ABSTRACT

Title of Dissertation: PROVIDING QOS WITH REDUCED ENERGY CONSUMPTION VIA REAL-TIME VOLTAGE SCALING ON EMBEDDED SYSTEMS

Shaoxiong Hua, Doctor of Philosophy, 2004

Dissertation directed by: Professor Gang Qu
Department of Electrical and Computer Engineering

Low energy consumption has emerged as one of the most important design objectives for many modern embedded systems, particularly the battery-operated PDAs. For some soft real-time applications such as multimedia applications, occasional deadline misses can be tolerated. How to leverage this feature to save more energy while still meeting the user required quality of service (QoS) is the research topic this thesis focuses on. We have proposed a new probabilistic design methodology, a set of energy reduction techniques for single and multiple processor systems by using dynamic voltage scaling (DVS), the practical solutions to voltage set-up problem for multiple voltage DVS system, and a new QoS metric.

Most present design space exploration techniques, which are based on application's worst case execution time, often lead to over-designing systems. We have

proposed the probabilistic design methodology for soft real-time embedded systems by using detailed execution time information in order to reduce the system resources while delivering the user required QoS probabilistically.

One important phase in the probabilistic design methodology is the offline/online resource management. As an example, we have proposed a set of energy reduction techniques by employing DVS techniques to exploit the slacks arising from the tolerance to deadline misses for single and multiple processor systems while meeting the user required completion ratio statistically.

Multiple-voltage DVS system is predicted as the future low-power system by International Technology Roadmap for Semiconductors (ITRS). In order to find the best way to employ DVS, we have formulated the voltage set-up problem and provided its practical solutions that seek the most energy efficient voltage setting for the design of multiple-voltage DVS systems. We have also presented a case study in designing energy-efficient dual voltage soft real-time system with (m, k)-firm deadline guarantee.

Although completion ratio is widely used as a QoS metric, it can only be applied to the applications with independent tasks. We have proposed a new QoS metric that differentiates firm and soft deadlines and considers the task dependency as well. Based on this new metric, we have developed a set of online scheduling algorithms that enhance quality of presentation (QoP) significantly, particularly for overloaded systems.

PROVIDING QOS WITH REDUCED ENERGY CONSUMPTION VIA REAL-TIME VOLTAGE SCALING ON EMBEDDED SYSTEMS

by

Shaoxiong Hua

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2004

DEDICATION

To my wife and our parents

ACKNOWLEDGEMENTS

First of all, I would like to sincerely thank my advisor, Professor Gang Qu, for his academic guidance, encouragement and patience over the past three years. When I switched my major to computer engineering, it is him who gave me a lot of advice and helped me find a good research topic. I wish to thank him for dedicating so much time to help me learn how to conduct high quality research and how to present our results. This dissertation would not be possible without his encouragement and suggestions.

I would also like to thank Professor Shuvra S. Bhattacharyya for his kind help and suggestions during my Ph.D. studies. I also wish to thank the other members of my dissertation committee, Professor Ralph Etienne-Cummings, Professor K. J. Ray Liu, and Professor Chau-Wen Tseng, for kindly serving on the committee with invaluable comments.

Next, I would like to thank Professor Bingen Yang of University of Southern California and Professor John S. Baras for their kind support and giving me the chance to USA and University of Maryland to pursue further education. I would like to thank Mr. Lane Smith and Mr. Fred Fischer, both in Agere Systems Inc., for kindly providing me the summer intern position in the last three years. Furthermore, I would like to thank Professor Manoj Franklin and Professor William Hawkins for their kind help during my Ph.D. studies.

I would like to thank my officemates and colleagues in the Embedded Systems Research Laboratory. I enjoyed the collaboration with other fellow graduate and undergraduate students, in particular Vida Kianzad, Guang Han, Pushkin R. Pari, Yuan Lin and Melissa Barker, to name a few.

I owe a great deal to my parents and parents-in-law for their love, encouragement and unconditional support throughout the course of my Ph.D. studies.

Finally and most importantly, I would like to especially thank my lovely wife, Huixian, for her endless love, patience, encouragement and support in these many years. Without her support, the dissertation would not have been completed.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

With the development of Very Large Scale Integration (VLSI) techniques, the total number of devices on a chip has doubled every 24 months, known widely as "Moore's Law". It is predicted that before 2010, the chip will have 1 billion transistors [37]! This results in the fact that embedded systems have been moving from board-level systems to System-on-Chips (SoCs) and it provides us the opportunities to find more embedded system applications.

On the other hand, the importance of time-to-market becomes more and more significant. The Semico Research Corporation reports that for Black & White TV, it took almost 18 years to reach the shipment of one million units while for DVD player it took only less than one year. The profit window is open for a short period of time and only to the early technology providers [38]. Therefore the design of embedded systems must move from craft to discipline to increase the productivity and decrease the time-to-market.

With the decrease in the transistor feature size and the increase in the number of transistors on a chip, the power dissipation has been increased dramatically. For example, when the transistor feature size is less than 100nm, the power dissipation will be more than 300Watts [1]. High power dissipation results in many hot spots in the thermal map of the working chip that will limit the allowed performance. More power consumption means more heat dissipation and we have to use more and bigger heatsinks and fans, bigger and more expensive motherboards and cases. And more power also means the lower reliability and shorter battery life time for the battery-operated systems. Because of the importance of power, low power has emerged as one of the most important design objectives for the embedded system designers.

In order to reduce the design cost (e.g., decrease the time-to-market and reduce the fabrication cost) and system cost (e.g., power), one needs to increase the level of design abstraction and develops system-level design methodology. In this dissertation, I will mainly focus on the soft real-time embedded system design. Specifically, we have developed energy reduction techniques by employing dynamic voltage scaling (DVS) while meeting user required quality of service (QoS) statistically. In this chapter, I will give the introduction of soft real-time embedded systems, dynamic voltage scaling and quality of service.

1.1 Soft Real-Time Embedded Systems

Embedded systems are widely used in many applications such as encoding and decoding of audio and video, digital control, the monitoring of large rotating machinery, radar signal processing and tracking and so on. Different from the general-purpose systems such as desktop systems, the embedded systems are those that use single or multiple microprocessors to implement the *dedicated applications* [20]. This means that every application is the subject of a special development that must directly produce the product satisfying user requirements, costs and deadline. Therefore it is possible for the designers to get the detailed information of the application such as sampling data rate, data distribution etc. that can be used during the design stage.

Most embedded systems are *real-time* systems, which are required to react to stimuli from the environment and complete their work and deliver their services within time intervals dictated by the environment [75]. In real-time systems, timeliness is the key characteristic. In such systems, we often use the *deadline* to be one of the important time constraints of the task (workload). There are three types of deadline:

- **Hard deadline:** A task has a **hard** deadline if it must be completed before the deadline otherwise the system will be in fault and deadline missing may cause catastrophic consequences.
- **Firm deadline:** A task has a **firm** deadline if it must be completed before the deadline otherwise although the system will not be in fault, it will not get any reward for serving the task.
- **Soft deadline:** A task has a **soft** deadline if the system can still benefit even if the deadline is missed, subjected to a deadline miss penalty.

In hard real-time systems all deadlines of the tasks are hard while in soft real-time systems portion of deadlines are soft. In this dissertation, we mainly focus on the design of soft real-time embedded systems that conduct the repetitive data processing that can be found in many DSP applications. Such systems require moderately high performance and can tolerate occasional deadline misses. The timing requirements of such systems are often specified in probabilistic terms [75].

And the actual execution time of the task is often varied and deviates from its worst case execution time (WCET), sometimes by a large amount.

1.2 Dynamic Voltage Scaling (DVS)

As low power/energy consumption has emerged as one of the most important design objectives, reducing the supply voltage (voltage scaling) becomes one of the most effective techniques to decrease the power/energy consumption [123].

There are three major sources of power consumption, i.e., switching component, direct-path short circuit current and leakage current [23]. Although leakage power dissipation is gaining more and more attention recently, switch component (dynamic) power still dominates in most embedded systems. Dynamic power in a CMOS circuit is proportional to $\alpha C_L V_{dd}^2 f_{clock}$, where αC_L is the effective switched capacitance, V_{dd} is the supply voltage, and f_{clock} is the clock frequency. As the power and energy are the quadratic functions of supply voltage, reducing the supply voltage can result in substantial power and energy saving and in general more effective than the technique that shuts down a processor when it is idle. Roughly speaking, system's power dissipation is halved if we reduce V_{dd} by 30% without changing any other system parameters. The switching of voltage can be done rapidly with negligible overhead by using efficient DC-DC converters [91]. However, this energy saving comes at the cost of reduced throughput, slower system clock frequency, or higher gate delay. The gate delay is proportional to $\frac{V_{dd}}{(V_{dd}-V_t)^\beta}$, where V_t is the threshold voltage and $\beta \in (1.0, 2.0]$ is a technology dependent constant. Dynamic voltage scaling, which varies the system's operating voltage and clock frequency according to the workload at run-time, can achieve the highest possible energy efficiency for the time-varying computational loads while providing desired performance [18]. It has been demonstrated as one of the most effective low power system design techniques.

In 1996, an actual hardware implementation using voltage scaling was described in [21]. The implementation applies voltage scaling to MPEG video decoding on a DSP. The clock frequency and voltage are adjusted to match the varying complexity of video frames. In [22], a dedicated cryptography processor was presented that uses voltage scaling to reduce the power and energy consumption. The research group in Berkeley Wireless Research Center has developed a dynamic voltage scaled microprocessor system in which the supply voltage and clock frequency can be dynamically varied so that the system can deliver high throughput when required while significantly extending battery life during the low speed periods [17, 18]. The system can dynamically vary the supply voltage from 1.2V to 3.8V in less than 70μs. In [94], they showed DVS can be efficiently integrated into existing operating systems without extensive modification. Pouwelse et al. [96] described a low-power microprocessor system that allows power-aware applications to quickly adjust the performance level of the processor whenever the workload changes. Hong et al. [45] developed a design methodology for the low power core-based real-time system-on-chip based on dynamically variable voltage hardware. Many modern microprocessors, such as Transmeta's Crusoe, AMD's K-6, Intel's XScale and Pentium III and IV, can support dynamic voltage scaling.

Early research on voltage scaling was on systems that have multiple simultaneous available voltages [24, 58, 72, 106]. For example, Raje and Sarrafzadeh [106] presented a supply voltage driven technique to minimize power consumption at the behavioral level. They used data flow graph to define systems and exploited the parallelism evident among all of operations. Some operations can be slowed down by applying a smaller supply voltage that reduces the system energy consumption and the system throughput still meets the given time constraint. Such variable voltage systems can be more energy-efficient because of the flexibility of choosing the operating voltage and clock frequency [100]. In [65, 66], the authors propose the energy efficient synthesis techniques for datapath circuits using dynamic frequency clocking and multiple voltages.

There are a lot of research on task-level scheduling strategies for adjusting CPU speed and supply voltage so as to reduce power and energy consumption of the systems. A scheduling method to reduce energy consumption by dynamically changing the clock speed along with the supply voltage of the processor was first proposed in [125] and was later extended in [39]. The voltage scheduling algorithms proposed in [39, 125] have been improved with PACE (Processor Acceleration to Conserve Energy), an approach to reducing the energy consumption of dynamic voltage scheduling (DVS) algorithms without affecting their performance [77]. The foundation for the simulation and analysis of DVS algorithms can be found in [93]. The above works [39, 77, 93, 125] are in the context of non-real-time workstation environment. Furthermore, Qu et al. [103] combined the variable voltage scaling with variable size packet fragmentation to minimize the power consumption in system-level pipelines under latency constraints.

Recently, many research groups have investigated the DVS problem for hard real-time systems [41, 46, 47, 64, 117, 128]. Yao, Demers and Shenker [128] have provided the minimum-energy preemptive static scheduling algorithm for a set of independent tasks with arbitrary arrival times and deadlines. They assumed that tasks are scheduled according to the earliest-deadline-first (EDF) scheduling policy [73]. In the same paper [128], the authors also proposed two on-line scheduling heuristics, called Average Rate Heuristic (AVR) and Optimal Available Heuristic, with the same model as in the static version. They showed that for the power function $P(s) = s^p (p \geq 2)$, the AVR has a constant competitive ratio r^p satisfying $p^p \leq r^p \leq 2^{p-1}p^p$. Hong etc. developed the non-preemptive offline variable voltage scheduling heuristic with the assumption of zero delay in changing voltage levels [45]. In [47], they focused on the preemptive variable voltage scheduling heuristic while taking into account the inherent limitation on the rates at which voltage and clock frequency can be changed by the power supply controllers and clock generators. The same group also describes an on-line scheduling algorithm for hard real-time tasks on variable voltage processor, where it is assumed that the release times of tasks are not known a priori [46]. Ishihara and Yasuura [56] presented some significant theorems for voltage scheduling and formulate the static voltage scheduling problem as an integer linear programming (ILP) problem. Recently, Quan and Hu [105] studied the problem of determining the optimal voltage schedule for a real-time system with fixed-priority jobs implemented on a variable voltage processor based on the assumption that the timing parameters of each job is known offline. Manzak and Chakrabarti [85] proposed variable voltage task scheduling algorithms (periodic as well as aperiodic) that minimize energy. Pillai and Shin [95] presented a class of algorithms called real-time DVS (RT-DVS) that modify the OS's real-time scheduler and task management service to provide significant energy savings while maintaining real-time deadline guarantees. The algorithms have been verified through simulations and a working prototype implementation.

Reducing power and energy consumption of processors is fundamentally equivalent to exploiting the idle intervals or slacks of processors [118]. Offline voltage scheduling algorithms [45, 56] use the worst-case execution time (WCET), which can be obtained through static analysis [71], profiling, or direct measurement, as one of the timing parameters for each of the tasks. However, the execution time of each task frequently deviates from its WCET, sometimes by a large amount. In order to exploit the slacks arising from the run-time variation of each task execution, the on-line voltage scheduling algorithms need to be applied with the offline algorithms in order to achieve more energy saving [41, 46, 64, 67, 117, 118]. Krishna and Lee [64] presented cyclic scheduling algorithm and EDF scheduling algorithm that is voltage-clock scheduling for the EDF (Earliest Deadline First) algorithm. Both algorithms consist an offline phase, in which voltage settings are picked to reduce energy consumption assuming that tasks run to their WCETs, and an on-line phase that adjusts the voltage setting on-the-fly to reclaim any slacks released by the tasks which actual execution time are less than their WCETs, thus making for a further round of energy saving. Shin and Choi [117] presented a power efficient version of a widely used fixed priority scheduling method. The method yields a power reduction by exploiting the slack times inherent in the system and those arising form variations of execution time task instances. The same authors also showed that combined offline and on-line components bring about more power saving [118]. In [41] Gruian also addressed scheduling for reduced energy of hard real-time tasks with fixed priorities assigned in a rate monotonic (RM) [73] or deadline monotonic (DM) [6]. Taking into account the real behavior of a real-time system, which is often better than the worst case, his method employs stochastic data to derive energy efficient schedules that are combined with on-line slack distribution to achieve energy reduction. Kumar and Srivastava [67] presented a power-saving prediction strategy that exploits the fixed priority scheduling of the real-time tasks running on the embedded systems. But there is a penalty of tasks missing their deadlines. Pouwelse, Langendoen and Sips [97] described the energy priority scheduling (EPS) heuristic and show that by requiring applications to be power aware (i.e. they must specify their future demands) much better energy reduction can be achieved while still meeting all deadlines.

Most of papers introduced so far focus on the DVS problem on task-by-task basis on a single processor. In recent several years there are many research works that concentrate on intra-task voltage scheduling or multiple processor voltage scheduling.

Intra-task voltage scheduling [116] that adjusts the supply voltage within individual task boundary may not involve operating system (OS) in adjusting the clock speed, so it has an advantage that existing OS can be used without any modifications on a variable voltage processor. Lee and Sakurrai [70] presented a novel run-time dynamic voltage scaling scheme for low-power real-time systems that fully exploits slack time arising from task execution time variation and reduces the energy consumption significantly. It partitions a task into several timeslots and performs run-time software feedback control of supply voltage on timeslot-by-timeslot basis. Shin, Kim and Lee proposed an intra-task voltage scheduling algorithm, which controls the supply voltage within an individual task boundary [116]. The proposed algorithm makes the voltage scaling decisions in compile time, not run time, and allows programmers with no knowledge on DVS to develop low-energy hard real-time applications. This idea can also be found in [90] that integrated compiler-assisted techniques with power aware operating system services and presented scheduling techniques to reduce energy consumption of applications that have deadlines. The difference between [116] and [90] is that [90] considers an embedded system with a single application that is divided into n sections or tasks, while [116] considers single-task applications or multi-task applications where one task is dominant in total execution time. Most recently, Dudami et al. [31] presented the energy-conserving feedback EDF scheduling for embedded systems with real-time constraints to exploit slack time generated by the invocation of the task at multiple frequency levels within the same invocation.

Tasks in real-world applications usually have control or data dependencies and many systems have multiple processors. Approaches in [8, 9, 42, 79, 80, 81, 113, 114] solve the energy minimization problem for dependent tasks on multiple variable voltage processors. Schmitz and Al-Hashimi [113] presented an efficient algorithm for voltage scaling of a distributed embedded system considering variations in the power dissipation among processes and inter process communications. The same authors [114] investigated the problem of considering DVS processing elements (DVS-PEs) power variations dependent on the executed tasks, during the synthesis of distributed embedded systems and its impact on the energy savings. Gruian and Kuchcinski [42] assumed a given task assignment and introduced a new scheduling approach, LEneS, that uses list-scheduling and a special priority function to derive static schedules with low energy consumption. Given a task scheduling Luo and Jha [79] presented a power-conscious algorithm for jointly scheduling multi-rate periodic task graphs and aperiodic tasks in real-time multiprocessor embedded systems in order to improve the response times of soft aperiodic tasks and reduce the power. The same authors have extended their work by using static and dynamic variable voltage scheduling algorithms to exploit the slacks more efficiently and achieve more energy saving [81]. In [80] they proposed static battery-aware scheduling algorithms in battery-powered distributed real-time embedded systems to increase the battery lifespan. Bambha and Bhattacharyya et al. have examined voltage scaling for multiprocessors under known computation time to reduce the overall power consumption under a given throughput constraint. The schedule of tasks on different processors is assumed to be known a priori. In [8], they proposed a local search approach for static voltage scaling based on the period graph model [10]. The same group presented a hybrid global/local search optimization framework for DVS in embedded multiprocessor systems [9]. They applied the simulated heating [131] approach to control parameterized local search such as hill climbing or Monte Carlo within a global search process in order to attain high search efficiency. Zhu et al. [130] introduced the concept of slack sharing on multiprocessor systems to reduce energy consumption. Based on this concept, they proposed two power-aware algorithms GSSR (global scheduling with shared slack reclamation) and LSSR (fixed-order list scheduling with shared slack reclamation) and simulation results showed the scheduling algorithms result in substantial energy saving compared to static power management. The essence of the above works is to utilize the slacks to allow voltage scaling to reduce power consumption without suffering any performance degradation. In [74], a power-aware scheduling algorithm was presented for mission-critical embedded systems with variable power constraints and heterogeneous power consumers, as well as different energy sources such as a non-rechargeable battery and a solar panel. It satisfies the min/max timing and max power constraints. In addition, it also tries to satisfy the minimum power constraint in order to fully utilize free power or to control power jitter. Mishra et al. [88] proposed a static power management algorithm (SPM) with considering processor parallelism for distributed real-time systems. They claimed that their algorithm is better than other existing algorithms such as simple SPM and Greedy SPM in terms of energy saving [88].

Finally, Zhang, Hu and Chen [129] presented a two-phase framework that integrates task assignment, ordering and voltage scaling (VS) together to minimize energy consumption of real-time dependent tasks executing on a given number of variable voltage processors. In the first phase, they applied an EDF scheduling that can be proved to be optimal for a single processor, and a scheduling with priority-based task ordering and a best-fit processor assignment for multiple processors. In the second phase, they formulated the VS problem as an Integer Programming (IP) problem and solve the IP efficiently. Schmitz et al. [115] presented a two-step iterative synthesis approach for distributed embedded systems containing dynamic voltage scalable processing elements by employing two nested genetic algorithms, where the outer GA generates the assignments and the inner one creates various orderings. This algorithm is not however efficient in terms of run time. Recently, Luo and Jha [82] presented an efficient algorithm, which performs execution order optimization of scheduled events, power-profile and timing-constraint driven slack allocation to minimize the power consumption for heterogeneous distributed real-time embedded systems.

1.3 Quality of Service (QoS)

With the increasing popularity of real-time multimedia and wireless communication applications, quality of service (QoS) attracts a lot of attention. Providing the required QoS guarantees becomes vital for the design of embedded systems that carry out such applications. For many embedded systems applications such as distributed multimedia applications, the QoS requirements can be assessed in terms of users' subjective wishes or satisfaction with the quality of the applications-performance, synchronization, cost, and so forth [124]. The assessment results will be mapped onto the constraints of QoS parameters such as processor completion ratio, network throughput, delay, jitter and reliability etc. for various system components or layers [120]. The QoS parameters of the operating systems (OS) in the end systems can have a strong impact on the QoS the users eventually perceive.

Various QoS requirements, such as bounded delay, minimal throughput, guaranteed synchronization or resolution, task completion ratio, were first addressed in the network and real-time operating systems (RTOS) communities. Lawrence [68] presented a QoS Model that is defined by three attributes: timeliness, precision and accuracy. These attributes can be used for system specification, instrumentation, and evaluation. Altmann and Varaiya [5] defined QoS as a combination of the basic quality metrics for the network layer: delay, jitter, bandwidth and reliability. Wijesekera and Srivastava [127] presented quality of service metrics for continuity and synchronization specifications in continuous media. The most formally sound and practically relevant QoS model based on the demand curve and the service curve in the networking community was proposed by Cruz [28]. Based on service curves, Sariowan and Cruz etc. [111] propose a new scheduling policy SCED(Service Curve-based Earliest Deadline first) that guarantees the service curve for the connection in Virtual Circuit Switched Networks. The main conceptual result in RTOS literature, i.e., Q-RAM (QoS-based Resource Allocation Model), was presented by Rajkumar et al. [107]. They introduced an analytic approach for satisfying multiple QoS dimensions under a given set of resource constraints. They showed that the problem is NP-hard and developed an approximation polynomial algorithm for the problem by transforming it into a mixed integer programming problem [108]. Lee et al. [69] presented a QoS management framework to analytically allocate resources for QoS optimization in systems that must satisfy application needs along multiple QoS dimensions for given relations between QoS dimensions and resources. Comprehensive survey of QoS research can be found in [7, 124].

Task completions [11, 19], deadline miss-ratio [78], and loss-rate [126] have been widely used as the measurement of QoS particularly for overloaded and real-time systems in both academic and industry [3]. Baruah et al. [11] studied how to maximize task completions for overloaded systems. They concluded that any online algorithm may perform arbitrarily poorly as compared to a clairvoyant scheduler, but discussed competitive online schedulers for a few special cases such as Equal Request Times, Equal Execution Times, Monotonic Absolute Deadlines and Equal Relative Deadlines. Mittal et al. [89] proposed integrated dynamic scheduling algorithms for hard and QoS degradable tasks, represented by the workload models such as imprecise computation [76] and the (m,k)-firm guarantee [109] that quantify the trade-off between schedulability and result quality, in multiprocessor real-time systems. The proposed algorithms improve schedulability by exploiting the properties of these models in QoS degradation.

Recently QoS-driven system design also received attention from EDA (Electronic Design Automation) community, in particular embedded system design automation and low power system design [86, 98, 99, 102]. Qu et al. studied system synthesis for synchronous multimedia applications, where they focused on how to minimize the chip size while providing synchronization guarantees [99]. The same authors later showed how to use dynamic voltage scaling technique to provide guaranteed QoS with the minimal energy consumption [102]. Kornegay et al. [63] outlined foundations and framework in which QoS system design trade-offs and optimization can be addressed. They concluded by identifying and discussing the future directions related to synthesis of QoS-sensitive systems. Qiu et al. modeled the power-managed multimedia system with QoS guarantees as a generalized stochastic Petri nets and used linear programming formulation to find the most energy-efficient solution [98]. Marculescu et al. presented a new methodology for system-level power and performance analysis based on the product of power and delay of wireless multimedia systems [86].

1.4 Contribution of this Dissertation

This dissertation mainly focuses on how to reduce the energy consumption of soft real-time embedded systems by employing dynamic voltage scaling while still delivering the user required quality of service. Fig. 1.1 shows the overview of my Ph.D. research.

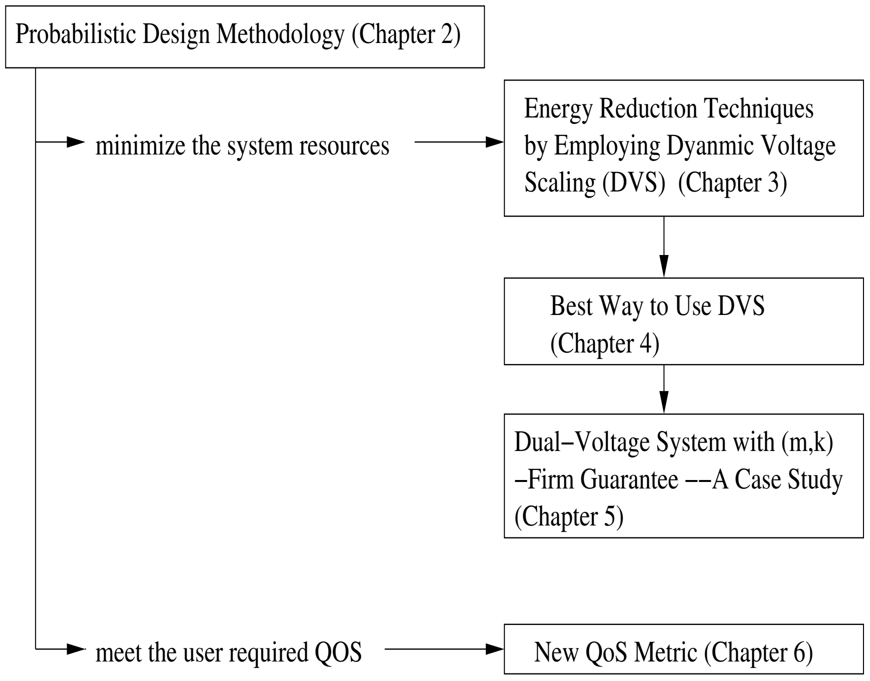

Probabilistic Design Methodology (Chapter 2)
minimize the system resources
Energy Reduction Techniques by Employing Dyanmic Voltage Scaling (DVS) (Chapter 3)
Best Way to Use DVS (Chapter 4)
Dual−Voltage System with (m,k) −Firm Guarantee −−A Case Study (Chapter 5)
meet the user required QOS
New QoS Metric (Chapter 6)

Figure 1.1: The overview of my Ph.D. research.

The main contributions of this dissertation are as follows:

- we have proposed a new design methodology, i.e. probabilistic design, for soft real-time embedded systems in order to reduce the system resources while meeting the user required quality of service statistically [54]. One important phase in the probabilistic design flow is offline/on-line resource management. By using energy/power as an example of resources, we have developed a set of system power management techniques by using dynamic voltage scaling to exploit the slack arising from the tolerance to deadline misses in both single and multiple processor systems [52, 53].

- In order to find the best way to use dynamic voltage scaling, we have first formulated the voltage set-up problem and presented practical solutions to this problem in order to minimize the energy consumption of multiple voltage DVS system in the system level [49]. This is a novel extension under current DVS research framework. We have also conducted the case study of provably most energy efficient voltage set-up for dual voltage system with (m,k)-firm deadline guarantee.

- As the traditional completion ratio metric can only be applied to independent tasks, we have proposed a new quality of service (QoS) metric to capture the different deadline types (firm or soft) and task dependency as well [50]. Furthermore, we have developed a set of low run-time overhead on-line scheduling algorithms to improve QoS and more importantly, to enhance quality of presentation (QoP) significantly with no extra hardware [51].

1.5 Organization of this Dissertation

The remainder of this dissertation is organized as follows.

In Chapter 2, we propose the novel concept of probabilistic design for soft real-time systems and a methodology to quickly explore such design space at an early design stage. The two important phases in the probabilistic design flow, i.e. estimating the probabilistic timing performance and managing system resources with probabilistic performance guarantee, are discussed in detail. The method takes advantage of soft real-time system's unique features (e.g., tolerance for occasional deadline misses, uncertainties in actual execution time) to relax the rigid hardware requirements for software implementation and eventually avoid over-designing the system.

In Chapter 3, we use energy as the example of system resources to explain how to conduct offline/on-line resource management with quality of service guarantee. Specifically, we developed a set of voltage scheduling techniques by taking the tolerance to deadline misses into account in conjunction with the modest non-determinism in application's execution time. First, we give a simple best-effort approach that achieves the maximum completion ratio; then we propose an enhanced on-line best-effort energy minimization ($BEEM$) approach and a hybrid offline/on-line completion ratio Q guaranteed energy minimization($QGEM$) approach. Simulation results show that significant energy savings for both single and multiple processor systems can be achieved while probabilistically meeting the completion ratio requirements.

In Chapter 4, in order to find the best way to employ DVS, we formulate the voltage set-up problem and provide the practical solutions to minimize the system's energy consumption. Voltage set-up problem is *how many levels and at* *which values should voltage be implemented for the multiple-voltage DVS system to achieve the maximum energy saving.* It challenges whether DVS technique's full potential in energy saving can be reached on multiple-voltage systems. In this chapter, (1) we derive analytical solutions for dual-voltage system. (2) For the general case that does not have analytic solutions, we develop efficient numerical methods. (3) We demonstrate how to apply the proposed algorithms on system design. (4) Interestingly, the experimental results suggest that multiple-voltage DVS systems, when the voltages are set up properly, can be very close to DVS technique's full potential in energy saving.

In Chapter 5, as a case study we discuss how to design dual-voltage soft real-time systems with (m,k)-firm guarantee for energy efficiency. We first propose an on-line greedy deterministic scheduler that provides the (m,k)-firm guarantee with the provably minimum energy consumption. We then develop a novel exact method to compute the scheduler's average energy consumption per iteration. This leads us to the numerical solution to the voltage set-up problem, which seeks for the values of the two supply voltages to achieve the most energy efficiency with (m,k)-firm guarantee. Simulation results show that dual-voltage system can reduce significant amount of energy over single voltage system. Our numerical method finds the best voltage set-ups in seconds, while it takes hours to obtain almost identical solutions by simulation.

In Chapter 6, we propose a new quantitative QoS metric based on task completion ratio while differentiating firm and soft deadlines and taking task dependency into consideration. Using the decoding of simulated MPEG movies as an example, we show that the proposed QoS metric is much better than completion ratio in measuring the quality of presentation (QoP) of the movies. However, when replacing the completion ratio by the new QoS metric, popular online algorithms, such as Earliest Deadline First (EDF) and Least Execution Time First (LETF), give only limited improvement on QoP. Therefore, we develop a set of online schedulers with low overhead to enhance QoP significantly, particularly when the system is overloaded.

Chapter 7 concludes this dissertation with a summary of my Ph.D. research work. Some possible directions for future work are also provided.

Chapter 2

Probabilistic Design Methodology

2.1 Introduction

Soft real-time embedded systems such as multimedia embedded systems are widely used in a lot of areas such as movies, education, entertainment, teleconferencing and information service. These systems require the processing of signal, image, and video data streams in a timely fashion to the end user's satisfaction. Such applications are often characterized by the repetitive processing on periodically arriving inputs, such as voice samples or video frames, and the tolerance to occasional deadline misses without being noticed by human visual and auditory systems. The deadline can be (implicitly) determined by the throughput requirement of the input data streams. For example, in packet audio applications, loss rates between 1% - 10% can be tolerated [15], while tolerance for losses in low bit-rate voice applications may be significantly lower [60]. Furthermore, in many multimedia DSP applications, although the execution time of a task can vary dramatically due to a number of factors such as cache miss(es) or conditional branches, it is possible to obtain the execution time distribution for each task by knowing (e.g., by sampling technique) detailed timing information about the system or by profiling the target hardware [122].

Prior design space exploration methods for hardware-software codesign of embedded systems, e.g., [32, 44, 83], guarantee no deadline missing by considering worst case execution time (WCET) of each task. As the soft real-time embedded systems can tolerate some violations of timing constraints, these methods will often lead to over-designed systems that deliver higher performance than necessary at the cost of expensive hardware, higher energy consumption, and other system resources.

There are plenty of studies on the estimation of soft real-time system's probabilistic performance when the application's computation time can be varied [48, 59, 122]. However, their goals are to improve system's performance or to provide probabilistic performance guarantees. To the best of our knowledge, there is no reported effort on systematically incorporating application's performance requirements, uncertainties in execution time, and tolerance for reasonable execution failures to guide rapid and economic design of real-time embedded systems.

In this chapter, we study the problem of how to integrate such tolerance to deadline misses into the design of soft real-time embedded systems. We propose the novel concept of probabilistic design for soft real-time embedded systems and a methodology to quickly explore such design spaces at an early design stage. Given the execution time distribution of each task and the tolerance to deadline misses (measured by the quantitative completion ratio), we have developed a set of algorithms to estimate the probabilistic timing performance and to manage system resources in such a way that the system achieves the required completion ratio probabilistically with a reduced amount of system resources. This method relaxes the rigid hardware requirements for software implementation to meet the WCET and eventually avoids over-designing the soft real-time embedded systems. In the next chapter we will use system's energy consumption, one of the most critical resources for soft real-time embedded systems, as an example to demonstrate how our approach can lead to significant energy-efficient designs.

The rest of the chapter is organized as follows: Section 2.2 describes the related work in design space exploration, performance analysis, and low power design techniques. Section 2.3 gives the overview of our probabilistic design space exploration methodology. Our method has two key steps, i.e., the probabilistic timing performance estimation, which is discussed in Section 2.4, and the offline/on-line resource management with the probabilistic performance guarantee, which is introduced in Section 2.5. We conclude the paper in Section 2.6.

2.2 Related Work

The most relevant work is on design space exploration and performance analysis, probabilistic performance estimation, and scheduling techniques for low power.

An integrated hardware-software codesign system should support design space exploration with optimization [33]. There are several works on performance analysis for design space exploration based on monoprocessor architecture. In PMOSS [32], the authors presented a methodology for rapid analysis, synthesis and optimization of embedded systems by providing modularity. Henkel and Ernst [44] have presented high-level estimation techniques for the hardware effort and hardware/software communication time. They claimed that the proposed techniques are well suited for fast design space exploration. In the LYCOS system [83], the authors used profiling techniques and evaluations of low-level execution time for hardware, software and communication to estimate the system performance. For the rapid prototyping of hardware-software codesigns, Chatha and Vemuri [25] introduced their performance evaluation tool to provide fast and accurate performance estimates based on profiling and scheduling. However, all of the above works specify the deadline as one of the design constraints that has to be met.

There are several papers on the probabilistic timing performance estimation for soft real-time systems design [48, 59, 122]. The general assumption is that each task's execution time can be described by a discrete probability density function that can be obtained by applying path analysis and system utilization analysis techniques [84]. In [122], the authors extended the scheduling algorithms and schedulability analysis methods developed for periodic tasks in order to provide probabilistic performance guarantee for semi-periodic tasks when the total maximum utilization of the tasks on each processor is larger than one. They described the transform-task method that transforms each semi-periodic task into a periodic task followed by a sporadic task. The method can provide an absolute guarantee for requests with shorter computation times and a probabilistic guarantee for longer requests. In [59], a performance estimation tool that outputs the exact distribution of the processing delay of each application was introduced. It can help the designers develop multimedia networked systems requiring soft real-time guarantees in a cost efficient manner. Given that the execution time of each task is a discrete random variable, Hu *et al.* [48] proposed a state-based probability metric to evaluate the overall probabilistic timing performance of the entire task set. Their experimental results show that the proposed metric reflects well the timing behavior of systems with independent and/or dependent tasks. However, their evaluation method becomes very time consuming when the task has many different execution time values.

Low power consumption is one of the most important design objectives. Power is proportional to the square of the supply voltage, therefore, reducing supply voltage can result in great power saving. Dynamic voltage scaling (DVS), which varies the clock frequency and supply voltage according to the workload at run-time, can achieve the highest possible energy efficiency for time-varying computation load [18]. For the literature review of DVS research, one can check the Section 1.2 in this dissertation.

2.3 Probabilistic Design Methodology Overview

Many design methods have been developed based on WCET to meet the timing constraints without any deadline misses. However, the actual execution time of each task frequently deviates from its WCET, sometimes by a large amount. Therefore these methods are pessimistic and are suitable for developing systems in a "hard real-time" environment, where any deadline miss will be catastrophic. However, there are also many "soft real-time" systems, such as multimedia systems, which can tolerate occasional deadline misses. The above pessimistic design methods can't take advantage of this feature and will often lead to over-designed systems. In order to avoid over-designing systems, we propose the concept of "probabilistic design" where we design the system to meet the timing constraints of periodic applications statistically. That is, the system may not guarantee the completion of each execution or iteration, but it will produce sufficiently many successful completions over a large amount of iterations to meet the user-specific completion ratio. Or even better, the probability that any execution will be completed is not lower than the desired completion ratio.

Clearly, the proposed "probabilistic design" will be preferred for many embedded systems such as portable multimedia systems where high portability, low power consumption, and reasonably good performance are equally important. However, the corresponding "probabilistic design space" becomes larger than the above mentioned pessimistic design space because it includes designs that fail some iterations while still meeting the desired completion ratio requirement statistically. This increases the design complexity and makes early design space exploration difficult. The "probabilistic design" will thrive only when designers can quickly explore the larger probabilistic design spaces.

Figure 2.1 depicts our probabilistic design space exploration approach for rapid and economic multimedia system design. We start with the popular dataflow graph representation of the embedded software, the system's performance requirements (in terms of timing and completion ratio constraints), and a pool of target system architectures to select from. We partition the application into a set of tasks and use profiling tools to collect detailed execution information of each task. Next, we estimate the system timing performance to check whether it is feasible for the current system configuration to achieve the desired performance. If not, we change the hardware configuration and/or apply software optimization techniques and update the software profiling results that will be used in the next round of system timing performance estimation. We mention that any change on the target hardware configuration and/or software optimization may affect the application's actual execution information and therefore the software profiling process needs to be re-started. This iterative design loop terminates when all the design requirements are met.

Once the completion ratio constraint can be met, we move on to the phase of

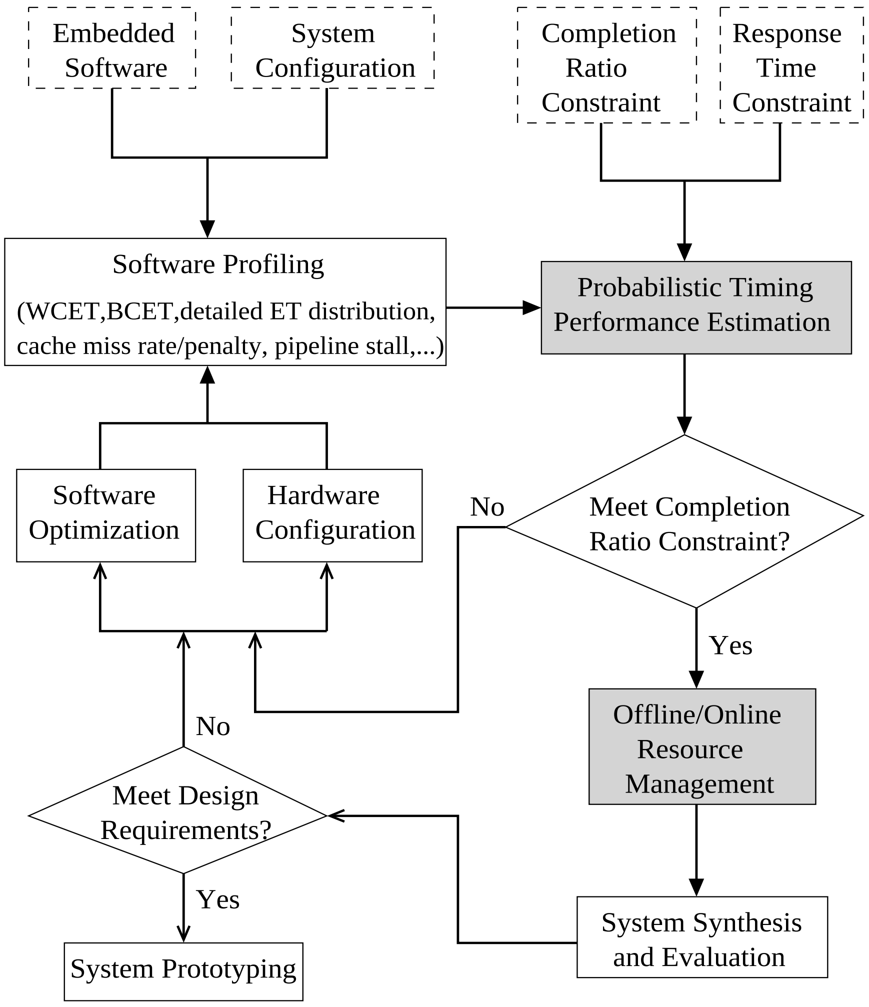

Embedded Software
System Configuration
Completion Ratio Constraint
Response Time Constraint
Software Profiling
(WCET,BCET,detailed ET distribution, cache miss rate/penalty, pipeline stall,...)
Probabilistic Timing Performance Estimation
Software Optimization
Hardware Configuration
No
Meet Completion Ratio Constraint?
Yes
Offline/Online Resource Management
No
Meet Design Requirements?
Yes
System Prototyping
System Synthesis and Evaluation

Figure 2.1: Design flow in the probabilistic design methodology.

offline/on-line resource management. This is the key step in the proposed probabilistic design where we 1) allocate minimum system resources to each task offline in order to make the desired completion ratio probabilistically achievable, and 2) develop real time schedulers to manage the resources at run time such that the required completion ratio can be achieved probabilistically. Finally, we conduct system synthesis, simulation, and evaluation before prototyping the system.

2.4 Estimating the Probabilistic Timing Performance

In order to determine whether a given system implementation can meet the desired completion ratio constraint, we need to estimate the system's probabilistic timing performance. Specifically, in this step we calculate the upper bound of the completion ratio that the system with current configuration can achieve to help us in exploring the probabilistic design space efficiently.

We consider the *task graph* $G = (V, E)$ for a given application. V is the set of vertices in the graph that represent the task computations and E is the the set of directed edges that represent the data dependencies between vertices. We adopt the assumption that the execution time of each vertex can be described by a discrete probability density function [48, 122]. Specifically, for each vertex v_i, we associate a finite set of possible execution times $\{t_{i,1}, t_{i,2}, \cdots, t_{i,k_i}\}$ (under a reference system configuration) and the set of probabilities $\{p_{i,1}, p_{i,2}, \cdots, p_{i,k_i} | \sum_{l=1}^{k_i} p_{i,l} = 1\}$ that such execution times will occur at run-time. That is, with probability $p_{i,j}$, vertex v_i requires an execution time of $t_{i,j}$. Such statistics on task's execution time can be obtained by profiling tools.

The *completion time* of the task graph G (or equivalently the given application) under a fixed execution order $< v_1 v_2 \cdots v_n >$, is the sum of each vertex's run-time execution time e_i: $C(< v_1 v_2 \cdots v_n >) = \sum_{i=1}^{n} e_i$. The *deadline* constraint $\mathcal{D}$ specifies the maximum time allowed to complete the application. The application (or its task graph) will be executed periodically and its period is no less than the deadline $\mathcal{D}$. We say that an iteration is *successfully completed* if $C(< v_1 v_2 \cdots v_n >) \leq \mathcal{D}$. The performance requirement is measured by a real-valued completion ratio $\mathcal{Q}_0 \in [0, 1]$, which is the minimum ratio of completions that the system has to maintain over a sufficiently large number of iterations. For the hard real-time system, $\mathcal{Q}_0 = 1$; and for the soft real-time system, $\mathcal{Q}_0 < 1$. Let K be the number of successfully completed iterations over a total of $N >> 1$ iterations, the actual completion ratio can be denoted by $\mathcal{Q} = \frac{K}{N}$. We say that the completion ratio constraint is *achievable* if $\mathcal{Q} \geq \mathcal{Q}_0$.

For a given system configuration, let t'_{i,j_i} be the time to execute task v_i that requires an execution time t_{i,j_i} under the reference configuration, where $j_i \in \{1, 2, \cdots, k_i\}$, we have a completion if the completion time is less than the deadline, that is, $\sum_{i=1}^{n} t'_{i,j_i} \leq \mathcal{D}$. The probability that this occurs is $\prod_{i=1}^{n} p_{i,j_i}$. Therefore, we have

Theorem 2.1. The maximum achievable completion ratio is given by:

$$\mathcal{Q}^{max} = \sum_{\sum_{i=1}^{n} t'_{i,j_i} \leq \mathcal{D}} \prod_{i=1}^{n} p_{i,j_i} \tag{2.1}$$

where the sum is taken over the execution time combinations that meet the deadline constraint $\mathcal{D}$ and the product computes the probability each such combination happens.

This is similar to the state-based feasibility probability defined in [48]. $\mathcal{Q}^{max}$ helps us to quickly explore the probabilistic design space. Specifically, if $\mathcal{Q}^{max} <$

$\mathcal{Q}_0$, which means that the completion ratio requirement is not achievable under current system configuration, we can make the early and correct decision to reconfigure the hardware or optimize the software implementation rather than further investigating the current system configuration.

The drawback of this estimation is that Equation (2.1) is computationally expensive particularly when there are many tasks and each task has multiple execution times. For example, a task graph with 50 vertices and each vertex having only the best, average, and worst case execution time yields 3^{50} different execution time combinations! Due to the importance of determining whether the required $\mathcal{Q}_0$ is achievable in designing fast probabilistic design space exploration techniques, we have developed the following polynomial heuristic.

Assuming that the task's execution times under the reference configuration are ordered such that $t_{i,1} < t_{i,2} < \cdots < t_{i,k_i}$, we define the prefix sum of the occurrence probability

$$P_{i,l_i} = \sum_{j=1}^{l_i} p_{i,j} \tag{2.2}$$

which measures the probability that the computation at vertex v_i is not longer than t_{i,l_i}. If we allocate time t_{i,l_i} to task v_i and drop the iteration if its actual execution time is longer, then we achieve a completion ratio

$$\mathcal{Q} = \prod_{i=1}^{n} P_{i,l_i} = \prod_{i=1}^{n} \sum_{j=1}^{l_i} p_{i,j} \tag{2.3}$$

We use a greedy approach to estimate whether completion ratio $\mathcal{Q}_0$ can be achieved within the deadline $\mathcal{D}$. First, we assign each vertex its WCET. This yields $\mathcal{Q} = 1$ but the total assigned completion time $\sum_{i=1}^{n} t_{i,k_i}$ will most likely exceed the deadline constraint. From Equation (2.3), if we cut the time slot of vertex v_i from t_{i,l_i} to $t_{i,(l_i-1)}$, the completion ratio will be reduced by the factor of $\frac{P_{i,(l_i-1)}}{P_{i,l_i}}$ and the total assigned time will be reduced by $t_{i,l_i} - t_{i,(l_i-1)}$. We iteratively cut the time slot of vertex v_j that yields the largest $(t_{j,l_j} - t_{j,(l_j-1)}) \cdot \frac{P_{j,(l_j-1)}}{P_{j,l_j}}$ as long as it gives a completion ratio larger than $\mathcal{Q}_0$. This greedy selection approach frees more assigned time slot at the minimum level of completion ratio reduction. When we cannot reduce the completion ratio any further and the total assigned time $\sum_{i=1}^{n} t_{i,l_i}$ is not larger than the deadline $\mathcal{D}$, our heuristic will conclude that the required Q_0 is achievable. Otherwise, it will report that Q_0 cannot be guaranteed, even though in some cases $\mathcal{Q}^{max}$ is actually larger than $\mathcal{Q}_0$. The complexity of the proposed heuristic is $O(n^2)$.

2.5 Managing System Resource under Probabilistic Performance Constraint

When $\mathcal{Q}^{max} \geq \mathcal{Q}_0$, it becomes theoretically possible to deliver the probabilistic performance guarantee (in terms of completion ratio) with the current system configuration. The resource management phase in our design space exploration aims to reduce the design cost. Specifically, we are given a task graph corresponding to the application that includes a set of vertices (tasks), each of which represents certain computation, and a set of directed edges, each of which represents data dependency. Each task has a finite set of possible execution time, which can be obtained by profiling or simulation on target hardware. Based on the above information together with the deadline constraint $\mathcal{D}$ and user required QoS Q_0, we want to 1) determine the minimum system resource required to provide the probabilistic performance guarantee; and 2) develop on-line scheduling algorithms to guide the system to achieve such guarantee at run time with the determined minimum resource.

As energy consumption has emerged as one of the most important concerns in the design of embedded systems particularly for the battery-operated portable systems, in this dissertation we consider energy as one example of resource to manage and present our newly developed offline/on-line energy reduction techniques with completion ratio guarantee. We achieve the energy saving by the dynamic voltage scaling method on multiple supply voltage and multiple threshold voltage system, which has been identified by the International Technology Roadmap for Semiconductors (ITRS) as the trend of future systems [2]. Specifically, we consider the following problem:

> *For a given task graph, its deadline, its completion ratio constraint and the task execution time distribution, find a scheduling strategy for a multiple voltage system such that the resource (e.g., energy) consumed to satisfy the completion ratio constraint is minimized.*

In the next chapter, we will present our recent results on energy/power management methods for both single and multiple processor systems in order to minimize the system energy consumption while meeting user required completion ratio.

2.6 Conclusions

In this chapter we present the novel concept of probabilistic design for soft real-time embedded systems and a methodology to quickly explore such design spaces at the early design stage in order to rapidly achieve economic system design. By taking advantage of soft real-time DSP application's unique features, namely application's performance requirements, uncertainties in execution time, and tolerance for reasonable execution failures, our method systematically relaxes the rigid hardware requirements for software implementation and eventually avoids over-designing the system. There are two key steps in our probabilistic design methodology, which are the probabilistic timing performance estimation and the offline/on-line resource management. In this chapter, we have introduced our heuristic method to rapidly estimate the probabilistic timing performance. In the next chapter, we will show how to design soft real-time embedded systems with reduced resource (energy consumption in our case) while providing the desired performance (completion ratio) statistically.

Chapter 3

Energy Reduction Techniques for Single and Multiple Processor Systems

In Chapter 2, we propose a probabilistic design methodology to avoid over-designing the systems. One important step in the design flow (see Fig. 2.1) is offline/on-line resource management. In this chapter, by using energy as one example of resource, we show how to reduce the system resource while the system still meets the user required quality of service (completion ratio).

3.1 Introduction

Performance guarantee and energy efficiency are becoming increasingly important for the design of embedded systems. Traditionally, the worst case execution time (WCET) is considered to provide performance guarantee, however, this often leads to over-designing the system (e.g., more hardware and more energy consumed than necessary), We discuss the problem of how to implement single or multiprocessor embedded systems to deliver performance guarantee with reduced energy consumption.

Many applications, such as multimedia and digital signal processing (DSP) applications, are characterized by repetitive processing on periodically arriving inputs (e.g., voice samples or video frames). Their processing deadlines, which are determined by the throughput of the input data streams, may occasionally be missed without being noticeable or annoying to the end user. For example, in packet audio applications, loss rates between 1% - 10% can be tolerated [15], while tolerance for losses in low bit-rate voice applications may be significantly lower [60]. Such tolerance gives rise to slacks that can be exploited when streamlining the embedded processing associated with such applications. Specifically, when the embedded processing does not interact with a lossy communication channel, or when the channel quality is high compared to the tolerable rate of missed deadlines, we are presented with slacks in the application that can be used to reduce cost or power consumption.

Typically, slacks arise from the run-time task execution time variation and can be exploited to improve real-time application's response time or reduce power. For example, Shin and Choi used fixed priority scheduling method to achieve power reduction by exploiting slack times in real-time systems [117]. Bambha and Bhattacharyya examined voltage scaling for multiprocessor with known computation time and hard deadline constraints [8]. Luo and Jha presented a power-conscious algorithm [79] and static battery-aware scheduling algorithms for distributed real-time battery-powered systems [80]. Zhu et al. introduced the concept of slack sharing on multi-processor systems to reduce energy consumption [130]. The essence of these works is to exploit the slacks by using voltage scaling to reduce energy consumption without suffering any performance degradation (execution failures).

The slack we consider in this chapter comes from the tolerance of execution failures or deadline missings. In particular, since the end user will not notice a small percentage of execution failure, we can *intentionally* drop some tasks to create slack for voltage scaling as long as we keep the loss rates to be tolerable. Furthermore, much richer information than task's WCET is available for many DSP applications. Examples include the best case execution time (BCET), execution time with cache miss, when interrupt occurs, when pipeline stalls or when different conditional branch happens. More important, most of these events are predictable and we will be able to obtain the probabilities that they may happen by knowing (e.g. by sampling technique) detailed timing information about the system or by simulation on the target hardware [122]. This gives another degree of freedom to explore online and offline voltage scaling for energy reduction.

Dynamic voltage scaling(DVS), which can vary the supply voltage and clock frequency according to the workload at run-time, can exploit the slack time generated by the workload variation and achieve the highest possible energy efficiency for time-varying computational loads [18, 100]. It is arguably the most effective technique to reduce the dynamic energy, which is still the dominate part of system's energy dissipation despite the fast increase of leakage power on modern systems. The relevant works on DVS can be found in Section 1.2 in this dissertation.

Finally, we mention that early efforts on single and multiple processor embedded system design range from the design space exploration algorithm [61] to the implementation of such systems [40, 121]. And scalable architectures and co-design approaches have been developed for the design of multiprocessor DSP systems (e.g., see [57, 112]). These approaches, however, do not provide systematic techniques to handle voltage scaling, non-deterministic computation time, or completion ratio tolerance. Performance-driven static scheduling algorithms that allocate task graphs to multiprocessors [119] can be used in conjunction with best- or average-case task computation time to generate an initial schedule for our proposed methods. It can then interleave performance monitoring and voltage adjustment functionality into the schedule to streamline its performance.

A Motivational Example

We consider a simple case when a multiple-voltage processor executes three tasks $\mathcal{A}, \mathcal{B}, \mathcal{C}$ in that order repetitively. Table 3.1(a) gives each task's only two possible execution time and the probabilities that they occur. Table 3.1(b) shows the normalized power consumption and processing speed of the processor at three different voltages.

Table 3.1: Characteristics of the tasks and the processor. (a): each entry shows the best/worst case execution time at V_1 and the probability this execution time occurs at run time. (b): *power* is normalized to the power at V_1 and *delay* column gives the normalized processing time to execute the same task at different voltages.

task	BCET	WCET
$\mathcal{A}$	(1, 80%)	(6, 20%)
$\mathcal{B}$	(2, 90%)	(7, 10%)
$\mathcal{C}$	(2, 75%)	(5, 25%)

(a) Three tasks.

voltage	power	delay
$V_1 = 3.3V$	1	1
$V_2 = 2.4V$	0.30	1.8
$V_3 = 1.8V$	0.09	3.4

(b) Processor parameters.

Suppose that each iteration of "$\mathcal{A} \to \mathcal{B} \to \mathcal{C}$" must be completed in 10 CPU units and we can tolerate 40% of the 10,000 iterations to miss their deadlines. We now compare the following three different algorithms:

(I) For each iteration, run at the highest voltage V_1 to the completion or the deadline whichever happens first.

(II) Assign deadline pairs (0,6), (5,8), and (10,10) to $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ respectively. For each task, terminate the current iteration if the task cannot be completed by its second and longer deadline at V_1; otherwise, run at the lowest voltage without violating its first and shorter deadline or run at V_1 to its completion.

(III) In each iteration, assign 1, 7, and 2 (a total of 10) CPU units to $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ respectively. Each task can only be executed within its assigned slot: if it cannot be finished at V_1, terminate; otherwise run at the lowest voltage to completion.

Assuming that the execution time of each task follows the above probability, for each algorithm, we obtain the completion ratio $\mathcal{Q}$, each iteration's average processing time (at different voltages) and power consumption (Table 3.2). We mention that 1) algorithm I gives the highest possible completion ratio; 2) algorithm II achieves the same ratio with less energy consumption; and 3) algorithm III trades unnecessary completion for further energy reduction. Although algorithm I is a straightforward best-effort approach, the settings for algorithms II and III are not trivial: *Why the deadline pairs are determined for $\mathcal{A}$ and $\mathcal{B}$? Is it a coincidence that such setting achieves the same completion ratio as algorithm I? How to set execution slot for each task in algorithm III to guarantee the 60% completion ratio, in particular if we cannot find 80% and 75% whose product gives the desired completion ratio?*

Table 3.2: Expected completion ratio and energy consumption for the three algorithms. $t@V_1$, $t@V_2$, and $t@V_3$ are the average time that the processor operates at three voltages for each iteration; E is the average energy consumption to complete one iteration; and the last column, obtained by $E \cdot 60\%/\mathcal{Q}$, corresponds to the case of shutting the system down once 6,000 iterations are completed.

	$\mathcal{Q}$	$t@V_1$	$t@V_2$	$t@V_3$	E	E@($\mathcal{Q} = 60\%$)
I	91.5%	6.94	0	0	6.94	4.55
II	91.5%	4.21	4.54	0	5.57	3.65
III	60%	2.56	0	4.90	3.00	3.00

In this chapter, i) we first formulate the energy minimization problem with deadline miss tolerance on single and multiple processor (DSP) systems; ii) we then develop on-line scheduling techniques to convert deadline miss tolerances into energy reduction via DVS; iii) this departs us from the conservative view of over-implementing the embedded systems in order to meet deadlines under WCET; iv) our result is an algorithmic framework that integrates considerations of iterative single or multiple processor scheduling, voltage scaling, non-deterministic computation time, and completion ratio requirement, and provides robust, energy-efficient single or multiple processor implementation of embedded systems for DSP applications. In the following sections, we will mainly focus on the energy reduction techniques for multiprocessor embedded systems. One can easily adapt and apply them to single processor embedded system design.

3.2 Problem Formulation

We consider the *task graph* $G = (V, E)$ for a given application. Each vertex in the graph represents one computation and directed edges represent the data dependencies between vertices. For each vertex v_i, we associate it with a finite set of possible execution time $\{t_{i,1} < t_{i,2} < \cdots < t_{i,k_i}\}$ and the corresponding set of probabilities $\{p_{i,1}, p_{i,2}, \cdots, p_{i,k_i} | \sum_{l=1}^{k_i} p_{i,l} = 1\}$ that such execution time may occur. That is, with probability $p_{i,j}$, vertex v_i requires an execution time in the amount of $t_{i,j}$. Note that t_{i,k_i} is the WCET and $t_{i,1}$ is the BCET for task v_i. We then define the prefix sum of the occurrence probability

$$P_{i,l} = \sum_{j=1}^{l} p_{i,j} \tag{3.1}$$

Clearly, $P_{i,l}$ measures the probability that the computation at vertex v_i can be completed within time $t_{i,l}$ and we have $P_{i,k_i} = 1$ which means that a completion is guaranteed if we allocate CPU to vertex v_i based on its WCET t_{i,k_i}.

A directed edge $(v_i, v_j) \in E$ shows that the computation at vertex v_j requires data from vertex v_i. For each edge (v_i, v_j), there is a cost for *inter-processor communication* (IPC) w_{v_i,v_j}, which is the time to transfer data from the processor that executes v_i to a different processor that will execute v_j. There is no IPC cost, i.e. $w_{v_i,v_j} = 0$, if vertices v_i and v_j are mapped to the same processor by the task scheduler. For a given datapath $< v_1 v_2 \cdots v_n >$, its *completion time* is the sum of the execution time at run-time, of each vertex, e_i, and all the IPC costs. That is,

$$C(< v_1 v_2 \cdots v_n >) = e_1 + \sum_{i=2}^{n} (w_{v_{i-1},v_i} + e_i) \tag{3.2}$$

The *completion time* for the entire task graph G (or equivalently the given application), denoted by $C(G)$, is equal to the completion time of its *critical path*, which has the longest completion time among all its datapaths. (Note for the single processor system, there is no IPC cost and $C(G)$ is equal to the sum of the execution time of each vertex in the entire task graph.)

We are also given a *deadline* constraint $\mathcal{D}$, which specifies the maximum time allowed to complete the application. The application (or its task graph) will be executed on a multiprocessor system periodically with its deadline $\mathcal{D}$ as the period. We say that an iteration is *successfully completed* if its completion time, which depends on the run-time behavior, $C(G) \leq \mathcal{D}$. Closely related to $\mathcal{D}$ is a real-valued *completion ratio* constraint(or requirement) $\mathcal{Q}_0 \in [0, 1]$, which gives the minimum acceptable completion ratio over a sufficiently large number of iterations. Alternatively, $\mathcal{Q}_0$ can be interpreted as a guarantee on the probability with which an arbitrary iteration can be successfully completed.

Finally, we assume that there are multiple supply voltage levels available at the same time for each processor in the multi-processor system. This type of system can be implemented by using a set of voltage regulators each of which regulates a specific voltage for a given clock frequency. In this way, the operating system can control the clock frequency at run-time by writing to a register in the system control state exactly the way as in [18] except that the system does not need to wait for the voltage converter to generate the desired operating voltage. In sum we can assume that each processor can switch its operating voltage from one level to another instantaneously and independently with the power dissipation $P \propto CV_{dd}^2 f$ and gate delay $d \propto \frac{V_{dd}}{(V_{dd}-V_{th})^{\alpha}}$ at supply voltage V_{dd} and threshold voltage V_{th}, where $1 < \alpha \leq 2$ is a constant depends on the technology [23]. Furthermore, on a multiple voltage system, for a task under any time constraint, the voltage scheduling with at most two voltages minimizes the energy consumption and the task is finished just at its deadline [100].

In this chapter, we consider the following problem:

> *For a given task graph G with its deadline $\mathcal{D}$ and completion ratio constraint $\mathcal{Q}_0$, find a scheduling strategy for a multi-processor multi-voltage system (a means of (1) assigning vertices to processors, (2) determining the execution order of vertices on the same processor, and (3) selecting the supply voltage for each processor) such that the energy consumption to satisfy the completion ratio constraint $\mathcal{Q}_0$ is minimized.*

It is well-known that the variable voltage task scheduling for low power is in general NP-hard [46, 100]. On the other hand, there exist intensive studies on multi-processor task scheduling problem with other optimization objectives such as completion time or IPC cost [87, 119]. In this chapter, We focus on developing on-line algorithms for voltage scaling (and voltage selection in particular) on a scheduled task graph. That is, we assume that tasks have already been assigned to processors and our goal is to determine *when* and *at which voltage* each task should be executed in order to minimize the total energy consumption while meeting the completion ratio constraint $\mathcal{Q}_0$.

3.3 Energy-Driven Voltage Scaling Techniques with Completion Ratio Constraint

In this section, we first obtain, with a simple algorithm, the best completion ratio on multi-processor system for a given task assignment. We then give a lower bound on the energy consumption to achieve the best completion ratio. Our focus will be on the development of on-line energy reduction algorithms that leverage the required completion ratio, which is lower than the best achievable.

3.3.1 A Naïve Best-Effort Approach

Even when there is only one supply voltage, which results in a fixed processing speed, and each task has its own fixed execution time, the problem of determining whether a set of tasks can be scheduled on a multi-processor system to be completed by a specific deadline remains NP-complete (this is the *multiprocessor scheduling* problem [SS8], which is NP-complete for two processors [36].). However, for a given task assignment and ordering, the highest possible completion ratio can be trivially achieved by simply applying the highest supply voltage on all the processors. That is, each processor keeps on executing whenever there exist tasks assigned to this processor ready for execution;and stops when it completes all its assigned tasks in the current iteration or when the deadline $\mathcal{D}$ is reached. In the latter, if any processor has not finished its execution, we say the current iteration is *failed*; otherwise, we have a *successful completion* or simply *completion*. Clearly this naïve method is a best-effort approach in that it tries to complete as many iterations as possible. Since it operates all the processors at the highest voltage, the naïve approach will provide the highest possible completion ratio, denoted by $\mathcal{Q}^{max}$. In another word, if a completion ratio requirement cannot be achieved by this naïve approach within the given deadline $\mathcal{D}$, then no other algorithms can achieve it either.

When the application-specified completion ratio requirement $\mathcal{Q}_0 < \mathcal{Q}^{max}$, a simple counting mechanism can be used to reduce energy consumption. Specifically we cut the N iterations into smaller groups and shut the system down once sufficient iterations have been completed in each group. For example, if an MPEG application requires a 90% completion ratio, we can slow down the system (or switch the CPU to other applications) whenever the system has correctly decoded

90 out of 100 consecutive frames. This counting mechanism saves total energy by preventing the system from over-serving the application.

For system with multiple operating voltages, we mention that energy could have been saved over the above naïve approach in the following scenarios: i) if we knew that an iteration would be completed earlier than the deadline $\mathcal{D}$, we could have processed with a lower voltage; and ii) if we knew that an iteration cannot be completed and have stopped the execution earlier. To save the maximal amount of energy, we want to *determine the lowest voltage levels to lead us to completions right at the deadline* $\mathcal{D}$ and *find the earliest time to terminate an incompletable iteration.* However, additional information about the task's execution time (e.g. WCET, BCET, and/or the probabilistic distribution) is required to answer these questions. In the rest of this section, we propose on-line voltage scaling techniques to reduce energy with the help of such information.

3.3.2 BEEM: On-Line Best-Effort Energy Minimization

The best-effort energy minimization (BEEM) technique gives the minimum energy consumption on single or multiple processor system to provide the highest achievable completion ratio. Here we propose algorithm BEEM1, which assumes task' execution time are know a priori, and BEEM2, which does not, on a multiprocessor systems. The BEEM technique on a single processor system can be found in [53].

We define the *latest completion time* T_l^v and the *earliest completion time* T_e^v for a vertex v using the following recursive formulas:

$$T_e^v = T_l^v = \mathcal{D} \qquad \text{(if } v \text{ is a sink node)} \tag{3.3}$$

$$T_e^{v_i} = \min\{T_e^{v_j} - t_{j,k_j} - w_{v_i,v_j} | (v_i, v_j) \in E\} \tag{3.4}$$

$$T_l^{v_i} = \min\{T_l^{v_j} - t_{j,1} - w_{v_i,v_j} | (v_i, v_j) \in E\} \tag{3.5}$$

where $t_{j,1}$ and t_{j,k_j} are the BCET and WCET of vertex v_j, w_{v_i,v_j} is the cost of IPC from vertices v_i to v_j which is 0 if the two vertices are assigned to the same processor.

Lemma 3.1. If an algorithm minimizes energy consumption, then vertex v_i's completion time cannot be earlier than $T_e^{v_i}$.

[Proof]: Clearly such algorithm will complete each iteration at deadline $\mathcal{D}$. Otherwise, one can always reduce the operating voltage and processing speed (or adjust the combination of two operating voltages) for the last task to save more energy.

Let t be vertex v_i's completion time at run time. If $t < T_e^{v_i}$, for any path from v_i to a sink node v, $u_0 = v_i, u_1, \cdots, u_k = v$, let $WCET_{u_j}$ be the worst case execution time of vertex u_j, then the completion time of this path will be

$$\begin{aligned}
T &\leq t + \sum_{j=0}^{k-1}(w_{u_j,u_{j+1}} + WCET_{u_{j+1}}) < T_e^{v_i} + \sum_{j=0}^{k-1}(w_{u_j,u_{j+1}} + WCET_{u_{j+1}}) \\
&= T_e^{u_0} + w_{u_0,u_1} + WCET_{u_1} + \sum_{j=1}^{k-1}(w_{u_j,u_{j+1}} + WCET_{u_{j+1}}) \\
&\leq T_e^{u_1} + \sum_{j=1}^{k-1}(w_{u_j,u_{j+1}} + WCET_{u_{j+1}}) \leq \cdots \leq T_e^{v} = \mathcal{D}
\end{aligned}$$

This implies that even when the WCET happens for all the successor vertices of v_i on this path, the completion of this path occurs before the deadline $\mathcal{D}$. Note that this is true for all the path, therefore the iteration finishes earlier and this cannot be the most energy efficient. Contradiction. $\Box$

Lemma 3.2. If vertex v_i's completion time $t > T_l^{v_i}$, then the current iteration is not completable by deadline $\mathcal{D}$.

[Proof]: Assuming that best case execution time occur for all the rest vertices at time t when v_i is completed, this gives us the earliest time that we can complete the

current iteration and there exists at least one path from v_i to one sink node v ($u_0 = v_i, u_1, \cdots, u_k = v$), and for each pair (u_j, u_{j+1}) $T_l^{u_j} = T_l^{u_{j+1}} - BCET_{u_{j+1}} - w_{u_j,u_{j+1}}$. The completion of this path happens at time

$$\begin{aligned} T &= t + \sum_{j=0}^{k-1}(w_{u_j,u_{j+1}} + BCET_{u_{j+1}}) > T_l^{v_i} + \sum_{j=0}^{k-1}(w_{u_j,u_{j+1}} + BCET_{u_{j+1}}) \\ &= T_l^{u_1} + \sum_{j=1}^{k-1}(w_{u_j,u_{j+1}} + BCET_{u_{j+1}}) = \cdots = T_l^{v} = \mathcal{D} \end{aligned}$$

$\square$

Combining these two lemmas and the naïve approach that achieves the highest possible completion ratio $\mathcal{Q}^{max}$, we have:

Theorem 3.3. (BEEM1) If we know the execution time t_e^v of vertex v, the following algorithm achieves $\mathcal{Q}^{max}$ with the minimum energy consumption.

Let t be the current time that v is going to be processed and t_e^v be v's real execution time,

- if $t + t_e^v > T_l^v$, terminate the current iteration;
- if $t + t_e^v < T_e^v$, scale voltage such that v will be completed at T_e^v;
- otherwise, process at the highest voltage as in the naïve approach;

However, it is unrealistic to have each task's real execution time known a priori, we hereby propose algorithm BEEM2, another version of BEEM that does not require tasks' real-time execution time to make decisions, yet still achieves the highest completion ratio $\mathcal{Q}^{max}$:

Algorithm BEEM2

Let t be the current time that v is going to be processed,

- if $t + BCET_v > T_l^v$, terminate the current iteration;
- if $t + WCET_v < T_e^v$, scale voltage such that $WCET_v$ will be completed at T_e^v;
- otherwise, process at the highest voltage;

Without knowing task's real execution time, BEEM2 conservatively i) terminates an iteration if it is incompletable even in vertex v's best case execution time $BCET_v$; and ii) slows down to save energy while still guaranteeing that vertex v's worst case execution time $WCET_v$ can be completed at its earliest completion time T_e^v. We mention that the pair $\{T_e^v, T_l^v\}$ can be computed offline only once and both BEEM1 and BEEM2 algorithms require at most two additions and two comparisons. Therefore, the on-line decision making takes constant time and will not increase the run time complexity. Finally, similar to our discussion for the naïve approach, further energy reduction is possible if the required completion ratio $\mathcal{Q}_0 < \mathcal{Q}^{max}$.

3.3.3 QGEM: Completion Ratio Guaranteed Energy Minimization

Both the naïve approach and BEEM algorithms achieve the highest completion ratio. Although they can also be adopted to provide exactly the required completion ratio $\mathcal{Q}_0$ for energy reduction, they may not be the most energy efficient way to do so when $\mathcal{Q}_0 < \mathcal{Q}^{max}$. In this section, we propose a hybrid offline on-line completion ratio $\mathcal{Q}$ guaranteed energy minimization (QGEM) algorithm, which consists of three steps:

In Step 1, we seek to find the minimum effort (that is, the least amount of computation t_s^i we have to process on each vertex v_i) to provide the required completion ratio $\mathcal{Q}_0$ (Fig. 3.1). Starting with the full commitment to serve every task's WCET (*line 2*), we use a greedy heuristic to lower our commitment the vertices along critical paths (*lines 6-13*). Vertex v_j is selected first if the reduction from its WCET t_{j,k_j} to t_{j,k_j-1} (or from the current $t_{j,l}$ to $t_{j,l-1}$) maximally shortens the critical paths and minimally degrades the completion ratio, measuring by their product (*line 10*).

```
/* Step 1: Minimum effort for completion ratio guarantee. */
 1. find a topological order of the vertices: $v_1, \cdots, v_n$;
 2. $t_s^i = t_{i,k_i}$;                    /* assign WCET to each vertex */
 3. $\mathcal{Q} = 1$; /* completion ratio must be 1 if each vertex gets its WCET */
 4. determine the completion time $L$;
 5. while ($\mathcal{Q} > \mathcal{Q}_0$)
 6. { for each vertex $v_j$ along critical paths;
 7.     { determine the completion time $L'$ when reduces $t_s^j$
          from its current $t_{j,l}$ to $t_{j,l-1}$;
 8.       compute the completion ratio $\mathcal{Q}'_j = \mathcal{Q} \cdot \frac{P_{j,l-1}}{P_{j,l}}$;
 9.     }
10.     pick the vertex $v_j$ that achieves the maximum gain $(L - L') \cdot \frac{P_{j,l-1}}{P_{j,l}}$;
11.     $\mathcal{Q} = \mathcal{Q} \cdot \frac{P_{j,l-1}}{P_{j,l}}$;
12.     if ($\mathcal{Q} > \mathcal{Q}_0$)      $t_s^j = t_{j,l-1}$ ;
13. }
```

Figure 3.1: QGEM's offline part to determine the minimum commitment to provide $\mathcal{Q}_0$.

The goal in Step 2 is to allocate the maximum execution time t_q^i for each task

v_i to process the minimum computation t_s^i and to have the completion time L close to deadline $\mathcal{D}$ (Fig. 3.2). *Lines 3-9* repetitively scale t_q^i for all the tasks. Because the IPCs are not scaled, maximally extending the allocated execution time to each task by a factor of $\mathcal{D}/L$ (*line 6*) may not stretch the completion time from L to $\mathcal{D}$. Furthermore, this unevenly extends each path and we re-evaluate the completion time (and critical path) at *line 7*. To prevent an endless repetition, we stop when the scale factor r is less than a small number ϵ (*line 5*), which is set as 10^{-6} in our simulation. *Lines 11-22* continue to scale t_q^i for vertices off critical paths in a similar way.

Now for vertex v_i, we have the pair (t_s^i, t_q^i) which represent the minimum amount of work and maximal execution time we commit to v_i. Define, recursively, the expected drop-time for v_i to be

$$D_i = t_q^i + \max\{D_k + w_{v_k,v_i} | (v_k, v_i) \in E\} \tag{3.6}$$

Step 3 defines the on-line voltage scheduling policy for the QGEM approach in Fig. 3.3, where we scale voltage to complete a task v_i by its expected drop-time D_i assuming that the real-time execution time requirement equals to the minimum workload t_s^i we have committed to v_i (*line 2*). If v_i demands more, it will be finished after D_i and we will drop the current iteration (*line 4*).

Note that if every task v_i has real execution time less than t_s^i in an iteration, QGEM's on-line scheduler will be able to complete this iteration. On the other hand, if longer execution time occurs at run-time, QGEM will terminate the iteration right after the execution of this task. From the way we determine t_s^i (in Fig. 3.1), we know that the required completion ratio $\mathcal{Q}_0$ will be guaranteed. Energy saving comes from two mechanisms: the early termination of *unnecessary* iterations (*line 5* in Fig. 3.3) and the use of low voltage to fully utilize the time from

```
/* Step 2: Maximum execution time allocation
            with deadline constraint.*/
 1. for each vertex $v_i$
 2.     $done(v_i) = 0$;   $t_q^i = t_s^i$;     /* allocate time $t_s^i$ to each vertex */
 3. determine the completion time $L$;
 4. $r = \frac{\mathcal{D}}{L} - 1$;
 5. while ( $r \geq \epsilon$ )                    /* to prevent an endless loop */
 6. { $t_q^i = t_q^i \cdot (1 + r)$;    /* scale the time allocated to each vertex */
 7.     determine the completion time $L$;
 8.     $r = \frac{\mathcal{D}}{L} - 1$;
 9. }
10. for each vertex $v_i$ on critical paths      $done(v_i) = 1$;
11. while ($done(v_i) = 0$ for some vertex $v_i$)
12. { determine the completion time $L$;
13.   while ($L < \mathcal{D}$)
14.     { for each vertex $v_i$ with $done(v_i) = 0$
15.         $t_q^i = t_q^i \cdot (1 + \delta)$;          /* $\delta$ is a small positive number */
16.       determine the completion time $L$;
17.     } /* $L$ may exceed deadline $\mathcal{D}$, so we have to scale back $t_q^i$. */
18.   for each vertex $v_i$ with $done(v_i) = 0$
19.     { $t_q^i = t_q^i / (1 + \delta)$;
20.       if $v_i$ is on the critical path      $done(v_i) = 1$;
21.     }    /* it is still possible to scale vertices off critical paths. */
22. }
```

Figure 3.2: QGEM's offline part to allocate execution time for each task.

now to a task's expected drop-time (*line 2* in Fig. 3.3). We will confirm our claim on QGEM's completion ratio guarantee and demonstrate its energy efficiency by simulation in the next section.

```
/* Step 3: On-line voltage scheduling. */
1. $t$ = current time when vertex $v_i$ is ready for processing;
2. scale voltage such that the fixed workload $t_s^i$ can be completed by time $D_i$;
3. execute task $v_i$ to its completion;
4. if the completion occurs later than $D_i$
5.     report failure; break and wait for the next iteration;
```

Figure 3.3: On-line scheduling policy for algorithm QGEM.

3.4 Simulation Results

In this section we present the simulation results to verify the efficacy of our proposed approaches. We have implemented the proposed algorithms and simulated them over a variety of real-life and random benchmark graphs. Some task graphs, such as FFT(Fast Fourier Transform), Laplace(Laplace transform) and karp10 (Karplus-Strong music synthesis algorithm with 10 voices), are extracted from popular DSP applications. The others are generated by using *TGFF* [30], which is a randomized task graph generator. We assume that there are a set of homogeneous processors available. However, our approaches are general enough to be applied to embedded systems with heterogeneous multiprocessors.

Before we apply our approaches to the benchmark graphs, we need to schedule all of tasks to available processors based on the performance such as latency. Here we use the dynamic level scheduling (DLS) [119] method to schedule the tasks, however, our techniques could be used with any alternative static scheduling strategy. The DLS method accounts for interprocessor communication overhead when mapping precedence graphs onto multiple processors in order to achieve the latency from the source to the sink as small as possible. We apply this method to the benchmarks and obtain the scheduling results which include the task execution order in each processor and interprocessor communication links and costs. Furthermore, we assume that the interprocessor communication is full-duplex and the intraprocessor data communication cost can be neglected.

After we obtain the results from DLS, we apply the proposed algorithms to them. There are several objectives for our experiments. First, we want to compare the energy consumption by using different algorithms under same deadline and completion ratio requirements. Secondly, we want to investigate the impact of completion ratio requirement and deadline requirement to the energy consumption of the proposed approaches. Finally, we want to study the energy efficiency of our algorithms with different number of processors.

We set up our experiments in the following way. For each task, there are three possible execution time, $e_0 < e_1 < e_2$, that occur at the following corresponding probabilities $p_0 >> p_1 > p_2$ respectively. All processors support real time voltage scheduling and power management (such as shut down) mechanism. Four different voltage levels, 3.3V, 2.6V, 1.9V, and 1.2V are available with threshold voltage 0.5V. For each pair of deadline $\mathcal{D}$ and completion ratio $\mathcal{Q}_0$, we simulate 1,000,000 iterations for each benchmark by using each algorithm. Because naïve, BEEM1 and BEEM2 all provide the highest possible completion ratio that is higher than the required $\mathcal{Q}_0$, in order to reduce the energy, we take 100 iterations as a group and stop execution once $100\mathcal{Q}_0$ iterations in the same group have been completed.

Table 3.3 reports the average energy consumption per iteration by different algorithms on each benchmark with deadline constraint $\mathcal{D}$ and completion ratio constraint $\mathcal{Q}_0(0.900)$. From the table we can see that both BEEM1 and BEEM2 provide the same completion ratio with an average of nearly **29%** and **26%** energy saving over naïve. Compared with BEEM2, BEEM1 saves more energy because it assumes that the actual execution time can be known a priori. However, without this assumption the QGEM approach can still save more energy than BEEM2 in most benchmarks. Specifically, it provides **36%** and **12%** energy saving over naïve and BEEM2 and achieves 0.9111 average completion ratio which is higher than the required completion ratio 0.9000. It is mentioned that for FFT2 benchmark, QGEM has negative energy saving compared to BEEM2 because the deadline $\mathcal{D}$ is so long that BEEM2 can scale down the voltage to execute most of the tasks and save energy.

Fig. 3.4 depicts the completion ratio requirement's impact to energy efficiency of different algorithms with same deadline $\mathcal{D}(9705)$. We can see that with the decrement of $\mathcal{Q}_0$, the energy consumption of each algorithm is decreased. However, different from naïve, BEEM1 and BEEM2, the energy consumption of QGEM doesn't change dramatically. Therefore, although under high completion ratio requirement ($\mathcal{Q}_0 > 0.75$ in Fig. 3.4), using QGEM consumes the least energy, it may consume more energy than BEEM1, BEEM2 even naïve when $\mathcal{Q}_0$ is low.

The deadline requirement's impact to the energy consumption is shown in Fig. 3.5 with the same $\mathcal{Q}_0(0.900)$. Because the naïve approach operates at the highest voltage till the required $\mathcal{Q}_0$ is reached, when the highest possible completion ratio

Table 3.3: Average energy consumption per iteration by naïve, BEEM1, BEEM2 and QGEM to achieve $\mathcal{Q}_0 = 0.900$ with deadline constraints $\mathcal{D}$. (n: number of vertices in the benchmark task graph; m: number of processors; $\mathcal{Q}$: the actual completion ratio achieved by QGEM without forcing the processors stop at $\mathcal{Q}_0$; energy is in the unit of the dissipation in one CPU unit at the reference voltage 3.3V.)

Bench-mark	n	m	No. IPCs	$\mathcal{D}$	naïve	BEEM1	BEEM2	QGEM		
					energy	saving vs. naïve	saving vs. naïve	saving vs. naïve	saving vs. BEEM2	$\mathcal{Q}$
FFT1	28	2	15	1275	1040	6.78%	6.07%	35.71%	31.56%	0.9118
FFT2	28	2	16	2445	2122	18.15%	18.15%	15.96%	-2.67%	0.9104
Laplace	16	2	13	2550	1800	42.75%	32.63%	45.12%	18.53%	0.9232
karp10	21	2	12	993	593	23.44%	15.84%	50.54%	41.23%	0.9392
TGFF1	39	2	20	4956	4438	33.98%	30.75%	38.94%	11.82%	0.9090
TGFF2	51	3	36	4449	6103	34.20%	31.27%	34.36%	4.49%	0.9185
TGFF3	60	3	51	5487	8541	29.73%	27.01%	33.13%	8.39%	0.9034
TGFF4	74	2	49	9216	8839	32.08%	30.68%	38.67%	11.53%	0.9109
TGFF5	84	3	74	6990	11138	29.38%	27.85%	34.56%	9.31%	0.9065
TGFF6	91	2	59	11631	10799	33.23%	32.25%	41.16%	13.16%	0.9057
TGFF7	107	3	89	9129	13608	31.15%	29.71%	36.23%	9.28%	0.9027
TGFF8	117	3	111	9705	15674	28.30%	27.07%	34.51%	10.21%	0.9074
TGFF9	131	2	85	15225	15166	31.00%	30.31%	37.81%	10.77%	0.9084
TGFF10	147	4	163	10124	21926	30.09%	29.04%	31.69%	3.71%	0.9029
TGFF11	163	3	159	13068	22984	25.61%	24.95%	31.76%	9.08%	0.9100
TGFF12	174	4	169	12183	25220	29.89%	29.08%	33.35%	6.02%	0.9074
average						28.73%	26.42%	35.84%	12.28%	0.9111

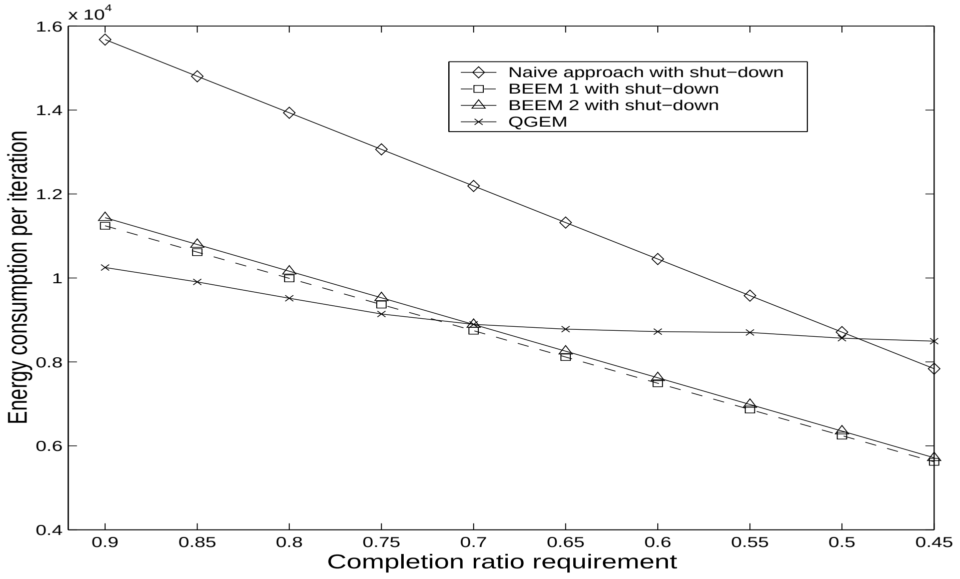


Figure 3.4: Different completion ratio requirement's impact to the average energy consumption per iteration on benchmark TGFF8 with 3 processors.

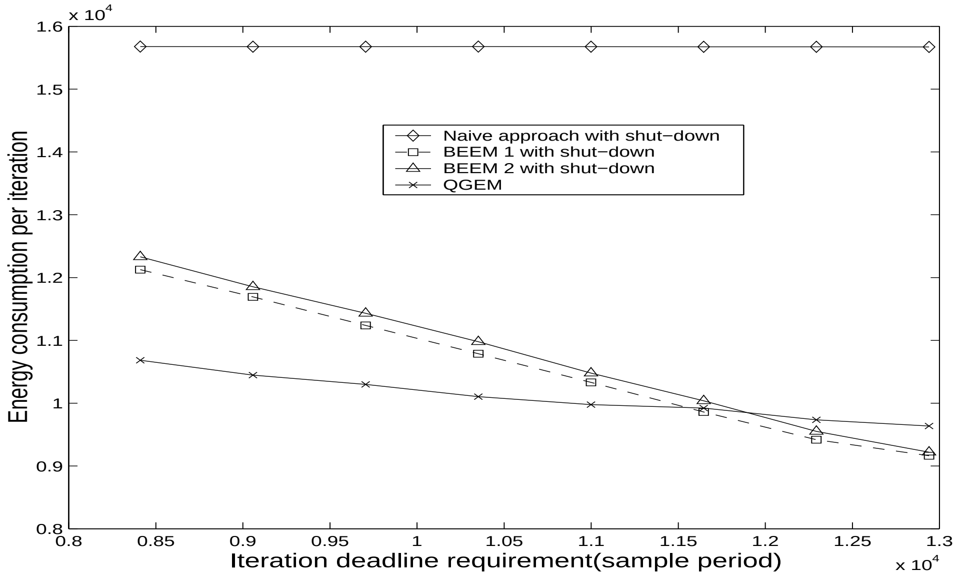


Figure 3.5: Different deadline requirement's impact to the average energy consumption per iteration on benchmark TGFF8 with 3 processors.

of the system is close to 1, its energy consumption keeps constant regardless of the change of the deadline $\mathcal{D}$. However in BEEM1 and BEEM2, the latest completion time T_l^v and the earliest completion time T_e^v for each vertex v depend on $\mathcal{D}$(see Equations (3.3)-(3.5)), and the energy consumption will be reduced dramatically with the increment of $\mathcal{D}$. For QGEM, the increment of deadline also has positive effect on the energy saving while it is not as dramatic as it does to BEEM1 and BEEM2. Similar to the completion ratio requirement's impact, we conclude that QGEM consumes less energy than BEEM1 and BEEM2 in the short deadline (with the condition that $\mathcal{Q}_0$ is achievable), while consuming more energy when the deadline is long.

From Table 3.3 and Fig. 3.4-3.5, we can conclude that QGEM save more energy than BEEM1 and BEEM2 when $\mathcal{Q}_0$ is high and $\mathcal{D}$ is not too long. Actually this conclusion is valid regardless of the number of multiple processors. Fig. 3.6 shows the energy consumption of different algorithms under different deadlines and different number of processors. With the increment of the number of processors, its latency will be reduced. So for the same deadline(e.g., 7275), it is not relatively long and QGEM saves more energy than BEEM1 and BEEM2 for the system with small number of processors (e.g., 4 processors), however, for the system with large number of processors(e.g., >5 processors), QGEM will consume more energy than BEEM1 and BEEM2.

3.5 Conclusions

Many embedded applications, such as multimedia and DSP applications, have high performance requirement yet are able to tolerate certain level of execution failures. We investigate how to trade this tolerance for energy efficiency, another

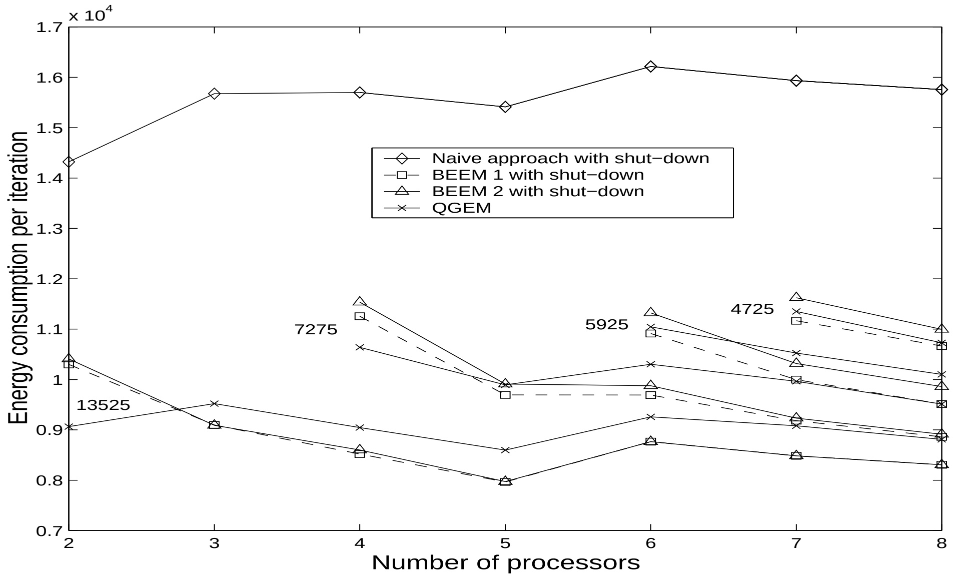


Figure 3.6: The average energy consumption per iteration on benchmark TGFF8 with different number of processors and different deadlines(13525, 7275, 5925 and 4725).

increasingly important concern in the implementation of embedded systems. In particular, we consider systems with multiple supply voltages that enable dynamic voltage scaling, arguably the most effective energy reduction technique. We present several on-line scheduling algorithms that scale operating voltage based on some parameters pre-determined offline. All the algorithms have low run-time complexity yet achieve significant energy saving while providing the required performance, measured by the completion ratio.

Chapter 4

Voltage Set-up Problem on Embedded Systems with Multiple Voltages

4.1 Introduction

Energy consumption has become a major design issue for modern embedded systems especially battery-operated portable devices. The aggressive push for low-power design has prompted the International Technology Roadmap for Semiconductors (ITRS) to predict that the future system will feature multiple supply voltages (V_{dd}), and multiple threshold voltages (V_{th}), on the same chip [2]. Although leakage power, which can be reduced from multiple V_{th}, becomes more significant in such systems, dynamic power still dominates in designs with the current technology such as most DSP systems. Dynamic voltage scaling (DVS) technique varies the clock frequency and supply voltage according to workload at run time to save energy. It achieves the highest energy efficiency for time-varying computational loads if voltage can be varied arbitrarily [18, 100]. However, physical constraints of CMOS circuit limit the applicability of having voltage varying continuously. Instead, it is more practical to make multiple discrete voltages simultaneously available for the system. Transmeta's Crusoe [35], AMD's Athlon 4 [4], Intel's XScale [55], and some DSPs developed in Bell Labs are all examples of advanced high-performance microprocessors that support voltage scaling for low power.

Most existing work on multiple voltage DVS systems assumes that the voltage set-up, which includes the number of voltage levels and the voltage value at each level, are given a priori and focuses on developing the voltage scheduling algorithms to minimize system's energy consumption [56, 70, 103, 105, 125]. However, for multiple voltage DVS systems, the energy consumption depends on not only the scheduler but also the voltage set-up. To the best of our knowledge, how to set up the voltages has been discussed in the following contexts: Chen and Sarrafzadeh [26, 27] studied the power minimization problem on dual-voltage system at gate level, where 5.0V was used as the high voltage and different voltages from 2.0V to 4.2V were used as the low voltage. They suggested that the voltages should be chosen carefully based on the slack distribution of the circuits. Qu and Potkonjak [103] gave analytical solutions on how to build energy efficient communication pipelines under latency constraints by voltage scaling and careful packet fragmentation, where each pipeline stage receives one fixed voltage. Dhar and Maksimovic [29] considered the design of finite impulse response filters and applied Lagrangian method to find the 2N+1 voltages for power minimization, where N is the order of the filter.

In this chapter, we consider the following voltage set-up problem at the application level: *how to determine the number of voltages and each voltage value on a multiple-voltage application specific DVS system such that the system's energy consumption is minimized?* The differences between our work and the ones mentioned above are: 1) we do voltage scaling at the application level, not the gate level, 2) we determine the voltage values for any number of voltages, not only for dual-voltage or levels tightly bounded to the applications, and 3) we also find the best number of voltage levels.

We first use an example to show multiple-voltage system's energy efficiency and the importance of the voltage set-up. Suppose that a system periodically executes one application with period equals to 8. The application's possible execution times, at the reference voltage 3.3V, are 6, 4, 3, and 2 that occur with probabilities 0.05, 0.20, 0.45, and 0.30 respectively. The application has a deadline that equals to its period. We normalize the average energy consumption per iteration at fixed supply voltage 3.3V and threshold voltage $V_{th} = 0.5V$ to be 1. As the worst case execution time (WCET) is 6 that is less than the deadline 8, we can reduce the voltage to 2.7V to utilize the slack and this best fixed voltage system consumes only 67% of the amount consumed by the same system running at 3.3V as shown in Table 4.1. The rest of the table gives the average energy consumption per iteration by various dual-voltage systems. We compute such system's energy consumption by the optimal voltage scaling strategy reported in [56] and [100].

From Table 4.1, we have the following findings:

- Multiple-voltage systems in general save more energy over fixed-voltage systems. For example, the voltage set-ups (V_{high}=3.0V, V_{low}=2.0V) and (V_{high}= 2.7V, V_{low}=1.8V) save more than 35% and 43% energy respectively over the best fixed-voltage system with the lowest voltage 2.7V without any deadline

Table 4.1: The average energy consumption per iteration of dual-voltage system with different voltage set-ups.

Set-up	1	2	3	4	5	6
V_{high}	3.3	2.7	3.3	3.0	3.0	2.7
V_{low}	–	–	1.0	1.0	2.0	1.8
Energy	1	0.67	0.83	0.70	0.43	0.38

missing.

- Different voltage set-ups result in significantly different energy reduction as we can see from the last four columns. Moreover, if not set properly, set-ups 3 (V_{high}=3.3V, V_{low}=1.0V) and 4 (V_{high}=3.0V, V_{low}=1.0V) for example, the multiple-voltage system may consume more energy than the best fixed-voltage system!

We formulate and provide practical solutions to the voltage set-up problem that seeks the most energy efficient voltage setting for the design of multiple-voltage DVS systems. This work is a novel extension under the DVS research framework. Our main contributions include: (1) analytical solutions and a linear search algorithm for dual-voltage DVS systems; and (2) an iterative approach and an approximation method for the general multiple-voltage DVS systems. These results can be used to guide system design as we show by simulation. Surprisingly, our results show that the 3- or 4-voltage system can actually be (almost) as energy-efficient as the ideal system that varies voltage arbitrarily.

The remainder of this chapter is organized as follows. In the next section, we formulate the voltage set-up problem and present the solutions in Section 4.2 and Section 4.3 respectively. Validation of our solutions and experimental results are reported in Section 4.4. We give the conclusion in Section 4.5.

4.2 The Voltage Set-up Problem

We consider the design of an embedded system to perform a set of applications (or a single application with uncertainties in execution time). The system supports DVS for energy minimization. In this section, we first introduce the application model and multiple-voltage DVS system model, then we propose the voltage set-up problem.

4.2.1 Application Model

Each application has a (or a set of) specific amount of computation requirement [90], or equivalently, a certain amount of CPU time to complete the computation before a deadline constraint. This situation occurs in systems (such as DSP systems) that run a single application characterized by the repetitive processing on periodically arriving input samples and each iteration must be completed during its period. It may also happen in systems that assign a fixed amount of time to each of the applications. Another example is an event-triggered system, in which the application requests arrive with a fixed deadline and the time between any two consecutive requests is not less than the deadline. For such system, there is typically one application at a time and the system executes the computation in the non-preemptive way.

Note that an application's execution time can vary dramatically due to a number of factors such as data locality and correlation, I/D cache misses, or pipeline stalls etc. However, it is possible to obtain the application execution time distribution from system's detailed timing information or from simulation on the target hardware [122]. For example, input sample statistics and throughput constraints can be used to model the execution time distribution for many DSP applications such as MPEG decoding. We adopt the assumption in [21] that the real execution time can be known a priori, which is possible particularly in application specific DSP systems. We also assume that the applications are characterized by triples $< e_i, d_i, p_i >$ $(i = 1, 2, \cdots, n)$, where e_i is the execution time, d_i is the deadline, and p_i is the probability that the system executes the application. We mention that e_i's can be the execution times for different applications or the different execution times for the same application.

4.2.2 Multiple-Voltage DVS System Model

We assume that the target multiple-voltage DVS system has m levels of supply voltage ($V_1 < V_2 < \cdots < V_m$) and supports the system shut-down mechanism for energy efficiency. Unlike the DVS system that uses voltage converter to control the operating voltage at run-time [18], our system has all the m voltages physically implemented on the chip, for example by using m standard voltage regulators each of which regulates a specific voltage V_i for a given clock frequency. In this way, the operating system can control the clock frequency at run time by writing to a register in the system control state exactly the way as in [18] except that the system does not need to wait for the voltage converter to stably generate the desired operating voltage. Furthermore, for the execution of each iteration, we will use no more than two different voltages [56]. In sum, we can assume that the system can instantaneously switch its operating voltage from one level to another with very small switching time/energy overhead. There exist hardware overhead, such as the power dissipation on the voltage regulators, to support multiple voltages. However, this is a constant overhead independent of how we set up the m voltage levels.

We adopt the following relationships among the multiple-voltage system's voltage, delay, power and energy consumption: suppose that at the reference (highest) supply voltage $V_{dd}(ref)$ and threshold voltage $V_{th}(ref)$, the processor's power dissipation is $P(ref)$ and the execution time is $T(ref)$ for a fixed amount of computation, then at supply voltage V_{dd} and threshold voltage V_{th}, to accumulate the same amount of computation, execution time T, power dissipation P, and energy consumption E are given by [23]:

$$T = \frac{V_{dd}}{(V_{dd}-V_{th})^2}\frac{(V_{dd}(ref)-V_{th}(ref))^2}{V_{dd}(ref)}T(ref) \tag{4.1}$$

$$P = \frac{V_{dd}(V_{dd}-V_{th})^2}{V_{dd}(ref)(V_{dd}(ref)-V_{th}(ref))^2}P(ref) \tag{4.2}$$

$$E = P \cdot T = \frac{V_{dd}^2}{V_{dd}(ref)^2}P(ref)T(ref) \tag{4.3}$$

Note that the voltage set-up problem exists regardless of how we model the relationships among the system's voltage, delay, power and energy consumption. Our presented approaches to solve this problem is still valid for the other models and similar results (but not exactly the same) can be expected.

Based on the above application and system models, we consider the following **voltage set-up problem**. For a given set of applications characterized by triples of $< e_i, d_i, p_i >$ $(i = 1, 2, \cdots, n)$, determine each voltage level for a multiple-voltage DVS system with m voltages $(V_1 < V_2 < \cdots < V_m)$ in order to minimize its energy consumption without missing any deadline; and determine m, the number of voltage levels, together with the value of each voltage to achieve the maximum energy saving. The first part of the problem considers the case when the system has a given number of voltages and seeks for the most energy efficient voltage set-up. The second part takes into consideration the overhead to support multiple voltages and questions both how many levels of voltage and what value of each voltage level should be implemented on the multiple-voltage system.

4.3 Solving the Voltage Set-up Problem

In this section we first introduce three basic lemmas and then present the analytic solution and an exact approach for the dual-voltage DVS system. We also propose an iterative approach and a linear (to the number of voltages) approximation method for solving the problem in the general case. Finally we discuss how to find the best voltage set-up (both the number of voltage levels and the value of each voltage) in order to achieve the maximum energy saving.

Suppose that the i-th application has deadline d_i and requests $e_i \leq d_i$ as execution time under the reference voltage $V_{dd}(ref)$. We define its ideal voltage V_i^0 to be the level at which the system will complete the workload e_i at d_i with minimum energy consumption [56]. From Equation (4.1), we can compute the value of V_i^0 (for a fixed threshold voltage) or determine the relationship between V_i^0 and its corresponding threshold voltage. Without loss of generality, we assume that $V_1^0 < V_2^0 < \cdots < V_n^0$ are the ideal voltages for the n applications characterized by $< e_i, d_i, p_i >$ $(i = 1, 2, \cdots, n)$ and $V_1 < V_2 < \cdots < V_m$ are the m voltage levels to be set up on the system. Any solution to the voltage set-up problem must satisfy the following lemmas:

Lemma 4.1: $V_m = V_n^0$.

Proof: If $V_m < V_n^0$, the system will not be able to complete the n-th application by its deadline.

If $V_m > V_n^0$, we consider a new voltage set-up where we replace V_m by $V'_m = V_n^0$. No deadline will be missed because $V'_m \geq V_i^0$. We only need to show that the new voltage set-up reduces energy consumption. It is well-known (see [56] or [100] for example) that on a multiple voltage system, the energy consumption for the i-th application is minimized if we use only two voltages V_j and V_{j+1}, which are immediate neighbors to V_i^0, such that $V_j < V_i^0 < V_{j+1}$. Therefore, the new voltage set-up will only affect the energy consumption for applications with ideal voltages higher than V_{m-1}. For such applications, the original set-up uses voltages V_{m-1} and V_m, while the new voltage set-up uses V_{m-1} and V'_m ($< V_m$). Due to the fact that the power/energy consumption is a convex function of the supply voltage, the energy consumption under the new voltage set-up will be less. ▯

Lemma 4.2: $V_1 \geq V_1^0$.

Proof: Similar to the proof of Lemma 4.1, if $V_1 < V_1^0$, we consider a new voltage set-up where we replace V_1 by $V'_1 = V_1^0$. The new voltage set-up will only affect the energy consumption for applications with ideal voltages lower than V_2. For such applications, the energy consumption under the original voltage set-up that uses voltages V_1 and V_2 is more than that under the new voltage set-up, which uses V'_1 ($> V_1$) and V_2. Therefore, setting up the lowest voltage V_1 lower than the lowest ideal voltage V_1^0 will not benefit any application. ▯

Lemma 4.3: There exists at most one $V_i \in (V_{k-1}^0, V_k^0]$ for any integer $k > 1$.

Proof: If there are two or more voltages in $(V_{k-1}^0, V_k^0]$, we can replace them by V_{k-1}^0 and V_k^0 without changing others. Similar to the proof of Lemma 4.1, this will result in more energy reduction. ▯

Example: Suppose that an application has 5 possible execution times, corresponding to 5 ideal voltages, 1.2V, 1.6V, 2.4V, 2.8V, and 3.2V. Lemma 4.1 says that the highest voltage must be set at 3.2V; Lemma 4.2 implies that the lowest voltage should not be lower than 1.2V; and Lemma 4.3 guarantees that any voltage set-up that has two or more voltages fall in the interval of (1.2,1.6], (1.6,2.4],(2.4,2.8], or (2.8,3.2] cannot be optimal. For example, none of the following set-ups is optimal: $\{1.6, 3.3\}$, $\{1.2, 2.4, 3.0\}$, $\{1.1, 2.4, 3.2\}$, $\{1.8, 2.0, 2.8, 3.2\}$.

These lemmas not only identify non-optimal voltage set-ups, they are also fundamental for our proposed solutions to the voltage set-up problem. In the rest of this section, we first address the problem of how to set up m-voltage systems, where m is given, for application(s) with n distinct possible execution times. Figure 4.1 gives the details on how we approach the problem. We then discuss how to determine both the number of voltage levels m and the voltage of each level in order to achieve the maximum energy saving.

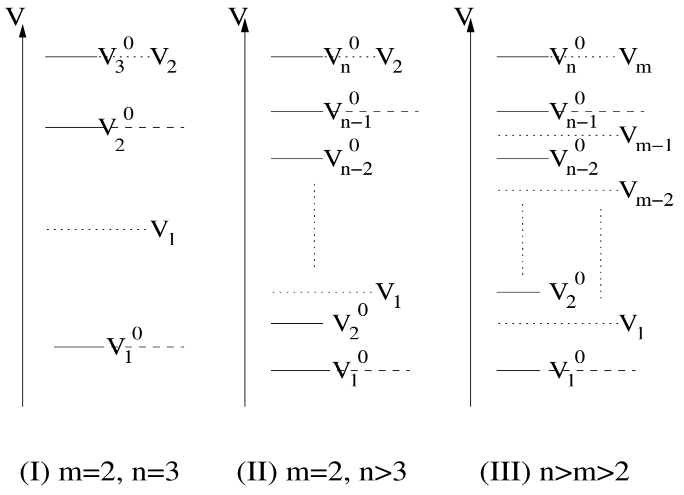


Figure 4.1: Summary of voltage set-up solutions for m-voltage system with n applications. (V_i^0 is the ideal voltage for i-th application, $V_1^0 \leq V_2^0 \leq \cdots \leq V_n^0$; V_j is the j-th supply voltage and $V_1 < V_2 < \cdots < V_m$.)

4.3.1 Case I: Dual Voltages Three Applications (m=2 and n=3)

We consider a dual-voltage system (m=2) with three applications (n=3). For simplicity, we assume that each application has one fixed execution time. (This does not lose the generality because one can treat an application with k different possible execution times as k applications.) Clearly, this is the simplest non-trivial case because one can simply use all the ideal voltages if $m \geq n$.

Let $V_1 < V_2$ be the system's two voltages and $V_1^0 \leq V_2^0 \leq V_3^0$ be the ideal voltages for three applications characterized by $< e_1, d_1, p_1 >, < e_2, d_2, p_2 >, and < e_3, d_3, p_3 >$. From the above lemmas, we know that $V_2 = V_3^0$ and $V_1 \in [V_1^0, V_2^0]$ (because $V_2 \in (V_2^0, V_3^0]$). Under such voltage set-up,

• The third application will be executed at V_2 and completed at its deadline d_3;

• For the second application, the system runs at the lower voltage V_1 for a certain amount of time to save energy before speeds up to V_2 to meet its deadline d_2;

• The first application will be executed at V_1 till its completion.

Therefore, the system's expected energy consumption can be expressed as:

$$E = \frac{P(ref)}{V_{dd}(ref)^2}[p_3 V_2^2 e_3 + p_2(V_2^2(e_2 - t_2) + V_1^2 t_2) + p_1 V_1^2 e_1] \tag{4.4}$$

where t_2 satisfies

$$\begin{aligned} \frac{V_2}{(V_2 - V_{th2})^2} \frac{(V_{dd}(ref) - V_{th}(ref))^2}{V_{dd}(ref)}(e_2 - t_2) & \\ + \frac{V_1}{(V_1 - V_{th1})^2} \frac{(V_{dd}(ref) - V_{th}(ref))^2}{V_{dd}(ref)} t_2 &= d_2 \end{aligned} \tag{4.5}$$

The physical meaning of t_2 is as follows. Suppose that W is the portion of the workload from the second application being executed at voltage V_1. t_2 is the time required to complete the same workload W at the reference voltage.

If V_1 and V_2 are associated with different threshold voltages V_{th1} and V_{th2}, we can prove that analytical solutions do not exist and the problem can only be solved numerically. However, if the threshold voltage remains the same, i.e. $V_{th1} = V_{th2} = V_{th}$, we can apply the first order condition and conclude that energy consumption (4.4) is minimized only if V_1 is the solution to the following equation:

$$\begin{aligned}(-2V_2p_2d_{2p} + 2V_2^2p_1e_1)V_1^3 + [(2V_2V_{th} - V_2^2 + 3V_{th}^2)p_2d_{2p} - 4V_2V_{th}^2p_1e_1]V_1^2 \\ +[(-4V_{th}^3 + 2V_2V_{th}^2)p_2d_{2p} + 2V_{th}^4p_1e_1]V_1 + p_2d_{2p}V_{th}^2(V_2 - V_{th})^2 = 0 \quad (4.6)\end{aligned}$$

where

$$d_{2p} = \frac{d_2(V_2 - V_{th})^2V_{dd}(ref)}{(V_{dd}(ref) - V_{th})^2} - V_2e_2 \quad (4.7)$$

The cubic equation (4.6) can be solved analytically and we conclude

Theorem 4.1. Analytical optimal solution exists for Case I with fixed threshold voltage.

4.3.2 Case II: Dual Voltages Multiple Applications (m=2 and n>3)

In this case, we know that $V_2 = V_n^0$ and $V_1 \in [V_1^0, V_{n-1}^0]$.

• The n-th application will be executed at V_2 to its completion;

• For applications with ideal voltages larger than V_1, both voltages will be used to meet the deadlines and save energy;

• For applications with ideal voltages less than V_1, only V_1 will be used as it is sufficiently fast to finish these applications earlier than their deadlines.

We seek for V_1 that minimizes the total energy consumption and meets all applications' deadlines. These two conditions can be expressed as:

$$E = \sum_{i=1}^{n} \frac{P(ref)p_i}{V_{dd}(ref)^2}[V_2^2(e_i - t_i) + V_1^2 t_i] \tag{4.8}$$

$$\begin{aligned} \frac{V_2}{(V_2 - V_{th2})^2}\frac{(V_{dd}(ref) - V_{th}(ref))^2}{V_{dd}(ref)}(e_i - t_i) + \\ \frac{V_1}{(V_1 - V_{th1})^2}\frac{(V_{dd}(ref) - V_{th}(ref))^2}{V_{dd}(ref)}t_i \;\leq\; d_i \end{aligned} \tag{4.9}$$

where t_i is defined the same as t_2 in equation (4.4).

These conditions are similar to equations (4.4) and (4.5) in Case I except that (4.9) imposes a set of nonlinear inequality constraints. It is well-known in the context of nonlinear programming that this makes the problem difficult to solve [14].

Figure 4.2 depicts an optimal algorithm with linear complexity, O(n), for the problem in this case. Assuming that $V_1 \in [V_{k-1}^0, V_k^0]$, we can remove the inequality constraints in (4.9). Specifically, for applications $k, \cdots, n$, deadlines will be met exactly for energy reduction (step 4); for the other applications, their deadlines will be satisfied automatically because V_1 is higher than their ideal voltages (step 5). Now this becomes the same problem as Case I and we can apply Theorem 4.1 to solve it optimally (step 6). Voltage V_1 that satisfies (4.8) and (4.9) must be in one of the above intervals, and we will find it when we visit that interval in step 3.

4.3.3 Case III: Multiple Voltages Multiple Applications (m>2)

Even when there are more than two voltages available, the system will still use at most two voltages to execute each application [56]. Define $\delta_{ij} = 1$ if voltage V_j is

```
Input: n applications $\{< e_i, d_i, p_i >: i = 1, 2, \cdots, n\}$ with their corresponding
        ideal voltages $V_1^0 \le V_2^0 \le \cdots \le V_n^0$.
Output: $V_1$ and $V_2$ that minimize (4.8) and satisfy (4.9).
Algorithm:
 1. $V_2 = V_n^0$;
 2. for each $k = 2, 3, \cdots, n-1$
 3. {  assume $V_1 \in [V_{k-1}^0, V_k^0]$;
 4.    replace " $\le$ " by " $=$ " for $i = k, k+1, \cdots, n$ in (4.9);
 5.    delete the rest of the inequalities in (4.9);
 6.    solve the problem as in Case I;
 7.    let $V_{1,k}$ be the voltage and $E_k$ be the energy;
 8. }
 9.    report the voltage $V_{1,k}$ that has the least $E_k$ as $V_1$.
```

Figure 4.2: Voltage set-up algorithm for the case of m=2, $n \ge 3$.

used during the execution of the i-th application and $\delta_{ij} = 0$ otherwise. Similar to t_2 defined in equation (4.5), define t_{ij} be the required execution time of the i-th application under the reference voltage to finish the same portion of computation that is done with V_j. We then can formulate this general voltage set-up problem as a nonlinear programming problem in Figure 4.3.

As analytic solutions for this general case do not exist, numerical approaches can be used to exhaustively search for the solution to this nonlinear programming problem. We can further speed up the search process by eliminating all the voltage set-ups that have two or more voltages between two consecutive ideal voltages (Lemma 4.3). However, this exhaustive search will still be expensive particularly

$$
\begin{array}{ll}
\textbf{Find} & V_1, V_2, \cdots, V_m \\
\textbf{Minimize} & E = \frac{P(ref)}{V_{dd}(ref)^2} \sum_{i=1}^{n} p_i \sum_{j=1}^{m} V_j^2 \delta_{ij} t_{ij} \\
\textbf{Subject to} & t_{ij} \geq 0, \\
 & V_j > 0, \\
 & \sum_{j=1}^{m} \delta_{ij} \leq 2, \ \ \delta_{ij} \text{ is 0 or 1}, \\
 & \sum_{j=1}^{m} t_{ij} = e_i, \\
 & \sum_{j=1}^{m} \frac{V_j}{(V_j - V_{thj})^2} \frac{(V_{dd}(ref) - V_{th}(ref))^2}{V_{dd}(ref)} t_{ij} \leq d_i.
\end{array}
\qquad (4.10)
$$

Figure 4.3: General voltage set-up problem as a nonlinear programming problem for the case of $m > 2$.

when m is large. We thus propose two heuristics, an iterative approach and an approximation method to efficiently search for the solution based on the convexity of the energy function.

An Iterative Approach:

• Start with the single voltage system with voltage $V_{1,1} = V_n^0$, at which the system has the least energy consumption;

• Apply the algorithm in Figure 4.2 to solve for $V_{2,1}$ and $V_{2,2}$, the best voltage set-up for dual-voltage system;

• For k-voltage (k≥3) systems repetitively do the following: let $V_{k,k} = V_{k-1,k-1}$, search $V_{k,i}$ between $V_{k-1,i-1}$ and $V_{k-1,i}$ for the most energy efficient set-up such that $V_1^0 \leq V_{k,1} \leq V_{k-1,1} \leq V_{k,2} \leq V_{k-1,2} \leq \cdots \leq V_{k,k-1} \leq V_{k-1,k-1} = V_{k,k} = V_n^0$.

Note that if we know the energy overhead E_k to support k voltages on the system, we can add it to the energy consumption of the best k-voltage system and determine how many voltages we should implement on the system.

An Approximation Method:

• Start with a random m-voltage set-up;

• Fix the (m-1) high voltages and compute the lowest voltage V_1 by a procedure similar to the algorithm in Figure 4.2;

• Determine V_2 by fixing the obtained V_1 and the other (m-2) high voltages;

• Continue till after we update the value of V_{m-1}, the second highest voltage; (This is one round of updating.)

• If there is energy improvement, go back to the second step with this new obtained voltage set-up;

• Report the optimal voltage set-up.

This method is based on the convexity of the energy function. Although we cannot guarantee how many rounds we need to update the voltage set-ups to reach the optimal values, simulation shows that the voltage set-up converges to the optimal solution (calculated by numerical method) after $2 \sim 3$ rounds.

Finally, we mention that these two techniques and Lemmas 1∼3 can be combined together to solve the problem efficiently.

4.3.4 Finding the Best Voltage Set-up

Once m, the number of voltages on the system, is fixed, we now know how to find the most energy-efficient voltage set-up from the above discussion. The corresponding average energy consumption per execution can be conveniently obtained from Equation (4.10). If we ignore the hardware overhead (e.g., the area and power on the voltage regulators or DC-DC converters) to support multiple levels of voltages, then clearly the more voltages we have, the less energy will be consumed. A simple reason is that m-voltage systems can also be treated as (m+1)-voltage systems where two of the (m+1) voltages have the same value.

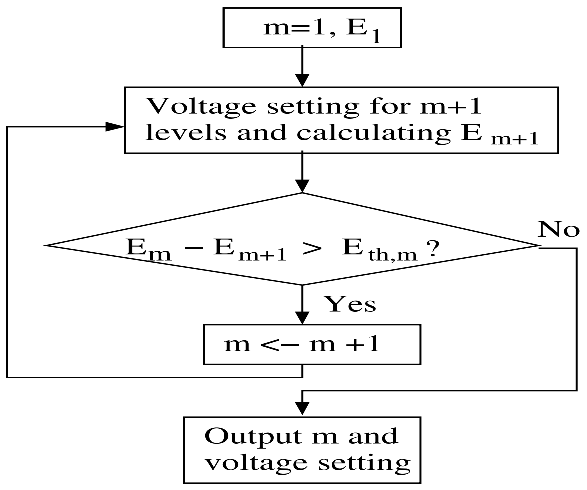

m=1, E1
Voltage setting for m+1 levels and calculating E m+1
Em – Em+1 > E th,m ?
No
Yes
m <– m +1
Output m and voltage setting

Figure 4.4: Flow to find the best voltage set-up.

However, supporting multiple voltages on the same system does require additional hardware and will cause area, delay, and also power penalties. It becomes important to investigate the trade-off between more voltage levels and the overhead they introduce. Figure 4.4 shows a scheme on how to find the best voltage set-up, i.e. the optimal number of voltage levels and the value of each level, to minimize the energy consumption, assuming that there is a threshold energy cost $E_{th,m}$. If the energy saving by including the (m+1)st voltage, $E_m - E_{m+1}$, is higher than this threshold $E_{th,m}$, then (m+1)-voltage system is more energy efficient than any m-voltage systems. Otherwise, it is not worth going to (m+1) voltages and we report the best m-voltage set-up as the overall optimal solution. The threshold energy cost $E_{th,m}$ can be measured by the additional hardware cost to have (m+1) voltages over m voltages that can be obtained empirically. We mention that in general this threshold energy cost increases as one attempts to implement more and more different voltages on the same system.

4.4 Simulation Results

There are two goals in our simulation: demonstrating the importance of voltage set-up problem and validating our proposed approaches. We formulate the voltage set-up problem in two occasions based on a set of randomly generated applications and the MPEG video encoder. The problems are then solved both analytically and numerically by using our approaches. Finally we compare the energy consumption under different voltage set-ups obtained by using exhaustive simulation in Matlab in order to test the correctness of the results and the effectiveness of our proposed methods. Note in this section the energy is in the unit of dissipation in one CPU unit at the reference voltage 3.3V.

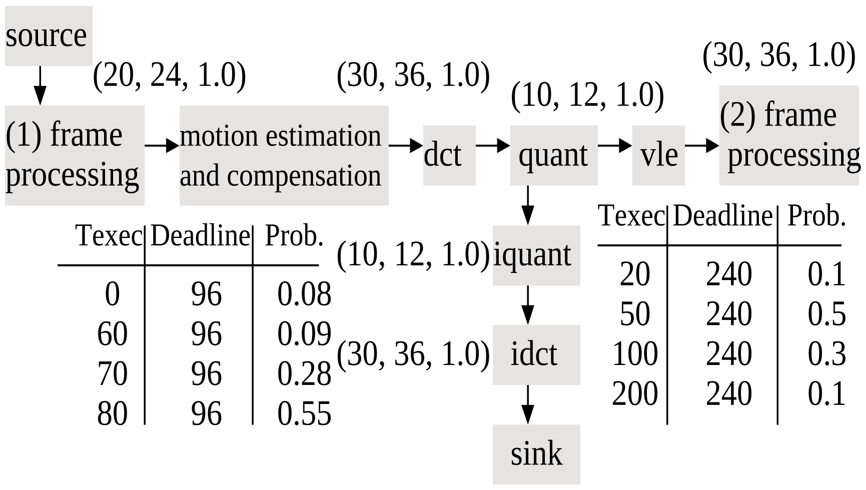
Table 4.2 describes the two randomly generated abstract applications with their deadlines, execution time distributions, and ideal voltages computed from Equation (4.1). Figure 4.5 depicts the flow of MPEG encoding process as a set of subtasks. Next to each subtask, its <execution time T_{exec}, deadline, probability> triple is reported [59].

For each example, we apply the proposed approaches to find the best voltage set-ups for dual-voltage, 3-voltage, and 4-voltage DVS systems as reported in Table 4.3. The dual-voltage case is solved by the algorithm in Figure 4.2. 3-voltage and 4-voltage solutions are obtained by the approximation method. We also list the energy consumption of the best fixed-voltage system and the ideal DVS system in the table for comparison. Note that the energy consumption of the ideal DVS system, where we have the ideal voltage for each possible execution time, is the lower bound of the system energy consumption.

For the first example of two ad hoc applications, multiple-voltage DVS systems save significant amount of energy over the fixed-voltage system. The saving is more

Table 4.2: Information on the two ad hoc applications.

Application	Deadline d_i	Execution Time e_i	Probability p_i	Ideal Voltage V_i^0
A	10	9	0.03	3.0564
		4	0.18	1.8124
		3	0.39	1.5516
B	8	6	0.04	2.6888
		4	0.10	2.0669
		3	0.12	1.7479
		2	0.14	1.4176



Texec	Deadline	Prob.
0	96	0.08
60	96	0.09
70	96	0.28
80	96	0.55

Texec	Deadline	Prob.
20	240	0.1
50	240	0.5
100	240	0.3
200	240	0.1

Figure 4.5: MPEG video encoder execution time distributions and corresponding deadlines in 10^4 cycles (redrawn from [59]). The lower left table is related to motion estimation and compensation; the lower right table is related to vle (variable length encoding).

Table 4.3: The optimal voltage set-ups and their corresponding average energy consumption per execution. (In the parenthesis of energy columns, "-" is the energy saving over the fixed voltage system, "+" is the "wasted" energy comparing to the ideal voltage system.)

DVS	2-Application		MPEG Encoder	
Systems	Voltages	Energy	Voltages	Energy
fixed-voltage	3.0564	2.9536 (+151.1%)	2.8934	26.7125 (+20.1%)
dual-voltage	3.0564 1.8124	1.3833 (-53.2%) (+17.6%)	2.8934 1.8511	23.1478 (-13.3%) (+4.0%)
3-voltage	3.0564 2.0688 1.5514	1.2337 (-58.2%) (+4.9%)	2.8934 1.8558 1.3031	22.4958 (-15.8%) (+1.1%)
4-voltage	3.0564 2.0768 1.8119 1.5509	1.2071 (-59.1%) (+2.6%)	2.8934 2.6374 1.8554 1.3031	22.3020 (-16.5%) (+0.2%)
ideal	–	1.1763	–	22.2506

than 53% when we carefully choose the second voltage on the dual-voltage system. With the addition of the third and fourth voltages, we see the continuous increase in energy reduction. (We did not consider the hardware overhead to support these new voltage levels. However, once such overhead is measured, we can easily tell whether the energy reduction is sufficient to cancel this overhead and decide how many levels of voltage should be implemented on the system.) Finally, we mention that, comparing to the lower bound in the ideal system, the best fixed-voltage set-up consumes more than 151% additional energy. But this "wasted" energy drops to 17.6%, 4.9%, and 2.6% for the dual-, 3-, and 4-voltage system respectively. It indicates the effectiveness of multiple-voltage DVS system's energy saving, which is very close to maximal energy saving by DVS when the number of voltage levels is large enough.

We have similar observations from the MPEG encoder example except that the energy saving (over the fixed-voltage system) is much lower, albeit a notable 13%. This is because that majority of the energy is consumed on the deterministic subtasks. However, multiple-voltage systems again successfully reduced the "wasted" energy from more than 20% (for fixed voltage) to 4.0%, 1.1%, and 0.2%.

To validate the correctness of our results, we use Matlab to simulate 100,000 iterations of each application under different voltage set-ups for dual-, 3-, and 4-voltage systems. In all the cases, this exhaustive search finds the same solution, within the precision of voltage increment 0.01V we set, as we reported in Table 4.3 by our methods. Figures 4.6 and 4.7 illustrate this for the dual-voltage system where the energy is in the unit of dissipation in one CPU unit at 3.3V. We set the high voltage V_2 to go from V_n^0 (3.0564V in Figure 4.6 and 2.8934V in Figure 4.7) to the reference voltage 3.3V, and the low voltage V_1 to go from 1.0V to 3.3V, both with an increment of 0.01V. In both figures, we see that the energy consumption is minimized at the same set-up as we obtained theoretically.

4.5 Conclusions

In this chapter, we consider the voltage set-up problem for application specific multiple-voltage DVS system design. The problem seeks to determine the number of voltage levels and the voltage at each level to minimize the average energy consumption for a given set of applications. We give optimal solutions in analytic

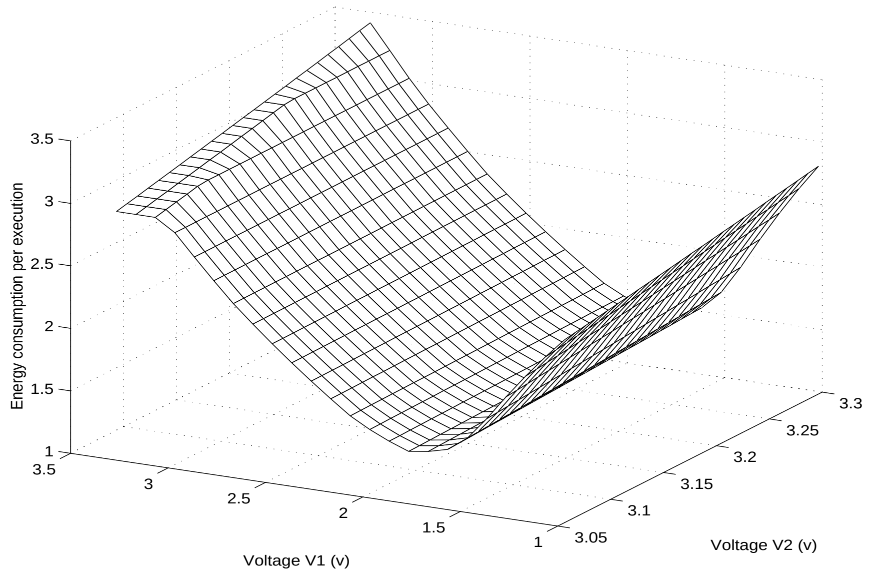


Figure 4.6: Dual-voltage system's average energy consumption for the two ad hoc applications with different voltage set-ups.

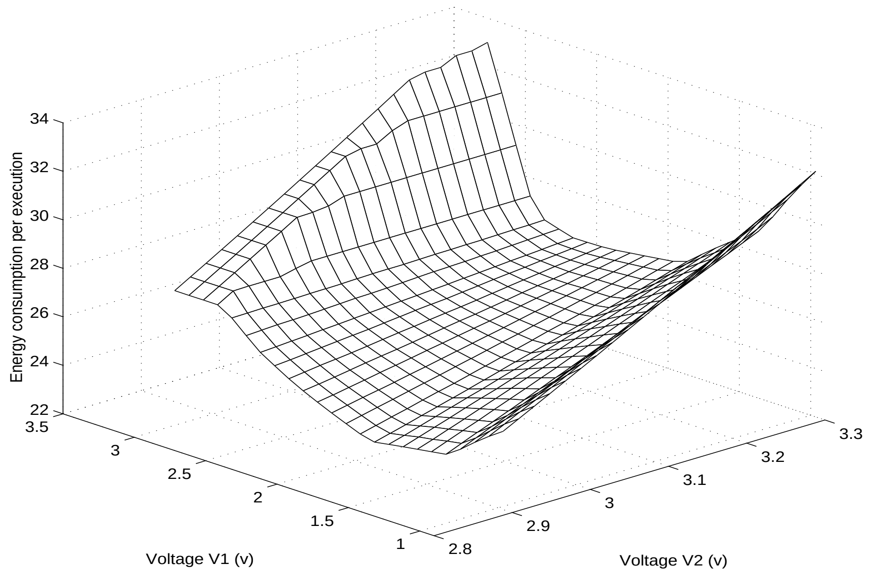


Figure 4.7: Dual-voltage system's average energy consumption for the MPEG encoder with different voltage set-ups.

form for the dual-voltage system and develop two heuristics (an iterative approach and an approximation method) for the general case. The hardware overhead to supply multiple voltages, once obtained, can be conveniently integrated into our techniques to solve the voltage set-up problem. We apply our methods to the designs of an ad hoc application specific system and the MPEG video encoder. Simulation results show the correctness and efficiency of our approaches. We also observe that multiple-voltage system, if the voltage levels are set properly, can indeed achieve energy reduction very close to the full potential by DVS.

Chapter 5

Energy-Efficient Dual-Voltage System with (m,k)-Firm Guarantee – A Case Study

In Chapter 4, we have formulated the voltage set-up problem and presented the practical solutions in order to achieve maximum energy saving of embedded systems. In this chapter, we conduct a case study. Specifically, we consider the low power design of soft real-time embedded systems. The soft real-time feature is captured by the (m,k)-firm deadline and the power/energy efficiency is achieved by using the dual-supply-voltage system.

5.1 Introduction

5.1.1 Motivation

Unlike hard real-time systems where deadlines must be met at all cost to avoid catastrophic consequences, soft real-time systems are characterized by their tolerance to occasional deadline misses. The targeted soft real-time applications, such as multimedia and electronic games among others, are often characterized by the repetitive process on periodically arriving inputs like voice samples or video frames, with their soft deadlines determined by the sample periods. The tolerance to deadline misses in these applications is largely due to the imperfect human visual/auditory systems.

Although the tolerance to deadline misses has been traditionally expressed as a maximum allowable loss percentage (or minimum completion ratio). A more accurate model, *(m,k)-firm deadline*, has been proposed recently to capture the timing constraint where at least m iterations in any window of k consecutive iterations meet their deadlines [43]. A *dynamic failure* occurs if the (m,k)-firm deadline is violated. Note that this not only gives a $\frac{k-m}{k}$ maximum loss rate, but also ensures that the deadline misses are adequately spaced to be acceptable. When m=k, the (m,k)-firm deadline becomes hard deadline. It has been shown to be a better measurement for the quality of service (QoS) provided by soft real-time systems [13, 43, 72, 110]. However, these studies focus on overloaded systems with traditional optimization goals such as reducing the dynamic failure and averaging response time.

Meanwhile, low energy consumption has emerged as one of the most important design objectives for many real-time embedded systems particularly battery-operated systems such as PDAs. Many energy-driven voltage scheduling algorithms have been developed in the past to reduce system's energy consumption while still meeting the hard timing constraints [90, 98, 101, 105]. But only recently has the energy reduction problem in soft real-time applications been discussed within the context of completion ratio guarantee [53]. In this chapter, we investigate how to leverage the (m,k)-firm deadline constraint to achieve energy efficiency on such soft real-time embedded systems that are normally not overloaded.

5.1.2 Problem and Contributions

We consider serving multiple applications (or input streams) on a dual-voltage system. Each stream consists of periodic real-time tasks with an (m_i, k_i)-firm deadline. Each task has an unknown execution time which is less or equal to a given worst case execution time (WCET$_i$) and a deadline that equals to its period. We seek to *determine the most energy efficient dual-voltage system that provides all the individual (m_i, k_i)-firm guarantees.*

By the term *most energy efficient*, we refer to that the average energy consumption per iteration, after a sufficiently large number of iterations, is minimized. The solution to this problem includes the values of voltages V_{lo} and V_{hi} (*the voltage set-up problem*), as well as an on-line scheduler that decides the voltage at which each task should be executed (*the voltage scheduling problem*).

To the best of our knowledge, this is the first work on energy reduction using dual supply voltages on soft real-time systems with (m,k)-firm deadlines. We formulate and solve the voltage scheduling and set-up problem by proposing

1. an on-line greedy scheduler that is provably the most energy efficient for dual-voltage system with (m,k)-firm guarantee;

2. a novel exact method that computes the average energy consumption per iteration for any dual-voltage system; and
3. a numerical method that is several magnitude faster than a simulation-based search, the only other way to solve the voltage set-up problem.

These results can be integrated into system design flow to implement energy efficient dual-voltage systems with (m,k)-firm deadline guarantee for multiple periodic streams.

5.1.3 A Motivational Example

It bas been long known that different voltage scheduling policies can result in very different energy savings on dual-voltage systems [24, 26, 106]. The following example shows the importance of the voltage set-up problem as being recognized recently [49].

Table 5.1: Characteristics of the iterations and the processor. (a): each entry shows execution time at V_1 and the probability this execution time occurs at run time. (b): *power* is normalized to the power at V_1 and *delay* column gives the normalized processing time to execute one iteration at different voltages.

case	execution time distribution		
I	(2, 90%)	(4, 9%)	(8, 1%)
II	(2, 1%)	(4, 90%)	(8, 9%)
III	(2, 1%)	(4, 1%)	(8, 98%)

(a) Application's execution time information.

voltage	power	delay
$V_1 = 3.3V$	1	1
$V_2 = 1.65V$	0.125	2
$V_3 = 0.825V$	0.016	4

(b) Processor's parameters.

Consider a system iteratively executing a periodic application stream. The application has a (1,2)-firm deadline and a period of 8. The possible execution times of each iteration are 2, 4, and 8. We consider three cases, as illustrated in Table 5.1(a), when the probabilities that those execution times occur are different. We can integrate up to two voltages from the set of {3.3V, 1.65V, 0.825V} onto the system. Table 5.1(b) gives the simplified power consumption and processing speed of the system at different voltages.

Table 5.2: The average energy consumption per iteration for systems with different voltage set-ups.

	Case I	Case II	Case III
A: 3.3V	8	8	8
B: 3.3V + shut-down	4	4	**4**
C: (3.3V, 0.825V)	**0.84**	4.04	4.04
D: (3.3V, 1.65V)	1.07	**1.58**	4.46

Table 5.2 reports the average energy consumption per iteration on four systems with different voltage set-ups. The "3.3V + shut down" system can also be treated as a special case of dual-voltage system where the second voltage is zero for shut down. System shut-down is not allowed for the other three settings. The energy figures of the best setting for the three different cases are shown in bold. They clearly indicate that voltage values must be selected carefully in order to achieve the most energy saving.

5.1.4 Previous Work

The (m,k)-firm deadline model was first proposed by Hamdaoui and Ramanathan for a more precise description of the maximum allowable deadline misses in overloaded soft real-time systems [43], where they developed a distance-based priority assignment scheme to reduce the probability of dynamic failure. This scheme was extended to deal with streams in networking traffic where messages traverse more than one hop in reaching their destination [72]. Bernat and Burns integrated the (m,k)-firm deadline constraint into dual priority scheduling to reduce the average response time of soft real-time tasks [13]. In [110], Ramanathan proposed a scheduling approach that partitions the tasks into mandatory and optional and provides deterministic (m,k)-firm guarantee to each task. Quan and Hu improved such partitions to better exploit the (m,k) constraints in overloaded systems [104]. They also gave a sufficient condition for the schedulability of a task set with arbitrary (m,k)-patterns.

Low power design using multiple supply voltages has been studied extensively in the past decade. Most work are on how to reduce power/energy consumption without sacrificing system's performance [24, 26, 49, 106]. Recently, there are plenty of approaches on how to trade performance (or QoS in general) for energy reduction. Qu and Potkonjak discussed how to partition the applications and allocate system resources to satisfy a given QoS requirement with minimum energy consumption [101]. Mossé et al. considered power-aware scheduling techniques at compiler and operating system levels for real-time applications [90]. Qiu et al. proposed a framework for the power management with guaranteed QoS in a distributed multimedia system [98]. Most recently, Hua et al. proposed scheduling algorithms to minimize the system's energy consumption while statistically meeting the completion ratio requirements [53].

Dual-voltage (and multiple voltage) systems have received special attention for practical reasons. There have been several reported studies on how to select the voltage levels on such systems to achieve energy efficiency. Raje and Sarrafzadeh [106] used dual-voltage (5.0V and 3.0V) and three-voltage (5.0V, 3.0V, and 2.4V) in their experiments [106]. Chang and Pedram used four levels (5.0V, 3.3V, 2.4V, and 1.5V) for no specific reasons [24]. Chen and Sarrafzadeh empirically studied dual-voltage system with 5.0V as high voltage and low voltage goes from 2.0V to 4.2V [26]. Hua and Qu first formulated the voltage set-up problem and provide analytic and numerical solutions with rather simplified assumptions [49].

5.1.5 Chapter Organization

The rest of the chapter is organized as follows. Section 5.2 presents our optimal solution to the voltage scheduling problem, where we give the most energy efficient on-line scheduler and analyze its competitive ratio. Section 5.3 focuses on our analytic and practical approach to the voltage set-up problem. We report the simulation results in Section 5.4 and conclude this chapter in Section 5.5.

5.2 Optimal On-Line Voltage Scheduling Policy

We only consider schedulers that meet the (m,k)-firm deadline. For a dual-voltage system with voltages (V_{lo}, V_{hi}), we seek for an on-line scheduler that provides a single stream's (m,k)-firm guarantee with the least amount of (average) energy consumption. We will show how our solution can be conveniently extended for multiple periodic streams at the end of next section. We necessarily assume that V_{hi}

is sufficiently high to guarantee a completion even when WCET occurs. Otherwise, if WCET happens $k - m$ or more times in k consecutive iterations, the (m,k)-firm deadline cannot be made.

Lemma 5.1. In any scheduler that minimizes the average energy consumption with the (m,k)-firm guarantee, high voltage V_{hi} is used only when there are exactly $k - m$ execution failures in the previous $k - 1$ iterations.

[Proof:] We prove this by contradiction. Let $\mathcal{S}$ be a most energy-efficient scheduler and the i-th iteration be the first time that $\mathcal{S}$ uses V_{hi} but there are less than $k - m$ failures in the previous $k - 1$ iterations. We now construct another on-line scheduler $\mathcal{S}'$ and show that it consumes less energy than $\mathcal{S}$.

First, define $\mathcal{S}'$ be identical to $\mathcal{S}$ in the first $i - 1$ iterations but use the low voltage V_{lo} for the i-th iteration. The voltage selection for iteration $(i + 1)$ and thereafter will be determined on the fly based on the execution status of the i-th iteration as follows:

Case I. $\mathcal{S}'$ completes iteration i with low voltage V_{lo}. Then we simply let $\mathcal{S}'$ makes the same voltage selection as $\mathcal{S}$. It will meet the (m,k)-firm deadline as long as $\mathcal{S}$ does with less energy consumption because if running V_{lo} at the i-th iteration.

Case II. $\mathcal{S}'$ fails to complete iteration i with low voltage V_{lo}. Define $\mathcal{S}'$ be the same as $\mathcal{S}$ for iterations $i+1, i+2, \cdots, j-1, j+1, j+2, \cdots$, where $j > i$ is the first iteration that $\mathcal{S}$ fails. For iteration j, $\mathcal{S}'$ selects V_{hi} if $j < i + k$ and V_{lo} otherwise. It suffices to show that $\mathcal{S}'$ meets the (m,k)-firm deadline and consumes less energy than $\mathcal{S}$.

When $j > i + k$, the only difference between $\mathcal{S}'$ and $\mathcal{S}$ is that $\mathcal{S}'$ fails to complete iteration i with V_{lo} but $\mathcal{S}$ completes it with V_{hi}. Clearly, $\mathcal{S}'$ is more energy efficient. To show that $\mathcal{S}'$ meets the (m,k)-firm deadline, we observe that the failure at iteration i may only affect whether the (m,k)-firm deadline is met for any k consecutive iterations between $i-k+1$ and $i+k-1$. From the hypothesis, $\mathcal{S}'$ (same as $\mathcal{S}$) has less than $k-m$ failures from $i-k+1$ to $i-1$, one more failure at i, and the next failure won't happen until iteration $j > i+k$. So there are at most $k-m$ failures between iterations $i-k+1$ and $i+k-1$, and hence the (m,k)-firm deadline will not be violated.

When $i < j < i+k$, $\mathcal{S}'$ will not violate the (m,k)-firm deadline before iteration j for the same reason. Because $\mathcal{S}$ fails iteration j with V_{lo} and $\mathcal{S}'$ completes it with V_{hi}, 1) from iteration i to $i+k-1$, $\mathcal{S}'$ and $\mathcal{S}$ have the same number of failures; 2) from iteration $i+k$ to $j+k-1$, $\mathcal{S}'$ has one failure less than $\mathcal{S}$; 3) from iteration $j+k$, the completion status of iteration j does not have any impact to the (m,k)-firm deadline anymore. This means that $\mathcal{S}'$ makes the (m,k)-firm deadline as long as $\mathcal{S}$ does. Recall that in this case, $\mathcal{S}'$ differs from $\mathcal{S}$ by using V_{lo} for iteration i and V_{hi} for iteration j, therefore, they have the same energy consumption. We mention that $\mathcal{S}'$ is statistically better because the failure at iteration i has less impact to the future than a later failure at iteration j. $\square$

Figure 5.1 outlines the On-Line Greedy scheduler based on Lemma 5.1, where we only need a counter n to track the number of execution failures in the previous $k-1$ iterations and make the voltage selection based on whether n has reached the threshold $k-m$.

We now perform the competitive ratio analysis of the On-Line Greedy scheduler. For any fixed voltage pair (V_{lo}, V_{hi}), let E_{lo} and E_{hi} be the average energy consumption per iteration at voltages V_{lo} and V_{hi} respectively, and l be the expected number of completed iterations at V_{lo} in k consecutive iterations, we have

Theorem 5.1. On-Line Greedy is the most energy-efficient deterministic on-line

```
On-Line Greedy scheduler:
1. n = 0;     /* initially no failure in the previous $k-1$ iterations */
2. for ($i = 1, 2, \cdots$)
3.     if ($n < k - m$)
4.         use $V_{lo}$ for the current iteration $i$;
5.         if (the current iteration fails)     $n = n + 1$;
6.     else
7.         use $V_{hi}$ for the current iteration $i$;
8.     if ( iteration $i - k + 1$ was failed )
9.         $n = n - 1$;        /* the failure on iteration $i - k + 1$ will not
                          affect the (m,k)-firm deadline after iteration $i$ */
```

Figure 5.1: The on-line greedy scheduler for (m,k)-firm guarantee.

scheduler that provides (m,k)-firm guarantee with a competitive ratio

$$1 + \frac{m \cdot (E_{hi} - E_{lo})}{k \cdot E_{lo}} \qquad \text{(if } l \geq m\text{)} \tag{5.1}$$

$$1 + \frac{l \cdot (E_{hi} - E_{lo})}{k \cdot E_{lo} + (m - l) \cdot (E_{hi} - E_{lo})} \qquad \text{(if } l < m\text{)} \tag{5.2}$$

[Proof]: Suppose that a scheduler chooses to operate at V_{lo} for n_{lo} iterations and V_{hi} for n_{hi} iterations without violating the (m,k)-firm deadlines, then the average energy consumption per iteration is

$$\bar{E} = \frac{E_{lo} \cdot n_{lo} + E_{hi} \cdot n_{hi}}{n_{lo} + n_{hi}} \tag{5.3}$$

Clearly, a scheduler is more energy efficient if it has more iterations running at voltage V_{lo}. On-Line Greedy scheduler selects the high voltage only when there have already been $k - m$ failures and the system cannot afford another failure due to the (m,k)-firm deadline. Lemma 5.1 indicates that this is the most energy efficient way. From the construction of the On-Line Greedy scheduler, we see that no other on-line deterministic scheduler will consume less energy.

The worst case for the On-Line Greedy scheduler happens when all the first $k - m$ iterations fail (at V_{lo}) and the system has to run at V_{hi} for the next m iterations to make the (m,k)-firm deadline. The total energy consumption is $(k - m) \cdot E_{lo} + m \cdot E_{hi}$. On the other hand, the best offline scheduler uses V_{hi} only when running at V_{lo} cannot produce m completions in any consecutive k iterations. Statistically, l completions are expected in any k consecutive iterations. If $l \geq m$, there is no need to use V_{hi} and the offline scheduler consumes energy $k \cdot E_{lo}$; if $l < m$, then $m - l$ iterations are expected to be processed at V_{hi} while the rest at V_{lo}, giving a total energy consumption of $(m - l) \cdot E_{hi} + (k - m + l) \cdot E_{lo}$. The competitive ratio of an on-line algorithm is defined as the ratio of its worst case performance over the best offline approach. Simple arithmetic operations give us Equations (5.1) and (5.2) as above. $\square$

5.3 Determining the Most Energy-Efficient Dual-Voltage System with (m,k)-Firm Guarantee

Theorem 5.1 shows that the On-Line Greedy scheduler is the most energy-efficient for any given dual-voltage system. However, the average energy consumption $\bar{E}$ depends on the values of V_{lo} and V_{hi} as we have seen in Equation (5.3). In this section, we solve the voltage set-up problem. That is, determining V_{lo} and V_{hi} to minimize $\bar{E}$. We first give efficient and accurate methods to compute $\bar{E}$ and then present our approach in finding V_{lo} and V_{hi} to minimize $\bar{E}$. We also explain how to generalize our results from single stream to multiple periodic streams with different (m,k)-firm requirements.

5.3.1 Computing $\bar{E}$ for Case $(k-1, k)$

For the i-th iteration, define $a_i = 0$ if the On-Line Greedy scheduler selects V_{lo} as the operating voltage; otherwise define $a_i = 1$. Therefore, we can represent the execution of n iterations by a bit stream of length n. Define $S_j = 00\cdots011\cdots1$ be a sequence of j 0's followed by $k-1$ 1's and S_0 be any sequence of zero or more 0's followed by no more than $k-2$ 1's.

Lemma 5.2. Any bit stream that represents the On-Line Greedy scheduler's voltage selection for $(k-1, k)$-firm guarantee can be uniquely decomposed to $S_{i_1}S_{i_2}\cdots S_0$, where $i_j > 0$.

[Proof:] The On-Line Greedy scheduler starts with V_{lo} (that is, $a_1 = 0$) and continues with V_{lo} until there is a failure at iteration i (that is $a_2 = a_3 = \cdots = a_i = 0$). To meet the $(k-1, k)$-firm deadline, the next $k-1$ iterations have to be processed at V_{hi}, $a_{i+1} = \cdots = a_{i+k-1} = 1$. This produces an instance of S_i with length $i + k - 1$. Then the On-Line Greedy scheduler starts with V_{lo} again yielding another sequence of 0's followed by $k-1$ 1's. The only exception occurs at the end of the execution: if the failure happens within the last $k-1$ iterations, we have S_0 which is a sequence of 0's followed by $k-2$ or less 1's; if there is no failure through the end, we will have S_0 which is a sequence of 0's only; if the last iteration happens to be last one of the $k-1$ 1's in the previous S_{i_j}, then we have an empty sequence S_0. ▯

Theorem 5.2. Let p_f be the probability that an arbitrary iteration can not be completed at low voltage V_{lo} before its deadline, then the On-Line Greedy scheduler's average energy consumption per iteration is

$$\bar{E} = \frac{E_{lo} + p_f \cdot (k-1) \cdot E_{hi}}{1 + p_f \cdot (k-1)} \tag{5.4}$$

[Proof:] Sequence S_i represents i iterations at V_{lo} followed by $k-1$ iterations at V_{hi}, therefore it consumes energy $E_i = i \cdot E_{lo} + (k-1) \cdot E_{hi}$. Any bit stream (of infinite length) satisfying the $(k-1, k)$-firm requirement can be uniquely decomposed to a sequences of S_i's. Let $P_{i_1 i_2 \cdots}$ be the probability that bit stream decomposition $S_{i_1} S_{i_2} \cdots$ occurs, then the On-Line Greedy scheduler's average energy consumption per iteration is

$$\bar{E} = \frac{\sum P_{i_1 i_2 \cdots} \cdot (E_{i_1} + E_{i_2} + \cdots)}{\sum P_{i_1 i_2 \cdots} \cdot [(i_1 + k - 1) + (i_2 + k - 1) + \cdots]} \tag{5.5}$$

where both sums are taken over all the possible bit streams that represents the On-Line Greedy scheduler's voltage selection decision.

Note that each S_i ends with $k-1$ iterations running at V_{hi} with guaranteed completions, therefore none of the previous failures can affect On-Line Greedy scheduler's voltage selection for the following iterations. In another word, the sequence S_i's are *independent* and the probability that each sequence S_i occurs will be $p_f \cdot (1-p_f)^{i-1}$. $\bar{E}$ from the above equation can be rewritten as:

$$\begin{aligned}
\bar{E} &= \frac{\sum_{i=1}^{\infty} p_f \cdot (1-p_f)^{i-1} \cdot E_i}{\sum_{i=1}^{\infty} p_f \cdot (1-p_f)^{i-1} \cdot (i+k-1)} \\
&= \frac{E_{lo} \sum_{i=1}^{\infty} i \cdot (1-p_f)^{i-1} + (k-1) \cdot E_{hi} \sum_{i=1}^{\infty} (1-p_f)^{i-1}}{\sum_{i=1}^{\infty} i \cdot (1-p_f)^{i-1} + (k-1) \sum_{i=1}^{\infty} (1-p_f)^{i-1}} \\
&= \frac{E_{lo} + p_f \cdot (k-1) \cdot E_{hi}}{1 + p_f \cdot (k-1)} \qquad \square
\end{aligned}$$

We now give two simple examples for Theorem 5.2. First, on a single voltage system, the energy consumption for each iteration will be identical and hence

$\bar{E} = E_{hi} = E_{lo}$. Considering this case as $V_{hi} = V_{lo}$, and $p_f = 0$, we get the same result from Equation (5.4). Now, if the system with single voltage V_{hi} can shut down for the entire iteration to save energy, the On-Line Greedy scheduler will repetitively shut down for the first iteration and then process for $k-1$ iterations. This results in $\bar{E} = \frac{k-1}{k} \cdot E_{hi}$. If we consider system shut-down as $V_{lo} = 0$ with $p_f = 1$ and $E_{lo} = 0$, then Equation (5.4) gives the same $\bar{E}$.

5.3.2 Computing $\bar{E}$ for Case (m, k)

We extend the concept of *independence* in the proof of Theorem 5.2 to compute On-Line Greedy scheduler's average energy consumption to meet the (m, k)-firm requirement. For iteration i, define string $b_i = v$ if we complete the i-th iteration and $b_i = x$ otherwise. A string that ends with m consecutive v's is called *independent*. m consecutive v's indicate that the last m iterations are all completions. So the On-Line Greedy scheduler can safely choose V_{lo} for the next $k-m$ iterations without causing any dynamic failure. After that, only the m completions followed by the $k-m$ new iterations, a total of k iterations, may impact the voltage decision in the future. The iterations priori to these m consecutive v's are in some sense blocked and cannot affect any iterations after these m consecutive completions. Similar to the discussion in the $(k-1, k)$ case, any string representing On-Line Greedy scheduler's decision can be decomposed to a series of independent substrings. We classify these independent substrings based on their first $k-1$ iterations. Let $E_{b_1 b_2 \cdots b_{k-1}}$, $L_{b_1 b_2 \cdots b_{k-1}}$, and $P_{b_1 b_2 \cdots b_{k-1}}$ be the expected energy consumption, the expected number of iterations of the independent substring starting with $b_1 b_2 \cdots b_{k-1}$ (to its end of m v's), and the probability that such substring may occurs respectively. Then we can use a formula similar to equation (5.5) to compute the average energy consumption per iteration $\bar{E}$. Now we explain how to obtain $E_{b_1b_2\cdots b_{k-1}}$, $L_{b_1b_2\cdots b_{k-1}}$, and $P_{b_1b_2\cdots b_{k-1}}$.

Theorem 5.3. Let $E'_{b_1b_2\cdots b_{k-1}}$ be the expected energy consumption to complete an independent substring starting with $b_1b_2\cdots b_{k-1}$ to its end, but does not include the first $k-1$ iterations. We have ($L'_{b_1b_2\cdots b_{k-1}}$ can be defined similarly)

$$E'_{xx\cdots vv\cdots v} = E_{hi} \tag{5.6}$$

$$E'_{b_1b_2\cdots b_{k-1}} = E_{hi} + E'_{b_2b_3\cdots b_{k-1}v} \qquad \text{(if } b_1b_2\cdots b_{k-1} \text{ has } k-m \ x\text{'s)} \tag{5.7}$$

$$E'_{b_1b_2\cdots b_{k-1}} = E_{lo} + p_f \cdot E'_{b_2b_3\cdots b_{k-1}x} + (1-p_f)\cdot E'_{b_2b_3\cdots b_{k-1}v} \tag{5.8}$$

[Proof]: If there are less than $k-m$ x's (failures) in $b_1b_2\cdots b_{k-1}$, On-Line Greedy scheduler selects V_{lo} for the current iteration (Step 4 in Figure 5.1) and consumes energy E_{lo}. When the execution fails with probability p_f, the latest $k-1$ iterations result in the substring $b_2b_3\cdots b_{k-1}x$ and requires in average $E'_{b_2b_3\cdots b_{k-1}x}$ energy to complete this independent substring; when we have a completion with probability $1-p_f$, the latest $k-1$ iterations become $b_2b_3\cdots b_{k-1}v$ and requires in average $E'_{b_2b_3\cdots b_{k-1}v}$ energy to complete this independent substring. This gives us Equation (5.8) for $E'_{b_1b_2\cdots b_{k-1}}$.

When there are already $k-m$ failures, the On-Line Greedy scheduler will select V_{hi} (Step 7 in Figure 5.1). This gives a completion and the required energy to complete this independent substring $E'_{b_2b_3\cdots b_{k-1}v}$ (Equation (5.7)). There is one special case: if there are $k-m$ x's (failures) followed by $m-1$ completions at V_{hi}, we need only one more iteration at V_{hi} to complete this independent substring as in Equation (5.6) $\square$

There are $\binom{k-1}{k-m}$ equations of type (5.6) and (5.7), $\binom{k-1}{k-m-1} + \binom{k-1}{k-m-2} + \cdots + \binom{k-1}{1} + \binom{k-1}{0}$ linear equations of type (5.8). One can solve this linear system to obtain $E'_{b_1 b_2 \cdots b_{k-1}}$. Adding the energy consumption for the first $k-1$ iterations $b_1 b_2 \cdots b_{k-1}$ gives us $E_{b_1 b_2 \cdots b_{k-1}}$. $L_{b_1 b_2 \cdots b_{k-1}}$ can be calculated in the same way and the probability $P_{b_1 b_2 \cdots b_{k-1}} = p_f^r \cdot (1-p_f)^t$, where r and t are the numbers of x's and v's, respectively, in $b_1 b_2 \cdots b_{k-1}$ before the first iteration that requires V_{hi}. We now use two examples to illustrate this approach.

Example 5.1. computing $\bar{E}$ for case $(k-1, k)$

There are k different types of independent substrings: $xvv \cdots v$, $vxv \cdots v, \cdots, vv \cdots vx$, and $vv \cdots v$. We have

$$\begin{aligned}
E'_{xvv\cdots v} &= E_{hi} \\
E'_{vxv\cdots v} &= E_{hi} + E'_{xvv\cdots v} \\
&\cdots \\
E'_{vv\cdots v} &= E_{lo} + p_f \cdot E'_{vv\cdots vx} + (1-p_f) \cdot E'_{vv\cdots v}
\end{aligned}$$

This can be easily solved and we have $E'_{vv\cdots v} = \frac{E_{lo}}{p_f} + (k-1) \cdot E_{hi}$ and $E'_{v\cdots xv\cdots v} = i \cdot E_{hi}$, where i is the index of the 'x' in $v \cdots xv \cdots v$. Note that each iteration before and including the 'x' is operated at V_{lo} and all the iterations after the 'x' are at V_{hi}, so we have

$$\begin{aligned}
E_{vv\cdots v} &= \frac{E_{lo}}{p_f} + (k-1) \cdot E_{hi} + (k-1) \cdot E_{lo} \\
E_{v\cdots xv\cdots v} &= i \cdot E_{hi} + i \cdot E_{lo} + (k-1-i) \cdot E_{hi} \\
\text{with} \qquad P_{vv\cdots v} &= (1-p_f)^{k-1} \\
P_{v\cdots xv\cdots v} &= p_f \cdot (1-p_f)^{i-1} \\
\text{and} \qquad L_{vv\cdots v} &= \frac{1}{p_f} + (k-1) + (k-1) \\
L_{v\cdots xv\cdots v} &= i + i + (k-1-i)
\end{aligned}$$

$$\bar{E} = \frac{P_{vv\cdots v} \cdot E_{vv\cdots v} + P_{xv\cdots v} \cdot E_{xv\cdots v} + \cdots P_{v\cdots vx} \cdot E_{v\cdots vx}}{P_{vv\cdots v} \cdot L_{vv\cdots v} + P_{xv\cdots v} \cdot L_{xv\cdots v} + \cdots P_{v\cdots vx} \cdot L_{v\cdots vx}}$$

$$= \frac{E_{lo} + p_f \cdot (k-1) \cdot E_{hi}}{1 + p_f \cdot (k-1)}$$

which is the same as we obtained earlier.

Example 5.2. computing $\bar{E}$ for case $(2,4)$

The independent substrings are partitioned into $\binom{3}{2} + \binom{3}{1} + \binom{3}{0} = 7$ groups based on their starting three iterations: $vvv, vvx, vxv, xvv, vxx, xvx, xxv$. We only list the equations corresponding to equations (5.6)-(5.8) and ignore the tedious calculation:

$$\begin{aligned}
E'_{xxv} &= E_{hi} \\
E'_{xvx} &= E_{hi} + E'_{vxv} \\
E'_{vxx} &= E_{hi} + E'_{xxv} \\
E'_{xvv} &= E_{lo} + p_f \cdot E'_{vvx} + (1-p_f) \cdot E'_{vvv} \\
E'_{vxv} &= E_{lo} + p_f \cdot E'_{xvx} + (1-p_f) \cdot E'_{xvv} \\
E'_{vvx} &= E_{lo} + p_f \cdot E'_{vxx} + (1-p_f) \cdot E'_{vxv} \\
E'_{vvv} &= E_{lo} + p_f \cdot E'_{vvx} + (1-p_f) \cdot E'_{vvv}
\end{aligned}$$

5.3.3 Determining the Optimal Dual-Voltage System

We consider a dual-voltage system that serves n real-time applications. Each application consists of a stream of periodic tasks with (m_i, k_i)-firm deadlines. Our goal is to determine the two supply voltages V_{hi} and V_{lo} at which the system consumes the minimum energy to process these streams without any dynamic failure. We assume that the execution time distribution of each stream is known a priori (e.g. by profiling), but do not require the real execution time of each iteration to be known.

The system uses Earliest Deadline First (EDF) to schedule the tasks from different streams. The utilization of the system at the reference (highest) voltage can be calculated by $U = \sum_{i=1}^{n} \frac{C_i}{T_i}$ [73], where C_i is the WCET and T_i is the period of the tasks in the ith stream. We necessarily assume that the tasks are schedulable at the reference voltage, i.e., $U \leq 1$; otherwise, dynamic failure becomes inevitable. For the same reason, the high voltage V_{hi}, if different from the reference voltage, should provide processing speed sufficiently fast such that the tasks are schedulable. On the other hand, let C_i' be the WCET of stream i and U' be the utilization of the system at V_{hi}. If $U' < 1$, we can reduce V_{hi} to save energy while keeping the tasks schedulable. Therefore, the high voltage V_{hi} is selected such that the system utilization at V_{hi} is exactly 1. Specifically, when $U < 1$ at the reference voltage V_{ref}, V_{hi} can be obtained by solving the following equation:

$$\frac{1}{U} = \frac{V_{hi}}{(V_{hi} - V_{th})^2} \frac{(V_{ref} - V_{th})^2}{V_{ref}} \tag{5.9}$$

where V_{th} is the threshold voltage.

Because the streams are independent each other, the proposed On-Line Greedy scheduler can be used for each stream in order to minimize the energy consumption. To determine the best value of low voltage V_{lo}, we follow the following procedure:

1. for each stream, compute $p_{f,i}$, the probability of execution failure at a fixed V_{lo}, from the execution time distribution of the ith stream;

2. for each stream, compute On-Line Greedy scheduler's average energy consumption per iteration $\bar{E}_i$, which is a single-variable function of V_{lo};

3. let LCM be the least common multiple of all the periods $(T_1, \cdots, T_n)$, the energy consumption during an LCM period (hyperperiod) is

$$\bar{E} = \sum_{i=1}^{n} \bar{E}_i \cdot \frac{LCM}{T_i} \tag{5.10}$$

4. (numerically) find the value of V_{lo} to minimize $\bar{E}$.

5.4 Simulation Results

Our simulation goals include: 1) verifying that On-Line Greedy is the most energy efficient scheduler; 2) demonstrating the energy efficiency of dual-voltage systems and the importance of voltage set-up; 3) validating our solutions to the voltage set-up problem; and 4) investigating the impact of parameters, such as m, k, WCET, and BCET (Best Case Execution Time).

We first consider dual-voltage systems that provide (m,k)-firm deadline guarantees to a single stream. The period tasks in the stream have execution time between BCET and WCET following normal distribution. The high voltage V_{hi}, at which level the system completes the WCET exactly on the deadline, is assumed to be the reference voltage 3.3V. The energy consumption is normalized in the unit of power dissipation in one unit of CPU time at 3.3V.

To show the optimality of our On-Line Greedy scheduler, we simulate an arbitrary on-line scheduling policy by the following *p-random* scheduler: for each iteration, using V_{hi} with a probability p unless that there are k-m failures in the previous k-1 iterations, in which case we use V_{hi} to guarantee the (m,k)-firm deadline. Clearly, our On-Line Greedy is 0-random. Figure 5.2 compares the On-Line Greedy with other p-random on-line schedulers, where p goes from 0 to 1 with an increment of 0.1, for (5,8)-firm deadline. For each p, we vary V_{lo} from 1.4V, the lowest voltage to complete BCET within the deadline, to 3.3V with an increment of 0.05V and simulate 100,000 iterations to obtain the average energy consumption per iteration. We see that On-Line Greedy consumes the least energy in all cases and the closer V_{lo} is to the best voltage setting, the more energy On-Line Greedy saves.

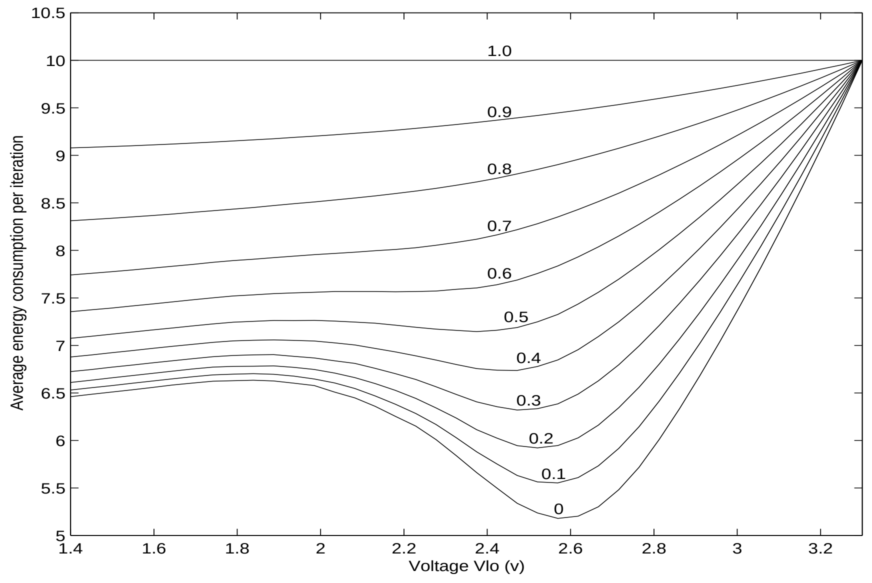


Figure 5.2: On-Line Greedy vs. *p-random* on-line schedulers.

Figure 5.2 also indicates the impact of the selection of V_{lo} to the system's energy efficiency. We now verify the accuracy of our method to calculate the average energy per iteration $\bar{E}$ as well as the efficiency of this method. Figure 5.3 depicts the values of $\bar{E}$, obtained by the proposed numerical method and a pure simulation based approach, for the (5,8)-firm guarantee with different BCET/WCET ratio. We see that the two methods give almost the same $\bar{E}$ (the difference is less than 0.5%). However, they have a huge discrepancy in run-time. While it takes more than 80 minutes for the pure simulation method to get a stable solution for each setting of V_{lo} and a BCET/WCET ratio, our numerical method needs less than one second on the same UNIX machine.

Both Figure 5.2 and Figure 5.3 show the energy efficiency of dual-voltage systems over fixed-voltage systems. In Figure 5.2, the $p = 1.0$ line, which means running at V_{hi} with probability 1, corresponds to the fixed 3.3V system without

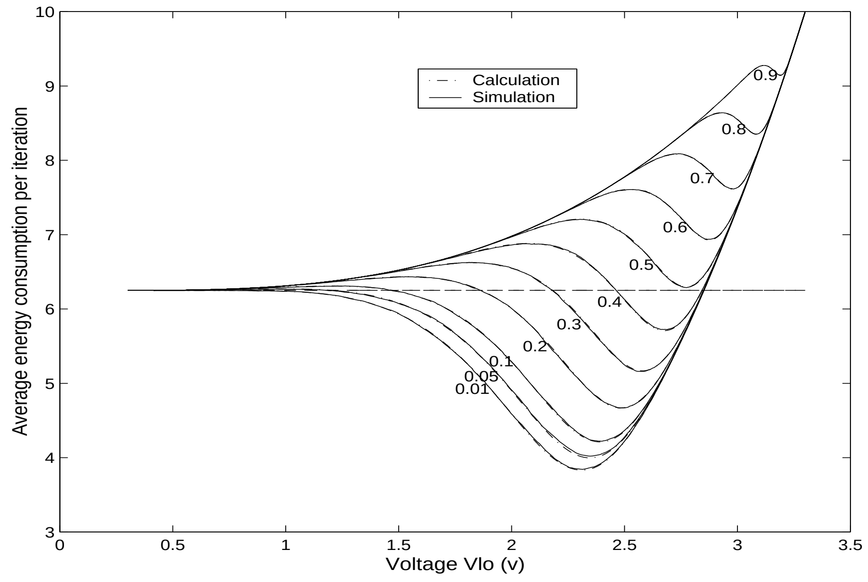


Figure 5.3: Accuracy of the proposed numerical method in computing the average energy consumption per iteration $\bar{E}$.

shut down. The dashed horizontal line in Figure 5.3 gives the average energy consumption $\bar{E}$ of the fixed 3.3V system with shut down. However, they also reveal that if V_{lo} is not selected properly, the full potential of dual-voltage system's energy saving may not be reached. For example, when BCET/WCET=0.1, setting $V_{lo} = 2.40V$ saves **32.6%** energy over the fixed-voltage system with shut-down and **57.9%** when system shut-down is not allowed. But $V_{lo} = 3.0V$ gives only 26.3% saving over the 3.3V system without shut-down and consumes 17.9% more energy than the 3.3V system with shut-down. In all the cases, the approach proposed in Section 3.3 finds the best V_{lo}. We will further demonstrate the correctness of this approach in Figure 5.4 for multiple application streams.

Table 5.3 reports the impact of m, when k is fixed, to the average energy consumption $\bar{E}$ for (m,k)-firm guarantee on the best dual-voltage system obtained by our method (E_1) and the fixed-voltage system with shut-down (E_2). As expected,

Table 5.3: Impact of different m to the selection of V_{lo} and $\bar{E}$.

	(1,8)	(2,8)	(3,8)	(4,8)	(5,8)	(6,8)	(7,8)
$V_{lo}(V)$	1.36	1.36	2.35	2.46	2.56	2.70	2.85
E_1	1.69	2.88	3.89	4.50	5.16	6.03	7.28
E_2	1.25	2.5	3.75	5	6.25	7.5	8.75
E_1 vs. E_2 saving(%)	-35.2	-15.1	-3.81	10.1	17.4	19.6	16.8

E_2 increases linearly with m. But E_1 increases at a much slower pace because the optimal value for V_{lo} gradually increases at the same time, which increases the number of completions at V_{lo}. We conclude that fixed-voltage system with shutdown is preferable for small m and dual-voltage system is more energy efficient when m is large.

Note that small m for fixed k implies less number of completions required for the (m,k)-firm guarantee. In such case, it is more beneficial to operate at V_{hi} to complete m iterations and then shut down, than to try running greedily at V_{lo} for most of the time. This applies to the similar situation when k is large and m is fixed as one can see from Table 5.4.

Finally, we show the energy efficiency of dual-voltage system serving multiple streams with different (m_i, k_i)-firm deadlines as given in Table 5.5. It takes only several seconds for our proposed approach to find the best voltage set-up $\{V_{hi} = 3.07V, V_{lo} = 2.41V\}$, which has energy consumption 12.77. We then simulate 100,000 hyperperiods on Matlab on dual-voltage systems with different voltage set-ups $\{V_{lo}, V_{hi}\}$, where V_{lo} goes from 1.0V to 3.3V and V_{hi} goes from 3.07V to

Table 5.4: Impact of different k to the selection of V_{lo} and $\bar{E}$.

	(5,6)	(5,7)	(5,8)	(5,9)	(5,10)	(5,11)	(5,12)
$V_{lo}(V)$	2.79	2.66	2.57	2.51	2.47	2.42	2.37
E_1	7.01	5.85	5.17	4.72	4.38	4.12	3.89
E_2	8.33	7.14	6.25	5.56	5.00	4.55	4.17
E_1 vs. E_2 saving(%)	15.8	18.1	17.2	15.1	12.3	9.44	6.55

Table 5.5: Information on three periodic streams.

Stream	BCET	WCET	Period	(m_i, k_i)
A	0.6	2	8	(5,8)
B	1.0	2	6	(6,7)
C	0.6	4	12	(2,3)

3.3V, both with an increment of 0.01V. After more than 10 hours of simulation, this exhaustive search finds the same solution (Figure 5.4), within the precision of voltage increment 0.01V we set, as the proposed approach.

5.5 Conclusions

In this chapter, we address the voltage scheduling and set-up problem for soft real-time dual-voltage systems that serve multiple streams with (m_i, k_i)-firm guarantee in order to minimize the energy consumption. First we propose an on-line

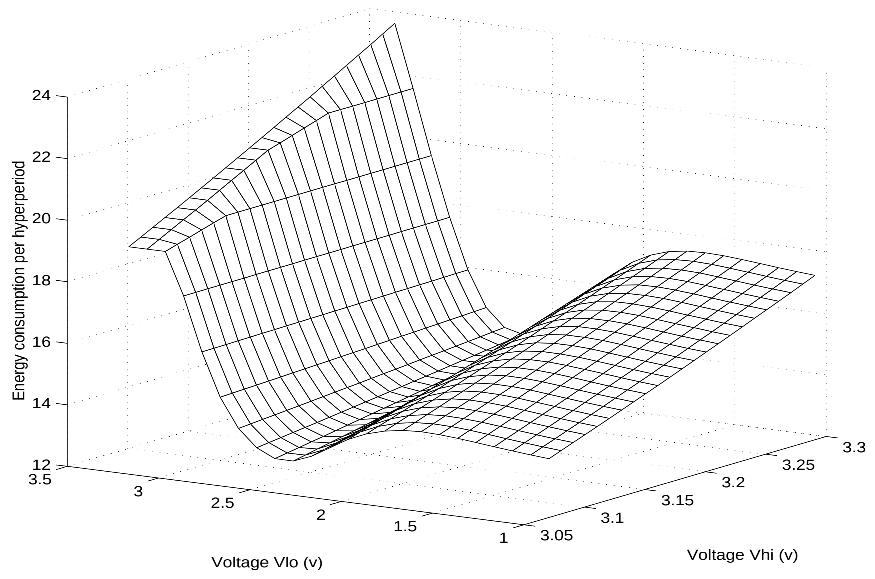


Figure 5.4: Simulated $\bar{E}$ for three applications with different (m,k)-firm requirements.

greedy scheduler which we prove is the most energy efficient deterministic on-line scheduler. Based on this scheduler, given the execution time distribution of the iteration and voltage levels, we present a novel energy calculation method which can fast and accurately obtain the average energy consumption per iteration for dual-voltage systems. Simulation results show that compared with the simulation method, the proposed calculation method can save significant CPU execution time while still obtaining very precise energy consumption value. Based on this energy calculation method, we have proposed a numerical approach to linearly search for the best voltage set-up. Simulation shows that different voltage set-ups give significantly different energy savings and the best set-up obtained from simulation coincides with our numerical solution.

Chapter 6

QoS-Driven Scheduling for Firm/ Soft Real-Time Applications

6.1 Introduction

With the increasing popularity of real-time multimedia and wireless communication applications, quality of service (QoS) attracts a lot of attention. Providing the required QoS guarantees becomes vital for the design of embedded systems that carry out such applications. The most popular way to specify time-related QoS requirements, such as synchronization and latency, is deadline. In hard real-time systems such as most control systems, deadlines are hard in the sense that missing deadline will cause fatal errors of the system. However, as the application-driven system design keeps on pushing for high performance, low energy consumption, light weight and high portability among others, it becomes difficult to meet these more and more system resource demanding QoS requirements. For example, one would like to view high-resolution movies one after another on a DVD player, but it cannot be done without recharging the battery. Consequently, we have soft real-time systems such as multimedia systems, where the deadlines can be either firm or soft. Firm deadlines are timing constraints that must be satisfied in order for the system to get rewards. Missing soft deadlines, on the other hand, still can bring system some rewards if the deadline failures are within an acceptable range. For instance, many MPEG video applications such as videoconferences require reliable communication and consistently high throughput, while being able to tolerate reasonable amount of packet error, jitter, or unsynchronization. Soft deadlines can also be found in many other applications such as web browsing and file transfer.

Task completion ratio [11, 19], which is equal to the percentage of completed tasks over all the requested tasks, has been widely used to measure QoS. However, it does not capture the firm/soft deadlines and data dependency that presents in many real-time applications. Therefore, it cannot accurately reflect the user perceived quality of presentation (QoP), which can be conveniently measured by the correctly completed tasks over the total tasks. (Note that due to the data dependency, some completed tasks may not be correct. For example, in MPEG decoding, B frames cannot be decoded correctly if the previous I/P frame has error.). This leads us to a new QoS metric to which every task completed before its deadline contributes, and every task completed after its soft deadline also contributes but subject to a penalty for missing its (soft) deadline. The new QoS metric decreases on any task drop according to the (dependent) tasks that may be affected. Putting these together, we define QoS as a weighted sum of the reward for completed tasks, the penalty for completing tasks after their soft deadlines, and the penalty for dropped tasks.

In this chapter, we first show that our new QoS metric describes QoP more accurately than the completion ratio metric. Then we modify several widely used on-line algorithms such as EDF (Earliest Deadline First), FCFS (First Come First Serve) and LETF (Least Execution Time First) [34] by replacing completion ratio with our new QoS metric. This results in better QoS and QoP. However, the QoP remains far below the system's computation power. That is, a significant portion of the completed tasks are computed incorrectly due to the factors such as data dependency. We then develop a new on-line scheduling algorithm, i.e., important task (frame) first, to improve the QoP. This scheduler makes on-line decisions based on our new QoS metric and achieves QoP very close to the system's computation power. It has the same run-time complexity as other on-line schedulers and therefore can be easily integrated into embedded systems to deliver better QoS or to provide the same QoS with less system resource (CPU, power, memory, etc.).

The rest of the chapter is organized as follows. In Section 6.2 we define the new quantitative QoS metric and show that it describes user perceived QoP better than the completion ratio on MPEG movies. Section 6.3 presents our on-line QoS-driven scheduling policies based on a drop lemma. In Section 6.4, we apply the general discussion to simulated MPEG movies and demonstrate that these simple scheduling algorithms are effective in improving not only our defined QoS but also the user perceived quality of presentation over classic scheduling policies such as EDF. We conclude this chapter in Section 6.5.

6.2 A New QoS Metric

6.2.1 QoS Model

We consider a system processing real-time applications. Each application consists of a sequence of tasks, and each task is characterized by $< a, d, e, f/s >$, where a is the arrival time, d is the deadline, e is the execution time which can be obtained a priori by pre-simulation or predicting, and f/s specifies whether the deadline is firm or soft.

- A task has a *firm* deadline if it must be completed before the deadline otherwise the system will not get the reward for serving the task and the application.
- A task has a *soft* deadline if the system can still benefit even if the deadline is missed, subjected to a deadline-miss penalty.
- A task is *non-preemptive* means that once the task gets the CPU, it will occupy the CPU until its deadline or completion, whichever comes earlier.
- A task is *preemptive* means that the task may lose control of the CPU during its execution, but when it gets the CPU back, it can resume the interrupted execution.

An online scheduler will allocate system resource to process the task it selects. The completion ratio is defined as the ratio of completed tasks over the total number of tasks according to the given scheduler. Although it has been widely used in real-time embedded systems, completion ratio may not give an accurate measure for the QoS due to the following reasons: 1) it does not distinguish the completion of tasks with firm deadlines and those with soft deadlines, on which the system may provide different QoS and get different rewards; 2) it does not distinguish tasks which are completed before their soft deadlines and those that are completed but miss their deadlines; and 3) it does not reflect data dependency among tasks because all deadline misses are treated in the same way. Based on these observations, we define our new QoS as follows:

Suppose that a scheduler S completes K_f firm-deadline tasks and K_s soft-deadline tasks out of a total of N tasks, the QoS provided by such scheduler is:

$$Q(S) = \frac{\alpha_s K_s + \alpha_f K_f}{N} - \frac{\beta}{N} \sum \frac{\delta_i}{d_i - a_i} - \frac{\gamma}{N} \sum 1_i \Delta_i \tag{6.1}$$

where α_s and α_f are the weights for the completion of soft-deadline tasks and firm-deadline tasks respectively (in general, $\alpha_s < \alpha_f$), β is the penalty parameter or the tolerance factor for deadline missing; δ_i is the difference between the task's deadline and completion time when the soft deadline is missed (if the task is completed before its deadline or eventually dropped, then δ_i is 0); $d_i - a_i$ is the life time of the task; γ is penalty parameter for task dropping; $1_i = 1$ if the i-th task is dropped, otherwise $1_i = 0$; Δ_i is the number of tasks that will be affected by the i-th task. In (6.1), the first term rewards task completion; in the other terms, the first sum is taken over all the completed tasks that miss their soft deadlines; and the second sum is taken over all the dropped tasks regardless of their deadline type.

The QoS defined in (6.1) is a direct extension of completion ratio, in the case when there is no penalty for missing soft deadlines ($\beta = 0$) or dropping tasks (γ =0) and firm deadline tasks are considered equally important as soft ones ($\alpha_s = \alpha_f = 1$), which has been used for QoS measurement in many occasions. Soft deadlines and firm deadlines are treated differently by assigning them different weights α_s and α_f. Soft deadline missing is penalized by the relative amount that the deadline has been missed with the penalty factor β. Data dependency is captured in the last term by reducing QoS in the amount of tasks depending on dropped tasks with a penalty factor γ.

From (6.1), we can see that the completion of the firm-deadline task will get more reward than the completion of the soft-deadline task before its deadline if $\alpha_f > \alpha_s$. Furthermore, their deadline misses lead to different rewards. For the firm-deadline task, the system will get no reward and even negative reward because of its deadline missing. However, for the soft-deadline task, the system may still have positive reward. Therefore, in order to maximize Q(s) as defined in (6.1), the system prefers to execute the firm-deadline task than execute the soft-deadline task. This exactly matches the fact that the firm deadline task is more important.

6.2.2 Simulation of MPEG Streams

We have implemented several widely used on-line algorithms such as EDF, FCFS and LETF and tested these algorithms on MPEG video streams decoding at the frame level. In the simulation we compare the completion ratio (CR), which only consider the number of completed frames, our proposed new QoS metric, and user perceived quality of presentation (QoP), which can be conveniently measured by the number of correctly decoded frames by using different online schedulers. Our objective is to demonstrate that our new QoS metric reflects user perceived QoP much better than the completion ratio.

Standard MPEG encoders generate three types of compressed frames: I frames (intra-pictures), P frames (predicted pictures) and B frames (bi-directional predicted pictures). In general, encoders use a fixed GOP (Group of Pictures) pattern when compressing a video sequence. A typical GOP in display order and decoding order is shown as in Fig. 6.1.

0	1	2	3	4	5	6	7	8	9	10	11	12	
I_0	P_1	B_2	B_3	P_4	B_5	B_6	P_7	B_8	B_9	I_{10}	B_{11}	B_{12}	decoding order
I_0	B_2	B_3	P_1	B_5	B_6	P_4	B_8	B_9	P_7	B_{11}	B_{12}	I_{10}	display order

Figure 6.1: A typical GOP pattern (I-to-I=12, I-to-P=3).

On average, I frames are the largest in size (since they are self-contained), followed by P frames and B frames. Krunz and Tripathi present a comprehensive model for MPEG video streams [66]. This model captures the bit-rate variations at multiple time scales. Long-term variations are captured by incorporating scene changes, which are noticeable in the fluctuations of I frames. Three models are introduced to simulate the frame sizes of different types of frames, and the complete model is finally obtained by intermixing these three sub-models according to a given GOP pattern. Statistically, the generated MPEG streams fit the empirical video and are sufficiently accurate in predicting the queuing performance for real video streams. We simulate the frame information for movies, Wizard of OZ, Star Wars, Silence of the Lambs, and Goldfinger, from the parameters reported in [66].

Based on the frame size and type, we generate the normalized execution time for each frame using a linear model of MPEG decoding [12]. In the simulation we assume that the execution time of MPEG-decoding can be obtained a priori by predicting based on the information from previously decoded frames and the size and type of MPEG-encoded frames [12]. Furthermore, in some scenario such as the voice-on-demand scenario we can get the exact information about the execution time directly from the user-data fields in the stream [16]. We also assume that the frames arrive in the decoding order and their inter-arrival times are independent with exponential distribution. The mean of the exponential distribution is approximately equal to the reciprocal of frame display rate (in terms of fps or frame per second) to generate a balanced loaded system. We simulate underloaded and overloaded systems by varying the fps requirement. The absolute deadline of each frame is monotonically increasing in its arrival time. We use several standard display rates (in terms of fps) in our simulation: 15, 30 (standard for computer video and graphics), 45 and 60 (suitable to sports and other fast-action programs). The deadline type is assigned to each individual frame based on the dependency of different frames. I frame is the most important because the correct processing of all the P frames and B frames in the same GOP depends on the completion of the corresponding I frame. P frame is also important because it is required by the following P and B frames in the same GOP. We assign I and P frames firm deadlines rather than giving them soft deadlines. We also assign soft deadlines to B frames to create tasks with mixed type of deadlines.

Each GOP can be viewed as one "application" independent of others as the correct decoding of all the frames in one GOP depends on the leading I frame. Each "application" consists of a set of tasks (frame decoding) and the drop of firm-deadline I and P frames will cause the incorrect decoding of the remaining frames in this "application". To better model the data dependency among "tasks", we assign different values Δ_I and $\Delta_{P,i}$, which are corresponding to the number of frames that will not be decoded correctly because of a dropped frame, to frames with firm deadlines. For example if I-to-I, the number of frames between two consecutive I frames (see Fig. 6.1), is 12, then we assign Δ_I =11; $\Delta_{P,i}$ are assigned 10, 7, and 4 for the three P frames in the GOP pattern based on Fig. 6.1; and $\Delta_B = 0$ because there is no frame depends on the B frame. As a result, I frames have higher priority than P and B frames; P frames have higher priority than B frames. This exactly matches the MPEG decoding mechanism. In sum, we use the following QoS, based on formula (6.1) with consideration of MPEG application's characteristics, in our simulation:

$$Q_{MPEG}(S) = \frac{K_s + K_f}{N} - \frac{\beta}{N}\sum_{i=1}^{K_s}\frac{\delta_i}{T_d} - \frac{\gamma}{N}(m_I\Delta_I + \sum_{i=1}^{n_P} m_{P,i}\Delta_{P,i}) \qquad (6.2)$$

Where T_d – the reciprocal of frame display rate;

Δ_I, $\Delta_{P,i}$ – the number of tasks that will be affected if the I frame or P frame is dropped;

m_I, $m_{P,i}$ – the number of dropping I, P frame;

n_P – the number of P frames in a GOP pattern;

K_S, K_f, β, γ, δ_i, N are same as in (6.1).

Note that this QoS measurement is calculated incrementally at run time and there are only a few arithmetic operations involved at each frame. The penalty parameter β and γ are *stream specific*. For example, the β and γ for decoding Cartoon Video (e.g., 0.8) should be smaller than those for decoding Action Video (e.g., 1.0) because the human being are less sensitive to the artificial movements in Cartoon Video, but are very sensitive to the smoothness of the motions in Action Video [92]. The values of these parameters can be stored as user-defined data within the stream. In the simulation, β and γ are both set to be the default value 1.

We have applied the popular online scheduling algorithms such as EDF, FCFS and LETF to the simulated MPEG movies. For each movie, we simulate under-loaded, balanced, and overloaded systems by changing the frame rate from 15 fps, to, 30, 45, and 60 fps. And for each case, we consider the case of non-preemptive and preemptive. Fig. 6.2 is the typical relationship of completion ratio, our proposed new QoS metric and user perceived QoP, which considers the actual number

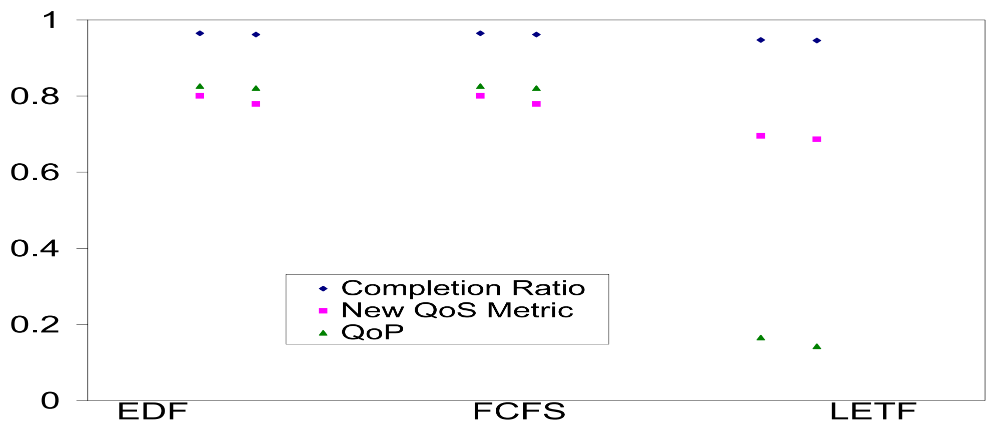


Figure 6.2: Comparison of some widely used online schedulers on movie "Goldfinger" in the frame rate of 30 fps in the case of, from left to right, non-preemptive and preemptive.

of correctly decoded frames, under different online scheduling policies (EDF, FCFS and LETF) on movie "Goldfinger" in the frame rate of 30fps. EDF and FCFS in our simulation are actually same because the system has monotonic absolute deadlines. From Fig. 6.2 we can see that the completion ratios under different schedulers are almost same, whereas the QoPs are very different. We can conclude that the completion ratio does not measure QoP properly and it cannot test different online schedulers. However, our new QoS metric is much closer to the QoP and it is necessary to develop low overhead online scheduler to maximize this new QoS metric in order to eventually improve user perceived QoP without using extra hardware.

In the next section, we consider the following QoS-driven online scheduling problem: *for a set of real-time tasks with mixed firm and soft deadlines on a single processor system, determine an online schedules S to maximize Q(S).*

6.3 Online Schedulers

Due to the uncertainty of the arriving tasks and the nature of online scheduling, it becomes unavoidable to drop tasks and hard to provide absolute QoS guarantees. Our objective is thus to develop online scheduling algorithms that give competitive average QoS. An online scheduling policy must have low complexity because it will be executed frequently on the fly. It should also specify its drop policy as the task drop becomes inevitable. In this section, we first give the drop lemma and then explain a set of online scheduling heuristics based on the widely used EDF, FCFS and LETF.

Lemma 6.1 (Drop Lemma):

If a scheduler (online or offline) maximizes the QoS as defined in Equation (6.1), then it must

1) drop task $< a, d, e, f >$ at time $t > d - e^*$

2) drop task $< a, d, e, s >$ at time $t > \frac{\alpha_s + \gamma \cdot \Delta}{\beta}(d - a) + (d - e^*)$

where e^* is the task's remaining execution time, and $e^* = e$ for non-preemptive tasks.

[Proof:] At time t, the earliest time that we can complete task $< a, d, e, f/s >$ is $t + e^*$, where e^* is the task's remaining execution time. If the task has a firm deadline d, it cannot be completed and will not contribute for QoS at time t when $t + e^* > d$. If the task has a soft deadline, we will execute it if and only if the benefit of completion (with deadline missing penalty if applicable) exceeds the penalty for dropping the task, that is, $\alpha_s - \beta\frac{\delta}{d-a} \geq -\gamma \cdot \Delta$, where $\delta = t + e^* - d$. A simple calculation leads us to 1) and 2) as above.

Intuitively, Drop Lemma suggests us to drop firm-deadline tasks as soon as we discover that we are unable to finish on time. However, for soft-deadline tasks, Drop Lemma implies that we should wait an extra period because soft deadline miss will still be beneficial to some extent. Clearly, the smaller is the deadline missing penalty parameter β, the larger is the weight of completion and drop penalty, the longer we should wait.

6.3.1 S2F: Soft to Firm Deadline Conversion

From Drop Lemma, we see that task $< a, d + \frac{\alpha_s + \gamma\cdot\Delta}{\beta}(d-a), e, f >$ and task $< a, d, e, s >$ will always be dropped at the same time although they have different type of deadlines. Based on this we propose the following online scheduler:

```
Algorithm S2F:
(1) For each soft deadline task $< a, d, e, s >$
(2)     change its deadline from d to $d + \frac{\alpha_s + \gamma\cdot\Delta}{\beta}(d-a)$;
(3)     change its deadline type from soft to firm;
(4) apply EDF on the new set of firm real-time tasks;
```

It converts soft deadline to firm and thus unifies task's deadline type. Its advantage is that online scheduling algorithms do not need to treat different types of deadlines. Moreover, the Drop Lemma shows that whenever EDF achieves the best QoS, S2F also gives the best QoS.

6.3.2 EDF*, FCFS* and LETF*

The EDF, FCFS and LETF service strategies are among the most popular ones for real-time applications. On the completion of one task, they aggressively schedule the next task with the earliest deadline, the earliest arrival time and the least execution time respectively. However, neither of them distinguishes firm deadlines and soft deadlines and they may decide to execute the task that should be dropped according to the Drop Lemma. We integrate the Drop Lemma into these three scheduling policies and propose scheduling algorithms EDF*, FCFS* and LETF*.

Algorithm EDF, FCFS* or LETF*:*

```
(1) On the completion of the current task $\tau$ or on the arrival of a new task if preemption is allowed)
(2) if preemption is assumed
(3)     replace the execution time of task $\tau$ by its remaining execution time;
(4) drop all the tasks that meet the condition in Drop Lemma;
(5) schedule the remaining tasks using EDF, FCFS or LETF.
```

Non-preemptive execution stops only at the completion of the current task. We are guaranteed that this completion will either meet the task's deadline or still gives positive contribution to the QoS even its soft deadline is missed. The reason is that the current task is the winner of all the tasks in the previous round, which mean it survives the drop policies. During the drop policy checking in step 4, unlike the original schedulers, EDF*, FCFS* and LETF* will treat firm and soft deadline tasks differently to maximize QoS. Finally, we argue that the drop policy checking takes only constant time. For example, in the implementation, we can first choose the task picked by EDF, FCFS or LETF and check whether it meets the drop policies. If the Drop Lemma is satisfied, we drop the task and ask EDF, FCFS or LETF for their next choice. Therefore, EDF*, FCFS* or LETF* will have the same run-time complexity as the original one.

6.3.3 IFF: Important Task (Frame) First

From Equation (6.1), we see that missing firm deadline immediately erases the efforts that we have already put on the task completely. However, when we miss the soft deadline, we still get the chance to improve the QoS by finishing the task in a reasonable amount of extra time. Thus, for the point view of maximizing QoS, we should assign tasks with firm deadline higher priority than those with soft deadlines. The IFF online scheduling algorithm is a variation of EDF based on this observation.

Algorithm IFF:

```
(1) On the completion of the current task $\tau$ (or on the arrival of a new task if preemption is allowed)
(2) if preemption is assumed
(3)     replace the execution time of task $\tau$ by its remaining execution time;
(4) drop all the tasks that meet the condition in Drop Lemma;
(5) select the task $\tau'$ with the earliest deadline in the ready list;
(6) if $\tau'$ is not the most important task in the ready list
(7)     check the drop policy at time t= current time + execution time of task $\tau'$ ;
(8)     if there is a more important task drop, unselect $\tau'$ and goto step 5;
(9) schedule the current pick;
```

IFF is similar to EDF* with special treatment to important tasks such as firm deadline tasks. In particular, if the task with earliest deadline is a soft, we check whether there will be any firm deadline task dropping because we execute this soft deadline task first. In another word, a task with soft deadline will be processed only if its execution will not cause any firm-deadline task drops. Furthermore, among the tasks with the same deadline type, there may exist data dependency. Therefore IFF also prioritizes certain same deadline tasks that potentially contribute more to the QoS measurement. The complexity of IFF is approximately the same as EDF*.

6.4 Experimental Results

We have implemented the proposed QoS-driven online schedulers and applied them to a set of simulated MPEG movies [66]. The setup of the simulation on MPEG movies is same as that in Section 6.2.2. In this section we report the simulation results.

We applied the proposed online scheduling algorithms to the simulated MPEG movies under different frame rates and different preemptive types. For underloaded system with a frame rate of 15fps, the deadlines are relatively loose and we observe that almost all the algorithms achieve the maximal QoS and QoP in the amount of 1 without the task drop and deadline missing. However, when the computation load increases, the system becomes balanced and overloaded eventually. Then we see, for instance in the movie of "Goldfinger" as shown in Fig. 6.3, different online schedulers provide very different QoP which have the same trends as the new defined QoS metrics. In general we can rank them in the increasing order of QoS: LETF*, EDF, EDF*, S2F, and IFF. When the system goes to overload state (such as 45 fps and 60 fps), the algorithm IFF achieves significant higher QoS and user perceived QoP comparing to other algorithms in both preemptive and non-preemptive cases.

It is of our particular interest to study overloaded systems where task drop and deadline missing become unavoidable. Fig. 6.4-6.7 give the detailed reports on the

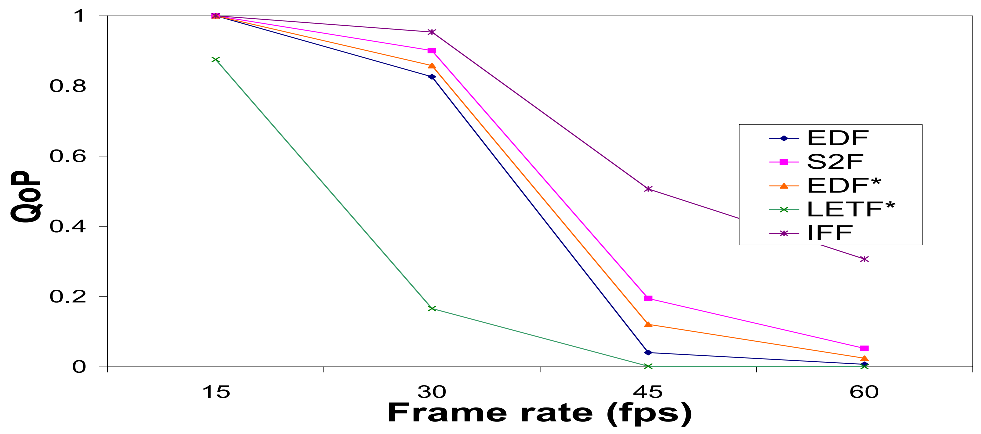


Figure 6.3: Comparison of QoP under different online schedulers on movie ”Goldfinger” in the case of non-preemptive with different frame rates (15, 30, 45, 60 fps).

new QoS metric as defined in Equation (6.2), completion ratio and QoP, achieved by different schedulers at certain frame rate. We mention that the negative QoS comes from the fact of task drop and deadline missing as well as their associated penalties. It is possible to give a more accurate modified measure of QoS in this case to keep QoS positive. For example, in the fast-forward mode, the task drop penalty should be much less, as is the soft deadline missing penalty, and more weight should be assigned for each completion as not all the frames are expected to be decoded in such mode. From these figures we can see that almost all the schedulers achieve similar performance for completion ratio, however, they behave very differently under the new QoS metric and QoP. The conclusion is that it is crucial to finish important tasks as many as possible, not the raw counter of task completions. It is mentioned that although LETF algorithm is 1/2 -competitive in the completion ratio on our monotonic-absolute-deadline task system [11], LETF*, which is better than LETF in QoP, achieves very bad user perceived QoP because, in general, the execution time of B frame is shorter than that of I or P frame, therefore, it will prefer to select B frame that actually is the least important frame.

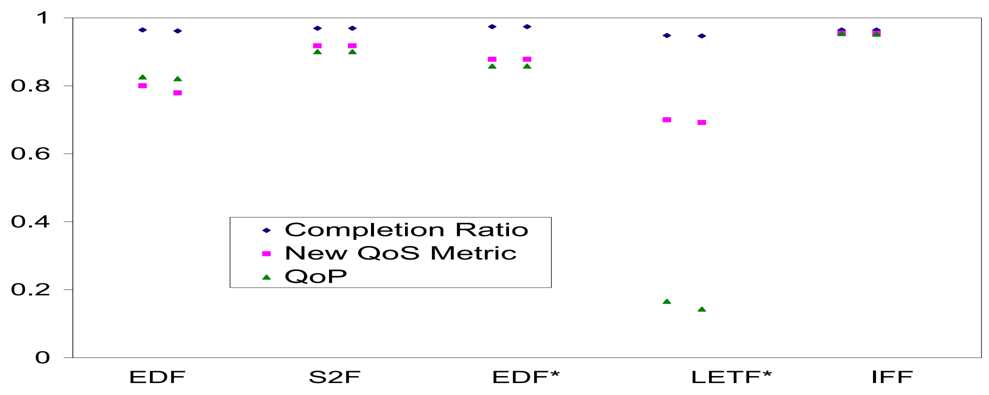


Figure 6.4: Comparison of different online scheduling policies on movie "Goldfinger" in the frame rate of 30fps in the case of, from left to right, non-preemptive and preemptive.

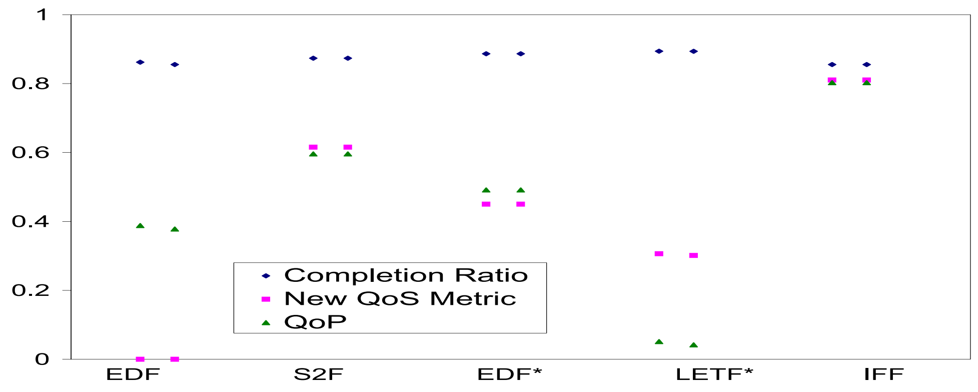


Figure 6.5: Comparison of different online scheduling policies on movie "Wizard of OZ" in the frame rate of 30fps in the case of, from left to right, non-preemptive and preemptive.

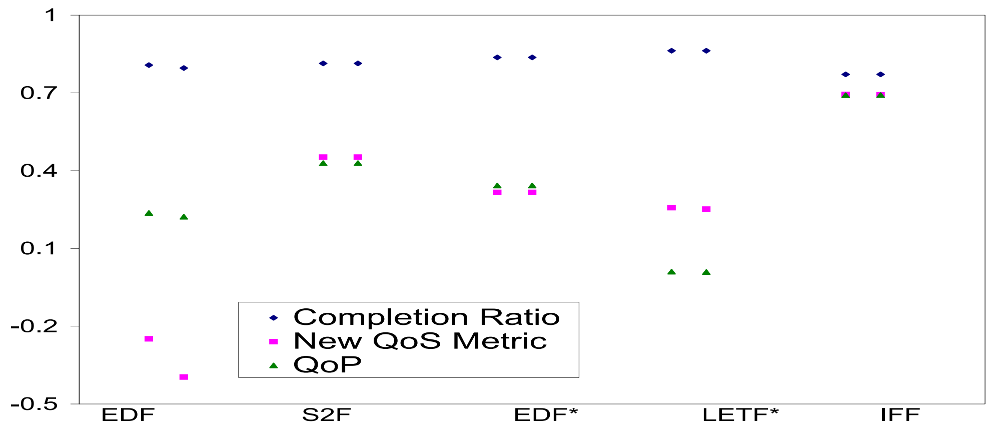


Figure 6.6: Comparison of different online scheduling policies on movie ”Silence of Lambs” in the frame rate of 45fps in the case of, from left to right, non-preemptive and preemptive.

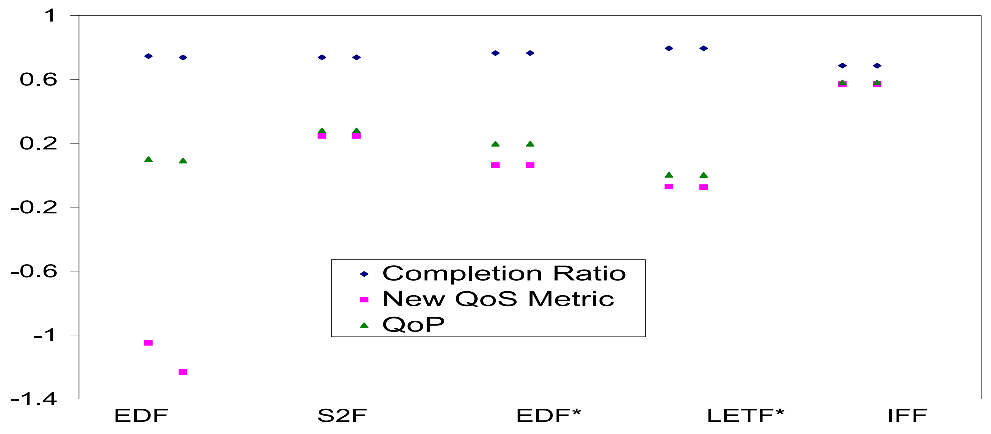


Figure 6.7: Comparison different online scheduling policies on movie ”Star Wars” in the frame rate of 45fps in the case of, from left to right, non-preemptive and preemptive.

6.5 Conclusions

With the increasing popularity of real-time multimedia and wireless communication applications, quality of service (QoS) attracts a lot of attention. In this chapter, we present a new metric on how to measure the QoS provided by an embedded system for real-time applications with mixed firm and soft deadlines. It captures the mixed firm and soft deadline nature of such applications and models data dependency as well. We show that the new defined QoS metric can reflect user perceived quality of presentation (QoP) much better than the completion ratio. We then find that the most commonly used online scheduling policies do not achieve good performance for such firm/soft real time applications. Based on the proposed quantitative QoS, we develop a set of online scheduling algorithms to maximize it. Simulations on popular MPEG movies show that most of them achieve much better QoS and QoP for users with about the same run time complexity and without extra hardware.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This dissertation has mainly focused on the soft real-time embedded system design. For the soft real-time embedded system, occasional deadline misses can be tolerated. When the highest achievable performance is higher than the user required performance, we can transfer this performance gap into energy saving. We have presented probabilistic design methodology and a set of energy reduction techniques by employing dynamic voltage scaling (DVS). We have also formulated the voltage set-up problem and proposed the practical solutions to this problem in order to find the best way to use DVS. Finally we have proposed our new QoS metric and a set of low overhead on-line schedulers in order to enhance both QoS and QoP, particularly for overloaded systems.

7.1.1 Probabilistic Design Methodology

As traditional design methodology only uses the worst case execution time (WCET) of each task in order to avoid any deadline missing, it often leads to over-designed systems, especially for the soft real-time embedded systems. We have proposed the novel concept of probabilistic design for soft real-time systems and a methodology to quickly explore such design space at an early design stage. The probabilistic design takes advantage of soft real-time embedded system's features, such as application's moderately high performance requirements, uncertainties in execution time (while the execution time distribution may be obtained by profiling or simulating on the target hardware) and tolerance to reasonable execution failures, to relax the rigid hardware requirements for software implementation and eventually minimize the system resources while still meeting the user required performance statistically. Our main contributions in this part are the probabilistic timing performance estimation method and the offline/on-line resource management approaches. We use energy as one example of resource and develop a set of energy reduction techniques by employing dynamic voltage scaling for both single and multiple processor systems. Our techniques exploit the slacks arising not only from the cases when the utilization of the processors based on the WCET is less than 1 or the run-time execution time deviates from WCET, but also from the intentional task (iteration) dropping. Simulation results show that our proposed techniques can significantly reduce the system energy consumption while statistically meeting the performance (e.g. completion ratio) constraint.

7.1.2 Voltage Set-up Problem

Dynamic voltage scaling (DVS) has been widely accepted as one of the most energy efficient techniques. Although ideal DVS system that supports continuous voltage changes gives more energy saving, the multiple-voltage system that supports discrete voltage changes is more practical and has been predicted as the future low power system [2]. We have formulated the voltage set-up problem that questions both how many levels of voltage and what value of each voltage level should be implemented on the multiple-voltage DVS system in order to minimize the energy consumption. Furthermore we have presented the practical solutions to this problem. Specifically, we derive analytical solutions for the dual-voltage system and efficient numerical methods for the general case. The experimental results validate our proposed approaches and suggest that multiple-voltage DVS systems, when the voltages are set up properly, can be very close to DVS technique's full potential in energy saving.

We also conduct a case study to address the voltage scheduling and set-up problem for the dual-voltage soft real-time systems that serve multiple streams with (m_i, k_i)-firm guarantee in order to minimize the energy consumption. First we propose an on-line greedy scheduler that we have proved is the most energy efficient deterministic on-line scheduler. Based on this scheduler we present a novel energy calculation method and a numerical approach to linearly search for the best voltage set-up. Simulation results have validated our approaches.

7.1.3 A New QoS Metric

Traditional completion ratio as a quality of service (QoS) metric does not measure quality of presentation (QoP) properly and cannot test different on-line schedulers as well. We have proposed a new quantitative QoS metric based on task completion ratio while differentiating firm and soft deadlines and taking task dependency into consideration. Using the decoding of MPEG movies as an example, we have shown that the proposed QoS metric is much better than the completion ratio in measuring the QoP of the movies. Based on this new QoS metric, we presented a set of new on-line algorithms that outperform popular scheduling algorithms such as earliest deadline first (EDF) and least execution time first (LETF) and enhance QoP significantly, particular for the overloaded systems. All the proposed on-line algorithms have low computation overhead and can be easily integrated into real-time operating systems (RTOS) to improve embedded system's performance and/or to save system resources.

7.2 Future Work

There are several possible directions to extend the work reported in this dissertation.

Although we have proposed the probabilistic design methodology, until now we only conduct simulation to show that it can reduce the system resources such as energy consumption while delivering the user required quality of service statistically. The future work includes applying the probabilistic design method to build prototype soft real-time embedded systems (e.g., multimedia systems) on FPGA devices, measuring the overall energy consumption, and evaluating the systems performance at the user level. It will also need to be extended to distributed embedded system by considering communication bandwidth and latency. The results will be compared with the systems developed by the traditional design methodology based on worst case execution time of each task.

As the feature sizes of the silicon VLSI shrink below 100nm, leakage (static) power is emerging as a significant contributor to power consumption in CMOS circuits [62]. The energy reduction techniques we have proposed in this dissertation only focus on reducing dynamic power. These techniques must be reinvestigated in future in order to reduce both dynamic and leakage power and conduct power-aware computing.

The energy reduction techniques for multiple processor systems that we have developed so far have the assumption that the task assignment and ordering are given a priori. However, different task assignments and orderings do affect the system energy consumption. If we simply apply the state-of-the-art task assignment and ordering algorithm, which goal is to minimize the completion time of the task graph, and our energy reduction techniques as two separated phases to the application task graph, we will not achieve the energy minimization. In future, we plan to create the framework that integrates the task assignment, ordering and voltage scheduling in order to save further energy. The framework will be applied not only for the homogeneous distributed real-time embedded systems but also for the heterogeneous systems.

We have conducted a case study for the design of energy-efficient dual-voltage soft real-time system with (m,k)-firm deadline guarantee. As most systems may have more than two voltages, how to determine the voltages for this kind of systems in order to minimize the energy consumption while still meeting (m,k)-firm deadline guarantee is the problem we need to solve. We also hope to create the framework to integrate the voltage set-up with task assignment ,ordering and voltage scheduling and apply it to real-life applications.

In this dissertation we have proposed a new QoS metric that differentiates firm and soft deadlines and captures the task dependency as well. More experiments on real multimedia systems need to be conducted in order to show that the new QoS metric is much better than the traditional completion ratio in terms of the QoP measurement. In future, we also plan to establish a solid performance modeling framework for integrated specification of throughput, delay, jitter and loss in embedded multimedia applications. This model is anticipated to reflect the QoP much better than the one we have proposed. Based on the new model, we need to develop new real-time scheduler to maximize th QoP. The new offline/on-line algorithms for better system resource management with probabilistic performance guarantees need also to be developed. All of above work will finally be conducted on the prototype multimedia systems.

BIBLIOGRAPHY

[1] EDA roadmap taskforce report–design of microprocessors. *Silicon Integration Initiative Inc. and Electronic Design Automation Industry Council*, Mar. 1999.

[2] International technology roadmap for semiconductors. *http://public.itrs.net*, 2001.

[3] Network monitoring using cisco service assurance agent. *http://www.cisco.com/univercd/cc/td/doc/product/software/ios121/121cgcr/fun_c/fcprt3/fcd301d.htm*, August 2001.

[4] Inc. Advanced Micro Devices. AMD Athlon 4 processors, data sheet reference no.24319. 2001.

[5] J. Altmann and P. Varaiya. INDEX project: User support for buying QoS with regard to user's preferences. *International Workshop on Quality of Service*, pages 101–104, 1998.

[6] N. Audsley, A. Burns, M. Richardson, and A. Wellings. Hard real-time scheduling: The deadline-monotonic approach. *Proceedings IEEE workshop on Real-Time Operating Systems and Software*, pages 133–137, 1991.

[7] C. Aurrecoechea, A. T. Campbell, and L. Hauw. A survey of QoS architectures. *Multimedia Systems*, 6(3):138–151, May 1998.

[8] N. K. Bambha and S. S. Bhattacharyya. A joint power/performance optimization technique for multiprocessor systems using a period graph construct. *Proceedings of the International Symposium on System Synthesis*, pages 91–97, 2000.

[9] N. K. Bambha, S. S. Bhattacharyya, J. Teich, and E. Zitzler. Hybrid search strategies for dynamic voltage scaling in embedded multiprocessors. *Proceedings of the International Workshop on Hardware/Software Co-Design*, pages 243–248, April 2001.

[10] N. K. Bambha, V. Kianzad, M. Khandelia, and S. S. Bhattacharyya. Intermediate representation for design automation of multiprocessor DSP systems. *Journal of Design Automation for Embedded Systems*, 7(4):307–323, Nov. 2002.

[11] S. K. Baruah, J. Haritsa, and N. Sharma. On-line scheduling to maximize task completions. *IEEE Real-Time Systems Symposium*, pages 228–236, Dec, 1994.

[12] A. C. Bavier, A. B. Montz, and L. L. Peterson. Predicting MPEG execution times. *ACM SIGMETRICS*, pages 131–140, June 1998.

[13] G. Bernat and A. Burns. Combining (n,m)-hard deadlines and dual priority scheduling. *Proceedings of Real-Time Systems Symposium*, pages 46–57, Dec. 1997.

[14] D. P. Bertsekas. Nonlinear programming. *Athena Scientific*, 1999.

[15] J. Bolot and A. Vega-Garcia. Control mechanisms for packet audio in the internet. *Proceedings of IEEE Infocom*, pages 232–239, March 1996.

[16] L. O. Burchard and P. Altenbernd. Estimating decoding times of MPEG-2 video streams. *Proceedings of the International Conference on Image Processing*, pages 560–563, Sep. 2000.

[17] T. D. Burd and R. W. Brodersen. Design issues for dynamic voltage scaling. *International Symposium on Low Power Electronics and Design*, pages 9–14, July 2000.

[18] T. D. Burd, T. Pering, A. Stratakos, and R. W. Brodersen. A dynamic voltage scaled microprocessor system. *IEEE Journal of Solid-State Circuits*, 35(11):1571–1580, Nov. 2000.

[19] G. Buttazzo, M. Spuri, and F. Sensini. Value vs. deadline scheduling in overload conditions. *IEEE Real-Time Systems Symposium*, pages 90–99, Dec. 1995.

[20] J. P. Calvez. Embedded real-time systems. *John Wiley & Sons*, 1993.

[21] A. Chandrakasan, V. Gutnik, and T. Xanthopoulos. Data driven signal processing: an approach for energy efficient computing. *International Symposium on Low Power Electronics and Design*, pages 347–352, 1996.

[22] A. P. Chandrakasan and J. Goodman. An energy-efficient reconfigurable public-key cryptography processor. *IEEE Journal of Solid-State Circuits*, pages 1808–1820, Nov. 2001.

[23] A. P. Chandrakasan, S. Sheng, and R. W. Broderson. Low-power CMOS digital design. *IEEE Journal of Solid-State Circuits*, 27(4):473–484, April 1992.

[24] J.-M. Chang and M. Pedram. Energy minimization using multiple supply voltages. *International Symposium on Low Power Electronics and Design*, pages 157–162, 1996.

[25] Karam S. Chatha and Ranga Vemuri. Performance evaluation tool for rapid prototyping of hardware-software codesigns. *9th International Workshop on Rapid System Prototyping*, pages 218–224, June 1998.

[26] C. Chen and M. Sarrafzadeh. Provably good algorithm for low power consumption with dual supply voltages. *IEEE/ACM International Conference on Computer Aided Design*, pages 76–79, 1999.

[27] C. Chen and M. Sarrafzadeh. Power reduction by simultaneous voltage scaling and gate sizing. *Design Automation Conference*, pages 333–338, 2000.

[28] R. L. Cruz. Quality of service guarantees in virtual circuit switched networks. *IEEE Journal on Selected Areas in Communications*, 3(6):1048–1056, August 1995.

[29] S. Dhar and D. Maksimovic. Low-power digital filtering using multiple voltage distribution and adaptive voltage scaling. *International Symposium on Low Power Electronics and Design*, pages 207–209, 2000.

[30] R. P. Dick, D. L. Rhodes, and W. Wolf. TGFF: Task graphs for free. *Proc. Int. Workshop Hardware/Software Codesign*, pages 97–101, Mar. 1998.

[31] A. Dudani, F. Mueller, and Y. Zhu. Energy-conserving feedback edf scheduling for embedded systems with real-time constraints. *ACM SIGPLAN Joint Conference LCTES'02 and SCOPES'02*, pages 213–222, June 2002.

[32] H. J. Eikerling, W. Hardt, J. Gerlach, and W. Rosenstiel. A methodology for rapid analysis and optimization of embedded systems. *International IEEE Symposium and Workshop on ECBS*, pages 252–259, March 1996.

[33] Rolf Ernst. Codesign of embedded systems: Status and trends. *IEEE Design & Test of Computers*, 15(2):45–54, 1998.

[34] A. Fiat and G. L. Woedinger (eds). On-line algorithms: the state of the art. *Springer, Berlin, Germany*, 1998.

[35] Marc Fleischmann. LongRun power management - dynamic power management for Crusoe processors. *Whitepaper, Transmeta Corp.*, 2001.

[36] M. R. Garey and D. S. Johnson. Computer and intractability: A guide to the theory of np-completeness. *W.H. Freeman and Company, New York, NY*, 1979.

[37] P. P. Gelsinger. Microprocessors for the new millennium: Challenges, opportunities , and new frontiers. *International Solid-State Circuits Conference*, pages 22–25, Feb. 2001.

[38] I. Goldberger and S. Kasapi. Current challenges in traditional design verification and its application in flip-chip devices. *International Electronics Manufacturing Technology Symposium*, pages 207–210, July 2003.

[39] K. Govil, E. Chan, and H. Wasserman. Comparing algorithms for dynamic speed-setting of a low-power CPU. *Proc. ACM International Conference on Mobile Computing and Networking*, pages 13–25, Nov. 1995.

[40] A. Grbic, S. Brown, and S. et al. Caranci. Design and implementation of the NUMAchine multiprocessor. *35th ACM/IEEE Design Automation Conference*, pages 65–69, June 1998.

[41] F. Gruian. Hard real-time scheduling for low-energy using stochastic data and DVS processors. *International Symposium on Low Power Electronics and Design*, pages 46–51, 2001.

[42] F. Gruian and K. Kuchcinski. LEneS: Task scheduling for low-energy systems using variable supply voltage processors. *Proc. of Asia and South Pacific Design Automation Conference*, pages 449–455, 2001.

[43] M. Hamdaoui and P. Ramanathan. A dynamic priority assignment technique for streams with (m,k)-firm deadlines. *IEEE Trans. on Computers*, 44(12):1443–1451, Dec. 1995.

[44] J. Henkel and R. Ernst. High-level estimation techniques for usage in hardware/software co-design. *Aisa and South Pacific Automation Conference*, pages 353–360, February 1998.

[45] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M. B. Srivastava. Power minimization of variable voltage core-based systems. *35th ACM/IEEE Design Automation Conference*, pages 176–181, 1998.

[46] I. Hong, M. Potkonjak, and M. B. Srivastava. On-line scheduling of hard real-time tasks on variable voltage processor. *IEEE/ACM International Conference on Computer Aided Design*, pages 653–656, 1998.

[47] I. Hong, G. Qu, M. Potkonjak, and M. B. Srivastava. Synthesis techniques for low-power hard real-time systems on variable voltage processors. *Proceedings of Real-Time Systems Symposium*, pages 178–187, 1998.

[48] X. Hu, T. Zhou, and E. H.-M. Sha. Estimating probabilistic timing performance for real-time embedded systems. *IEEE Trans. on VLSI systems*, 9(6):833–844, December 2001.

[49] S. Hua and G. Qu. Approaching the maximum energy saving on embedded systems with multiple voltages. *IEEE/ACM International Conference on Computer Aided Design*, pages 26–29, November 2003.

[50] S. Hua and G. Qu. A new QoS metric for hard/soft real-time applications. *International Conference on Information Technology: Coding and Computing*, pages 347–351, April 2003.

[51] S. Hua and G. Qu. QoP-driven scheduling for MPEG video decoding. *IEEE Transactions on Consumer Electronics*, 49(4):1341–1347, November 2003.

[52] S. Hua, G. Qu, and S. S. Bhattacharyya. Energy-efficient multi-processor implementation of embedded software. *3rd ACM International Conference on Embedded Software*, pages 257–273, October 2003.

[53] S. Hua, G. Qu, and S. S. Bhattacharyya. Energy reduction techniques for multimedia applications with tolerance to deadline misses. *40th ACM/IEEE Design Automation Conference*, pages 131–136, June 2003.

[54] S. Hua, G. Qu, and S. S. Bhattacharyya. Exploring the probabilistic design space of multimedia systems. *14th IEEE International Workshop on Rapid System Prototyping*, pages 233–240, June 2003.

[55] Intel. The Intel Xscale microarchitecture. *Technical Summary*, 2000.

[56] T. Ishihara and H. Yasuura. Voltage scheduling problem for dynamically variable voltage processors. *International Symposium on Low Power Electronics and Design*, pages 197–202, 1998.

[57] R. S. Janka and L. M. Wills. A novel codesign methodology for real-time embedded COTS multiprocessor-based signal processing systems. *Proceedings of the International Workshop on Hardware/Software Co-Design*, pages 157–161, 2000.

[58] M. C. Johnson and K. Roy. Datapath scheduling with multiple supply voltages and level converters. *ACM Trans. on Design Automation of Electronic Systems*, 2(3):227–248, 1997.

[59] A. Kalavade and P. Moghe. A tool for performance estimation of networked embedded end-systems. *35th ACM/IEEE Design Automation Conference*, pages 257–262, 1998.

[60] M. J. Karam and F. A. Tobagi. Analysis of the delay and jitter of voice traffic over the internet. *Proceedings of IEEE Infocom*, pages 824–833, April 2001.

[61] I. Karkowski and H. Corporaal. Design space exploration algorithm for heterogeneous multi-processor embedded system design. *35th ACM/IEEE Design Automation Conference*, pages 82–87, June 1998.

[62] N. Kim, T. Austin, D. Blaauw, and T. Mudge et al. Leakage current: Moore's law meets static power. *IEEE Computer Special Issue on Power- and Temperature- Aware Computing*, pages 68–75, Dec. 2003.

[63] K. T. Kornegay, G. Qu, and M. Potkonjak. Quality of service and system design. *IEEE Computer Society Annual Workshop on VLSI, Theme: System Level Design*, pages 112–117, 1999.

[64] C. M. Krishna and Y. Lee. Voltage-clock-scaling adaptive scheduling techniques for low power in hard real-time systems. *6th IEEE Real Time Technology and Applications Symposium*, pages 156–165, 2000.

[65] V. Krishna, N. Ranganathan, and N. Vijaykrishnan. Energy efficient datapath synthesis using dynamic frequency clocking and multiple voltages. *Proc. of the 12th International Conference on VLSI Design*, pages 440–445, 1999.

[66] M. Krunz and S. K. Tripathi. On the characterization of VBR MPEG streams. *ACM SIGMETRICS*, pages 192–202, 1997.

[67] P. Kumar and M. Srivastava. Predictive strategies for low-power RTOS scheduling. *IEEE International Conference on Computer Design*, pages 343–348, 2000.

[68] T. F. Lawrence. The quality of service model and high assurance. *Proceedings of High-Assurance Engineering Workshop*, pages 38–39, 1997.

[69] C. Lee, J. Lehoczky, R. Rajkumar, and D. Siewiorek. On quality of service optimization with discrete QoS options. *IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 276–286, June 1999.

[70] S. Lee and T. Sakurai. Run-time voltage hopping for low-power real-time systems. *37th Design Automation Conference*, pages 806–809, 2000.

[71] S. Lim and Y. et al. Bae. An accurate worst case timing analysis for RISC processors. *IEEE Transactions on Software Engineering*, 21(7):593–604, July 1995.

[72] Y. R. Lin, C. T. Hwang, and A. C. H. Wu. Scheduling techniques for variable voltage low power designs. *ACM Trans. on Design Automation of Electronic Systems*, 2(2):81–97, 1997.

[73] C. L. Liu and J. W. Layland. Scheduling algorithm for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61, Jan. 1973.

[74] J. Liu, P. H. Chou, N. Bagherzedeh, and F. Kurdahi. Power-aware scheduling under timing constraints for mission-critical embedded systems. *Proc. Design Automation Conference*, pages 840–845, June 2001.

[75] J. W. S. Liu. Real-time systems. *Prentice Hall*, 2000.

[76] J. W. S. Liu, W. K. Shih, K. J. Lin, R. Bettati, and J. Y. Chung. Imprecise computations. *Proc. IEEE*, 82(1):83–94, Jan. 1994.

[77] J. R. Lorch and A. J. Smith. Improving dynamic voltage scaling algorithms with PACE. *Proceedings of the ACM SIGMETRICS*, pages 50–61, June 2001.

[78] C. Lu, J. A. Stankovic, T. F. Abdelzaher, G. Tao, S. H. Son, and M. Marley. Performance specifications and metrics for adaptive real-time systems. *IEEE Real-Time Systems Symposium*, pages 13–23, Dec. 2000.

[79] J. Luo and N. K. Jha. Power-conscious joint scheduling of periodic task graphs and aperiodic tasks in distributed real-time embedded systems. *IEEE/ACM International Conference on Computer Aided Design*, pages 357–364, 2000.

[80] J. Luo and N. K. Jha. Battery-aware static scheduling for distributed real-time embedded systems. *Design Automation Conference*, pages 444–449, 2001.

[81] J. Luo and N. K. Jha. Static and dynamic variable voltage scheduling algorithms for real-time heterogeneous distributed embedded systems. *Proc. of Asia and South Pacific Design Automation Conference*, pages 719–726, Jan. 2002.

[82] J. Luo and N. K. Jha. Power-proflie driven variable voltage scaling for heterogeneous distributed real-time embedded systems. *Proc. of the 16th International Conference on VLSI Design*, pages 369–375, 2003.

[83] J. Madsen, J. Grode, P. V. Knudsen, M. E. Petersen, and A. E. Haxthausen. LYCOS: the lyngby co-synthesis system. *Journal for Design Automation of Embedded Systems*, 2(2):195–235, March 1997.

[84] S. Malik, M. Martonosi, and Y. S. Li. Static timing analysis of embedded software. *Design Automation Conference*, pages 147–152, June 1997.

[85] A. Manzak and C. Chankrabarti. Variable voltage task scheduling algorithm for minimization energy. *International Symposium on Low Power Electronics and Design*, pages 279–282, 2001.

[86] R. Marculescu, A. Nandi, L. Lavagno, and A. Sangiovanni-Vincentelli. System-level power/performance analysis of portable multmedia systems communicating over wireless channels. *International Conference on Computer Aided Design*, pages 207–214, Nov. 2001.

[87] C. L. McCreary, A. A. Khan, J. J. Thompson, and M. E. McArdle. A comparison of heuristics for scheduling DAGs on multiprocessors. *Proceedings of the International Parallel Processing Symposium*, pages 446–451, April 1994.

[88] R. Mishra, N. Rastogi, D. Zhu, D. Mosse, and R. Melhem. Energy aware scheduling for distributed real-time systems. *International Parallel and Distributed Processing Symposium*, April 2003.

[89] A. Mittal, G. Manimaran, and C. Siva Ram Murthy. Integrated dynamic scheduling of hard and QoS degradable real-time tasks in multiprocessor systems. *Journal of Systems Architecture*, 46(9):793–807, July 2000.

[90] D. Mosse, H. Aydin, B. Childers, and R. Melhem. Compiler-assisted dynamic power-aware scheduling for real-time applications. *Workshop on Compiler and OS for Low Power*, Oct. 2000.

[91] N. Namgoong, M. Yu, and T. Meng. A high efficiency variable-voltage CMOS dynamic DC-DC switching regulator. *Proc. IEEE International Solid-State Circuits Conference*, pages 380–381, 1997.

[92] J. K. Ng, K. R. Leung, W. Wong, V. C. Lee, and C. K. Hui. Quality of service for MPEG video in human perspective. *International Conference on Real-Time and Embedded Computing Systems and Applications*, pages 233–241, March 2002.

[93] T. Pering, T. Burd, and R. W. Brodersen. The simulation and evaluation of dynamic voltage scaling algorithms. *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 76–81, August 1998.

[94] T. Pering, T. Burd, and R. W. Brodersen. Voltage scheduling in the lparm microprocessor system. *International Symposium on Low Power Electronics and Design*, pages 96–101, 2000.

[95] P. Pillai and G. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. *Proc. of the 18th ACM Symposium on Operating Systems Principles*, pages 89–102, 2001.

[96] J. Pouwelse, K. Langendoen, and H. Sips. Dynamic voltage scaling on a low-power microprocessor. *Proceedings of the 7th Conference on Mobile Computing and Networking*, pages 251–259, 2001.

[97] J. Pouwelse, K. Langendoen, and H. Sips. Energy priority scheduling for variable voltage processors. *International Symposium on Low Power Electronics and Design*, pages 28–33, 2001.

[98] Q. Qiu, Q. Wu, and M. Pedram. Dynamic power management in a mobile multimedia system with guaranteed quality-of-service. *ACM/IEEE Design Automation Conference*, pages 834–839, 2001.

[99] G. Qu, , M. Mesarina, and M. Potkonjak. System synthesis of synchronous multimedia applications. *International Sysmposium and System Synthesis*, pages 128–133, Nov. 1999.

[100] G. Qu. What is the limit of energy saving by dynamic voltage scaling? *IEEE/ACM Interantional Conference on Computer-Aided Design*, pages 560–563, Nov. 2001.

[101] G. Qu and M. Potkonjak. Power minimization using system-level partitioning of applications with quality of service requirements. *IEEE/ACM Intl. Conference on Computer-Aided Design*, pages 343–346, 1999.

[102] G. Qu and M. Potkonjak. Energy minimization with guaranteed quality of services. *ACM/IEEE Intl. Symposium on Low Power Electronics and Design*, pages 43–48, 2000.

[103] G. Qu and M. Potkonjak. Techniques for energy-efficient communication pipeline design. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 10(5):542–549, October 2002.

[104] G. Quan and X. Hu. Enhanced fixed-priority scheduling with (m,k)-firm guarantee. *Proceedings of Real-Time Systems Symposium*, pages 79–88, 2000.

[105] G. Quan and X. Hu. Energy efficient fixed-priority scheduling for real-time systems on variable voltage processors. *38th IEEE/ACM Design Automation Conference*, pages 828–833, 2001.

[106] S. Raje and M. Sarrafzadeh. Variable voltage scheduling. *International Symposium on Low Power Electronics and Design*, pages 9–14, 1995.

[107] R. Rajkumar, Lee C., J. Lehoczky, and D. Siewiorek. A resource allocation model for QoS management. *Proceedings of Real-Time Systems Symposium*, pages 298–307, 1997.

[108] R. Rajkumar, Lee C., J. Lehoczky, and D. Siewiorek. Practical solutions for QoS-based resource allocation problems. *Proceedings of Real-Time Systems Symposium*, pages 296–306, 1998.

[109] P. Ramanathan. Graceful degradation in real-time control applications using (m,k)-firm guarantee. *Proc. IEEE Fault-Tolerant Computing Symposium*, pages 132–141, 1997.

[110] P. Ramanathan. Overload management in real-time control applications using (m,k)-firm guarantee. *IEEE Trans. on Parallel and Distributed Systems*, 10(6):549–559, 1999.

[111] H. Sariowan, R. L. Cruz, and G. C. Polyzos. Scheduling for quality of service guarantees via service curves. *Proc. Fourth International Conference on Computer Communications and Networks*, pages 512–520, 1995.

[112] D. Scherrer and H. Eberle. A scalable real-time signal processor for object-oriented data flow applications. *Proceedings of the International Conference on Parallel and Distributed Computing Systems*, pages 183–189, September 1998.

[113] M. T. Schmitz and B. M. Al-Hashimi. Low power process assignment for distributed embedded systems using dynamic voltage scaling. *Proceedings IEE Hardware-Software Co-Design*, pages 7/1–7/4, 2000.

[114] M. T. Schmitz and B. M. Al-Hashimi. Considering power variations of DVS processing elements for energy minimisation in distributed systems. *Proceedings of 14th International Symposium on System Synthesis*, pages 250–255, 2001.

[115] M. T. Schmitz, B. M. Al-Hashimi, and P. Eles. Energy-efficient mapping and scheduling for DVS enabled distributed embedded systems. *Design, Automation and Test in Europe Conference*, pages 514–521, March 2002.

[116] D. Shin, J. Kim, and S. Lee. Intra-task voltage scheduling for low-energy hard real-time applications. *IEEE Design and Test of Computers*, 18(2):20–30, 2001.

[117] Y. Shin and K. Choi. Power conscious fixed priority scheduling for hard real-time systems. *36th ACM/IEEE Design Automation Conference*, pages 134–139, 1999.

[118] Y. Shin, K. Choi, and T. Sakurai. Power optimization of real-time embedded systems on variable speed processors. *International Conference on Computer-Aided Design*, pages 365–368, 2000.

[119] G. C. Sih and E. A. Lee. A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures. *IEEE Tran. on Parallel and Distributed Systems*, 4(2), February 1993.

[120] R. Steinmetz. Analyzing the multimedia operating system. *IEEE Multimedia*, 2(1):68–84, Spring 1995.

[121] R. A. Sutton, V. P. Srini, and J. M. Rabey. A multiprocessor DSP system using PADDI-2. *35th ACM/IEEE Design Automation Conference*, pages 62–65, June 1998.

[122] T. S. Tia, Z. Deng, M. Shankar, M. Storch, J. Sun, L.-C. Wu, and J. W.-S. Liu. Probabilistic performance guarantee for real-time tasks with varying computation times. *Proc. Real-Time Technology and Applications Symposium*, pages 164–173, 1995.

[123] V. Tiwari, D. Singh, S. Rajgopal, G. Mehta, R. Patel, and F. Brez. Reduced power in high-performance microprocessors. *35th ACM/IEEE Design Automation Conference*, pages 732–737, June 1998.

[124] A. Vogel and B. et al. Kerherve. Distributed multimedia and QoS: a survey. *IEEE Multimedia Magaazine*, 2(2):10–19, Summer 1995.

[125] M. Weiser, B. Welch, A. Demers, and S. Shenker. Scheduling for reduced CPU energy. *USENIX Symposium on Operating Systems Design and Implementation*, pages 13–23, 1994.

[126] R. West, K. Schwan, and C. Poellabauer. Scalable scheduling support for loss and delay constrained media streams. *IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 24–33, June 1999.

[127] D. Wijesekera and J. Srivastava. Quality of service (QoS) metrics for continuous media. *Multimedia Tools and Applications*, 3:127–166, 1996.

[128] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced CPU energy. *36th Annual Symposium on Foundations of Computer Science*, pages 374–382, 1995.

[129] Y. Zhang, X. Hu, and D. Z. Chen. Task scheduling and voltage selection for energy minimization. *Design Automation Conference*, pages 183–188, 2002.

[130] D. Zhu, R. Melhem, and B. Childers. Scheduling with dynamic voltage/speed adjustment using slack reclamation in multi-processor real-time systems. *IEEE 22nd Real-Time Systems Symposium*, pages 84–94, 2001.

[131] E. Zitzler, J. Teich, and S. S. Bhattacharyya. Optimizing the efficiency of parameterized local search within global search: A preliminary study. *Proceedings of the Congress on Evolutionary Computation*, pages 365–372, July 2000.