legend for the averages of actual workload for periodic tasks)

For further reasoning of the results in Figure 4-14 (b), (c), and (d), execution times in different mode from the mode assigned in worst-case are accumulated in our simulation and plotted in Figure 4-17. Since the simulation is done under a fixed execution demand of 1.2, the voltage settings for the *S1* schedule are independent on the increase in workload of aperiodic

tasks. The curves with dotted lines are for *T-LH*'s and represent the times executed in high-speed mode from low-speed mode assigned in *S2* schedule to switch scheduling policy from *S1* to *S2*. Also, *HtoL*'s are plotted in solid lines and represent the times executed in low-speed mode assigned in *S1* schedule from high-speed mode assigned in *S2* schedule for the intervals consist of periodic tasks.

T-LH's are increased with respect to the increase of the actual workload of periodic tasks. Clearly, the chance for switched mode execution from high-speed to low-speed is increased due to more workload of periodic tasks. However, Figure 4-17 (b), (c), and (d) show the highest amounts at $\omega = 0.7$, $\omega = 0.8$, and $\omega = 0.9$ unlike Figure 4-17 (a) showing monotonic decreases as we increase the energy budget. This is the reason why the responsiveness gets the minima at those specific constraint energy budgets as shown in Figure 4-14 (b), (c), and (d). And these are due to two facts. Since the probability of *H-mode* assignment in *S2* schedule is low when the energy budget is minimum, the workload difference to be reduced by switched mode execution is relatively lower. On the other hand, when the energy budget is higher, most tasks are assigned in high-speed mode in *S2* schedule. Especially, due to all high-speed mode assignment when $E_C = (E_{min}+E_{diff})$, none of tasks can be executed from low-speed to high-speed mode.

HtoL's show monotonic increase to the augments in energy budget regardless of the actual workload of periodic tasks in Figure 4-17 (a). As the constrained energy budget is higher, more assignment in the high-speed mode is available for *S2* schedule such that there is more probability to be executed in low-speed. When the average of the actual workload is 0.6, the switched executions from high-speed to low-speed for energy saving is the lowest and others show similar results. As we increase the execution demand of aperiodic tasks as shown in Figure 4-17 (b), (c), and (d), the *HtoL* is getting lower. This is due to decreased high-speed mode assignment for *S2* schedule from the lower energy allocation to periodic tasks with the increase in energy allocation for aperiodic tasks. And unlike Figure 4-17 (a), regardless of the actual workload of periodic tasks, *HtoL*'s show upward-convexities over the change of energy budget,

showing the maxima at specific energy budgets. This is due to two facts. The first is that the probability of *H-mode* assignment in *S2* schedule is low when the energy budget is minimum. On the other hand, when the energy budget is high, increased *H-mode* assignment in *S2* schedule cause transition intervals to be increased.

As the actual workload of periodic tasks is raised in Figure 4-17 (b), (c), and (d), the *HtoL*'s are reduced, forcing the energy consumption difference between two scheduling schemes to be closer each other as shown in Figure 4-12 (b), (c), and (d).

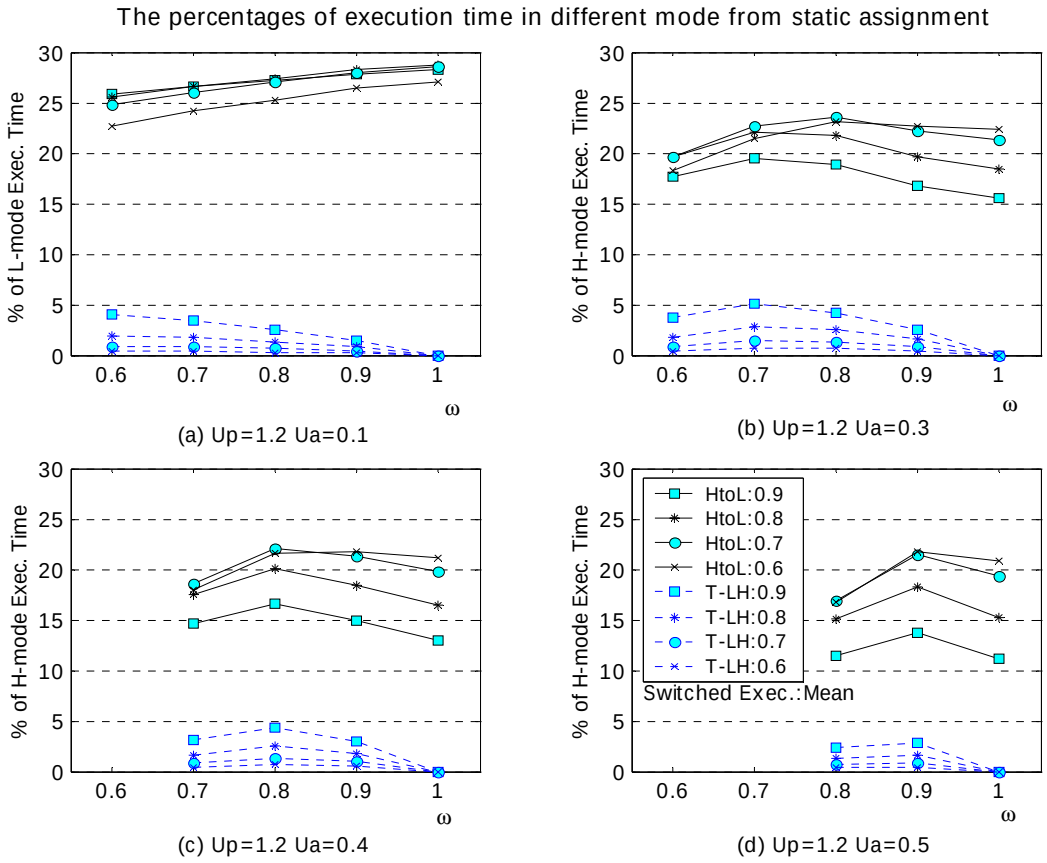


Figure 4-17 The percentages of execution time in changed mode execution from assigned mode to the total execution times when U_P=1.2
(Solid and dashed lines are for '*HtoL*' and for '*T-LH*' by *S12* schedule, respectively, and the values in the legend for the averages of actual workload for periodic tasks)

Speaking of responsiveness, the single-policy dynamic scheduling outperforms the dual-policy dynamic scheduling for any the energy budget allocation. But under the condition of lower

energy budgets and even lower actual execution workloads of periodic tasks, the dual-policy dynamic scheduling shows reasonable results comparing to the single-policy dynamic scheduling.

4.7 Conclusion

In this chapter, using EDF scheduling we present a dynamic scheduling algorithm to carry out voltage clock scaling for workloads consisting of mixed hard and soft real-time tasks. The objective of the scheduling is to minimize energy consumption and to improve the response time of aperiodic tasks in energy-constrained real-time systems, while all deadlines of periodic tasks are met. Based on two specific intervals classified by the events occurring in runtime scheduling for both mixed hard and soft real-time tasks, we propose dual-policy dynamic scheduling that switches two elementary scheduling policies, exploiting reduction of energy consumption from the increased low-speed execution time. However, the backlogged workload (workload difference) may be generated from executing the same tasks in different speeds by two policies and may jeopardize the schedulability of the entire system. For the guarantee of schedulability by EDF scheduling, the backlogged workload at switching point needs to be reduced to the level that a target scheduling policy can afford to. In addition to the slack periods that play big roles for the reduction of the workload difference, we introduce transitory schedules that execute all periodic tasks in high-speed mode to advance the time of switching to targeting elementary schedule. Until the schedulability of the target schedule is guaranteed, aperiodic tasks are scheduled by infinite deadline based on modified total bandwidth algorithm.

Our simulation study follows the result of a constrained energy allocation model, where the VCS-EDF scheduling gets the fastest responsiveness when the extra energy budget is allocated to aperiodic tasks at their maximum energy demand such that all of them can be run in *high-voltage* and *high-speed mode*. As for the energy saving performance, our results show that the dual-policy dynamic scheduling saves more energy as actual workload of both periodic and aperiodic tasks is lower and available energy budget is higher for a fixed periodic workload. Under the fixed aperiodic workload of 0.3, the energy consumption decreases maximum 50.4 %

over the given total energy budget when the periodic workload demand is 1.0. This is doubled energy saving of the SPDS. In the comparisons of responsiveness, the dual-policy dynamic scheduling shows longer response time than the single-policy dynamic scheduling, which sticks to the worst-case schedule thoroughly. But, as available energy budget or periodic workload demand is getting lower, the response time of the dual-policy dynamic scheduling gets closer to the single-policy dynamic scheduling.

CHAPTER 5
SCALING OF ENERGY CONSUMPTION AND RESPONSIVENESS

5.1 Introduction

As for responsiveness and energy consumption of a system, they are fully dependent on the aimed performance of application implemented in the system. Some need fast response time for background tasks regardless of energy consumption of the system and so does some low total energy consumption regardless of their responsiveness. In general, an efficient optimization mechanism is needed to guarantee better performance, making the best of the available energy budget. With regard to best sharing of the bounded energy budget in a battery-driven real-time embedded system, an energy budget allocation model is proposed in Chapter 3.

In sharing a constrained energy budget among mixed hard and soft real-time tasks based on total bandwidth server algorithm and two-mode VCS-EDF scheduling, reduced response times are brought by assigning the energy budget to aperiodic tasks to execute them in high-voltage and high-speed mode. Also, the extra energy budget allocated to periodic tasks can result in a lowering of the CPU utilization reserved for periodic tasks. This, in turn, leaves more available CPU utilization for aperiodic tasks and causes shorter deadlines as defined in the total bandwidth server-scheduling scheme. The energy budget is allocated to get better performance for aperiodic task without missing any deadline of periodic tasks.

Under the energy budget assigned by the energy allocation model, we propose an energy efficient dynamic scheduling called dual-policy dynamic scheduling (DPDS) in Chapter 4, in which a dynamic scheduler schedules mixed tasks by switching scheduling policies between two schedules, exploiting the characteristics in the pattern of event occurrence by periodic and aperiodic tasks in busy cycles.

As a set of periodic tasks shares the processor's capacity with aperiodic tasks by total bandwidth server algorithm, the portion of periodic tasks is less than when they are scheduled with the whole computational capacity. To guarantee the schedulability of a system, the more execution times should be assigned in *H-mode* in VCS-EDF scheduling by sharing the total energy with aperiodic tasks. And this requires more energy consumption. Combined with the constrained-energy allocation, two sets of voltage settings for periodic tasks are determined for elementary schedules called *S1* and *S2*, the one demanding the lowest and the other the worst-case energy consumption under the condition of feasible scheduling, in DPDS. Since the system schedules both mixed hard and soft real-time tasks, basically *S2* schedule should be applied throughout the scheduling for both periodic and aperiodic tasks. But, for the intervals having only periodic tasks' events, the tasks are scheduled by *S1* schedule instead of *S2* schedule since there is no aperiodic task. Switching scheduling policies occurs when the scheduler confronts the specific occurrence of aperiodic tasks' events, the first arrivals after periodic intervals or complete finishes of all arrived aperiodic tasks in a busy cycle. Due to the switched scheduling policies, DPDS outperforms the single-policy dynamic scheduling (SPDS), which applies only *S2* schedule, in reduction of power consumption. And DPDS shows reasonable responsiveness when the actual execution time demands change with a high variance.

The reduction of energy consumption is mainly coming from switched execution in low-speed from high-speed mode assignment in worst-case schedule *S2* for the intervals having only periodic tasks. Compared with the SPDS, slightly increases in response time are originated from postponed execution of aperiodic tasks while handed-over workload from *S1* to *S2* schedule is reduced to a schedulable level before the switching to *S2* schedule is performed. The backlogged workload is originated from increase in low-speed execution instead of high-speed mode assignment, by choosing *S1* schedule instead of *S2* schedule.

From the point of switching policies in the DPDS, the SPDS can be considered as a special case of DPDS, in which the scheduler never changes the scheduling policy to *S1* schedule.

Compared to the DPDS, sticking to the worst-case scheduling leads to more energy consumption but faster responsiveness. Therefore, taking into account that aperiodic tasks are already assigned in high-voltage and high-speed mode for the best performance of responsiveness, if *S1* schedule is decided to consume more energy closer to *S2* schedule, better response times than DPDS are expected from reduced amount of low-speed execution switched from high-speed assignment.

On the other hand, due to the relationship in utilization and energy consumption of a set of periodic tasks, simple change in the utilization of the elementary *S1* schedule can control the combination of voltage settings in the *S1* schedule. Eventually, the scaling of *S1* schedule's utilization under available energy budget for *S2* schedule is directly linked to the calibration of energy consumption of the DPDS.

In this chapter, we propose fine-tuning control mechanism over both energy saving and responsiveness of DPDS, which jointly schedules hard periodic and aperiodic tasks under a bounded energy budget. By the energy sharing in total bandwidth server and the interplay between utilization and energy consumption for a set of periodic tasks, both energy consumption and responsiveness of DPDS can be easily adjusted to the aimed performance of a system. The tunable ranges are determined by the schedulability under energy budget allocation.

The rest of the chapter is structured as follows. In section 5-2, we discuss the feasible range of *S1* schedule from the relationship between energy consumption and utilization for a set of periodic using a task-based energy profiling and a proper allocation of a bounded energy budget. Then, we propose the adjusting mechanism in DPDS by introducing a parameter and explain it in detail under a constraint energy budget in section 5-3. The effectiveness of the proposed tuning mechanism algorithm is evaluated through simulations in section 5-4. In section 5-5, a short conclusion is given.

5.2 Feasible Ranges of Utilization and Energy Consumption for Periodic Tasks

As explained in chapter 3, given m_i, the voltage setting either *L-mode* or *H-mode*, and α_i (m_i), the speed of task τ_i at mode m_i, for all of periodic task τ_i, the energy demand for periodic task of E_p is defined as

$$E_P(m_i) = \sum \frac{1}{\alpha_i(m_i)} \frac{\overline{C_i}}{T_i} p(m_i) \tag{5-1}$$

where $p(m_i)$ is the power consumption at mode m_i, $\overline{C_i}$ the average execution time of task τ_i In addition, the worst-case utilization is given by

$$U_P(m_i) = \sum \frac{1}{\alpha_i(m_i)} \frac{C_i}{T_i} \tag{5-2}$$

There is a trade-off relationship between utilization and energy consumption for a set of periodic tasks. As the utilization is increased, the energy consumption shows decrement because the total computation time in *L-mode* is increased. In the other way, the worst-case utilization is decreased along the increment in energy consumption by the affordability to assign more execution time in H-mode under given energy consumption. Due to the tradeoff relationship, there are certain ranges of utilization and energy consumption by a set of periodic tasks, meeting the condition of schedulability.

For the maxima and minima, *max* E_P and *min* E_P are denoted as the maximum and minimum for the energy consumption and *max* U_P and *min* U_P for the utilization, respectively. Regarding the feasibility of the energy constraint and the worst-case utilization, constraint energy E_C must be greater than *min* E_P and *min* U_P should be no greater than 1. By its definition, if *min* U_P is greater than unity with all *H-mode* executions, it is impossible to find voltage settings to ensure that all tasks meet their deadlines. Guaranteeing its feasibility, the minimum utilization *min* U_P is $U_P(H) = \sum \frac{1}{\alpha_H} \frac{C_i}{T_i} \le 1$, and then the maximum energy consumption *max* E_P

becomes $E_P(H) = \sum \frac{1}{\alpha_H} \frac{\overline{C_i}}{T_i} p_H$. If *max* U_P is less than 1, the tasks are schedulable with all *L-mode* assignments and energy consumption can never be less than *min* E_P. In the case, *max* U_P becomes $U_P(L) = \sum \frac{1}{\alpha_L} \frac{C_i}{T_i}$ and *min* E_P does $E_P(L) = \sum \frac{1}{\alpha_L} \frac{\overline{C_i}}{T_i} p_L$.

Thus, the feasible ranges of utilization (U_P) and energy consumption (E_P) for a set of periodic tasks shown in Chapter 3 and [83] are shown in Figure 5-1 and defined as follows:

- *max* U_P and *min* E_P

  If $U_P(L) \le 1$, $maxU_P = U_P(L) = \sum \frac{1}{\alpha_L} \frac{C_i}{T_i}$ and $minE_P = E_P(L) = \sum \frac{1}{\alpha_L} \frac{\overline{C_i}}{T_i} p_L$.

  Otherwise, *max* $U_P = U_P(m_i) = 1$ and $minE_P = E_P(m_i) = \sum \frac{1}{\alpha(m_i)} \frac{\overline{C_i}}{T_i} p(m_i)$.

- *min* U_P and *max* E_P

  If $U_P(H) \le 1$, $minU_P = U_P(H) = \sum \frac{1}{\alpha_H} \frac{C_i}{T_i}$ and $maxE_P = E_P(H) = \sum \frac{1}{\alpha_H} \frac{\overline{C_i}}{T_i} p_H$.

  Otherwise, it is implausible.

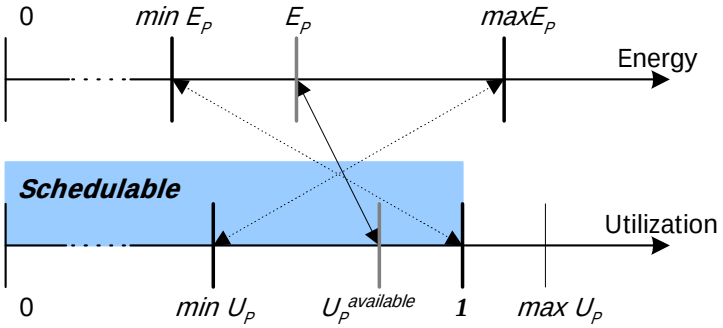


Figure 5-1 The relationship between power consumption and utilization for a set of periodic tasks

If energy budget for periodic tasks is assigned to E_P in the range from *min* E_P to *max* E_P, $U_P^{available}$ is the available utilization corresponding to the allocated energy budget E_P. And, by searching a set of voltage settings meeting the given energy budget and schedulability, energy

demand and utilization for periodic tasks are determined as $E_P(m_i) \leq E_P$ and $U_P(m_i) \geq U_P^{available}$, respectively.

5.3 Scaling Energy Saving and Responsiveness in Dual-Policy Dynamic Scheduling

Under a constraint energy budget, how can a real-time scheduler tune its performance of responsiveness and energy consumption level to the requirement of a real-time application? DPDS has the feature of switching scheduling policies between *S1* and *S2* schedules instead of applying *S2* schedule thoroughly. *S1* schedule is a set of voltage settings of periodic tasks consuming minimum energy by maximizing their utilization as much as possible under schedulability. And it is chosen for the intervals having only periodic tasks in the DPDS, giving noticeable energy saving. If the SPDS is regarded as a special case of DPDS, in which *S2* schedule is chosen instead of *S1* schedule for the intervals of only periodic tasks, there must be ranges of utilization and energy consumption determined by *S1* and *S2* schedules accordingly. When either energy consumption or utilization of *S1* schedule is calibrated within the feasible range, DPDS can control its responsiveness and energy consumption.

For the two elementary schedules, *S1* and *S2*, in DPDS, let's define energy consumption and utilization as $E_P(S1)$, $E_P(S2)$, $U_P(S1)$, and $U_P(S2)$, respectively. Then, the energy consumption for *S1* schedule $E_P(S1)$ is $min\ E_P$ and its utilization $U_P(S1)$ becomes the maximum, $max\ U_P$, which is near unity. In addition, DPDS has the constraints of energy and schedulability, $E_P(S2) + E_A \leq E_C$ and $U_P(S2) + U_A(S2) \leq 1$, respectively. Considering the result of DPDS, if another set of voltage settings having much lower utilization than $max\ U_P$ is used for *S1* schedule, higher energy consumption than $min\ E_P$ is required and faster response time is expected due to the increased utilization for aperiodic tasks. In this way, the adjustment in the execution demand of *S1* schedule affects the execution times in low-speed in *S1* schedule. The total switched execution time from high to low-speed mode is also reduced in the DPDS, leading to decrease in energy saving and average response time.

Figure 5-2 shows the big picture of relationship in energy allocation, utilization, and adjustment of energy saving in the DPDS.

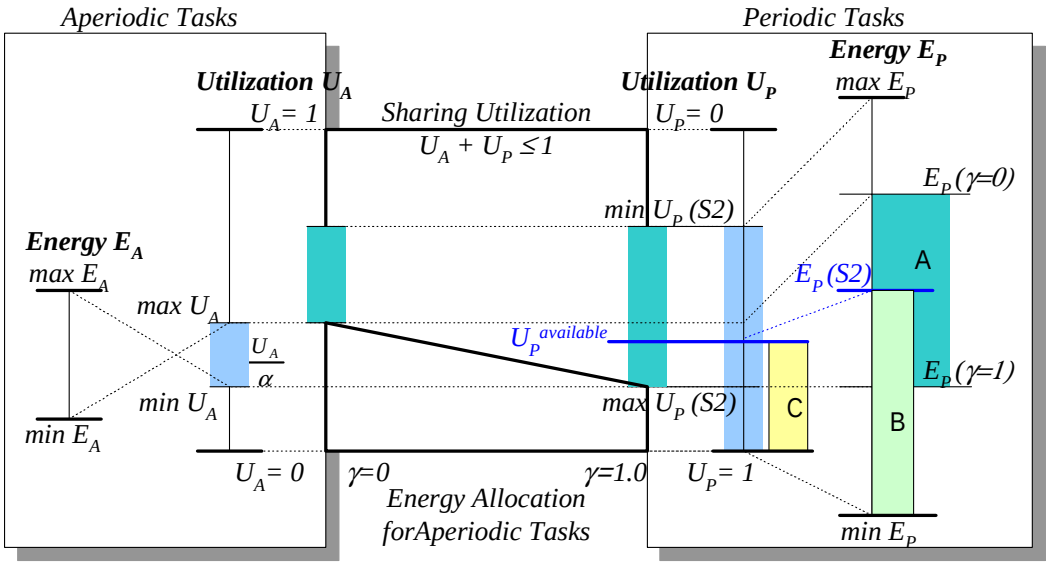


Figure 5-2 Sharing of energy consumption and utilization in Dual-Policy Dynamic Scheduling

First, the running modes for *S2* schedule are determined in a constrained energy allocation model, which has the energy budget constraint of $E_P + E_A \leq E_C$ and the utilization constraint of $U_P + U_A \leq 1$. Then, the energy allocation for aperiodic tasks is varied in the range of $min\ E_A \leq E_A \leq max\ E_A$ with the energy allocation factor γ and so is for periodic tasks $E_P(\gamma=0) \leq E_P(S2) \leq E_P(\gamma=1)$ as shown as range *A* within the their available range of $min\ E_P \leq E_P \leq max\ E_P$. The utilizations for aperiodic and periodic tasks are varied in the respective ranges of $min\ U_A \leq U_A \leq max\ U_A$ and $min\ U_P(S2) \leq U_P(S2) \leq max\ U_P(S2)$. Allocating a constrained energy budget among mixed tasks, the constrained energy allocation model results in a reduced response time when energy budget is assigned to aperiodic tasks so that their execution can be done in high-voltage and high-speed mode. Then, the energy budget for aperiodic tasks is allocated as $E_A = max\ E_A$ and the energy budget left from the total energy budget is given to periodic tasks as $E_P = E_C - max\ E_A$. Given allocated energy budget E_P, the voltage settings for periodic tasks are found, consuming $E_P(S2)$ that is less than or equal to E_P, and having the worst-case workload demand $U_P^{available}$ for the

determined voltage settings. Also, it is required that ($1 - U_P^{available}$) is greater than the workload demand, U_A, of aperiodic tasks for the schedulability by total bandwidth server algorithm. Therefore, either E_P *(S2)* in the region *B* or $U_P^{available}$ in the region *C* indicates that the given tasks are not schedulable, representing E_P *(S2)* is too low to schedule tasks.

5.3.1 Scaling Factor δ

To calibrate energy consumption of DPDS by tweaking the workload demand of *S1* schedule, U_P *(S1)*, we define a new parameter δ called *energy scaling factor* and ranged over $0 \leq \delta \leq 1$. Given a set of periodic tasks, parameter δ indicates the utilization range of $min\ U_P \leq U_P\ (S1) \leq 1$ in schedulable regions of utilization and energy consumption. Since *min* U_P is obtained when all periodic tasks are run in *H-mode*, $\delta = 0$ tells that $U_P\ (S1) = U_P/\alpha_H$ and *L-mode* is naught in the voltage settings. Figure 5-3, under energy allocation E_P *(S2)*, δ becomes 1 when $U_P\ (S1)=1$ and so does zero when $U_P\ (S1)= min\ U_P$. As δ is getting closer to 1, the energy saving increases.

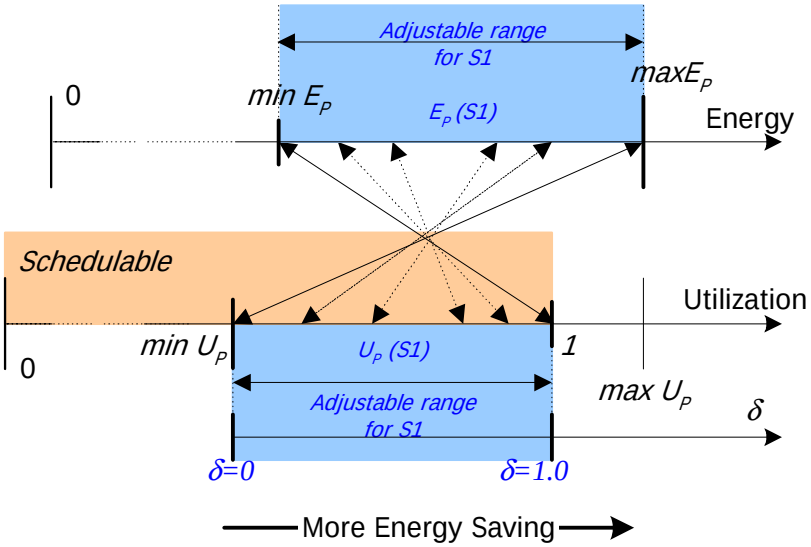


Figure 5-3 The range of energy scaling factor δ

5.3.2 *S1* Schedule with Scaling Factor

According to the definition of δ, a set of periodic tasks should be scheduled in such a way that the utilization is less than or equals to ($min\ U_P + \delta$) and unity as well. Given scaling factor δ, there exist a set of voltage settings *S1*: $\{m_i\}$ so that the worst-case utilization

$$U_P(m_i) = \sum \frac{1}{\alpha(m_i)} \frac{C_i}{T_i} \le \left(minU_P + \delta\right) \le 1 \text{ is maximized.} \tag{5-3}$$

Then, by the set of voltage settings determined in equation (5-3), the energy consumption is also determined as E_P (m_i) by the equation

$$E_P(m_i) = \sum \frac{1}{\alpha(m_i)} \frac{\overline{C_i}}{T_i} p(m_i), \tag{5-4}$$

where E_C must be greater than E_P (m_i), regarding the feasibility of energy constraint.

5.3.3 Scaling under a Constrained Energy Budget

Under a constraint energy budget E_C, the adjustable ranges of utilization and energy consumption by *S1* schedule are shown in Figure 5-4. If energy budget for periodic tasks is assigned to E_P *(S2)* in the range from *min* E_P to *max* E_P, the available utilization $U_P^{available}$ corresponding to the allocated energy budget E_P *(S2)*. And, by searching a set of feasible voltage settings in VCS-EDF scheduling that makes the best of the energy budget E_P *(S2)*, the demand of energy and utilization for periodic tasks are determined as E_P (m_i) $\le E_P$ *(S2)* and U_P (m_i) $\ge$ $U_P^{available}$, respectively. Due to the relationship between utilization and energy consumption for a set of periodic tasks, tuning the utilization of periodic tasks can be interchangeable with adjusting their energy consumption. To find voltage settings that can reduce energy consumption as much as possible, consuming less energy than the energy allocation E_P *(S2)*, corresponds to finding the ones that occupy the processor's capacity as much as getting closer to 1 from $U_P^{available}$. Thus, the adjustment of δ in the range of $\delta_{min} \le \delta < 1$ corresponds to selection any utilization in region *A* and so does energy consumption in the region *B*. On the other hand, if system application required

faster responsiveness, by assigning the utilization closer to $U_P^{available}$ the voltage settings with increased execution times of *H-mode* in *S1* schedule can provide for its aimed performance.

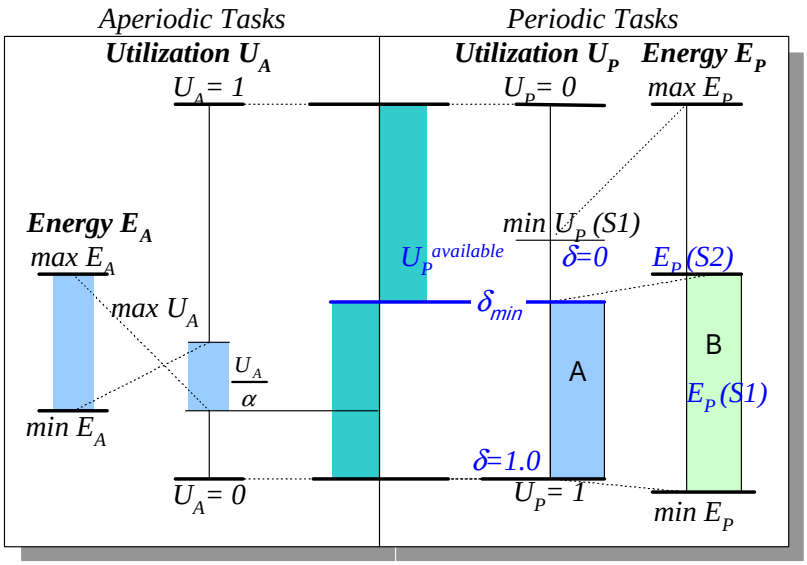


Figure 5-4 Feasible scaling ranges of energy consumption and responsiveness under a constraint energy budget

5.4 Performance Evaluation

Here, we investigate how the variations in *L-mode* assignment in *S1* schedule impact on responsiveness and energy saving in the Dual-Policy Dynamic Scheduling. Single-Policy Dynamic Scheduling is used for the comparisons whenever it is needed. We also define parameter ω in the range of $0 \le \omega \le 1$ to represent the constrained energy budget in a system such that E_C. is allowed to get the energy budget as $(E_{min} + \omega E_{diff})$, where $0.6 \le \omega \le 1$. Then, the constrained energy is set at each of several energy levels from $E_C = (E_{min}+0.6E_{diff})$ to $E_C = (E_{min}+E_{diff})$ by the increment of $0.1E_{diff}$. From 0 to 1, δ is increased by 0.1 and the workload demand of *S1* schedule is tuned within available range by the given periodic tasks. For instance, when the workload demand of periodic tasks is forced as 1.2, the workload demand of *S1* schedule covers from 0.6 to 1.0 by an increase of 0.04 with respect to δ.

Dual-policy dynamic scheduling saves energy consumption by the increased *L-mode* execution time, which comes from the switched speed mode execution from assigned in *S2*

schedule to *S1* schedule for the interval having only periodic tasks. The more *L-mode* assignment *S1* schedule has, the more energy the dual-policy dynamic scheduling saves. Over several energy budgets and sets of voltage settings controlled by ω and δ, respectively, we keep tracks of the executed times in *H*- and *L-mode* of periodic and aperiodic tasks including idle times and present them as percentages to the total execution time as stacks of bar in Figure 5-5.

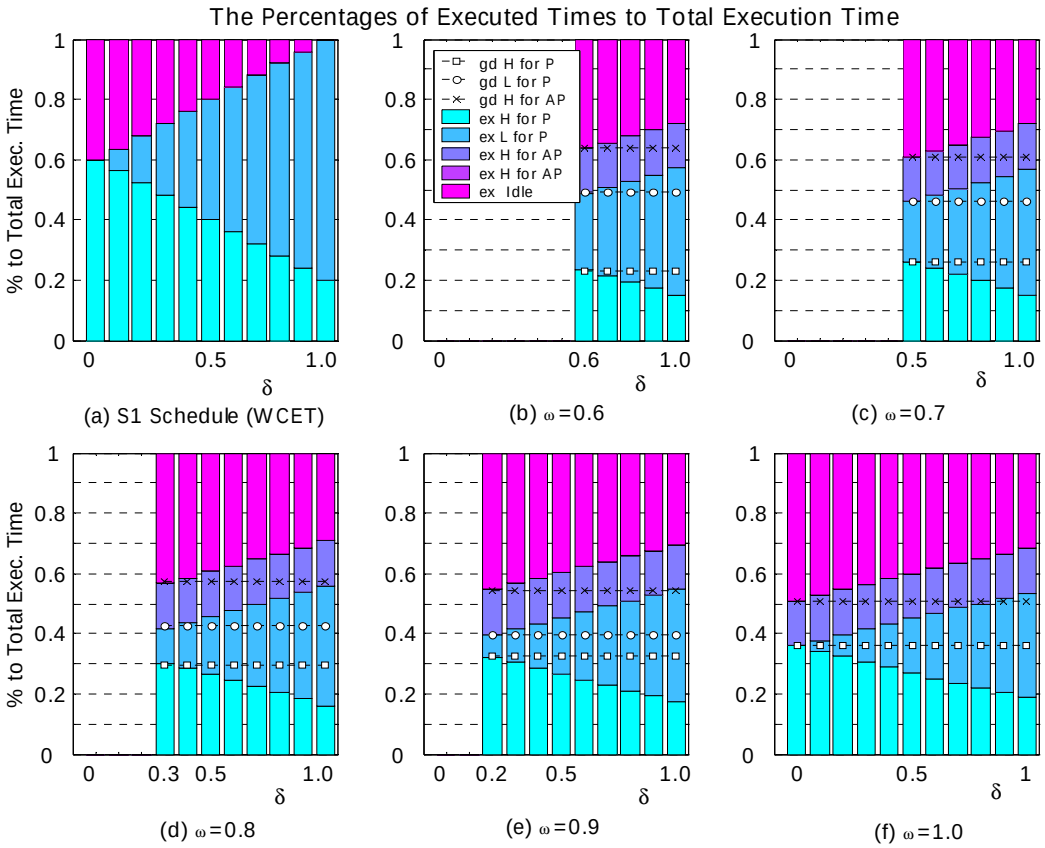


Figure 5-5 The percentages of executed times to the total execution times over a various energy budget in cases of U_P=1.2 for a fixed U_A=0.3
(Average of real workload for periodic tasks= 0.6)

The length of any bar represents how much it occupies processor's computing capacity. '*ex H for P*', '*ex L for P*', '*ex H for AP*', and '*ex L for AP*' are representing how much *H-mode* (*H:*high-speed) and *L-mode* (*L:*low-speed) executions are carried out for periodic (*P*) and aperiodic (*AP*) tasks. '*ex Idle*' represents the ratio of processor's idling times without computation to the total execution time. Also, dashed lines denoted as '*gd H for P*', '*gd L for P*', and '*gd H for*

AP' represent the generated workloads in *H-* or *L-mode* and show flat lines regardless of the increase in δ. Since an extra energy budget can assign more tasks in *H-mode*, '*gd H for P*' and *idle time* show increase from (b) to (f), whereas decreases '*gd L for P*'.

At δ_{min} for all the constraints of energy budgets, '*ex L for P*' and '*ex H for P*' almost amount to '*gd L for P*' and '*gd H for P*'. But they are increased and decreased along with the increment of δ, respectively. The difference between '*ex H for P*' and '*gd H for P*' exactly contributes to the increase of '*ex L for P*' because some of the assigned amount in high-speed mode has been executed in low-speed mode. To measure the impact of switched executions on the energy consumption, we calculate the energy consumption by the increased low-speed execution and the decreased high-speed execution times and also plot them in Figure 5-6.

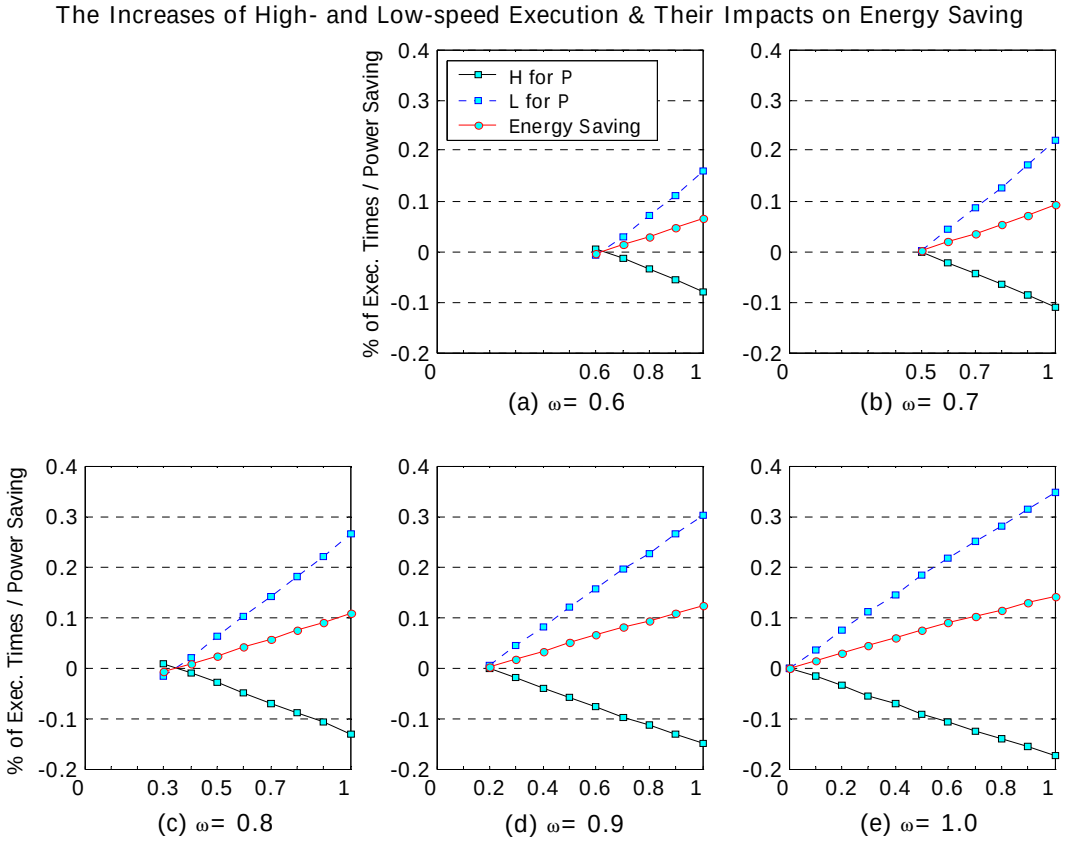


Figure 5-6 The increases in high- and low-speed mode execution times and energy saving by them when U_P=1.2 for a fixed U_A=0.3
(Average of actual periodic workload = 0.6)

Over the effective range of δ, i.e. $\delta_{min} \leq \delta \leq 1$, the variation of execution times influence to energy consumption as energy saving. Due to the energy consumption difference by different speed modes in VCS, the benefit of switched execution goes to energy saving. As for the change from the generated workload in each speed mode, the increase of high-speed mode execution turns out to be negative, whereas the increase of the low-speed mode execution does to be positive. And along the increase in the constraints energy budgets, the percentage in *H-mode* execution time diminishes and the one in *L-mode* is increased. This represents that the possibility of switched mode execution gets higher because the percentage of '*gd H for P*' is increased while the percentage of '*gd L for P*' in *S1* schedule remains at the same level. For an instance, at δ_{min}=1 in *S1* schedule, '*gd L for P*' is 79.9%. Since '*gd H for P*'s in *S2* schedule are 22.8%, to 26.1%, 29.3%, 32.5%, and 36.0%, respectively, the probability of switched execution from high- to low-speed mode increase to 0.285, 0.0.327, 0.367, 0.407, and 0.451.

The influences of the switching between dual-execution modes to energy consumption are measured in differences of energy saving from the SPDS when workload demands for periodic and aperiodic tasks are 1.2 and 0.3, respectively and plotted in Figure 5-7 over the change in average of actual workload of periodic tasks from 0.6 to 0.9. Figure 5-7 (a) represents all the cases of energy budgets in one graph. Throughout the increase in average of actual workload of periodic tasks, the same phenomena as shown in Figure 5-6 and Figure 5-7 (a) are shown in Figure 5-7 (b), (c), and (d) except the strength of energy saving is gradually decreased, accordingly. This is indicative of the sensitivity for different *S1* schedules. Compared to Figure 5-7 (d), Figure 5-7 (a) shows faster change in energy saving when average of actual workload of periodic tasks is 0.6. This is from that the shorter busy cycle because the higher slack times are available for lower average of actual workload of periodic tasks such that the switched mode execution can be increased.

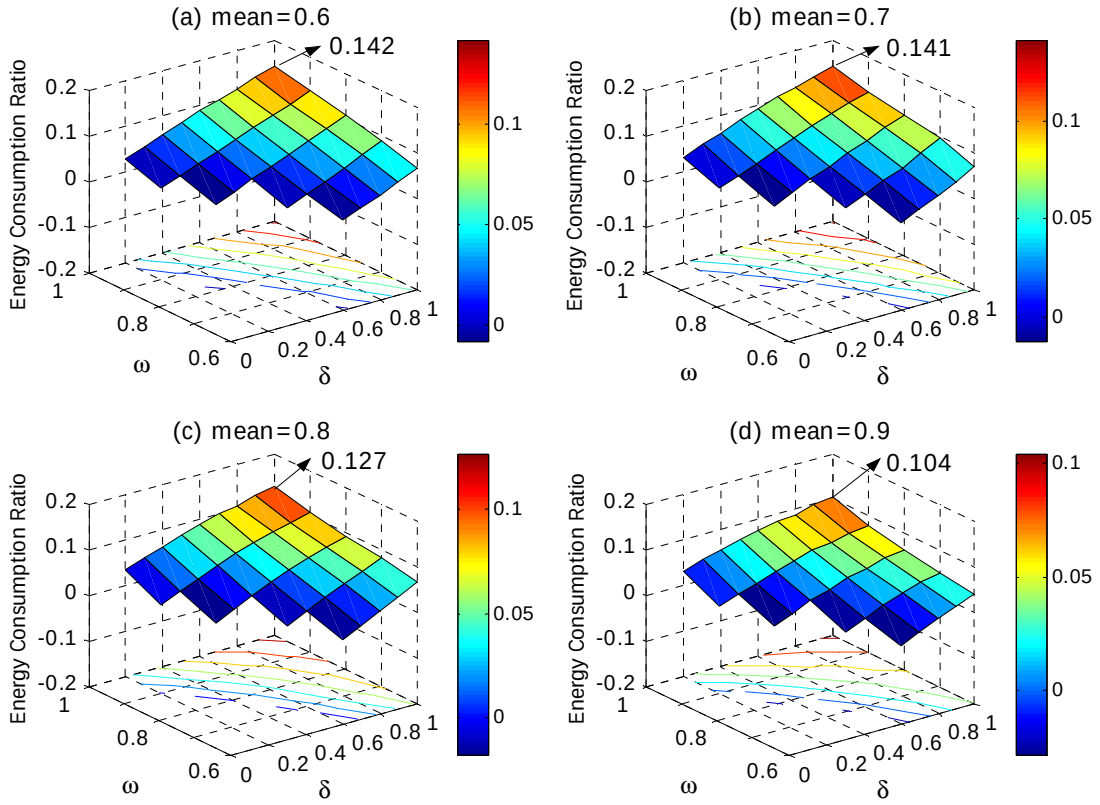


Figure 5-7 Energy saving with respect to Single-Policy Dynamic Scheduling with the variations of δ (*S1* schedule) and constraint energy budgets when U_P=1.2 and U_A=0.3

To see the impact on responsiveness by the increases in utilization of *S1* schedule we measure average response times under the same condition as Figure 5-7, when $E_C = E_{max}$ *(i.e.* ω*=1),*U_P = 1.2, and U_A = 0.3, and illustrate in Figure 5-8.

One interesting result in Figure 5-8 is that the graphs show a trend that average response times become minima when the energy budget is given as ω=0.7 regardless of average of actual workload of periodic tasks. There are two main factors for the trend. The one is that for the transitory intervals on the way from schedule *S1* to *S2*, the low-speed assigned tasks in *S2* schedule are getting lower as the energy budget is increased. In the DPDS, in addition to slack times, more low-speed assigned tasks in *S2* schedule is needed to get fast reduction of the workload difference caused by the switched execution from high-speed to low-speed execution.

Especially, when $E_C = E_{min}$, *S2* schedule has no assignment for low-speed mode so that possible reduction of workload difference only comes from slack times for the transitory intervals before switching scheduling policy from *S1* to *S2* schedules. The other is that the more low-speed execution makes aperiodic tasks get delayed completion. Since the percentage of low-speed execution is relatively high comparing to the higher energy budget, reduced energy budget results in a longer response time for aperiodic tasks. Therefore, the response times of DPDS are minima at $\omega = 0.7$, at which both factors are moderate.

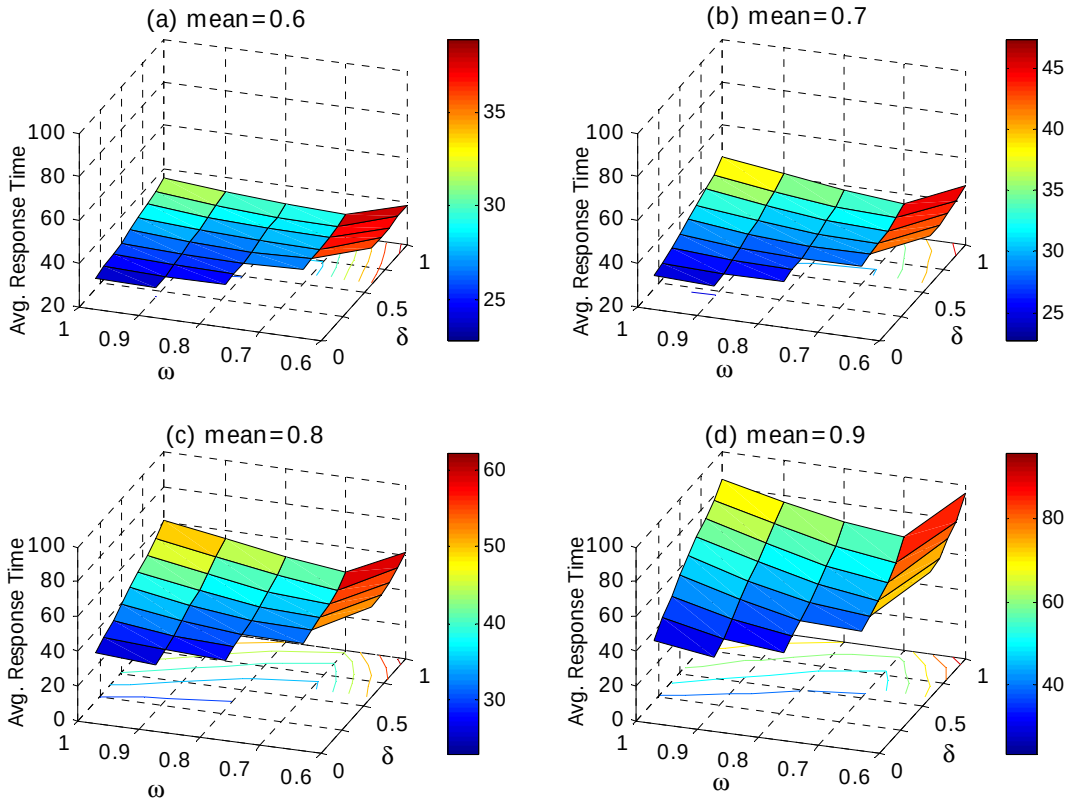


Figure 5-8 Average response times with respect to energy scaling factor δ (*S1* schedule) and constraint energy budgets when U_P=1.2 and U_A=0.3

Due to the property of graph representation, the entire graphs do not have corresponding results when δ = 0 and 0.1 although the average response times are 22.853, 23.286 when $E_C = E_{max}$ for Figure 5-8 (a), respectively. To compensate for the insufficient plots, we compare average

response times to the different δ and average of actual workload in Table 5-1, especially when the energy budget $E_C = E_{max}$. At $\delta = 0$, there's no big difference in average response time between DPDS and SPDS. But, average response time increases along with the increase of δ because the increased low-speed execution times for periodic tasks in *S1* schedule influence the responsiveness of aperiodic tasks.

Table 5-1 Average response times with respect to energy saving factor δ (S1 schedule) when $E_C = E_{max}$, U_P=1.2, U_A=0.3

Mean	δ	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.6	DP	22.85	23.29	24.25	25.32	26.67	27.70	29.00	30.41	32.33	33.82	36.05
	SP	22.87	22.87	22.87	22.87	22.87	22.87	22.87	22.87	22.87	22.87	22.87
0.7	DP	22.81	23.80	25.36	27.28	29.62	31.61	33.92	36.37	39.29	41.76	45.55
	SP	22.91	22.91	22.91	22.91	22.91	22.91	22.91	22.91	22.91	22.91	22.91
0.8	DP	22.93	24.70	28.03	31.73	36.05	39.86	43.69	47.57	51.97	55.66	60.82
	SP	22.96	22.96	22.96	22.96	22.96	22.96	22.96	22.96	22.96	22.96	22.96
0.9	DP	23.05	27.48	35.17	42.60	49.87	56.53	62.16	67.50	73.02	77.94	84.63
	SP	23.06	23.06	23.06	23.06	23.06	23.06	23.06	23.06	23.06	23.06	23.06

For the comparison to the SPDS, we plot average response times over the range of $0.6 \leq \delta \leq 1.0$ in Figure 5-9. As we decrease the energy budget, the dashed line for average response time of the SPDS increases monotonically by $\omega = 0.7$, but shows abrupt increments from $\omega = 0.7$ to $\omega = 0.6$. The slope of the abrupt increase is sharper as the average of actual workload of periodic tasks increases from Figure 5-9 (a) to (d). This is also indicative of the impact of increased percentage in low-mode execution time on the responsiveness. On the other hand, the solid lines for DPDS become minima when $\omega = 0.7$. The increased response time at low energy budget is explained by the same reason in the SPDS. But, as the energy budget is increased, the solid lines are proportionally increased along the increment of energy budget unlike the cases of the SPDS.

For all the cases in the figure, the responsiveness is gradually degraded along the increment of δ. And it should be noticed there's no big difference in the average response time between DPDS and SPDS when $\delta = 0.6$ and $\omega = 0.7$. The range of response time shown in Figure

5-9 (a) is very narrow, compared to Figure 5-9 (d). Considering the factors affect the average response times, the primary factor is how fast workload difference, which is generated for the intervals having only periodic tasks, is reduced to the amount that the *S2* schedule can afford to guarantee the schedulability, at which the aperiodic tasks can be executed and finished.

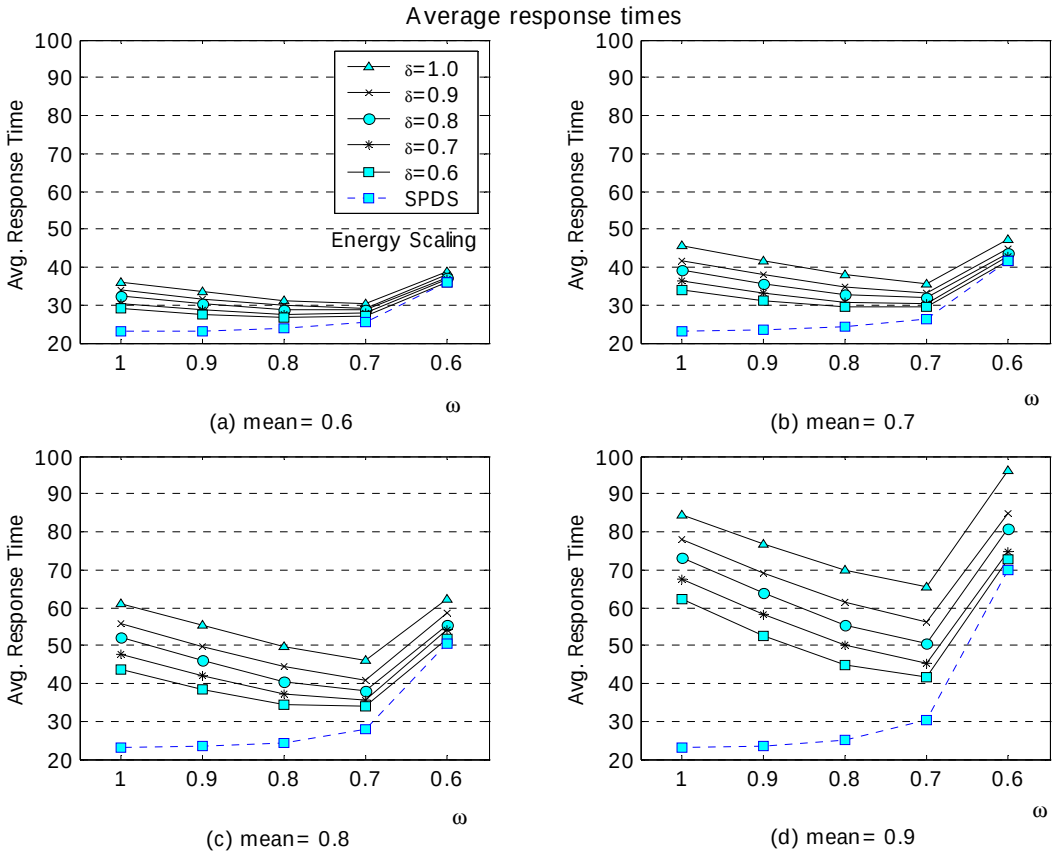


Figure 5-9 Comparisons of responsiveness between SPDS and DPDS with respect to constraint energy budget, $E_C = E_{min} + \omega E_{diff}$, and energy scaling factor δ
(In the range of $0.6 \le \omega \le 1.0$ and $0.6 \le \delta \le 1.0$ respectively, when U_P=1.2 and U_A =0.3)

At low average of actual workload of periodic tasks, slack times are the primary factor to reduce the workload difference so that the variation range of response times is close to SPDS with respect to the increase of δ. However, at higher average of actual workload of periodic tasks, we can observe the response times in Figure 5-9 (d) distribute loosely over wider range than Figure 5-9 (a). It tells that the reduction of the workload difference by slack times is weaker and

switched high-speed execution from low-speed assignment for the transitory intervals becomes more influential to the reduction as the average of actual workload of periodic tasks gets higher.

To find the causes that lead to the phenomena in Figure 5-8 and Figure 5-9, we now investigate how the workloads executed in switched speed mode by transitory schedule *S12* can impact on the responsiveness. We define '*T-HL*' as accumulated time of the switched execution in high-speed mode switched from the low-mode assigned in worst-case for the transitory intervals, when *S12* schedule is selected for scheduling policy. And the ratios of '*T-HL*' to the total execution time are plotted in Figure 5-10. The ratios of '*T-HL*' show bend-down surfaces over the change of energy budget, showing the maximum at the energy budget ω= 0.7. Comparing the result showing minimum response times at energy budget ω= 0.7 in Figure 5-8 and Figure 5-9, the maximum ratios of '*T-HL*' in Figure 5-10 are matched with the minimum response times at the specific energy budget. Furthermore, as we increase the average of actual workload of periodic tasks, the intensity of bend-downs is stronger and the maximum value of '*T-HL*' is getting higher as well. These trends are also matched with the wider variation in responsiveness from (a) to (d) in Figure 5-8 and Figure 5-9. It supports the fact that the reduction of workload difference is mainly caused by the switched execution '*T-HL*' rather than slack times as the average of actual workload of periodic tasks increases.

In addition, the workload executed in switched speed mode from high to low also affects to the response times. The more workload difference comes from switching schedule policies from *S2* to *S1*, the more time is required to nullify the non-zero workload difference and to select *S2* schedule that schedules aperiodic tasks. The time intervals that make the tasks schedulable by *S2* schedule delay the start to execute aperiodic tasks, leading longer response times. So, as defined in chapter 4, we here use the accumulator '*HtoL*' again to measure how much workload executed in switched speed mode for the interval having only periodic tasks.

By doing that, we can estimate how the workload difference can give an impact on the responsiveness. The ratios of '*T-HL*' are plotted in the range of $0.6 \leq \delta \leq 1.0$ as well as '*HtoL*' in Figure 5-11. Concerning energy consumption, '*HtoL*' tells us how much energy saving can be obtained, because energy consumption is reduced by the difference in energy consumption between two-speed modes.

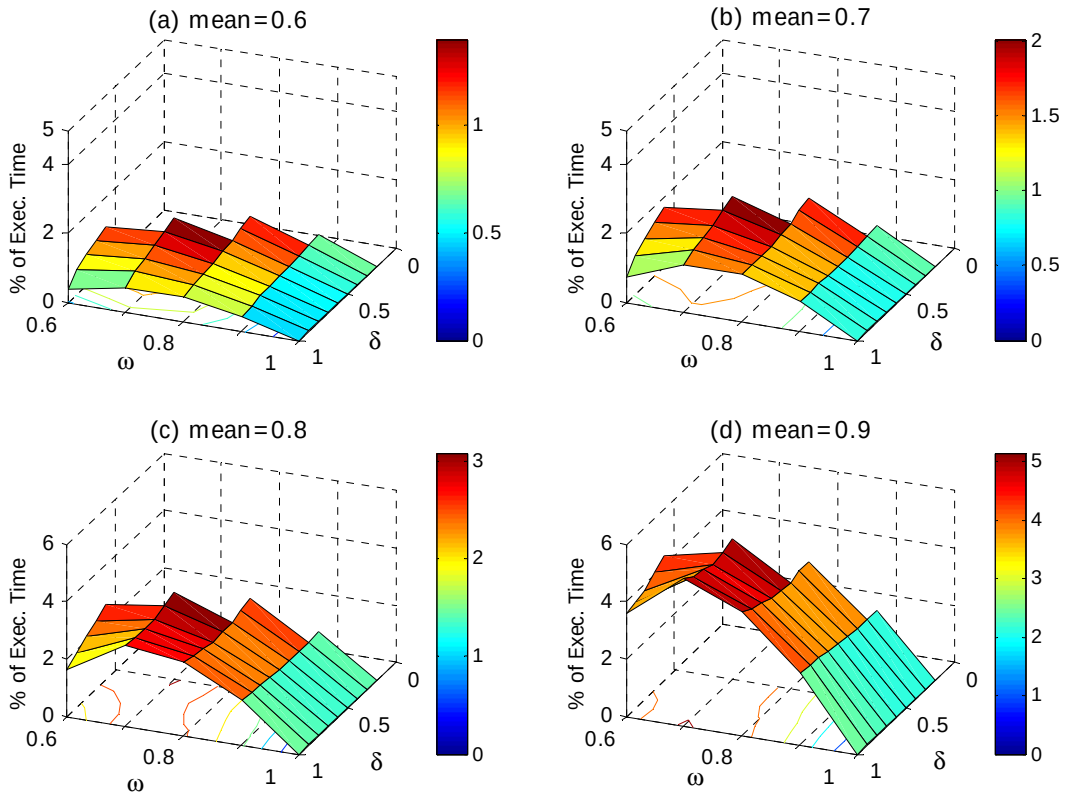


Figure 5-10 The percentages of switched execution times from assigned mode in worst-case for the intervals when the transitory schedule *S12* is selected to total execution time (In the range of $0.6 \leq \omega \leq 1.0$ and $0 \leq \delta \leq 1$ respectively, when U_P=1.2 and U_A=0.3)

Compared to the results in Figure 5-7, as the average of actual workload of periodic tasks increases, the workload ratio of '*HtoL*' is reduced and the variation range over the given range of δ is also narrowed. This is evidenced by that the shorter busy cycle is available for low actual workload of periodic tasks and the probability of busy cycle started with periodic tasks is higher accordingly such that the switched mode execution is increased. Additionally, the space between

the consecutive δs is wider when actual workload is low. This shows up as the slope of the energy saving in Figure 5-7, where the difference of energy saving between two adjacent δs is high in low workload.

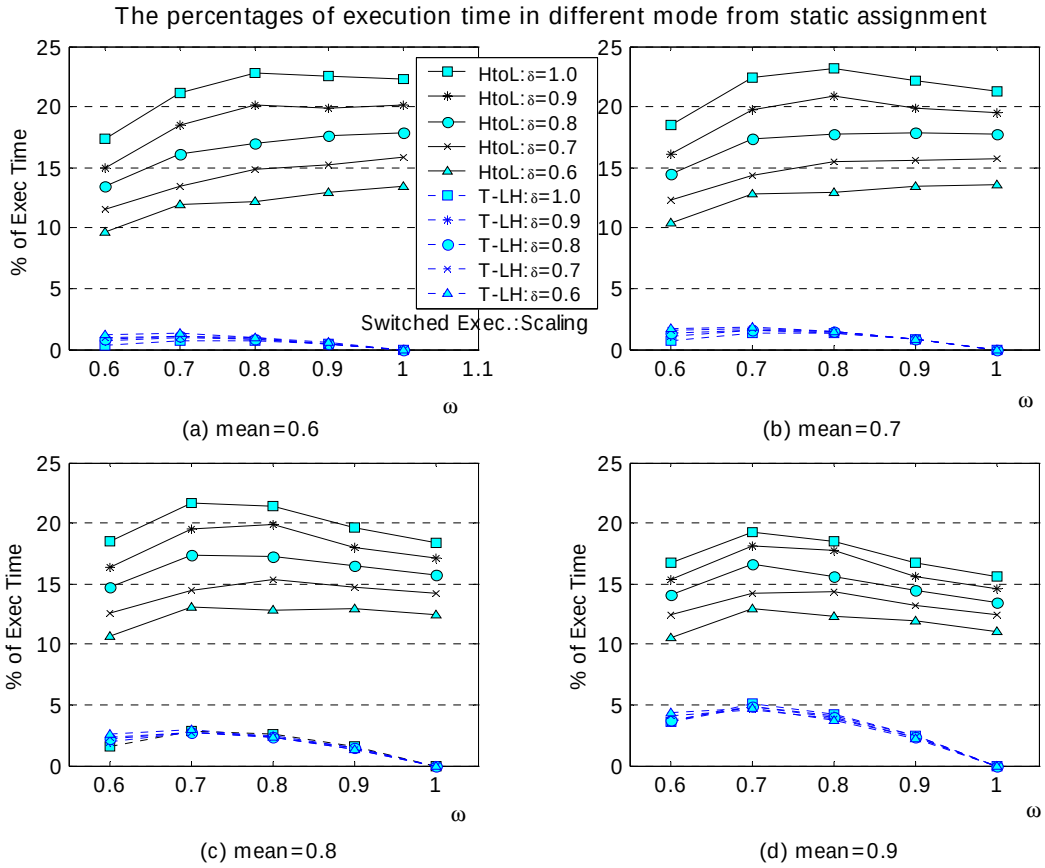


Figure 5-11 The percentages of execution time in changed mode execution from assigned mode in the worst-case with respect to constraint energy budget, $E_C = E_{min} + \omega E_{diff}$, and energy scaling factor δ
(In the range of $0.6 \le \omega \le 1.0$ and $0.6 \le \delta \le 1.0$ respectively, when U_P=1.2 and U_A =0.3)

To see how much energy saving can be obtained by adjusting the *S1* schedule with regard to the variation of periodic tasks' workload, we compute the ratio of energy consumption to static scheduling from our simulation and illustrate them as solid lines for DPDS and dashed lines for SPDS, in Figure 5-12, respectively. Fixed workload of aperiodic tasks as 0.3, both scheduling scheme show increase as we increase the workload of periodic tasks from 0.8 to 1.4 and decrease as we increase the energy budget from ω= 0.6 to ω= 1. But the difference of energy consumption

between them increases with respect to the increase of energy budget. This is the same trend as shown in Figure 5-7 and due to that the possibility of switched mode execution gets higher because the percentage of '*gd H for P*' is increased while the percentage of '*gd L for P*' in *S1* schedule remains at the same level as shown in Figure 5-6. Along the increase of the workload of periodic tasks from 0.8 to 1.4, the ratios by the change of *S1* schedule are closer each other and this is because the total execution time in high-speed mode is increased, while the applicable S1 schedules are invariant with the changes in the workload of periodic tasks.

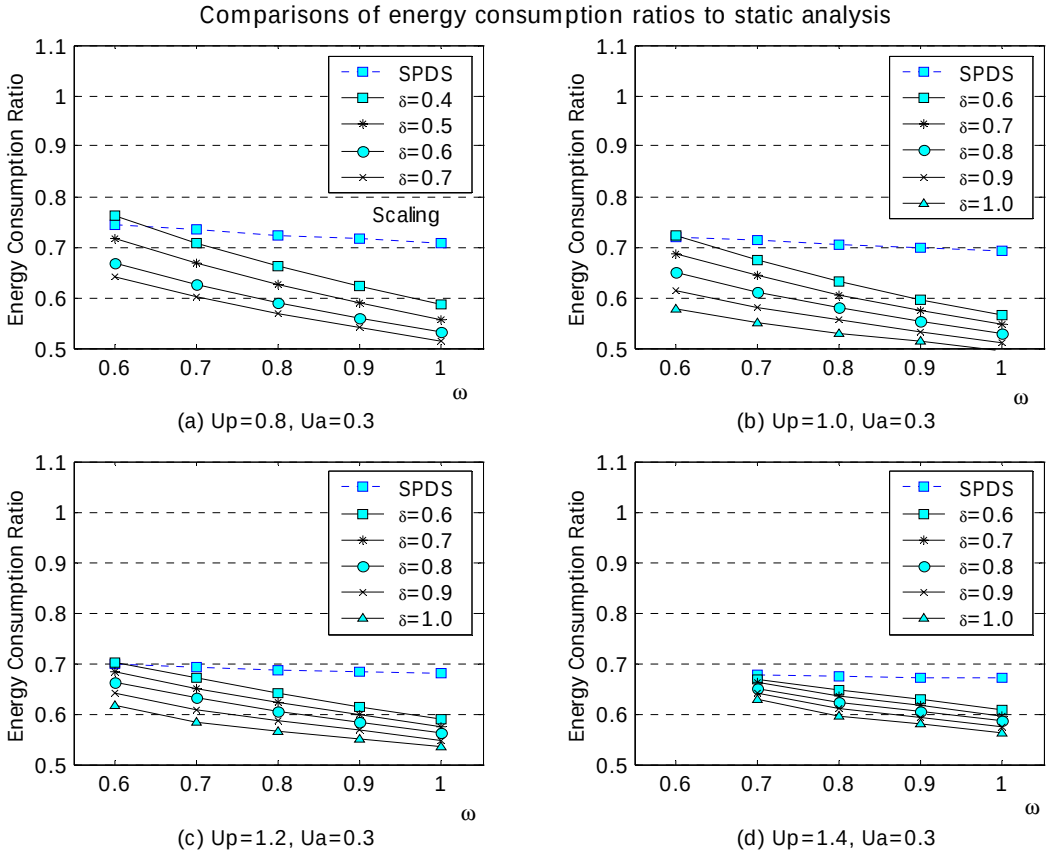


Figure 5-12 Comparisons of energy consumption ratios to static analysis between SPDS and DPDS with respect to constraint energy budget $E_C = E_{min} + \omega E_{diff}$ over the variation of workload demand in periodic tasks
(In the range of $0.6 \leq \omega \leq 1.0$ and available range of δ, respectively, when the average of actual periodic workload is 0.6)

One interesting thing in Figure 5-12 (a) is that there is no result for the cases of δ from 0.7 to 1.0. As the workload of periodic tasks is given as 0.8 maximally, but the range of δ is

defined from *min* U_P to unity, the maximum energy saving factor δ corresponding to *min* E_P becomes 0.7 for Figure 5-12 (a). Given aperiodic workload as shown in Figure 5-12, DPDS under periodic workload of 1.0 and the utilization of *S1* schedule of 0.99 consumes maximally 23% and 48% less energy than the SPDS and static scheduling, respectively. The reason why the energy consumption become minimum when periodic workload is 1.0 comparing with others is the same as the cause explained in section 4-5. That is the increment of low-speed execution time by the switched policies from *S2* to *S1* schedule become maximal when periodic workload is 1.0.

Also, we track the response times with respect to the change in periodic workload and plot them in Figure 5-13.

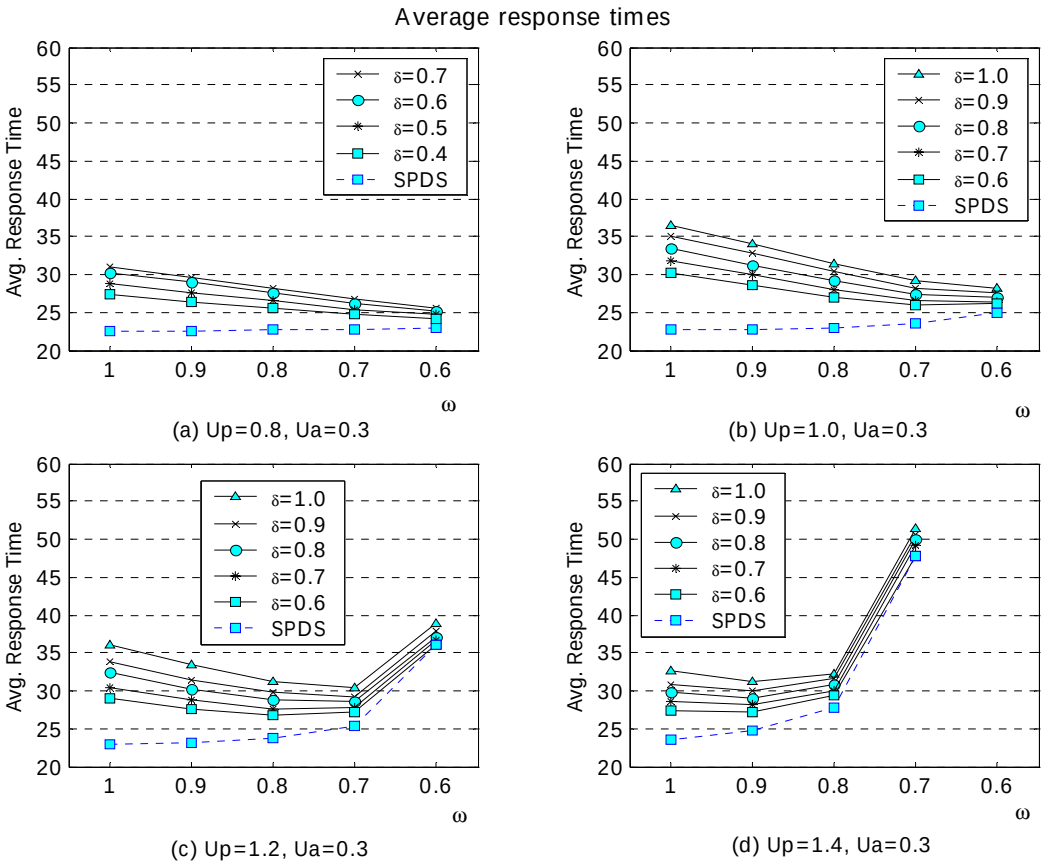


Figure 5-13 Comparisons of average response times between SPDS and DPDS with respect to constraint energy budget $E_C = E_{min} + \omega E_{diff}$ over the variation of workload demand in periodic tasks
(In the range of $0.6 \le \omega \le 1.0$ and available range of δ, respectively, when the average of actual periodic workload is 0.6)

To the fixed aperiodic workload, when the periodic workload are not bigger than 1.0, the average response times show monotonic decrement and increment for DPDS and SPDS, respectively, as we decrease energy budget as shown in Figure 5-13 (a) and (b). On the other hand, for the cases of that periodic workload are bigger than 1.0, Figure 5-13 (c) and (d) show abrupt increments at ω= 0.6 and ω= 0.7, respectively, after showing monotonic changes with the decrement of energy budget. The cause of these phenomena is also from the processing the overloaded workload for S2 schedule for the transitory interval as explained in Figure 5-8 and Figure 5-9, which show the same case in Figure 5-13 (c).

From the result of gradual differences in energy consumption and response time with the change of *S1* schedule shown in Figure 5-13, simple scaling of *S1* schedule' utilization can lead the adjustment of the energy consumption and responsiveness of DPDS to aimed performance of the real-time application.

5.5 Conclusion

In this chapter, we propose a scaling method that can control the levels of energy consumption and response times in scheduling of mixed hard and soft real-time under a bounded energy budget, i.e., dual-policy dynamic scheduling. DPDS takes two scheduling policies, the one consumes the worst-case energy in feasible range of energy consumption and utilization for mixed periodic and aperiodic tasks and the other does only for periodic tasks, according to the pattern of mixed tasks events. From the viewpoint of periodic tasks, the two sets of running modes are at extremes in feasible range of energy consumption and utilization of the DPDS. Utilizing the difference of energy consumption for periodic tasks existing in-between the two scheduling policies, it significantly reduces the energy consumption. Thus, applying the voltage settings/running speeds in-between extremes can control the degrees of energy consumption and average response time. We introduce a new factor for scaling in energy saving and responsiveness and adjust the dual-policy dynamic scheduling to the performance requirement of a real-time system application.

Our simulation studies show that simple increment or decrement of utilization in scheduling policy for the intervals having only the event of periodic tasks directly related to the energy consumption and responsiveness of dual-policy dynamic scheduling. Scaling up or down in energy consumption and responsiveness is proportionate to the adjustment of the utilization of periodic tasks. As shown in the results of DPDS, the scaling is possible with the increased or decreased execution time in low-speed mode switched from high-speed assignment during the intervals that have purely periodic tasks. Since scaling is plausible in the schedulable range of real-time tasks, the intensity of scaling is dependent on how much workloads are given for respective periodic and aperodic tasks in the system.

CHAPTER 6
CONCLUSIONS AND FUTURE WORK

6.1 Contributions

The objectives of the dissertation are to design a low-power scheduling for the improvement of total power consumption for hard-real-time applications and to build power-aware real-time systems supporting energy efficiency and high performance for real-time application consisting of hard and soft tasks under bounded energy budget. The main contributions of the dissertation are as follows:

A low-power scheduling for hard real-time systems. We devise comprehensive scheduling algorithms for integrated real-time systems. They include

(1) a fundamental two-mode scheduling theory,

(2) reduction of power consumption using power-aware voltage-clock scaling,

(3) dynamic reclaiming of early released resources, and

(4) optimal voltage setting in VCS-EDF scheduling.

Power-aware scheduling for mixed real-time tasks under a constrained energy budget. In order to develop a power-aware static analysis for real-time embedded system, the relationship of energy and workload demands is investigated. We develop an energy budget allocation model for battery-driven real-time systems, including not only small appliances being widely used, but also sensor network nodes for special purposes. The model concerns

(1) integrated scheduling of both hard and soft real-time tasks in a system,

(2) profiling of energy and processor's capacity usages on the basis of task

(3) efficient share of a bounded energy budget for battery-driven real-time systems, and

(4) better performance in non-real-time applications.

Dynamic scheduling for mixed real-time tasks considering power efficiency. We demonstrate that more sophisticated and effective power-aware dynamic scheduling may enhance the overall power consumption for real-time systems. They include

(1) a fundamental dual-policy dynamic scheduling theory,

(2) static analysis using the energy budget allocation model,

(3) modified Total Bandwidth Server for scheduling aperiodic tasks, and

(4) reduction of total energy consumption.

Optimizing energy efficiency and responsiveness. Given the energy-utilization tradeoffs in power-aware real-time scheduling, we show that the adjustment of the periodic tasks' workload within schedulable range can lead to an effective utilization of the limited energy budget from the understanding of the energy and performance implications. The scaling of periodic tasks' workload implies

(1) making the best use of a constraint energy budget,

(2) appropriate sharing of energy and processor's computation capacity in run-time scheduling,

(3) proper adjustment of energy consumption and responsiveness to the system's performance, and

(4) reduction of total energy consumption pursuing system's target performance.

6.2 Future Research Directions

Although much research has been devoted to energy efficient system design and utilization, this area has not yet reached complete maturity. As we can catch in the trend of power-aware real-time computing, the design of low-power or energy-efficient system revolves total aspects of computer engineering fields such as energy aware software and compilers, the properties of power sources, the memory hierarchy and controlling method, communication and so on. Considering these aspects, we describe future research directions, which should be extended or developed with the dissertation as a starting point, as follows:

Scalable VCS-EDF scheduling. We are planning to enhance the scheduling algorithm in three directions. First, we will extend the two-voltage mode scheduling to dynamic multiple-voltage modes such that it can benefit from the modern voltage-variable processors for the low-power system design. Second, we will extend the current uni-processor architecture to the multiprocessor and distributed real-time computing environment. Third, we will develop additional scheduling strategies, which consider resource-sharing environment.

Enhancement of the constrained energy budget allocation model. We are investigating the impacts of discharge and recharge properties of power source on the battery-powered real-time embedded systems and planning to solve the energy budget allocation problem considering the knowledge of power source. In addition, we will develop additional energy budget allocation strategies in the environment having multiple energy sources, which is getting popular in the field.

Architectural or functional energy allocation and scheduling analyses. Considering most power consuming function in a real-time embedded system, computations in processor are the main source, but the other elements such as memory access, wired or wireless communication including application software also increase along with the various demands in complex functionalities. We will scrutinize and extend our model to each component in real-time systems based on the study other application requirements. Or we can merge the more detailed energy profiles and characteristics that have been studied in many researches into our energy allocation model, and static and dynamic scheduling analyses.

LIST OF REFERENCES

[1] E. P. Harris, S. W. Depp, W. E. Pence, S. Kirkpatrick, M. Sri-Jayantha, and R. R. Troutman, "Technology Directions for Portable Computers," Proceedings of the IEEE, Institute of Electrical and Electronics Engineers, 1995 83 (4): 636-658.

[2] K. Kawamoto, J. G. Koomey, B. Nordman, R. E. Brown, M. A. Piette, and A. K. Meier, "Electricity Used by Office Equipment and Network Equipment in the U.S.: Detailed Report and Appendices," NBNL-45917, Ernest Orlando Lawrence Berkeley National Laboratory, CA, February 2001, Available at http://enduse.lbl.gov/Info/45917b-abstract.html, Site last visited May 2002.

[3] L. Benini and G. D. Micheli, "System-level Power Optimization: Techniques and Tools," ACM Trans. on Design Automation of Electronic Systems 2000: 288-293.

[4] L. Benini, A. Bogliolo, and G. D. Micheli, "A Survey of Design Techniques for System-level Dynamic Power Management," IEEE Trans. on Very Large Scale Integration Systems 2000; 8 (3): 299-316.

[5] Microsoft, "OnNow: The Evolution of the PC platform," Available at http://www.microsoft.com/HWDEV/desinit/onnow1.htm, Site last visited January 2002.

[6] Compaq, Intel, Microsoft, Phoenix, and Toshiba, "Advanced Configuration and Power Interface specification," Available at http://www.intel.com/technology/IAPC/tech.htm, Site last visited May 2000.

[7] A. Glenn, "Linux ACPI HOWTO," January 2001, Available at http://www.columbia.edu/~ariel/acpi/acpi_howto.txt, Site last visited August, 2001.

[8] T. Simunic, L. Benini, P. Glynn, and G. D. Micheli, "Dynamic Power Management for Portable Systems," Proceedings of International Conference on Mobile Computing and Networking, New York, NY, ACM, 2000: 22-32.

[9] T. Simunic, L. Benini, and G. D. Micheli, "Event-driven Power Management," Proceedings of International Symposium on System Synthesis, Los Alamitos, CA, IEEE Computer Society Press, 1999: 18-23.

[10] G. Paleologo, L. Benini, A. Bogliolo, and G. D. Micheli, "Policy Optimization for Dynamic Power Management," IEEE Trans. on CAD 1999.

[11] Q. Qiu, and M. Pedram, "Dynamic Power Management based on Continuous Time Markov Decision Process," Proceedings of Design Automation Conference 1999: 555-561.

[12] DARPA, Power Aware Computing/Communication (PAC/C), Available at http://www.darpa.mil/ito/research/pacc/index.html, Site last visited January, 2003.

[13] M. Doyle, T. F. Fuller, and J. Newman, "Modeling of Galvanostatic Charge and Discharge of the Lithium/Polymer/Insertion Cell," Journal of Electrochemistry Society 1993; 140: 1526-1533.

[14] T. L. Martin and D. P. Siewiorek, "Non-ideal Battery Properties and Low Power Operation in Wearable Computing," Proceedings of International Symposium on Wearable Computers, Los Alamitos, CA, IEEE Computer Society Press, 1999.

[15] L. Benini, G. Castelli, A. Macii, E. Macii, M. Poncino, and R. Scarsi, "A Discrete-time Battery Model for High-level Power Estimation," Proceedings of the Design, Automation and Test in Europe, Paris, France 2000 March: 35-41.

[16] L. Benini, G. Castelli, A. Macii, E. Macii, and R. Scarsi "Battery-driven Dynamic Power Management of Portable Systems," Proceedings of International Symposium on System Synthesis, Los Alamitos, CA, IEEE Computer Society Press, 2000 September.

[17] D. Panigrahi, C. Chiasserini, S. Dey, R. Rao, A. Raghunathan, and K. Lahiri, "Battery Life Estimation of Mobile Embedded Systems," Proceedings of International Conference on VLSI Design, Los Alamitos, CA, IEEE Computer Society Press, 2001.

[18] D. Brooks, V. Tiwari, and M. Martonosi "Wattch: A Framework for Architectural-level Power Analysis and Optimizations," Proceedings of Annual International Symposium on Computer Architecture, Los Alamitos, CA, IEEE Computer Society Press, 2000 June: 83-94.

[19] W. Ye, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin. "The Design and Use of SimplePower: A Cycle Accurate Energy Estimation Tool," Proceedings of Design Automation Conference, Los Alamitos, CA, IEEE Computer Society Press, 2000 June: 340-345.

[20] J. Flinn and M. Satyanarayanan, "PowerScope: a Tool for Profiling the Energy Usage of Mobile Applications," Proceedings of IEEE Workshop on Mobile Computing Systems and Applications, Los Alamitos, CA, USA, IEEE Computer Society Press, 1999 February: 2-10.

[21] M. Lajolo, A. Raghunathan, S. Dey, and L. Lavagno, "Efficient Power Co-estimation Techniques for System-on-Chip Design," Proceedings of Design, Automation and Test in Europe, Los Alamitos, CA, IEEE Computer Society Press, 2000 March.

[22] X. Fan, C. S. Ellis, and A. R. Lebeck, "The Synergy between Power-aware Memory Systems and Processor Voltage Scaling," Technical Report No.: CS-2002-12, Duke University 2002 November.

[23] D. Shin, H. Shim, Y. Joo, H.Yun, J. Kim, and N. Chang "Energy-monitoring Tool for Low-power Embedded Programs," IEEE Design & Test of Computers, Los Alamitos, CA, IEEE Computer Society Press, 2002 July-August.

[24] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS Digital Design," IEEE Journal of Solid-state Circuits 1992; 27 (4): 473-484.

[25] K. Nose and T. Sakurai, "Optimization of V_{DD} and V_{TH} for Low-power and High-speed Applications," Proceedings on the Conference of Asia and South Pacific Design Automatiion Conference, Los Alamitos, CA, IEEE Computer Society Press, 2000 January; A6.1: 469-474.

[26] T. Kuroda, K. Suzuki, S. Mira, T. Fujita, F. Yamane, F. Sano, C. Akihiko, Y. Watanabe, M. Yoshinori, K. Matsuda, T. Maeda, T. Sakurai, and F. Tohru, "Variable Supply-voltage Scheme for Low-power High-speed CMOS Digital Design," IEEE Journal of Solid State Circuits 1998 March; 33 (3): 454-462.

[27] J. Heeb, "The next generation of StrongArm," Embedded Processor Forum, MDR, 1999 May.

[28] Intel Corporation, "Mobile Pentium III Processor in BGA2 and micro-PGA2 packages," [datasheet] Order #245302-002, 2000.

[29] Transmeta Corporation, "TN5400Processor Specification," Available at http://www.transmeta.com/, Jan. 2000, Site last visited May 2001.

[30] Intel Corporation, "The Intel XScale Technology," Available at http://intel.com/design/intelxscale, Site last visited November 2002.

[31] J. Pouwelse, K. Langendoen, and H. Sips, "Dynamic Voltage Scaling on a Low-power Microprocessor," Proceedings of the International Symposium on Mobile Multimedia Systems & Applications, Los Alamitos, CA, IEEE Computer Society Press, 2000: 157-164.

[32] O. Serlin. "Scheduling of Time Critical Processes," Proceedings of the Spring Joint Computer Conference, Montvale, NJ, AFIPS Press, 1972; May: 925-932.

[33] C. L. Liu and J. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-time Environment," Journal of the ACM 1973; 20: 46-61.

[34] J. P. Lehoczky, L. Sha, and Y. Ding, "The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior," Proceedings of IEEE Real-Time Systems Symposium, New York, NY, IEEE, 1989: 166-171.

[35] B. Sprunt, L. Sha and J. P. Lehoczky, "Aperiodic Task Scheduling for Hard Real-time Systems," The Journal of Real-Time Systems 1989; 1: 27-60.

[36] J. K. Strosnider, J. P. Lehoczky, and L. Sha, "The Deferrable Server Algorithms for Enhanced Aperiodic Responsiveness in Hard Real-Time Environments," IEEE Trans. on Computers 1995 January; 44 (1): 73-91.

[37] J. P. Lehoczky and S. Ramos-Thuel, "An Optimal Algorithm for Scheduling Soft-aperiodic Tasks in Fixed-priority Preemptive Systems," Proceedings of IEEE Real-Time Systems Symposium, New York, NY, IEEE, 1992 December: 110-123.

[38] T. M. Ghazalie and T. P. Baker. "Aperiodic Servers in a Deadline Scheduling Environment," The Journal of Real-Time Systems, Boston, Kluwer Academic Publishers, 1995: 31- 67.

[39] M. Spuri and G. Buttazzo, “Efficient Aperiodic Service under Earliest Deadline Scheduling,” Proceedings of IEEE Real-Time System Symposium, New York, NY, IEEE, 1994: 2-21.

[40] M. Spuri and G. Buttazzo, “Scheduling Aperiodic Tasks in Dynamic Priority Systems,” The Journal of Real-Time Systems, Boston, Kluwer Academic Publishers, 1996 March; 10 (2): 179-210.

[41] G. Lipari, G. Buttazzo, and L. Abeni, “A Bandwidth Reservation Algorithm for Multi-application Systems,” Proceedings of the IEEE International Conference on Real-Time Computing Systems and Applications, Los Alamitos, CA, IEEE Computer Society Press, 1998 October: 77-82.

[42] M. Caccamo, G. Lipari, and G. Buttazzo, “Sharing Resources among Periodic and Aperiodic Tasks With Dynamic Deadlines,” Proceedings of the IEEE Real-Time Systems Symposium, Los Alamitos, CA, IEEE Computer Society Press, 1999 December: 284-293.

[43] G. Lipari and G. Buttazzo, “Schedulability Analysis of Periodic and Aperiodic Tasks with Resource Constraints,” Journal of Systems Architecture 2000 January; 46 (4): 327-338.

[44] M. Weiser, B. Welch, A. Demers, and S. Shenker, “Scheduling for reduced CPU energy,” Proceedings of USENIX Symposium on Operating Systems Design and Implementation, Berkeley, CA, USENIX Association, 1994: 13-23.

[45] K. Govil, E. Chan, and H. Wasserman, “Comparing Algorithms for Dynamic Speed-setting of a Low Power CPU,” Proceeding of Mobile Computing and Networking, New York, ACM, 1995 November: 13-25.

[46] T. Pering, T. Burd, and R. Brodersen, “The Simulation and Evaluation of Dynamic Voltage Scheduling algorithms,” Proceedings of International Symposium on Low Power Electronics and Design, New York, ACM, 1998: 76-81.

[47] T. Pering, T. Burd, and R. Brodersen, “Voltage Scheduling in the IpARM Microprocessor System,” Proceedings of the International Symposium on Low Power Electronics and Design, New York, ACM, 2000 July; 96-101.

[48] D. Grunwald, P. Levis, K. Farkas, C. Morrey, and M. Neufeld. “Policies for dynamic clock scheduling,” Proceedings of USENIX Symposium on Operating Systems Design and Implementation, 2000 October.

[49] T. E. Truman, T. Pering, R. Doering, and R. W. Brodersen, “The InfoPad Multimedia Terminal: A Portable Device for Wireless Information Access,” IEEE Trans. on Computers, CA, USENIX Association, 1998; 47 (10): 1073-1087.

[50] A. Sinha and A. Chandrakasan, “Dynamic Voltage Scheduling Using Adaptive Filtering of Workload Traces," Proceedings of International Conference on VLSI Design, Los Alamitos, CA, IEEE Computer Society Press, 2001 January.

[51] T. Pering and R. Brodersen, “Energy Efficient Voltage Scheduling for Real-time Operating Systems,” Proceedings of IEEE Real-Time Technology and Applications Symposium, Works In Progress Session, Los Alamitos, CA, IEEE Computer Society Press, 1998.

[52] V. Swaminathan and K. Chakrabarty, “Real-time Task Scheduling for Energy-aware Embedded Systems,” Proceeding of IEEE Real-Time Systems Symposium (Work-In-Progress Sessions), New York, NY, IEEE, 2000 November.

[53] F. Yao, A. Demers, and S. Shenker, “A Scheduling Model for Reduced CPU Energy,” IEEE Symposium on Foundations of Computer Science, Los Alamitos, CA, IEEE Computer Society Press, 1995: 374-382.

[54] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M. B. Srivastava, “Power Optimization of Variable Voltage Core-based Systems,” Proceedings of Design Automation Conference, Los Alamitos, CA, IEEE Computer Society Press, 1998: 176-181.

[55] I. Hong, G. Qu, M. Potkonjak, and M. B. Srivastava, “Synthesis Techniques for Low-power Hard Real-Time Systems on Variable Voltage Processors,” Proceeding of IEEE Real-Time Systems Symposium, New York, NY, IEEE, 1998: pp178-187.

[56] G. Quan and X. Hu “Energy Efficient Fixed-priority Scheduling for Real-Time Systems on Variable Voltage Processors,” Proceedings of Design Automation Conference, Los Alamitos, CA, IEEE Computer Society Press, 2001.

[57] I. Hong, M. Potkonjak, and M. Srivastava, “On-line Scheduling of Hard Real-time Tasks on Variable Voltage Processor,” Proceedings of IEEE/ACM International Conference on Computer-Aided Design, Los Alamitos, CA, IEEE Computer Society Press, 1998: 653-656.

[58] Y. H. Lee and C. M. Krishna, “Voltage-Clock Scaling for Low Energy Consumption in Real-Time Embedded Systems,” Proceedings of Real-Time Computing Systems and Applications, Los Alamitos, CA, IEEE Computer Society Press, 1999 December.

[59] T. Ishihara and H. Yasuura, “Voltage Scheduling Problem for Dynamically Variable Voltage Processors,” Proceedings of International Symposium on Low Power Electronics and Design, New York, ACM, 1998: 197-202.

[60] R. Jejurikar and R. K. Gupta, “Energy Aware Task Scheduling with Task Synchronization for Embedded Real Time Systems,” Technical Report No.: #02-21, University of California at Irvine 2002 June.

[61] L. Geppert and T. Perry, “Transmeta’s Magic Show,” IEEE Spectrum 2000 May; 37 (5): 22-32.

[62] T. Simunic, H. Vikalo, P. Glynn, and G. D. Micheli, "Energy Efficient Design of Portable Wireless Systems," Proceedings on the International Symposium on Low Power Electronics and Design, New York, ACM, 2000: 49-54.

[63] T. Simunic, L. Benini, A. Acquaviva, P. Glynn, and G. D. Micheli, “Dynamic voltage scaling and power management for portable systems,” Proceedings of conference on Design Automation Conference, Los Alamitos, CA, IEEE Computer Society Press, 2001: 524-529, 2001.

[64] M. Kim and S. Ha, “Hybrid Run-time Power Management Techniques for Real-time Embedded System with Voltage Scalable Processor,” ACM SIGPLAN Workshop on

Languages, Compilers, and Tools for Embedded Systems, New York, Springer, 2001: 11-19.

[65] J. Wegener and F. Mueller, "A Comparison of Static Analysis and Evolutionary Testing for the Verification of Timing Constraints," Proceedings of Real-Time Technology and Applications Symposium, Los Alamitos, CA, IEEE Computer Society Press, 1998 November; 21(3): 241-268.

[66] D. Linden, Handbook of batteries, 2nd ed. McGraw-Hill, Highstown, N.J., 1995.

[67] W. R. Hamburgen, D. A. Wallach, M. A. Viredaz, L. S. Brakmo, C. A. Waldspurger, J. F. Bartlett, T. Mann, and K. I. Farkas. "Itsy: Stretching the Bounds of Mobile Computing," IEEE Computer 2001 April; 34 (4) 28-36.

[68] T. L. Martin, "Balancing Batteries, Power, and Performance: Systems Issues in CPU Speed-setting for Mobile Computing," [dissertation], Carnegie Mellon University; 1999 August.

[69] T. L. Martin and D. P. Siewiorek, "The Impact of Battery Capacity and Memory Bandwidth on CPU Speed-setting: A Case Study," Proceedings of the 1999 International Symposium on Low power Electronics and Design, New York, ACM, 1999 August: 200-205.

[70] T. Simunic, L. Benini, and G. D. Micheli, ""Energy Efficient Design of Battery-powered Embedded Systems," Special Issue of IEEE Transactions on VLSI 2001 May.

[71] M. Pedram and Q. Wu, "Battery-powered Digital CMOS Design," Proceedings of the Design, Automation and Test in Europe Conference and Exhibition, New York, ACM, 1999: 17-23.

[72] M. Pedram, and Q. Wu, "Design Considerations for Battery-powered Electronics," Proceedings of the Design Automation Conference, Los Alamitos, CA, IEEE Computer Society Press, 1999 June: 861-866.

[73] L. Benini, G. Castelli, A. Macii, and R. Scarsi, "Battery-driven Dynamic Power Management," IEEE Design & Test of Computers, Los Alamitos, CA, IEEE Computer Society Press, 2001; 18 (2): 53-60.

[74] J. Luo and N. K. Jha, "Battery-aware Static Scheduling for Distributed Real-time Embedded Systems," Proceedings of Conference on Design Automation Conferences, Los Alamitos, CA, IEEE Computer Society Press, 2001 June: 444-449.

[75] J. Liu, P. H. Chou, N. Bagherzadeh, and F. Kurdahi, "Power-aware Scheduling under Timing Constraints for Mission-Critical Embedded Systems," Proceedings of Conference on Design Automation Conference, Los Alamitos, CA, IEEE Computer Society Press Press, 2001: 840-845.

[76] Advanced RISC Machines Ltd, "Introduction to Thumb," ARM Documentation.

[77] Motorola, MPC860 PowerPC Hardware Specification, MPC860EC/D, 1998 December.

[78] C. M. Krishna and Y. H. Lee, "Voltage-Clock Scaling Adaptive Scheduling Techniques for Low Power in Hard Real-time Systems," IEEE Proceedings of Real Time Technology and Applications Symposium, Los Alamitos, CA, IEEE Computer Society Press, 2000 May.

[79] K. Lahiri, A. Raghunathan, S. Dey, and D. Panigrahi, "Battery-driven System Design: A New Frontier in Low Power Design," Proceedings of the International Conference on VLSI Design /ASP-DAC, Los Alamitos, CA, IEEE Computer Society Press, 2002 January: 261-267.

[80] J. A. Stankovic, M. Spuri, K. Ramamritham and G. C. Buttazzo, Deadline Scheduling for Real-Time Systems: EDF and Related Algorithms, Boston, Kluwer Academic Publishers, 1998.

[81] M. Stemm and R. H. Katz, "Measuring and Reducing Energy Consumption of Network Interfaces in Hand-held Devices," IEICE Transactions on Communications 1997; E80-B (8): 1125-1131.

[82] H. Zeng, X. Fan, C. Ellis, A. Lebeck, and A. Vahdat, **"**ECOSystem: Managing Energy as a First Class Operating System Resource," Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems 2002.

[83] Y. Doh, D. Kim, Y. H. Lee, and C. M. Krishna, "Constrained Energy Budget Allocation for Mixed Hard and Soft Real-time Tasks," Proceedings of the IEEE International Conference on Real-Time Computing Systems and Applications, To be published in Lecture Notes in Computer Science, Heidelberg, Springer-Verlag, 2003.

BIOGRAPHICAL SKETCH

Yoonmee Doh was born in Taegu, South Korea. She got BE and ME degrees in electronic engineering from Kyungbook National University, South Korea, in 1989 and 1991 respectively. She has been a member of the research staff at the Electronics and Telecommunications Research Institute (ETRI), South Korea, since 1992. She participated in the development of the ATM LAN/WAN switching system and led the embedded S/W and H/W group at ETRI. Pursuing a doctoral degree, she joined the Computer and Information Science and Engineering Department at the University of Florida, Gainesville, FL, in May 1998. During her doctoral study, she had been a visiting research scholar in the Department of Computer Engineering at Arizona State University from January 2001. Her research interests revolve around power aware computing, low power scheduling, real-time systems (scheduling, communication, and operating systems), wireless and mobile networks, distributed and fault tolerant systems, video on demand systems, and QOS in high speed networks.