System Level Optimization Techniques for Low Power VLSI Design

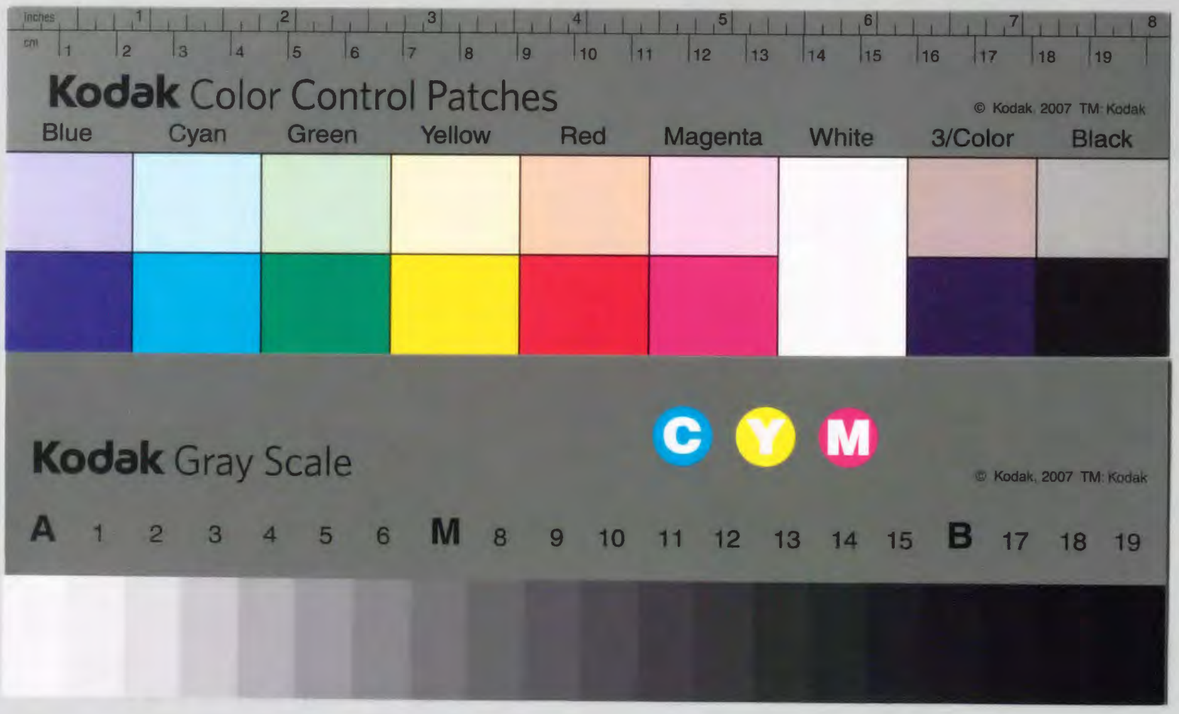
石原, 亨
九州大学システム情報情報工学

https://doi.org/10.11501/3166839

出版情報：九州大学, 1999, 博士（工学）, 課程博士
バージョン：
権利関係：

System Level Optimization Techniques for Low Power VLSI Design

石原亨



System Level Optimization Techniques for Low Power VLSI Design

Tohru Ishihara

Kyushu University

November 1999

Abstract

Low power design has emerged as an attractive theme both practically and theoretically in modern VLSI (Very Large Scale Integration) system design. Recently many power optimization techniques at various levels of abstractions such as at the layout, circuit, architectural, and system levels are proposed. In todays system design, power optimizations at higher level of abstraction are required, because decisions at the higher level of abstraction strongly affect to the cost, performance, and power consumption of final products. This thesis presents system-level power optimization techniques. A brief summary of system-level low-power design approaches combined with my own works will be described. It reviews some techniques that have been proposed to overcome power issues and gives guidelines for prospective system-level low-power designs.

One of the most effective approaches for power reduction is voltage reduction, because the power consumption of CMOS circuits is quadratically proportional to the supply voltage. However, lowering the supply voltage leads to an increase of circuit delay. Therefore, system designers have to determine operating supply voltage, taking the power-delay trade-off into account. This thesis, first, introduces a new concept called voltage scheduling. This deals with a dynamically variable supply voltage. The voltage scheduling can formally be defined as follows: *For a given task, determining a schedule of the processor's supply voltage so as to minimize the energy consumption for the task under a time constraint.* An integer linear programming (ILP) model, theorems and an algorithm for the voltage scheduling problem are also presented in this thesis. Target systems include a variable voltage processor which can dynamically vary its supply voltage but can use only a single supply voltage at a time. For a given application program and the dynamically variable voltage processor, the algorithm finds a voltage schedule for each task so as to minimize the energy consumption satisfying a timing constraint.

This thesis also presents *Power-Pro* architecture (Programmable Power Management Architecture), a novel processor architecture for power reduction. The *Power-Pro* architecture has two key functions : (i) The supply voltage and clock frequency of a microprocessor can be dynamically varied, and (ii) the active datapath width can be dynamically adjusted to the precision of each operation. This architecture aims to enable the software to dynamically control the active datapath width and the supply voltage. To make this possible, *Power-Pro* architecture employs complex instructions. With those instructions, programmers can dynamically vary the supply voltage, the clock frequency and the active datapath width. Experimental results show that power consumption for a variety of applications are dramatically reduced by the *Power-Pro* architecture.

Reducing the energy consumption throughout the whole system including hardware and software is one of the goals of this study. The hardware is often constructed by an embedded processor, an instruction memory and a data memory. The software and data which are executed and processed in the processor are usually stored in instruction memory and a data memory, respectively. Therefore, not only the processor but also the memories should be optimized for low power. A memory power optimization technique based on object code merging is also presented in this thesis. Basic idea is to merge sequences of frequently executed object codes into a set of single complex instructions. This complex instructions are restored by an instruction decompressor before decoding the object codes. The decompressor is implemented by ROM, and no modification to processor architecture is required for any application programs. Therefore, the technique is well suited for systems with embedded IP (Intellectual Property) cores whose internal architecture cannot be modified. In many programs, only a few object code sequences are frequently executed. Therefore, merging these frequently executed sequences into a set of single instructions leads to a significant energy reduction. Our experiments with actual read only memory (ROM) modules and some benchmark programs demonstrate significant energy reductions of up to more than 50% compared with processor-based systems without the object code merging.

The power reduction techniques which are proposed in this thesis can be applied together to a wide range of digital systems. Therefore, significant power reduction can be expected by these techniques, even if the power improvement by each of the techniques is modest. The variable datapath width control scheme can reduce the energy consumption of both the data memory and the processor's datapath by 35%. The energy consumption of the processor can

be reduced by 90% at best via the variable voltage control scheme. The energy consumption of the instruction memory can be halved by the object code merging technique. For example, if the control logic circuit of the processor, the processor's datapath, the instruction memory, and the data memory dissipate 25% of the total energy in a conventional system, we can reduce two third of the total energy dissipation in the same system by applying our proposed techniques.

Contents

Chapter 1

Introduction

1.1 Background

In past years, the most serious concerns for the VLSI designer were area, performance, cost, and reliability. Therefore, power considerations were mostly of only secondary issue. Recently, however, this paradigm is shifted, and power becomes more and more important issue. The primary driving factor of this paradigm shift must be explosive growth in the portable systems which demand high-speed computation and complex functionality with low power consumption.

In these applications power consumption is a critical design concern. The projected power consumption for a portable multimedia terminal when implemented components are not optimized for low-power operation is around the range of 10-50 W[58]. With advanced Nickel-Metal-Hydride battery technologies yielding around 65 watt-hours/kilogram, this terminal would require an unacceptable six kilograms of batteries for ten hours of operation between recharges[37].

There also exists a strong pressure for designers of high-end products to reduce their power consumption. Contemporary performance optimized microprocessors dissipate as much as 15-50W at 100-200MHz clock rates[58]. In the future, it can be expected that a 10 cm^2 microprocessor, clocked at 500MHz would consume about 300W. The cost associated with packaging and cooling such devices is huge. Consequently, there is a clear advantage to reducing the power consumed in high performance systems. Conversely, lowering the power is indispensable for consumer products whose sales are strongly affected by its price. There-

fore, for VLSI chips which are embedded in the consumer products, a rising heat of the chips becomes one of limiting factor to realize higher transistor density and computational speed. Though the motivations for reducing power consumption differ from application to application, power reduction is an essential theme in whole of today's electronics industry.

The most of low power system design process can be divided into the following two phases. (i) The power estimation, and (ii) the power optimization. In todays system design, power estimation and optimization at higher level of abstraction are required, because decision at the higher level of abstraction strongly affect to the cost, performance, and power consumption of final products. This thesis will present both power estimation and optimization techniques at the system level abstraction. A VLSI system design can be represented at several levels of abstraction such as at the layout, logic circuit, architectural, and system levels. The hardware design process is often performed by gradually detailing the abstract specification to lower level of abstraction. A behavioral level specifies the functionality of the design and may contain no structural information. A structural level represents the circuit as an interconnection of elements or building blocks. Synthesis tools can automatically convert or refine a design from a higher level of abstraction to a lower level of abstraction, or can convert a behavioral level description to a structural level description. At the system level, the design may be modeled as a set of abstract communicating processes or tasks, with no knowledge of whether the tasks are implemented in hardware or compiled into software running on an embedded processor. System level synthesis involves partitioning the tasks into hardware and software, choosing the processor architecture that will execute the software, determining the hardware/software communication mechanism, and so on. This thesis proposes power optimization techniques at the system level abstraction. These techniques perform the power optimization taking both hardware and software into account, and have much more impacts on the quality of the final products than lower level optimization techniques have.

1.2 Goal of This Research

The goal of this research is to develop not only a low power system but also a high performance system with low power dissipation. Low power technologies support development higher density transistor and increased computational speeds in future VLSIs, because the heat produced by the VLSI chips is one of the limiting factors in developing larger scale and higher

speed computer systems. Low power is also essential to high performance battery powered systems. Another important issue for these applications is not only power consumption, but also energy consumption, that is a summation of power consumption. The goal of this research is to reduce both power consumption and energy consumption of VLSI systems.

To realize high performance computation with low power consumption, a power-delay optimization should be done. Basic idea of the power-delay optimization is to lower the supply voltage as much as possible satisfying a computation time constraint. The power-delay optimization technique can dramatically reduce power and energy consumption without essential performance degradation. This thesis presents theories for the power-delay optimization in the system design level. This thesis also proposes a technique to reduce redundant switching on datapath circuits. This technique can reduce energy consumption in the datapath circuits and a data memory with a trivial performance degradation.

Developing low power system within very short time is also the goal of this study, because turn around time (TAT) in VLSI system design has strong impacts on the sales of products in these days. One of the most effective ways to reduce the time to market is design reuse. This thesis propose a memory power optimization technique which aims to develop low power instruction memory within very short design time.

Energy reduction techniques presented in this thesis can be applied together to wide range of digital systems. Therefore, the great energy reduction can be expected by these techniques, even if the energy improvement by each technique is modest.

1.3 Policy of This Research

Since the energy consumption of CMOS circuits is almost proportional to switching activity, load capacitances and the square of supply voltage, lowering these design parameters leads to the energy reduction. However, lowering the supply voltage causes performance degradation, because the clock frequency is almost proportional to the supply voltage. Our policy is to reduce energy consumption without essential performance degradation as described below.

- Eliminate wasteful switching activities which are not essential to the result of the computation.

- Saving energy by eliminating extra computational speed which is excessive for a required computational speed.
- Reduce load capacitances of frequently used parts, even if the load capacitances of rarely used parts are increased.

Basic idea of the approach to eliminate wasteful switching is to inactivate unnecessary bits of a datapath. For example, if all bits of a datapath in a 32-bit processor are switched for the computation of 8 bit data, many wasteful switching are occurred in upper 24 bits of the datapath. Inactivating such a wasteful switching activity, we can drastically reduce the power consumption without any changes of computation scheme.

A scheme to save energy by eliminating an extra computational speed which is excessive for a required computational speed is also proposed in this thesis. In CMOS transistor, power dissipation is quadratically proportional to supply voltage. Lowering the supply voltage has strong impact on power reduction for CMOS circuits. However, this causes computational speed degradation. We can reduce the power consumption by lowering the supply voltage until the computational speed of a system matches a desired computational speed. A compiler technique which can find the supply voltage which adjust the computational speed of the target system to the required computational speed are addressed in this thesis.

This thesis also presents a compiler technique to reduce load capacitances of frequently used parts in an instruction memory. A basic idea is to merge frequently executed basic blocks into a set of single complex instructions. Although the load capacitances of rarely used memory blocks are increased, total energy consumption can be dramatically reduced.

1.4 Contributions of This Research

The energy consumption of whole system, E_{global}, can be defined as the summation of both partial and temporal power consumption of circuits as shown in (1.1).

$$E_{global} = \sum_{i=1}^{X} \sum_{k=1}^{N} p_{ik} \tag{1.1}$$

The p_{ik} denotes a gate g_k's power dissipated during the ith period of time, X the execution time of a program (software), and N the number of gates in the VLSI chip. In this thesis, we treat E_{global} as an objective function to be optimized, because the energy consumption

directory affects the heat of chips and the life of battery. Since the p_{ik}s are dynamically varied according to the behavior of the software and a location of the logic gate on a chip, both the software and the hardware should be taken into account to reduce the energy consumption. This thesis proposes the system level optimization techniques which consider behavior of both the hardware and the software. The most important contribution of this research is to develop methods which enables the software to manage the energy consumption, and methods to optimize application specific systems utilizing a history of the application programs. Detailed contributions of this research are described below.

- A compiler technique to determine the optimal supply voltage which minimizes energy consumption under a time constraint is proposed.
- Some theorems which give guidelines to find an optimal operating supply voltage for microprocessors in practical situations are proved.
- A microprocessor architecture which enables the software to control the supply voltage and active datapath width dynamically is proposed.
- A reconfigurable instruction memory architecture targeting low power application specific systems is proposed.
- An object code merging technique to reduce power dissipation in an instruction memory is proposed.

First, a new microprocessor architecture, called *Power-Pro : Programmable Power Management Architecture*, is presented. This architecture can vary its supply voltage and active datapath width by special instructions. They make it possible to control the power consumption and the performance of microprocessors by software.

Next, a compiler technique to determine the optimal supply voltage for dynamically variable voltage processors, which minimizes energy consumption under a time constraint is proposed. A procedure to deciding optimal supply voltage for given tasks is called voltage scheduling. Two kinds of information are required for the voltage scheduling. One is a voltage-delay relations of the dynamically variable voltage processor. The other information is a program source in which real-time constraints are explicitly specified. Firstly, the compiler estimate the worst case execution cycles of the application program. Secondly, the

compiler finds an optimal voltage schedule, using the informations of the estimated execution cycles of the program and the voltage-delay relations of the processor. The voltage scheduling is done based on the theorems presented in chapter 3. The optimal voltage schedule minimizes energy consumption without any real-time violations. Finally, an object code including the special instructions which can vary the supply voltage is generated for the dynamically variable voltage processor. A problem to find optimal voltage schedule is formulated as an integer linear programming (ILP) problem. An voltage scheduling algorithm for the problem is also proposed in this thesis.

A compiler technique with object code merging to reduce power dissipation of an instruction memory is also proposed. This technique targets a microprocessor which has an instruction decompressor to restore the merged sequences of object codes. Merging frequently executed sequences of object codes into a set of single complex instructions reduces energy during memory access, because the number of memory access to the main program memory is extremely reduced. However, merging too many sequences of object codes into single instructions leads to an increase of energy consumption in the instruction decompressor. The proposed technique finds optimal point of this trade-off where total energy consumption is minimized. A problem to find the optimal point of the trade-off is formulated as an integer linear programming (ILP) problem. An algorithm to solve the problem is also proposed.

1.5 Organization of This Thesis

This thesis addresses mainly tree topics, variable voltage scheduling, a programmable power management architecture, and an object code merging for application specific systems, and is organized as follows.

First, in Chapter 2, low power design methodologies at the system level abstraction are outlined.

Chapter 3 presents the variable voltage scheduling method for the dynamically variable voltage processors. Chapter 4 presents a programmable power management architecture which enables the software to control supply voltage and active datapath width. Chapter 5 proposes a new architecture for embedded instruction memories, and presents a memory power optimization technique with an object code merging.

Chapter 6 concludes this thesis with a summary and a direction of future research.

Chapter 2

Low Power System Design

2.1 Power Dissipation Models

With the popularization of portable electronics and the rising demands for cooling down the heat of high-end products, it becomes more important for VLSI systems to reduce power consumption. Recently, many power minimization schemes at various levels of abstraction, such as at device, circuit, layout, architectural and system levels are proposed[43]. As for the low level design, power optimization techniques are well studied. However, there is much scope left to study in the system level area such as architectural, algorithm, or software level. In this chapter, we present system level power and energy reduction approaches.

In many applications, not only power dissipation but also energy dissipation is critical design concern. We can define the energy consumption as the summation of both partial and temporal power consumption in a VLSI circuit as shown in Figure2.1 and 2.3.

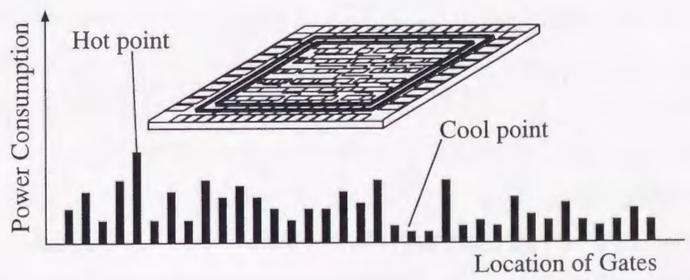


Figure 2.1: Partial power dissipation in CMOS circuits.

In this thesis, we treat the energy consumption as an objective function to be improved, because the energy consumption directory affects the heat of chips and the life of the battery. Our challenge is to model and optimize the energy consumption of whole system at the higher level of abstraction. The dominant source of energy dissipation throughout a digital CMOS circuit synchronized by a system clock is the dynamic energy dissipation,

$$E_i = \sum_{k=1}^{M} CL_k \cdot Swit_{ik} \cdot V_{DD}^2 \tag{2.1}$$

where E_i is energy dissipation while the ith clock cycle is executed, M the number of gates in the circuits, CL_k the load capacitance of a gate g_k, $Swit_{ik}$ the switching count of g_k while the ith clock cycle is executed, and V_{DD} the supply voltage. Basically, the dynamic energy arises only when the capacitive load of the output of CMOS circuit is charged through the power supply or is discharged to ground as shown in Figure 2.2.

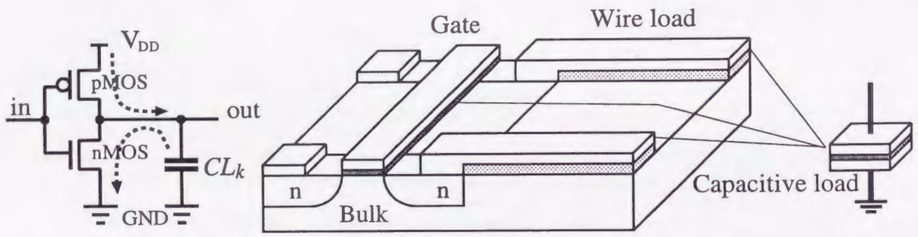


Figure 2.2: Power dissipation in CMOS circuits.

Some researchers have proposed several accurate energy dissipation model for CMOS circuits [59, 10, 28, 5], and make sure accuracy of their models. We have also examined the accuracy of (2.1) by measuring the energy dissipation of actual chips[48, 49]. An analog ammeter, an oscilloscope, and an in-house energy measuring instrument are used to measure the energy of the chips. The experimental result with actual chips and many kinds of test vectors demonstrates that the maximum error of (2.1) is 12% at most, and the accuracy of (2.1) is very good.

Next, let us consider a task j with the number of total execution cycles X_j, where the task means a fragment of a program. Since the energy for the task is a summation of E_i by the number of the execution cycle X_j, the energy consumption for the task is formulated as (2.2).

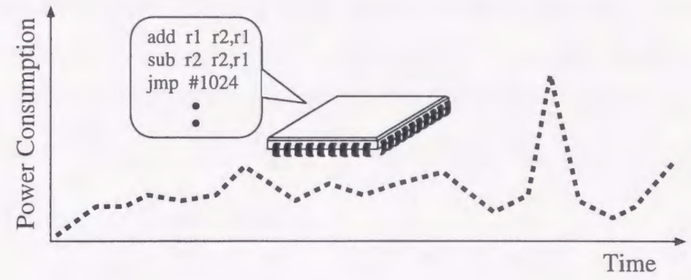


Figure 2.3: Temporal power dissipation in CMOS circuits.

$$E_{task} = \sum_{i=1}^{X_j} E_i = \sum_{i=1}^{X_j} \sum_{k=1}^{M} CL_k \cdot Swit_{ik} \cdot V_{DD}^2 \tag{2.2}$$

We can reduce the energy consumption for the task by lowering V_{DD}, $Swit_{ik}$, CL_k, M, or X_j. However, lowering these design parameters causes increase of execution time for the task. The circuit delay τ which determines maximum clock frequency of VLSI systems synchronized by a system clock is formulated as (2.3), and an execution time for the task T_{task} can be formulated as (2.4),

$$\tau \propto \frac{V_{DD}}{(V_G - V_T)^2} \sim \frac{1}{V_{DD}} \tag{2.3}$$

$$T_{task} = \tau \cdot X_j \tag{2.4}$$

where V_T is the threshold voltage, and $V_G (\sim V_{DD})$ the voltage of the input gate. Formulas (2.2) and (2.4) demonstrate design trade-offs in CMOS VLSI systems, because of the following reasons.

- The E_{task} is quadratically proportional to the V_{DD}, and the T_{task} is inversely proportional to the V_{DD}.
- The X_j strongly depends on the CL_k, M, and the $Swit_{ik}$. For example, highly parallelized circuits which require large CL_ks, M, and $Swit_{ik}$s can execute the task within a small X_j. Conversely, serialized computation which requires small M, CL_ks, and $Swit_{ik}$s needs large X_j.

Our challenge is to minimize the total energy consumption for the task under a given computation time constraint. According to the relations among V_{DD}, $Swit_{ik}$, CL_k, X_j, E_{task} and T_{task}, we can consider the following approaches for the minimization of the energy consumption.

1. Lowering the V_{DD} under the computation time constraints.

2. Reducing the $\sum_{i=1}^{X_j} \sum_{k=1}^{M} (Swit_{ik})$ keeping the T_{task}.

3. Reducing the $\sum_{i=1}^{X_j} \sum_{k=1}^{M} (CL_k \cdot Swit_{ik})$ keeping the T_{task}.

4. Optimizing the X_j so as to minimize the E_{task} under the computation time constraints.

In this chapter, we will make a brief survey on the above approaches in system level LSI design and show several examples in detail. The rest of the chapter is organized in the following way. In Section 2.2, we present approaches for supply voltage optimization. Approaches for lowering the switching activity and frequently activated capacitance are presented in section 2.3 and 2.4, respectively. Section 2.5 presents the optimization of the number of the execution cycles to minimize the energy for the task. The chapter is concluded in Section 2.6.

2.2 Optimization of Supply Voltage

Since energy dissipation is quadratically proportional to supply voltage V_{DD} (see equation (2.2)), controlling V_{DD} has strong impacts on energy reduction[41]. A main difficulty of energy reduction by V_{DD} optimization is how to resolve the trade-off between system performance and energy dissipation formulated in (2.2) and (2.4). We have to develop design methods to minimize energy consumption keeping system performance required in specifications of systems. Basically, we have the following two ways to reduce V_{DD} in system level design.

2.2.1 Power Reduction by Parallel Computation

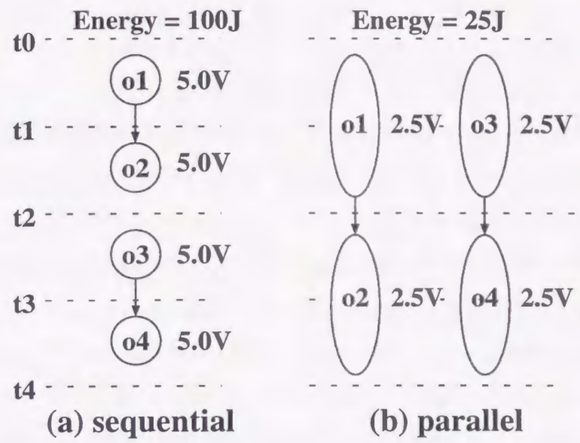


Figure 2.4: Power reduction by parallel computation.

This method utilizes performance improvement by parallel computation. Suppose that our computation is completely parallelizable into two parallel tasks. Introducing two circuits for the tasks, we can reduce the performance of each task without loss of system performance as shown in Fig. 2.4. If ideally we can reduce the performance of each circuit into half of the original one, we can reduce V_{DD}s of them into half because of the power-performance relation of equation (2.3). Although load capacitance and switching activities in a cycle may increase up to twice respectively, we can reduce the number of cycles into half and V_{DD}^2 into quarter. Although this is a very attractive approach, parallelization of computation is difficult in general because of the control and data dependency in many programs. Some computations are inherently sequential and it is difficult to enjoy the power reduction scheme for them.

In the past few years, multiple supply voltage scheduling technique were proposed [26, 39, 34]. The techniques refer to the assignment of a supply voltage to each operation in a data flow graph so as to minimize the average energy consumption under given computation time or throughput constraints or both. Since in such techniques, supply voltage is statically

assigned to each functional module, they become ineffective if performance requirements dynamically change according to the operating conditions.

2.2.2 Power Reduction by Adaptive Voltage Scaling

More practical approach is adaptive power control techniques. Since the load of our computation is not constant, we can control power consumption according to the required computation load. Preparing multiple supply voltages, we can assign them for tasks of computation keeping the total performance of the system. The assignment can be done statically or dynamically. In the rest of this section, we will discuss the adaptive power control.

Low power techniques with dynamic voltage scaling also have been studied [60, 33, 18]. Nielsen et al. have shown a low-power system using self-timed circuits. The circuits achieve maximum power savings by lowering the supply voltage until the performance of chip can meet the specific performance requirement. In this system, the supply voltage is controlled by hardware directly. This can be a disadvantage for some kinds of applications, because it is difficult to know detailed behavior of the application program from the hardware. Therefore, a technique to control the supply voltage from software is required for sophisticated power management system.

Intel and Microsoft have proposed a specification called Advanced Power Management (APM)[13]. This specification defines an interface between power management software in BIOS, and a hardware-independent power management driver in operating system. This driver can manage APM-aware applications, by notifying them processor state changes. Although the APM makes possible to manage the power consumption from application program, how to decide the optimal operating supply voltage, or hardware specification which control the supply voltage from application program are not presented.

We have proposed a dynamically variable voltage processor architecture called *Power-Pro* architecture in [52, 50]. The architecture can use dynamically variable supply voltage but can use a single supply voltage at a time. The architecture has special instruction, called *voltage control instruction* to vary its own supply voltage and clock frequency. Using the instruction, software programmers can directly specify the operating supply voltage of the hardware in an application program. The most important issue for the variable voltage processor is how to select suitable voltages according to the operating condition. We have also proposed a voltage

scheduling problem which treats dynamically variable supply voltage, and have proposed theorems and an algorithm for the problem in [53, 50, 56]. In[24, 63] theories to determine the optimal supply voltage for dynamically variable voltage processors are presented.

In many practical applications, the number of cycles is changed depending on the sort of data. Since it is difficult to know statically the number of execution cycles, which are often much smaller than the worst case, an application of the voltage scheduling by the compiler will be limited. Dynamic voltage scheduling by operating systems will be an important technique for system level power reduction[63]. A real-time operating system which assigns voltage as low as possible to each task without any real-time violations has proposed[56].

Developing low power DC-DC converters is one of the most important themes for variable voltage VLSIs, because the power dissipation of the DC-DC converter is not negligible for very low power VLSI circuits. Recently, many types of DC-DC converters have been proposed [18, 44, 41, 46, 16]. In [16], a DC-DC converter whose energy efficiency reaches up to 95% in the supplying power range of 40-100mW is proposed. These low power DC-DC converters promise the variable voltage techniques to be essential in prospective low power applications.

2.3 Reducing the Switching Activity

In system level design, we have three basic approaches to reduce the switching activity.

2.3.1 Reducing Wasteful Switching Activity

Usually, computation contains many wasteful switching activities, which are not essential to the result of the computation. For example, all bit lines of data bus in a 32-bit processor are switched for the computation of 8 bit data. If we take care of these wasteful switches, we can drastically reduce power consumption without any changes of computation scheme. Gated clock is a popular method in this approach[36, 61, 8].

2.3.2 Optimizing Signal Protocols and Encoding

Protocols and encoding schemes of long communication lines, which have large capacitive loads, strongly affect power consumption. In system level design, the protocols and coding schemes can be designed for minimizing power consumption. For application specific system

design, statistic information on data and/or control flows are useful for the optimization. As examples of this approach, optimization of ordering of data transfer and coding on buses are discussed by several researchers[42, 21].

2.3.3 Optimization by Algorithm Selection

Above two approaches do not require essential change of computation algorithms. The number of switching is, however, inherently depends on the computation algorithm. Unfortunately, techniques of algorithms selection for power reduction have not been established yet. But it is a very important research area both theoretically and practically. In order to demonstrate the effects of the algorithms selection, we show, in table 2.2, an example of energy consumption of different multiplication algorithms in table 2.1[55]. All multiplier circuits are automatically synthesized with an automatic synthesis tool of SYNOPSYS co. ltd., and automatic place and route tool of Avant! co. ltd.. Energy consumption of each circuit is estimated with post-layout simulation. Bit width of the all the multipliers are 16 bits. Toggle count is a criterion of switching activity. The toggle count of Mult-3 is 70% of one of Mult-4. The energy consumption of Mult-3 is also 70% of one of Mult-4.

Table 2.1: Specifications of multipliers.

Circuits	Algorithms
Mult-1	A array multiplier with Booth's algorithm
Mult-2	A Wallace tree multiplier with Booth's algorithm
Mult-3	A multiplier with 7-3 parallel counters
Mult-4	A multiplier with redundant binary adders

Table 2.2: Switching activity of multipliers.

Circuits	Area[μm^2](# of cells)	Delay[ns]	Toggle count/Cycle	Energy/Cycle [pJ]
Mult-1	666462.7 (786)	23.76	926.95	65.48
Mult-2	800846.2 (1,067)	15.77	779.78	65.40
Mult-3	978017.0 (1,344)	12.98	604.37	63.97
Mult-4	896409.4 (1,440)	13.95	861.42	87.15

2.4 Reducing the Frequently Switched Capacitance

2.4.1 Memory Power Optimization

Some parts of a system are more active than other parts. According to equation (2.2), reducing load capacitance of the active parts leads to energy reduction of a whole system. This kinds of design efforts are usually done in circuit and layout design levels. But there are rooms to discuss efforts on the switching capacitance reduction in system level design. In many applications, only a few parts of programs are frequently used. Furthermore, some parts of the programs or some parts of circuits are not activated for some kinds of input data. Therefore, reducing the load capacitance of frequently accessed memory blocks or frequently activated parts of circuits may be effective way to reduce total energy consumption[32]. Here, we will show a challenge of the power reduction of memory-intensive systems, such as video image processing and speech recognition [54]. Detailed discussion will be done in chapter 5.

2.4.2 Power Reduction for Cache Memory Systems

Because of the large load capacitance in off chip buses, local computation with cache memory is one of the most effective way to reduce power consumption during memory access. Some low power cache approaches which are aiming to reduce the cache miss ratio have proposed [40, 29, 20]. In [30], way-predictable set-associative cache memory architecture is proposed. Basic idea is to predict which way of four way contain a data desired by a processor. The experimental results demonstrate that about 70% energy improvement is achieved by the way predicting cache with 10% performance degradation, compared with a conventional set-associative cache.

2.4.3 Optimization of Standard Cell Library

An application specific library generation techniques for low power VLSI design has been proposed[19, 47]. Main purpose of the techniques are to suit gate size, P/N size ratio, or a function set to the applications. In [47], a function set of cell library is determined for each target circuit so as to take frequently switching interconnects into cells. For example, if an output node of a NOR gate in Figure 2.5 is frequently switched, this node should be taken into a complex gate so as not to dissipate a wasteful power in this node. When the

logic function Y is realized by a complex gate and the logic value of the C is **1**, no energy is dissipated in the complex gate ideally.

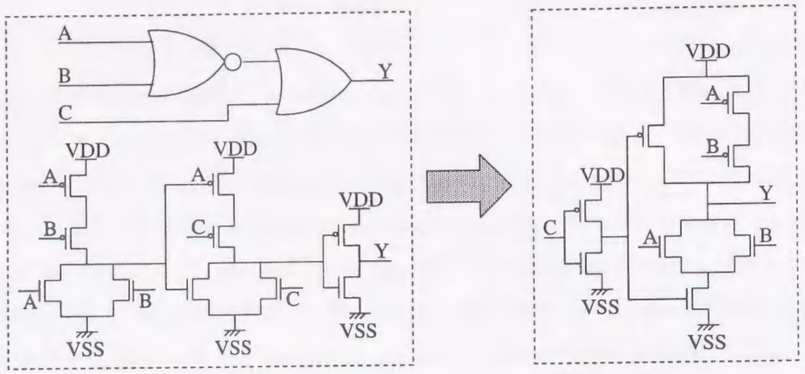


Figure 2.5: Power optimization with complex gates.

It is wide consensus that a power consumption in interconnects becomes much dominant in the near future. Therefore, the application specific library generation techniques can be much important power reduction technique in the future. The experimental results with some benchmark circuits demonstrate that this library generation technique can reduce much power consumption.

2.5 Optimization of the Number of Execution Cycles

A glance at the equations (2.2) and (2.4) shows that reducing the number of execution cycles (X_j) leads to the reduction of both energy consumption and execution time for the task. However, reducing the number of execution cycles without increase of M, CL_k, or $Swit_{ik}$ is difficult in general (See equations (2.2) and (2.4)). In addition, reducing X_j may cause an increase of circuit delay τ (See equation (2.3)), because the most simple approach to reduce the X_j is utilizing highly parallelized large circuits, and large circuits often contains long interconnects which cause large wire delay. Therefore, techniques which resolve trade-offs among CL_k, $Swit_{ik}$, τ and X_j are required for energy reduction. In this section, we present

a technique to resolve the trade-offs among CL_k, $Swit_{ik}$, and X_j by tuning up the datapath width of a core processor.

The datapath width of a processor strongly affects to the power consumption in the system including, the processor, data RAMs, and instruction ROMs. In the following discussion, we suppose that the precision of computation should be preserved.

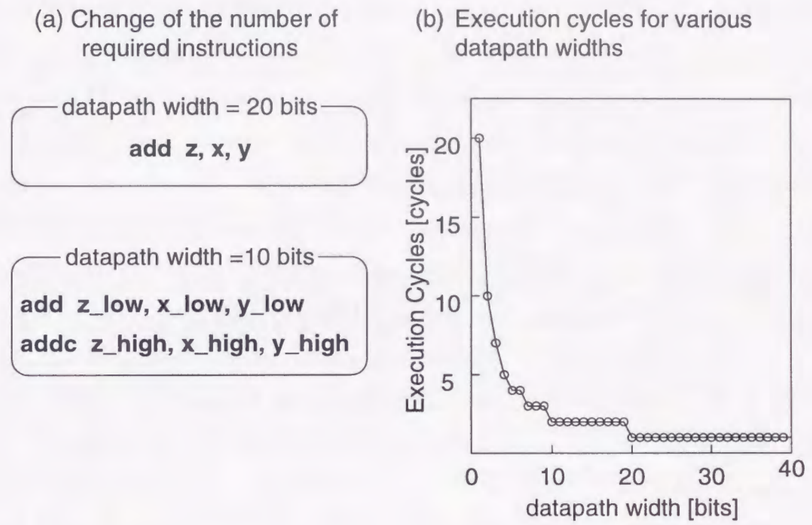


Figure 2.6: The number of execution cycles for various datapath width.

The datapath width of the processor also affects to the number of execution cycles of a given task, i.e., narrowing the datapath width causes the increase of the number of execution cycles because of the increase of cycles for multiple-precision operations. Assume that an addition of 20 bit data shown in Figure 2.6 is executed by only one instruction on a 20 bit processor. If the datapath width is 10 bits, two instructions including additions of lower 10 bits and higher 10 bits with carry (See Fig. 2.6 (a)) are required. Figure 2.6 (b) demonstrate trade-offs between datapath width and the number of execution cycles. Although a processor with narrower datapath width dissipates lower power per clock cycle, the total energy for the task can not always be reduced by narrowing the datapath width[1].

To determine whether computing with wide bit width and small number of execution cycles, or computing with narrow bit width and large number of execution cycles is a major concern for a hardware and software design. In this section, we present a hardware/software codesign approach for the datapath width optimization which minimize the energy consumption under computation time constraints.

Most important issue for the datapath width optimization is how to determine the effective size of variables. The effective size of a variable stands for the minimum required size which can hold both the largest and smallest values of the variable. A method to analyze effective sizes of variables in C program is proposed in [22]. A programming language and a compiler technique which support programmers to specify the minimum required bit width for each variable directly in source program was proposed in [23, 2]. The proposed language and the compiler are called *Valen-C* and *Valen-C* compiler, respectively. The *Valen-C* compiler preserves the precision of programs in the following manner: If a variable v is defined as a n bit variable, the *Valen-C* compiler allocates physical storage resources (registers and memory words) of not less than n bits for v. Arithmetic and logic operations for v are also performed with the precision of not less than n bits. In cases that the precision of an operation is larger than the datapath width, the operation is performed by more than one machine cycles as a multi-precision operation.

A new design methodology of embedded systems based on the *Valen-C* and a user defined application specific processor, called a *soft-core* processor, is proposed in [23]. The *soft-core* processor is a prototype of an embedded processor design, which has some design parameters, such as the datapath width, the number of registers, the kind of functional units, and the size of memories[17]. Designers can change the parameters for each application, and then obtain a customized processor optimized for the application. In other words, the total power consumption of a system including the *soft-core* processor, ROM and RAM can be optimized by changing the word length.

A system level energy optimization technique using *Valen-C* and the *soft-core* processor is proposed in [1]. Their objective is to optimize the datapath width so as to minimize the energy consumption for the task using the method of variable size analysis, the *Valen-C*, and the *soft-core* processor.

2.6 Summary

This chapter reviews some system level low power design techniques that have been proposed to overcome the power issue, and demonstrate that a system level decision has strong effect to energy consumption as well as to system cost and performance. Although hardware approaches for power reduction at higher level of abstraction are studied well, there may be much scope left to explore in software area. The key technologies that we believe to be important in prospective VLSI system design are summarized below.

- Create a model of the system including hardware and software at higher level of abstraction.
- Estimate power consumption of the system including hardware and software at higher level of abstraction.
- Optimize the total energy consumed by the system for given applications considering behavior of both hardware and software.

Power concerns must be most important issue for VLSI system design, and breakthrough to overcome the power issue will be required in the near future. This thesis will outline future challenges to develop high performance and low power VLSI systems.

Chapter 3

Variable Voltage Scheduling

3.1 Background

Lowering the supply voltage has strong impact on power reduction, because the power consumption of CMOS circuits is quadratically proportional to the supply voltage. However the circuit delay is almost inversely proportional to the supply voltage. Basically, therefore, system designers have to consider the trade-off between a computational speed and energy consumption to realize high performance computation with low power consumption. In recent researches, datapath scheduling techniques and behavioral synthesis techniques with multiple supply voltage were proposed [26, 39, 62]. The proposed scheduling problem refers to the assignment of a supply voltages to each operation in a data flow graph so as to minimize the average energy consumption for given computation time or throughput constraints or both. The experimental results demonstrates 20-30% power reduction. However, these techniques can sometimes be ineffective when the computation time constraints are dynamically varied, because such techniques statically assign the supply voltage to each functional module. Power reduction techniques which treat dynamically variable supply voltage have much effect for actual systems in which the computation time constraints are changed according to the performance requirements of applications.

3.1.1 Motivation

In the past few years, some low power techniques by dynamic voltage scaling have been studied [60, 33, 18, 45]. L. Nielsen et al. has shown a low-power system using self-timed circuits

and adaptive scaling of the supply voltage in [33]. The self-timed circuits achieve maximum power savings by lowering the supply voltage until the chip can just meet the specific performance requirement. Their approach scales supply voltage dynamically according to the quantity of processing data per unit time. Similar to our approach, their approach can find optimal supply voltage which is fitting to the desired performance. However, the problem definition is quite different from our approach, because their approach is based on an adaptive method. Our approach is based on an intentional voltage scheduling method[53, 50]. In this chapter, we address a voltage scheduling problem and formulate it to an ILP(Integer Linear Programming) problem. Target systems include the dynamically variable voltage processor which can dynamically vary its supply voltage but can use only a single voltage at a time[52].

This chapter is organized in the following way. In section 3.2 and 3.3, we prove basic and more generalized theorems, respectively. Section 3.4 and 3.5 present ILP formulation and an algorithm for the problem, respectively. Experimental results are shown in section 3.6. Section 3.7 concludes this chapter.

3.1.2 Power Delay Trade-off

The dominant source of energy dissipation in a digital CMOS circuit is the dynamic energy dissipation, as shown in (2.1), and is quadratically proportional to the supply voltage. Conversely, the circuit delay is inversely proportional to the supply voltage as shown in (2.3). As is shown in Fig. 3.1, (2.1) and (2.3) demonstrates the power-delay trade-off in CMOS circuits.

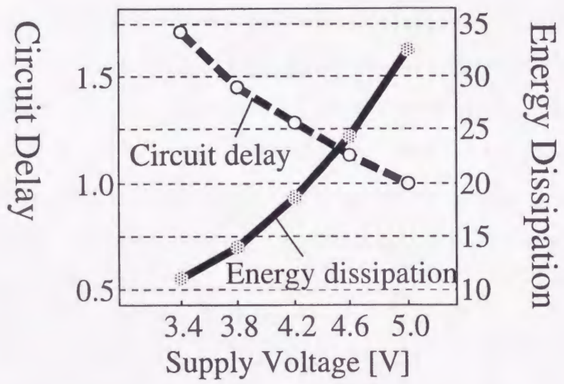


Figure 3.1: The power-delay trade-off in CMOS circuits.

System designers have to take this design trade-off into account to reduce energy consumption without essential performance degradation.

3.1.3 Motivational Example

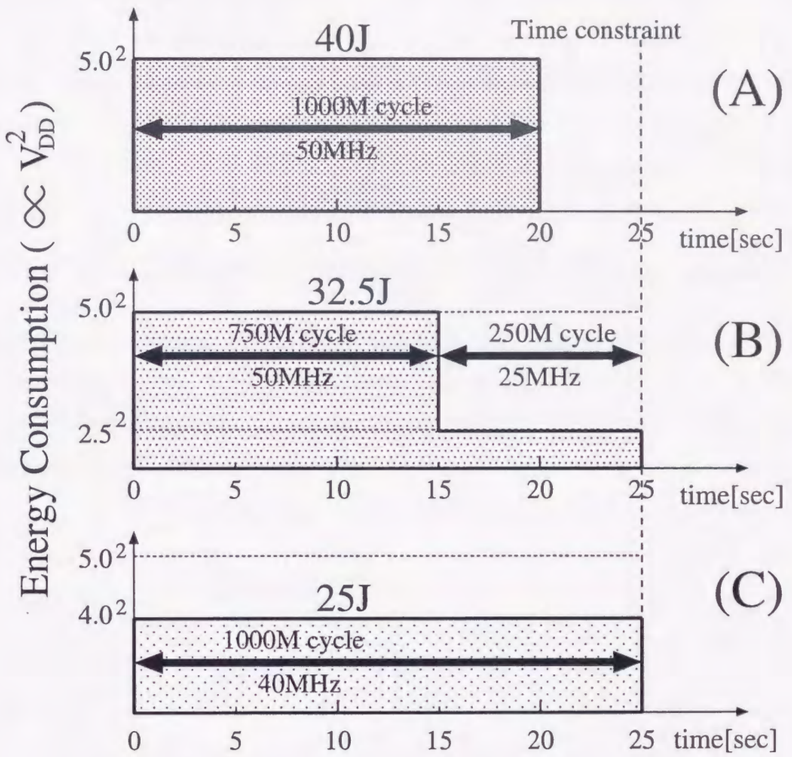


Figure 3.2: An example of power-delay optimization.

The main purpose of our power saving approach is to optimize the power-delay trade-off. The goal of the power-delay optimization is minimizing the power consumption by scheduling the supply voltage for each task under a given time constraint, where the task is a fragment of a program. Assume that the energy consumptions for a given program are 10nJ/cycle, 25nJ/cycle and 40nJ/cycle at 2.5V, 4.0V and 5.0V, respectively. The computational speeds

of the processor with 5.0V, 4.0V, and 2.5V are $(50 \times 10^6$ clock cycles)/second, $(40 \times 10^6$ clock cycles)/second, and $(25 \times 10^6$ clock cycles)/second, respectively. This assumption roughly accords to (2.1) and (2.3). Figure 3.2 shows three voltage schedules for the given program whose total execution cycles are 1000×10^6 cycles. In Fig. 3.2(A), the total energy consumption is 40J, because a processor uses only a 5.0V supply voltage. Given a time constraint of 25 seconds, the voltage scheduling with 2.5V and 5.0V which adjusts the finishing time to the given timing deadline reduces the energy consumption from 40J to 32.5J. Fig. 3.2(C) shows the lower bound case of this example. If the processor uses a single supply voltage which adjusts the finishing time just to the given timing deadline, the total energy consumption is minimized. We generalize these features in next section.

3.2 Basic Theorems on a Simple Model

In this section, we target processors which assume to use continuously variable voltage between 0[V] and V_{max}[V](> 0). The energy consumption is assumed to be independent from operation types or input data but depends only on supply voltage. We regards the number of execution cycles to be a continuous value in this chapter, since the number of execution cycles is assumed to be enough big to regard a continuous value.

In deep submicron electronics, it is necessary to consider the effect of mobility degradation. After this, we use more accurate delay model (3.1) rather than (2.3), which takes the effect of mobility degradation into account[12],

$$\tau = \frac{k \cdot V_{DD}}{(V_{DD} - V_T)^\alpha} \tag{3.1}$$

where α is a parameter which reflects the effect of mobility degradation upon circuit delay, and k is constant.

Lemma 1 *If a processor completes a program before the timing deadline T ($0 <$ execution time $< T$), the energy consumption is not minimized.*

Proof. At first, we prove that the circuit delay $\tau(v)$ and energy consumption of CMOS circuits are the monotonous function of supply voltage v.

$$\text{Let} \quad \tau(v) = \frac{k \cdot v}{(v - V_T)^\alpha} \qquad (k \text{ is a constant})$$

can not minimize energy consumption if the processor can use continuously variable voltages.

Proof. For any v, x, and α which satisfy $0 < V_T < V_1 < v < V_2 < V_{max}$, $0 \le x \le Cycle$, and $1 \le \alpha \le 2$, let us prove (3.3) under the constraint (3.2). Where $Cycle$ stands for total execution cycles of the given program.

$$\frac{x \cdot V_1}{(V_1 - V_T)^\alpha} + \frac{(Cycle - x) \cdot V_2}{(V_2 - V_T)^\alpha} = \frac{Cycle \cdot v}{(v - V_T)^\alpha} \tag{3.2}$$

$$V_1^2 \cdot x + V_2^2 \cdot (Cycle - x) > v^2 \cdot Cycle \tag{3.3}$$

The left side of (3.2) and (3.3) represent the total execution time and normalized energy consumption, respectively, when x cycles are executed with voltage V_1 and $Cycle - x$ cycles are executed with voltage V_2. The right side of (3.2) and (3.3) represent, total execution time and energy consumption, respectively, when all cycles are executed with voltage v.

$$\text{Put} \quad T = \frac{x \cdot V_1}{(V_1 - V_T)^\alpha} + \frac{(Cycle - x) \cdot V_2}{(V_2 - V_T)^\alpha} = \frac{Cycle \cdot v}{(v - V_T)^\alpha}$$

$$f = V_1^2 \cdot x + V_2^2 \cdot (Cycle - x), \quad \text{and} \quad g = v^2 \cdot Cycle,$$

then, x and v are functions of T. f and g are functions of x and v respectively.

Therefore, both f and g are function of T. The result which is differentiated by T is described below.

$$\frac{d\,f}{dT} = \frac{(V_1^2 - V_2^2) \cdot (V_1 - V_T)^\alpha \cdot (V_2 - V_T)^\alpha}{V_1 \cdot (V_2 - V_T)^\alpha - V_2 \cdot (V_1 - V_T)^\alpha} = constant < 0,$$

$$\text{and,} \quad \frac{d^2 g}{dT^2} = \frac{\alpha \cdot \{2 \cdot V_T + (\alpha - 1) \cdot v\}(v - V_T)^{2\alpha+1}}{Cycle^2 \cdot \{V_T + (\alpha - 1) \cdot v\}^3} > 0$$

It is obvious that f is a linear function of the timing constraint T as shown in Figure3.4. In addition, g is a below convex function of the timing constraint T.

Consequently, we have (3.4) under the condition (3.2).

$$f - g > 0 \tag{3.4}$$

$$\text{we have, } \frac{d\tau(v)}{dv} = -\frac{(\alpha-1)+k\cdot V_T}{(v-V_T)^{\alpha+1}}$$

Therefore, for α $(1.0 \leq \alpha \leq 2.0)$ and V_T $(0 < V_T < V)$, the $\tau(v)$ is monotonously decreased function of voltage v as shown in Figure3.3. The α strongly depends on the mobility degradation of electrons in MOS transistors, and tends to be decreased from 2.0 to 1.0 according to the channel size shrinkage. Threshold voltage V_T also tends to be decreased so as to improve transition delay and propagation delay of transistors.

It is immediate that the energy consumption is monotonously increased function of voltage v as shown in Figure3.3. Therefore, high speed processing with high energy can be replaced by low speed processing with low energy. Namely, if a processor completes the processing before the timing deadline, slower and lower energy processing than such processing is possible. This lemma substantiates a well known power-delay trade-off in CMOS circuits for any practical α and V_T. □

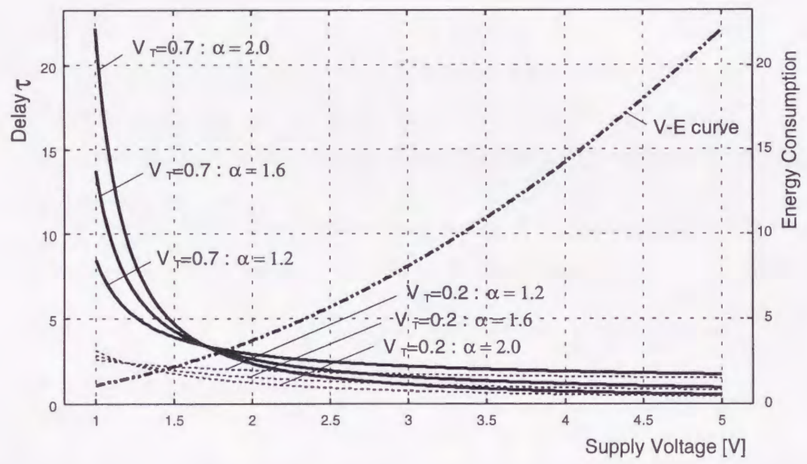


Figure 3.3: V-τ curve for various V_T and α.

Lemma 2 *If a processor uses a single supply voltage v and completes a program just at a timing constraint $T(>0)$, the v is an unique supply voltage which minimizes energy consumption for the given program. In other words, voltage scheduling with multiple supply voltage*

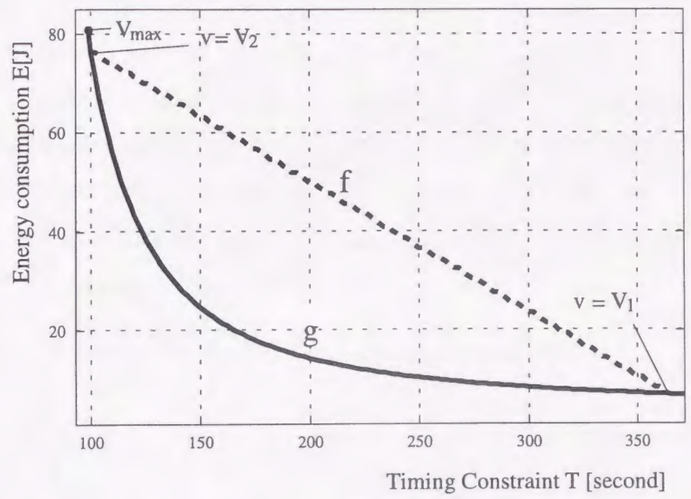


Figure 3.4: Energy consumption versus timing constraint.

(3.4) indicates that the voltage scheduling with two voltages can not minimize energy consumption. Furthermore, it is obvious that the voltage scheduling with more than two voltages can not minimize energy consumption. □

An important information from *Lemma 1* and *Lemma 2* is that the voltage scheduling with the single voltage is optimal even if the transistor size and V_T are scaled down.

For ideal systems which can supply continuously variable voltages, *Lemma 2* can be derived. However, using continuously variable voltages may be infeasible[11], since suppling any kinds of stable supply voltages and clock frequency wastes the significant power and hardware cost. In other words, processors do not always have a specified voltage which minimizes energy consumption. For processors which have only a small number of discretely variable voltages, we have the following theorem.

Theorem 1 *If a processor can supply only a small number of discretely variable voltages, the voltage scheduling with at most two voltages minimizes the energy consumption under any time constraint.*

Proof. Consider the following two kinds of voltage scheduling : *schedule-1* is the schedule with three voltages V_1, V_2, and V_3 ($0 < V_1 < v_{ideal} < V_2 < V_3$), and *schedule-2*, the schedule with two voltages V_1 and V_2. Now, let the execution cycles with V_1, V_2, and V_3 in *schedule-1* be $x1(\geq 0)$, $x2(\geq 0)$, and $x3(\geq 0)$, respectively. In addition, $y1(\geq 0)$ and $y2(\geq 0)$ cycles are assumed to be executed with V_1 and V_2 in *schedule-2*, respectively. We prove (3.5), under the constraints of (3.6) and (3.7).

$$V_1^2 \cdot x1 + V_2^2 \cdot x2 + V_3^2 \cdot x3 \geq V_1^2 \cdot y1 + V_2^2 \cdot y2 \tag{3.5}$$

$$\frac{x1 \cdot V_1}{(V_1 - V_T)^\alpha} + \frac{x2 \cdot V_2}{(V_2 - V_T)^\alpha} + \frac{x3 \cdot V_3}{(V_3 - V_T)^\alpha} = \frac{y1 \cdot V_1}{(V_1 - V_T)^\alpha} + \frac{y2 \cdot V_2}{(V_2 - V_T)^\alpha} \tag{3.6}$$

$$x1 + x2 + x3 = y1 + y2 \tag{3.7}$$

The left and right sides of (3.5) represents the energy consumption of *schedule-1* and *schedule-2*, respectively. (3.6) and (3.7) explain that the total execution time and the execution cycles of two schedules must be same.

Firstly, assuming $x1 < y1$. If $x1 < y1$, the constraints (3.6) and (3.7) can not simultaneously be satisfied. Therefore, the situation $x1 < y1$ can not be occurred. We have only to consider the situation $x1 \geq y1$ as shown in Figure 3.5. In this situation, (3.5), (3.6) and (3.7) can be deformed as (3.8), (3.9), (3.10), and (3.11).

$$(x1 - y1) + x3 = y2 - x2 \tag{3.8}$$

$$\frac{(x1 - y1) \cdot V_1}{(V_1 - V_T)^\alpha} + \frac{x3 \cdot V_3}{(V_3 - V_T)^\alpha} = \frac{(y2 - x2) \cdot V_2}{(V_2 - V_T)^\alpha} \tag{3.9}$$

$$V_1^2 \cdot (x1 - y1) + V_3^2 \cdot x3 \geq V_2^2 \cdot (y2 - x2) \tag{3.10}$$

$$x1 - y1 > 0, \quad y2 - x2 > 0, \quad x3 > 0 \tag{3.11}$$

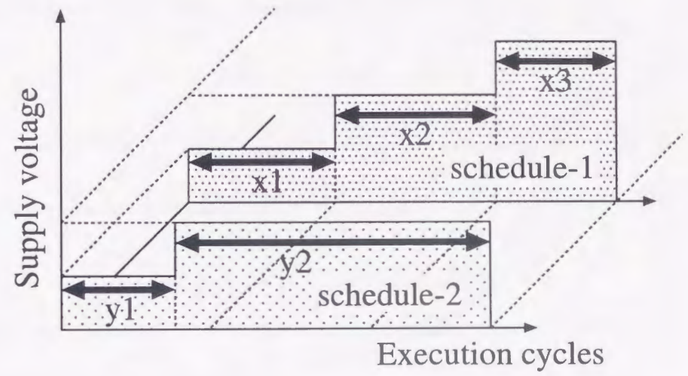


Figure 3.5: Voltage schedule-1 versus schedule-2.

From *Lemma 2*, (3.10) can be immediately proved under the constraints of (3.8), (3.9), and (3.11). Therefore, voltage scheduling with V_1, V_2, and V_3 can not minimize energy consumption, when $0 < V_1 < v_{ideal} < V_2 < V_3$. Similarly, it is easy to prove that the energy consumption of voltage scheduling with V_2 and V_3 is smaller than that with V_1, V_2 and V_3, when $0 < V_1 < V_2 < v_{ideal} < V_3$. Consequently, voltage scheduling with three voltage can not minimize energy consumption. The voltage scheduling with more than three voltages also can not minimize energy consumption. □

Theorem 2 *If a processor can use only a small number of discretely variable voltages, the two voltages which minimize the energy consumption under a time constraint T are immediate neighbors to the voltage v_{ideal}.*

Proof. Assume that the processor can supply three voltages V_1, V_2, and V_3. It has already been proved that the energy consumption is a linear function of the time constraint when the processor uses only two voltages as shown in Figure 3.6. If the time constraint T satisfies $t_3 < T < t_2$, voltage scheduling with V_3 and V_2 minimize energy consumption. In case that the time constraint T satisfies $t_2 < T < t_1$, voltage scheduling with V_2 and V_1 minimize energy consumption. Therefore, it is clear that the voltage scheduling with V_3 and V_1 can not minimize energy consumption. Consequently, the scheduling with two voltages which are immediate neighbors to the ideal voltage v_{ideal} minimizes the energy consumption. □

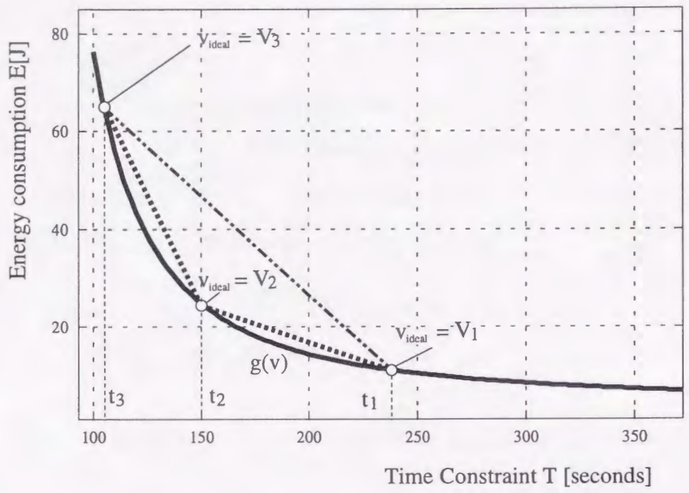


Figure 3.6: Voltage scheduling with two voltages.

Theorem 1 and *Theorem 2* demonstrate significant feature of voltage scheduling. That is, only one time voltage alteration can minimize the energy consumption. Therefore time and power overhead of voltage alteration can be neglected for the practical real-time constraints.

3.3 Generalized Theorems on a More Realistic Model

Up to here, we do not consider loads of tasks, which can be associated with charging and discharging the capacitive load occurred by the tasks. For example, charged and discharged capacitive load for an addition operation is quite different from the capacitive load for a multiply operation. If the capacitive load of each task is taken into account, processing the program with a single voltage which adjusts the execution time to the timing deadline does not always minimize energy consumption. In this section, we introduce a new term *capacitive load* which is formulated by (3.12).

Now, let us define new terms to make explanation more clearly.

- $task_j$ The jth task. $task_j = \{X_j, C_j\}$.
- X_j The number of execution cycles of the jth task. $(1 \leq j \leq N)$.

- C_j Average *capacitive load* for the jth task.
- M The number of gates in the processor.
- $Swit_{ijk}$ The switching count of $gate_k$ while the ith cycle of $task_j$ is executed.

The average *capacitive load* per cycle of $task_j$ can be calculated by (3.12).

$$C_j = \frac{\sum_{i=1}^{X_j} \sum_{k=1}^{M} CL_k \cdot Swit_{kij}}{X_j} \tag{3.12}$$

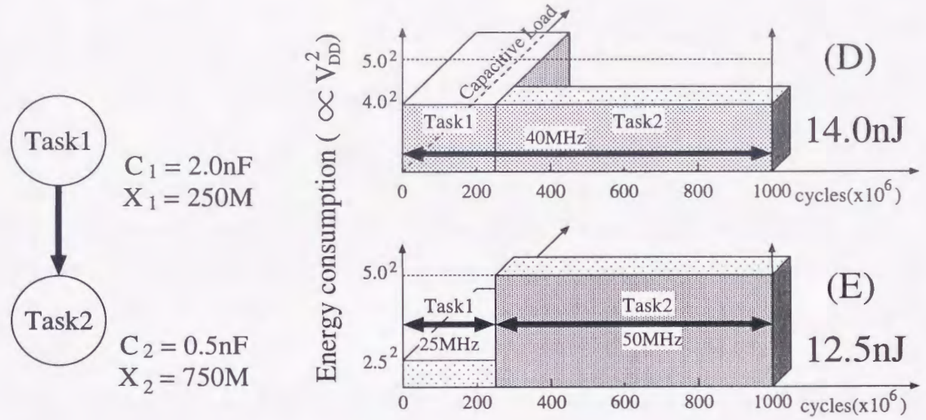


Figure 3.7: Power-delay optimization considering *the capacitive load.*

If the C_js are much different from each other, voltage scheduling with multiple voltages sometimes minimizes the energy consumption for the processing. For a given task set = $\{task_1, task_2\}$ as shown in Figure3.7, the voltage scheduling with 2.5V and 5.0V as shown in Figure3.7(E) reduces much energy compared to that with a single voltage(4.0V) as shown in Figure3.7(D). For the both voltage scheduling, processor completes the 1000M cycles of program just at the timing constraint. The x, y, z axis of both graphs (D) and (E) in Figure 3.7 indicate the execution cycles, the square of supply voltage, and *the capacitive load* respectively. The volume of cubes indicates the energy consumption for processing.

Therefore, minimizing the total volume of cubes leads to a solution of the voltage scheduling problem.

$$Enegy\ for\ scheduling(D) = (2.0nF \cdot 250M \cdot 4.0^2) + (0.5nF \cdot 750M \cdot 4.0^2) = 14.0J$$

$$Enegy\ for\ scheduling(E) = (2.0nF \cdot 250M \cdot 2.5^2) + (0.5nF \cdot 750M \cdot 5.0^2) = 12.5J$$

This example indicates that the voltage scheduling with a single voltage does not always minimize the energy consumption when C_j is much different for each task. For task sets which consists of tasks whose C_js are different from each other, following lemma is useful for voltage scheduling.

Lemma 3 *If a processor uses continuously variable voltages, the voltage scheduling which assigns a single voltage for each task minimizes energy consumption to process a given program under a timing constraint.*

Proof. Assume that more than a single voltage are scheduled to a certain task. From *Lemma 2*, it is obvious that this scheduling can be replaced by the scheduling with a single voltage, remaining the execution time unchanged and reducing the energy consumption for the task. □

If C_js are different from each other, the following theorem is useful for energy reduction.

Theorem 3 *If a processor can use only a small number of discretely variable voltages and C_js are different from each other, the voltage scheduling with at most two voltages for each task minimizes energy consumption to process the program under a timing constraint.*

Proof. Similar to Proof for *Lemma 3*, *Theorem 3* can easily be derived from *Lemma 2*. □

Theorem 4 *Even if a processor can use only a small number of discretely variable voltages and C_js are different from each other, the number of tasks which are scheduled with two voltages is only one or less. Therefore, all tasks except the specific task are scheduled with a single voltage.*

Proof. Assume that two tasks $task_1$ and $task_2$ are given and each task is scheduled with two voltage as shown in Fig.3.8. The $task_1$ and $task_2$ are scheduled with V_n and V_{n+1}, and V_m and V_{m+1}, respectively. Let $x1$, $X_1 - x1$, $x2$, and $X_2 - x2$ denote the execution cycles with V_n, V_{n+1}, V_m, and V_{m+1}, respectively. In this situation, we have (3.13) and (3.14), where T_{finish} and E_{tasks} denotes the finish time and total energy dissipation for a given set of tasks.

$$T_{finish} = \frac{x1 \cdot V_n}{(V_n - V_T)^\alpha} + \frac{(X_1 - x1) \cdot V_{n+1}}{(V_{n+1} - V_T)^\alpha} + \frac{x2 \cdot V_m}{(V_m - V_T)^\alpha} + \frac{(X_2 - x2) \cdot V_{m+1}}{(V_{m+1} - V_T)^\alpha} \tag{3.13}$$

$$E_{tasks} = C_1 \cdot \{x1 \cdot V_n^2 + (X_1 - x1) \cdot V_{n+1}^2\} + C_2 \cdot \{x2 \cdot V_m^2 + (X_2 - x2) \cdot V_{m+1}^2\} \tag{3.14}$$

If $x1$ and $x2$ are varied keeping T_{finish} constant, E_{tasks} behaves as a linear function of $x1$ and $x2$ within the range of $0 \le x1 \le X_1$ and $0 \le x2 \le X_2$, because both (3.13) and (3.14) are linear function of $x1$ and $x2$. Therefore, if $x1$ and $x2$ are varied keeping T_{finish} constant, the $x1$ and $x2$ which reduce or preserve E_{tasks} can surely be found until one of the following conditions is satisfied.

- $x1 = 0$
- $X_1 - x1 = 0$
- $x2 = 0$
- $X_2 - x2 = 0$

Consequently, the number of tasks which are scheduled with two voltages is only one or less. We can also derive this when the number of the tasks is more than two.

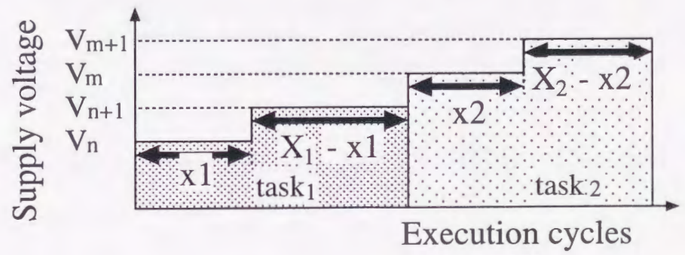


Figure 3.8: Voltage schedule with two voltages.

□

Corollary 1 *For a given set of tasks, if $C_n > C_m$, scheduling a $task_n$ with higher voltage than a voltage assigned to $task_m$ is not optimal for energy reduction.*

Proof. Let X_n and X_m be execution cycles of $task_n$ and $task_m$, respectively. Now, consider a voltage schedule which assign V_i to $task_m$ and $V_{i+1}(> V_i)$ to $task_n$, respectively. This schedule corresponds a schedule (A) shown in Figure3.9. If $X_n < X_m$, we can replace the schedule (A) with a schedule (B), keeping the finishing time of the tasks. The energy consumption for the schedule (B) is clearly smaller than that of the schedule (A). Therefore, if $X_n < X_m$, a voltage schedule which assigns higher voltage to $task_n$ and lower voltage to $task_m$ is not optimal. The same proof can be done when $X_n \geq X_m$.

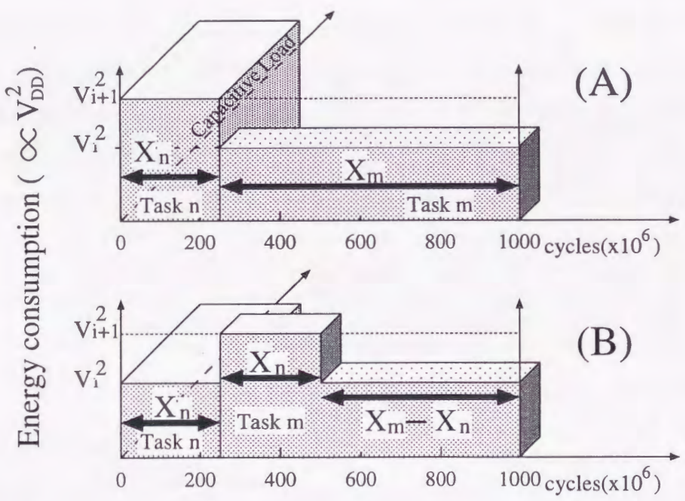


Figure 3.9: Voltage scheduling considering the capacitive loads.

□

3.4 ILP Formulation

If a processor can use only a small number of discretely variable voltages, a single voltage may not minimize the energy consumption. This situation is feasible, because there is some cost involved in supporting several different supply voltages[27, 3]. If the processor can not supply a specified voltage, voltage scheduling with multiple voltages which can be supplied by the processor minimizes the energy consumption. Consequently, the voltage scheduling with a single voltage which adjusts the execution time to the timing deadline does not always minimize the energy consumption. In the ILP formulation of the voltage scheduling problem, we target a processor which equips a small number of discretely variable voltages.

3.4.1 Assumptions

In a simplified voltage scheduling problem, we target systems which assume the following:

- The system is a processor based real-time system.
- The processor can vary its supply voltage dynamically.
- The processor uses only a single supply voltage at a time.
- The processor uses only a small number of discrete voltages.
- The processor equips adaptive clock scheme which closely tracks over the supply voltage.
- The processor can vary the supply voltage at any clock cycle.
- Time overhead to vary the supply voltage and clock frequency is negligible.
- Power loss for the DC-DC level converter is negligible.
- Worst case execution cycles of the given program can be estimated statically.

3.4.2 Notation

The variables used in the formulation are defined as follows.

- N The number of tasks. $N = |\{task_j\}|$.
- $task_j$ The j th task. $task_j = (X_j, C_j)$.

- X_j The number of execution cycles of the j th task. $(1 \leq j \leq N)$.
- C_j Average *capacitive load* for the j th task.
- L The number of variable voltages of target processor. $L = |\{mode_i\}|$.
- $mode_i$ The i th execution mode of the processor. $mode_i = (V(i), F(i))$.
- $V(i)$ The i th voltage. $(1 \leq i \leq L)$.
- $F(i)$ The clock frequency when the supply voltage is $V(i)$. (where $F(i)$ is calculated by the formula(2.3)).
- T The time constraint until which all given tasks must be completed.
- x_{ij} The number of cycles of the task j executed with voltage $V(i)$.

3.4.3 Formulation

The voltage scheduling problem is formulated as follows:

Minimize

$$E = \sum_{j=1}^{N} \sum_{i=1}^{L} C_j \cdot x_{ij} \cdot V(i)^2 \tag{3.15}$$

subject to

$$\sum_{i=1}^{L} x_{ij} = X_j, \quad \sum_{j=1}^{N} \sum_{i=1}^{L} \frac{x_{ij}}{F(i)} \leq T \tag{3.16}$$

The voltage scheduling problem is formally defined as follows. *"For the given $\{task_j\}$ and $\{mode_i\}$, find x_{ij} which minimize E satisfying the time constraint T."*

The both objective function and constraint are function of the linear variable x_{ij}. The computation time to solve the voltage scheduling problem strongly depends on $N \times L$. When all C_js are same from each other, the size of the problem is reduced dramatically. In this situation, an optimal solution can be obtained only to solve the problem of a single task(N=1).

3.5 A Voltage Scheduling Algorithm

The complexity of the voltage scheduling problem depends on the computation time to solve an ILP problem of $N \times L$ variables. An algorithm described in Figure 3.10 makes computation time of the problem to depend on solution time of an ILP problem of N variables.

The voltage scheduling algorithm is shown in figure 3.10. The inputs to the algorithm include a set of tasks $TASK$, a set of execution modes $MODE$, and time constraint T. Each task $task_j \in TASK$ is characterized by its execution cycles X_j, and its average capacitive load C_j. Each execution mode $mode_i \in MODE$ is characterized by operating voltage of the processor $V(i)$, and clock frequency $F(i)$. A $level_j$ ($level_j \in [1 \cdots L]$) represents a level of supply voltage which is assigned to $task_j$. The $V(levelj)$ is smaller than the $V(levelj+1)$. A $assign_i$ is a set of $level_j$. Any $assign_n$ and $assign_m$ $(n \neq m)$ are not the same from each other. The number of the elements of $assign_i$ is the number of tasks N. Firstly, the algorithm calculate a finishing time of the program when processor runs with highest clock frequency. If the finishing time is beyond the time constraint T, the algorithm is finished without any appropriate voltage scheduling. If there are voltage schedules which make processor complete the program before the time constraint T, the algorithm successively assigns lower voltages to each task. *Corollary 1* reduces candidates of voltage assignments to be searched. Voltage assignments where higher voltage is assigned to a task whose *capacitive load* is bigger can be removed from candidates of voltage assignments. Therefore, we have only to calculate the finishing time and the energy consumption only when the voltage schedules satisfy $level_k \leq level_{k+1} \forall\ k \in [1 \ldots N-1]$. Temporarily, the algorithm assigning a single voltage to each task. *Theorem 4* makes search space for the algorithm very small. If finishing time of the program is less than the time constraint, the algorithm selects an only one task which should be scheduled with two kinds of supply voltages. *Theorem 3* demonstrates that the voltage schedule with the two voltages which are neighboring each other for the specified task can minimize energy consumption. If the two conditions $(\frac{X_j}{F(i-1)} - \frac{X_j}{F(i)} + T_{finish} > T)$ and $(E_{temp} - C_j \cdot X_j \cdot \{V(i)^2 - V(i-1)^2\} < E_{min})$ are satisfied for the selected task j, the algorithm performs voltage scheduling for the task j with the two voltages which are neighboring each other. This process is repeatedly performed until all patterns of voltage assignments are done. Finally, the algorithm returns $\{x_{ij}\}$ which minimize the energy consumption satisfying the time constraint T.

Algorithm Voltage Scheduling

$T_{finish} = \sum_{j=1}^{N} \frac{X_j}{F(L)}$ /* $F(L)$ is the biggest clock frequency */

if $(T_{finish} > T)$

exit the algorithm without solutions;

end if

sort tasks according to the magnitude of C_j; /* $C_j \geq C_{j+1}$ */

for all $(i \in [1 \ldots L]$ and $j \in [1 \ldots N])$

$x_{ij} = 0$;

end for

Let $assign_i = \{level_j \mid level_j \in [1 \ldots L] \; \forall \; j, \; j = 1 \ldots N\}$, $E_{min} = \infty$;

Let $A = \{assign_i \mid i = 1 \ldots L^N \;\; and \;\; assign_n \neq assign_m \forall n, m \; (n, m \in [1 \ldots L^N], \; n \neq m)\}$

repeat

$assign = \{level_j \mid j = 1 \ldots N\} \in A$;

if $(level_k \leq level_{k+1} \forall \; k \in [1 \ldots N-1])$;

$T_{finish} = \sum_{j=1}^{N} \frac{X_j}{F(level_j)}$; $E_{temp} = \sum_{j=1}^{N} C_j \cdot X_j \cdot V(level_j)^2$;

if $(T_{finish} = T \;\; \& \;\; E_{temp} < E_{min})$

$E_{min} = E_{temp}$; $x_{ij} = X_j$;

else if $(T_{finish} < T)$

$x = 0$;

for each $j \in \{j \mid 2 \leq level_j \leq L\}$

$i = level_j$;

if $(\frac{X_j}{F(i-1)} - \frac{X_j}{F(i)} + T_{finish} > T \;\; \& \;\; E_{temp} - C_j \cdot X_j \cdot \{V(i)^2 - V(i-1)^2\} < E_{min})$;

find x which satisfy $\frac{x}{F(i-1)} - \frac{x}{F(i)} + T_{finish} = T$

$E_{min} = E_{temp} - C_j \cdot x \cdot \{V(i)^2 - V(i-1)^2\}$;

$x_{i-1j} = x$; $x_{ij} = X_j - x$;

end if

end for

end if

end if

$A = A - assign$;

until ($A = \phi$)

Output $\{x_{ij}\}$;

end Algorithm

Figure 3.10: Voltage scheduling algorithm.

3.6 Experimental Results

At first, we show the results for a set of tasks : $TaskSet = \{task_1, task_2, task_3\}$. The average *capacitive loads* and the number of execution cycles of these three tasks are $\{C_1 \; , \; X_1\} = \{50pF \; , \; 50 \times 10^9\}$, $\{C_2 \; , \; X_2\} = \{100pF \; , \; 50 \times 10^9\}$, and $\{C_3 \; , \; X_3\} = \{150pF \; , \; 50 \times 10^9\}$, respectively. The purpose of this experiment is making clear the effects of the variety of the variable voltages to the energy reduction. In this example, we target seven kinds of variable voltage processors as shown in 3.1. The processors are assumed to be able to vary the supply voltage dynamically but can use single voltage at a time. The results are obtained by the algorithm described in the previous section.

Table 3.1: Variation of variable supply voltages.

cases	Variable supply voltages
Processor-1	Only 3.3V
Processor-2	3.3V and 0.9V
Processor-3	3.3V and 1.7V
Processor-4	3.3V and 2.5V
Processor-5	3.3V, 2.5V and 0.9V
Processor-6	3.3V, 2.5V, 1.7V, and 0.9V
Processor-7	Any voltage between 3.3V and 0.9V

The results are shown in Figure 3.11. The energy consumption of the Processor-1 is constant, even if the time constraint is relaxed, because total amount of the task is constant. It can be said that the much more energy consumption can be reduced, the more the kinds of variable voltages are. If the time constraint is relaxed to $\times 10$, the energy consumption can be reduced to 1/10 with an ideal processor (Processor-7). Using only two voltages, we can reduce the energy consumption down to less than one third, if the time constraint is relaxed to $\times 10$. Selecting suitable voltages for the time constraint leads to drastic energy reduction even if the number of variable voltages is small. Therefore, determining the kinds of variable voltages which are supplied by the processor is the most important for the optimization of the variable voltage scheduling.

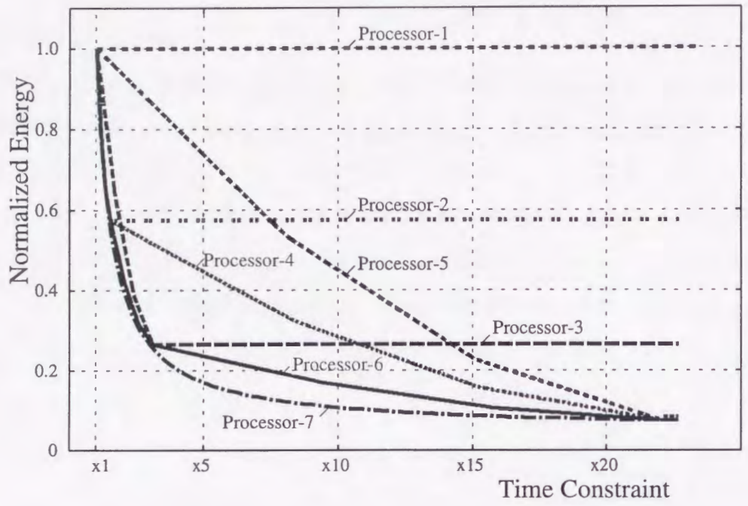


Figure 3.11: Variation of voltages versus energy reduction.

Next, we show the experimental result for four task sets : TaskSet $= \{task_1\ , task_2\ , task_3\}$) as shown in the table 3.2. The numbers in the left side parentheses represent the number of execution cycles of task1, task2, and task3 respectively, and right side parentheses represent the average *capacitive load* of task1, task2, and task3 respectively. Used processor is *processor-2* appeared in Table3.1. The number of execution cycles of tasks are the same from each other in this experiment. Sum of C_js of each task set is the same from each other. The assumption for clock frequency properly obeys (2.3). The V_T in the (2.3) is assumed to be 0.6[V] in this experiment. The experimental results are shown in Figure3.12.

Table 3.2: Specification of sample programs.

Task Sets	(X_1, X_2, X_3)	(C_1, C_2, C_3) [pF]
TaskSet-1	(50, 50, 50)$\times 10^9$	(100, 100, 100)
TaskSet-2	(50, 50, 50)$\times 10^9$	(80, 100, 120)
TaskSet-3	(50, 50, 50)$\times 10^9$	(40, 100, 160)
TaskSet-4	(50, 50, 50)$\times 10^9$	(20, 40, 240)

The results demonstrate that the voltage scheduling with multiple voltages becomes more effective for energy reduction, if the difference among C_js becomes bigger, even if the sum of C_js is a constant. Comparing TaskSet-1 with TaskSet-4, the energy consumption is reduced by 30% even if the time constraint is the same from each other at x6. If the C_js of given tasks are much different from each other, scheduling the lower voltage to the tasks whose C_js are smaller, and higher voltage to the tasks whose C_js are bigger produces drastic energy reduction. Of course, energy consumptions are decreased according to the time constraint is relaxed.

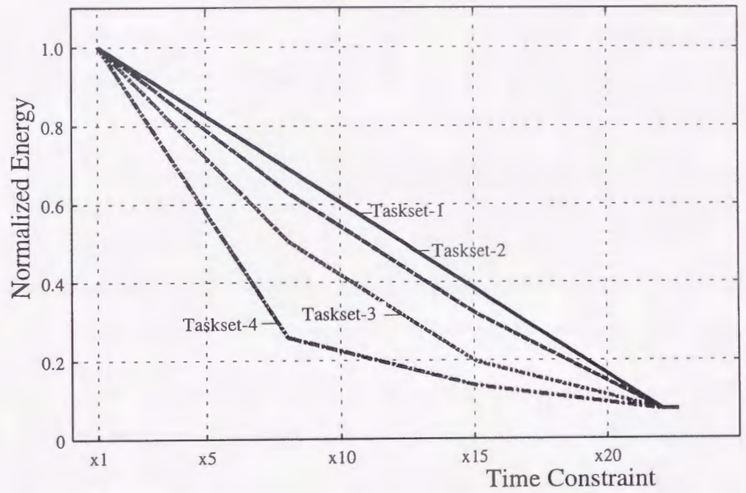


Figure 3.12: Variation of C_j versus energy reduction.

From the results, we have the following observations:

1. Increase of the number of variable voltages may have strong impact on energy reduction. However, if the number of the variable voltages is more than four, the behavior of the energy consumption is very similar to that of the ideal processor.

2. Selecting suitable voltages for the time constraint leads to significant energy reduction even if the number of variable voltages is very small.

3. Even if the time constraint is constant, assigning lower voltage to the tasks whose C_js are bigger and higher voltage to the tasks whose C_js are smaller reduces total energy consumption by 30% at maximum case. For the application program whose capacitive load is widely varied by operating condition, voltage scheduling is much effective for the energy reduction.

3.7 Summary

In this chapter, we present theorems, the integer linear programming (ILP) model, and the algorithm for the voltage scheduling problem. The targeted processor is the dynamically variable voltage processor which can vary the supply voltage dynamically, but can use only a single voltage at a time. Important theorems presented in this chapter are summarized below.

- Power-delay trade-off in CMOS circuits are substantiated for any practical V_T and α (*Lemma 1*).
- If a processor can use continuously variable voltage, only a single voltage can minimize energy consumption satisfying the timing constraint (*Lemma 2*).
- If a processor can use only a small number of discrete voltages, the voltage scheduling with at most two voltages finds the optimal solution, and both of the voltages are immediate neighbors to the voltage v_{ideal} which minimizes the energy consumption when continuously variable voltages are supplied. (*Theorem 1*)

Theorem 1 demonstrates important feature of the voltage scheduling. That is, only one time voltage alteration can minimize the energy consumption. Therefore time and power loss for voltage alteration can be neglected if the real-time system gives a long period of timing deadlines compared to the time overhead to vary the supply voltage. The theorem can be extended for a set of tasks where the C_js are much different from each other. From the experimental results with tree tasks whose C_js differ from each other, we have the following observations:

- Increase of the number of variable voltages may have strong impact on energy reduction. However, if the number of the variable voltages is more than four, the energy reductions by the variable voltage processor is very similar to the results for the ideal processor.
- If a processor employs suitable voltages, significant energy reduction can be achieved even if the number of variable voltages is very small.
- Even if the time constraint is constant, assigning lower voltage to the tasks whose C_js are bigger, and higher voltage to the tasks whose C_js are smaller reduces total energy consumption by 30% at maximum case. For the application program whose capacitive load is widely varied by operating condition, voltage scheduling is much effective for the energy reduction.

Our future work will be devoted to extend the proposed optimization problem considering actual design of DC-DC level converter and real-time task scheduling.

Chapter 4

Programmable Power Management Architecture

4.1 Background

With the recent popularity in portable, battery-powered devices such as digital cellular telephones and personal digital assistants (PDAs), minimizing power consumption of CMOS VLSI circuits becomes a more important issue.

Conventionally, many power optimization techniques at various levels of abstractions are proposed, and power consumption of processors optimized for specific application programs can be dramatically reduced in contrast to general purpose processors. At the architectural design level, customizing the supply voltage and datapath width for each application program has a strong impact on power reduction[4].

However, as application programs which run on a portable system diversify, so too must performance requirements and precision requirements for each application or each operation. Therefore, the optimized processor for a specific application will dissipate large amounts of power for other application programs. Now, let us assume that the application programs *ap1* and *ap2* are run successively on a single processor system, and *ap1* requires a much higher performance than that required by *ap2*. If the processor is optimized for *ap2*, the performance requirement of *ap1* may not be satisfied. On the other hand, optimizing the processor for *ap1*, a great speed potential of the processor is unused for *ap2*, and unused power is dissipated when *ap2* is processed. Similarly, unused power may be dissipated in wide

datapath circuits when the short length data is processed, because many redundant switchings occur in extra bits of datapath. Therefore, we need an mechanism to save power consumption by inactivating extra bits which are not essential to the results of computation. Contribution of this chapter is to propose a dynamically reconfigurable processor architecture which can realize high performance or high precision computation with high power consumption, or low power consumption with low performance or low precision computation according to the applications.

In this chapter we propose the *Power-Pro* architecture (Programmable Power Management Architecture), a novel processor architecture for power reduction. The *Power-Pro* architecture has the following two key functions.

- *PVC*: Programmable V_{DD} Control.
- *PADWC*: Programmable Active Datapath Width Control.

The *PVC* scheme makes the software possible to dynamically manipulate the V_{DD} (supply voltage) and the clock frequency. For application programs which do not need high performance, this scheme can drastically reduce power consumption by lowering both V_{DD} and the clock frequency.

For data or operations whose precisions are shorter than the datapath width of the processor, the *PADWC* scheme can reduce unused energy consumption by adjusting the active datapath width to the precision of each operation. We have designed and fabricated a single pipeline processor which employs two functions of the *Power-Pro* architecture to evaluate the effectiveness of the *Power-Pro* architecture.

The rest of this chapter is organized as follows: Section 4.2 presents our basic idea for power reduction. In section 4.3, we propose the *Power-Pro* architecture. Applications and Experimental results are presented in section 4.4 and 4.5, respectively. Section 4.6 reports characteristics and simulation results of the pilot chip. Section 4.7 concludes this chapter.

4.2 Reduction of Wasteful Power Consumption

Our power saving approaches mainly aim to cut down the wasteful power dissipation caused by the following situations.

- The performance of the processor is sometimes much higher than the performance requirements of the applications.
- The datapath width of the processor is sometimes much wider than the precisions required by operations and data.

4.2.1 Power-Delay Optimization

As is described in subsection 3.1.3, we can dramatically reduce the energy consumption for tasks by the supply voltage scheduling. One of the most important technology for the voltage scheduling is a voltage control scheme. Since software knows the behavior of a whole system including hardware and software better than hardware generally, controlling the supply voltage from software has much advantage for power reduction. Therefore, a scheme which enables software programmers to easily determine an optimal supply voltage for the tasks without any real-time violation is required. In real-time system, each task which is a fragment of a program characterized by its arrival time, a worst case execution time and a deadline time. Utilizing these informations, software can dynamically select the optimal operating supply voltage. The *PVC* scheme enables software to control the supply voltage, and suits for practical systems.

4.2.2 Reduction of Wasteful Power in Datapath Circuits

Since requirements of application programs become diverse, activated hardware resources also differ from each application program or each operation. For a certain application program, there are many unused functional modules, but power consumption of unused functional modules can not be neglected. This causes significant increase of wasteful power dissipation.

The idea of clock-gating has been well known as a technique to reduce the power dissipated in unused parts[11]. One difficulty for the conventional clock-gating scheme is to reduce the power consumption in datapath circuits, since datapath circuits(including ALU, datapath register, data-bus, off-chip driver and data memory) of highly pipelined processor are always activated. However the length of data on datapath is quite different from each operation and many redundant switchings occur for short length data. Therefore, more sophisticated techniques to power consumption in datapath circuits are required. The basic idea of *PADWC* is to inactivate the extra bits which are not essential to the results of computation.

Now, let us consider the three data A, B, and C are processed through a 16 bits pipeline register successively. The precisions of A, B, and C are assumed to be 16-bits, 4-bits, and 12-bits respectively. Figure 4.1 and 4.2 shows the state of the pipeline register and switched bits for each clock cycle. In Figure 4.1, the total number of switching bits is 23, because wasteful bits of datapath are switched. The wasteful bits stand for the bits which are not essential for the results of the computation. Stopping the clock supply to wasteful bits of the datapath register produces drastic power reduction, because no data transition is occurred in the datapath register where clock supply is disabled.

	Decimal value	Value on pipeline register	Switching bits
A.	-12349	1100 1111 1100 0011 (16bit)	11
B.	4	0000 0000 0000 0100 (4bit)	12
C.	-209	1111 1111 0010 1111 (12bit)	Total : 23

Figure 4.1: Switching bits in fixed datapath with architecture.

Figure 4.2 shows behavior of a variable datapath width architecture. In this example, total switched bits are reduced from 23 to 9 by gating the clock supply of wasteful bits in the datapath register. Power consumption in datapath circuits can dramatically be reduced, using only three kinds(16, 12, and 4 bits) of datapath width. As is shown in this example, using the small number of datapath width may be feasible, because hardware cost and power dissipation for additional circuits can not be negligible.

	Decimal value	Value on pipeline register	Switching bits
A.	-12349	1100 1111 1100 0011 (16bit)	3
B.	4	1100 1111 1100 0100 (4bit)	6
C.	-209	1100 1111 0010 1111 (12bit)	Total : 9

Figure 4.2: Switching bits in variable datapath with architecture.

4.3 The *Power-Pro* Architecture

We present two functions, *PVC* and *PADWC*, of the *Power-Pro* architecture in this section.

4.3.1 PVC: Programmable V_{DD} Control

Basic concept of the *PVC* scheme is enabling the software to control the supply voltage and clock frequency. Optimal supply voltage and the optimal timing to vary the supply voltage are decided in the following manner. (i) At first, the worst case execution cycles of the program is estimated statically at the compiling phase[64]. A static technique to estimate the worst case execution time has been studied in [14]. (ii) Next, the optimal supply voltage and the optimal timing to vary the supply voltage are decided. The optimization problem which finds optimal voltage schedule for a given program under a timing constraint has already formulated as an ILP (Integer Linear Programming) problem in Chapter 3. The optimal voltage and alteration timing are also decided at compiling phase. (iii) At last, the compiler insert special instructions to vary the supply voltage into object code, or the timer interrupt control routine vary the supply voltage at prescribed timing.

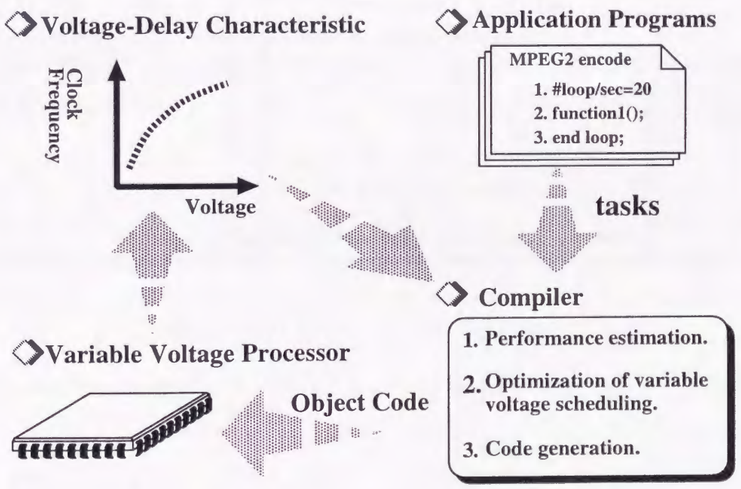


Figure 4.3: Variable voltage optimization.

Compiler techniques to decide the optimal supply voltage which minimize energy consumption under a timing constraint is very important technology for *PVC* scheme. Next, we present a optimization of the *variable voltage scheduling* with the compiler. Two kinds of informations are required for the optimizations of the *variable voltage scheduling*. One of the required information is a voltage-delay characteristic of the dynamically variable voltage processor. The other information is a program source in which real-time constraints are explicitly specified. Firstly, the compiler estimate the worst case execution cycles of the application program. Secondly, the compiler finds an optimal voltage schedule, using the informations of the estimated execution cycles of the program and the voltage-delay characteristic of the processor as shown in Figure4.3. The optimal voltage scheduling minimize energy consumption without any real-time violation. Finally, an object code including *voltage control instructions* is generated for the variable voltage processor.

A detailed optimization techniques have already been presented in chapter 3. In this chapter, we describe the *PVC* scheme concentrated on an architectural feature for power-delay optimization.

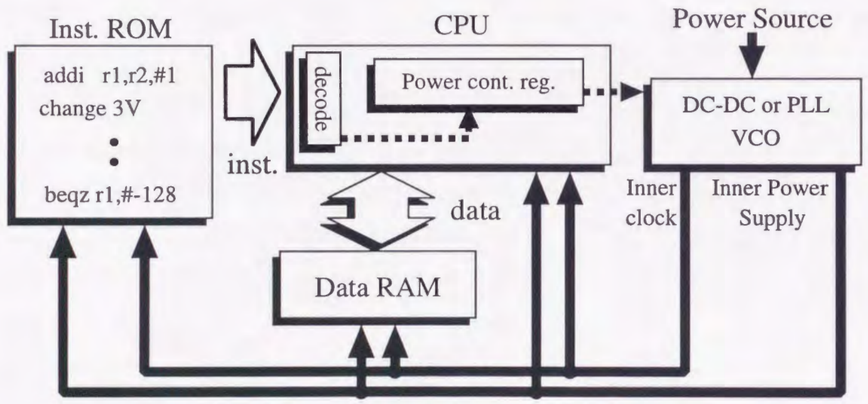


Figure 4.4: Programmable V_{DD} control.

Details of the *PVC* scheme are described below.

- The architecture has a register named *power control register* which saves a level of V_{DD} as shown in figure 4.4.

- The architecture employs special instructions by which programmers can set an intended value to the *power control register*.

- The architecture has a power converter which can control supply voltage of the processor according to the value of the *power control register*.

- The architecture employs the variable clock frequency which closely synchronizes the supply voltage.

The most unique point of the *Power-Pro* architecture is that V_{DD} and clock frequency are varied by the special instructions named *power control instructions*.

Detailed design of the power converter such as DC-DC converter or PLL is beyond the scope of this chapter, and any architecture of power converter does not lose originality of the *Power-Pro* architecture. Current research in low power portable electronics includes designs of low voltage on-chip DC-DC converters, and efficiencies above 90% have been reported[18, 44, 41]. However, for very low power implementation, the following features must be necessary.

1. A energy consumption for voltage conversion is much smaller than current designs.

2. Overhead time to vary the V_{DD} and clock frequency is negligible compared with running time of the application programs.

4.3.2 PADWC: Programmable Active Datapath Width Control

Basic concept of the *PADWC* scheme is to vary the active datapath width by disabling clock supply to extra bits of datapath registers. One of important technologies which makes the *PADWC* scheme more effective is a software technique which supports programmers to specify the bit width for each variable in programs[23, 7]. However, software techniques to realize such mechanism is beyond the scope of this chapter. This chapter concentrates on architectural methods to adjust the active datapath width for each variable whose bit width is specified by the programmers.

4.3.3 Architecture for *PADWC* scheme

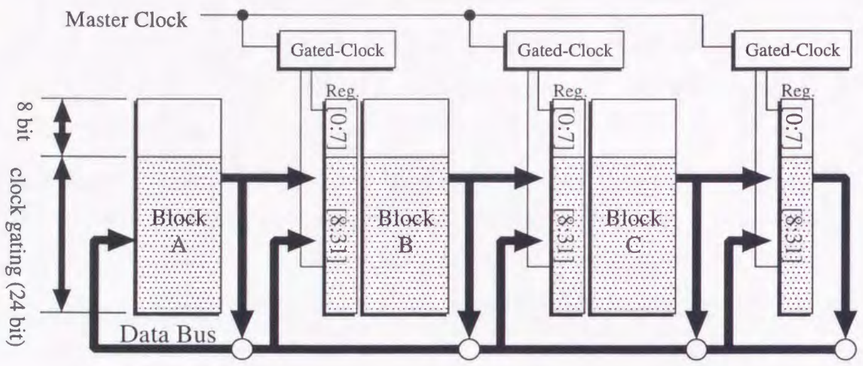


Figure 4.5: Architecture for *PADWC* scheme.

For the application programs which treat short length data very frequently, the *PADWC* scheme works effectively for power reduction. For example in Fig. 4.5, if an application program which has only 8-bits width variables are processed on 32-bit microprocessor, disabling the upper 24 bits of datapath reduces power consumed by clock ports of flip-flops. In addition, when 8-bits width data and 32-bits width data are operated alternately by the 32-bit microprocessor, power consumed by datapath circuits may be reduced by disabling clock supply to unnecessary bits of datapath register for each operation.

Detailed functions of the *PADWC* scheme are described below.

- Each instruction can specify not only operation but also active datapath width.
- Clock supply to certain bits of datapath register can be stopped by gated clocks according to the instructions.

4.3.4 Special Instruction for *PADWC* scheme

All instructions of the *Power-Pro* architecture has tag field to specify the active datapath width. The bit width of the tag field which specifies the active datapath width is $\lceil log_2 V \rceil$ (V represents the number of variation of variable datapath width).

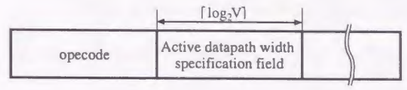


Figure 4.6: The instruction to specify an active datapath width.

We do not define an instruction set architecture for the *Power-Pro* architecture, except the functions which mentioned above. Any operation set is applicable to the *Power-Pro* architecture. However, the limited number of variations of the datapath width may be feasible, because hardware cost and power expense of additional circuits for the *PADWC* scheme may be significant, if any bits of the datapath can be disabled.

4.4 Applications

4.4.1 Applications for *PVC*

The best application for the *PVC* scheme may be a hard real-time systems. In hard real-time systems, programmer and hardware itself must guarantee that the worst case execution time satisfies the timing deadlines. Many real-time operating systems rely on this data for process scheduling. In processor design, the worst case execution time strongly depends on execution cycles and clock period. Therefore, for given execution cycles, the power consumption is minimized lowering the supply voltage until the total execution time meets to the specific timing deadline.

Now we show an example of voltage controlling. To make an example more simple, we assume a target system as follows.

- Core processor can vary its supply voltage at any clock cycles.
- Variation of variable supply voltages are 5.0V and 1.8V.
- Clock frequencies of the processor with 5.0V and 1.8V are 100MHz and 25MHz, respectively. This data is based on (2.3). The V_T in (2.3) is assumed to be a 0.7V in this example.
- Execution cycles of application program is 26×10^9 cycles.

- Timing deadline is 50 seconds

The limited number of variations of the variable voltages may be feasible, because preparing any kinds of stable supply voltages and clock frequency wastes the significant power and hardware cost. However, in an ideal system model which can prepare any kinds of supply voltage without any power loss, the energy consumption is minimized when the processor uses a single supply voltage just to meet the given timing deadline.

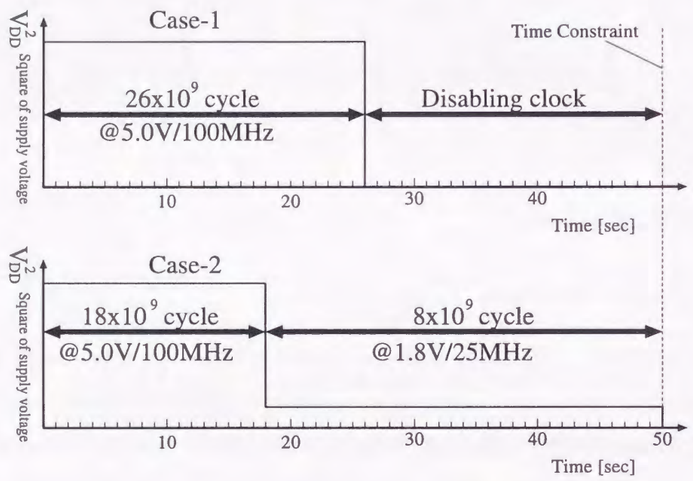


Figure 4.7: Example for dynamic voltage controlling.

Two ways of voltage controlling can be considered as shown in Fig. 4.7. Horizontal axis and vertical axis of the graph indicate time and square of supply voltage, respectively. One way is to execute always with 5.0V and then stopping the power supply after the program is executed. Other way is to execute with 1.8V as long as possible to satisfy the timing constraint. Of course, scaling down the supply voltage causes increase of the execution time even if the execution cycles are constant. Therefore, the *PVC* scheme achieves maximum power savings processing with lower voltage as long as possible adjust finishing time to the deadline time, as shown in the Processor-2 of Fig. 4.7. For example in Fig. 4.7, energy consumption in Processor-2 is reduced by 20% compared with Processor-1. The most important knowledge

from this example is that processing a program lazily is better than processing snappy for power reduction.

4.4.2 Applications for *PADWC*

The *PADWC* scheme is effective when short length data are processed on a microprocessor which equips very wide datapath circuits. Such as PDAs (Personal Digital Assistances) which have an audio processing functions, image compression functions, an electrical dictionary, a simple calculator, a telephone directory and so on, treat many kinds of precisions. For example, audio data requires much higher precision than that of *character type* data. In addition, real-time image compression requires high performance computation, and therefore, highly parallelized datapath circuits are required for real-time processing. For such a applications, processors usually employ large bit width of datapath circuits. Therefore, much power is dissipated when the *character type* data are processed on such large width datapath. When only *character type* data are operated, power reduction can be performed by disabling the clock supply to extra bits of datapath registers. However, *PADWC* scheme becomes ineffective for operations which need large precisions and use full bits of datapath width, because there are no wasteful bits in datapath circuits when such operations are executed.

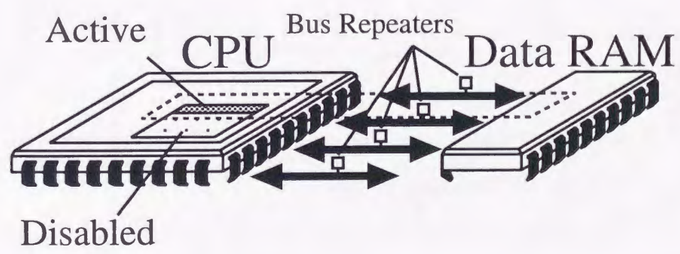


Figure 4.8: Power reduction for off-chip driving.

The *PADWC* scheme has secondly effect on reducing power for off-chip driving. For example in Fig. 4.8, which holds values of off-chip data-bus with bus repeater, not only switching

in processor core but also the power of off-chip data-bus can be cut off. Since power dissipation in off-chip driver, off-chip data-bus and bus repeater(sense amplifier) is vary dominating, cutting off wasteful switching in datapath leads to drastic power reduction.

4.5 Experimental Results

4.5.1 Experimental Results for *PVC*

In this section we evaluate *PVC* scheme with 416x386 pixels MPEG2 encoding program. This sort of real-time applications must process given data until a certain timing deadline. So, if the processed data per unit time are decreased, processor can slow down the computing speed. In this experiment, we would like to make sure how much power can be reduced by *PVC* scheme when the operations per unit time are reduced.

At first, we measured the number of executed instructions for the MPEG2 encoding. The numbers of total execution cycles are estimated by (4.1). Where, N, e_i and n_i represent the number of basic blocks in the program, the execution counts of basic block i, and the number of instructions in basic block i, respectively.

$$Total\ \ execution\ \ cycles \ = \ \sum_{i=1}^{N}(e_i \times n_i) \tag{4.1}$$

Estimated execution cycles per second for each frame rate is shown in Table 4.1. The frame rate stand for the number of frames processed per second. The optimal supply voltage and optimal timing to vary the supply voltage can be calculated by the data in Table 4.1 and (2.3). Since execution cycles are reduced in proportional to the reduction of frame rate, energy consumption can also be reduced in proportional to the decrease of frame rate at the worst estimate.

Table 4.1: Execution cycles for decreased frame rate.

Frame Rate	25	19	13	9	5
Exec. Cycles($\times 10^9$)	19.79	13.63	10.83	7.45	3.89

Four kinds of target variable voltage processors are shown in Table 4.2 to make clear the relation between the number of the variable supply voltages and power reduction of the processor. In Fig. 4.9, we show experimental results with MPEG2 encoding for each variable voltage processor in Table 4.2.

Table 4.2: Variable supply voltages.

Processors	Variable supply voltages
Processor-1	Only 3.3[V]
Processor-2	3.3[V] and 1.0[V]
Processor-3	3.3[V], 2.2[V] and 1.0[V]
Processor-4	Any level between 3.3[V] and 1.0[V]

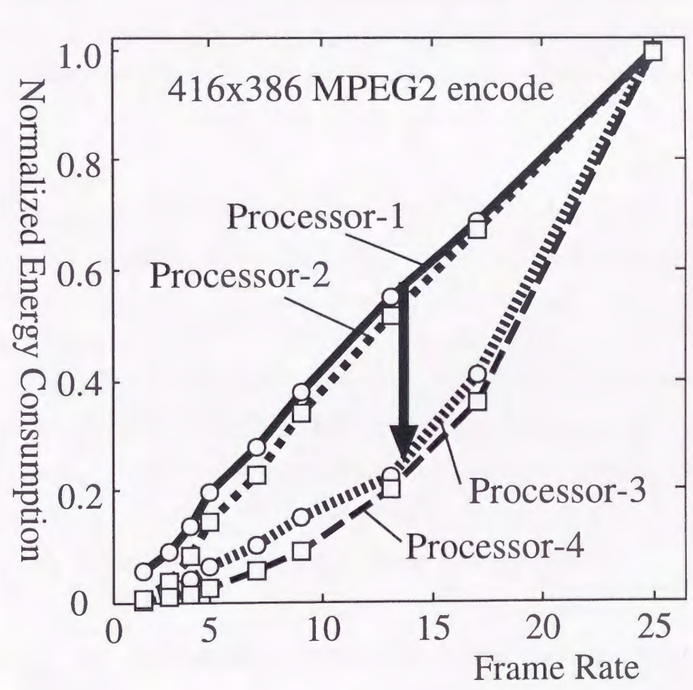


Figure 4.9: Experimental results with MPEG2 encoding.

From the experimental results in Fig. 4.9 demonstrate the following.

1. If the processor can use only a single supply voltage(3.3[V]), the energy consumption is reduced in proportion to reduction of frame rate (Processor-1).

2. If the processor can use three kinds of supply voltages (Processor-3), energy consumption is reduced by 50% compared with the Processor-1.

3. If the processor can use any kinds of supply voltages between 3.3[V] and 1.0[V], energy consumption is minimized at any frame rates.

These results show that energy consumption can be reduced dramatically by controlling the supply voltage sophisticatedly, even if the variable supply voltages are few kinds.

4.5.2 Experimental Results for *PADWC*

We use benchmark programs shown in Table 4.3 to evaluate *PADWC* scheme. We use only byte data for bubble sorting program in this experiment. Accordingly, all programs in Table 4.3 treat a byte data vary frequently, because these programs process mainly for a *character type* data.

Table 4.3: Set of benchmark programs.

Program	Description
WC	Word counter from UNIX application.
Split	File divider from UNIX application.
Sort	Bubble sorting program from UNIX application.

For each program, we compare the power consumption of the following two types of processors.

PADWC A RISC processor which equips functions for *PADWC* scheme. Variable datapath width are 32-bits and 8-bits.

Fixed A 32-bits RISC processor whose active datapath width is fixed.

Table 4.4: Expense for *PADWC* scheme.

Processor	Number of Cells	Critical Path Delay
PADWC	3,478 (42,524 tr.)	28.46ns
Fixed	3,199 (39,496 tr.)	27.47ns

Table 4.5: Power reduction by *PADWC*.

Program	**PADWC**	**Fixed**	Power Reduction Rate
WC	52.79mW	63.04mW	16.27 %
Split	60.57mW	72.39mW	16.32 %
Sort	69.18mW	96.05mW	27.98 %

$$Power\ reduction\ rate = \frac{Power\ of\ \textbf{Fixed} - Power\ of\ \textbf{PADWC}}{Power\ of\ \textbf{Fixed}}$$

Power consumption is estimated by post layout simulation. In the power estimation, power consumption of logic circuits, clock system and off-chip driving are considered. A 25MHz clock frequency is assumed for each power estimation.

Experimental résults in Table 4.4 shows that area and critical path delay of **PADWC** processor is increased by 7.6% and 3.6%, respectively compared with **Fixed** processor. Since the clock controller for *PADWC* scheme does not include critical path, the expense in delay time for **PADWC** processor is very small. Increase of hardware cost can also be negligible.

Experimental results in Table 4.5 show that power consumption of **PADWC** processor is reduced by 28% compared with **Fixed** processor at the maximum case.

4.6 Simulation Results of Pilot Chip

We designed and fabricated a **Power-Pro**^{mf} which equips minimal functions of the *Power-Pro* architecture[51].

Table 4.6: Specification of *Power-Pro*^{mf}.

Process	0.5 μ m CMOS double metal
Chip size	$4.76m \times 4.76m = 22.66mm^2$
The number of cell	2907 (35,282 tr.)
Signal pin	76 pin
Clock frequency	25 MHz

We joined VDEC (VLSI Design and Education Center) pilot chip project so as to implement an actual chip. The **Power-Pro**^{mf} is a 32-bit RISC microprocessor with five pipeline stages, and its basic architecture is presented in a well-known textbook by J.L.Hennessy and D. A. Patterson[25]. Of course, the **Power-Pro**^{mf} equips functions to vary the V_{DD}, clock frequency and active datapath width of its own dynamically by the special instructions.

We used VHDL for logic design, SYNOPSYS tools for logic synthesis and simulation, and automatic P&R tools of Avant! co.,ltd.. Used process is 0.5 μm CMOS double metal standard cell array technology provided by NEL (NTT Electronics Technology co.,ltd.). We took about two weeks to complete the design.

Since the gated clock scheme is adopted to vary active datapath width and to invalidate a clock activity of inactive modules, we use 65 logic gates for clock control. Chip specifications are shown in Table 4.6.

We verified designed chip and estimated power consumption by the post-layout simulation. Estimated power consumption is shown in Table 4.7.

Table 4.7: Power consumption of *Power-Pro*^{mf}.

Voltage/Clock	Datapath Width	Power Consumption
3.3V/25MHz	32 bit mode	106.86mW
	8 bit mode	76.46mW
2.0V/15MHz	32 bit mode	23.55mW
	8 bit mode	16.85mW

32-bit mode All instructions are executed in 32-bits datapath width.

8-bit mode Operations whose precisions are shorter than 8-bits are executed in 8-bits datapath width.

Power consumption in the 8-bit mode is less than that in the 32-bit mode by 29%. Since power consumption of the datapath circuits account for only 50% of the total power consumption, the power of the 8-bit mode can not be quarter of the 32-bit mode. When clock frequency is slowed down to 15MHz, the processor can run correctly in 2.0V, and power consumption is less than quarter of the power consumption in 3.3V.

4.7 Summary

In this chapter, we propose a novel processor architecture and its key functions, the *PVC* scheme and the *PADWC* scheme for power reduction.

Experimental results with MPEG2 encoding shows that the *PVC* scheme can halve the power consumption compared with that of fixed voltage processor. The most important knowledge from this result is that processing a program lazily is better than processing snappy for power reduction. This concept can also be applied to power optimization techniques of process scheduling[50].

For the application programs which treat a lot of byte data, the *PADWC* scheme works effectively for power reduction. Experimental result with bubble sorting program shows that the power consumption of the processor which adopts the *PADWC* scheme is reduced by 28% compared with that of the fixed datapath width processor at maximum case.

Since, these two low power techniques are independent from each other, these techniques are applicable in the same system. Therefore, the total effect of power reduction is sum of each earnings, and power consumption can be one third of conventional microprocessors.

In future, we would like to apply the system level power management techniques to the *Power-Pro* architecture .

Chapter 5

Memory Power Optimization with Code Merging

5.1 Background

An important class of digital systems includes applications, such as video image processing and speech recognition, which are extremely memory-intensive. In such systems, a much power is consumed by memory accesses. In most of today's microprocessors, an instruction memory including a cache memory is one of the main power consumer. The on-chip caches of the 21164 DEC Alpha chip dissipate 25% of the total power of the processor. The StrongARM SA-110 processor from DEC, which specifically targets low power applications, dissipate about 27% of the power in the instruction cache[40]. Thus, utilizing low-power memory organizations can greatly reduce the overall power consumption in the system. In addition, it is a noteworthy that the power consumed by memory accesses certainly increases as memory size is increased[9, 65, 31]. Therefore, low power techniques which reduce power consumed by memory accesses become more important for prospective memory-intensive applications.

Especially for embedded systems, these low power oriented applications also require design flexibility, which results in the need for implementation on programmable hardware platform. Current semiconductor technology allows the integration of processor cores and memory modules on a single die, which enables the implementation of a system on a single chip. Consequently, the design productivity along with the traditional synthesis process has not followed the exponential growth of both applications and implementation technologies. The

shrinked time-to-market has made this situation worse. There is a wide consensus that only a reuse of highly optimized cores can match the demands of the pending applications and the potential ultra large scale integration. Therefore, a low power core-based system-on-chip, consisting of easily reconfigurable cores attracts much interest of all VLSI system designers.

In this chapter, we propose an application specific power optimization technique with object code merging. The decompressor is used to restore the merged object codes and implemented by a small ROM which is placed between main program memory and CPU core. Frequently executed sequence of object codes are merged into a single instruction. Our memory optimization technique targets a typical application specific system-on-chip, consisting of a simple processor core and reconfigurable two instruction memories. A major premise of the target system is that the programs are stored in embedded ROM because the programs need not to be modified after system design is completed. We target the system-on-chip which assumes that (i) instruction memories are organized by two on-chip memories, a main program memory and a instruction decompressor, (ii) these two memories can be independently powered-up or powered-down by a special instruction of a core processor, and (iii) a compiler optimizes size of the two memories and determines which sequence of object codes should be merged into a single instructions so as to minimize average of read energy consumption.

The rest of the chapter is organized in the following way. In section 5.2, we prove that the lower bound of read energy consumption in memory is square root of memory size. Based on this fact, we discuss the motivations for our work and present our concept to optimize the instruction memory organization. A novel power optimization technique with object code merging is proposed in section 5.3. Section 5.4 presents experimental results and discussion on the effectiveness of the approach. This chapter is concluded in Section 5.5.

5.2 Motivations and Our Approach

Our memory optimization technique is based on the following two notable facts.

- Read energy consumption in memory is certainly increased as the size of the memory is increased.
- Most of parts of the program is rarely executed.

5.2.1 Area-Power Correlation

If memory circuits satisfy following conditions, we can derive Corollary1.

- Memory cells whose size are the same from each other are located into a two-dimensional matrix.
- Capacitive loads of bit lines are large enough to take into account and are in proportion to its length.
- Only horizontal or vertical bit lines are available.
- Every stored data is read out from a single point on boundary of memory.
- Every memory cells are accessed in the same probability.

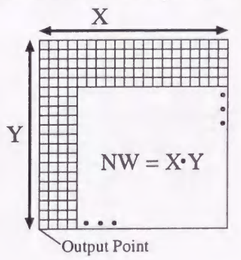
(Corollary1)

A lower bound of an average of read energy consumption is $\Omega(NW^{1/2})$, where NW represents the number of words of a memory circuit.

(Proof)

To make explanation more simply, we target a memory system which assumes 1 bit for bit width and NW words for the number of words. These assumptions do not lose generality.

Let X, Y, λ_x, and λ_y be the number of columns, the number of rows, width of a memory cell, and height of a memory cell, respectively. Then, average of bit line length from any memory cell to the output point is



$$\frac{\lambda_y + \lambda_x + \lambda_x \cdot X + \lambda_y \cdot Y}{2} \quad . \tag{5.1}$$

Since $NW = X \times Y$, values of X and Y which minimize average of bit line length (5.1) are

$$X = \sqrt{\frac{\lambda_y \cdot NW}{\lambda_x}}, \quad Y = \sqrt{\frac{\lambda_x \cdot NW}{\lambda_y}} \text{ , respectivery.}$$

Consequently, average of capacitive load of bit line from any memory cell to the output point is in proportion to the square root of the number of words. If capacitive loads of bit lines are enough big to take into account, a lower bound of an average of read energy consumption is also in proportion to the square root of the memory size. □

Many researchers have proposed low power techniques which make energy consumed during memory access not to be in proportion to memory size[38, 35, 15, 57]. The basic idea is to divide the memory in different blocks and powered-up only a subset of them for any one access. However, even if a memory block partitioned into small sub-blocks and a bit line from each sub-block to an output point is independently charged or discharged, average of energy consumption can not be lower than the order of square root of memory size.

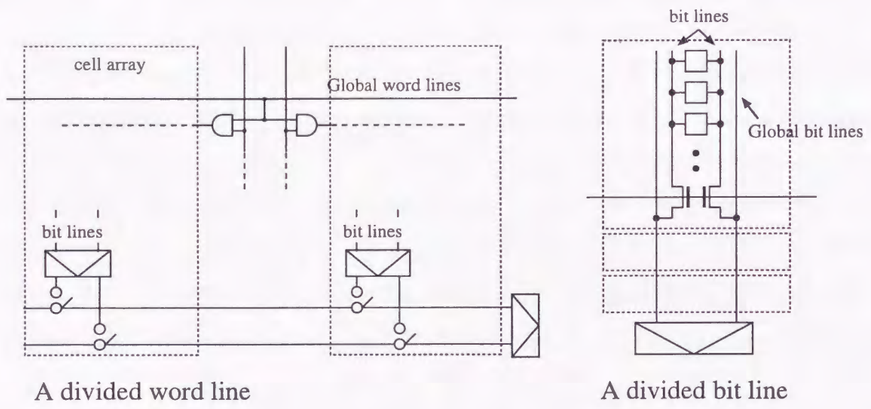


Figure 5.1: A divided Word-Line and a divided Bit-Line structure.

A curve labeled **Divided Array** appeared in Figure5.1 shows energy consumption in ROM modules whose array part is divided so as to minimize energy consumption. The basic idea of the divided array is to divide the memory in different blocks and powered-up only a subset of them for any one access[38, 35, 15, 57]. This memory power model assumes weak correlation between power consumption and size of memory, compared with the memory power model used in [65]. This curve is approximated using information of load capacitance of sense amplifiers, bit lines, word lines, and address decoders of actual ROMs. The 32 bits ROMs

ranging in words from 64 to 4,096 are generated by Alliance CAD System Ver. 3.0 with $0.5\mu m$ double metal CMOS technology. The curve demonstrates that the energy consumption in ROMs is proportional to square root of the number of words. We can approximate **Divided Array** curve as (5.2). The energy consumption of 32 bits RISC microprocessor designed with $0.5\mu m$ double metal CMOS technology is also appeared in Figure5.1. Experimental results shown in Section 5.4 are derived using (5.2).

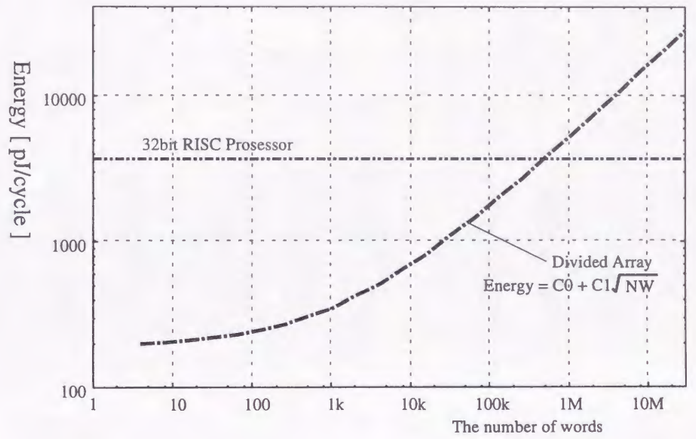


Figure 5.2: The number of words VS. read energy consumption.

$$E_{model} = 190 + 4.8 \cdot \sqrt{NW} \quad [pJ/cycle] \tag{5.2}$$

5.2.2 Memory Reference Locality

In many applications, only a few parts of programs are frequently executed. Furthermore, some parts of the programs are not executed for some kinds of input data. To demonstrate this fact, we measured access locality of instruction memory with three kinds of programs: Arithmetic calculator, MPEG2 decoder, and MPEG2 encoder. The measured results are shown in Fig. 5.3. Vertical axis of a line chart in Fig. 5.3 represents the number of execution cycles produced by a few basic blocks whose total size(the number of words) is only 1% of

total program size. A *basic block* means a sequence of consecutive program statements in which flow of control enters at the beginning and leaves at the end without halt or possibility of branching except at the end[6]. In the benchmark programs, only a few basic blocks whose total size is only 1% of total program size produce more than 50% of execution cycles. Reducing the energy dissipation of frequently accessed memory blocks is effective way to reduce total energy consumption in memory.

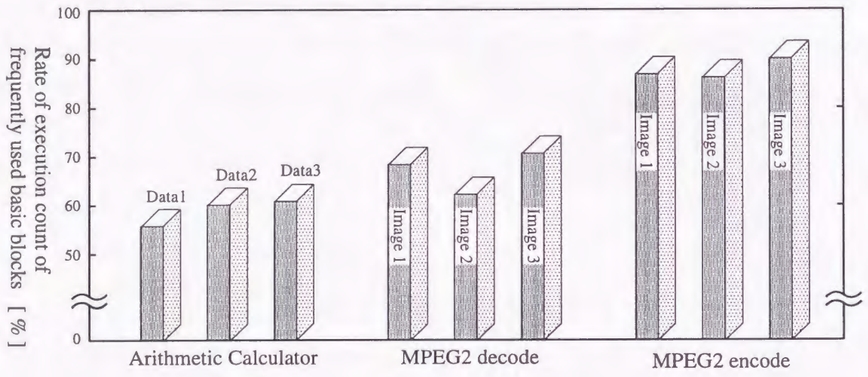


Figure 5.3: Memory reference locality for sample programs.

5.2.3 Our Approach

As mentioned in Section 5.2.1, read energy consumption strongly depends on the physical memory size. Therefore, large size memory consumes higher energy than smaller size memory. Under this situation, allocating frequently executed instructions into smaller memory block called an instruction decompressor leads to the significant energy reduction. However, merging too many instructions into the instruction decompressor causes an increase of energy consumption in the instruction decompressor. Our power optimization technique finds optimal point of this trade-off where average of energy consumption is minimized. The optimization technique targets systems which assume the following.

- Instruction memories are organized by two on-chip ROMs, a main program memory and an instruction decompressor.

- These two ROMs can be independently powered-up or powered-down by special instructions.

- A compiler optimizes size of the ROMs and determines which sequence of object codes should be merged into single instructions so as to minimize total read energy consumption.

In many embedded systems, object code of the application programs need not to be modified after system design is completed. Therefore, the object code of the programs are stored into ROM. In this chapter, we also target systems which assume that the object code is stored into ROM and is not modified after the system organization is fixed.

The most important features of the technique are that (i) the size of two instruction ROMs and allocation of object code are optimized for each application, and (ii) no other structures except for instruction ROMs need to be modified for each application. The feature (i) and (ii) produce significant energy reduction of instruction ROMs, and short turn-around-time of embedded system design, respectively. Our optimization technique realizes application specific low power system design within short turn-around-time.

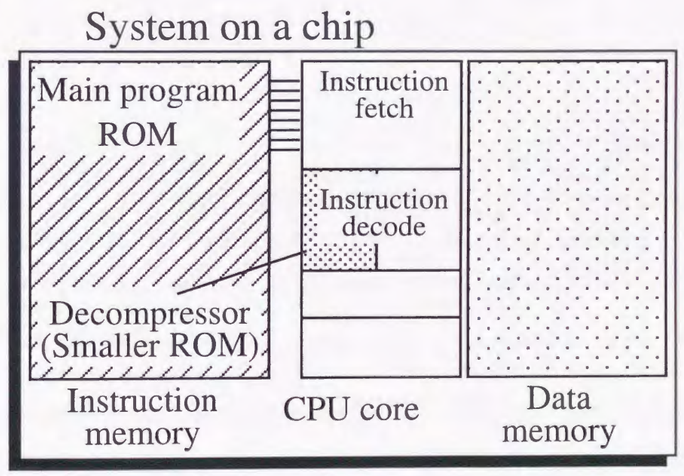


Figure 5.4: Power reduction considering memory reference locality.

5.3 Power Optimization with Object Code Merging

At first, we propose a novel power optimization technique with object code merging, in this section. Next, we formulate a *memory optimization problem* as an ILP(Integer Linear Programming) problem, where *memory optimization problem* is a problem to determine which sequences of object codes should be merged into single instructions.

5.3.1 Optimization Flow

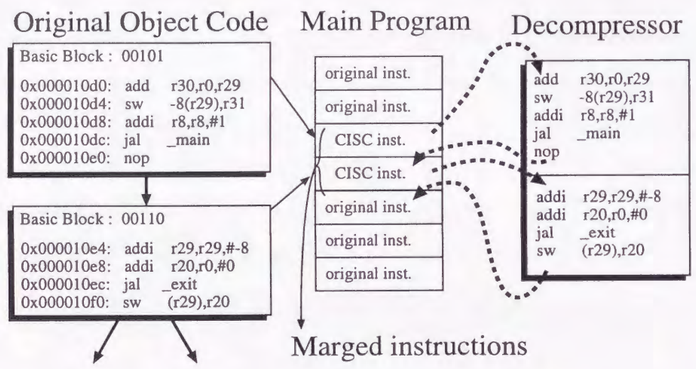


Figure 5.5: Power reduction with an instruction decompressor.

The procedure of our power optimization technique is described below.

1. Given an object code of the target program.

2. Measure the execution count of each basic block using some sample data.

3. Formulate energy dissipation of ROM modules as the function of memory size. The energy dissipation model must be made considering circuit and process technologies applied to ROM modules. Detailed discussion of ROM power models has been presented in Section 5.2.1.

4. For a given set of basic blocks with the information of the execution count of each basic block and a given energy dissipation model of ROM modules, optimize memory organization so as to minimize average of energy consumption. We give a detailed explanation about the *memory optimization problem* in Section 5.3.3.

5. Adjust the physical size of both the main program memory and the instruction decompressor to the minimum required area.

5.3.2 Architecture

The frequently executed basic blocks are replaced with a special instruction, named **CISC instruction** and the special instructions are allocated into main program memory as shown in Fig. 5.5. When the **CISC instruction** is executed, the decompressor is activated. The instruction decompressor is also implemented by ROM circuit. When instructions except for the **CISC instruction** are fetched from the main program memory, these instructions are executed without activating the decompressor. The key point of this power optimization method is that only a few parts of object code are replaced with the special instruction at compiling phase, so as to keep energy consumption in the decompressor very small.

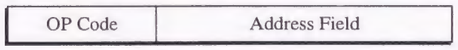


Figure 5.6: CISC instruction format.

As shown in Fig. 5.6, the **CISC instruction** consists of an operation code field and an address field. The original object codes are started to readout from the address of the decompressor.

Our optimization technique targets a system as shown in Fig. 5.7. In this system, an instruction is fetched from main program memory, at first. If the instruction fetched from main program is CISC instruction, the decompressor is activated, otherwise, the fetched instruction is directly executed without activating the decompressor. When the CISC instruction is fetched, main program memory is powered-down until a branch instruction is fetched from

the decompressor. If branch instructions are fetched from the decompressor, the decompressor is powered-down and main program memory is powered-up. The instruction next to the branch instruction is always fetched from main program memory.

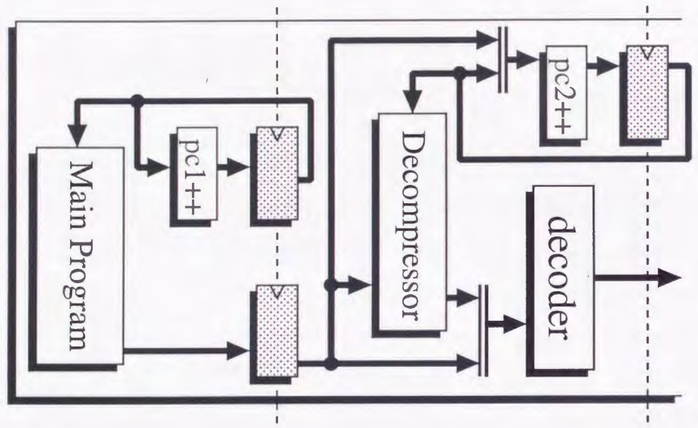


Figure 5.7: Memory architecture for power optimization.

We define a state of processor while the instructions are being fetched from the instruction decompressor as **decompression state**. In this state, the main program memory does not dissipate any energy. Conversely, a state while the main program memory is being accessed is defined as **main program state**. In this state, decompressor is inactivated and does not dissipate energy. State transition is occurred only when **CISC instruction** is fetched in the **main program state** or branch instruction is executed in the **decompression state**. The execution of instruction is delayed one cycle only after the branch instruction is executed. Since the decompressor is accessed at the next of a pipeline stage where main program is accessed, the **CISC instruction** needs no delay cycles. We suppose that the interruption routine is processed only in the **main program state**. Detailed discussion of interruption processing is our future work.

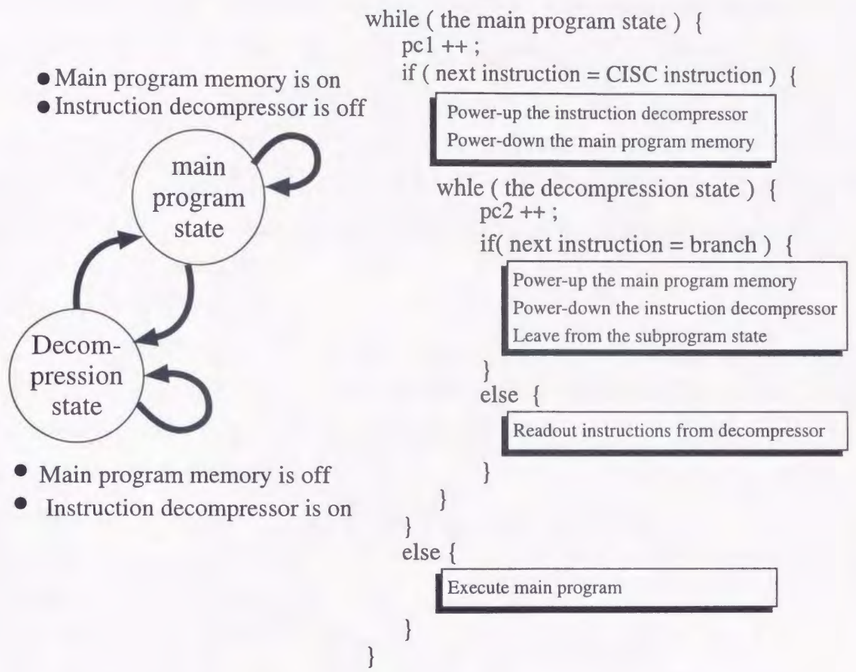


Figure 5.8: State transition graph of execution mode.

5.3.3 ILP Formulation

In this section, we present an ILP(Integer Linear Programming) formulation for the *memory optimization problem*. Firstly, we give notations used in the formulation. Next, we present an ILP formulation.

- N : The number of basic blocks appear in a given program.
- B : A set of basic blocks
- b_i : The ith basic block. $b_i = (S_i, X_i, a_i) \in B$
- S_i : The number of instructions included in ith basic block.
- X_i : The execution count of ith basic block.

- $E(S)$: The average of read energy consumption in ROM module whose size is S bit.
- a_i : Integer variables associated with b_i.
  $a_i = 1$ if b_i is merged into the CISC instruction; otherwise, $a_i = 0$.
- $Main$: The average of read energy consumption in the main program memory.
- Dec : The average of read energy consumption in the decompressor.

$$OBJ = Main \cdot \sum_{i=0}^{N} \{S_i \cdot X_i \cdot (1 - a_i)\} + Main \cdot \sum_{i=0}^{N} (X_i \cdot a_i) + Dec \cdot \sum_{i=0}^{N} (S_i \cdot X_i \cdot a_i) \quad (5.3)$$

$$Main = E\left(\sum_{i=0}^{N} \{S_i \cdot (1 - a_i)\} + \sum_{i=0}^{N} a_i\right) \quad (5.4)$$

$$Dec = E\left(\sum_{i=0}^{N} (S_i \cdot a_i)\right) \quad (5.5)$$

The *memory optimization problem* can be formally defined as follows.

" For a given set of basic blocks B, find a set of a_i which minimize OBJ".

5.3.4 Algorithm

The worst case computation time to solve the *memory optimization problem* is $O(2^N)$, where N represents the number of basic blocks appeared in a program. Therefore, if a naive algorithm is applied, the problem can not be solved within feasible time for programs which consist of a lot of basic blocks. Experimental results shown in the succeeded section are derived by the greedy algorithm shown in Fig. 5.9. The complexity of this heuristic algorithm is $O(N^2)$.

The input to algorithm *Memory optimization* is a set of basic blocks B, where each basic block $b_i \in B$ is characterized by its execution count X_i, its size S_i, and its location a_i. The a_is are integer variables associated with b_i The a_is are set to one if b_i is merged into the CISC instruction, otherwise, a_is are set to zero. All the a_i is set to zero at first step. This means that all the basic blocks are allocated into the main program memory. Next, the algorithm provisionally selects a single basic block and merges a basic block into the CISC instruction. The selected basic block is temporarily allocated into the decompressor.

After calculating $Cost = OBJ$, the provisionally selected basic block is returned to the main program memory. This process is performed for each basic block in the program. After that, the algorithm selects a single basic block which minimize the $Cost$, and the selected basic block is allocated to the decompressor so as to be restored the original object codes. Basic blocks located in main program memory is successively moved in this manner while the $Cost$ is reduced. If the $Cost$ becomes not to be improved, the algorithm stops after outputting an updated set of basic block B.

Given: a set of basic blocks B, where each basic block $b_i \in B$ is characterized by its execution count X_i, its size S_i, and its location a_i.

Algorithm Memory optimization

```
for each b_i
    a_i=0;
end for
E_min = ∞;
while (the algorithm leads to reduction in E_min)
  for (i = 0 ... N − 1)
    if (a_i = 0)
      b_i = (S_i, X_i, 1) ;
      Cost = OBJ ;
      if (Cost < E_min)
        E_min = Cost ;      m = i ;
      end if
      b_i = (S_i, X_i, 0) ;
    end if
  end for
  b_m = (S_m, X_m, 1) ;
end while
Output a set of basic blocks B ;
```

end Algorithm

Figure 5.9: Algorithm for the memory optimization.

5.4 Experimental Results

We use three benchmark programs shown in Table 5.1, in the experiments. The benchmark programs are compiled by gcc-dlx compiler which is based on GNU CC Ver. 2.7.2 for DLX architecture[25]. Table 5.2 shows description of three kinds of sample video images used as input to the MPEG2 program. Three kinds of sample input data, were also used for Arithmetic calculator and TV remote controller, respectively.

Table 5.1: Description of benchmark programs.

Benchmark	The number of basic blocks
Arithmetic Calculator	2,103
TV Remote Controller	2,994
MPEG2 decoder	5,361
MPEG2 encoder	6,087

Table 5.2: Description of sample data.

Data	The number of frames	The size of frames
Image1	50	416x386
Image2	26	352x224
Image3	14	704x480

The following two object code merging techniques are evaluated in this section.

- A basic block merging approach

  This approach merges a frequently executed basic block into the CISC instruction. Detailed discussion is done in succeeding subsection. The basic block merging approach is also evaluated under the memory area constraints.

- A sequence merging approach

  This approach merges together a few basic blocks into the CISC instruction. Detailed discussion is done in the Section 5.4.2.

5.4.1 A Basic Block Packing Approach

Our power optimization techniques can be expected to get much effectiveness under the following conditions.

1. The execution count of the frequently executed basic blocks and that of the rarely executed basic blocks are extremely different from each other.

2. The execution count of each basic block weakly depends on input data.

In section 5.2.2, we have shown that the execution count of each basic blocks are extremely different from each other. Next, we experimented our memory optimization technique by the following way.

1. Measure the execution count of each basic block with a sample data.

2. Determine which basic blocks are packed into the CISC instruction. Consequently, physical size of the two memories and allocation of basic blocks to the memories are decided.

3. Estimate the average of energy consumption for three kinds of input data with the previously optimized memory organization.

The purpose of this experiment is to show the effects of data which are used for memory optimization on finally evaluated energy consumption. The optimized memory size are shown in Table 5.3. Each value represents the number of words of two instruction memories which are optimized for each input data. The results indicate that the size of the instruction decompressor are from 2% to 4% of main program memory for all benchmark programs. It is a key point of our memory optimization technique that the size of the instruction decompressor is restrained very small. This leads to a drastic reduction in total read energy consumption.

Table 5.3: The optimized memory size.

Arithmetic Calculator			
Optimized for	DATA1-1	DATA1-2	DATA1-3
Decompressor	271	308	272
Main program	10,682	10,645	10,679
TV Remote Controller			
Optimized for	DATA2-1	DATA2-2	DATA2-3
Decompressor	1,067	357	1,083
Main program	14,151	14,777	14,135
MPEG2 decoder			
Optimized for	Image1	Image2	Image3
Decompressor	1,134	1,212	988
Main program	28,407	28,338	28,542
MPEG2 encoder			
Optimized for	Image1	Image2	Image3
Decompressor	752	787	622
Main program	35,975	35,946	36,088

Approximated access time in ROMs are shown in Figure 5.10. A target process technology is 1.2 micron CMOS. A solid line represents the measured access time of actual 32 bits ROMs ranging from 64 words to 4096 words. A broken line is approximated by the measured values.

The access time includes the time for a precharge and evaluation. This figure demonstrates that the access time of the decompressor is almost half of that of the main program memory.

It is also possible to develop the decompressor with the wired logic. However, developing the decompressor with the wired logic requires modification to the processor architecture. It needs much more turn around time than the decompressor design with ROMs does. The merits of utilizing ROMs for the decompressor are design flexibility and suitability for IP-base system design. Our code merging technique is well-suited for systems employing IP cores whose internal architecture can not be modified, because our technique requires no

modification to the processor architecture.

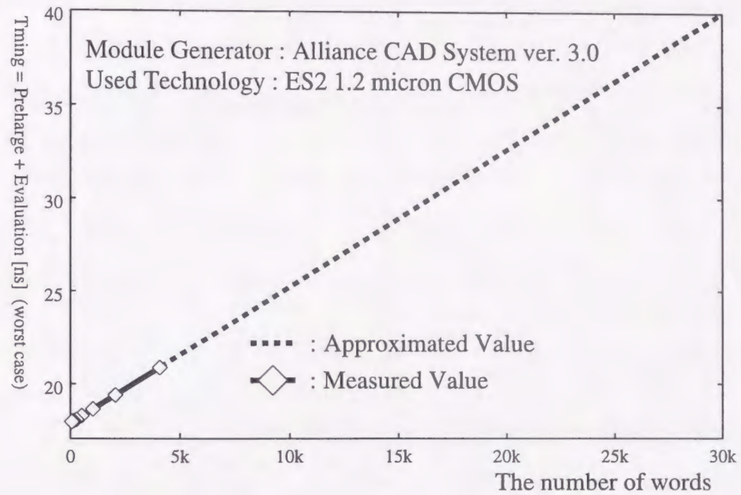


Figure 5.10: Approximated access time in ROMs.

The energy reduction rates are shown in Table5.4. All the values appeared in Table5.4 are calculated by (5.6).

$$\frac{Optimized\ energy\ with\ code\ Marging}{Apploximated\ energy\ accouding\ to\ (5.2)} \times 100(\%) \tag{5.6}$$

The second column of Table5.4 shows the input data which are used for memory optimization. The leftmost row of Table5.4 shows the input data which are used for evaluation of energy consumption with the previously optimized memory organization. We have used the divided bit and word lines structure as memory models in this experiment. Therefore, read energy consumption is assumed to be in proportion to the square root of memory size.

The results show that energy reductions strongly depends on the kinds of program, but weakly depends on input data. Therefore, optimizing memory organizations with appropriate input data can be almost optimal memory organizations for the other input data. In addition, the results show that the energy is always the smallest when the data used for optimizations

and the data for energy evaluations are the same. Therefore, we can regard that the heuristic algorithm described in Fig. 5.9 can find almost optimal solutions which are very close to the optimal solutions.

Table 5.4: The energy consumption in memory optimized without area constraints.

Arithmetic Calculator			
Optimized for	DATA1-1	DATA1-2	DATA1-3
Executed data			
DATA1-1	**58.45%**	59.70%	58.82%
DATA1-2	60.27%	**59.72%**	60.08%
DATA1-3	59.78%	59.96%	**59.69%**
TV Remote Controller			
Optimized for	DATA2-1	DATA2-2	DATA2-3
Executed data			
DATA2-1	**64.84%**	67.51%	66.39%
DATA2-2	63.29%	**61.57%**	62.94%
DATA2-3	64.32%	66.93%	**65.83%**
MPEG2 decoder			
Optimized for	Image1	Image2	Image3
Executed data			
Image1	**45.82%**	46.11%	46.48%
Image2	46.81%	**46.46%**	48.80%
Image3	46.42%	46.23%	**45.76%**
MPEG2 encoder			
Optimized for	Image1	Image2	Image3
Executed data			
Image1	**48.65%**	48.90%	48.88%
Image2	50.11%	**49.67%**	50.38%
Image3	49.00%	49.22%	**48.82%**

5.4.2 A Sequence Merging Approach

The issue for the basic block packing technique is an increase of the energy dissipation for reading the CISC instructions from the main program memory. The CISC instructions are frequently readout from main program memory and the energy for reading CISC instruction from the main program memory can not be neglected. Therefore, merging a sequence of few basic blocks into the single CISC instruction as shown in Figure 5.11 is more effective. The basic idea is to merge more than one basic block, which is on the frequently executed loops, into the CISC instruction. Of course, this technique needs some extra jump instructions to return to the main program from the basic blocks allocated in the decompressor. These jump instructions cause performance overhead. Evaluation of the worst case execution time considering these overhead is our future work. We applied the sequence merging technique to the benchmark programs appeared in Table 5.1. The environment for the experiments are the same from the environment of experiments described in the previous subsection. The optimized memory sizes and energy consumptions of memories in which the sequence merging technique is applied are shown in Table 5.5 and 5.6, respectively. We can see the tables 5.5 and 5.6 in the same way as the tables 5.3 and 5.4, respectively.

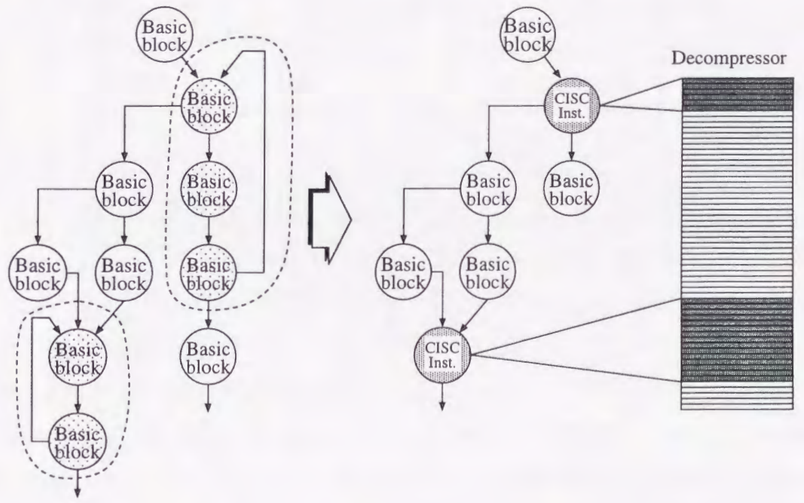


Figure 5.11: An example of sequence merging.

Table 5.5: The optimized memory size.

Arithmetic Calculator			
Optimized for	DATA1-1	DATA1-2	DATA1-3
Decompressor	312	296	327
Main program	10,620	10,626	10,596
TV Remote Controller			
Optimized for	DATA2-1	DATA2-2	DATA2-3
Decompressor	1,534	401	1,504
Main program	13,827	14,935	13,863
MPEG2 decoder			
Optimized for	Image1	Image2	Image3
Decompressor	1,027	1,171	1350
Main program	28,483	28,337	28,165
MPEG2 encoder			
Optimized for	Image1	Image2	Image3
Decompressor	921	880	734
Main program	35,768	35,816	35,970

The size of the decompressors and the main program memories shown in Table 5.5 are evenly matched to the size of memories which are applied the basic block packing technique. This is because the basic blocks allocated into the decompressor by the sequence merging technique will be almost the same from the basic blocks allocated by the basic block packing technique. Because of higher memory reference locality, the energy reduction rate for stream data processing programs will be higher than that for control dominated programs. For example in the MPEG2 encoder, more than 65% energy reduction can be achieved, which is better than the results for the TV controller by 25%. From 7% to 10% improvements in energy consumption are achieved by the sequence merging technique compared with the systems in which the basic block packing technique is applied.

Table 5.6: The energy consumption of memory with a sequence compression.

Arithmetic Calculator			
Optimized for	DATA1-1	DATA1-2	DATA1-3
Executed data			
DATA1-1	**47.00%**	49.95%	47.65%
DATA1-2	50.39%	**47.86%**	48.71%
DATA1-3	49.23%	49.64%	**48.66%**
TV Remote Controller			
Optimized for	DATA2-1	DATA2-2	DATA2-3
Executed data			
DATA2-1	**53.35%**	60.71%	52.98%
DATA2-2	52.22%	**51.56%**	52.05%
DATA2-3	52.20%	59.75%	**51.97%**
MPEG2 decoder			
Optimized for	Image1	Image2	Image3
Executed data			
Image1	**40.70%**	41.32%	42.45%
Image2	41.41%	**40.35%**	42.14%
Image3	48.67%	48.15%	**42.25%**
MPEG2 encoder			
Optimized for	Image1	Image2	Image3
Executed data			
Image1	**37.35%**	37.81%	38.66%
Image2	33.31%	**32.85%**	34.14%
Image3	36.54%	36.52%	**36.59%**

5.4.3 A Basic Block Packing Approach under Area Constraints

Up to here, we discuss on the memory optimization without considering the area of memory. If we can ignore an increase of memory area, it can be said that reducing energy consumption with divided bit and word lines structure is feasible solution. However, this solution may be infeasible when an increase of memory area is not negligible, since dividing bit lines or word lines lead increase in area as shown in Fig. 5.12. Therefore, memory modules which follows (5.2) can not always be developed under area constraints. In this section, we discuss on the memory optimization under the area constraints.

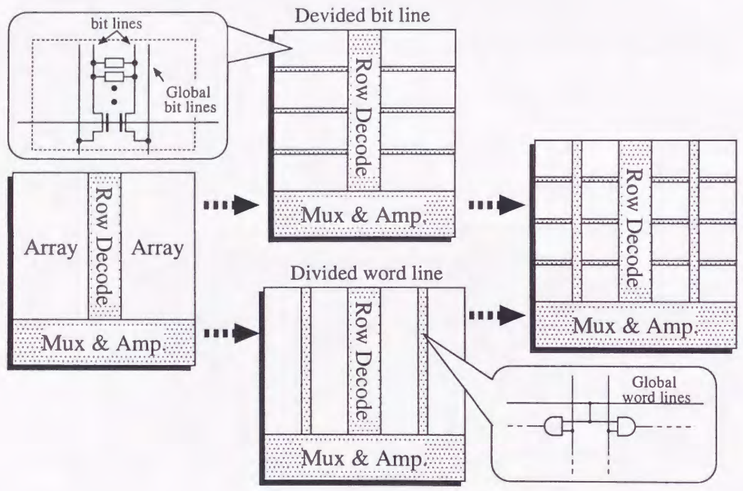


Figure 5.12: Increase of memory area caused by divided bit and word line structure.

Area of memory depends not only on the number of bits but also on the number of divided sub-blocks. Therefore, the size of divided sub-blocks should be restricted for the area reduction. To take the area of memory into account, we use an original area model based on actual designs of ROM modules which are created by Alliance CAD System Ver. 3.0 with $0.5\mu m$ double metal CMOS process technology. To make possible to activate only a single specified sub-block, some logic cells and MOS switches are required as shown in Figure 5.12.

The power-area trade-off in the memory should be considered to develop a low-power and small memory. The energy consumption for the arithmetic calculator, TV remote controller, MPEG2 decoder, and MPEG2 encoder under area constraints are shown in Fig. 5.13, 5.14, 5.15, and 5.16, respectively. Curves labeled **Divided array structure** shown in Fig. 5.13, 5.14, 5.15, and 5.16, represent energy consumption in ROM modules which employ only divided bit and word lines structure under area constraint. We assume that only a single specified memory block is activated if the divided array structure is applied. Curves labeled **Object code packing** represent energy consumption in ROM modules optimized by the object code packing technique under the area constraints. Of course, both the main program memories and the instruction decompressors are applied the divided bit and word lines technique. The horizontal axis of Fig. 5.13, 5.14, 5.15, and 5.16 represent rates of memory area normalized by the memory which is optimized so as to minimize the area occupancy without applying divided bit line, divided word line, or the basic block packing technique.

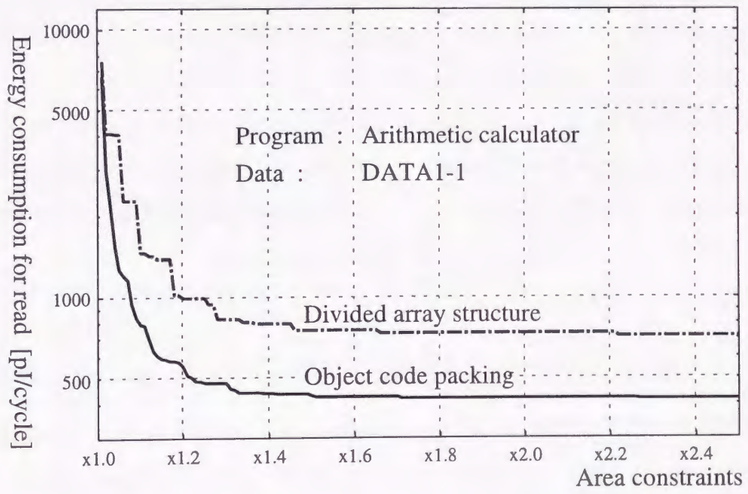


Figure 5.13: The results for the arithmetic calculator.

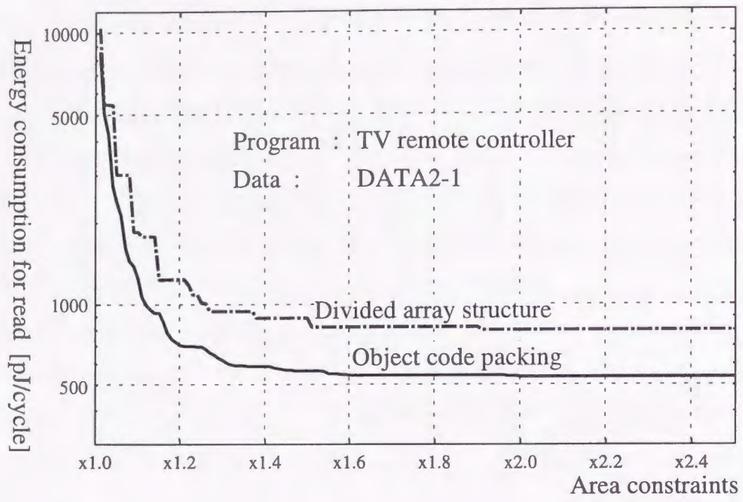


Figure 5.14: The results for the TV remote controller.

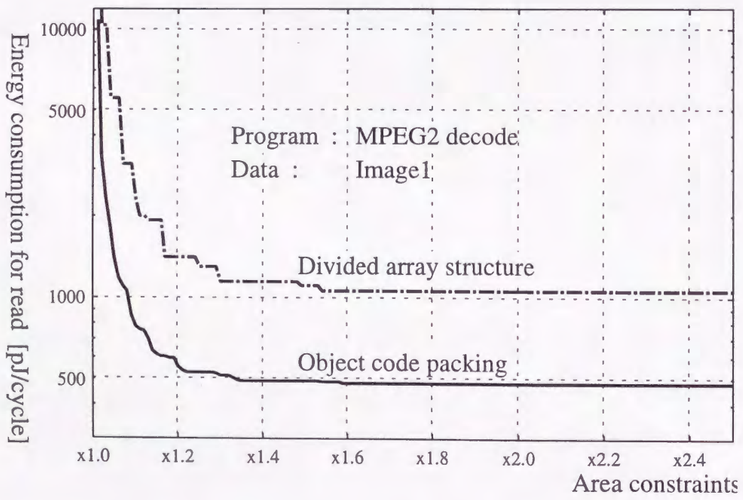


Figure 5.15: The results for the MPEG2 decoder.

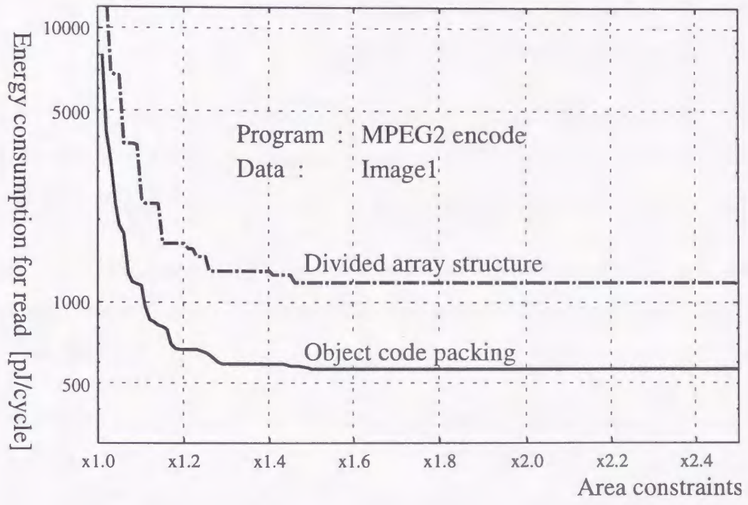


Figure 5.16: The results for the MPEG2 encoder.

The results show that energy consumption in memories optimized by our optimization technique can be less than one third of energy of memories which is applied divided bit and word lines structure. For example, experimental results for MPEG2 decoder demonstrate that the energy consumption in the memories optimized by the memory optimization technique exactly be one third of energy consumption in memory which is applied divided bit and word lines structure where area constraint is $\times 1.09$. Therefore, our memory power optimization technique is more effective under the area constraints. The results demonstrate that the proposed memory optimization technique can reduce energy consumption with smaller area loss.

5.5 Summary

In this chapter we propose an application specific power optimization technique for embedded systems utilizing small instruction decompressor. In addition, we experimentally show that average of read energy consumption in a memory is proportional to square root of the size of the memory. Therefore, frequently executed sequences of object codes should be merging

into the low power decompressor so as to reduce total read energy consumption.

Experimental results demonstrate the following.

- The energy consumption in the memories optimized by our memory power optimization technique can be less than half of energy consumption in memories which are applied only divided bit and word line structure.
- The energy reductions strongly depends on the kinds of program, but weakly depends on the kinds of input data. Therefore, optimizing memory organizations with appropriate input data can be almost optimal memory organizations for the other input data.
- The energy consumption in memories optimized by the proposed technique under the area constraints can be less than one third of energy of memories which are applied only divided bit and word lines structure.

Our future work will be devoted to extend the proposed optimization technique considering access time of memory and to apply this technique to the data memory.

Chapter 6

Conclusions

6.1 Summary of Contributions

Today, one of the most serious concerns for the VLSI designer is a power consumption. The primary driving factor of the power issue must be explosive growth in the portable systems which demand high-speed computation and complex functionality with low power consumption. Furthermore, there is a clear financial advantage to reducing the power consumed in high performance systems, because the cost associated with packaging and cooling such devices is huge. Therefore, both a heat reduction of chips and a battery weight reduction are required in the prospective VLSI systems.

In todays system design, power optimizations at higher level of abstraction are required, because decision at higher level of abstraction strongly affect to the cost, performance, and power consumption of final products. A VLSI system design can be represented at several levels of abstraction such as a layout level, a logic circuit level, an architectural level, or a system level. The system level optimization involves partitioning the tasks into hardware and software, choosing the processor architecture that will execute the software, determining the hardware/software communication mechanism, and so on. This thesis has concentrated on power optimization techniques at the system level abstraction. The techniques addressed in this thesis have performed the power optimization taking both hardware and software into account, and have demonstrated much more impacts on the quality of the final products than lower level optimization techniques have.

The goal of this research is to realize not only low power system but also high performance

system with low power dissipation. Reducing the energy consumption in a whole system including hardware and software is also the goal of this study. The hardware often implemented by embedded processor. The software and data which are executed on the processor are usually installed in an instruction memory and a data memory, respectively. Therefore, not only the processor but also the memories should be optimized for low power.

In Chapter 2, low power system design methodologies concentrated on the system level approaches are outlined. These approaches demonstrate that a system level decision has strong impacts on energy consumption as well as on system cost and performance.

In Chapter 3, we have introduced theories for the dynamically variable supply voltage processor. We also have proposed a voltage scheduling problem and a voltage scheduling algorithm. The study of the variable voltage scheduling is a challenge to answer the basic question that how low energy processing can be done for a given task under a time constraint. The theorems described in chapter 4 give answers for this question. We can summarize basic theorems to find the optimal voltage schedule as follows:

- If a processor can use arbitrary variable voltage, a voltage which adjusts the finishing time of the program to the deadline time is an unique supply voltage which minimizes energy consumption.

- If a processor can use only a small number of discretely variable voltages, the voltage schedule with at most two voltages minimizes the energy consumption;.

This research is very important in both theoretical and practical points of view.

In Chapter 4, a new microprocessor architecture, called *Power-Pro* architecture, have been presented. It is a first challenge for a system design that datapath width and supply voltage which are "hard" part so far are enabled to softly be controlled by the software. Since software usually knows how much clock frequency, supply voltage and datapath width are required for processing a program, it often have an advantage to control them by the software. In these days, some papers propose ideas where supply voltage or datapath width are controlled by the operating system (OS) or application programs. It gradually becomes common sense of low power system design that enabling supply voltage and datapath width to be controlled by the software has much advantage for power reduction. Experimental results with several benchmark programs demonstrate energy reduction by 35% by using the variable datapath

width scheme, and energy reduction down to 1/10 at best case by the variable supply voltage scheme.

In Chapter 5, memory power optimization techniques with object code merging have been proposed. It can be said that the object code merging technique is one of the typical hardware/software codesign approach. Especially for the digital signal processor (DSP) designers, merging frequently executed sequence of object codes into a single complex code has conventionally been done. For example, in MPEG decoding and encoding programs, many *addition* instructions are executed immediately after the *multiply* instructions. This results in the need for a new complex instruction called *multiply-add* instruction which performs the addition immediately after the multiplication within a single clock cycle. Determining what kinds of complex instructions are effective for the system performance, cost, and power is the most important process for hardware/software partitioning. This research has challenged to generalize the process to optimize an instruction set architecture of complex instruction set computer (CISC) processor. Utilizing the object code merging technique, the energy consumption during the instruction memory access can be halved without any modifications of computational scheme.

The power reduction techniques which have been presented in this thesis can be applied together to wide range of digital systems. Therefore, the great power reduction can be expected by these techniques, even if the power improvement by each of the techniques is not so much. The variable datapath width control scheme can reduce the energy consumption of the data memory and the datapath part of the processor by 35%, respectively. The energy consumption of the processor can be reduced by 90% at best case by using the variable voltage control scheme. The energy consumption of the instruction memory can be halved by the object code merging technique. If the control logic circuit of a processor, the processor's datapath, an instruction memory, and a data memory dissipate 25% of total energy dissipation in a conventional system, we can reduce two third of the total energy dissipation in the same system by applying out proposed techniques.

6.2 Future Directions

Our future work will be devoted to extend the work presented in this thesis to the following directions.

For the variable voltage scheduling technique which has been presented in this thesis, the worst case execution cycles of the application program must statically be estimated. However, we have never established the method to estimate the worst case execution cycles for actual systems which include cache memory or support a interruption processing. The method to accurately estimate the worst case execution cycles for actual systems should be established. The study of dynamic voltage scheduling with operating systems is going on now[56]. The voltage scheduling algorithms which take switching delay to change the supply voltage into account should also be established.

This thesis also proposes a novel microprocessor architecture which enables the software to control the datapath width, the supply voltage, and clock frequency of the processor. However, the methodology to determine what kind of datapath width and supply voltage should be employed by the processor is not presented. The method to determine the kinds of variable datapath width and variable supply voltage to be equipped by the processor should be studied.

The object code merging technique utilizing a history information of the execution counts of basic blocks has been proposed and has demonstrated the good improvements in energy dissipation. However, packing the basic blocks into a single complex instructions sometimes is not optimal solution. Merging a short object code sequence which are smaller than a basic block into a single instruction can achieve more energy reductions. The generalization of the object code merging approach should be done in the future.

Acknowledgment

I would specially like to express the sincerest gratitude to my advisor, Professor Hiroto Yasuura, for giving opportunities to work in the fields of the low power system design, and for making excellent suggestions and helpful comments on various parts of this thesis. It is my sincere pleasure to have studied under a guidance of Professor Yasuura for six years.

I also would like to thank Professor Kazuaki Murakami and Professor Mizuho Iwaihara of Kyushu University for giving invaluable suggestions and directions throughout this work.

I am very grateful to Professor Yukinori Kuroki and Professor Rinichirou Taniguchi for serving on committee members of this thesis and providing thoughtful suggestions.

I am greatly indebted to the past and present members of Yasuura Laboratory, who gave numerous helpful suggestions and guidance to the author, including Dr. Hiroyuki Tomiyama (currently with University of California, Irvine) who gave helpful suggestions and advises to me, and Dr. Hiroki Akabosi (currently with NEC Corporation) who gave helpful suggestions and guidance to me when I started the study in this area. I am also thankful to Akihiko Inoue, Koji Inoue, Kei Hirose, and Eko Fajar Nurprasetyo for their cooperations. I also grateful to Makoto Sugihara, Takanori Okuma, and the other members of the laboratory who made me to have good time in this laboratory.

The author also would like to thank the members of the ISIT/KYUSHU (Institute of Systems & Information Technologies / KYUSHU). Special thanks are due to Koji Kai who gave helpful suggestions and advises to me. I am also thankful to Dr. Hirosi Date, Hideaki Fujikake, and the other members of ISIT for their cooperations.

I also would like to thank the members of the pilot chip design project sponsored by the VLSI Design and Education Center (VDEC) in the University of Tokyo. Special acknowledgments are due to Dr. Makoto Ikeda of the University of Tokyo, Dr. Kazutoshi Kobayashi of Kyoto University, Dr. Hiroyuki Ochi of Hiroshima City University, Mr. Makoto Nagata of

Hiroshima University, and Professor Shoji Kawahito of Toyohashi University y of Technology for giving numerous suggestions in this research at several conferences and meetings.

I have had opportunities to discuss with Professor Ing-Jer Huang of National Sun Yat-Sen University and Professor Sri Parameswaran of University of Queensland when they visited Kyushu University. I am indebted to them for their technical insights and constructive comments.

Finally, but not least, I thank my parents and friends for their continuous supports and encouragements all through my career.

My financial support for this study has been provided by the Research Fellowship of the Japan Society for the Promotion of Science for Young Scientists.

Bibliography

[1] A. Inoue, H. Tomiyama, T. Ishihara, and H. Yasuura, "System-Level Energy Optimization for Embedded Systems with a Variable Datapath Width Processor (in Japanese), " *IEICE Technical Report,* VLD98–29, DSP98-58, June 1998.

[2] A. Inoue, H. Tomiyama, T. Okuma, H. Kanbara, and H. Yasuura, "Language and Compiler for Optimizing Datapath Widths of Embedded Systems, " *IEICE Trans. on Fundamentals*, vol. E81–A, no. 12, pp. 2595–2604, December 1997.

[3] A. J. Stratakos, R. W. Brodersen, and S. R. Sanders, "A Low-Voltage CMOS DC-DC Converter for a Portable Battery-Operated System, " In *Proc. of the IEEE Power Electronics Specialists Conference*, pp. 105–110, Apr. 1994.

[4] R. Mehra J. Rabaey A. P. Chandrakasan, M. Potkonjak and R. W. Brodersen, "Optimizing Power Using Transformation, " *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 1, pp. 12–31, January 1995.

[5] Abhijit Ghosh, Srinivas Devadas, Kurt Keutzer, and Jacob White, "Estimation of Average Switching Activity in Combinational and Sequential Circuits, " In *Proc. of 29th Design Automation Conference*, pp. 253–259, 1992.

[6] A. V. Aho, R. Sethi, and J. D. Ullman, *Compilers: Principles, Techniques, and Tools*, Addison-Wesley, 1986.

[7] B. Shackleford, M. Yasuda, E. Okushi, H. Koizumi, H. Tomiyama, and H. Yasuura, "Memory-CPU Size Optimization for Embedded System Designs, " In *Proc. of 34th Design Automation Conference*, pp. 246–251, 1997.

[8] D. Brooks and M. Martonosi, "Dynamically Exploiting Narrow Width Operands to Improve Processor Power and Performance, " In *Proc. of High-Performance Computer Architecture*, January 1999.

[9] C.-L. Su and A. M. Despain, "Cache Design Trade-offs for Power and Performance Optimization: A Case Study, " In *International Symposium on Low Power Design (ISLPD'95)*, pp. 282–286, 1995.

[10] C.-Y. Tsui, M. Pedram, and A. M. Despain, "Efficient Estimation of Dynamic Power Consumption under a Real Delay Model, " In *Proc. of International Conference on Computer Aided Design (ICCAD'93)*, pp. 224–228, 1993.

[11] A. P. Chandrakasan and R. W. Brodersen, *LOW POWER DIGITAL CMOS DESIGN*, Kluwer Academic Publishers, 1995.

[12] A. P. Chandrakasan and R. W. Brodersen, *LOW POWER CMOS DESIGN*, IEEE Press, 1998.

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

[14] E. Kligerman and A. D. Stoyenko, "Real-time Euclid: A language for reliable real-time systems, " *IEEE Trans. on Software Engineering*, vol. SE–12, no. 9, pp. 941–949, September 1986.

[15] Edwin de Angel and Earl E. Swartslander, Jr., "Survey of Low Power Techniques for ROMs, " In *Proc. of International Symposium on Low Power Electronics and Design (ISLPED'97)*, pp. 7–11, 1997.

[16] F. Ichiba, K. Suzuki, S. Mita, T. Kuroda, and T. Furuyama, "Variable Supply-Voltage Scheme with 95%-Efficiency DC-DC Converter for MPEG-4 Codec, " In *Proc. of International Symposium on Low Power Electronics and Design (ISLPED'99)*, pp. 54–59, 1999.

[17] F. N. Eko, H. Tomiyama, A. Inoue, and H. Yasuura, "Soft-Core Processor Architecture for Embedded System Design, " *IEICE Trans. on Electronics*, vol. E81–C, no. 9, pp. 1416–1422, September 1998.

[18] Gu-Yeon Wei and Mark Horowitz, "A Low Power Switching Power Supply for Self-Clocked Systems, " In *Proc. of International Symposium on Low Power Electronics and Design (ISLPED'96)*, pp. 313–317, 1996.

[19] H. Onodera, A. Hirata, T. Kitamura, and K Tamaru, "P2Lib: Process-Portable Library and Its Generation System, " In *Proc. of Custom Integrated Circuit Conference (CICC'97)*, pp. 341–344, 1997.

[20] H. Tomiyama, and H. Yasuura, "Code Placement Techniques for Cache Miss Rate Reduction, " *ACM Trans. Design Automation of Electronic Systems (TODAES)*, vol. 2, no. 4, pp. 410–429, October 1997.

[21] H. Tomiyama, T. Ishihara, A. Inoue, and H. Yasuura, "Instruction Scheduling to Reduce Switching Activity of Off-Chip Busses for Low-Power Systems with Caches, " *IEICE Trans. on Fundamentals*, vol. E81–A, no. 12, pp. 2621–2629, December 1997.

[22] H. Yamashita, H. Tomiyama, A. Inoue, F. N. Eko, T. Okuma, and H. Yasuura, "Variable Size Analysis for Datapath Width Optimization, " In *Proc. of Asia Pacific Conference on Hardware Description Languages (APCHDL'98)*, pp. 47–52, July 1998.

[23] H. Yasuura, H. Tomiyama, A. Inoue, and F. N. Eko, "Embedded System Design Using Soft-Core Processor and Valen-C, " *Journal of Information Science and Engineering,*, vol. 14, no. 3, pp. 587–603, September 1998.

[24] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M. B. Srivastava, "Power Optimization of Variable Voltage Core-Based Systems, " In *Proc. of 35th Design Automation Conference*, pp. 176–181, June 1998.

[25] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann Publishers, Inc., 2nd edition, 1996.

[26] J.-M. Chang and M. Pedram, "Energy Minimization Using Multiple Supply Voltages, " *IEEE Trans. VLSI Systems*, vol. 5, no. 4, pp. 436–443, December 1997.

[27] J. M. Rabaey and M. Pedram, *Low Power Design Methodologies*, Kluwer Academic Publishers, 1996.

[28] Jiing-Yuan Lin, Tai-Chien Liu, and Wen-Zen shen, "A Cell-Based Power Estimation in CMOS Combinational Circuit, " In *Proc. of International Conference on Computer Aided Design (ICCAD'94)*, pp. 304–309, 1994.

[29] Johnson Kin, Munish Gupta, and William Mangione Smith, "The Filter Cache: An Energy Efficient Memory Structure, " In *Proc. of International Symposium on Low Power Electronics and Design (ISLPED'97)*, pp. 184–193, 1997.

[30] K. Inoue, T. Ishihara, and K. Murakami, "A High-Performance and Low-Energy Cache Architecture with Way-Prediction Technique, " In *Proc. of International Symposium on Low Power Electronics and Design (ISLPED'99)*, pp. 273–275, 1999.

[31] K. Ogawa, "PASTEL: A Parameterized Memory Characterization System, " In *Proc. of Design, Automation and Test in Europe*, March 1998.

[32] L. Benini, A. Macii, E. Macii, and M. Pancino, "Selective Instruction Compression for Memory Energy Reduction in Embedded Systems, " In *Proc. of International Symposium on Low Power Electronics and Design (ISLPED'99)*, pp. 206–211, 1999.

[33] L. S. Nielsen, C. Niessen, J. Sparsϕ, and K. V. 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 1994.

[34] M. Igarashi, K. Usami, K. Nogami, F. Minami, Y. Kawasaki, T Aoki, M. Takano, C. Mizuno, T. Ishikawa, M. Kanazawa, S. Sonoda, M. Ichida, and N. Hatanaka , "A Low-Power Design Method Using Multiple Supply Voltages, " In *Proc. of International Symposium on Low Power Electronics and Design (ISLPED'97)*, pp. 36–41, Aug. 1997.

[35] M. Isobe, J. Matsunaga, T. Sakurai, T. Ohtani, K. Sawada, H. Nozawa, T. Iszuka, and S. Kohyama, "A Low Power 46ns 256K bit CMOS Static RAM with Dynamic Double Word Line, " *IEEE Journal of Solid State Circuits*, vol. SC-19, no. 5, pp. 578–585, May 1984.

[36] M. Ohnishi, A. Yamada, H. Noda, and T. Kambe, "A Method of Redundant Clocking Detection and Power Reduction at RT Level Design, " In *Proc. of International Symposium on Low Power Electronics and Design (ISLPED'97)*, pp. 131–136, 1997.

[37] M. Pedram, "Power Minimization in IC Design: Principles and Applications, " *ACM Trans. Design Automation of Electronic Systems (TODAES)*, vol. 1, no. 1, pp. 3–56, January 1996.

[38] M. Yoshimoto, K. Anami, H. Shinohara, T. Yoshihara, H. Takagi, S. Nagao, S. Kayano, and T. Nakano, "A Divided Word-Line Structure in the Static RAM and its Application to a 64K Full CMOS RAM, " *IEEE Journal of Solid-State Circuits*, pp. 479–485, June 1983.

[39] Mark C. Johnson and Kaushik Roy, "Datapath Scheduling with Multiple Supply Voltage and Level Converters, " *ACM Trans. Design Automation of Electronic Systems (TODAES)*, vol. 2, no. 3, pp. 227–248, July 1997.

[40] Nikolaos Bellas, and Ibrahim Hajj, "Architectural and Compiler Support for Energy Reduction in the Memory Hierarchy of High Performance Microprocessors, " In *Proc. of International Symposium on Low Power Electronics and Design (ISLPED'98)*, pp. 70–75, 1998.

[41] P. Machen, M. Degrauwe, M. Van Paemel, and M. Oguey, "A Voltage Reduction Technique for Digital Systems, " In *Proc. of International Solid-State Circuits Conference (ISSCC'90)*, pp. 238–239, 1990.

[42] P. R. Panda and N. D. Dutt, "Reducing Address Bus Transitions for Low Power Memory Mapping, " In *Proc. of European Design and Test Conference (ED&TC96)*, pp. 63–67, 1996.

[43] S. Devadas and S. Malik, "A Survey of Optimization Techniques Targeting Low Power VLSI Circuits, " In *Proc. of 32nd Design Automation Conference*, pp. 242–247, 1995.

[44] S. Gray, C. Dietz, J. Eno, G. Gerosa, S. Park, and H. Sanchez, "The PowerPC 603 Microprocessor: A Low-Power Design for Portable Applications, " In *Proc. of the 39th IEEE Computer Society International Conference*, March 1994.

[45] S. Sakiyama, H. Nakahira, M. Fukuda, A. Yamamoto, M. Kinoshita, A. Matsuzawa, H. Yamamoto, Y. Kato, Y. Matsuya, S. Mutoh, H. Fukuda, Y. Nishino, and T. Sakurai, "A lean power management technique : The lowest power consumption for the given operating speed of LSIs, " In *Proc. of Symposium on VLSI Circuits*, pp. 99–100, 1997.

[46] S. Sakiyama, J. Kajiwara, M. Kinoshita, K. Satomi, K. Ohtani and A. Matsuzawa, "An On-Chip High-Efficiency and Low-Noise DC/DC Converter Using Divided Switches with Current Control Technique, " In *Proc. of International Solid-State Circuits Conference (ISSCC'99)*, pp. 156–157, 1999.

[47] T. Ishihara and H. Yasuura, "A Library Generation Technique for Low Power VLSI Design (in Japanese), " In *Proc. of IPSJ DA Symposium '98*, pp. 185–190, Jul. 1998.

[48] T. Ishihara and H. Yasuura, "Basic Experimentation on Accuracy of Power Estimation for CMOS VLSI Circuits, " In *Proc. of International Symposium on Low Power Electronics and Design (ISLPED'96)*, pp. 117–120, 1996.

[49] T. Ishihara and H. Yasuura, "Experimental Analysis of Power Estimation Models of CMOS VLSI Circuits, " *IEICE Trans. Fundamentals*, vol. E80–A, no. 3, pp. 480–486, March 1997.

[50] T. Ishihara and H. Yasuura, "Optimization of Supply Voltage Assignment for Power Reduction on Processor-Based Systems, " In *Proc. of Synthesis and System Integration of Mixed Technologies (SASIMI'97)*, pp. 51–58, 1997.

[51] T. Ishihara and H. Yasuura, "Power-Pro: Programmable Power Management Architecture, " In *Proc. of Asia South Pacific Design Automation Conference*, 1998.

[52] T. Ishihara and H. Yasuura, "Programmable Power Management Architecture for Power Reduction, " *IEICE Trans. on Electronics*, vol. E81–C, no. 9, pp. 1473–1480, September 1998.

[53] T. Ishihara and H. Yasuura, "Voltage Scheduling Problem for Dynamically Variable Voltage Processors, " In *Proc. of International Symposium on Low Power Electronics and Design (ISLPED'98)*, pp. 197–202, 1998.

[54] T. Ishihara and H. Yasuura, "A Power Optimization Technique for Application Specific Memory Designs (in Japanese), " *IEICE Technical Report,* VLD98–141, ICD98-287, March 1999.

[55] T. Ishihara, K. Hirose, K. Shiomi, M. Sugihara, and H. Yasuura, "A Multiplier Chip Design for an Analysis of Multiplication Algorithms (in Japanese), " In *Record of Joint Conference of Electrical and Electronics Engineers in Kyushu*, October 1997.

[56] T. Okuma, T. Ishihara, and H. Yasuura, "Real-Time Task Scheduling for Variable Voltage Processor (in Japanese), " *IEICE Technical Report,* VLD98–129, CPSY98-149, December 1998.

[57] T. Sakurai and T. Iizuka, "Double Word Line and Bit Line Structure for VLSI RAMs — Reduction of Word Line and Bit Line Delay —, " In *Extended Abstracts of the 15th Conference on Solid State Devices and Materials*, pp. 269–272, 1983.

[58] T. Sakurai and T. Kuroda, "Low-Power Circuit Design for Multimedia CMOS VLSIs, " In *Proc. of 6th Workshop on Synthesis and Systems Integration of Mixed Technologies (SASIMI'96)*, pp. 3–9, 1996.

[59] T. Utino, H. Minami, and N. Goto, "Switching Activity Analysis using Boolean Approximation Method, " In *Proc. of International Conference on Computer Aided Design (ICCAD'95)*, pp. 20–25, 1995.

[60] V. Gutnik and A. P. Chandrakasan, "Embedded Power Supply for Low-Power DSP, " *IEEE Trans. VLSI Systems*, vol. 5, no. 4, pp. 425–435, December 1997.

[61] T. Xanthopoulos and A. Chandrakasan, "A Low-Power DCT Core Using Adaptive Bitwidth and Arithmetic Activity Exploiting Signal Correlations and Quantization, " In *Proc. of Symposium on VLSI Circuits*, June 1999.

[62] Y.-R. Lin, T.-T. Hwang, and A. C.-H. Wu, "Scheduling Techniques for Variable Voltage Low Power Designs, " *ACM Trans. Design Automation of Electronic Systems (TODAES)*, vol. 2, no. 2, pp. 81–97, April 1997.

[63] Y. Shin and K. Choi, "Power Conscious Fixed Priority Scheduling for Hard Real-Time Systems, " In *Proc. of 36th Design Automation Conference*, pp. 134–139, June 1999.

[64] Y.-T. S. Li, S. Malik, and A. Wolfe, "Performance Estimation of Embedded Software with Instruction Cache Modeling, " In *Proc. of International Conference on Computer Aided Design (ICCAD'95)*, pp. 380–387, 1995.

[65] Y. Yoshida, B. Y. Song, H. Okuhata, T. Onoye, and I. Shirakawa , ""An Object Code Compression Approach to Embedded Processors", " In *Proc. of Int'l Symposium on Low Power Electronics and Design*, pp. 265–268, Aug. 1997.

List of Publications by the Author

Journal Publications

[J-1] T. Ishihara, and H. Yasuura, "Experimental Analysis of Power Estimation Models of CMOS VLSI Circuits," *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences*, vol. E80–A, no. 3, March 1997.

[J-2] T. Ishihara, and H. Yasuura, "Programmable Power Management Architecture for Power Reduction," *IEICE Trans. Electronics*, vol. E81–C, no. 9, September 1998.

[J-3] H. Tomiyama, T. Ishihara, A. Inoue and H. Yasuura, "Instruction Scheduling to Reduce Switching Activity of Off-Chip Buses for Low-Power Systems with Caches," *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences*, vol. E81–A, no. 12, December 1998.

[J-4] T. Ishihara and H. Yasuura, "A Memory Power Optimization Technique for Application Specific Embedded Systems," *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences*, vol. E82-A, no. 11, November. 1999.

[J-5] H. Yasuura and T. Ishihara, "System LSI Design Methods for Low Power LSIs," To be appeared in *IEICE Trans. Electronics*, vol. E83-C, no. 2, February. 2000.

[J-6] K. Inoue, T. Ishihara, and K. Murakami, "A High-Performance and Low-Power Cache Architecture with Speculative Way-Selection," To be appeared in *IEICE Trans. Electronics*, vol. E83-C, no. 2, February. 2000.

International Conference Publications

[C-1] H. Akaboshi, T. Ishihara, and H. Yasuura, "An Approximate Estimation of Power for

Processor Architecture Design," In *Proc. of 3rd Asia Pacific Conference on Hardware Description Languages, (APCHDL'96)*, pp. 23–27, Bangalore, India, January 1996.

[C-2] T. Ishihara, and H. Yasuura, "Basic Experimentation on Accuracy of Power Estimation for CMOS VLSI Circuits," In *Proc. of International Symposium on Low Power Electronics and Design (ISLPED'96)*, pp. 117–120, Monterey, CA, USA, Oct. 1996.

[C-3] T. Ishihara, and H. Yasuura, "Optimization of Supply Voltage Assignment for Power Reduction on Processor-Based Systems," In *Proc. of 7th Workshop on Synthesis and Systems Integration of Mixed Technologies (SASIMI'97)*, pp. 51–58, Osaka, Japan, Dec. 1997.

[C-4] T. Ishihara and H. Yasuura, "Power-Pro: Programmable Power Management Architecture," In *Proc. of 3rd Asia and South Pacific Design Automation Conference 1998 (ASP-DAC'98)*, pp. 321–322, Yokohama, Japan, Feb. 1998.

[C-5] H. Tomiyama, T. Ishihara, A. Inoue, and H. Yasuura, "Instruction scheduling for power reduction in processor-based system design," In *Proc. of Design Automation and Test in Europe (DATE98)*, pp. 855–860, Paris, France, Feb. 1998.

[C-6] T. Ishihara and H. Yasuura, "Voltage Scheduling Problem for Dynamically Variable Voltage Processors," In *Proc. of International Symposium on Low Power Electronics and Design (ISLPED'98)*, pp. 197–202, Monterey, CA, USA, Aug. 1998.

[C-7] T. Ishihara, and H. Yasuura, "Power-Pro: Programmable Power Management Architecture for Power Reduction," In *Proc. of International Symposium on Future of Intellectual Integrated Electronics(ISFIIE)*, pp. 295–296, Sendai, Japan, Mar. 1999.

[C-8] K. Inoue, T. Ishihara, and K. Murakami, "Way-Predicting Set-Associative Cache for High Performance and Low Energy Consumption," In *Proc. of International Symposium on Low Power Electronics and Design (ISLPED'99)*, pp. 273–275, San Diego, CA, USA, Aug. 1999.

[C-9] H. Date, T. Ishihara, H. Yamashita, A. Hyoudou, F. N. Eko, S. Nakamura, A. Inoue, and H. Yasuura, "RSA Chip Design Based on Soft-Core Processor," In *Proc. of 6th Asia Pacific Conference on Hardware Description Languages (APCHDL'99)*, pp. 78–83, Fukuoka, Japan, Oct. 1999.

[C-10] T. Okuma, T. Ishihara, and H. Yasuura, "Real-Time Task Scheduling for a Variable Voltage Processor," To be appeared in *Proc. of 12th International Symposium on System Synthesis (ISSS'99)*, San Jose, CA, USA, Nov. 1999.

Technical Society Meeting and Domestic Conference Publications

[T-1] T. Ishihara, T. Nakagawa, and H. Yasuura, "Realization of Low-Power Pipelined Processors by Voltage Cut Down (in Japanese)," In *Record of Joint Conference of Electrical and Electronics Engineers in Kyushu*, p. 664, Sep. 1994.

[T-2] T. Ishihara, and H. Yasuura, "On Accuracy of Switch Level Power Estimation for CMOS LSI Circuits (in Japanese)," *IPSJ SIG Notes*, DA–75–4, May. 1995.

[T-3] T. Ishihara, and H. Yasuura, " On Accuracy of Switch Level Power Estimation for CMOS LSI Circuit (in Japanese)," In *Proc. of the 1995 Engineering Sciences Society Conference of IEICE*, A-54, Sep. 1995.

[T-4] T. Ishihara, and H. Yasuura, "Some Experimental Results on Low Power Design with Gated Clock (in Japanese)," In *Proc. of IPSJ SIG Notes*, ARC–115–16 and DA–78–16 Dec. 1995.

[T-5] T. Ishihara, and H. Yasuura, "Basic Experimentation on Accuracy of Power Estimation for CMOS VLSI Circuits (in Japanese)," *Proc. of 9th Karuizawa Workshop*, pp. 247–252, Apr. 1996.

[T-6] H. Tomiyama, A. Inoue, T. Ishihara, and H. Yasuura, "Instruction scheduling to minimize bus transitions for low power design (in Japanese)," In *Proc. of IPSJ DA Symposium '96*, pp. 159–164, Aug. 1996.

[T-7] T. Ishihara, and H. Yasuura, "On Relation between Area and Energy of CMOS Adder Circuits (in Japanese)," *Proc. of the 1996 Engineering Sciences Society Conference of IEICE*, SA–2–2, Sep. 1996.

[T-8] T. Ishihara, K. Kai, and H. Yasuura, "Power-Pro : Programmable Power Management Architecture (in Japanese)," *IEICE Technical Report*, VLD96–72 and CPSY96–84, Dec. 1996.

[T-9] T. Ishihara, H. Yasuura, and M. Iwaihara, "A Programmable Word Length Architecture with Gated-Clock for Power Reduction (in Japanese)," *Proc. of the 1996 IEICE General Conference*, SA–2–2, Mar. 1997.

[T-10] T. Ishihara, H. Yasuura, and M. Iwaihara, "A Programmable Word Length Architecture with Gated Clock for Power Reduction (in Japanese)," *Proc. of the 10th Karuizawa Workshop*, pp. 451–456, Apr. 1997.

[T-11] K. Shiomi, K. Okino, T. Kawasaki, T. Ishihara, and H. Yasuura, "Development of A Standard Cell Library For VDEC (in Japanese)," In *IEICE Technical Report*, VLD97–31, CAS97–31, and DSP97–46, Jun. 1997.

[T-12] T. Ishihara, K. Hirose, K. Shiomi, M. Sugihara, and H. Yasuura, "A Multiplier Chip Design for an Analysis of Multiplication Algorithms (in Japanese)," *Record of Joint Conference of Electrical and Electronics Engineers in Kyushu*, p. 5 Oct. 1997.

[T-13] T. Ishihara, and H. Yasuura, "Power Optimization with Variable Voltage Processor (in Japanese)," *Technical Report of IEICE*, CPSY97–77, *IPSJ SIG Notes*, DA–85–14, Oct. 1997.

[T-14] T. Ishihara and H. Yasuura, "Basic Theorems for Variable Voltage Low Power Processors (in Japanese)," *IEICE Technical Report*, ICD98–45 and FTS98–25, May. 1998.

[T-15] A. Inoue, H. Tomiyama, T. Ishihara and H. Yasuura, "System-Level Energy Optimization for Embedded Systems with a Variable a Variable Datapath Width Processor (in Japanese)," *IEICE Technical Report*, CAS98–29, DSP98–58, and VLD98–29, Jun. 1998.

[T-16] K. Jamaleddine, T. Ishihara, K. Shiomi, K. Okino, T. Kawasaki, and H. Yasuura, "A Standard Cell Library for VDEC Users (in Japanese)," In *Proc. of IPSJ DA Symposium '98*, pp. 185–190, Jul. 1998.

[T-17] T. Ishihara, and H. Yasuura, "Programmable Power Management Architecture for Power Reduction (in Japanese)," In *Proc. of IPSJ DA Symposium '98*, pp. 191–196, Jul. 1998.

[T-18] T. Ishihara, and H. Yasuura, "Theory of the Static Voltage Scheduling for Dynamically Variable Voltage Processors (in Japanese)," In *Proc. of IPSJ DA Symposium '98*, pp. 287–292, Jul. 1998.

[T-19] K. Inoue, T. Ishihara and K. Murakami, "A High-Performance/Low-Energy Cache Architecture with Way-Prediction Technique (in Japanese)," *IEICE Technical Report*, VLD98–44, ICD98–147, and FTS98–71, Sept. 1998.

[T-20] A. Hyoudou, T. Ishihara, and H. Yasuura, "Evaluation of Standard Cell Libraries for VDEC (in Japanese)," *IEICE Technical Report*, VLD98–45, ICD98–148, and FTS98–72, Sept. 1998.

[T-21] K. Inoue, T. Ishihara and K. Murakami, "A High-Performance Set-Associative Cache Architecture with Speculative Way-Selection (in Japanese)," *IEICE Technical Report*, DSP98–94, ICD98–181, and CYSY98–96, Oct. 1998.

[T-22] T. Okuma, T. Ishihara, and H. Yasuura, "Real-Time Task Scheduling for Variable Voltage Processor (in Japanese)," *IEICE Technical Report*, VLD98–129, CPSY98–149, Dec. 1998.

[T-23] T. Ishihara, and H. Yasuura, "A Power Optimization Technique for Application Specific Memory Designs (in Japanese)," *IEICE Technical Report*, VLD98–141, ICD98–247, Mar. 1999.

[T-24] T. Ishihara, "Standard Cell Libraries for VDEC User and it's Development Environment (in Japanese)," *Proc. of the 1999 IEICE General Conference*, PA–2–6, Mar. 1999.

[T-25] T. Ishihara, and H. Yasuura, "A Library Generation Technique for Low Power VLSI Design (in Japanese)," In *Proc. of IPSJ DA Symposium '99*, pp. 231–236, Jul. 1999.