BLOCK LEVEL VOLTAGE/FREQUENCY SCHEDULING FOR LOW POWER CONSUMPTION UNDER TIMING CONSTRAINTS

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR
THE DEGREE OF RESEARCH MASTER
in the School
of
Electronic Engineering

by
Li-Chuan Weng
B.Sc. ECE, National Chiao-Tung University, Taiwan
September 2004

BLOCK LEVEL VOLTAGE/FREQUENCY SCHEDULING FOR LOW POWER CONSUMPTION UNDER TIMING CONSTRAINTS

Dissertation of Research Master Degree
School of Electronic Engineering
Dublin City University
Ireland

by
Li-Chuan Weng

September 2004

Declaration

I hereby certify that this material, which I now submit for assessment on the program of study leading to the award of Research Master Degree of Electronic Engineering is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed : [illegible]

ID No.: 51161974

Date : 2004. 9.16.

To my parents

Acknowledgements

This thesis would not have been possible without my advisors, Dr. XiaoJun Wang and Dr. Alan P. Su. Many thanks to Dr. Wang for his inspiration and the time and patience spent in reading this thesis, thanks to Dr. Su for the discussions regarding to my research. Dr. Wang allowing me the freedom to work on topics that I enjoyed. I would like to acknowledge the help of Dr. Valentin Muresan with the contribution towards this thesis. Special thanks to my undergraduate mentors in ECE NCTU, Taiwan, including, but not limited to, Professor B.-F. Wu, D.-C. Liaw, K.-T. Song and to K.-Y. Yang. They are really kind teachers whom I am deeply indebted.

During these years of Master research, I had the privilege of working in two environments, that both provided a pleasant and friendly place. The journey of this thesis began while I was in DCU, Ireland and continued after I started doing research in ITRI, Taiwan. Many thanks to all the technicians and faculties in EEng, DCU where I started my study and the colleagues at ITRI, where I finally completed this thesis.

This thesis could never be finished without the help and contributions of many friends and colleagues, therefore, I would like to especially thank my friends from everywhere in the world.

The friends I made while in Dublin were good companion and continue to bring joy into my life, these people include but not limited to Dr. J. Kanatharana, E. Dautbegovic and E. Aviotte, who are like my sisters, the big Romanian community in EEng DCU, the basketball and volleyball sport group in DCU, friends from various corner around the world, and the overseas Taiwanese society in Ireland. I would never forget the encouragement friendship from them. They made my life full of pleasure throughout the years spent in Dublin.

There is a strong support from my friends back home during my abroad study. The encouragement from them sustained me during the hard moments. Among them, I wish to express my appreciation to W.-P. Chen, C.-Y. Pan, C.-M. Wu. and Y.-H. Wu. They believed in me and provided support to me during the whole journey.

Last but not least, I want to express my gratitude to my family. I am deeply in debt to them. My deepest gratitude and love to my parents, without my mother's encouragement I won't be able to go through all the procedure. Thanks to my brothers Chia-Yang and Chia-Chun who support the family while I was abroad. At the end, I would like to dedicate this honor to my father, who was always nice to people but does not have chance to see this moment personally. His spirit always lives in my mind, now and forever.

It is impossible to list every single person who directly or indirectly company and help me to go through there years. I am very grateful to all the people who influenced my life during my past life and I do appreciate for your company in the rest journey. Thank you all.

- This work was supported in part by Ministry of Economic Affairs, Taiwan under project code A331XS1YR0, *Power Aware SoC Using Dynamic Voltage/ Frequency Scheduling and Minimal Leakage Current input.*

Abstract

Over the past years, state-of-art power optimization methods move towards higher abstraction levels that result in more efficient power savings. Among existing power optimization approaches, dynamic power management (DPM) is considered to be one of the most effective strategies. Depending on abstraction levels, DPM can be implemented in different formats but here we focus on scheduling that is more suitable for real-time system design use. This differs from the concurrent scheduling approaches that start from either the HLS (High-Level Synthesis) or RTS (Real-Time System) point of view, we propose a synergy solution of both approaches, namely block-level voltage/frequency scheduling (BLVFS). The presented block-level voltage/frequency scheduling approach shows a generic solution for low power SoC (System on Chip) system design while the approaches which belong to the HLS and RTS categories have a strong dependency on the system functionalities. Consider a SoC as a combination of heterogeneous functional blocks, our approach provides efficient power savings by dynamically scheduling the scaling of voltage and frequency at the same time. Simulation results indicate that by using heuristic based strategies significant power savings can be achieved.

Keywords: *dynamic power management, scheduling, voltage/frequency scaling*

Contents

List of Tables

List of Figures

List of Abbreviations

ACPI	:	Advanced Configuration and Power Interface
ALAP	:	As Late As Possible
ASAP	:	As Soon As Possible
BLS	:	Block Level Dynamic Voltage/Frequency Scaling Algorithm
BLVFS	:	Block Level Voltage/Frequency Scheduling
CAD	:	Computer Aided Design
CFG	:	Control Flow Graph
CDFG	:	Control/Data Flow Graph
CMOS	:	Complementary Metal-Oxide-Semiconductor
CVS	:	Clustered Voltage Scaling
DFG	:	Data Flow Graph
DFS	:	Dynamic Frequency Scaling
DP	:	Dynamic Programming
DPM	:	Dynamic Power Management
DVFS	:	Dynamic Voltage/Frequency Scheduling
DVS	:	Dynamic Voltage Scaling
EDF	:	Earliest Deadline First
GALS	:	Globally Asynchronous Locally Synchronous
HDD	:	Hard Drive Disk
HLS	:	High-Level Synthesis
ILP	:	Integer Linear Programming
IP	:	Intellectual Properties
LEA	:	Left-Edge Algorithm
NP	:	Nondeterministic Polynomial-time
OS	:	Operating System
RM	:	Rate Monotonic
RMS	:	Root Mean Square
RT	:	Real-Time
RTL	:	Register-Transfer-Level
RTOS	:	Real-Time Operating System
RTS	:	Real Time System
SoC	:	System On Chip
VLSI	:	Very Large Scale Integration

List of Symbols

α	:	a process dependent parameter varies between 1 and 2, used for circuit propagation delay
δ	:	propagation delay
τ	:	a set of operation period
τ_i	:	operation period i
a	:	voltage factor
b	:	frequency factor
B	:	set of functional blocks
B_j	:	functional block j
C_j	:	average (effective) capacitor load of the j_{th} functional block
C_{out}	:	load capacitance
E	:	energy
E_{ij}	:	the energy consumption of block j during operation period i
E'_{ij}	:	the energy consumption of block j during operation period i after scheduling
E_{sd}	:	shutdown energy consumption
E_{total}	:	total energy consumption for the system during all operation periods
E_{wu}	:	wake up energy consumption
f	:	working frequency
f'	:	new working frequency
f_{ij}	:	the working frequency of block j during operation period i
F^k	:	frequency setting at working mode k
G_j	:	transistor number of functional block j
$I_{leakage}$	:	leakage current
k	:	a process dependent constant, used for circuit propagation delay
L_i	:	usage list of operation period i
M_i	:	working mode k
N	:	node transition activity
N_j	:	nod transition activity of functional block j
P	:	power consumption
P_{avg}	:	average power consumption
$P_{dynamic}$	:	dynamic power consumption
P_i	:	power consumption of operation period i
P_{ij}	:	power consumption of block j during operation period i
$P_{leakage}$	:	due to the leakage current

P_{max}	:	the maximum power consumption over all system operations
P_{short}	:	power consumption caused by short current
P_{static}	:	static power consumption
$P_{switching}$	:	the switching component of power
P_{short}	:	due to the direct-path short circuit current
P_{rms}	:	root mean square power
P_s	:	sleeping power
P_w	:	working power
t	:	time
t_i	:	operation period i
t_{ij}	:	execution time of block j during operation period i
t_{total}	:	total execution time of a set of operation periods
T_{be}	:	break-even time
T_{sd}	:	shutdown delay
T_{to}	:	fixed timeout
T_{wu}	:	wake up delay
V_{dd}	:	supply voltage
$V_{dd'}$	:	new supply voltage
V_{ij}	:	working voltage of block j during operation period i
V_k	:	voltage at working mode k
V_t	:	threshold voltage

Chapter 1

Introduction

1.1 Motivation

Nowadays, one of the most important considerations for VLSI/SoC designs is low power. It is especially so for mobile platforms, not only for longer battery life but also for solving heat dissipation problems. Similar to other factors like area, performance, and reliability, the power issue should be taken into consideration at early design stages, i.e. at high levels, in order to achieve better efficiency. The work presented focuses on power minimization at system level by scheduling voltage and frequency scaling based on dynamic power management (DPM) policies.

1.2 Thesis Scope

Low power system design methods include power estimation as well as power minimization. Power minimization can be performed at various levels but is generally more effective at higher abstraction levels [CIB01]. Here we introduce some related survey methods of both power estimation and power minimization. This investigation emphasizes the power minimization which is the wider scope of this thesis.

1.2.1 High-level Synthesis for Low Power

A *high-level synthesis* (HLS) system (also called, *behavioral synthesis*) takes a behavioral description of a design and produces a register-transfer-level (RTL) implementation of the design [BM99]. The synthesis process includes aspects such as *scheduling*, *resource allocation*, and *binding*. There are various approaches used in high-level synthesis for low power. Some people use *integer linear programming* (ILP) [JR97a] to schedule the voltage applied on a datapath in order to minimize power while others optimize transformation techniques like loop unrolling, pipelining, and retiming to achieve higher concurrency of a design in order to get a better architecture for lower power consumption [CB95].

1.2.2 Dynamic Power Management

Dynamic Power Management (DPM) is considered to be the most powerful approach among the techniques for low-power design of integrated circuits and systems [BM98]. As a first glance, the fundamental idea of DPM is based on the fact that the workload of a system is not constant, and the DPM approach focuses on the utilization of the slack time that is caused by the non-uniform workload [PR02, Wol02]. Unlike many existing approaches, DPM is actually an abstract concept that is independent of technology and can be applied at different abstraction levels. DPM research highlights the application of state transitions and tries to predict the system's working status in order to further reduce power dissipation [BBM00].

1.2.3 Dynamic Voltage/Frequency Scheduling

Status scheduling, i.e. to scale supply voltage and clock frequency according to available slack time, is an implementation of DPM. Putting a system into power down mode while idle is another way of implementing DPM. The characteristics of logic circuits are more suitable for dynamic voltage and frequency scheduling than power down and provides better results of power minimization [HPK99]. The methodology uses the scheduling of voltage and frequency scaling to implement DPM. This thesis presents a heuristic based methodology that reduces power consumption in SoC through the control of voltage/frequency scaling.

1.3 Thesis Structure

This chapter has briefly introduced the thesis scope while the remainder of this thesis is organized as follows. Chapter 2 describes the theoretical background, and covers the the sources of power consumption in Complementary Metal-Oxide-Semiconductor (CMOS) circuits. It is followed by descriptions of current power optimization methodologies, high-level synthesis for low power, with emphasis on contemporary dynamic power management methodologies. Chapter 3 describes the problem definition and the methodologies investigated here. This is followed by Chapter 4 where detailed algorithm and experimental results are presented. Finally, Chapter 5 concludes the thesis with a summary of the key contributions and the future research directions.

Chapter 2

Theoretical Background

Chapter 1 explained the motivations of power-aware design, and this chapter provides the related background. First, the related concepts and terminologies of low power CMOS VLSI circuits used throughout the thesis are introduced. Then, some of the relevant research fields including modern power optimization techniques, high-level synthesis for low power and dynamic power management are covered.

2.1 Preliminaries and Infrastructures

In this section, we focus on power-related background that offers preliminaries for the rest of this thesis. We first discuss the elements which cause power consumption in CMOS circuits, then we explain some terminologies used in the field of low power design that are often confusing. Finally, we briefly discuss some power optimization techniques.

2.1.1 Sources of CMOS Power Consumption

The starting point for low-power design is to understand where the power is consumed in a target system. Since CMOS devices are the most important cell elements in modern digital systems, we will examine the related contributing factors of power consumption in CMOS systems. Generally speaking, the average power dissipation on CMOS devices can be decomposed into two parts as follows [CB95, Ped96, RP96, BM98, RP00].

$$P_{avg} = P_{dynamic} + P_{static} \tag{2.1}$$

From Equation 2.1, the two components of average power consumption are *dynamic power* ($P_{dynamic}$) and *static power* (P_{static}). As implied by the name, static power is independent of the circuit activities but dependent on technology process. P_{static} is mainly caused by the *leakage currents* ($I_{leakage}$) and occurs as long as power is supplied to the CMOS device. Since the factors that influence P_{static} are determined at manufacture, system designers have very little control of them, plus P_{static} is relatively small compared to $P_{dynamic}$, therefore, this thesis concentrates only on the minimization of $P_{dynamic}$. It is worth noting that this assumption of P_{static} is much smaller than $P_{dynamic}$ will change with the shrinking of feature size. It is predicated that P_{static} will be equal in proportion to $P_{dynamic}$ at about 45 $nano - meter$ scale technology.

The most significant source of CMOS power consumption is $P_{dynamic}$, it is dissipated as the circuit switches states. As shown in Equation 2.2, $P_{dynamic}$ contains *short-circuit power* (P_{short}) and *switching power* ($P_{switching}$). Among the two elements, $P_{dynamic}$ is subject to $P_{switching}$ resulting from charging and discharging the capacitive load (C_{out}) at the output of the gate.

$$P_{dynamic} = P_{short} + P_{switching} \tag{2.2}$$

Here we emphasize on the minimization of $P_{switching}$ that dominates over 90% of the total power dissipation and can be expressed by Equation 2.3:

$$P_{swithcing} = N \cdot C_{out} \cdot V_{dd}^2 \cdot f \tag{2.3}$$

There are four parameters in the equation above that can be considered to change [KL99], i.e. the node transition activity factor (N), load capacitance (C_{out}), supply voltage (V_{dd}), and working frequency (f). Equation 2.3 demonstrates that switching power consumption decreases quadratically with the decrease of V_{dd} and decreases linearly with the decrease of the other parameters. These parameters are targeted for power optimization.

From the above discussions, we can further approximate the power dissipated on a CMOS circuit as Equation 2.4.

$$P_{avg} \approx P_{dynamic} \approx P_{swithcing} = N \cdot C_{out} \cdot V_{dd}^2 \cdot f \tag{2.4}$$

Among the four parameters, N and C_{out} are mainly relevant at low levels of design, since C_{out} is highly dependent on the process technology and N is strongly dependent on the circuit activities. On the other hand, many approaches targeting V_{dd} and f focus on high level applications. The reason is that voltage and frequency can be controlled at higher levels although the implementation of the control mechanism also needs support from lower levels too. In order to exploit the fact that the voltage source has a quadratic influence on power consumption, the two most commonly used approaches are shutdown and scaling of supply voltage for power saving. A few frequency scaling approaches such as clock-gating or lowering frequency can also be found in the literature [RVB98, PCP$^+$02]. The details regarding these approaches are explained in the following section. In this thesis, we intend to lower the power consumption by scaling down voltage and frequency at the same time.

2.1.2 Low Power Design Terminologies

It is impossible to bring all aspects of power related design issues together in a single document. To avoid confusion, some concepts and terminologies that are usually used in the power-aware design field are listed here. Some of them are often used interchangeably but the minor differences between the definitions should be noticed.

Power/Energy. From Equation 2.5 and Figure 2.1 we can see that energy is the integral of power consumption over period of time, i.e. power is the curve and energy is the area underneath the curve. Compared to power, energy has a temporal aspect that should be taken into consideration. In other words, even though power equals energy divided by time, the efficiency of power saving might sometimes be different from energy saving. Take battery-powered applications for example, battery life is the main concern, therefore, energy is the focus rather than power in such systems. It might be possible to lower the average power consumption but extended execution time of a design, which could actually increase rather than decrease total energy consumption. The energy used purely to execute a task should be the same but extended execution time may mean more static power consumption and more leakage power.

$$E = \int_0^t P(t)dt \tag{2.5}$$

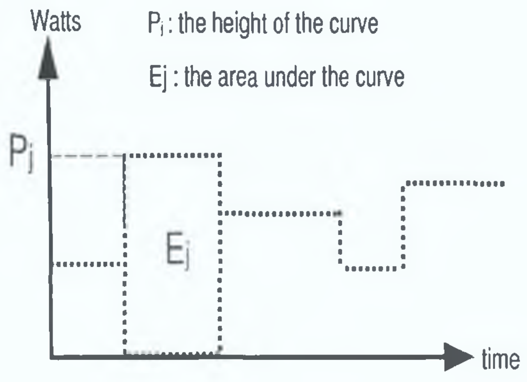


Figure 2.1: *Energy and power comparison*

Average Power v.s. Maximum/Peak Power. The average and maximum/peak power are related but not the same. Maximum power deals with the worst case of a system while average power discusses the general situation. Based on the definition, it is trivial to see there is a difference between average power and maximum/peak power consumption in a system [MK95]. In practice, there exists cases that lowering maximum/peak power can cause the increasing of average power. The emphasis of this research is on the lowering of average power dissipation.

Power-aware Design v.s. Low Power Design. Low power design directly implies minimizing power/energy under certain constraints like performance. On the other hand, power aware design does not look for the lowest power but makes a tradeoff between the power and performance and may sometimes increase the power consumption instead of decreasing it [PR02, UK03]. In other words, power-aware design may actually increase average power consumption in order to lower peak power consumption.

Power Estimation/Simulation. Power related research can be classified as estimation/evalutaion and optimization. Power estimation and evaluation for systems is important during system design as it helps designers to meet the power requirements of the system specification. Further information can be found in the literature [MPS97, FGSS98, CIB01].

Power Dissipation/Consumption. Power dissipation is defined as the amount of energy dissipated, i.e. converted into another form of energy. On the other hand, power consumption implies the consumed energy from a power supply in unit time [PR02]. While power consumption relates to power delivery costs, performance and reliability, power dissipation affects more parameters including performance, packaging, reliability, environmental impacts and heat removal cost.

2.1.3 Power Optimization Techniques

A number of power optimization techniques have been proposed. Here we classify the research on power optimization along two axes, one is based on the target types and the other is according to the abstraction layers of logic circuit design.

Miscellaneous Approach Types

The mainstream of power-related research is focused on CMOS circuits while there are miscellaneous ways to explore the system utilization. For instance, some research groups concentrate on battery models including battery capacity, discharge current etc. The domain of low power design is boundless, however, in this thesis we confine our core discussion to synchronous CMOS circuits. More specifically, we focus on a generic synchronous CMOS system, which is different from research that deals with specific circuits types, e.g. memory circuits, and asynchronous circuits. We concentrate on dynamic power only while some researchers concerned with other elements that cause power consumption like *leakage current* and *glitches* etc.

In widely used synchronous circuits, clock power can be a big portion of the overall power dissipation. On the other hand, asynchronous logic or self-timed circuits use a handshaking scheme which does not need the clocks that usually consume significant power in the system. Therefore, some people work on asynchronous designs that operate without an externally supplied clock as in [NCNvB94], the most dramatic self-timed example would be the research on AMULET processor cores [FEG+01]. Nevertheless, due to incompatibility with the current synchronous designs, this approach has not been widely discussed. There are approaches that mix synchronous with asynchronous design methodologies so as to take advantages of both sides, e.g. *globally asynchronous locally synchronous (GALS)*. These mixed design methodologies supposedly have great potential [HMK+99]. Low power design covers a wide range of hardware and software design abstractions, discussions in the next paragraph focus on the techniques applied to the common synchronous circuits.

Low Power Synchronous Circuits

Optimization methods on CMOS circuits can be classified according to the design abstraction levels they are applied to, namely the *layout level*, *device level*, *circuit level*, *logic level*, *architectural level*, *algorithmic level*, and *system level*. Techniques applied at lower levels include dual supply voltage designs [UNI+97, CS99], multi-threshold voltage, adiabatic circuits etc. The approaches at lower levels are more mature, but power optimizations should be addressed at both higher and lower levels in order to achieve better results. Comprehensive overviews on power optimization can be found in survey papers [RJD98, BdM00, BBM00, Ped01]. Detailed discussions about power dissipation at device level can be referred to [CPM+95, CB95]. Techniques applied at higher abstraction levels need to be addressed in order to efficiently manage power. Various approaches at system level have been used, including low-power module selection, power-conscious storage allocation and data mapping solutions, algebraic transformations for low power, and operator shutdown etc. In addition, there are researches on power-efficient compilation approaches. System-level power optimization promises the most efficient approaches to low power design.

Approaches above System-level

The higher the abstraction level targeted the higher the efficiency. Currently, there are attempts to manage the scheduling at operating system (OS) level or above, especially for real-time embedded systems. Some research is aimed at higher levels such as OS-level [Lu01, FM02], application level, and network layer, however, power optimization techniques at these levels are still at their infancy and more research is required. A complete classification of system level low power techniques for real-time systems can be found in [UK03]. Meanwhile, industrial standards like OnNow [Mic97] and *Advanced Configuration and Power Interface (ACPI)* [HPIM+] have been proposed to facilitate extreme high level power management. Combined with the current real-time system scheduling schemes, the full utilization of system level scheduling in low power applications can be expected soon.

2.2 Low Power System Synthesis Techniques

System synthesis (also known as *behavioral synthesis*) refers to the process of mapping a high-level specification of a design into an RTL implementation [GVNG94]. Numerous researches on behavioral synthesis for low power have been conducted. In this section, we briefly review some of the most relevant contributions to the field of low power system synthesis with the main focus on scheduling methods.

The input of system-synthesis is some typical design representation in the form of *data-flow graph (DFG)*, *control-flow graph (CFG)* or *control/data flow graph (CDFG)*. DFG represents the essential ordering of operations in the program imposed by the data relations in the specification, CFG is derived directly from the explicit order given in the program and from the compiler's choice of how to parse the arithmetic expressions, and CDFG is a heterogeneous model combined the CFG with DFG that can represent both the data dependence and the control sequence of a system in a single representation. The primary system synthesis steps are *transformation*, *scheduling*, *allocation*, and *binding* [Wol02]. Transformations for low power can be achieved by reducing the number of operators and thus the number of functional units. Examples of applying transformations to the DFG of an algorithm that reduce the power consumption can be found in [CPM+95]. Allocation and binding carry out the selection and assignment processes of hardware resources for a given design. The aforementioned low power approaches like low power module selection and sharing belong to this category. We do not intend to describe the procedures in detail but more discussion can be found in the literature [Mic94].

2.3 DPM Methodology Based Approaches

Modelling and optimizing power at higher levels of abstraction are needed for power-aware system designs. The reason that DPM can save power is based on the hypothesis that most systems are designed for peak use. The DPM strategy transits components into low-power states when they are idle in order to reduce energy. Basically, DPM is a powerful methodology that can be applied at several levels of abstraction. [BBM00] is a survey of DPM related approaches. In other words, DPM strategies make a trade-off between power and performance. In this section, we focus on DPM methodology based scheduling approaches.

According to the policy characteristics, current DPM can be classified into *heuristic* and *stochastic* policies. Each category has its pros and cons yet there is no final conclusion regarding which is superior. In short, heuristic policies are easier to implement while stochastic policies guarantee optimized solution but are more complex to implement. The details of these strategies can be referred to in [QP99, CBM99, SBM01, Sim01] etc.

There are numerous ways to implement DPM methodology. To make use of the fact that energy consumption in CMOS circuits is quadratically proportional to its supply voltage, voltage shutdown and voltage scaling approaches are conceived. In short, voltage shutdown policies decide whether to switch on or switch off the supply voltage while voltage scaling strategies try to find a suitable voltage level for system/component's operation. The former ensures the power saving but has the drawback of losing performance due to the requirement of time to wake up. The latter is more suitable for real-time systems which have stringent execution-timing requirements. It has been illustrated in [HPK99] that voltage scaling is superior to shutdown. The following investigation clarifies both power down and scaling approaches and indicates the significance of efficient DPM-centric scheduling approach.

2.3.1 Power Down Related Approaches

To turn off power of an idle component is a radical way of thinking. Several mechanisms (e.g. *supply voltage, V_{dd}* and *working frequency, f*) have been used for shutting down a device. Obviously, the most effective way to save power in idle modules is to shut down supply voltage, which reduces power dissipation to zero, hence shutdown has often been used in DPM methodology. It is worth emphasizing that power down does not imply only voltage shutdown that is the most commonly used approach but also includes other approaches such as clock gating, i.e. to gate the clock in order to freeze the clock of the idle device at RTL. The shutdown approach was firstly applied on mechanical devices like hard drive disks (HDD) [GBS+95] and was adopted in computer systems and web servers. Since shutdown is the most common approach in power saving, systems or devices using the shutdown method are numerous.

According to the policies characteristics, shutdown methods can be categorized as *static* and *dynamic* ones. Static shutdown implies that the system does not change its prediction of timeout period. A typical example of static shutdown is a fixed timeout policy, in which components are transferred into power-saving mode after a certain threshold idle time. On the other hand, dynamic shutdown methods will try to predict the idle period, and there is no fixed waiting time before transferring idle components into power saving mode. A few shutdown approaches have been investigated and can be referred to [SCS00].

The traditional shutdown approach turns off the power supply of the idle components and eliminates waste of power. Although shutdown does save power consumption during the shutdown period, it also incurs performance and power cost for state transitions. Some researches dynamically shutdown the idle components but such an approach is accompanied with penalties as well. This overhead occurred because of the need of power and time for state transitions, i.e. the transitions from working state to shutdown state and vice versa. In order to explain whether shutdown is worthwhile, a concept called *break-even time* is used, i.e. the minimum length of idle time to achieve power saving [LM01], is defined as follows.

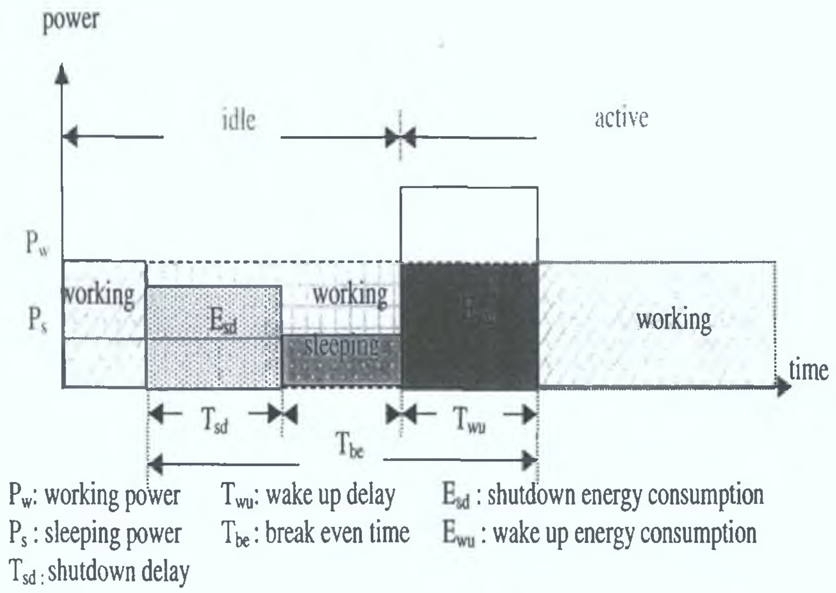


Figure 2.2: Break-even time deduction

$$\begin{aligned} P_w \cdot t &\geq E_{sd} + E_{wu} + P_s \cdot (t - T_{sd} - T_{wu}) \\ t &\geq \frac{E_{sd} + E_{wu} - P_s \cdot (T_{sd} + T_{wu})}{P_w - P_s} \\ T_{be} &= \frac{E_{sd} + E_{wu} - P_s \cdot (T_{sd} + T_{wu})}{P_w - P_s} \end{aligned} \tag{2.6}$$

If we do not consider the drawback of wake up delay but only consider from the energy point of view, the system should shutdown only when idle period t is longer than T_{be}. Figure 2.2 and Equation 2.6 explain the term T_{be}. To sum up, the shutdown approaches should be applied only when the performance and power penalty is acceptable by using breakeven times as a metric to determine when to switch power states. A system should only be shutdown when its idle period $t \geq T_{be}$. For the same idle time, a longer break-even time T_{be} means less power saving due to the overhead for recovering state [BBM00]. Some approaches attempt to predict the fluctuations in components' workload in order to lower the waiting time caused by shutdown [HW97].

2.3.2 Scheduling Approach at System Synthesis

In this section we consider the application of system-level scheduling on low power designs. During the system synthesis process, scheduling determines the concurrency of the resulting implementation and affects system performance [Mic94]. Through scheduling, a sequencing graph developed from an algorithm's operand can determine the precise start time of each task while satisfying the original dependencies. The typical system-level scheduling problems can be modelled either unconstrained, such as the *As Soon As Possible (ASAP) algorithm*; latency-constrained (the time available to execute the operations), such as the *As Late As Possible (ALAP) algorithm*; resource-constrained (the number of resources for each type of operation, e.g. adder, multiplier etc.) such as *List Scheduling*; or both resource and timing constrained scheduling, such as *force-directed list scheduling.* More detailed information on the conventional scheduling approaches may be found in reference [Mic94]. A number of researchers have developed systems or proposed methods that utilize *high-level synthesis* (HLS) for low power [JR97b, LHW97, HPK99, MC02, CIC$^+$03] via scheduling, we survey some of them and explain the need for new strategies. Further review of existing approaches on HLS for low power can be referred to related literatures [Jha01].

A number of algorithms using variable voltage settings with behavioral synthesis and datapath scheduling frequently appear in recent research [CP97, JR97a, SC98, HPK99, MC02, CIC$^+$03]. These multiple voltage synthesis algorithms have some things in common. In short, they all attempt to minimize energy consumption through the use of multiple voltages, many of them use DFG as algorithm input, and schedule different voltage levels on functional units such as adders, multipliers, registers etc. based on the infrastructural scheduling mechanisms including ASAP, ALAP and List Scheduling [RS95, LHW97, CP97, SC98, MC02]. Compared to the traditional HLS scheduling, there is an additional set of resource constraints in low power oriented HLS, that is, the corresponding operating voltages that can be used.

One of the first research that uses HLS scheduling for low power is [RS95], which schedules variable supply voltage on the datapath operators under timing constraints in order to minimize power. Both [JR97a] and [CP96] propose approaches to schedule working voltages of datapath operations from a specified list of candidate voltages to optimize power but use *integer linear programming (ILP)* and *dynamic programming (DP)* respectively. All the HLS approaches have the potential drawback that they have to rely on the system design, in other words, the mechanism they offer is limited by the example circuits or the benchmark suite they used (e.g. EW Filter, AR Filter, DIFF EQ benchmarks are used by [CP97, SC98], [CIC$^+$03] use ISCA89 benchmark suite). Another shortfall is that, these approaches are static and can not be applied dynamically hence they can not achieve the best optimization possible as it is difficult for the designer to foresee the variations of system operations. These approaches are therefore not generic as they can not be applied to complex hard real-time systems. Based on this thinking, a new strategy of operation period scheduling will be proposed in the next chapter.

2.3.3 Real-Time System Scheduling for Low Power

In real-time systems, the OS is in charge of scheduling programs according to their priorities. Scheduling algorithms guarantee timeliness of the task deadline. Typical real-time operating system (RTOS) scheduling approaches include Rate Monotonic (RM), Earliest Deadline First (EDF) techniques etc. The scheduling approaches can be classified as *static scheduling* and *dynamic scheduling*. The former one implies a fixed schedule which is determined at compile-time and hence it does not change at run time. The latter means the execution sequence is controlled or determined at run-time, or say, on-line.

As many embedded systems are real-time in nature, there has been some research working on different approaches using current real-time system scheduling techniques to further optimize power [HPS98, SC99, QWP00, Gru02]. Such DPM approaches applied in RT scheduling for reducing energy consumption can be classified into CPU-centric and I/O centric classes. Most research [IY98, MC01, MC02, LXC03, YK03] belong to the first category and a few [SCI01, SC03] investigate the latter. The *Dynamic Voltage Scaling* (DVS) schemes that use existing RT scheduling to minimize the energy consumption while maintaining acceptable performance are processor-centric approaches. [YK03] has proved that optimal voltage scheduling problem is NP-hard. Generally speaking, these approaches improve an existing task schedule with voltage and frequency scaling based on different modelling formulation and problem solving techniques (e.g. ILP).

2.4 Voltage and Frequency Scaling Mechanism

For the sake of convenience, we repeat Equation 2.4 below, it is observed that reduction of V_{dd} and/or f can save substantial power. In practical designs, voltage and frequency are closely related parameters and it should be possible to tackle both together for power saving purposes. The following discussions contain the current DVS and DFS (*dynamic frequency scaling*) mechanisms that are used as bases in this thesis.

$$P_{avg} \approx N \cdot C_{out} \cdot V_{dd}^2 \cdot f \tag{2.7}$$

2.4.1 Voltage Scaling

As mentioned before, supply voltage V_{dd} is the most dominant factor that affects the power consumption of VLSI systems. An intuitive thinking is to lower the system supply voltage. However, aimlessly decreasing the supply voltage also increases the circuit delay, δ, given by Equation 2.8.

$$\delta = \frac{C_{out} \cdot V_{dd}}{k \cdot (V_{dd} - V_t)^{\alpha}} \tag{2.8}$$

where k is a process dependent constant, V_t is the threshold voltage, and α is another process-dependent parameter which varies between 1 and 2. From Equation 2.8, the reason we can not simply lower supply voltage is, when the supply voltage is reduced to close to threshold voltage, the performance goals may not be met, even if we lower the threshold voltage at the meanwhile, it may make noise margin and $I_{leakage}$ increase hence P_{static} increases. Also, the lowering of V_{dd} requires the decreasing of V_t which needs the support of the technology process and is beyond the control of system designers. In short, the smaller the V_{dd}, the higher the delay, so we have to make a tradeoff between power saving and the increased circuit delay when voltage scaling is applied.

Based on the above discussion, we can further consider the relation of supply voltage and system performance. Most systems are over-designed in the way that system voltage is provided for maximum use. It is only natural to think of applying different voltage supply for different uses. High supply voltage implies high execution speed and is used for tasks with tight real-time constraints. On the other hand, lower supply voltage means lower execution speed and can be applied to tasks with loose time constraints.

DVS is a technique that promises substantial power reduction but we need to do smarter voltage management because simply lowering supply voltage also causes an increase of the delay. Most voltage scaling approaches require that the whole IC operates at the same supply voltage at the same time although the supply voltage could be different at different times, but DVS intends to scale the supply voltage according to the workload and get the benefit from it.

There is some research targeted at register-transfer-level (RTL) implementation with voltage scaling such as MOVER [JR97a], a tool which reduces the energy dissipation using datapath scheduling among multiple supply voltages. Some low level techniques like the Clustered Voltage Scaling (CVS) scheme uses two supply voltages V_{ddH} and V_{ddL} and the combination with Row by Row optimized Power supply scheme (RRPS) etc. [UH95, UNI+97, IUN+97]. In this thesis, we propose an original scheme that applies voltage scaling in the system, and an embedded selector in each functional block to achieve more efficient power savings.

2.4.2 Frequency Scaling

As the clock distribution network can consume dominant power in common synchronous circuits design, the clock is therefore another candidate for power minimization. Different from clock gating which stops the clock of idle components, *Dynamic Frequency Scaling (DFS)* reduces the power consumption by reducing the clock frequency of active components. However, lowering the frequency results in lowering the speed. In other words, it might lower the instant power but not the total energy consumed. Here we do not want to "freeze" the clock but lower its performance. Compared to clock gating approaches, frequency scaling of synchronous circuits can be applied to more occasions according to system workload.

The literature on frequency scaling is relatively limited than the ones on clock gating. There are various works based on the frequency scaling idea but mainly for improving processor performance. The research on DFS for power minimization are relatively few apart from [RVB98, PCP+02]. We can use a clock divider to generate multiple clock frequencies based on a master clock as in [RVB96, RVB98]. To be more precise, we can use a control signal to select the appropriate output from the clock divider and use it as the clock input of a block. We can adopt this concept of frequency scaling and use the functional block as shown in Figure 2.3 to implement in circuit.

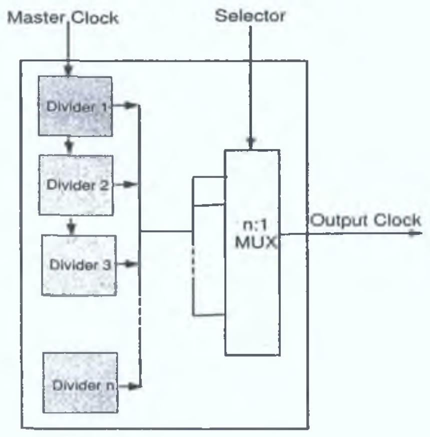


Figure 2.3: Frequency scaling mechanism

2.4.3 Dynamic Voltage and Frequency Scheduling

Having established how DVS performs and disscussed how DFS works, it is now appropriate to introduce the Dynamic Voltage and Frequency Scheduling (DVFS) scheme which is proposed to tackle the power minimization problem in this thesis. The main idea of DVFS is to save energy by changing the working state of a system. The term, scheduling itself is actually ambiguous. As we discussed above, the current scheduling methodologies are based on either *high level synthesis* scheduling or *real-time* scheduling. Our work does not fall into either of the two categories but incorporates features from both. We target the two factors that affect power dissipation mostly, i.e. voltage and frequency. The scheduling in our strategy is to deal with the scheduling of supply voltage and working frequency and adopt the concept of task deadlines in real-time scheduling. We are looking for effective scheduling techniques that treat voltage and frequency as variables to be determined, in addition to conventional task scheduling and allocation.

Chapter 3

Problem Scope and Objective

We intend to develop a DPM strategy which dynamically schedules the scaling of voltage and frequency, hence, our problem scope is to search for methods of DPM scheduling. So far, research into high-level DPM scheduling algorithms can be broadly divided into two categories: one that seeks power optimization at the synthesis stage, and the other starts from the real-time operating system scheduling. Here we first propose an approach that starts from the functional block level. This alternative strategy, which makes use of voltage and frequency scaling in HLS scheduling and task deadlines in RTS scheduling, can take advantages and avoid many drawbacks from both HLS scheduling and RTS scheduling. Before any further discussion on the methodologies and algorithm used in this dissertation, we explain the attempted solving of the problem as well as the assumptions made in doing so. In this chapter, models and methods are presented by figures with explanations. The objective of our approach is to schedule voltage and frequency of functional blocks in a SoC system that is composed of heterogeneous components. Notice that here we do not distinguish between I/O devices and processors, instead treat all functional blocks the same. Based on DPM methodology, our final target is to find a generic voltage/frequncy scheduling strategy to lower system power consumption.

3.1 Problem Scope

Nowadays, time-to-market is an important issue in system design, many system companies "reuse" and integrate pre-designed cores for new systems. Some vendors might adopt Intellectual Properties (IP) from different sources and use these IPs as functional units in their systems. Without loss of generality, in our experiments we consider a system as a combination of different functional blocks as shown in Figure 3.1, we do not consider the functionality of the blocks but treat them as black boxes. This is because we care about the power consumption rather than the function of the blocks. Such abstraction and modularity provide a structural view of the overall system.

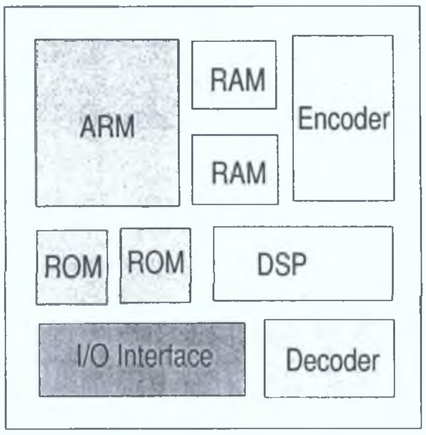
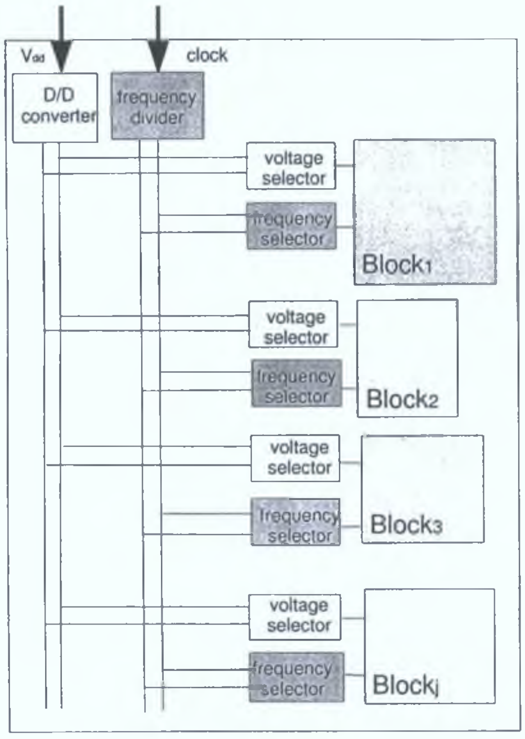


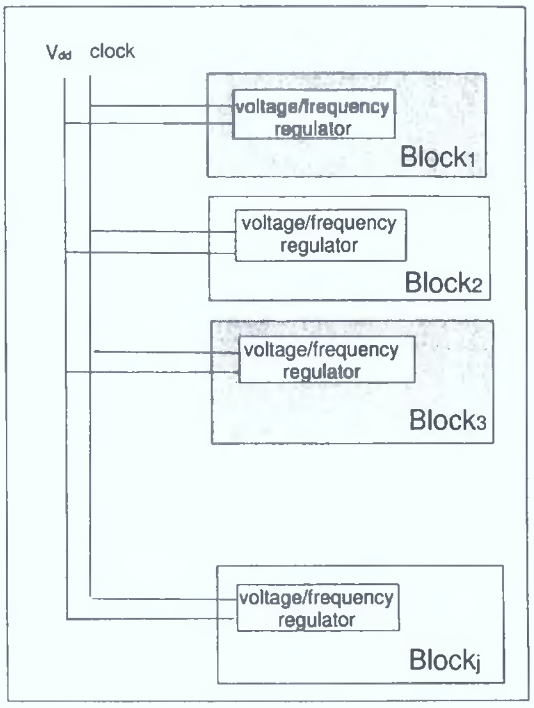
Figure 3.1: An internal view of a SoC architecture

Similar to many VLSI design automation problems, the voltage/frequency scheduling for power minimization is a nondeterministic polynomial-time decidable (*NP-hard*) problem. Looking for solutions to solve any NP-hard problem optimally is impractical. Hence we make certain assumptions to transform it into an *NP-complete* problem and look for efficient algorithms that provide near-optimum and acceptable solutions instead of finding exact optima since it is not polynomial time solvable. In this thesis, we use a heuristic algorithm to solve the voltage/frequency scheduling problem. For any NP-complete or NP-hard problems in CAD for VLSI, heuristics seems to be the only way to solve these problems.

As we do not distinguish functional blocks, this allows us to develop a higher abstract view of a system, and propose a power saving strategy based on a global view. Two alternative structures of the control mechanism are shown in Figure 3.2 (a) and (b). In either configuration, variable voltage/frequency can be set in individual functional blocks.



(a) external



(b) internal

Figure 3.2: Conceptual implementation of voltage/frequency scaling on chip

Before we explain *how* we schedule the voltage/frequency settings, we first look at the question *why* do we use voltage and frequency as parameters to lower power. From Equation 2.3 we know that different parameters can be adjusted in order to lower the power. However, we concentrate only on the two parameters that can be controlled at system level, i.e. voltage and frequency. There have been research proposals using either voltage or frequency scaling to lower power consumption. However, considering the strong link between voltage and frequency, it is quite reasonable to design a mechanism that utilizes a combination of both to lower power consumption.

3.2 Experimental Methodology

As mentioned previously, a SoC is composed of a set of interacting components. This functional block level view of a SoC makes it possible to select an appropriate voltage and frequency setting for each block according to its timing constraint. Our algorithm will generate a schedule, then select from a set of pre-defined supply voltage levels and frequency settings. Our approach combines voltage scaling with frequency scaling based on DPM principles. Some of the assumptions made throughout this thesis are summarized below.

1. The application programs are given in operation sequences that can be obtained in the specification, and we define that an operation period is generated whenever there are slack times.

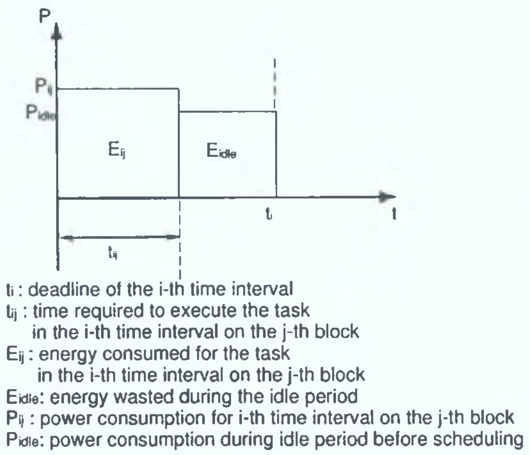
2. The system provides a set of variable voltage/frequency settings for each individual component, i.e. the voltage/frequency on each functional block can be adjusted in different time intervals according to the calculated schedule. It is a trend to support multi-voltage and multi-frequency for SoC, and the implementation can be seen in the foreseeable future.

3. The power and time overhead caused by the state transition mechanism are relatively small and can be neglected.

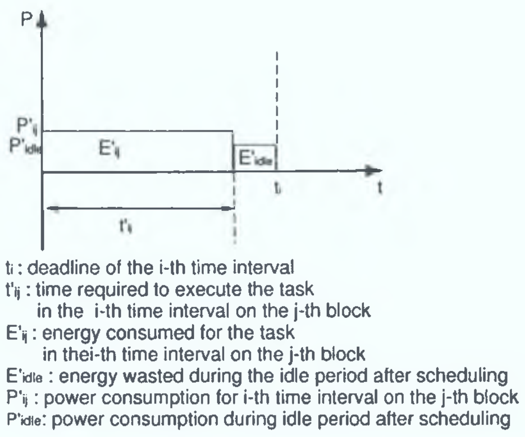
4. The input switching activity N and the capacitive load C of each functional block remains the same before and after scheduling. Because at system level we can ignore minor fluctuations of N and C.

3.2.1 Design Framework

The result of the scheduling is a schedule of the voltage/frequency settings for each functional block at different time intervals. For each operation, we first apply the maximum voltage/frequency on each functional block. For those blocks that are not on the critical path and with long slack time, we look for the lowest supply voltage and frequency setting that still gives non-negative slack time. Our approach is then to search for the lowest voltage/frequency setting for each block that lowers the power consumption within the available slack time. Therefore, our approach attempts to generate a schedule for the functional blocks along the time axis, and we scale the functional blocks' voltage and frequency according to the schedule.


P
P_ij
P_idle
E_ij
E_idle
t_i
t
t_ij
t_i : deadline of the i-th time interval
t_ij : time required to execute the task in the i-th time interval on the j-th block
E_ij : energy consumed for the task in the i-th time interval on the j-th block
E_idle: energy wasted during the idle period
P_ij : power consumption for i-th time interval on the j-th block
P_idle: power consumption during idle period before scheduling

(a) Before scheduling


P
P'_ij
P'_idle
E'_ij
E'_idle
t_i
t
t'_ij
t_i : deadline of the i-th time interval
t'_ij : time required to execute the task in the i-th time interval on the j-th block
E'_ij : energy consumed for the task in thei-th time interval on the j-th block
E'_idle : energy wasted during the idle period after scheduling
P'_ij : power consumption for i-th time interval on the j-th block
P'_idle: power consumption during idle period after scheduling

(b) After scheduling

Figure 3.3: Comparison of energy consumption

Figure 3.3 graphically illustrates the mechanism we propose in this thesis. The main motivation for block-level scheduling comes from the fact that significant power can be saved by lowering supply voltage and working frequency. Consider the working status of a common functional block that works under maximum supply voltage/frequency. It happens very often that the operation has been completed well before the actual deadline (i.e. t_{ij} is shorter than t_i) and this has the consequence of power wasted (E_{idle}) as shown in the shadowed area in Figure 3.3 (a). After the completion of the operation, the component continues consuming power during t_{ij} to t_i. Although the power consumed during the slack time is slightly smaller than P_{ij}, the energy wasted during this period is still dramatic. On the other hand, the approach proposed in this thesis is to adopt the concept of block-level scheduling that lowers voltage/frequency setting. By applying this scaling approach, the system can yield significant energy savings as presented in Figure 3.3 (b). From which, we see the execution time after scheduling t'_{ij} has been extended that accompanied the lowering of the power P'_{ij}. By doing so, the values of E_{ij} and E'_{ij} in Figure 3.3 are actually the same, but the wasted energy in (b) E'_{idle} during the slack time is much less than in (a) E_{idle} hence saves enormous energy. From the above discussions, we can understand why lowering voltage/frequency can provide significant energy savings. Under this scenario, the problem we attempt to solve is *to minimize the energy consumed by block-level voltage/frequency scheduling under given timing constraints.* We then consider the hypothesis and assumptions more concretely in the next section.

3.2.2 Mathematical Model

We use Equation 2.7 and Equation 2.8 to model the power and delay respectively. In order to compare the changes caused by scaling supply voltage and clock frequency, we use the following notations. First, the scaled voltage, V'_{dd}, is denoted by the original supply voltage V_{dd} as well as a scaling factor a, i.e. $V'_{dd} = a \cdot V_{dd}$. As for the scaled clock frequency, it can be obtained by a similar relationship, $f' = b \cdot f$, where b is the frequency scaling factor. Consequently, we can use (V_{dd}, f) and (aV_{dd}, bf) to denote the voltage/frequency settings before and after scheduling respectively. The following are discussions of the power and timing models we used and the efficiency of our approach.

Delay Model

Equation 2.8 defines the strong dependency between propagation delay δ and the supply voltage, V_{dd}, as well as the threshold voltage, V_t, i.e. $\delta \propto \frac{V_{dd}}{(V_{dd}-V_t)^\alpha}$. On the other hand, clock frequency f is inversely proportional to circuit delay, i.e. $\delta \propto \frac{1}{f}$, so we have the following relationship:

$$\delta \propto \frac{V_{dd}}{(V_{dd} - V_t)^\alpha} \cdot \frac{1}{f} \tag{3.1}$$

Let the working voltage and frequency be (V_{dd}, f) before scheduling, and (aV_{dd}, bf) after scheduling. In practical CMOS designs, the threshold voltage V_t ranges from 0.2V to 0.7V, and α is a constant varies between 1 and 2 depending on the technologies. According to the settings, the relationship between the propagation delay before and after scheduling can then be represented by Equation 3.2.

$$\delta(aV_{dd}, bf) = \frac{a}{b} \cdot \frac{(V_{dd} - V_t)^\alpha}{(aV_{dd} - V_t)^\alpha} \cdot \delta(V_{dd}, f) \tag{3.2}$$

Power Model

Equation 2.4 indicates the four factors that influence power consumption. As we want to emphasize the influences of voltage/frequency scaling on power consumption, we assume the first two parameters N, and C remain the same before and after the scaling of frequency and voltage. Under such assumptions, the power equation can be conducted as $P \propto V_{dd}^2 \cdot f$. The relationship between new and original power can then be represented by Equation 3.3.

$$P(aV_{dd}, bf) = a^2 b \cdot P(V_{dd}, f) \tag{3.3}$$

System Model

With the delay model and power model derived, we can now represent the delay time and power consumption as functions of the supply voltage and working frequency. The reason that we adopt both V_{dd} and f in a fixed combination is due to the fact that the delay of combinational logic circuits increases with the decrease of supply voltage, therefore, it is necessary to lower the clock frequency according to the increased circuit delay, so that the state machine can work at the same cycle status. If we lower only the supply voltage, with the increased delay it can happen that the circuit does not work properly with the original clock frequency. On the other hand, simply scaling down the frequency itself does not save as much power as it would by scaling supply voltage at the time time as well. In other words, when scaling down the supply voltage, the clock frequency should also be appropriately scaled down. Consider a system that supports four different voltage/frequency settings, (5V, 80 MHz), (3.3V, 40 MHz), (2.4V, 20 MHz), (1.5V, 10 MHz), we can then represent the normalized relationships of delay and power of Equation 3.2 and Equation 3.3 in Table 3.1.

As the supported modes are given, we can easily get the voltage and frequency factors (a and b) by using (5V, 80 MHz) as the reference setting since it is the normal working mode. Based on a and b, we then further calculate the delay ratio $\frac{a}{b} \cdot \frac{(V_{dd}-V_t)^\alpha}{(aV_{dd}-V_t)^\alpha}$ and power ratio a^2b for each working mode. Take M_2 for instance,

Table 3.1: A table of delay and power ratios of the default setting modes

setting	(V, F)	a	b	delay ratio[1]	power ratio[2]
M_1	(5V, 80 MHz)	1	1	1	1
M_2	(3.3V, 40 MHz)	0.66	0.5	2.1213	0.2178
M_3	(2.4V, 20 MHz)	0.48	0.25	3.7893	0.0576
M_4	(1.5V, 10 MHz)	0.30	0.125	10.7991	0.0113

[1,2] assume $\alpha = 1$, $V_t = 0.5$ in Equation 3.2 and Equation 3.3 respectively.

the working voltage and clock frequency are 0.66 and 0.5 times the ones in mode M_1. From the information in Table 3.1, we know the power consumption at mode M_2 consumes only 21.78% of the power consumed at mode M_1 as the power ratio explains, but this causes the delay which increases to 2.1213 times. By using these relationship and the available slack time, the working mode can then be transited to the lowest power cost mode while meets the timing constraint. Note that here we set α to 1 and V_t to 0.5 V for the sake of convenience, and we will use these settings for the example and experimental simulations in the next chapter.

With the parameters in Table 3.1, the execution time and power consumption as a function of mode transition is plotted in Figure 3.4. The horizontal axis is the power while the vertical axis represents the delay. The power and delay at mode M_1 are used as a reference to quantize the variations of state transition in both axes, i.e. use the information provided in the two rightmost column in Table 3.1. It is important to say that mathematically speaking the assumptions made overly simplify the modelling complexity, as we emphasize the scheduling strategy here so we assume some parameters like N_j and C_j remain the same before and after voltage/frequency scaling. On the other hand, it is important to mention that while our approach use the above settings to simulate the functions, the approach itself is universal and can be adopted to different system settings by using different scaling factors of a and b in Equation 3.2 and 3.3. In this thesis, we assume that all the functional blocks in the system support the same set of voltage/frequency combinations, but the algorithm proposed here does allow different voltage/frequency configuration for different functional blocks.

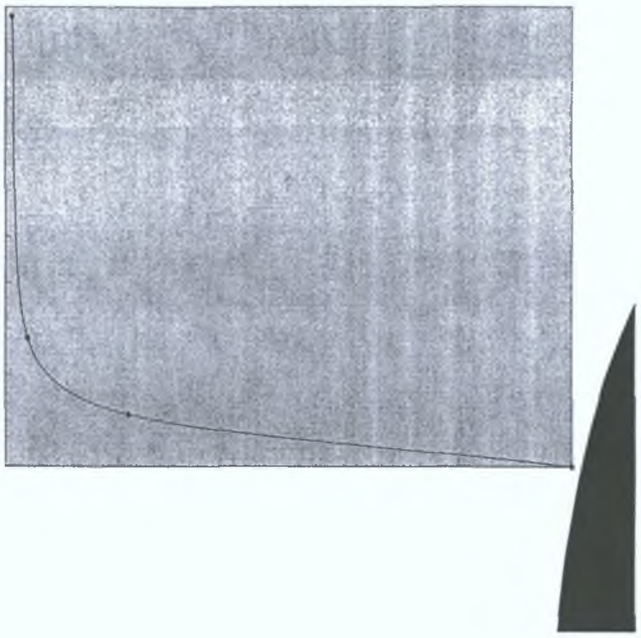


Figure 3.4: Relationships between normalized power versus delay

3.3 Experimental Scheme

3.3.1 Notation

The variables used in the notation are defined as follows. Three categories of variables are used. One set of variables with subscript i is used to indicate the attribute of the i_{th} operation period. Another set of variables with subscript j is used to indicate the j_{th} block characters. The last group of variables combines both i and j to indicate the parameter on the j_{th} block during i_{th} operation period. The inputs of our scheduling policy (the conditions of a given system) include the following items: a set $\mathcal{T}= \{\tau_1, \tau_2, ...\tau_i, ...\tau_m\}$ of m operation periods, a set $B = \{B_1, B_2, ...B_j, ...B_n\}$ of n blocks and s settings of voltages and frequencies supported in the system, $(V^1, F^1), (V^2, F^2), ...(V^k, V^k,)...(V^s, F^s)$. Parameters associated with each operation period $\tau_j \in \mathcal{T}$ are listed below.

- the duration of each operation period: t_i

- a device usage list: L_i consisting of the functional blocks needed in τ_j

Each block B_j has the following parameters,

- transistor number: G_j
- average switching activity: N_j
- average (effective) capacitive load: C_j

Based on the above notations, we can further represent some individual parameters listed below.

- execution time of the *i-th* operation period on the *j-th* component: t_{ij}
- voltage setting of the *i-th* operation period on the *j-th* component: v_{ij}, the value of v_{ij} equals to one of the voltage level settings supported in the system
- power consumption of the *i-th* operation period on the *j-th* component: P_{ij}
- energy consumption of the *i-th* operation period on the *j-th* component: E_{ij}

To avoid confusion, we summarize the parameters related to a block j and operation period i in Table 3.2.

Table 3.2: A table of parameters related to the i_{th} operation period on the j_{th} block

	τ_i
B_j	(v_{ij}, f_{ij}), t_{ij}, P_{ij}, E_{ij}

The power consumed on a specific block for each individual operation period can then be represented by Equation 3.4 as follows.

$$P_{ij} = G_j \cdot N_j \cdot C_j \cdot v_{ij}^2 \cdot f_{ij} \tag{3.4}$$

From Equation 2.5, we know that the energy is the integral of power, hence we can model the energy consumption on a specific block (the j_{th} block) during a specific time interval period (the i_{th} operation period) by Equation 3.5. Among these parameters, E_{ij} represents the energy consumption, t_i means the time period, G_j, N_j, C_j stands for the transistor numbers, switching activities and the output capacitive load of the block respectively, and v_{ij} and f_{ij} are the voltage and frequency being used.

$$\begin{aligned} E_{ij} &= \int_0^t P(t)dt = P_{ij} \cdot t_i \\ &= t_i \cdot G_j \cdot N_j \cdot C_j \cdot v_{ij}^2 \cdot f_{ij} \end{aligned} \tag{3.5}$$

3.3.2 Quality Measurement

We use four different parameters to represent the quality measures of the efficiency of the scheduling. These parameters are *average power*, *maximum (peak) power*, *Root Mean Square (RMS) power*, and *total energy* consumption separately. Total energy consumption (E_{total}) is the main consideration of our design. Average power consumption (P_{avg}) is the most commonly and widely discussed type of power calculation. RMS power (P_{rms}) is usually used to represent the characteristics of long term behavior, with special attention to the peak values. The representation of RMS power is shown below in Equation 3.6. The maximum power (P_{max}), refers to the highest power consumption value during the whole system execution; it determines the power ground wiring designs and impacts signal noise margins and reliability analysis. The RMS power is used for circuit sizing and inducing the allowable parasitic inductance delivered to the device, which are essential system characteristics. All these parameters allow us to evaluate the efficiency of the scheduling strategy.

$$P_{rms} = \sqrt{\frac{1}{n} \cdot (\sum_{i=0}^{n-1} P_i^2)} \tag{3.6}$$

The final goal of this research is to minimize the power consumption, P_{avg} during the whole execution period (i.e. the total energy consumption, E_{total}) on a SoC that is represented by Equation 3.7.

$$E_{total} = \sum_{i=1}^{m} \sum_{j=1}^{n} E_{ij} \tag{3.7}$$

Although we focus mainly on the variations in total energy consumption, other parameters are used as an aid to understanding the characteristic changes before and after the scheduling.

3.4 Objective and Expected Applications

In this dissertation we deal with the problem of minimizing the power consumption of a given SoC. The main idea of this approach is to reduce power by scaling the supply voltage and the clock frequency for components while meeting the timing constraint of the system. Based on DPM methodology, we can apply different voltages and clock frequencies to different components in a system within the available slack times. i.e. provide the same performance.

We constructed a model of a template SoC system and developed a simulator. This computer-based simulator can carry out voltage/frequency scheduling and display the outcomes so that we can study the results and the behavior of the proposed algorithm. We model the system at a very-high abstraction level in which we view the system as a combination of functional blocks. The voltage and frequency of these functional blocks can be set to different states at different time intervals. Theoretically speaking, the more different settings supported in a system the more energy savings can be achieved. However, considering the current design, the voltage level settings for our simulator are adopted from literature [CP99] and listed as follows, {5V, 3.3V, 2.4V, 1.5V}. Corresponding to the provided voltages, the working frequencies are chosen as well. The key question is how to assign a proper speed to each operation period dynamically while guaranteeing all deadlines. We use a heuristic algorithm to find a good solution.

The problem we address is that of determining a schedule of voltage/frequency for the functional blocks such that the energy consumed by the system during the whole period $E_{total} = N \cdot (\sum_{i=1}^{m}(\sum_{j=1}^{n} G_j \cdot C_j \cdot V_{ij}^2 \cdot f_{ij}) \cdot t_i)$ is minimized while ensuring that all operation periods meet their deadlines. The objective of our experiment is to find effective scheduling techniques that determine the voltage and frequency settings for each component in order to reach the goal of minimizing power. Our approach provides another alternative that can be considered at the design stage.

The power increasing with growing density and complexity of digital systems can deteriorate the problem of power consumption very rapidly unless efficient approaches for managing system power are adopted. The disadvantage of a stochastic approach is the calculation complexity which can take much more time and circuit area than the heuristic ones. A simple heuristic is proposed here to find a voltage/frequency schedule which works sufficiently well.

Chapter 4

Experimental Scheduling

Following the definition of the BLVFS problem in the last chapter, here we propose a ***Block Level Dynamic Voltage/Frequency Scaling (BLS) Algorithm*** to solve the BLVFS problem. After a detailed explanation of the algorithm, the simulation experiments of the algorithm are demonstrated. The proposed algorithm has been implemented using Visual C++. The experiments have been simulated under a Windows XP environment on a Pentium IV 2.8 GHz Personal Computer. Using the developed simulator, we simulate the scheduling using system settings that are given as inputs. All testbenches including environmental settings and system inputs were randomly generated. The parameters used in the random generation of each testbench like the settings of the work mode, information of operation periods and functional blocks etc. are set at the beginning of the simulation and used as inputs. After scheduling, the newly generated schedule is output in a text file.

4.1 Scheduling Simulator

Following the discussion of the proposed experiment scheme, we now introduce the functions provided in the simulator. Figure 4.1 demonstrates the process of the simulator graphically. The entire simulation environment includes the following elements:

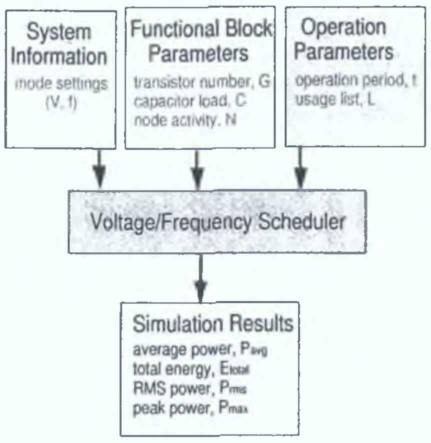


Figure 4.1: Process of the simulator

Environment Settings:

- voltage/frequency settings (modes)
- a set of functional blocks B, where each block $B_j \in B$ is characterized by its transistor number G_j, node transition activity N_j, and load capacitance C_j

Inputs:

Given a set of ordered operation periods $\mathcal{T}= \{\tau_1, \tau_2, ...\tau_i, ...\tau_m\}$. Each operation period τ_i is characterized by its deadline t_i, a list of used functional blocks and the execution time of each individual functional block. It is represented in the format: $\tau_i = (t_i, \{[B_j, t_{ij}]\})$, $\forall$ i, and j, where $1 \leq i \leq m$, and $1 \leq j \leq n$. m is the number of operation periods, and n is the number of functional blocks.

Outputs:

The simulator schedules one operation period at a time and the process is repeated until all operation periods are scheduled. The simulator outputs a schedule in a list format as in Table 4.1. For example, in operation period τ_1, functional block B_1 is scheduled to a configuration of voltage v_{11} and frequency f_{11}, functional block B_2 is configured with voltage v_{12}, and frequency f_{12} and so on. Note that the configured settings of voltage and frequency (v_{ij},f_{ij}) equals to one of the settings supported in the system.

Measurement Expressions:

Table 4.1: Template of scheduling results

	B_1	B_2
τ_1	(v_{11}, f_{11})	(v_{12}, f_{12})
τ_2	(v_{21}, f_{21})	(v_{22}, f_{22})

The final results include four measurement parameters that can be used to assess the quality of a schedule. They are total energy consumption E_{total}, average power consumption P_{avg}, RMS power P_{rms} and peak power P_{max}. The evaluation parameters allow us to compare the alternative solutions to the problem. The proposed algorithm minimizes the average power consumption to achieve the target of lowering total energy consumption. The scheduling procedures of the heuristic algorithm are described in the following section along with its pseudocode.

4.2 Block Level Dynamic Voltage/Frequency Scaling Algorithm

In this section, we introduce the BLS algorithm. The operation of the BLS algorithm is based on a simple idea, i.e. for each functional block at each operation period, find out the lowest voltage/frequency settings that still meets the deadline of the operation period. The scheme of the BLS algorithm is simple, it consists of two iterations, i.e. *operation periods* and *blocks*. In the first step, all operation periods are scheduled with the normal voltage/frequency settings (i.e. 5V, 80 MHz) and the normal time needed for each functional block to complete its operation is known. The algorithm schedules one operation interval at a time according to the order of the operation periods. For each operation period, a feasible schedule is found in the first step, in the second step the supply voltage and clock frequency are adjusted.

```
1   Given:
2   - A list of functional blocks that make up the system
3   - A list of supported work modes, where each mode is corresponding to its
4       voltage/frequency setting
5   - A sequence of operation periods, where each section is characterized by its
6     execution time $t_i$, the usage list L that contains the needed functional blocks
7   - Calculate the original $P_{ij}$, $P_i$, $E_i$, $E_{total}$, $P_{avg}$, $P_{rms}$, $P_{max}$ under the normal
8       voltage/frequency setting,
9   - Initialize the total energy consumption $E_{total}$
10  For (each operation period $i$, from $i=1$ to $m$)
11      {
12          Initialize the power consumption of each operation period $P_i = 0$
13          For (each functional block $j$, from $j=1$ to $n$) //until all functional
14          blocks are scheduled
15              {   /*minimize the power consumed by each block for each
16              operation period*/
17              Try to find the lowest voltage/frequency setting that still meets
18              the deadline
19              Record the chosen mode setting for the functional block
20              Calculate new $P_{ij}$
21              Update $P_i$ with the new $P_{ij}$ component
22              }  //  end for
23          Calculate new $E_i=P_i * t_i$, update $E_{total}$ with new $E_i$ component
24      }// end for
25  Calculate $P_{avg}$, $P_{max}$, $P_{rms}$
```

Figure 4.2: Program pseudocode of the BLS algorithm

Let us now look at the algorithm pseudocode in Figure 4.2. Assume that the system settings are initially given (line 2 to line 6). The measurement parameters at normal voltage/frequency setting are calculated. As can be figured out from the pseudocode itself, the algorithm is repeated until all the functional blocks are scaled in accordance with the operation sequence (line 10 to line 24). Before the scheduling starts, the parameters are initialized (line 9, line 12). The scaling process can be found in the inner *for* loop from line 13 to line 22. For each functional block, search for a lowest feasible voltage/frequency setting. Finally, we calculate E_{total}, P_{avg}, P_{rms}, and P_{max}. The algorithm generates a new optimized schedule from the input information. This BLS algorithm is an iterative heuristic that selects a feasible voltage/frequency setting for each functional block in a SoC at different time intervals.

4.3 Experimental Results

Experimental results are reported in this section to demonstrate the effectiveness of the proposed heuristic algorithms. We present the simulation results of some randomly generated testbenches for the algorithm. We use the total energy consumption E_{total} before and after the voltage/frequency scaling to calculate the percentage of energy saved. In all three testbenches, we consider the systems support the voltage/frequency setting as in Table 4.2. The voltage settings are adopted from the literature [CP99], while the frequency settings are chosen according to the current system design.

Table 4.2: List of supported voltage/frequency settings used in the experiment

setting	(V, F)
M_1	(5V, 80 MHz)
M_2	(3.3V, 40 MHz)
M_3	(2.4V, 20 MHz)
M_4	(1.5V, 10 MHz)

The switching activity parameters N_j of all functional blocks are set at 0.5 for the sake of convenience. The settings of each functional blocks like transistor number G_j and average capacitor load C_j are randomly generated at the beginning of simulation. These experiments test the impact of the scheduling algorithm. In each experiment, the results show the P_{max}, P_{rms}, P_{avg}, E_{total}, and compare the efficiency of power savings.

The target of the experiment is to find a schedule of voltage/frequency settings that saves power while meeting the timing constraint. From Figure 3.4, we know that the delay of a functional block varies with the mode settings as shown in the delay ratio provided in Table 3.1. With the original execution time given as input, we can calculate the new execution time after voltage/frequency scaling, by multiplying the original execution time with the delay ratio and see whether the new execution time meets the timing constraints.

First Testbench

The system here contains five functional blocks, and the related parameters are given in Table 4.3 followed by the usage lists of the 10 execution periods. The total execution time of the set of the operation periods in the first experiment is 107 *ms*.

Table 4.3: Parameters of functional blocks

	B_1	B_2	B_3	B_4	B_5
G	41k	36k	40k	10k	31k
N	0.5	0.5	0.5	0.5	0.5
C	3pF	1pF	5pF	4pF	5pF

τ_1 (6, {[B_1, 5], [B_2, 4], [B_3, 3], [B_4, 1]})

τ_2 (13, {[B_1, 4], [B_2, 1], [B_3, 2], [B_4, 12], [B_5, 5]})

τ_3 (8, {[B_3, 3], [B_4, 4], [B_4, 4]})

τ_4 (16, {[B_1, 9], [B_2, 2], [B_3, 10], [B_4, 7], [B_5, 5]})

τ_5 (8, {[B_1, 4], [B_2, 6], [B_3, 6], [B_4, 4], [B_5, 2]})

τ_6 (14, {[B_1, 10], [B_2, 13], [B_3, 12], [B_4, 7], [B_5, 1]})

τ_7 (8, {[B_1, 4], [B_2, 15], [B_3, 3], [B_4, 14], [B_5, 15]})

τ_8 (2, {[B_1, 1], [B_2, 1]})

τ_9 (18, {[B_2, 6], [B_3, 1], [B_4, 12], [B_5, 1]})

τ_{10} (4, {[B_1, 3], [B_2, 1], [B_3, 1], [B_4, 2], [B_5, 2]})

To interpret the input sequence, consider operation period τ_1 with (6, {[B_1, 5], [B_2, 4], [B_3, 3], [B_4, 1]}). In which, 6 represents the operation period in *ms*, and four functional blocks are used in this operation period, B_1 to B_4. [B_1, 5] means the execution time of functional block B_1 during the operation period τ_1 is 5 *ms*, in other words, there is 1 *ms* slack time. However, according to Table 3.1, the delay will be 2.1213 times longer after scaling down to the (3.3V, 40 MHz) setting which will result in deadline missing. This means we can not scale down the voltage/frequency for functional block B_1 at the operation period τ_1 but should maintain B_1 at the normal operation setting. During the same operation period, functional block B_4 has a normal execution time of 1 ms. Scaling the operation setting down to (3.3V, 40 MHz) will still meet the deadline but not if scale down to (2.4V, 20 MHz), so we set (v_{14}, f_{14}) = (3.3V, 40 MHz) for functional block B_4 during the operation period τ_1. The scheduling results are shown in Table 4.4 with voltage settings only, as the voltage and frequency settings are in combined set so frequency settings are not listed here to avoid redundancy.

Table 4.4: Voltage settings of the scheduling results for the first testbench

	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6	τ_7	τ_8	τ_9	τ_{10}
B_1	5	3.3	0	5	5	5	3.3	5	0	5
B_2	5	2.4	0	3.3	5	5	5	5	3.3	3.3
B_3	5	3.3	3.3	5	5	5	3.3	0	2.4	3.3
B_4	3.3	5	5	3.3	5	5	5	0	5	5
B_5	0	3.3	5	3.3	3.3	2.4	5	0	2.4	5

From the information provided in Table 4.4, we then get the four measurement parameters (E_{total}, P_{avg}, P_{max}, and P_{rms}) by the following explanations. According to Equation 3.4, $P_{ij} = G_j \cdot N_j \cdot C_j \cdot v_{ij}^2 \cdot f_{ij}$. Consider the power consumption for the functional block B_1 at operation period τ_1, P_{11}, that can be calculated by Equation 4.1.

$$P_{11} = 41k \cdot 0.5 \cdot 3\,p \cdot 5^2 \cdot 80\,M = 123\;(unit\,power) \tag{4.1}$$

For each specific (i_{th}) operation period, we can calculate the power consumption P_i by Equation 4.2. The values of P_i in each operation periods are used to represent the power transitions during whole period in Figure 4.3.

$$P_i = \sum_{j=1}^{n} P_{ij} \tag{4.2}$$

Based on the P_i, we can then get the other measurement parameters as follows.

$$E_{total} = \sum_{i=1}^{m} P_i \cdot t_i = 29.61\ (unit\,energy) \tag{4.3}$$

Via total energy consumption, we can get the average power consumption by Equation 4.4 below, where $t_{total} = \sum_{i=1}^{m} t_i = 107\,ms$.

$$P_{avg} = \frac{E_{total}}{t_{total}} = 276.7\ (unit\,power) \tag{4.4}$$

and the maximum power,

$$P_{max} = max\ \{\ P_i\ \} \tag{4.5}$$

In Equation 3.6, RMS power is defined as $P_{rms} = \sqrt{\frac{1}{n} \cdot (\sum_{i=0}^{n-1} P_i^{\,2})}$. Because each operation period has different execution time, we can not simply use P_i to calculate the P_{rms}. Equation 4.6 also takes into consideration of the execution time of each operation period as the weighting of each P_i in calculating P_{rms}.

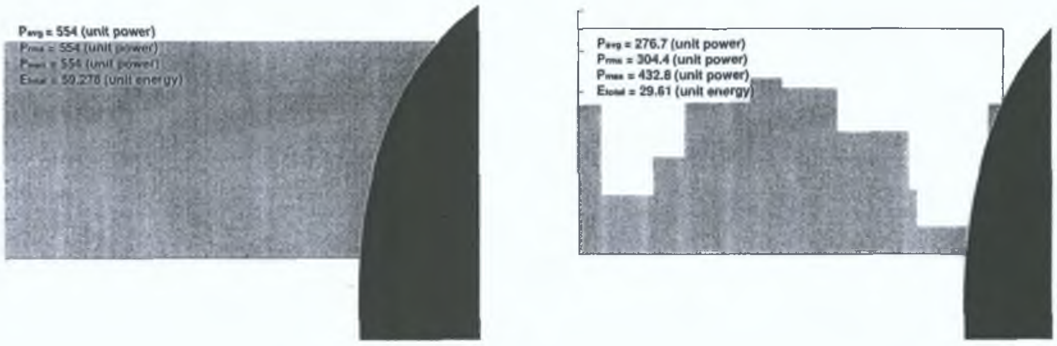
$$P_{rms} = \sqrt{\frac{1}{t_{total}} \cdot (\sum_{i=1}^{m} P_i^{\,2} \cdot t_i)} \tag{4.6}$$

For the original energy consumption, we assume the system operates at (5V, 80 MHz) setting for all functional blocks hence the the power consumption at each operation remains the same for whole system operation period 107 ms. Since there is no scaling of the voltage/frequency in the system, the average power equals both RMS power and maximum power (554 $unitpower$). The original energy consumption at settings without scheduling modification was 59.278 $unit\ energy$ while the one after the scheduling is 29.61 $unit\ energy$ which implies a saving of 50.05%. The RMS power is 304.4 $unit\,power$, and the maximum power is 432.8 $unit\,power$. A comparison of the four parameters is listed in Table 4.5 below.

Figure 4.3 is used to represent the variation of P_i before scheduling (a) and after scheduling (b) along the time. In Figure 4.3 (b), the dark area implies the total energy consumption for the whole operation period and the white area in Figure 4.3 represents the energy saved. In the following testbenches, we use the same calculation process for the measurement parameters, but we do not give the detailed calculation again to avoid repetition. Detailed simulation results can be found in Appendix A.

Table 4.5: Comparison of the four parameters before and after scheduling

	Before Scheduling	After Scheduling
P_{avg}	554	276.7
P_{rms}	554	304.4
P_{max}	554	432.8
E_{total}	59.278	29.61



(a) Before scheduling

(b) After scheduling

Figure 4.3: Comparison of power consumption before and after scheduling in the first testbench

Second Testbench

In the second testbench, we test a system consists of 10 different functional blocks. Among them, five have the same characteristics as in the first testbench. The parameters of the functional blocks are given in Table 4.6. The operation period and usage list are listed below.

τ_1 (16, {[B_1, 7], [B_2, 1], [B_5, 6], [B_6, 8], [B_7, 9], [B_8, 9], [B_9, 9], [B_{10}, 2]})

τ_2 (10, {[B_1, 4], [B_2, 3], [B_4, 6], [B_5, 7], [B_6, 8], [B_7, 5], [B_8, 3], [B_9, 8], [B_{10}, 7]})

τ_3 (11, {[B_1, 10], [B_2, 5], [B_3, 1], [B_4, 5], [B_5, 2], [B_6, 9], [B_7, 2], [B_8, 8], [B_9, 9], [B_{10}, 10]})

τ_4 (2, {[B_1, 1], [B_5, 1], [B_9, 1]})

τ_5 (4, {[B_1, 3], [B_2, 1], [B_3, 1], [B_4, 2], [B_5, 2], [B_6, 3], [B_7, 2], [B_8, 1], [B_{10}, 2]})

τ_6 (18, {[B_1, 8], [B_2, 17], [B_3, 13], [B_4, 1], [B_5, 10], [B_6, 6], [B_7, 13], [B_8, 10], [B_9, 10], [B_{10}, 6]})

τ_7 (8, {[B_1, 1], [B_2, 3], [B_3, 6], [B_4, 4], [B_5, 7], [B_6, 6], [B_7, 2], [B_8, 1], [B_9, 5], [B_{10}, 3]})

τ_8 (12, {[B_1, 8], [B_2, 6], [B_3, 1], [B_4, 11], [B_5, 1], [B_6, 10], [B_7, 8], [B_8, 3], [B_9, 5]})

τ_9 (1, {ϕ})

τ_{10} (2, {[B_1, 1], [B_4, 1], [B_5, 1], [B_6, 1], [B_7, 1]})

Table 4.6: Parameters of functional blocks in the second and third testbench

	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}
G	41k	36k	40k	10k	31k	18k	16k	44k	34k	33k
N	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
C	3pF	1pF	5pF	4pF	5pF	1pF	3pF	2pF	3pF	2pF

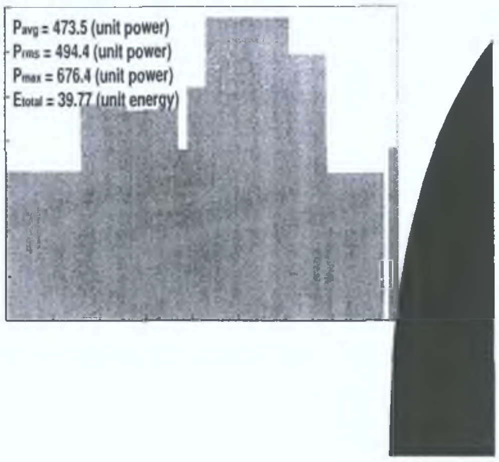


Figure 4.4: Scheduling results of the second testbench

Similar to the first testbench, the simulation results can be represented. To avoid the redundancy, we shows only the graphic result in Figure 4.4. The energy consumption without scheduling was 73.584 *unit energy* and 39.77 *unit energy* with scheduling hence saves 45.95% of energy consumption. The total execution time of system in this testbench is 84 *ms* which means the average power consumption P_{avg} before and after scheduling are 876 *unit energy* and 473.5 *unit energy* respectively. As for the other tow parameters, P_{max} and P_{rms}, they are the same value as the average power consumption before scheduling scheduling and the values after scheduling are shown in Figure 4.4.

Third Testbench

Use the same components as in the second testbench to construct a system, but with 10 extra operation periods given below. The total execution time is 222 *ms*. The energy consumption after scheduling is 109.7 *unit energy*, it represents a saving of 43.59 % compared to the 194.472 *unit energy* before the scheduling was applied. Similarly, the scheduling result is graphically shown in Figure 4.5. The original values of P_{avg}, P_{max}, and P_{rms} are all equal to 876 *unit energy* and the values after scheduling are 494 *unitpower*, 700 *unitpower*, and 513.8 *unitpower* respectively.

τ_{11} (10, $\{[B_2, 5], [B_3, 6], [B_4, 4], [B_5, 1], [B_6, 9], [B_7, 9], [B_8, 5], [B_9, 3], [B_{10}, 4]\}$)

τ_{12} (14, $\{[B_1, 11], [B_2, 4], [B_3, 6], [B_4, 3], [B_5, 13], [B_6, 4], [B_7, 10], [B_8, 7], [B_9, 2], [B_{10}, 10]\}$)

τ_{13} (14, $\{[B_1, 5], [B_2, 7], [B_3, 12], [B_5, 13], [B_6, 8], [B_8, 7], [B_9, 2], [B_{10}, 11]\}$)

τ_{14} (18, $\{[B_1, 11], [B_2, 3], [B_3, 15], [B_4, 2], [B_5, 1], [B_6, 13], [B_7, 5], [B_1, 16], [B_9, 5], [B_1, 6]\}$)

τ_{15} (18, $\{[B_1, 13], [B_2, 15], [B_3, 12], [B_4, 10], [B_5, 13], [B_6, 4], [B_7, 2],[B_9, 1], [B_{10}, 14]\}$)

τ_{16} (9, $\{[B_1, 1], [B_2, 1], [B_3, 4], [B_5, 1], [B_6, 4], [B_7, 6], [B_8, 6], [B_9, 4]\}$)

τ_{17} (4, $\{[B_2, 2], [B_3, 1], [B_4, 1], [B_6, 2], [B_7, 2], [B_8, 2], [B_{10}, 2]\}$)

τ_{18} (17, $\{[B_1, 11], [B_2, 15], [B_3, 3], [B_4, 5], [B_5, 3], [B_6, 16], [B_7, 7], [B_8, 16], [B_9, 1], [B_{10}, 10]\}$)

τ_{19} (16, $\{[B_1, 7], [B_2, 14], [B_4, 14], [B_5, 4], [B_6, 3], [B_7, 12], [B_8, 6], [B_9, 3], [B_{10}, 10]\}$)

τ_{20} (18, $\{[B_1, 8], [B_2, 8], [B_3, 10], [B_4, 11], [B_5, 15], [B_6, 14], [B_7, 11], [B_8, 12], [B_9, 10], [B_{10}, 6]\}$)

Based on the above simulations and the results in Figure 4.3, 4.4, and 4.5, we can see the dramatic energy savings by applying BLS algorithm in the system design.

4.4 Experimental Evaluation and Conclusion

The complexities of the original BLS algorithm is $O(mn)$, for m operation periods and n functional blocks. Although the BLVFS problem itself is NP-complete

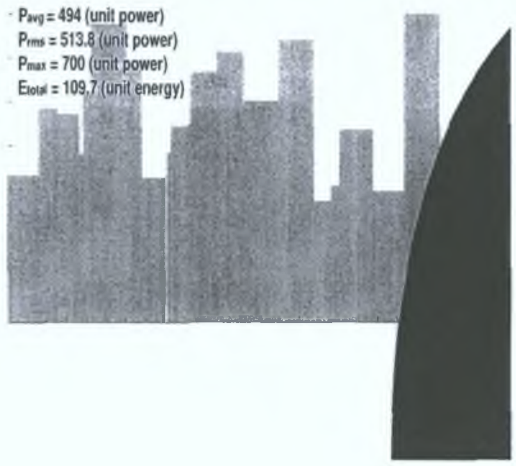


Figure 4.5: Scheduling results of the third testbench

in general, heuristic approaches can be used to solve the problem efficiently in polynomial time. This kind of sub-optimal solutions could be further improved by more sophisticated algorithms. The above simulations were done on system information generated randomly within the scope given beforehand. The experimental results prove that a BLVFS scheme can save significant energy. The main drawback with respect to simulative approaches is the lack of accurate power model. The scheduling algorithm can achieve better results (i.e. save more power) when the timing constraint is more loose as functional blocks can be set at modes with lower power consumption.

From the discussions of the innovative BLS algorithm, we see the most distinguishing point is its adaptivity in any system. For systems constructed by only a few types of functional blocks, there might be some chance to optimize the power usage by exploiting the common functional block characteristics in it. For this reason we also try to adapt the *Left Edge Algorithm* (LEA) to the BLVFS problem. The LEA is a track assignment algorithm and is famous for its application in channel-routing for minimizing the number of tracks used to connect nets on the channel boundary. LEA was then applied to register-allocation problem for minimizing the number of registers [KP87] and applied to block-level test scheduling problem for balancing power dissipation during test [Mur01]. The basic idea of the LEA in BLVFS is to try to make full use of a functional block, and activate another functional block of the same type only if the current functional block can not perform all the required operations within available time. This means trying to schedule the same operations to a functional block while there is still slack time. For the simulations in this case, we consider systems that contain multi-functional blocks of the same type. Surprisingly, the simulation results are contrary to our expectation as we expected to further optimize the energy consumption by their similar characteristics. Other than the problem that LEA does not perform better than BLS, there is another problem of using LEA since the design will require extra routing to transfer the workload between the functional blocks of the same type.

The work presented in this thesis is based mainly on voltage scaling coupled with frequency scaling. Note that our approach does not rely on any specific power model and is thus applicable to any power model. Even though it does not guarantee optimal voltage/frequency setting solutions, its final result can be used as a starting point by near-optimal block-level voltage/frequency scheduling approaches (e.g. *simulated annealing*, *genetic algorithms*, *tabu search*) to get improved solutions.

Chapter 5

Conclusion and Future Work

5.1 Summary and Conclusions

Reducing power dissipation is a design goal for both portable and high end systems. As most systems do not need peak performance at all time, it is possible to transit system components into low-power states by adjusting their frequency and voltage of operation while slack time exists. The whole thesis surrounds a DPM methodology that can gain significant power savings. The aim of this thesis is to propose a novel voltage and frequency scaling scheme at functional block level that can be applied at run-time. We discussed power optimization in Chapter 2. Both *heuristic* and *stochastic* DPM methodologies can be applied in *power down* and *voltage/frequency scaling* approaches. In this thesis, our target is to lower $P_{dynamic}$ of a SoC at system level, so we focus on voltage and frequency which can be managed by system designers.

BLVFS takes the advantage of the fact that typical modern systems are over-designed and IPs have become widely used in the industry. The algorithm aims at stretching the execution time of the functional blocks in each operation period through voltage/frequency scaling. For real-time SoCs, the proposed BLVFS scheme focuses on minimizing energy consumption, while still meeting the performance requirements. In order to simplify the analysis and to allow for the derivation of analytical formulas, we assume N and C are not affected by voltage and frequency scaling during simulation. The primary advantage of BLVFS is its conceptual simplicity and adaptivity. The proposed heuristic for block level voltage and frequency scaling is of low complexity, yet it can generate close to optimal solutions.

This thesis presents a study of BLVFS in order to derive a new approach that can minimize power consumption. Designers focus on worst-case circumstances, therefore, power management exploits idleness resulting from system under-utilization. We proposed a tentative algorithm for possible implementation of voltage and frequency scaling. Heuristic algorithms generally need shorter run-time than stochastic algorithms hence they are more suitable for real-time use. The difference from the conventional techniques is that the proposed methodology is very flexible and can be extended to various other applications. The main contributions of this thesis is to provide an innovative BLVFS idea and to prove the efficiency of the proposed heuristics for dramatic power savings. Besides, since the short-circuit power P_{short} is approximately proportional to $(V_{dd} - 2 \cdot V_{th})^3$ so the short-circuit power is also consequently reduced with the lowering of the supply voltage.

5.2 Future Work

Future work includes three main directions. One is to further research the algorithms that can be applied to the BLVFS problem in order to further optimize the scheduling results. The second research direction is to investigate the applicability of the BLVFS in practical circuit designs, i.e. hardware implementation issues. The third direction is to combine the BLVFS method with other power-aware design considerations like power estimation and static power minimization, etc.

The previous chapters have provided a first look at the issues involved in BLVFS problems and proposed greedy heuristics. We expect to see the use of optimization heuristics like *tabu search*, *simulated annealing*, and *generic algorithm* etc. to solve the BLVFS problem in the near future. Many real-time SoC systems can be modelled using our approach, we can generalize our approach to handle more general models like dependent operation interactions, and preemptive scheduling cases.

To implement the BLVFS in a system that scales the working voltage and frequency will mean some extra control unit, i.e. the implementation of high-efficiency DC-DC converters and frequency dividers in hardware. The power management schemes make a complex tradeoff between power, speed, and area whereas we emphasize only the first two in the algorithm, there is also power and time cost of the extra control unit need to be considered. Future work should look at the practical implementation aspects involved, to combine resource allocation and other lower level design approaches along with the scaling algorithm so that the whole system design is optimized, for instance, the consideration of the consequential implementation issues during binding and layout stage should be further discussed as well [Wol02].

The discussions in this thesis focus on the dominant component of power consumption, $P_{dynamic}$, however, with the shrinking of feature size, static power dissipation, P_{static} will increase to the same level as $P_{dynamic}$. Further investigation of leakage power and glitch should be taken into account for future research directions.

The proposed algorithm in this thesis can be viewed as the beginning of a mechanism to support low power design. In the course of time, the BLVFS scheme may provide a distinctively different dimension for low power designs as compared to classical techniques that are problem-specific. To improve the design accuracy, there is the need for cycle-accurate power estimation of functional modules. System designers lack high-level CAD tools to support power-related design issues so far (see Appendix B). In the commercial arena, we have not seen an efficient solution so we would stress again the need of Electronic Design Automation (EDA) tools to encapsulate the current approaches that provides power managing scheme at early stage. So far, the system and high level power-aware design is still in its infancy and needs more efforts.

Bibliography

[BBM00] Luca Benini, Alessandro Bogliolo, and Giovanni De Micheli. A survey of design techniques for system-level dynamic power management. *IEEE Transactions on Very Large Scale Integration Systems*, 8(3):299–316, June 2000.

[BdM00] Luca Benini and Giovanni de Micheli. System-level power optimization: techniques and tools. *ACM Transactions on Design Automation of Electronic Systems*, 5(2):115–192, 2000.

[BM98] Luca Benini and Giovanni De Micheli. *Dynamic power management – design techniques and CAD tools.* Kluwer Academic Publishers, 1998.

[BM99] Luca Benini and Giovanni De Micheli. System-level power optimization: techniques and tools. In *Proceedings 1999 international symposium on Low power electronics and design*, pages 288–293. ACM Press, 1999.

[CB95] Anantha P. Chandrakasan and Robert W. Brodersen. *Low power digital CMOS design.* Kluwer Academic Publishers, 1995.

[CBM99] Eui-Young Chung, Luca Benini, and Giovanni De Micheli. Dynamic power management using adaptive learning tree. In *Proceeding of the 1999 international conference on Computer-aided design*, pages 274–279. IEEE, 1999.

[CIB01] Rita Yu Chen, Mary Jane Irwin, and Raminder S. Bajwa. Architecture-level power estimation and design experiments. *ACM Transactions on Design Automation of Electronic Systems*, 6(1):50–66, 2001.

[CIC$^+$03] Noureddine Chabini, Ismail, Chabini, El Mostapha Aboulhamid, and Yvon Savaria. Methods for minimizing dynamic power consumption in synchronous designs with multiple supply voltages. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(3):346–351, 3 2003.

[CP96] Jue-Ming Chang and Massoud Pedram. Energy minimization using multiple supply voltages. In *Proceedings of the 1996 international symposium on Low power electronics and design*, pages 157–162. IEEE, 1996.

[CP97] Jue-Ming Chang and Massoud Pedram. Energy minimization using multiple supply voltages. *IEEE Transactions on Very Large Scale Integration Systems*, 5(4):436–443, 12 1997.

[CP99] Jue-Ming Chang and Massoud Pedram. *Power Optimization and Synthesis at Behavioral and System Levels Using Formal Methods.* Kluwer Academic Publishers, 1999.

[CPM+95] Anantha P. Chandrakasan, Miodrag Potkonjak, Renu Mehra, Jan Rabaey, and Robert W. Brodersen. Optimizing power using transformations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14(1):12–31, 1 1995.

[CS99] Chunhong Chen and Majid Sarrafzadeh. Provably good algorithm for low power consumption with dual supply voltages. In *Proceeding of the 1999 international conference on Computer-aided design*, pages 76–79. IEEE, 1999.

[FEG+01] S.B. Furber, A. Efthymiou, J.D. Garside, D.W. Lloyd, M.J.G. Lewis, and S. Temple. Power management in the amulet microprocessors. *IEEE Design and Test of Computers*, 18(2):42–52, March-April 2001.

[FGSS98] William Fornaciari, Paolo Gubian, Donatella Sciuto, and Christina Silvano. Power estimation of embedded systems: a hardware/software codesign approach. *IEEE Transactions on Very Large Scale Integration Systems*, 6(2):266–275, 6 1998.

[FM02] Krisztian Flautner and Trevor Mudge. Vertigo: Automatic performance-setting for linux. In *OSDI*, 2002.

[GBS+95] Richard Golding, Peter Bosch, Carl Staelin, Tim Sullivan, and John Wilkes. Idleness is not sloth. In *Proceedings of the Winter Usenix Conference*, pages 201–212, 1995.

[Gru02] Flavius Gruian. *Energy-centric scheduling for real-time systems.* PhD thesis, Lund University, 11 2002.

[GVNG94] Daniel D. Gajski, Frank Vahid, Sanjiv Narayan, and Jie Gong. *Specification and design of embedded systems.* Prentice Hall, 1994.

[HMK+99] A. Hemani, T. Meincke, S. Kumar, A. Postula, T. Olsson, P. Nilsson, J. Oberg, P. Ellervee, and D. Lundqvist. Lowering power consumption in clock by using globally asynchronous locally synchronous design style. In *Proceedings of the 36th ACM/IEEE conference on Design automation conference*, pages 873–878. ACM Press, 1999.

[HPIM+] Hewlett-Packard, Intel, Microsoft, Phoenix, and Toshiba. Advanced configuration and power interface. [Online] http://www.acpi.info.

[HPK99] Inki Hong, Miodrag Potkonjak, and Ramesh Karri. Power optimization using divide-and-conquer techniques for minimization of the number of operations. *ACM Transactions on Design Automation of Electronic Systems*, 4(4):405–429, 1999.

[HPS98] Inki Hong, Miodrag Potkonjak, and Mani B. Srivastava. On-line scheduling of hard real-time tasks on variable voltage processor. In *Proceedings of the 1998 IEEE/ACM international conference on Computer-aided design*, pages 653–656. ACM, 1998.

[HW97] Chi-Hong Hwang and Allen C.-H. Wu. A predictive system shutdown method for energy saving of event-driven computation. In *Proceedings of the 1997 IEEE/ACM international conference on Computer-aided design*, pages 28–32. IEEE, 1997.

[IUN+97] Mutsunori Igarashi, Kimiyoshi Usami, Kazutaka Nogami, Fumihiro Minami, Yukio Kawasaki, Takahiro Aoki, Midori Takano, Chiharo Mizuno, Takashi Ishikawa, Masahiro Kanazawa, Shinji Sonoda, Makoto Ichida, and Naoyuki Hatanaka. A low-power design method using multiple supply voltages. In *Proceedings of the 1997 international symposium on Low power electronics and design*, pages 36–41. ACM, 1997.

[IY98] Tohru Ishihara and Hiroto Yasuura. Voltage scheduling problem for dynamically variable voltage processors. In *Proceedings 1998 international symposium on Low power electronics and design*, pages 197–202. ACM, 1998.

[Jha01] Niraj K. Jha. Low power system scheduling and synthesis. In *Proceedings of the 2001 IEEE/ACM international conference on Computer-aided design*, pages 259–263, 2001.

[JR97a] Mark C. Johnson and Kaushik Roy. Datapath scheduling with multiple supply voltages and level converters. *ACM Transactions on Design Automation of Electronic Systems*, 2(3):227–248, 1997.

[JR97b] Mark C. Johnson and Kaushik Roy. Scheduling and optimal voltage selection for low power multi-voltage dsp datapaths. In *Proceedings of the 1997 IEEE International Symposium on Circuits and Systems*, pages 2152–2155. IEEE, 1997.

[KL99] James B. Kuo and Jea-Hong Lou. *Low-voltage CMOS VLSI circuits.* Wiley-Interscience, 1999.

[KP87] Fadi J. Kurdahi and Alice C. Parker. Real: A program for register allocation. In *24th ACM/IEEE Design Automation Conference*, pages 210–215, 1987.

[LHW97] Yann-Rue Lin, Cheng-Tsung Hwang, and Allen C.-H. Wu. Scheduling techniques for variable voltage low power designs. *ACM Transactions on Design Automation of Electronic Systems*, 2(2):81–97, April 1997.

[LM01] Yung-Hsiang Lu and Giovanni De Micheli. Comparing system-level power management. *IEEE Design and Test of Computers*, 18(2):10–19, March-April 2001.

[Lu01] Yung-Hsiang Lu. *Power-aware operating systems for interactive systems.* PhD thesis, Stanford University, December 2001.

[LXC03] Dexin Li, Qiang Xie, and Pai H. Chou. Scalable modeling and optimization of mode transitions based on decoupled power management architecture. In *Proceedings of the 2003 Design Automation Conference*. ACM, 2003.

[MC01] Ali Manzak and Chaitali Chakrabarti. Variable voltage task scheduling algorithms for minimizing energy. In *Proceedings of the 2001 international symposium on Low power electronics and design*, pages 279–282. ACM, 2001.

[MC02] Ali Manzak and Chaitali Chakrabarti. A low power scheduling scheme with resources operating at multiple voltages. *IEEE Transactions on Very Large Scale Integration Systems,*, 10(1):6–14, February 2002.

[Mic94] Giovanni De Micheli. *Synthesis and optimization of digital circuits.* McGraw-Hill, 1994.

[Mic97] Microsoft. Onnow: the evolution of the pc platform, 1997.

[MK95] Raul San Martin and John P. Knight. Power-profiler: optimizing asics power consumption at the behavioral level. In *Proceedings of the 32nd ACM/IEEE conference on Design automation conference*, pages 42–47. ACM, 1995.

[MPS97] Enrico Macii, Massoud Pedram, and Fabio Somenzi. High-level power modeling, estimation, and optimization. In *Proceedings of the 34th annual conference on Design automation conference*, pages 504–511. ACM, 1997.

[Mur01] Valentin Muresan. *Block-level Test Scheduling under Power Dissipation Constraints.* Ph. d thesis, Dublin City University, December 2001.

[NCNvB94] Lars S. Nielsen, Jens Sparso Cees Niessen, and Kees van Berkel. Low-power operation using self-timed circuits and adaptive scaling of the supply voltage. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 2(4):391–397, December 1994.

[PCP$^+$02] Kiran Puttaswamy, Kyu-Won Choi, Jun Cheol Park, III Vincent J. Mooney, Abhijit Chatterjee, and Peeter Ellervee. System level power-performance trade-offs in embedded systems using voltage and frequency scaling of off-chip buses and memory. In *Proceedings of the 15th international symposium on System Synthesis*, pages 225–230. ACM, 2002.

[Ped96] Massoud Pedram. Power minimization in ic design: principles and applications. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 1(1):3–56, January 1996.

[Ped01] Massoud Pedram. Power optimization and management in embedded systems. In *Proceedings of the conference on Asia South Pacific Design Automation Conference*, pages 239–244. ACM, 2001.

[PR02] Massoud Pedram and Jan Rabaey. *Power aware design methodologies.* Kluwer Academic Publishers, 2002.

[QP99] Qinru Qiu and Massoud Pedram. Dynamic power management based on continuous-time markov decision processes. In *Proceedings of the 36th ACM/IEEE conference on Design automation conference*, pages 555–561. ACM, 1999.

[QWP00] Qinru Qiu, Qing Wu, and Massoud Pedram. Dynamic power management of complex systems using generalized stochastic petri nets. In *Proceedings of the 37th conference on Design automation*, pages 352–356. ACM, 2000.

[RJD98] Anand Raghunathan, Niraj K. Jha, and Sujit Dey. *High-level power analysis and optimization.* Kluwer Academic Publishers, 1998.

[RP96] Jan M. Rabaey and Massoud Pedram. *Low power design methodologies.* Kluwer Academic Publishers, 1996.

[RP00] Kaushik Roy and Sharat C. Prasad. *Low-power CMOS VLSI circuit design.* John Wiley and Sons, 2000.

[RS95] Salil Raje and Majid Sarrafzadeh. Variable voltage scheduling. In *Proceedings 1995 international symposium on Low power design*, pages 9–14. ACM, 1995.

[RVB96] N. Ranganathan, N. Vijaykrishnan, and N. Bhavanishankar. A vlsi array architecture with dynamic frequency. In *Proceedings of the 1996 IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pages 137–140. IEEE, 1996.

[RVB98] N. Ranganathan, N. Vijaykrishnan, and N. Bhavanishankar. A linear array processor with dynamic frequency clocking for image processing applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(4):435–445, August 1998.

[SBM01] Tajana Simunic, Luca Benini, and Giovanni De Micheli. Energy-efficient design of battery-powered embedded systems. *IEEE Transactions on Very Large Scale Integration Systems*, 9(1):15–28, 2 2001.

[SC98] Wen-Tsong Shiue and Chaitali Chakrabarti. Low power scheduling with resources operating at multiple voltages. In *Proceedings of the 1998 IEEE International Symposium on Circuits and Systems*, pages 437–440. IEEE, 1998.

[SC99] Youngsoo Shin and Kiyoung Choi. Power conscious fixed priority scheduling for hard real-time systems. In *Proceedings of the 36th ACM/IEEE conference on Design automation conference*, pages 134–139. ACM, 1999.

[SC03] Vishnu Swaminathan and Krishnendu Chakrabarty. Energy-couscious, deterministic i/o device scheduling in hard real-time systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(7):847–858, 7 2003.

[SCI01] Vishnu Swaminathan, Krishnendu Chakrabarty, and S. S. Iyengar. Dynamic i/o power management for hard real-time systems. In *Proceedings of the ninth international symposium on Hardware/software codesign*, pages 237–242. ACM, 2001.

[SCS00] Youngsoo Shin, Kiyoung Choi, and Takayasu Sakurai. Power optimization of real-time embedded systems on variable speed processors. In *Proceedings of the 2000 IEEE/ACM international conference on Computer-aided design*, pages 365–368. IEEE, 2000.

[Sim01] Tajana Simunic. *Energy efficient system design and utilization.* PhD thesis, Stanford University, February 2001.

[UH95] Kimiyoshi Usami and Mark Horowitz. Clustered voltage scaling technique for low-power design. In *Proceedings 1995 international symposium on Low power design*, pages 3–8. ACM, 1995.

[UK03] Osman S. Unsal and Israel Koren. System-level power aware design techniques in real-time systems. *Proceedings of the IEEE*, 91(July):1–15, 7 2003.

[UNI$^+$97] Kimiyoshi Usami, Kazutaka Nogami, Mutsunori Igarashi, Fumihiro Minami, Yukio Kawasaki, Takashi Ishikawa, Masahiro Kanazawa, Takahiro Aoki, Midori Takano, Chiharu Mizuno, Makoto Ichida, Shinji Sonoda, Makoto Takahashi, and Naoyuki Hatanaka. Automated low-power technique exploiting multiple supply voltages applied to a media processor. In *Custom Integrated Circuits Conference*, pages 131–134. IEEE, 1997.

[Wol02] Wayne Wolf. *Modern VLSI design.* Prentice Hall, 2002.

[YK03] Han-Saem Yun and Jihong Kim. On energy-optimal voltage scheduling for fixed-priority hard real-time systems. *ACM Transactions on Embedded Computing Systems*, 2(3):393–430, 8 2003.

Appendix A: Simulation Results of the First Testbench

This is the result BEFORE scheduling!

****

The generated scheduling settings

	OP 1	OP 2	OP 3	OP 4	OP 5	OP 6	OP 7	OP 8	OP 9	OP 10
B1	5	5	5	5	5	5	5	5	5	5
B2	5	5	5	5	5	5	5	5	5	5
B3	5	5	5	5	5	5	5	5	5	5
B4	5	5	5	5	5	5	5	5	5	5
B5	5	5	5	5	5	5	5	5	5	5

power consumption P_ij (unit power)

	OP 1	OP 2	OP 3	OP 4	OP 5	OP 6	OP 7	OP 8	OP 9	OP 10
B1	123	123	123	123	123	123	123	123	123	123
B2	36	36	36	36	36	36	36	36	36	36
B3	200	200	200	200	200	200	200	200	200	200
B4	40	40	40	40	40	40	40	40	40	40
B5	155	155	155	155	155	155	155	155	155	155

total execution time of the system is 107 ms

E_total = 59.28 (unit energy)

P_avg = P_rms = P_max = 554 (unit power)

Parameters related to each operation period P_i

operation period 1 : t = 6 ms, P = 554 (unit power), E = 3.324 (unit energy)

operation period 2 : t = 13 ms, P = 554 (unit power), E = 7.202 (unit energy)

operation period 3 : t = 8 ms, P = 554 (unit power), E = 4.432 (unit energy)

operation period 4 : t = 16 ms, P = 554 (unit power), E = 8.864 (unit energy)

operation period 5 : t = 8 ms, P = 554 (unit power), E = 4.432 (unit energy)

operation period 6 : t = 14 ms, P = 554 (unit power), E = 7.756 (unit energy)

operation period 7 : t = 18 ms, P = 554 (unit power), E = 9.972 (unit energy)

operation period 8 : t = 2 ms, P = 554 (unit power), E = 1.108 (unit energy)

operation period 9 : t = 18 ms, P = 554 (unit power), E = 9.972 (unit energy)

operation period 10 : t = 4 ms, P = 554 (unit power), E = 2.216 (unit energy)

```
    This is the result AFTER scheduling!
****************************************************
The generated scheduling settings
      OP 1   OP 2   OP 3   OP 4   OP 5   OP 6   OP 7   OP 8   OP 9   OP 10
 B1   5      3.3    0      5      5      5      3.3    5      0      5
 B2   5      2.4    0      3.3    5      5      5      5      3.3    3.3
 B3   5      3.3    3.3    5      5      5      3.3    0      2.4    3.3
 B4   3.3    5      5      3.3    5      5      5      0      5      5
 B5   0      3.3    5      3.3    3.3    2.4    5      0      2.4    5

    power consumption P_ij (unit power)
      OP 1   OP 2   OP 3   OP 4   OP 5   OP 6   OP 7   OP 8   OP 9   OP 10
 B1   123    26.79  0      123    123    123    26.79  123    0      123
 B2   36     2.074  0      7.841  36     36     36     36     7.841  7.841
 B3   200    43.56  43.56  200    200    200    43.56  0      11.52  43.56
 B4   8.712  40     40     8.712  40     40     40     0      40     40
 B5   0      33.76  155    33.76  33.76  8.928  155    0      8.928  155

    total execution time of the system is 107 ms
E_total = 29.61(unit energy)
P_rms = 304.4 (unit power)
P_avg = 276.7 (unit power)
P_max = 432.8 (unit power)


    Parameters related to each operation period P_i
operation period 1 : t = 6 ms, P = 367.7 (unit power), E = 2.206 (unit energy)
operation period 2 : t = 13 ms, P = 146.2 (unit power), E = 1.9 (unit energy)
operation period 3 : t = 8 ms, P = 238.6 (unit power), E = 1.908 (unit energy)
operation period 4 : t = 16 ms, P = 373.3 (unit power), E = 5.973 (unit energy)
operation period 5 : t = 8 ms, P = 432.8 (unit power), E = 3.462 (unit energy)
operation period 6 : t = 14 ms, P = 407.9 (unit power), E = 5.711 (unit energy)
operation period 7 : t = 18 ms, P = 301.3 (unit power), E = 5.424 (unit energy)
operation period 8 : t = 2 ms, P = 159 (unit power), E = 0.318 (unit energy)
operation period 9 : t = 18 ms, P = 68.29 (unit power), E = 1.229 (unit energy)
operation period 10 : t = 4 ms, P = 369.4 (unit power), E = 1.478 (unit energy)
```

Appendix B: Power-related EDA Tools

Carleton U.	Power-Profiler - Behavioral Synthesis
CMU	PowerScope - Profile energy usage in mobile applications
IMEC	Atomium - PowerEscape prototype
MIT	JouleTrack - web-based software profiling tool
Northwestern U.	PACT - automated compiler making tradeoffs between performance, time, and power
Penn State	SimplePower - Architecture level
	SoftWatt - system power simulator
Princeton U.	Wattch - Architecture-level performance simulator with power model
UCI	IMPACCT - System-level hardware/software codesign tool
ASC	Power Synthesizer - Behavioral Level
BullDAST	PowerChecker - RTL power estimation and optimization
Cadence	VoltageStorm - Transistor-accurate power grid analysis product
ChipVision	ORINOCO - Behavioral Level power estimation and optimization
IBM	PowerTimer - Architecture-level
Intel	TEM^{2}P^{2}EST - Micro-architecture power/performance analysis simulator
PowerEscape	PowerEscape Analyer/Cache - Algorithm Level Power Optimization
Sequence	PowerTheater - RTL power design/optimization, gate-level power verification
Synopsys	Galaxy Design Platform - RTL-to-GDSII solution for dynamic and leakage power
VeriTools	Power Tool - PLI based power calculation tool

Appendix C: Publications

- Li-Chuan Weng, XiaoJun Wang, and Bin Liu, *Survey of Dynamic Power Management*, The 3rd IEEE International Workshop on System-on-Chip for Real-Time Applications (IWSOC 2003), pages 48-52, 30 June - 2nd July, 2003, Calgary, Albert, Canada
- Li-Chuan Weng, XiaoJun Wang, and Bin Liu, *Scheduling for Low Power*, International Conference of PhD Students, pages 255-259, 11-17 August 2003, Miskolc, Hungary
- Li-Chuan Weng, XiaoJun Wang, Alan P. Su, and Bin Liu, *Low Power Heuristic Block-level Voltage/Frequency Scheduling*, The 2004 International Conference on VLSI (VLSI-04), pages 577-581, June 2004, Las Vegas, Nevada, USA

Biographical Sketch

Ms. Li-Chuan Weng was born in HsinChu, Taiwan. She majored Electrical and Control Engineering (ECE) at National Chiao-Tung University (NCTU) and got her bachelor degree in 2001. Ms. Weng is currently working on her Master degree in School of Electronic Engineering (EE) in Dublin City University (DCU), Ireland. Since 2003, she works as an EDA engineer in SoC Technology Center (STC), Industrial Technology Research Institute (ITRI), Taiwan. Her research interests include CAD and SoC design. Her current research focuses on low power scheduling. She is a member of the ACM and the IEEE.