國立東華大學資訊工程學系

碩士論文

藉由工作排程和電壓選擇技術來降低系統能量的消耗

Energy Optimization with Task scheduling and Voltage selection for Variable voltage Processors

研究生：李映辰　撰

指導教授：雍 忠　博士

中華民國九十五年七月

Energy Optimization with Task Scheduling and Voltage Selection for Variable Voltage Processors

by

Ying-Chen Lee

A thesis submitted in partial fulfillment

of the requirements for the degree of

Master of Science

Department of Computer Science and Information Engineering

National Dong Hwa University

July 2006

Approved: _____

Chung Yung

國立東華大學
學位論文授權書

※說明※
本授權書請撰寫並簽名後，裝訂於紙本論文書名頁之次頁。

本授權書所授權之論文為本人在國立東華大學 資 訊 工 程 系所＿＿＿＿組

九十四 學年度第 二 學期取得 碩 士學位之論文。

論文名稱：藉由工作排程和電壓選擇技術來降低系統能量的消耗
(Energy Optimization with Task scheduling and Voltage selection for Variable voltage Processors)

指導教授姓名：雍忠 教授

學生姓名：李映辰

學號：69321056

本人具有著作財產權之上列論文全文資料，基於資源共享理念、回饋社會與學術研究之目的，非專屬、無償授權國立東華大學及國家圖書館，得不限地域、時間與次數，以微縮、光碟或數位化等各種方式重製散布、發行或上載網路，提供讀者非營利性質之線上檢索、閱覽、下載或列印。

上述數位化公開方式限：

■校內、校外公開。

□校內公開，校外因

1.上列論文申請專利（案號： ），請於 3 年後公開。

2.上列論文內容隱私權之需要（請指導教授附函說明特殊原因），請於 3 年後公開。

□校內公開，校外因＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿，請於 1 年後公開。

授權內容均無須訂立讓與及授權契約書，授權之發行權為非專屬性發行權利。依本授權所為之收錄、重製、發行及學術研發利用均為無償。數位化公開方式若未勾選，本人同意視同授權校內、校外公開。

簽　　名 (親筆正楷)		日期	中華民國 九十五 年　　月　　日

0950215 表 LB702

Acknowledgements

First of all, I must offer my heartfelt thanks to my advisor, Professor Chung Yung. In the process of writing my thesis, Professor Yung supervises my progress, reads and revises my drafts carefully and gives me insightful remarks. Thank him to put him's heart and soul into help me to finish the thesis. Without him, it is impossible for me to finish the thesis. I thanks him so much, very very much. Ching-Ho Cheng also give assistance to me and encourage me to go forward. He is really a good person, I want to thank to him.

In addition, I would like to thank laboratory members, Sin-Jhe Yu, Bing-Liang Shih, Wen-Kai Tu, and Wei-Jia Shen. They give me the whole support to my thesis. We go through happiness and sorrow. I am so glad to be friend with them. Finally, I want to give this honor to my family. They are my mainstay.

Energy Optimization with Task Scheduling and Voltage Selection for Variable Voltage Processors

Ying-Chen Lee

Advisor: Chung Yung

Abstract

In this thesis, we propose a task scheduling and voltage allocation technique in dynamically variable voltage processors, the purpose of which is minimization of processor energy consumption. Task scheduling is a technique to minimize power consumption with task assignment and ordering. In the static voltage scheduling technique, the CPU time and supply voltage are assigned to tasks in order to minimize the total energy consumption. In this thesis, we formulate the problem as an integer linear programming problem with constraints on the dependency tasks. We extend the problem to include the case in which each task may have different switched capacitances and solve it optimally. We simulate the problem of industry processors with variable voltages, such as Transmeta Crusoe TM5800, Intel P2, Intel P3. By our task scheduling and voltage allocation technique, the experimental benchmark shows a 25.3% reduction of energy consumption over EDF scheduling and a 3.2% reduction over Ishihara and Yasuura's approach. Note that Ishihara and Yasuura's approach does not take the dependency of tasks into consideration.

藉由工作排程和電壓選擇技術來降低系統能量的消耗

李映辰

指導教授 : 雍 忠 教授

摘要

在本論文中，我們研究低功率最佳化的相關技術。在嵌入式系統中能量的消耗和電源電壓是成正比的，所以要有效降低能量的消耗最直接的方法就是降低電源電壓，但是改變電源電壓會造成工作執行的時間改變，可能會讓某些工作無法在期限內完成，我們想要找出一個方法能使得能量消耗是最小的又能讓所有的工作都在它的期限內完成，所以我們使用工作排程和電源電壓選擇的技術，來降低能量的消耗。工作排程是一種藉由工作指派和重新排列工作的順序來讓能量消耗最少的一種技術。我們把這個問題當成一個整數線性規劃的問題，多考慮了幾個因素。實驗數據顯示我們的整數線性規劃演算法能夠有效的減少能量的消耗達到 25%。

Contents

List of Figures

Chapter 1

Introduction

Energy is one of the important metrics for optimization in the design and operation of embedded systems. Task scheduling is an approach to solve this problem. Because technology is more and more developed, the CPU can switch over different voltages. Voltage selection is one of the approaches to reduce energy consumption. In recent years, integer linear programming(ILP) techniques have been proposed to solve the task scheduling problem. This thesis proposes an ILP formulation for the task scheduling and voltage selection problem as an integer IL with constraints on the dependency tasks and switched capacitances, and solves it optimally.

The progress of deep-submicron technologies has enabled all of a systems functionalities to be implemented on a single chip(i.e., system-on-chip (SOC) design)[IY98]. One of the most important SOC design considerations is the minimization of the energy consumption. Energy is a important issue in the design and operation of embedded systems such as mobile phone, PDAs and

wireless communication systems. There are two primary ways to reduce power consumption in embedded computing systems: processor shutdown and processor slowdown. Slowdown using frequency or voltage scaling is more effective in power consumption. Scaling the frequency and voltage of a processor leads to an increase in the execution time of a task. In real-time systems, we want to minimize energy while adhering to the deadlines of the tasks.

Since the energy consumption is determined by the equation:

$$E_i = R_i \cdot C_i \cdot V_i^2$$

where R_i is the total number of cycles required for the execution of task J_i, C_i is the average switched capacitance per clock cycle for the task, and V_i is the supply voltage used for the execution of the task. It clear that reduction of V_i is the most effective for energy reduction. But reducing supply voltage causes increase the execution time of tasks and the tasks will miss its deadlines. Energy and deadlines are often contradictory goals and we have to judiciously manage time and power to achieve minimizing energy.

The task scheduling problem is the problem of assigning the tasks in the system in a manner that will optimize the overall performance of the application, while assuring the correctness of the result. The focus of this thesis direct against the task scheduling and voltage selection problem and minimize energy consumption.

The power reduction techniques which make it possible to scale supply voltage(V_{DD}) of processor just to meet the performance of the application program have been proposed as follow.

Yao et al.[YDS95] proposed an optimal algorithm for finding a schedule of tasks and voltage allocation under the assumption that the number and range of supply voltages are infinitely large(i.e., continuously variable voltage), which is practically impossible.

Hong et al.[HI98] proposed a heuristic to schedule-mixed workloads of static and dynamic tasks, based on Yao et al.'s algorithm.

Ishihara and Yasuura[IY97] proposed an optimal voltage allocation technique using discontinuously variable voltage processor. However, the optimality of the technique is confined to a single task.

Shin and Choi.[SC99] proposed a fixed priority based on-line scheduling scheme for a continuously variable voltage processor. The technique is simple, but cannot fully exploit the fact that the arrival times and deadlines of most real-time tasks in embedded system applications are known in advance. In addition Shin et al.[SCS00] proposed an off-line fixed-priority based scheduling algorithm for a continuously variable voltage processor. One major limitation of the technique is that the arrival times of the tasks are assumed to be all identical.

Pering et al. presented an online scheduling algorithm for soft real-time systems.[PBB00] This algorithm releases the deadline constraints and allows application frames to complete after their deadlines. The scheduler can then absorb the effects of high frame-to-frame application variance, which might otherwise increase energy.

Burd et al. have demonstrated dynamic voltage scaling on a complete em-

bedded processor system.[BT00]. Their approach find optimal supply voltage which is fitting to the desired performance.

Quan and Hu.[QH01] proposed an off-line fixed priority based voltage scheduling algorithm which overcomes the limitations in [SC99] and [SCS00]. For a set of tasks with statically assigned priorities, it produces a voltage schedule which always results in energy consumption below the minimum constant voltage and which shuts down the system when it is idle. However, the technique is not optimal, even if its application is useful in periodic tasks where the execution priorities among tasks are given. In[QH02], Quan and Hu proposed two optimal algorithms to the same voltage scheduling problem as in[QH01] in a fix-priority real-time system.

There are a lot of factors need to be consider in task scheduling and voltage selection problem, such as dependency and capacitances. In [IY97, SC99, SCS00], they only consider energy optimization with the constraint that execution time of tasks can not exceed its time constraint with respect to switched capacitances of each task, In[QH01, QH02] only consider energy optimization with the dependency constraints and the time constraint. Hence, our objective is to formulate the problem as an integer linear programming problem with constraints on the dependency tasks, and we extend the problem to include the case in which each task may have different switched capacitances.

$Task$	S_i	DL_i	EC_i	SC_i
$Task_1$	0	10	2000×10^6	$0.4 \times 10^{-7} F$
$Task_2$	3	8	1200×10^6	$1.4 \times 10^{-7} F$
$Task_3$	5	10	1600×10^6	$1.0 \times 10^{-7} F$
$Task_4$	8	11	1600×10^6	$0.8 \times 10^{-7} F$

Figure 1.1: Specification of task 1-4.

1.1 Motivational examples

We illustrate our idea wit a simple example. Figure1.1 shows four tasks $task_1$, $task_2$, $task_3$, and $task_4$ with their timing constraints when running on a Intel P3 system[SAN]. The supply voltages of P3 is 1.15 V and 1.40 V, and the clock speed is 400MHz and 800MHz,respectively.

Figure1.2(A) shows an optimal voltage allocation with a feasible schedule for $task_1$, $task_2$, $task_3$, and $task_4$ computed by the application of Yao et al.'s algorithm[YDS95]. The highest voltage used is 1.316 V and the lowest voltage is 1.233 V. The total energy consumption E_{total}=849.487 J. Figure1.2(B) shows transformation of continuously variable voltage allocation into discontinuously variable voltage allocation[KK03]. There are two optional voltages 1.15 V and 1.40 V. The total energy consumption E_{total}= 834.74 J. Figure 1.2(C) shows the better solution for the task set which consider switched capacitances and the time constraint. The total energy consumption E_{total}= 808.74 J.

In Figure 1.2(C) shows another solution for this problem and it consume less energy than (A), and (B). Hence, We have an idea and we want to find out this approach and we formulate the problem as an integer linear programming problem and consider the case in which each task may have different switched

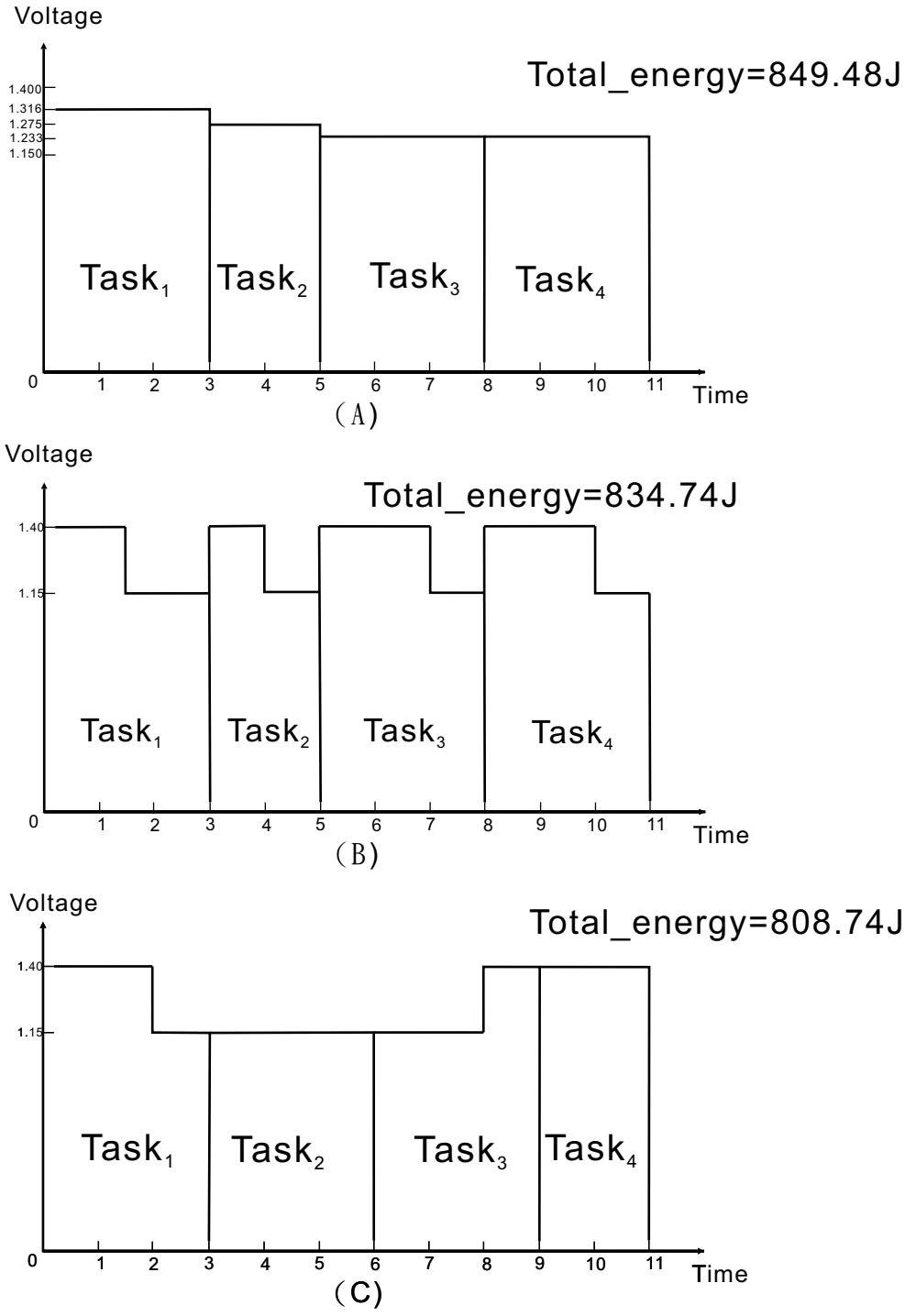

Figure 1.2: An example illustrating our ILP formulation (A)A continuously variable voltage allocation for tasks in Figure 1.1;(B) Transformation of continuously variable voltage allocation into discontinuously variable voltage allocation;(C)shows the better solution for this problem

capacitances and solve it optimally.

1.2 Thesis Organization

In the next Chapter, we introduce some background and an experiment tools such as DVS(what is DVS and why we use DVS to solve task scheduling problem), AMPL(A Mathematical Programming Language), CPLEX(the solver). In the Chapter 3 first we describes the target system assumptions next give the notation and our ILP formulations. Finally we propose an example is given to demonstrate our algorithm. Chapter 4 includes our experiments result and experiments environment. Finally we describes some phenomenon which found in the experiment. Chapter 5 gives the conclusion.

Chapter 2

Background

2.1 Dynamic Voltage Scaling

2.1.1 What is DVS ?

DVS tries to address the tradeoff between performance and battery life by taking into account two important characteristics of most current computer systems: (1) the peak computing rate needed is much higher than the average throughput that must be sustained; and (2) the processors are based on CMOS logic. The first characteristic effectively means that high performance is needed only for a small fraction of the time, while for the rest of the time, a low-performance, low-power processor would suffice. We can achieve the low performance by simply lowering the operating frequency of the processor when the full speed is not needed. DVS goes beyond this and scales the operating voltage of the processor along with the frequency. This is possible because static CMOS logic, used in the vast majority of microprocessors today, has a

voltage-dependent maximum operating frequency, so when used at a reduced frequency, the processor can operate at a lower supply voltage. Since the energy dissipated per cycle with CMOS circuitry scales quadratically to the supply voltage P to be proportional to V_{dd}^2,[BB95] DVS can potentially provide a very large net energy savings through frequency and voltage scaling.

In time-constrained applications, often found in embedded systems like cellular phones and digital video cameras, DVS presents a serious problem. In these real-time embedded systems, one cannot directly apply most DVS algorithms known to date, since changing the operating frequency of the processor will affect the execution time of the tasks and may violate some of the timeliness guarantees.

2.1.2 Why DVS ?

Power requirements are one of the most critical constraints in mobile computing applications, limiting devices through restricted power dissipation, shortened battery life, or increased size and weight. The design of portable or mobile computing devices involves a tradeoff between these characteristics. For example, given a fixed size or weight for a hand held computation device/platform, one could design a system using a low-speed, low-power processor that provides long battery life, but poor performance, or a system with a (literally) more powerful processor that can handle all computational loads, but requires frequent battery recharging.

This simply reflects the cost of increasing performance – for a given technology, the faster the processor, the higher the energy costs (both overall and per unit of computation).

The discussion in this paper will generally focus on the energy consumption of the processor in a embedded systems for two main reasons. First, the practical size and weight of the device are generally fixed, so for a given battery technology, the available energy is also fixed. This means that only power consumption affects the battery life of the device. Secondly, we focus particularly on the processor because in most applications, the processor is the most energy-consuming component of the system. This is definitely true on small hand held devices like PDAs ,[ECS] which have very few components, but also on large laptop computers [LJR98] that have many components including large displays with backlighting.

Table 1 shows measured power consumption of a typical laptop computer. When it is idle, the display backlighting accounts for a large fraction of dissipated power, but at maximum computational load, the processor subsystem dominates, accounting for nearly 60% of the energy consumed. As a result, the design problem generally boils down to a tradeoff between the computational power of the processor and the system's battery life.

2.2 Related Tools

2.2.1 SimpleScalar

SimpleScalar was created by Todd Austin. Development began while he was a Ph.D. student at the University of Wisconsin in Madison. Early versions of the tool set included contributions by Doug Burger and Guri Sohi. Today, SimpleScalar is developed and supported by SimpleScalar LLC.

The SimpleScalar licensing model permits users to build onto the SimpleScalar foundation and make their enhancements available to licensed SimpleScalar users. Examples of projects that are extending the capabilities of the SimpleScalar tool set include the MASE and PowerAnalyzer projects at the University of Michigan, the BullsEye project at University of Texas at Austin, Wattch from Princeton, and SimplePower from Penn State.

The SimpleScalar tool set is a system software infrastructure used to build modeling applications for program performance analysis, detailed microarchitectural modelling, and hardware-software co-verification. Using the SimpleScalar tools, users can build modelling applications that simulate real programs running on a range of modern processors and systems. The tool set includes sample simulators ranging from a fast functional simulator to a detailed, dynamically scheduled processor model that supports non-blocking caches, speculative execution, and state-of-the-art branch prediction. The SimpleScalar tools are used widely for research and instruction, for example, in 2000 more than one third of all papers published in top computer architecture conferences used the SimpleScalar tools to evaluate their designs. In addition

to simulators, the SimpleScalar tool set includes performance visualization tools, statistical analysis resources, and debug and verification infrastructure.

SimpleScalar LLC is unique in the EDA community. We distribute our software under an open source model, trusting that our users will license the software that they use. The tool set is distributed with all source code, making it possible for users extend SimpleScalar, and to adapt existing models to their own ideas.

SimpleScalar simulators can emulate the Alpha, PISA, ARM, and x86 instruction sets. The tool set includes a machine definition infrastructure that permits most architectural details to be separated from simulator implementations. All of the simulators distributed with the current release of SimpleScalar can run programs from any of the above listed instruction sets. Complex instruction set emulation (e.g., x86) can be implemented with or without microcode, making the SimpleScalar tools particularly useful for modelling CISC instruction sets.

The PISA instruction set (a.k.a. the portable instruction set architecture) is a simple MIPS-like instruction set maintained primarily for instructional use. A GNU GCC-based cross-compiler and pre-built libraries are also available for this target. The PISA target is particularly useful for computer engineering instruction as the tools can be built on a wide range of host platforms, including Linux/x86, Win2000, SPARC Solaris, and others.

SimpleScalar builds on most 32-bit and 64-bit flavors of UNIX and Windows NT-based operating systems. The internal software architecture of the

tool set includes a host interface module, permitting fast porting to other host platforms. The host interface module permits cross-endian emulation, thus it is possible to use emulate a target on a host platform with a different endian, e.g., running Alpha ISA emulation on a SPARC Solaris host platform. Most SimpleScalar users and developers (including SimpleScalar LLC) use SimpleScalar on Linux/x86.

2.2.2 Wattch

Wattch is the foundations for the power-modeling infrastructure are parameterized power models of common structures present in modern superscalar microprocessors. The power consumption of the units modeled was considered to depend particularly on the internal capacitances for the circuits that make up the processor. The main modeled processor units fall into four categories: (1) Array structures, (2) fully associative content-addressable memories, (3) combinational logic and wires, and (4) clocking.

Wattch was found to be 1000x faster than existing layout-level power tools, and yet maintains accuracy within 10% of their estimates. Three possible usage flows of Wattch were presented in [**?**] making it as handy for architects as for compiler writers who might use Wattch to evaluate power consumption in their design process. The first usage scenario applies to cases where the user is interested in comparing several design configurations that are achievable simply by varying parameters for hardware structures already modeled. The second usage scenario is for software or compiler development, where a single hardware configuration is used and several programs are simulated and compared. The last scenario highlights Wattchs modularity. Additional hardware modules can be added to the simulator. When performing power studies, Wattch provides result for a variety of metrics like power, performance, energy, and energy-delay product.

2.2.3 AMPL

Introduction

Practical large-scale mathematical programming involves more than just the minimization or maximization of an objective function subject to constraint equations and inequalities. Before any optimizing algorithm can be applied, some effort must be expended to formulate the underlying model and to generate the requisite computational data structures.

If algorithms could deal with optimization problems as people do, then the formulation and generation phases of modeling might be relatively easy. In reality, however, there are many differences between the form in which human modelers understand a problem and the form in which algorithms solve it. Reliable *translation* from the "modeler's form" to the "algorithm's form" is often a considerable expense.

In the traditional approach to translation, the work is divided between human and computer. First, a person who understands the modeler's form writes a computer program whose output will represent the required data structures. Then a computer compiles and executes the program to create the algorithm's form. This arrangement is often costly and error-prone; most seriously, the program must be debugged by a human modeler even though its output-the algorithm's form-is not meant for people to read.

In the important special case of linear programming, the largest part of the algorithm's form is the representation of the constraint coefficient matrix. Typically this is a very sparse matrix whose rows and columns number in

the hundreds or thousands, and whose nonzero elements appear in intricate patterns. A computer program that produces a compact representation of the coecients is called a matrix generator. Several programming languages have been designed specifically for writing matrix generators (Haverly Systems 1977, Creegan 1985) and standard languages like Fortran are also often used (Beale 1970).

Many of the difficulties of translation from modeler's form to algorithm's form can be circumvented by the use of a computer modeling language for mathematical programming. A modeling language is designed to express the modeler's form in a way that can serve as direct input to a computer system. Then the translation to the algorithm's form can be performed entirely by computer, without the intermediate stage of programming. The advantages of modeling languages over matrix generators have been analyzed in detail by Fourer (1983). Implementations such as GAMS (Bisschop and Meeraus 1982; Brooke, Kendrick and Meeraus 1988) and MGG (Simons 1987) were under way in the 1970's, and the pace of development has increased in recent years.

AMPL (A Mathematical Programming Language)[FGK90] is a high-level language for describing mathematical programs. AMPL allows a mathematical programming model to be specified independently of the data used for a specific instance of the model. AMPL's language for describing mathematical programs closely follows that used by humans to describe mathematical programs to each other. For this reason, modelers may spend more time improving the model and less time on the tedious details of data manipulation

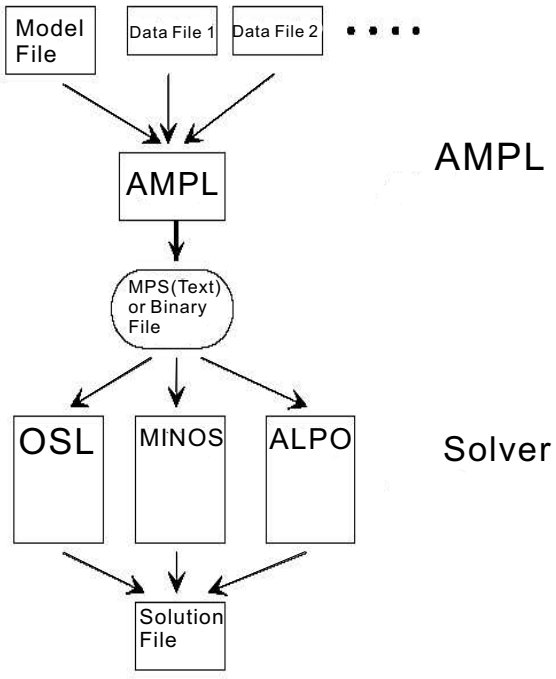

Figure 2.1: The Example of a Data Flow Graph

and problem solution.

A functional diagram of how AMPL is used is shown below. To start, AMPL needs a mathematical programming model, which describes variables, objectives and relationships without refering to specific data. AMPL also needs an instance of the data, or a particular data set. The model and one (or more) data files are fed into the AMPL program. AMPL works like a compiler: the model and input are put into an intermediate form which can be read by a solver. The solver actually finds an optimal solution to the problem by reading in the intermediate file produced by AMPL and applying an appropriate algorithm. The solver outputs the solution as a text file, which can be viewed directly and cross-referenced with the variables and constraints specified in the model file.

A commercial version of AMPL has been ported to most UNIX worksta-

tions at the University of Michi- gan's Computer Aided Engineering Network (CAEN). AMPL may be used with several different mathematical program solvers, including MINOS version 5.4, ALPO [VRJ], CPLEX citecplex) and OSL [IBMC90] version 2. AMPL may be used interactively or may be used in batch mode. Used interactively, a modeler may selectively modify both the model and its data, and see the changes in the optimal solution instantaneously. Used in batch mode, a modeler may obtain detailed and well-formatted solutions to previously defined mathematical programs. A shell script that may be used on any UNIX platform has been developed to simplify batch operation. Using this script, one can directly see the optimal solution to an AMPL problem without knowing UNIX or the details of the CAEN network.

Components

The five major parts of an algebraic model - sets, parameters, variables, objectives and constraints - are also the five kinds of components in an AMPL model.

- *Sets* : A set can be any unordered collection of objects pertinent to a model.
- *Parameters* :A parameter is any numerical value pertinent to a model. The simplest kind of parameter is a single, independent value.
- *Variables* : A linear program's variables are declared much like its parameters. The only substantial difference is that the values of the vari-

ables are to be determined through optimization, whereas the values of the parameters are data given in advance.

- *Objective* : An objective function can be any linear expression in the parameters and variables.

- *Constraints* : A constraint may be any linear equality or inequality in the parameters and variables. Thus a model's constraints use all the same kinds of expressions as its objective.

Chapter 3

ILP Formulations

In this chapter, we propose a approach to minimize the total energy consumption which satisfies the dependency constraints and the total-time constraint. The ILP formulation given by Ishihara. In this thesis we address the following basic concepts to reduce power consumption of given tasks under the dependency constraints and the total-time constraint.

- If given tasks have different performance requirement each other, assigning lower supply voltage to the tasks which require not so high performance.
- If given tasks have different switched capacitance((load capacitance) $\times$ (switching activity)) with each other, assign lower supply voltage to the heavier tasks and higher supply voltage to the lighter tasks.

In section 3.1 we describes the target system assumption. In section 3.2, we define the problem. In section 3.3, we give the notation used in these constraints. In section 3.4, we show the ILP formulation with constraints.

Finally, the ILP formulation is demonstrate with the example.

3.1 Assumptions

The voltage assignment method proposed in this thesis targets systems which satisfy the following assumptions:

- Some tasks are processed on a single processor system under a time constraint.

  The task means fragment of application programs. All tasks must be finished processing until the time constraints.

- Required performances (execution cycle count per second) and the switched capacitance ((load capacitance) $\times$ (switching activity)) of each given task are basically different each other.

  The variance of execution cycles and the switched capacitance ((load capacitance) $\times$ (switching activity)) is strongly related to the effect of power reduction. It seems that the variance of execution cycles and the switched capacitance becomes bigger, the effect of energy reduction by proposed technique becomes also bigger.

- The number of execution cycles of each job can be statically estimated.

  Since the proposed method targets application whose execution cycle counter per unit time must be predicted statically , one of the applications of the proposed method is an real-time processing. In the real-time

system, the number of execution cycles which must be finished processing until specified time constraints.

Next, we discuss on assumptions of microprocessor which embedded in the target system. Following microprocessor must be necessary to allocation optimal supply voltage to each task.

- The V_{DD} and the clock frequency of the microprocessor [IY96] can be varied by an instruction.

  A mechanism to vary the V_{DD} dynamically. Only to have an instruction to vary the V_{DD} and a register to save a state of V_{DD}, V_{DD} scaling is simply realized. In this thesis we call this register and this instruction, *Power control register* and *Power control instruction* respectively. Of course, a compiler support which inserts the Power control instruction into compiled assembly code to vary V_{DD} dynamically is necessary.

- A clock frequency tracks over the variation of the V_{DD} synchronously.

  We need the variable clock scheme whose clock frequency tracks over the V_{DD}. A number of researchers have proposed using a delay ring oscillator [DPO90] to match the critical path of a circuit, and using the frequency of the ring as an indicator of the chip's performance. The chip and the oscillator are built on the same die so that they closely track over process, temperature and voltage.

- A energy consumption of power converter itself such as DC-to-DC converter must be negligible.

  The proposed technique needs a support of a controller to vary the V_{DD}

such as DC-to-DC converter. A power consumption of the controller must be negligible compared with that of the microprocessor. Current research in low power portable electronics includes design of low voltage on-chip DC-to-DC converters, and efficiencies above 90% have been reported[YH96, NYM97]. But we needs still higher efficiencies to achieve lower power consumption of microprocessor.

- Overhead time to vary the V_{DD} and clock frequency is negligible compared with execution time of tasks.

  It is necessary for the V_{DD} assignment optimization technique that on overhead time to vary the V_{DD} and clock frequency is negligible compared with the execution time of the given tasks. The biggest latency to restart the PLL(Phase Lock Loop) tasks 10-100u sec is reported[GS94]. However this is for an analog PLL. A digital PLL has been implemented with a reported lock time of under 2 u sec, drastically reducing the start-up time when fully powered-down mode[LJ94]. It is better for the proposed technique to vary the V_{DD} as faster as possible.

3.2 Voltage Allocation Problem

In this section, we define the voltage allocation optimization problem under a time constraint. We given some constraints in this problem definition, and all tasks must conform with constraints.

Our goal is to minimize the total energy consumption for a given set of tasks satisfies the dependency constraints and the total-time constraint.

We use a DAG(Direct Acyclic Graph) $G = (V, E)$ to represent a task set. In the DAG, a node u represents $task_u$, while an edge $e(u, v)$ indicates that $task_v$ can only start after $task_u$ finishes. A task set after scheduling can still be represented as a DAG with additional edges, if necessary, to capture processor sharing.

We assume that time overhead for changing the supply voltage and clock frequency is negligible. As described in chapter 3 the power loss for the DC-to-DC level converter is negligible.[YH96, NYM97, IY97, IY98, LS00]

The voltage allocation problem is formally defined as follow:

Definition "*For a given set of tasks($task_1, task_2, ...task_N$) and supply voltages $(V_1, V_2, ..., V_L)$, find a voltage allocation sequence $Y_{i,j,k}$ which minimizes the total energy consumption satisfying the dependency constraints and the total-time constraint.*"

3.3 Notation

We define the variables used in the formulation as follows:

- L : the number of variable voltages.
- N : the number of tasks.
- V_j : the j^{th} voltage level ($1 \leq j \leq$L).
- T_{con} : the time constraint until which all tasks must be completed.
- $task_i$: the i^{th} task ($1 \leq i \leq$N).
- S_i : the start time of $task_i$.

- DL_i : the deadline of $task_i$.

- EC_i : the number of execution cycles for $task_i$.

- CT_j : the cycle time when the system runs at voltage j, and

$$CT_j = \frac{k \cdot V_{dd}}{(V_{dd} - V_{th})^2}.$$

- $Y_{i,j,k}$: the voltage allocation variable. $Y_{i,j,k}$ =1 if $task_i$ is executed at voltage j at time k, otherwise $Y_{i,j,k} = 0$.

- $X_{i,k}$: the scheduled variable. $X_{i,k} = 1$ if the $task_i$ scheduled at time k, otherwise $X_{i,k} = 0$. Note that,

$$\sum_{j=1}^{L} Y_{i,j,k} = X_{i,k}.$$

- $NC_{i,j}$: the number of execution cycles for $task_i$ executed at voltage j, and

$$NC_{i,j} = \sum_{k=1}^{T_{con}} Y_{i,j,k}.$$

- d_i : the execution time of $task_i$, d_i can be calculated as follows:

$$d_i = \sum_{j=1}^{L} NC_{i,j} \cdot CT_j = \sum_{j=1}^{L} \sum_{k=1}^{T_{con}} Y_{i,j,k} \cdot CT_j.$$

- SC_i : the average switched capacitance for $task_i$, and

$$SC_i = \frac{\sum_{j=1}^{EC_i} \sum_{k=1}^{M} LC_k \cdot Switch_{i,j,k}}{EC_i},$$

where M is the number of gates in the processor, LC_k is the load capacitance of a gate g_k, and $Switch_{i,j,k}$ is the switching count of g_k while of $task_i$ executes at the j^{th} cycle.

Note that:

- We assume, without loss of generality, that SC_i is usually given in practice.
- In this thesis, a task is defined by its number of execution cycles, and its average switched capacitance.

$$task_i = \{EC_i, SC_i\}$$

3.4 ILP formulation

We formulate the voltage scheduling problem as follows:

(a)*Object Function* : The objective function is to minimize the total energy consumption, which is written as follows.

$Minimize$:

$$\sum_{i=1}^{N}\sum_{j=1}^{L} SC_i \cdot NC_{i,j} \cdot V_j^2 \tag{5}$$

(b)*Switching constraints* : These constraints make sure that the number of switching voltages cannot exceed the number of variable voltages.

$$|\sum_{j=1}^{L}\sum_{k=1}^{T_{con}}(Y_{i,j,k+1} - Y_{i,j,k+1})| \leq 2L \qquad \forall i \in 1 \leq i \leq N \tag{6}$$

(c)*Node constraints* : These constraints guarantee each task must be schedule once only.

$$\sum_{k=1}^{T_{con}} X_{i,k} = 1 \qquad \forall i \in 1 \leq i \leq N \tag{7}$$

(d)*Dependency constraints* : These constraints guarantee that $task_v$ starts after $task_u$ completes if $(u, v) \in$ E.

$$S_v \geq S_u + d_u \qquad \forall (u, v) \in E \tag{8}$$

(e)*Start-time constraints* : These constraints guarantee that the start time of each task will make the task to complete before its deadline.

$$0 \leq S_i \leq DL_i - d_i \qquad \forall i \in 1 \leq i \leq N \tag{9}$$

(f) *Total-time constraint* : The total number of execution cycles for all tasks is no greater then T_{con}.

$$\sum_{i=1}^{N} \sum_{j=1}^{L} NC_{i,j} \cdot CT_j \leq T_{con} \tag{10}$$

3.4.1 Theorem and proof

Theorem The proposed ILP formulation finds a voltage allocation sequence of a task set, with the dependency constraints and the total-time constraint, that consumes the minimal total energy consumption.

Proof : Suppose VA_a is a solution of voltage allocation sequence computed by our proposed ILP formulation, in which each voltage allocation variable is denoted as $Y^a_{i,j,k}$. The number of execution cycles $NC^a_{i,j}$ computed by $NC^a_{i,j} = \sum_{k=1}^{T_{con}} Y^a_{i,j,k}$. The schedule variable $X^a_{i,k}$ is computed by $X^a_{i,k} = \sum_{j=1}^{L} Y^a_{i,j,k}$.
The total energy consumption

$$\sum_{i=1}^{N} \sum_{j=1}^{L} SC_i \cdot NC^a_{i,j} \cdot V_j^2 \tag{A}$$

is minimal subject to

$$\sum_{j=1}^{L}\sum_{k=1}^{T} Y_{i,j,k}^{a} \leq L \qquad \forall i \in 1 \leq i \leq N,$$

$$\sum_{j=1}^{L}\sum_{k=1}^{T} X_{i,k}^{a} = 1 \qquad \forall i \in 1 \leq i \leq N,$$

$$S_v \geq S_u + d_u \qquad \forall (u,v) \in E,$$

$$0 \leq S_i \leq DL_i - d_i \qquad \forall i \in 1 \leq i \leq N, and$$

$$\sum_{i=1}^{N}\sum_{j=1}^{L} NC_{i,j}^{a} \cdot CT_j \leq T_{con}.$$

Assume that there exists another voltage allocation sequence VA_b ($VA_b \neq VA_a$), in which each voltage allocation variable is denoted as $Y_{i,j,k}^{b}$ such that

- The number of times of switching voltage for VA_b cannot exceed the number of variable voltages.
- Each task in VA_b is scheduled once only.
- If two tasks in VA_b have dependency relationship, e.g., $(u,v) \in E$, $task_u$ must complete before $task_v$ starts.
- Each task in VA_b complete processing before its deadline.
- The total execution time of VA_b cannot exceed the time constraint.
- VA_b consumes less energy than VA_a

According to VA_b,The number of times of switching voltage for VA_b cannot exceed the number of variable voltages, and hence

$$\sum_{j=1}^{L}\sum_{k=1}^{T} Y_{i,j,k}^{b} \leq L. \qquad \forall i \in 1 \leq i \leq N$$

Each task in VA_b is scheduled once only, and hence

$$\sum_{j=1}^{L}\sum_{k=1}^{T} X_{i,k}^{b} = 1 \qquad \forall i \in 1 \leq i \leq N.$$

If two tasks in VA_b have dependency relationship, e.g., $(u, v) \in E$, $task_u$ must complete before $task_v$ starts, and hence

$$S_v \geq S_u + d_u \qquad \forall e(u, v) \in E.$$

Each task in VA_b completes processing before its deadline, and hence

$$0 \leq S_i \leq DL_i - d_i \qquad \forall i \in 1 \leq i \leq N.$$

The total execution time of VA_b cannot exceed the time constraint, and hence

$$\sum_{i=1}^{N}\sum_{j=1}^{L} NC_{i,j}^{b} \cdot CT_j \leq T_{con}.$$

The energy consumes of VA_b less than VA_a, and hence

$$\sum_{i=1}^{N}\sum_{j=1}^{L} SC_i \cdot NC_{i,j}^{b} \cdot V_j^2 \leq \sum_{i=1}^{N}\sum_{j=1}^{L} SC_i \cdot NC_{i,j}^{a} \cdot V_j^2.$$

Since the energy consumption of VA_a is minimal by assumption, we have reached a contradiction with (A).

This completes the proof that the proposed ILP formulation finds a voltage allocation sequence of the task set, with the dependency constraints and the total-time consumption, that consumes the minimal total energy consumption.

□

3.5 Example

As an example, the figure 1.1 shows four tasks. We assume $task_1$ must complete before $task_2$.We give ILP formulation for the task set. They are the switching constraints, the node constraints, the dependency constraints, the start-time constraints, and total-time constraint.

3.5.1 Switching constraints

$Y_{1,1,1}+Y_{1,1,2}+Y_{1,1,3}+Y_{1,1,4}+Y_{1,1,5}+ Y_{1,1,6}+Y_{1,1,7}+Y_{1,1,8}+Y_{1,1,9}+Y_{1,1,10}+$
$Y_{1,1,11}+Y_{1,2,1}+Y_{1,2,2}+Y_{1,2,3}+Y_{1,2,4}+ Y_{1,2,5}+Y_{1,2,6}+Y_{1,2,7}+Y_{1,2,8}+Y_{1,2,9}+$
$Y_{1,2,10}+Y_{1,2,11} \leq 2$

$Y_{2,1,1}+Y_{2,1,2}+Y_{2,1,3}+Y_{2,1,4}+Y_{2,1,5}+ Y_{2,1,6}+Y_{2,1,7}+Y_{2,1,8}+Y_{2,1,9}+Y_{2,1,10}+$
$Y_{2,1,11}+Y_{2,2,1}+Y_{2,2,2}+Y_{2,2,3}+Y_{2,2,4}+ Y_{2,2,5}+Y_{2,2,6}+Y_{2,2,7}+Y_{2,2,8}+Y_{2,2,9}+$
$Y_{2,2,10}+Y_{2,2,11} \leq 2$

$Y_{3,1,1}+Y_{3,1,2}+Y_{3,1,3}+Y_{3,1,4}+Y_{3,1,5}+ Y_{3,1,6}+Y_{3,1,7}+Y_{3,1,8}+Y_{3,1,9}+Y_{3,1,10}+$
$Y_{3,1,11}+Y_{3,2,1}+Y_{3,2,2}+Y_{3,2,3}+Y_{3,2,4}+ Y_{3,2,5}+Y_{3,2,6}+Y_{3,2,7}+Y_{3,2,8}+Y_{3,2,9}+$
$Y_{3,2,10}+Y_{3,2,11} \leq 2$

$Y_{4,1,1}+Y_{4,1,2}+Y_{4,1,3}+Y_{4,1,4}+Y_{4,1,5}+ Y_{4,1,6}+Y_{4,1,7}+Y_{4,1,8}+Y_{4,1,9}+Y_{4,1,10}+$
$Y_{4,1,11}+Y_{4,2,1}+Y_{4,2,2}+Y_{4,2,3}+Y_{4,2,4}+ Y_{4,2,5}+Y_{4,2,6}+Y_{4,2,7}+Y_{4,2,8}+Y_{4,2,9}+$
$Y_{4,2,10}+Y_{4,2,11} \leq 2$

3.5.2 Node constraints

$$\sum_{j=1}^{L} Y_{i,j,k} = X_{i,k}.$$

$$X_{1,1}+X_{1,2}+X_{1,3}+X_{1,4}+X_{1,5}+X_{1,6}+ X_{1,7}+X_{1,8}+X_{1,9}+X_{1,10}+X_{1,11} = 1$$

$$X_{2,1}+X_{2,2}+X_{2,3}+X_{2,4}+X_{2,5}+ X_{2,6}+X_{2,7}+X_{2,8}+X_{2,9}+X_{2,10}+ X_{2,11} = 1$$

$$X_{3,1}+X_{3,2}+X_{3,3}+X_{3,4}+ X_{3,5}+X_{3,6}+X_{3,7}+X_{3,8}+X_{3,9}+ X_{3,10}+X_{3,11} = 1$$

$$X_{4,1}+X_{4,2}+X_{4,3}+ X_{4,4}+X_{4,5}+X_{4,6}+X_{4,7}+X_{4,8}+ X_{4,9}+X_{4,10}+X_{4,11} = 1$$

3.5.3 dependency constraints

$3 \geq 0 + d_u$

3.5.4 Start-time constraints

$0 \leq 0 \leq 10 - d_1$

$0 \leq 3 \leq 8 - d_2$

$0 \leq 5 \leq 10 - d_3$

$0 \leq 8 \leq 11 - d_4$

3.5.5 Total-time constraint

$$NC_{1,1} \cdot CT_1 + NC_{1,2} \cdot CT_2 + NC_{2,1} \cdot CT_1 + NC_{2,2} \cdot CT_2 + NC_{3,1} \cdot CT_1 + NC_{3,2} \cdot CT_2 + NC_{4,1} \cdot CT_1 + NC_{4,2} \cdot CT_2 \leq 11$$

Chapter 4

Experimental Results

We use AMPL to express the mathematical constraints and optimizations before feeding the ILP problem into the CPLEX solver.

We show the experimental result for a set of tasks and its detail as shown in Figure 4.1 and Figure 4.3. When the set of tasks are given and processed until the time constraint, we plot the total energy consumption varying the dependency constraints and the total-time constraint in Figure 4.2 and 4.4. In the experiments in this section, we assume that different voltages can be assigned to every 1.0×10^8 cycles for convenience. This assumption makes accuracy of optimal solution worse a little, but this decrease of accuracy is negligible compared with energy reduction by proposed technique.

At first, we show the experimental result for the set of tasks as shown in the Figure 4.1. Column S_i, DL_i, EC_i, SC_i show the the start time of $task_i$, the number of execution cycles for $task_i$, the deadline of $task_i$, the average switched capacitance for $task_i$. The variable supply voltage, L is two: 1.3V and 0.8V,

$TaskSet$	S_i	DL_i	EC_i	$SC_i(nF)$
$TaskSet_1$	(0,10,15)	(10,20,40)	$(50, 50, 50) \times 10^8$	(10,10,10)
$TaskSet_2$	(2,8,14)	(15,25,38)	$(50, 50, 50) \times 10^8$	(8,10,12)
$TaskSet_3$	(1,6,13)	(10,24,33)	$(50, 50, 50) \times 10^8$	(6,10,14)
$TaskSet_4$	(3,7,12)	(9,22,35)	$(50, 50, 50) \times 10^8$	(4,10,16)
$TaskSet_5$	(0,5,10)	(12,23,42)	$(50, 50, 50) \times 10^8$	(2,10,18)
$TaskSet_6$	(1,7,13)	(10,27,38)	$(50, 50, 50) \times 10^8$	(2,8,20)
$TaskSet_7$	(4,9,14)	(10,21,40)	$(50, 50, 50) \times 10^8$	(2,6,22)
$TaskSet_8$	(1,9,14)	(6,20,37)	$(50, 50, 50) \times 10^8$	(2,4,24)
$TaskSet_9$	(2,10,16)	(8,24,41)	$(50, 50, 50) \times 10^8$	(2,2,26)
$TaskSet_{10}$	(3,9,15)	(9,23,43)	$(50, 50, 50) \times 10^8$	(1,1,28)

Figure 4.1: Specification of given task set 1-4.

$TaskSet$	EDF(nJ)	Our ILP(nJ)	Reduction(%)
$TaskSet_1$	253.5	218.5	13.8%
$TaskSet_2$	253.5	211.5	16.6%
$TaskSet_3$	253.5	207.3	18.2%
$TaskSet_4$	253.5	197.5	22.1%
$TaskSet_5$	253.5	190.5	24.8%
$TaskSet_6$	253.5	183.5	27.6%
$TaskSet_7$	253.5	176.5	30.3%
$TaskSet_8$	253.5	169.5	33.1%
$TaskSet_9$	253.5	162.5	35.9%
$TaskSet_{10}$	253.5	155.5	38.6%

Figure 4.2: Energy consumption for task set 1-4

and the number of execution cycles for task sets are the same in this example. If the load of processing(switched capacitance) can be biased intentionally, task scheduling to reduce energy consumption can be achieved. Also the switched capacitances differ from case to case, sum of switched capacitances of each task in this example is constant. experimental results are shown in Figure 4.2.

The result in Figure 4.2 shows that if the switched capacitance ((load capacitance) $\times$ (switching activity)) differ from case to case, assigning lower supply voltage to the heavier tasks and higher supply voltage to the lighter

tasks leads to power reduction satisfying the dependency constraints and total-time constraint. Comparing $taskset_1$ with $taskset_10$, energy reduction by 29%. Energy consumptions are decreased according to the time constraint is increased. This is clearly because, when the time constraint is increased V_{DD} can be reduced under the dependency constraints and the total-time constraint.

Next, we show the experimental result for the tasks as shown in the Figure 4.3. This experiment makes clear the impact of variation of variable V_{DD} upon energy reduction. In this example we prepare two variations of set of variable V_{DD}. One set of V_{DD} includes two level: 1.3V and 0.8V, and the other set includes three levels: 1.3V, 1.0V, 0.8V. The combination of the execution count of each task are same in this example. Although the combinations of switched capacitances of each task are differ from case to case, total switched capacitances are constant. Experimental results are show in Figure 4.4. The L in Figure 4.4 represent the kinds of variable voltage levels. For example if $L = 3$, the processor can choose the supply voltage from 1.3V, 1.0V and 0.8V.

The result in Figure 4.4 shows that if the kinds of variable voltage level is increased, energy consumption of any cases are decreased at least.

Summary of all experimental results mentioned above are as follows.

- If the V_{DD} of processor is selectable from 1.3V and 0.8 V, the energy consumption is halved at maximum case when T_{con} is enough long. Increase of T_{con} has strong impact on the energy reduction, if the microprocess equips with variable V_{DD} and clock scheme. Energy for the processing of

$TaskSet$	S_i	DL_i	EC_i	$SC_i(nF)$
$TaskSet_a$	(0,18,32)	(20,32,50)	$(100, 100, 100) \times 10^8$	(10,10,10)
$TaskSet_b$	(1,15,30)	(20,34,52)	$(100, 100, 100) \times 10^8$	(8,10,12)
$TaskSet_c$	(2,19,34)	(23,37,49)	$(100, 100, 100) \times 10^8$	(6,10,14)
$TaskSet_d$	(0,17,31)	(18,26,51)	$(100, 100, 100) \times 10^8$	(4,10,16)
$TaskSet_e$	(1,16,30)	(15,35,56)	$(100, 100, 100) \times 10^8$	(2,10,18)
$TaskSet_f$	(2,15,33)	(16,32,59)	$(100, 100, 100) \times 10^8$	(2,8,20)
$TaskSet_g$	(4,20,34)	(18,33,64)	$(100, 100, 100) \times 10^8$	(2,6,22)
$TaskSet_h$	(1,16,35)	(19,37,62)	$(100, 100, 100) \times 10^8$	(2,4,24)
$TaskSet_i$	(3,18,30)	(21,38,53)	$(100, 100, 100) \times 10^8$	(2,2,26)
$TaskSet_j$	(2,19,32)	(22,34,58)	$(100, 100, 100) \times 10^8$	(1,1,28)

Figure 4.3: Specification of given task set a-c.

$TaskSet$	EDF(nJ)	$L = 2(nJ)$		$L = 3(nJ)$	
$TaskSet_a$	507	381.0	24.85 %	309.37	38.98 %
$TaskSet_b$	507	368.4	27.33 %	302.48	40.34 %
$TaskSet_c$	507	355.8	29.82 %	298.26	41.17 %
$TaskSet_d$	507	318.0	37.28 %	286.13	43.56 %
$TaskSet_e$	507	297.0	41.42 %	267.14	47.30 %
$TaskSet_f$	507	280.2	44.73 %	261.94	48.33 %
$TaskSet_g$	507	263.4	48.04 %	253.92	49.92 %
$TaskSet_h$	507	246.6	51.36 %	240.28	52.60 %
$TaskSet_i$	507	229.8	54.67 %	226.64	55.30 %
$TaskSet_j$	507	210.9	58.40 %	209.32	58.71 %

Figure 4.4: Energy consumption for task set a-c

application program which need only low performance can be halved.

4.1 Discussion

If the variable V_{DD} level L is 2, the proposed problem is solved easily by greedy algorithm. A naive algorithm which assigns higher voltage to the task which has lowest switched capacitance prior finds optimal solution. But for the large L and N, algorithms for reduction of computation time is required.

For large N which represents the number of given tasks, computation time to find best solution can be very large. If the N is large for solving the ILP problem within practical time, we can solve LP problem and find quasi-optimal solution instead. If we formulate the proposed problem as LP problem, we must get round the solution of LP problem into an integer. But if the numbers of cycles of tasks are increased, rounding error is negligible compared with energy reduction.

For large L, searching the whole search space can be inefficient in terms of computation time. But considering practical use of power converter such as DC-to-DC converter, L will not exceed 5 or there abouts. Since L and N are small enough considering practical use of the proposed method, most of voltage allocation optimization problem must be solved in practical time.

Rather than employing to establish techniques for speed-up, incorporating more precise model is important. we must be paying attention to following expansion of proposed model.

- Incorporating overhead time and energy of DC-to-DC converter into the

model.

This expansion is easily achieved. Only to reflect the effect of overhead time and the energy consumption of DC-to-DC converter in our proposed Formulation, accuracy of the optimization becomes better. Actual values will be measured from actual design of DC-to-DC converter.

Chapter 5

Conclusion

In this thesis, we present and formally defined a system level power optimization problem : A voltage allocation problem to assign optimal supply voltage to each task under the dependency constraints and the total-time constraint, and formulate this optimization problem as an integer linear programming(ILP) problem.

To evaluate the effects of power reduction by the proposed method, we use some examples of tasks and estimate the energy consumption under variable time constraints. Experimental results show as follow.

- If the supply voltage of processor is selectable from 0.8V and 1.3V, the energy consumption is halved at maximum case when T_{con} is enough loosened. To loosen the T_{con} has strong effect on the energy reduction, if the microprocessor equips with variable supply voltage and clock scheme.

- Even if the T_{con} is constant, assigning lower supply voltage to the heavier tasks and higher supply voltage to the lighter tasks reduces total energy

consumption by 30% at maximum case. For the application program whose load is widely varied by operation, proposed method reduces energy consumption.

- Increase of variety of variable supply voltage has weak impact on energy reduction. If variable supply voltage level is increased from L= 2 to L= 3 total energy consumption is reduce by 10% at most. It can be said that variable voltage level L= 2 or 3 is appropriate.

Bibliography

[BT00] T.Burd et al., "A Dynamic Voltage Scaled Microprocessor System," *Proc. IEEE Int'l Solid-State Circuits Conf.(2000 ISSCC), IEEE Press,Piscataway, N.J.*,2000, pp.294-295.

[BB95] T. Burd and R. Brodersen., "Energy efficient CMOS Microprocessor Design," *Proc. 28th International Conference on System Sciences.,IEEE Press, Annual Hawaii*,1995, pp.288-297

[BB96] Thomas D. Burd and Robert W. Brodersen, "Processor design for portable systems," *Journal of International VLSI Signal Processing*, 1996

[CC96] Intel Corporation and Microsoft Corporation, *Advance Power Management(APM) BIOS Interface Specification Revision 1.2*, Februaly 1996 http://www.intel.com/IAL/powermgm/apmv12.pdf.

[DPO90] M. DegrauweP. Machen, M. Van Paemel, and M. Oguey, " A Voltage Reduction Technique for Digital Systems," *In Proc. of IEEE* ISSCC, pp.238-239, 1990.

[ECS] C.S. Ellis., "The Case for Higher-Level Power management," *Proc. 7th IEEE Workshop On Hot Topics in Operating Systems*, 1999, pp. 162-167.

[EC99] C. Ellis. "The case for higher-level power management" *In IEEE 7th Workshop on Hot Topics in Operating Systems*, pp.162-167, 1999

[FGK90] R. Fourer, D. Gay, and B. Kernighan, 1990. *AMPL: A Mathematical Programming Language*, Scientific Press, San Francisco, CA.

[GS94] S. Gray, et al., "The PowerPC 603 Microprocessor: A Low-Power Design for Portable Applications," *in Proc. of the 39th IEEE Computer Society International Conference,* March 1994

[HI98] I Hong et al.,"Power Optimization of variable Voltage Core-Based Systems,"*Proc. 35th Design Automation Conf.(DAC98), ACM Press, New York*,1998, pp. 176-181.

[HP98] I. Hong, M. Potkonjak,and M.B. Srivastava, "On-Line Scheduling of Hard Real-Time Tasks on Variable-Voltage Processor," *Proc. IEEE/ACM Int'l Conf. Computer-Aided Design A(ICCAD 98), ACM Press, New York*,1998, pp.653-656.

[IBMC90] International Business Machines Corporation, 1990."Optimization Subroutine Library," Licensed Program Numbers 5688-137, 5601-469, 5621-013.

[IY96] T. Ishihara and H. Yasuura, "Power-Pro: Programmable Power Management Architecture (in Japanese)," Technical Report VLD96-72, IEICE, DECEMBER 1996.

[IY97] T. Ishihara and H. Yasuura. "Optimization of Supply Voltage As-

signment for Power Reduction on Processor-Based System".*in Proc. of SASIMI '97*, 1997, pp. 51-58.

[IY98] T. Ishihara and H. Yasuura, "Voltage Scheduling Problem for Dynamically Variable-Voltage Processors," *Proc. 1998 Int'l Symp. Low Power Electronics and Design(ISPLED98), ACM Press, New York*, 1998,pp. 197-202

[KK03] Kwon, W., and Kim, T. "Optimal voltage allocation techniques for dynamically variable voltage processors".*Proc. IEEE DAC*, June 2003, pp. 125V130

[LS00] S. Lee and T. Sakurai, "Run-Time Power Control Scheme Using Software Feedback Loop for Low-Power Real-Time Application,"*Proc. 2000 Asia and South Pacific Design Automation Conf.(ASPDAC 2000)*, ACM Press, New York, 2000, pp.381-386.

[LS00] S. Lee and T. Sakurai, "Run-Time Voltage Hopping for Low-Power Real-Time Systems," *Proc. 37th Design Automation Conf.(DAC 00)*, ACM Press, New York, 2000,pp. 806-809.

[LJ94] J. Lundberg, et al., "A 15-150 MHz All-Digital Phase-Lock Loop with 50-Cycle Lock Time for High-Performance Low-Power Microprocessors," *In Proc. of symposium on VLSI Circuit,* June, 1994.

[LJR98] J.R. Lorch and A.J. Smith., "Apple Macintosh's energy consumption."*IEEE Micro*,1998, 99.54-63.

[MS00] T. Ma and K. Shin. "A User-Customizable Energy-Adaptive Combined Static/Dynamic Scheduler for Mobile Applications." *In Proceedings of 21th IEEE Real-Time systems Symposium*, pp. 227-236, 2000

[MOICG97] H. Mehta, R. M. Owens, M. J. Irwin, R. Chen and D. Ghosh. "Techniques for low Energy Software", *In Proceedings of the 1997 International Symposium on Low Power Electronics*

[NNSB97] Lars S. Nielsen, Cees Niessen, Jens Spars, and Kees van Berkel, "Low-Power operation Using Self-Timed Circuits and Adaptive Scaling of the Supply Voltage," *IEEE Trans. on VLSI system*, vol.2, no.4 pp.391-397, December 1997

[NYM97] N. Namgoong, M. Yu, and T. Meng, "A High-Efficiency Variable-Voltage CMOS Dynamic DC-DC Switching Regulator," *ISSCC'97*, pp. 380-381.

[OIY99] T. Okuma, T. Ishihara, and H. Yasuura, "Real-Time Task Scheduling for a Variable-Voltage Processor," *Proc. 12th Int'l Symp. System Synthesis(ISSS98)*, IEEE CS Press,Los Alamitos,Calif.,1999,pp.25-29.

[OnNow] http://www.microsoft.com/hwdev/onnow.html.

[PBB00] T.Pering,T.Burd,and R.Brodersen,"Voltage Scheduling in the lpARM Microprocessor System," *Proc. Int'l Symp. Low Power Electronics and Design(ISPLED 00), ACM Press, New York*, 2000, pp.96-101.

[QH01] Gang Quan, Xiaobo Hu, "Energy efficient fixed-priority scheduling for real-time systems on variable voltage processors", *Proceedings of the*

38th conference on Design automation, p.828-833, June 2001, Las Vegas, Nevada, United States.

[QH02] Gang Quan, Xiaobo Hu, "Minimum Energy Fixed-Priority Scheduling for Variable Voltage Processor", *Proceedings of the conference on Design, automation and test in Europe*, p.782, March 04-08, 2002

[SAN] http://www.sandpile.org/

[SC99] Y. Shin and K. Choi, "Power Conscious Fixed Priority Scheduling for Hard Real-Time Systems,"*Proc. 36th Design Automation Conf.(DAC99), ACM Press, New York*,1998, pp.134-139.

[SCS00] Youngsoo Shin , Kiyoung Choi , Takayasu Sakurai, "Power optimization of real-time embedded systems on variable speed processors", *Proceedings of the 2000 IEEE/ACM international conference on Computer-aided design*, November 05-09, 2000, San Jose, California

[TC00] "Crusoe Processor," Transmeta Corp., Santa Clara, Calif.,http://www.transmeta.com/crusoe/.

[TCE95] C. E. Tellez, A. Farrahi and M. Sarrafzadeh. "Activity-driven clock design for low power circuits." *In proceedings of the IEEE International Conference on Computer Aided Design*, 1995

[VRJ] R. J. Vanderbei, "ALPO: Another Linear Programming Optimizer," ORSA J. Computing, to appear.

[YDS95] F. Yao , A. Demers , S. Shenker, "A scheduling model for reduced

CPU energy", *Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS'95)*, p.374, October 23-25, 1995

[YH96] Gu-Yeon Wei and Mark Horowitz, "A Low Power Switching Power Supply for Self-Clocked Systems," *In Proc. of ISLPED'96,* pp.313-317, 1996.