

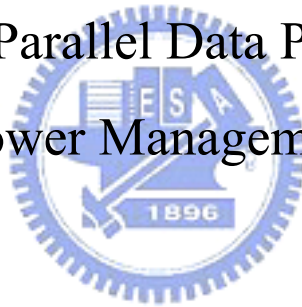
國立交通大學

電信工程學系碩士班

碩士論文

整合動態功率管理之平行資料路徑設計

The Design of a Parallel Data Path with Dynamic
Power Management



研究生：王志軒

指導教授：闕河鳴 博士

中華民國九十三年七月

整合動態功率管理之平行資料路徑設計

The Design of a Parallel Data Path with Dynamic Power
Management

研究生：王志軒

Student : Jyh-Shiuan Wang

指導教授：闕河鳴 博士

Advisor : Dr.Herming Chiueh

國立交通大學
電信工程學系碩士班
碩士論文



A Thesis

Submitted to Department of Communication Engineering
College of Electrical Engineering and Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Communication Engineering

July 2004

Hsinchu, Taiwan

中華民國九十三年七月

整合動態功率管理之平行資料路徑設計

研究生:王志軒

指導教授: 闕河鳴

國立交通大學

電信工程研究所

摘要

極長指令字組(VLIW)處理器是一種包含許多運算功能單元(functional unit)並可多重發派指令的處理器，許多 VLIW 處理器都可提供一些在平台式晶片系統設計(Platform-Based SoC Design)和可重置性的架構(Reconfigurable Architecture)中所需的基頻數位訊號處理(DSP)的運算[1-3]，如果使用 VLIW 處理器作為一個平台式晶片系統設計的中心微處理器，那麼某些 DSP 硬體的將可被微處理器所取代而減少硬體設計的複雜度；然而一般的 VLIW 處理器往往存在著低指令密度和低硬體使用率的缺點，而這些缺點也造成嚴重的功率消耗，低指令密度會造成在管線暫存器的功率浪費，低硬體使用率則使得某些運算功能單元閒置而造成靜態功率的浪費。在這一篇文章中，我們針對低指令密度和低硬體使用率所造成的功率議題，挑選了闡控時脈和電壓分離這兩種動態功率管理技術，並整合在一顆相似於一具三發派指令的 VLIW 處理器的平行資料路徑；除此之外，我們也利用現有的 EDA 軟體，發展出一套可用來實現闡控時脈和電壓分離的設計流程，本論文中所有的設計和驗證都是使用 UMC 0.18 um CMOS 製程參數來完成，而如同我們所預期的，在最後功率模擬分析的結果中發現，除了

此平行資料路徑的功率消耗有明顯的改善，其效能和功率消耗之間也變得可微調，而具有此特性的資料路徑之 VLIW 處理器是更適合應用在一個平台式晶片系統設計和可重置性的架構。



The Design of a Parallel Data Path with Dynamic Power Management

Student: Jyhshiuang Wang

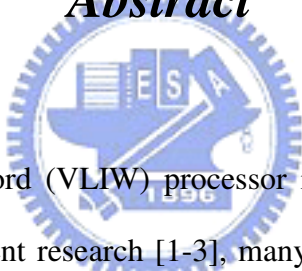
Advisor: Herming Chiueh

Institute of Communication Engineering

National Chiao Tung University

Hsinchu, Taiwan 30050

Abstract



Very long instruction word (VLIW) processor is a multi-issue processor with many functional units. In recent research [1-3], many VLIW processors can provide the functions of baseband digital signal processing (DSP) calculations such as discrete cosine transform (DCT), finite impulse response (FIR) filter, and motion estimation. These general purpose DSP calculations are very common in platform-based design and reconfigurable architecture for wireless communications and multi-media applications. So if a VLIW processor is applied as the microprocessor in platform-based design or reconfigurable architecture, some general purpose DSP blocks can be replaced by VLIW processor and the design complexity of hardware in platform-based design and reconfigurable architecture can be reduced. However, power dissipation in VLIW processor can be a serious problem due to the low code density and hardware utilization. In this thesis we applied dynamic power management techniques in the 16-bits parallel data path which is similar to a data path

in a three-issue VLIW processor to reduce the overhead of power dissipation. Two dynamic power managements: clock gating and voltage separation are chosen since they are suitable for the parallel data path. Clock gating can reduce the power dissipation caused by low code density in pipeline registers and voltage separation can reduce power wastage in functional units that is caused by low hardware utilization. Furthermore, we also explored an appropriate design flow for these two power managements with current electronic design automation (EDA) tool. All the design and verification are completed with UMC 0.18 um process. Power analysis is accomplished with five test benches. The power dissipation of the parallel data path is successfully reduced and the power and performance of the data path become scalable as we expected. A VLIW processor with such a data path is provided as a good option for microprocessor in platform-based design and reconfigurable architecture.



誌謝

本篇論文可以順利完成，首先要感謝我的指導教授闕河鳴博士，這兩年來給於我許多寶貴的建議和指正，每當在研究的過程中遇到瓶頸時，老師都能耐心的引導，使我能順利解決所遇到的問題。另外，老師平日培養學生獨立思考與分析能力，更讓我理解到做研究時正確的態度與方法。志軒由衷感激。

其次我要感謝我的父母和兄長，一路走來陪我渡過許多難關，最後我要感謝我的同窗詹謹鴻、劉明治、吳書豪、張智閔、賴昭宏、李佳勤、鍾文狀，李昭宏與我一同努力奮鬥，並在研究所這段期間給予我的幫助及啟發。此外也要感謝學長張祐誠，大學同學黃彥超，和在晶片系統設計實驗室的學弟學妹們，有了各位的支持與相伴，讓我在碩士生涯中，能更過的多采多姿，也讓我的研究生活增色不少。

最後再一次感謝你們，謝謝。



Content:

Chapter 1 Introduction.....	1
Chapter 2 Background and Related Research.....	7
2.1 Architecture of the Parallel Data Path.....	7
2.2 Dynamic Power Management.....	12
2.2.1 Clock Gating	12
2.2.2 Voltage Separation	14
Chapter 3 System Level Design	19
3.1 Parallel Data Path with Dynamic Power Management.....	19
3.2 Instruction Set Design.....	21
3.3 Clock Gating Implementation.....	22
3.4 Voltage Separation Implementation	24
Chapter 4 Design Flow and Simulation Result	28
4.1 CAD Flow	28
4.2 Test Configuration.....	38
4.3 Simulation Result.....	39
4.3.1 Clock Gating Simulation.....	39
4.3.2 Test Bench Simulation	46
Chapter 5 Power Analysis	51
5.1 Benchmark Definition.....	51
5.1.1 Matrix Addition.....	51
5.1.2 Idle Process	53
5.1.4 Matrix Calculation with Functional Unit MAC.....	54
5.1.5 Matrix Calculation with Functional Unit ALU.....	55
5.2 Simulation Result.....	57
5.2.1 Matrix Addition.....	57
5.2.2 Idle Process	59
5.2.3 Matrix Calculation with All Functional Units	61
5.2.4 Matrix Calculation with Functional Unit MAC.....	64
5.2.5 Matrix Calculation with Functional Unit ALU.....	66
5.2.6 Comparison	69
5.3 Summary	72
Chapter 6 Conclusion	74
<i>Bibliography:</i>.....	76
Appendix A Instruction set summary	79
Appendix B Instruction format	80

List of Figures:

Figure 1.1. An example of a platform-based design	2
Figure 1.2.a. An example of reconfigurable architecture.....	3
Figure 1.3. A reconfigurable architecture with VLIW processor.	4
Figure 2.1.1. Block diagram of the parallel data path.....	9
Figure 2.1.2. Normal pipeline operation	10
Figure 2.1.3. Pipeline operation when data dependence occurs	11
Figure 2.1.4. A not-taken branch instruction operation	12
Figure 2.2.1 Illustration of clock gating.....	13
Figure 2.2.2. Timing-critical voltage islands [19].....	15
Figure 3.1.1. The parallel data path with power management unit.....	20
Figure 3.2.1. Normal instruction format	21
Figure 3.2.2. Instruction with power control bits.....	21
Figure 3.3.1. Clock gating implementation	22
Figure 3.3.2. Clock gating on the pipeline registers of ALU	23
Figure 3.3.3. Clock gating operation timing diagram.....	23
Figure 3.4.1. Voltage Separations	25
Figure 3.4.2. Implementation of voltage separation on ALU	25
Figure 3.4.3. Implementation of voltage separation with high vt transistors	26
Figure 4.1.1. Cell-base design flow	29
Figure 4.1.2. Basic auto place and route flow.....	30
Figure 4.1.3. Design flow of this implementation	31
Figure 4.1.4. A clock gating example	32
Figure 4.1.5. Timing diagram of Figure 4.1.4.....	33
Figure 4.1.6. Design flow for this implementation of auto P&R.....	34
Figure 4.1.7. Layout of the parallel data path with dynamic power management.....	36
Figure 4.3.1. Simulation of instruction fetch	40
Figure 4.3.2. The simulation of pipeline operation with power management units	45
Figure 4.3.3. Matrix A and matrix B.....	46
Figure 4.3.4 The operation flow of this chip.....	47
Figure 4.3.5 The operation of WRITE mode	48
Figure 4.3.6 Insertion of matrix A	48
Figure 4.3.7 Insertion of matrix B	49
Figure 4.3.8. EXECUTION mode	49
Figure 4.3.9. Matrix A after execution.....	50
Figure 4.3.10. TEST mode.....	50
Figure 5.2.1. Total power dissipation of matrix addition.....	58

Figure 5.2.2. The power dissipation of the three functional units58
Figure 5.2.3. Total power dissipation of idle process60
Figure 5.2.4. Total power dissipation of the three functional units61
Figure 5.2.5 Total power dissipation of matrix calculation with all functional units ..63
Figure 5.2.6 Total power dissipation of the three functional units63
Figure 5.2.7. Total power dissipation of matrix calculation with MAC65
Figure 5.2.8. Total power dissipation of the three functional units66
Figure 5.2.9. Total power dissipation of matrix calculation with ALU68
Figure 5.2.10. Total power dissipation of the three functional units68
Figure 5.2.11. The power dissipation of matrix calculation70



List of Tables:

Table 2.1.1. The instructions corresponding to the three functional units	8
Table 2.1.2. Description of pipeline stage.....	9
Table 4.1.1. EDA tools used in this implementation.....	35
Table 4.1.2. Circuit summaries	35
Table 4.1.3. The definitions of the IO pads.....	37
Table 4.3.1. Definitions of the signals in Figure 4.3.1	41
Table 4.3.2 Definitions of the signals in Figure 4.3.2.....	42
Table 4.3.3. The assemble language of the program $A=A+B$	46
Table 5.1.1 The assemble language of Matrix Addition	52
Table 5.1.2. The assemble language of matrix calculation with all functional units ...	53
Table 5.1.3 The assemble language of matrix calculation with MAC.....	54
Table 5.1.4. The assemble language of calculation with ALU	55
Table 5.2.1. Power distribution of matrix addition (without power management) :....	57
Table 5.2.2. Power distribution of matrix addition (with clock gating) :.....	57
Table 5.2.3. The power analysis result of matrix addition.....	59
Table 5.2.4. Power distribution of idle process (without power management) :	59
Table 5.2.5. Power distribution of idle process (with clock gating) :	59
Table 5.2.6. The power analysis result of idle process	61
Table 5.2.7. Power distribution of matrix calculation (without P.M.) :	62
Table 5.2.8. Power distribution of matrix calculation (with clock gating) :	62
Table 5.2.9. The power analysis of matrix calculation with all functional units	64
Table 5.2.10 Power distribution of matrix calculation with MAC (without P.M.) :	64
Table 5.2.11 Power distribution of matrix calculation with MAC (with C.G.) :.....	64
Table 5.2.12. The power analysis of calculation with MAC.....	66
Table 5.2.13. Power distribution of matrix calculation with ALU (without P.M.)	67
Table 5.2.14. Power distribution of matrix calculation with ALU (with C.G)	67
Table 5.2.15. The power analysis of matrix calculation with ALU	68

Chapter 1 Introduction

Platform-based design is a design approach of reuse methods for embedded system in SoC[4][5][6]. Generally, a platform-based design contains common architecture and the supporting technologies (Intellectual Property libraries and develop tools) [5]. The common architectures typically include a microprocessor, memory and the communication bus. The Intellectual Property (IP) of the IP libraries are all designed with the same microprocessor and bus protocol. The initial configuration formed by the common architectures can be extended with functional units (F.U.) for different application. The functional units, are embedded DSP blocks such as discrete cosine transform (DCT), fast fourier transform (FFT), finite impulse response filter (FIR) filter, motion estimation...etc, can be available from the IP libraries of the platform or designed by the application developer. For example, the common architectures with DCT and motion estimation can be used for video application, or with FFT and FIR filter can be used for wireless communication. An example of a platform-based design is shown in Figure 1.1. The microprocessor, memory and bus are the common architecture in this platform. The other devices include the functional units for application-specific extension, and other important hardware components such as I/O controller and bus bridge. The major character of platform-based design is to reduce the design time. Since all the devices are based on the same bus protocol and microprocessor, and they can be quickly integrated.

A platform-based design is accomplished by software and hardware co-design. Designer will partition the functionality of applications into the software and hardware and choose appropriate microprocessor and other hardware blocks. The software implementation ran in the microprocessor and other coprocessor, and the hardware part will be executed on the customized signal processing functional units.

In recent research [5], the microprocessor has become the most important elements in a platform-based design such as ARM [7] and Philips Nexperia [8].

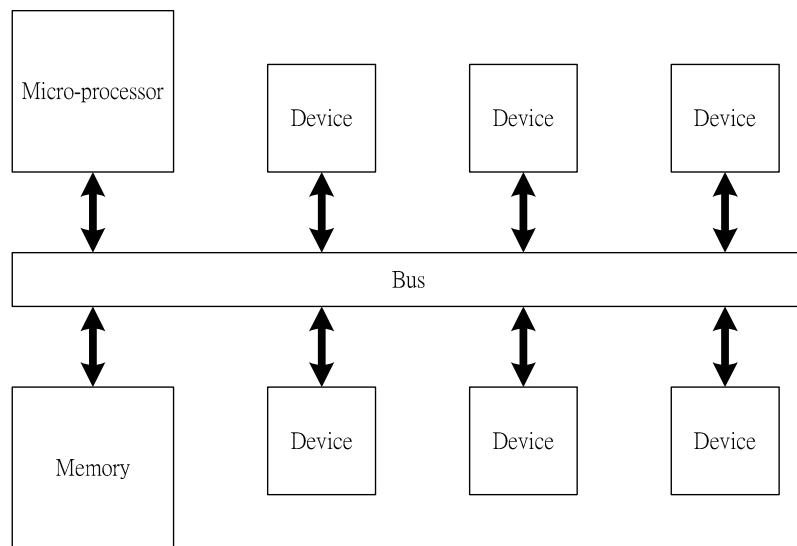


Figure 1.1. An example of a platform-based design

Some general purpose DSP blocks such as DCT, motion estimation, motion prediction, FIR filter, and viterbi decoder...etc, are common blocks for many applications. If a platform-based design includes those general DSP blocks, the platform-based design can be used to implement a reconfigurable architecture. Figure 1.2 gives an example. In Figure1.2.a, the platform is operating an application for 3G wireless communication where a viterbi decoder is required. If the platform will operate another application for MPEG decode where a DCT is required, user only has to reconfigure the data path of the hardware blocks and implement other software on the microprocessor as shown in Figure1.2.b. One advantage of reconfigurable architecture is that the used hardware and data path are reconfigurable. This advantage provides a great flexibility for wide different application. Furthermore, if a reconfigurable architecture includes power management unit, system can scale the power and performance of the reconfigurable architecture. When the required performance is high, more functional units on a reconfigurable architecture will be active and consume larger power. While the power consumption of the reconfigurable

architecture has to be reduced, system can reduce active hardware and turn them off with power management unit. So the power and performance of a reconfigurable architecture can be scalable if there is power management unit on the architecture.

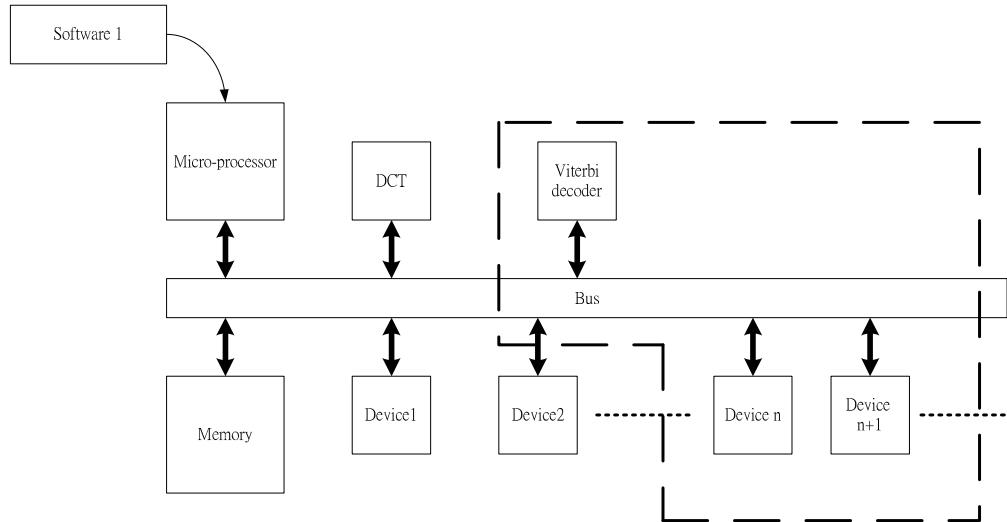


Figure 1.2.a. An example of reconfigurable architecture

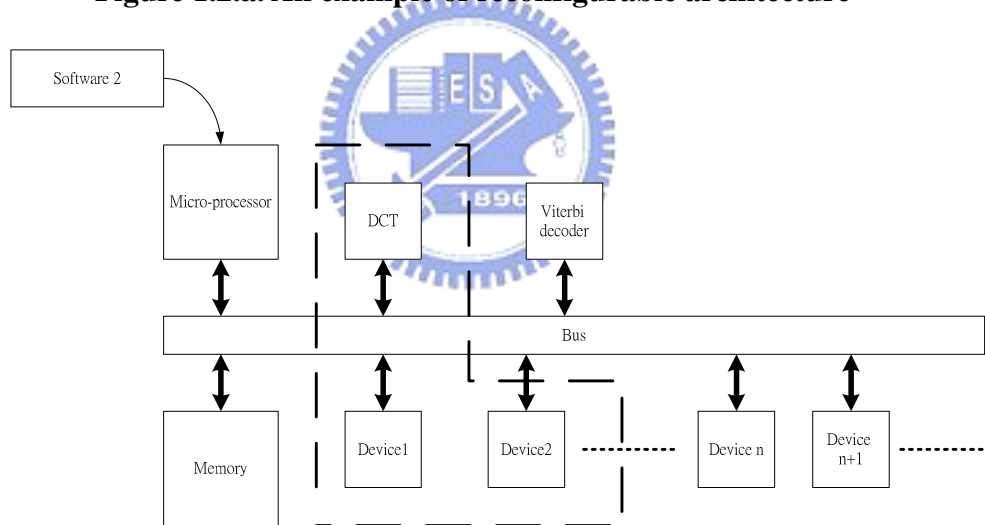


Figure 1.2.b. An example of reconfigurable architecture

Very long instruction word (VLIW) processors is a kind of multi-issue processor and is suitable of high-performance real-time DSP application [9]. In recent research [1][2][3][9], many VLIW processors can provide the functions of those general DSP blocks. So a VLIW processor can be used to replace the microprocessor and some general DSP blocks in a reconfigurable architecture and this is shown in Figure 1.3. If the VLIW processor is well-defined, such a design shown in Figure 1.3 can simplify

the design of reconfigurable architecture since the complexity of the interconnection in the hardware blocks is reduced.

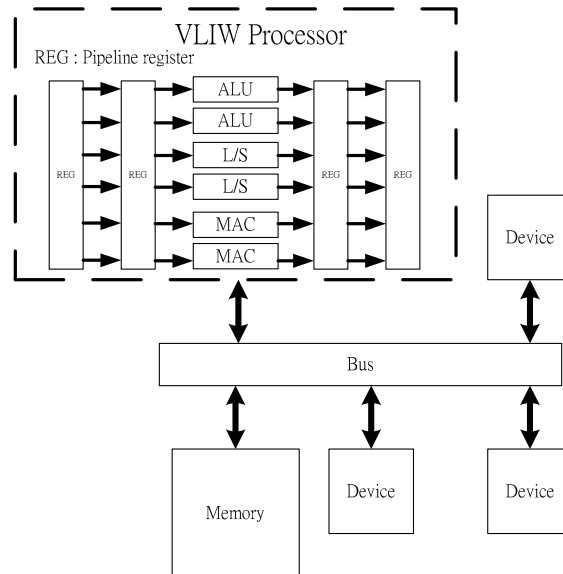


Figure 1.3. A reconfigurable architecture with VLIW processor.

However, some disadvantages such as low code density and low hardware utilization exist in general VLIW processors [1][9]. Very long instruction word is the characteristic of VLIW processor and pipeline registers will be longer. Low code density means that the power wasted in pipeline registers and low hardware utilization also means large power wastage in unused functional units. The problem in power wastage will especially be serious when executing idle process. When the general DSP functional units are not replaced by VLIW processor, the unused one can be turned off by power management unit on the reconfigurable architecture, and the power and performance of reconfigurable architecture can be scalable. However, while these general DSP blocks are replaced by VLIW processor, the reconfigurable architecture will lose the scalability of power and performance due to the power issue in VLIW processor. To overcome such a problem it is necessary to apply power management on VLIW processor. If power management method can be applied in VLIW processor and overcome those power issue, VLIW processor is able to provide

a good option for microprocessor in reconfigurable architecture.

In recent years, some power management methods for SoC design are proposed, such as Variable Threshold Voltage CMOS (VTCMOS) [10][11], Multi-Threshold CMOS technology (MTCMOS) [12][13][14][15], Clock Gating [16][17], Dynamic Voltage Scaling (DVS) [18][19][20], Voltage Islands [21][22], Adaptive Supply Voltage and Body Bias (ASB) [23][24][25]. From the ideas of these power management methods, the clock gating and voltage separation are suitable to be applied in VLIW processor. Clock gating is a useful method to reduce dynamic power dissipation by reducing unnecessary clock switching. Low code density will cause a lot of power wastage in VLIW processor due to large unnecessary clock switching on pipeline registers. So clock gating can be used to reduce the overhead of low code density. Furthermore, low hardware utilization rate in VLIW processor means that many functional units are idle during program operation. Those idle functional units will cause large static power consumption in advance process. If the power supply of the functional units in VLIW processor can be separated and managed individually, the system can turn off the power supply of the unused functional units and reduce the overhead in static power consumption due to the low hardware utilization.

In this thesis we designed a 16-bits parallel data path which contain three common functional units in VLIW processor, they are ALU, Load/Store and MAC [2][3][9]. Then the data path is used to simulate the data flow in a three-issue VLIW processor and apply clock gating and voltage separation on this data path. Clock gating will disable the unnecessary clock switching on the pipeline registers and voltage separation will turn off the power supply of idle functional units. So even the code density and hardware utilization are low, there will not be a great overhead in power dissipation. Furthermore, the performance and power of this data path will become scalable after applying these two power management mechanisms. The power

dissipation will be according to the required performance. If the performance requirement is high, all the functional units in this data path will all be used and the power dissipation will be high. If the required performance is normal, only some of the functional units are used and the unnecessary power dissipation on idle elements will be reduced by the dynamic power management mechanisms. So the power and performance of the parallel data path will depend on the system requirement. This characteristic of power and performance scalable will be suitable for a data path of a powerful VLIW processor in a reconfigurable architecture.

This thesis focused on the power management design and implementation in the parallel data path which is for test vehicle. The remaining of this thesis is organized as the following.

In Chapter 2 the background and related research are presented. The architecture of the parallel data path for test vehicle is presented. The clock gating and voltage separation for dynamic power management are briefly explained.

In Chapter 3 the detail design of all the system is presented. The implementation of power management and the impact on the instruction set architecture design of the parallel data path are presented.

In Chapter 4 a detail electronic design automation (EDA) design flow of implementation will be presented.

In Chapter 5 the power simulation of the parallel data path with clock gating and voltage separation is presented.

In Chapter 6 a conclusion is addressed.

Chapter 2 Background and Related Research

In this chapter, the architecture of the parallel data path is presented which is the test vehicle for the dynamic power management. In addition, the applied dynamic power management methods will be explained. The functional units, the instruction sets design and the pipeline operation of the parallel data path are shown in Section 2.1. The detail of the applied dynamic power management, clock gating and voltage separation, are presented in Section 2.2.

2.1 Architecture of the Parallel Data Path

The very long instruction word (VLIW) processor is one type of multi-issue processors. This is a method to decrease the cycle per instruction (CPI) by issuing a fixed number of instructions per cycle. In VLIW processor, the instruction parallelism (ILP) exploiting and data hazards detection are accomplished by compiler. So the instruction issue and the instruction scheduling are all static and the hardware cost is much less than the other type of multi-issue processors such as superscalar. In the recent years, VLIW processor is broadly applied in the market. Such as Trimedia TM32 [26] and Transmeta Crusoe [27] are all VLIW type processor. And Philips Nexperia, one of the leading platform for multimedia, is based on MIPS with a VLIW processor. The major advantage of VLIW processor is the low cost of hardware. But on the other hand the complexity of compiler is high, this is the trade-off in VLIW processor.

In this research, the test vehicle for dynamic power management is a 16-bits parallel data path with three functional units. Like a VLIW processor, this data path issues three instructions per clock cycle and the ILP is explored by compiler. When the data path is ready to operate, who will compile program and write the instructions into the instruction memory. Then the data path will fetch the instruction form

instruction memory every clock cycle with a program counter. All the instructions are executed in five stage pipeline which are instruction fetch (IF), instruction decode (ID), execution (EX), memory access (MEM), and write back (WB) respectively. The three functional units are Arithmetic logic unit (ALU), load/store (L/S) unit, and multiply-accumulator (MAC) unit. ALU can execute some simple arithmetic and control instructions. The load/store unit consists of an adder which is used to calculate data address in memory. It can also execute add and subtract operation. MAC unit contains a multiplier and an adder. It can be used to execute accumulator and multiplier operation. The block diagram of the data path is shown in Figure 2.1.1. The instructions corresponding to the three functional units are summarized in Table 2.1.1. The pipeline stages in Figure 2.2.1 are listed in Table 2.1.2. In the following paragraph we will introduce normal pipeline operation and analyzing two special cases that impact the compiler and pipeline operation in this data path.

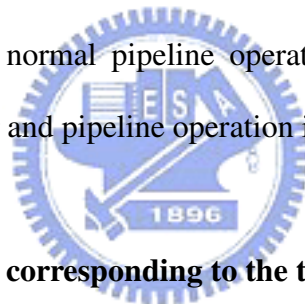


Table 2.1.1. The instructions corresponding to the three functional units

Functional Unit	Instructions
ALU	ADD, SUB, AND, OR, XOR, SLL, SRL, SRA, SLT, JR, JUMP, JAL, BEQ, HALT
LOAD / STORE	ADDI, SUBI, LOAD, STORE
MAC	MAC, MUL

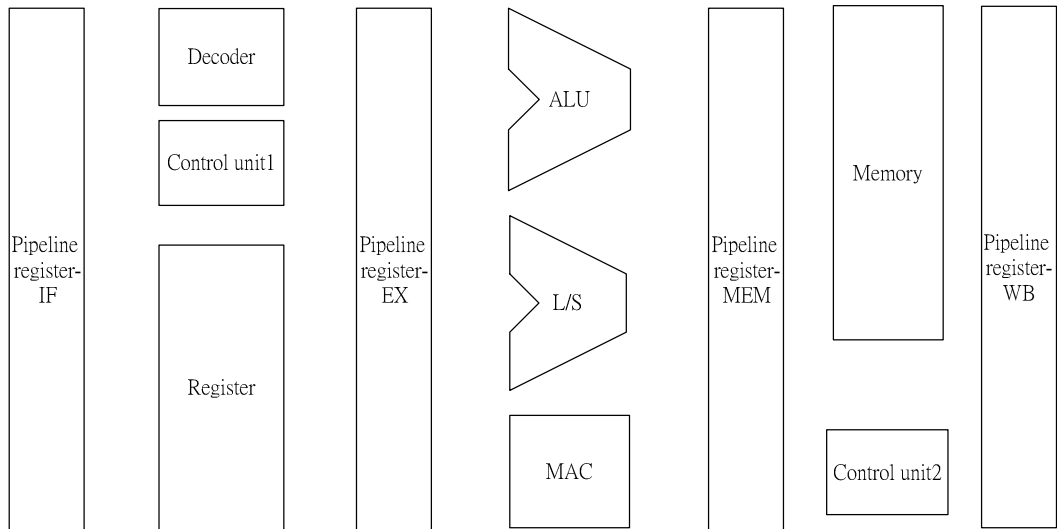


Figure 2.1.1. Block diagram of the parallel data path

Table 2.1.2. Description of pipeline stage

Pipeline register – IF	The first stage in pipeline that is used to save the instruction fetched from memory.
Decoder	That decode the OP code of instruction in Pipeline register – IF, and decide the register file accessing.
Control unit1	That controls the access to register file of every instruction.
Register file	A 16 x 32 register which save data during program execution.
Pipeline register – EX	That save the instruction after decode and the necessary data from register file.
Pipeline register – MEM	That saves the result of execution for memory access.
Memory	This is a 16 x 256 one port SRAM.
Control unit2	That controls the access to register file of every instruction

Pipeline register – WB	That save the data after execution and the data from memory for writing back to Register file
------------------------	---

An instruction will be saved in Pipeline register – IF at first cycle when it is fetched from instruction memory. Decoder will decode the op code of the instruction and read out the necessary data from register file at second cycle. Then the functional units will deal with the data with the result of decoding at third cycle. At the next cycle, this data path will access memory if a load/store or control instruction is fetched. At the final cycle the data path will write back the data generated in third and forth cycle to register file. Every instruction is executed in five clock cycles and the timing diagram is shown in Figure 2.1.2.

	Clock cycle					
Instruction	1	2	3	4	5	6
Instruction i	IF	ID	EX	MEN	WB	
Instruction i+1		IF	ID	EX	MEN	WB
Instruction i+2			IF	ID	EX	MEN
Instruction i+3				IF	ID	EX

Figure 2.1.2. Normal pipeline operation

Figure 2.1.2 show the timing diagram of the data path when there is no data dependence. If there are data dependences between two continue instructions, these two instructions can't be fetched continually or the latter one will read the wrong data. Data dependence will be detected when compiling program, and the compiler will schedule the instruction to avoid this situation. But if the data dependence still exist after scheduling. The compiler will insert stall instructions. An example is shown bellow:

Instruction i : ADDI R1, R2, R3

Instruction i+1 : ADDI R4, R1, R5

Instruction i produces a result that will be used by Instruction i+1. If Instruction i+1 is decoded before Instruction i write back the result, then a data hazard will happen. For avoiding this situation two stall instructions should be inserted between these two instructions by compiler. The instruction execution timing diagram is depicted in Figure 2.1.3.

Clock cycle						
Instruction	1	2	3	4	5	6
Instruction i	IF	ID	EX	MEN	WB	
Stall Instruction		Stall	Stall	Stall	Stall	Stall
Stall Instruction			Stall	Stall	Stall	Stall
Instruction i+1				IF	ID	EX

Figure 2.1.3. Pipeline operation when data dependence occurs

Beside data dependence, there is still one case that compiler will insert stall instruction when compiling program. That is the situation when a control instruction like branch is generated. A control instruction may change the program counter while a control instruction is fetched and this will be known in ID stage. So before a control instruction is decoded the next instruction shouldn't be fetched. Figure 2.1.4 show the timing diagram of this case, the compiler will also insert a stall instruction after branch instruction.

Clock cycle						
Instruction	1	2	3	4	5	6
Branch	IF	ID	EX	MEN	WB	
Stall Instruction		Stall	Stall	Stall	Stall	Stall
Instruction i+1			IF	ID	EX	MEN
Instruction i+2				IF	ID	EX

Figure 2.1.4. A not-taken branch instruction operation

2.2 Dynamic Power Management

The concept of dynamic power management in proposed design is to provide a scalable performance according to a real-time system requirement. In such design, some dynamic power management mechanisms such as multi-threshold CMOS (MTCMOS) technology [10], clock gating [14], voltage islands [19] [20], are proposed in recent years. We use two power management methods, clock gating and voltage separation, which are suitable for parallel data path in VLIW processor. Clock gating is an efficient solution to reduce the power dissipation caused by clock and voltage separation is a sub-step of voltage islands. In the following section, we will briefly explain those two power management.

2.2.1 Clock Gating

In the modern high performance VLSI design, the power dissipation due to clock tree is always the domination of dynamic power. This is because the clock tree represents a very large load due to that the clock signal switches all the time. As the incensement in operation frequency, the power consumed by clock can grow ever larger. But in fact, a portion of this power consumption is wasted, and this is caused

by unnecessary switching activity. For example, if the input data and output data of a flip-flop are identical, the switching of clock just wastes power. Actually, not all the clocked elements in a chip change its data all the clock cycle. And even when the data in a storage element remains unchanged, the switching of clock consumes significant power.

In a digital design, if the clock signal can be controlled, then the power wastage caused by unnecessary switching can be saved. A popular and effective way to do this is clock gating which is one method of dynamic power management and can eliminate unnecessary switching by gating clock signal with qualifying signals. For example, if the gated clock is through an AND gate which is shown in Figure 2.2.1. Only when something useful is computed in a given clock cycle, the qualifying signal will be set to high or it remains low. Then the clock to a clocked element is controlled, and the unnecessary clock switching can be avoided.

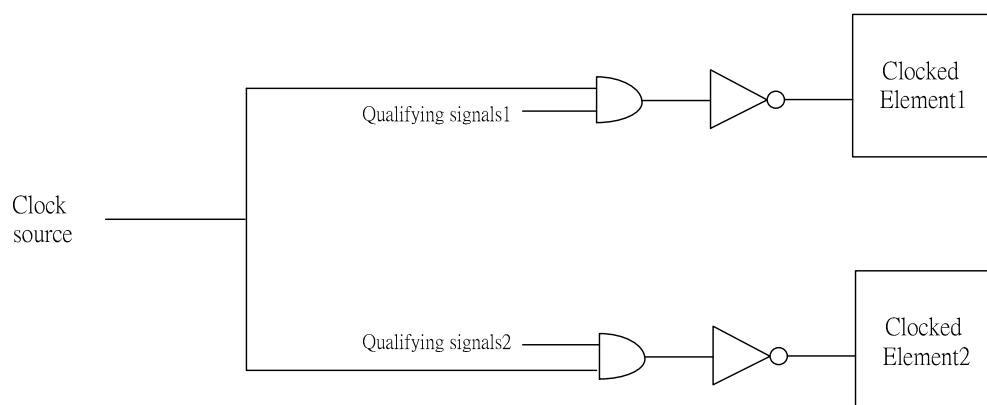


Figure 2.2.1 Illustration of clock gating

2.2.2 Voltage Separation

Voltage islands is addressed by IBM in 2002 [19]. It is a dynamic power management method and is used to optimize the power supply of individual functional units in SoC design. Voltage separation is a sub-step of voltage islands and can be accomplished with current EDA tools. In this thesis, we will apply voltage separation on the three functional units so that the power supply of these functional units can be managed individually. In this section, we will briefly explain voltage islands and voltage separation.

In early years, large functional units were not integrated on a single chip, so that different functional units could be supplied by different voltage level and the power supply could be optimized. However, with the era of SoC designs come, more and more functional units can be integrated on a single chip. As a result, if a chip is still supplied by fix level voltage. The chip will lose the optimization and the flexibility of the power supply in hybrid solution.

The voltage islands technique is to supply multi-level voltage in a single chip. Commonly a complex SoC design consists of a number of functional units, but not all of which always active at any given time. For achieving the optimization in power supply, functional units in a SoC design will be separated into different islands according to its power characteristics and every island has its own power control unit. By this unit system can turn of the power supply if the corresponding island is idle, and turn on again when it's time to active. An example of voltage islands is shown in Figure 2.2.2 [19]. In this chip, the most performance-critical functional units need voltage supply of 1.2v to meet the require performance, and the rest elements in this chip including memory could meet the timing requirement by 1.0 voltage supply.

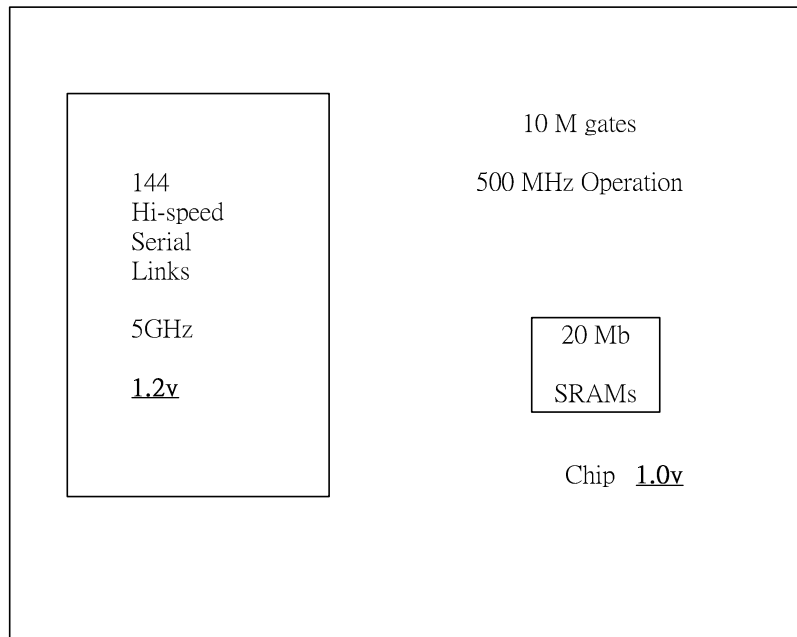


Figure 2.2.2. Timing-critical voltage islands [19].

Without new technology, the entire chip is supplied by 1.2v voltage due to the requirement of the high performance element. But that waste power, especially when the functional units only active in few percentage of all the operation time. This picture shows a solution of the problem described above. The high performance elements are separated into a voltage island and are supplied by individual voltage 1.2v. The rest of the design could be only supplied by voltage 1.0v. Due to the v_{dd}^2 term in the active power equation, the power of the design can be greatly reduced. Furthermore, voltage islands technique can also scale the power supply of each island. For an example, if an island only active for 2% of the operation time, then the power management unit of voltage islands can turn off the power supply at the rest 98% of the operation time. Then the power consumption can be optimized.

Voltage islands is a useful technology to reduce to power consumption of a SoC design. But there are some difficulty existing in implementation. In the following paragraph these issues will be introduced.

A traditional method of implementation for SoC design includes the following steps:

- **Architecture design**
- **Functional implementation (RTL)**
- **Synthesis and timing consideration**
- **Timing verification and simulation**
- **Floor planning and physical design**
- **Final timing verification and tape out**

When implementing a voltage islands design, there will be some additional consideration that will affect each step in the design flow. The following will briefly describe the requirement consideration.

- **Functional partition:** The designer should partition the functional components of the design into different islands according to its power characteristic, and each individual voltage island should be written into individual RTL module. This step should consider the performance requirement and the period of active and inactive of each functional component. For example, if component A and component B require the same voltage supply level, active in the same period and the duration of the inactive period exceed the minimum time for power on-to-off and off-to-on of switching. Then these two components could be classified into the same islands.
- **Synthesis and timing consideration:** Once the RTL has been completed, the designer can begin to synthesis the design. When synthesis and optimize the design, the effect caused by signal traveling different voltage level should be take into consideration. In the other words, the major problem is the delay calculation of the signal level shifting. Take the design shown in Figure 2.2.2 for example. The signal path from 1.0v level to 1.2 level or from 1.2v level to 1.0v level

should include a signal level shifter, and the designer should make sure that the delay caused by the level shifter won't affect the correction of operation. Furthermore, each island has its working voltage range, so that the timing calculation of each island should be different, and the voltage difference also represent difference in the calculation of clock skew. The problem described above almost caused by different islands and its different voltage level, and these effect should be taken into account when synthesis.

- **Physical planning and implementation:** In order to enable independent power supply, each island should be placed isolated from others, and designer should arrange the power supply of the power management and the power control of each island carefully. These two components shouldn't be influenced by the power switching of each island.
- **Logic simulation:** When simulating the logic functionality of a voltage islands design, the output of the power-off islands should be observed as unknown. The observed state is important for verifying the functionality of the power management and the power control in the power on-to-off and off-to-on switching.

The voltage islands technology provide an opportunity to reduce significant power consumption and achieve the power optimization for SoC's. But when implementing voltage islands there are some issue should be taken into consideration, and these considerations will affect the design flow. Some of each are even not supported by EDA tool, and should be accomplished by manufacturing test now. This is still a big challenge for voltage islands implementation.

In this thesis, we implement the functional partition and voltage separation which are sub-steps of voltage islands and can be accomplished with current EDA tool. In the parallel data path, the three functional units are separated into three different areas

in the same chip and their voltage supplies are individual connected. Thus, if any functional unit is idle, the power management units can detect the situation and turn off the power supply. The voltage separation operates in the two levels of on and off with the same voltage level supply, so the design can be accomplished without signal level shifting consideration. The detailed design will be presented in next chapter.



Chapter 3 System Level Design

In this Chapter, we will present the power management units for clock gating and voltage separation in this data path in Section 3.1. Actually, these power management units are portion of the pipeline registers and are designed by being based on instruction. So the instruction set architecture design is different from the traditional one and this will be presented in Section 3.2. Finally, the detail design of clock gating and voltage separation in this parallel data path will be presented in Section 3.3 and 3.4.

3.1 Parallel Data Path with Dynamic Power Management

The power consumption in a chip can be separated into two parts, the dynamic power and the static power. In this data path, the power consumption of clock tree is the domination of the dynamic power, and the power dissipated by the clock of the pipeline registers occupy about 50 percent among the power consumption caused by clock tree. Due to the low code density, the power wastage in the pipeline register can be large. In order to save the dynamic power dissipation of this part, we apply clock gating technique to control the clock for each pipeline registers. Furthermore, there are three major functional units in this parallel data path. When this data path is operating, not all functional units will be active or even some of which will keep in idle during operation due to the disadvantage of low hardware utilization. Then there will be power wasting if we keep supplying power to the idle functional unit. However, the implementation of clock gating in the pipeline register at ID stage will save the dynamic power consumption here. But leakage power remains, and can be significant in high performance technologies. Because of the characteristic of low hardware utilization in parallel data path and for better power optimization we implement voltage separation on the three functional units.

Figure 3.1.1 shows the block diagram of the parallel data path combined with power management unit for clock gating and voltage separation. The power management includes power management registers and gate control logic. The first two power management registers consist of 6 bits. The others only contain 3 bits. Every power management register uses 3 bits for clock gating. If a stall instruction is fetched, this mechanism will disable the clock with the control logic gate. And the extra 3 bits in the first two power management registers are used to control the power supply of the three functional units. If any functional unit is standing by, we can make use of the power management register to detect the situation and turn off the power supply for them. Thus we can save the static power of the functional units when those units are idle. The static power of the three functional units occupies about 20 percent of all the static power for the data path.

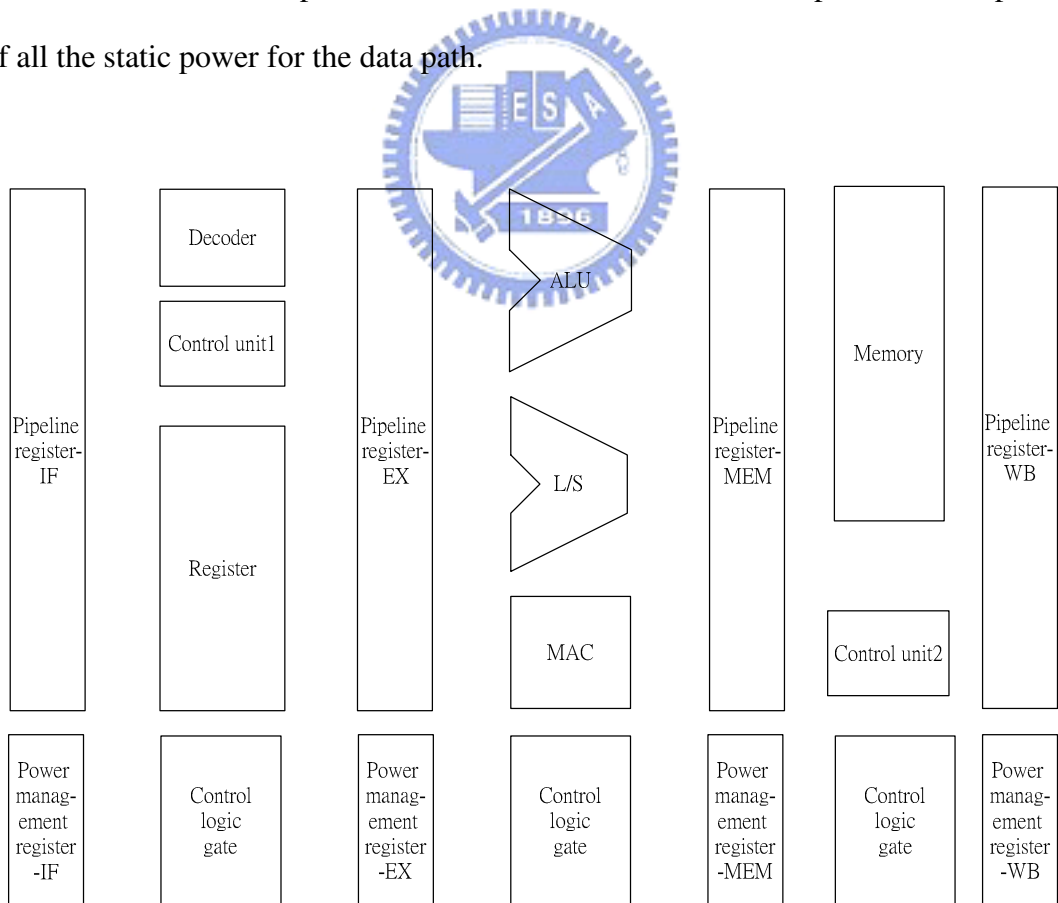


Figure 3.1.1. The parallel data path with power management unit

3.2 Instruction Set Design

This data path is a 3-address machine. The instruction format corresponding to a traditional 3-address machines includes only four fields: one that specify the operation, two that specify the source data address in register file, one that specify the address where to put the result. This format is shown in Figure 3.2.1. But the instruction format of this data path is a little different form the traditional, because we take the power consumption into consideration. One field that specifies the power control of clock gating and power supply for the functional units is contained in the instruction format of this data path beside the four fields mentioned above. This format is shown in Figure 3.2.2.

OP Code	Source and Destination Address
----------------	---------------------------------------

Figure 3.2.1. Normal instruction format

OP Code	Power Control	Source and Destination Address
----------------	----------------------	---------------------------------------

Figure 3.2.2. Instruction with power control bits

The field of power control contains two bits, of which one control the clock gating and the other control the power supply of the functional units. These two bits will be decided by compiler. If the Instruction $i+1$ is not a stall instruction, the bit of the power control for clock controlling in Instruction i will be set to 1, otherwise it will be set to 0. The decision for other bit will depend on the distribution of stall instructions after a program is fully compiled. If the number of the continuing stall instructions is large enough and the corresponding functional units will stand by long time enough, then compiler will set the bit for functional unit power supply control of these continuing stall instructions to 0. Otherwise if the functional unit doesn't stand by for long enough time, the bit will be set to 1. We have to emphasize that the power

control of each instruction are all decided by compiler, and the hardware only operate power management mechanism according to the data in power control field.

When instruction is executed, the part of power control will be saved in power management registers as mentioned in Section 3.1 and the power management of the data path will be according to the data in this part.

3.3 Clock Gating Implementation

In this parallel data path, the pipeline registers of the three functional units occupy fifty percent among the clocked elements. When a stall instruction is fetched, the clock switching on pipeline register only causes power wastage. Even turn off the clock in this period the executing result won't be influenced, but the power wastage can be saved. So we implement the clock gating to all the pipeline registers as shown in Figure 3.3.1. Figure 3.3.2 shows an example in functional unit ALU of this implementation.

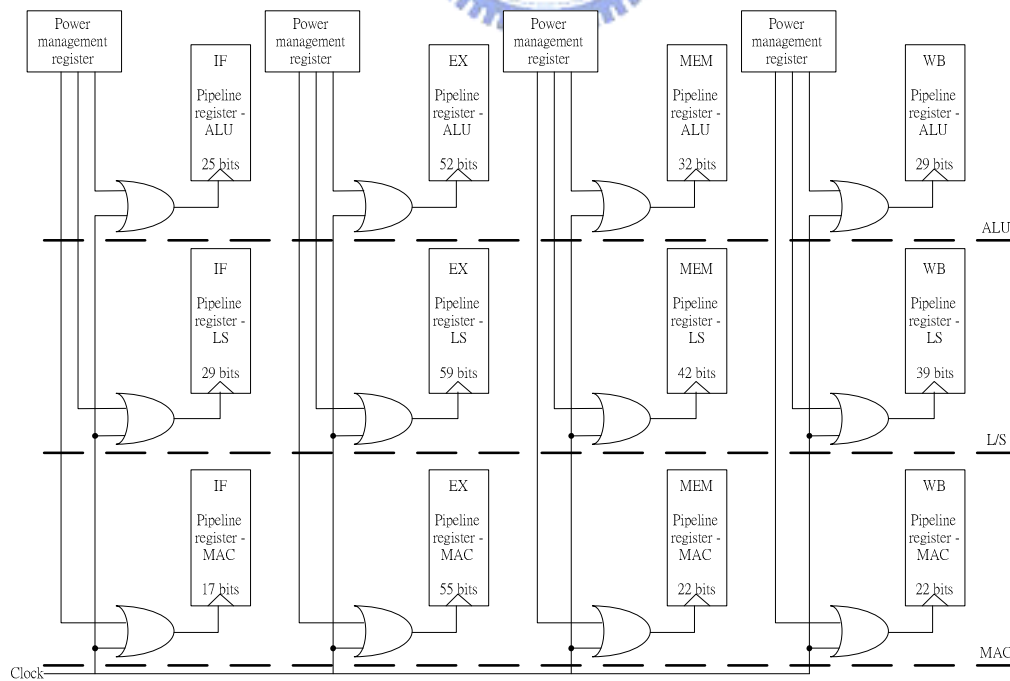


Figure 3.3.1. Clock gating implementation

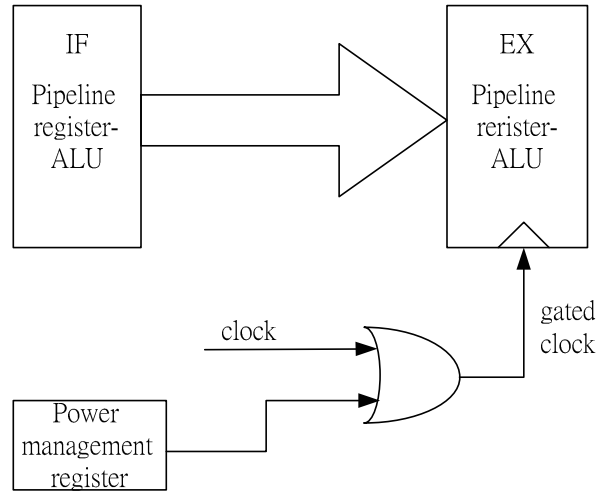


Figure 3.3.2. Clock gating on the pipeline registers of ALU

In Figure 3.3.2, the pipeline register-IF and pipeline register-EX is the first two stage in pipeline. The pipeline register IF is used to save the instruction fetched from the instruction memory and the pipeline register EX will save the result after instructions are decoded. The power management register will supply the qualifying signal and control the clock to the next pipeline register. The operation timing diagram of Figure 3.3.2 is shown in Figure 3.3.3.

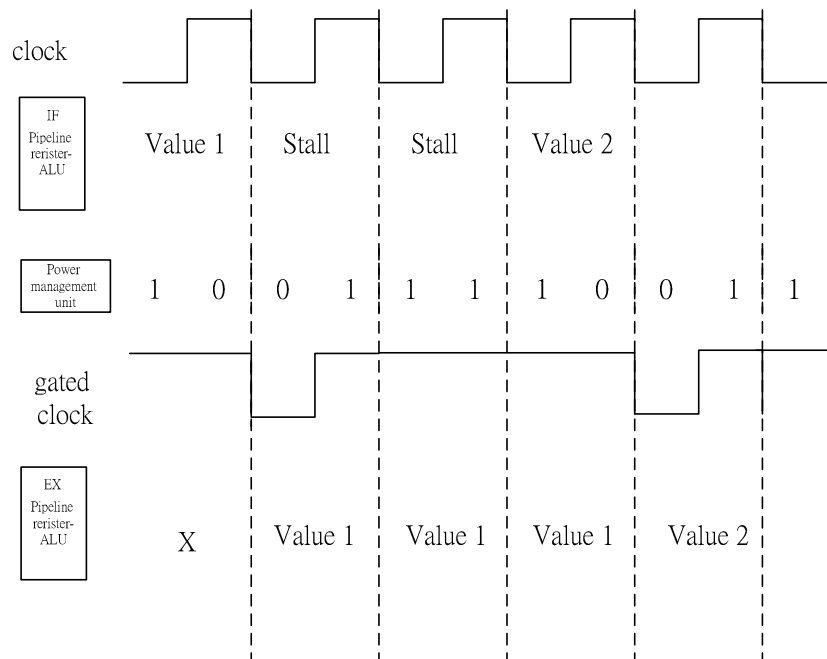


Figure 3.3.3. Clock gating operation timing diagram

As shown in Figure 3.3.3, all the pipeline registers are all triggered by negative edge of clock and all the power management registers are positive triggered. When the instruction fetched is not a stall, the power management register will be set to low. Then the clock signal can pass through the OR gate and the pipeline register EX will read the value at the next negative trigger. If the instruction fetched is a stall instruction, the power management register will be set to high and the signal to trigger pipeline register EX will remain in high state. Then the stall instruction will not be pipelined and the data in pipeline register EX will be kept until the next useful instruction is fetched.

3.4 Voltage Separation Implementation

The voltage separation in the three functional units is shown in Figure 3.4.1. In this chip, the three individual functional units are supplied by different voltage source and the power supply is controlled by the power management unit mentioned in Section 3.1. Figure 3.4.2 shows an example of functional unit ALU. The data path is an architecture with five pipeline stages. Before data arrives at functional unit, there are two pipeline stages and we will make use of these two stages to define the control of power supply. As shown in Figure 3.4.2, the pipeline register is used to save the data during instruction execution. The 2 flip-flop of power management register-IF and power management register-EX will arrange the state that will control the power supply. Only the state “00” the power supply of ALU will be turned off.

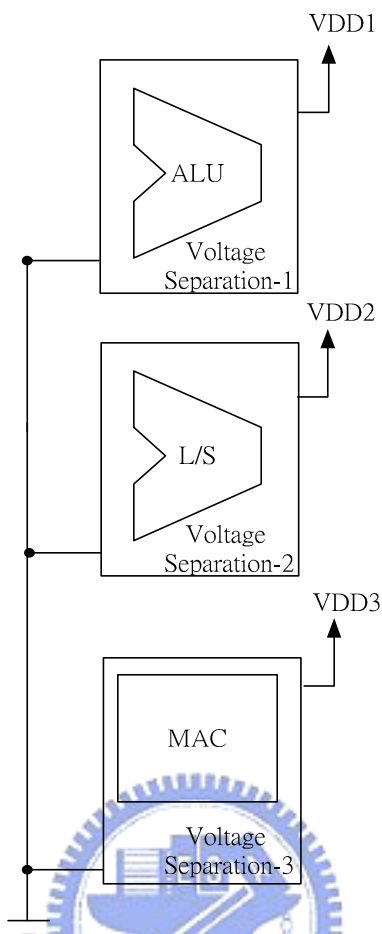


Figure 3.4.1. Voltage Separations

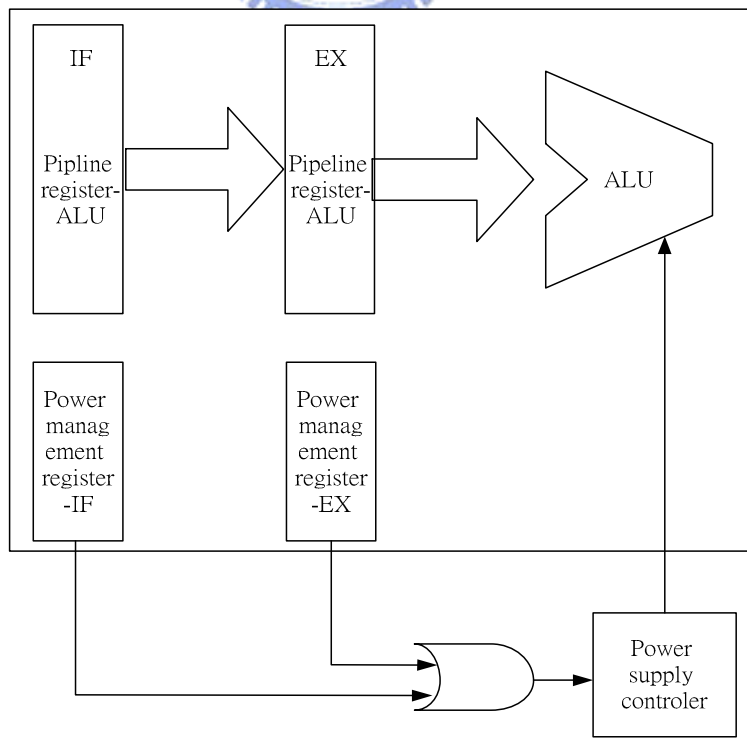


Figure 3.4.2. Implementation of voltage separation on ALU

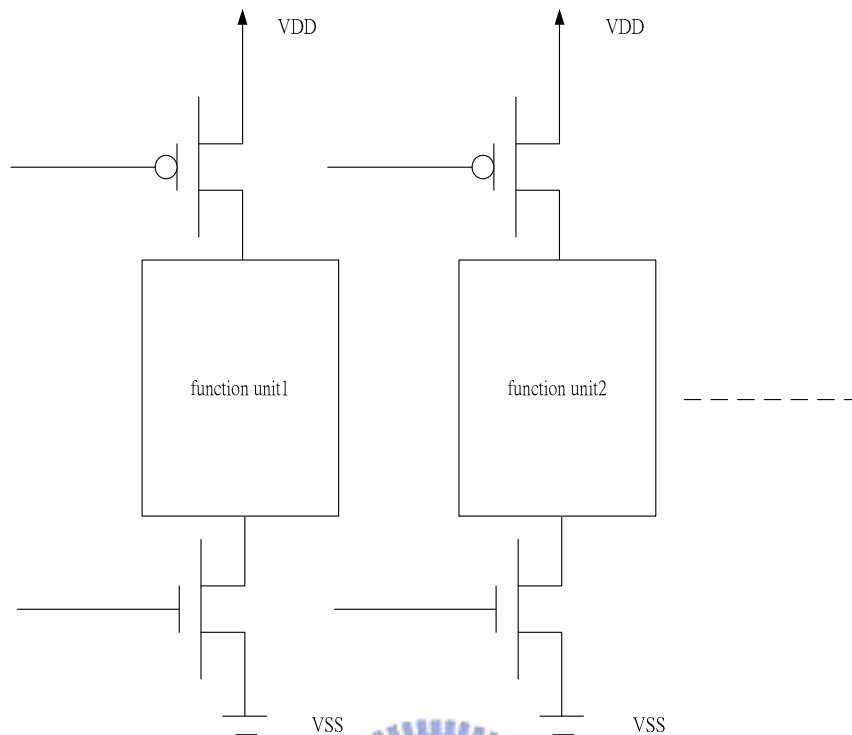


Figure 3.4.3. Implementation of voltage separation with high v_t transistors

A complete voltage separation design is shown in Figure 3.4.3. Two transistors with high V_t value and larger size will be applied to control the power supply of the functional units in the parallel data path. This design should be completed with the process which supports multi V_t transistor. However, the present process provided by CIC [28] does not support this technology and to implement the power supply controller in present process is very difficult in the present environment. Due to the factor above, we implement the power management inside the chip and the power controller is external connected.

When implementing voltage separation, the overheads of circuit charging and discharging are critical factors. One of these overheads is the charging and discharging timing. If the required timing is too long to charge the functional units to stable before data arrives, the operation won't be correct. In recent research [15], the stable required timing to charge such a design as shown in Figure 3.4.3 is within 4ns.

This settling time is reasonable for one or two cycles in a low-power system. The other overhead is the power dissipation caused by circuit charging and discharging. If the power dissipation for charging and discharging is too large, the overhead here could be possibly larger than the static power we save. Due to these two overheads, the control of power supply is very important. We have to reemphasize that although the control signal for power supply is decided by the data in the power management registers, but the data is also a part of instruction and is generated by compiler not by hardware. So the compiler will keep a great flexibility in state arrangement which is used to control the power supply of functional units. But such a compiler should take the impact of charging and discharging into consideration. In the future we will estimate the necessary time to charge and discharge functional units and the power dissipation per charging and discharging by the experiments with this chip. After finishing the experiments, the compiler will be designed according to the results.



Chapter 4 Design Flow and Simulation Result

The parallel data path with clock gating and voltage separation is completed with cell-base design method. However, there should be some extra steps beside the traditional design flow when implementing the circuit with voltage separation and clock gating. In section 4.1 we will show the design flow of this chip implementation and explain why there are some differences from the traditional one. In section 4.2 we will introduce how to test this chip. Finally, the functionality simulation of this chip is shown in section 4.3.

4.1 CAD Flow

Figure 4.1.1 shows the cell-base design flow, it contains the following steps:

1. **Architecture design:** This is the first step to design an integrated circuit (IC). The designer should decide the architecture which includes the detail data path and spec.
2. **RTL:** After the architecture is decided, the designers can begin to write the hardware descript language (HDL) and verify it's functionality.
3. **Synthesis:** If the verification of functionality is correct, the designers can synthesis their design and constrain the timing, delay, and area to their design to meet the required performance. After this step is completed, the simulation with gate delay can be performed.
4. **Physical design:** After the result of gate-level simulation is correct, the physical design can be started. This step includes 2 sub-steps. The first is automatic placement & routing (P&R) which can be completed with EDA tool and create the layout of the design. In general automatic placing and routing (P&R) EDA tool, the basic design flow is formed by the flowing step: specify global net connection, floor planning setup, timing setup, placement and optimization,

synthesis clock tree, connect global nets, routing and optimization, and stream out. This flow is shown in Figure 4.1.2. The second is post-layout simulation which contain the gate delay and wire delay consideration.

5. **Physical verification:** If the result of the simulation is correct, the designer can begin physical verification which includes design rule check (DRC), layout parameter extraction (LPE), and power analysis. These are also the least three steps.

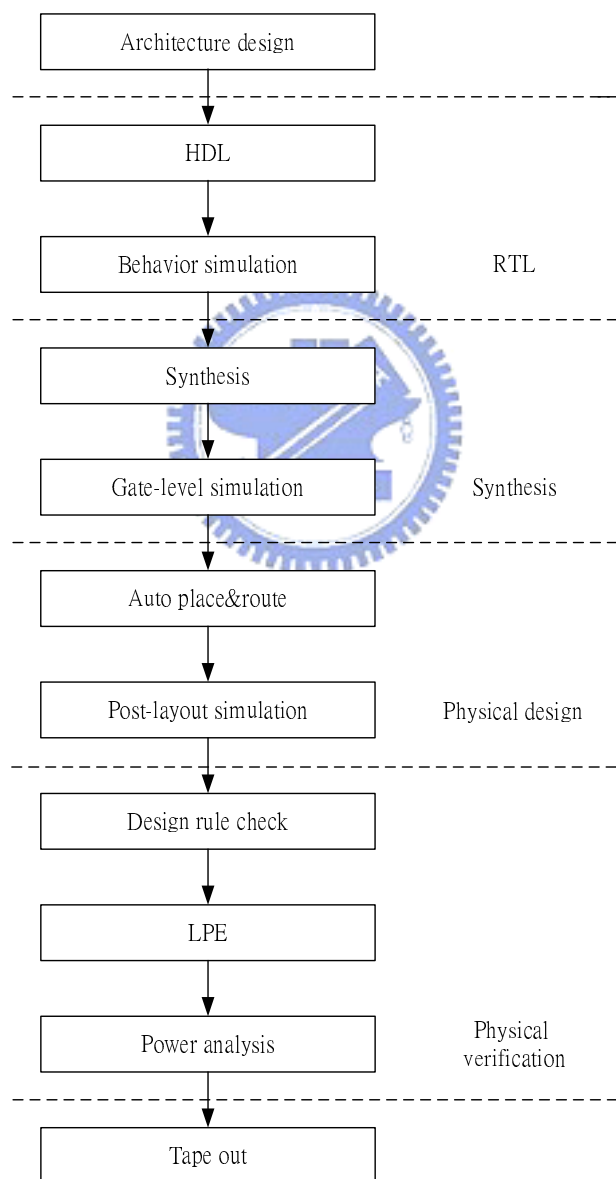


Figure 4.1.1. Cell-based design flow

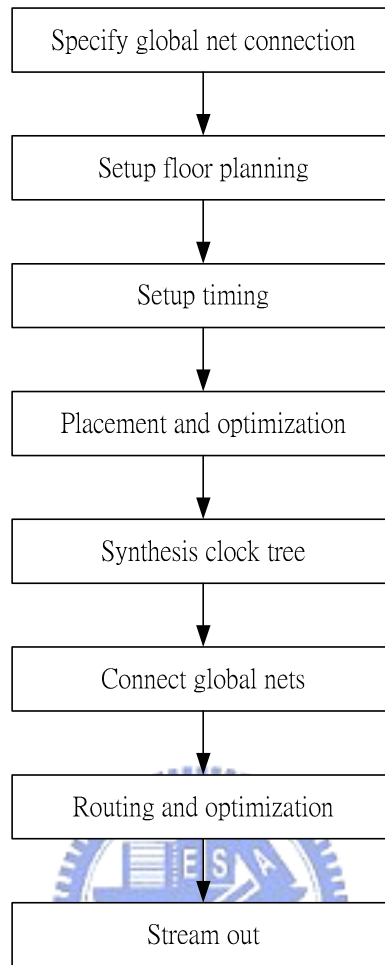


Figure 4.1.2. Basic auto place and route flow

The traditional design flow described above can handle a lot part of digital design. But when implement the circuit with voltage separation and clock gating there should be some extra steps. Figure 4.1.3 shows the design flow of the parallel data path with voltage separation and clock gating implementation. The remaining of this section will present the differences.

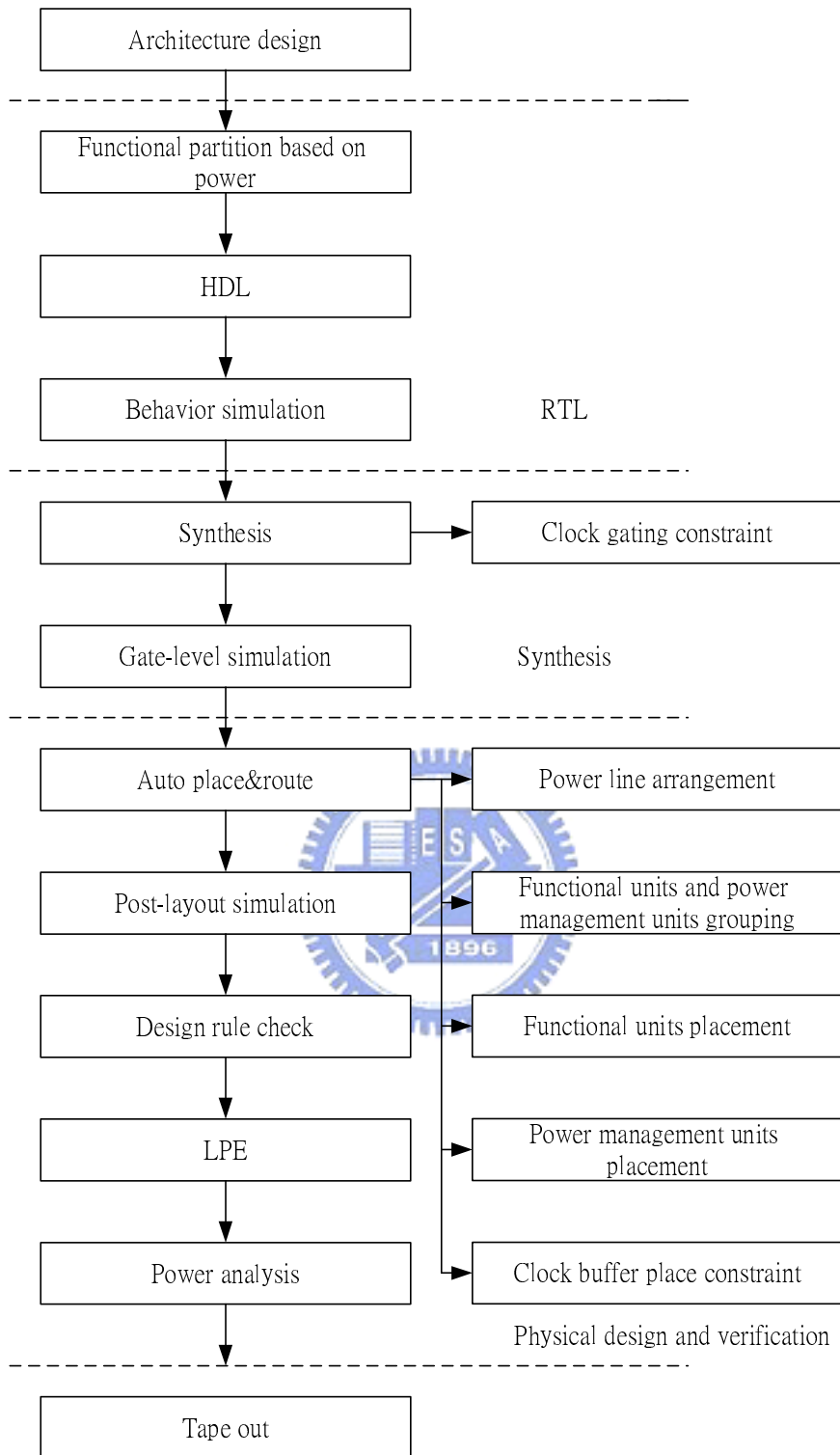


Figure 4.1.3. Design flow of this implementation

- **Functional partition based on power:** When implement voltage separation design, the designer should partition the architecture into different parts based on the power characteristic of each partition before writing the HDL. The partition

consideration includes the active period of each parts and their required performance. In this data path, we implement the voltage separation on the three functional units. So that the data path is partitioned into five parts, the functional unit ALU, the functional unit LS, the functional unit MAC, the memory, and the rest elements. Each part is written into individual module with verilog code, and this will help when separating each part in physical design stage.

- **Set clock gating constraint when synthesis:** The delay of the control signal must be constrained if a design contains gated clock. Figure 4.1.4 gives an example. There are two flip-flops in this example. FF1 is negative edge triggered, and FF2 is positive edge triggered. The output of FF1 will pass through a combinational logic network and form the control signal of the gated clock to FF2. Figure 4.1.5 shows the timing diagram of this example. In the ideal case, there won't be any problem. But if the delay of the combinational logic network is too long, then the gated clock will not operate correctly. Thus the delay of the control signal must be constrained when synthesis

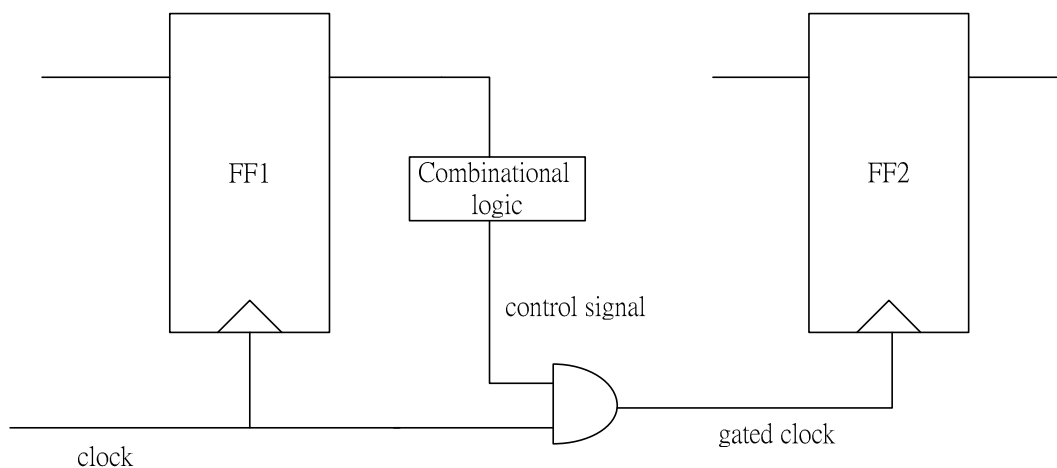


Figure 4.1.4. A clock gating example

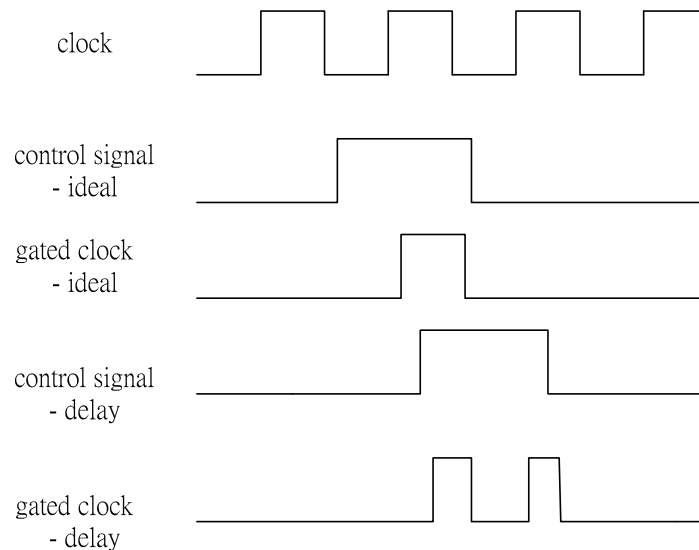


Figure 4.1.5. Timing diagram of Figure 4.1.4

- **Power line arrangement when P&R:** This is a sub-step of floor planning setup. Designer should arrange the power line distribution in his chip. The step will decide the power pad placement and affect the placement of the functional units.
- **Functional units and power management units grouping when P&R:** The standard cells corresponding to different functional unit should be grouped individually, so that the EDA tool can separate functional units and place them as the designer wish. We also group the standard cells of the power management units for clock gating. Because these cell should be placed in the center of the data path for better synthesis of clock tree.
- **Functional units placing when P&R:** Designer should specify the area to place the functional units. Each functional unit should be placed closely to its power supply for reducing the length of power nets, so that this step will be influenced by the power line arrangement.
- **Power management units placing when P&R:** The power management units should not be placed in the area of the three functional units, and it's better to be placed in the center of the data path for better clock tree synthesis.

- **Clock buffer place constraint when P&R:** When EDA tool synthesizes clock tree, designer should note that the clock buffer can't be placed in the three areas of the functional units. Because the clock tree shouldn't be affected by the power switching of the three functional units. So that when clock tree is synthesized, the area of the clock buffer placement must be constrained.

The last five differences are about the automatic P&R, and are shown in Figure 4.1.6.

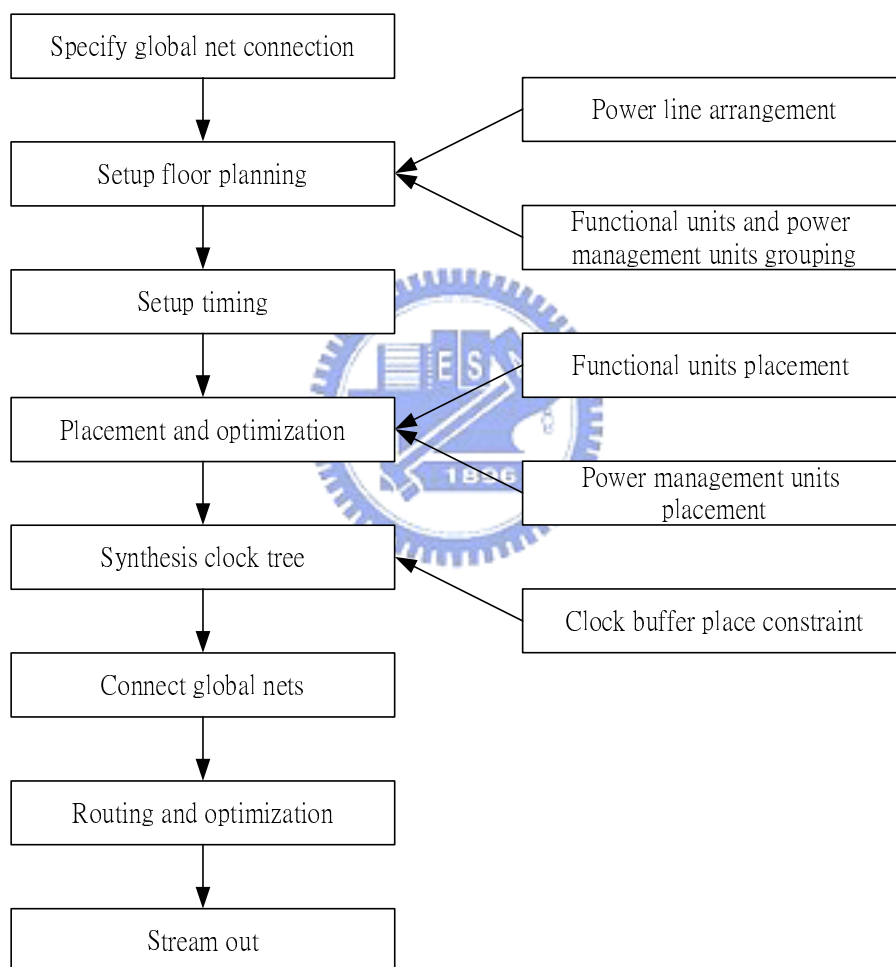


Figure 4.1.6. Design flow for this implementation of auto P&R

The EDA tools used in this implementation are summarized in Table 4.1.1. The whole design is synthesized by Synopsys Design analyzer, and the layout is generated by using Synopsys Apollo. The characteristic of the chip is summarized in Table 4.1.2, and the layout is shown in Figure 4.1.7.

Table 4.1.1. EDA tools used in this implementation

Step	EDA tool	Provider
RTL	verilog	Cadence
Behavior simulation	Debussy	Novas
Synthesis	Design analyzer	Synopsys
Gate level simulation	Debussy	Novas
Auto P&R	Apollo	Synopsys
Post-layout simulation	Debussy	Novas
DRC	Calibre-DRC	Metor Graphic
LVS	Apollo	Synopsys
LPE	Calibre-LPE	Metor Graphic
Power analysis	NANOSIM	Synopsys

Table 4.1.2. Circuit summaries

Technology	UMC 0.18um Mixed Signal (1P5M) CMOS
Library	Artisan SAGE-X Standard Cell Library
Pad Core Size	2.2 mm x 1.8mm
Core Size	1.535 mm x 1.073 mm
On-Chip Memory	1 7x64 single port SRAM 4 16x64 single port SRAM 1 16x256 single port SRAM
Gate Count	45328
Work Clock Rate	40 MHz
Input Pad	40 pins
Output Pad	22 pins
Power Pad	22 pins
Power dissipation	20 mW

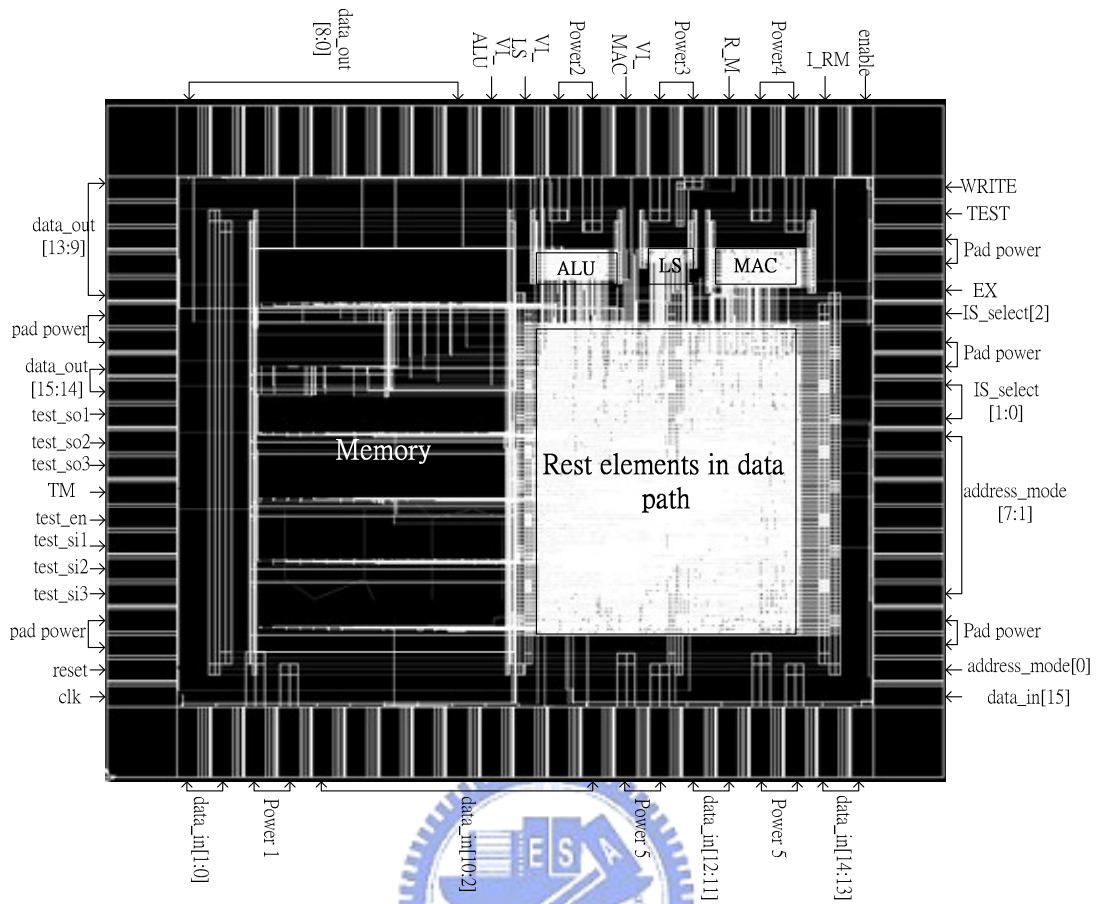


Figure 4.1.7. Layout of the parallel data path with dynamic power management

There are six memories in this chip. The 7x64 and the four 16x64 signal port SRAM are instruction memory, which will save the instructions for execution. The 16x256 signal port SRAM is the data memory, which will save the required data for program execution. All memories in this chip are supplied by power supply 1, and the functional unit ALU is supplied by power supply 2, the L/S is supplied by power supply 3, the MAC is supplied by power supply 4, and the rest elements are supplied by power supply 5.

This chip contains 84 IO pads, in which there are 40 input pads and 22 output pads and 22 power pads. The definitions of the IO pads are summarized in Table 4.1.3.

Table 4.1.3. The definitions of the IO pads

IO pad	IO	Function
clk	Input	The clock signal to this chip
reset	Input	The reset signal to this chip
data_in	Input	This is a 16-bits input, user can insert data into the memory in this chip with these ports.
IS_sellect	Input	This is a 5-bits input. There are five instruction-memory in this chip, this input port will specify which will be written or tested.
address_mode	Input	This is a 8-bits input, user can specify the address of the write or test target memory by this input port.
I_RM	Input	This is a 1-bit input. “0” mean that the writing or testing target is instruction-memory. “1” means that the writing or testing target is data memory or the register file.
R_M	Input	This is a 1-bit input. “0” mean that the write or test target is instruction-memory. “1” means that the write or test target is data memory or the register file.
data_out	Output	This is a 16-bits input. User can get the data in the memory or register file form this port in TEST mode.
VI_ALU	Output	The output of the power management for ALU, this signal will control the power supply to ALU.

VI_LS	Output	The output of the power management for LS, this signal will control the power supply to LS.
VI_MAC	Output	The output of the power management for MAC, this signal will control the power supply to MAC.
TM	Input	Input port for scan chain test
test_en	Input	Input port for scan chain test
test_si	Input	Input port for scan chain test
test_so	output	Output port for scan chain test
Power 1	Power	The power supply to the memory in this chip
Power 2	Power	The power supply to the ALU in this chip
Power 3	Power	The power supply to the LS in this chip
Power 4	Power	The power supply to the MAC in this chip
Power 5	Power	The power supply to the data path in this chip
Pad power	Power	The power supply to the IO pad

4.2 Test Configuration

In this section we will present the test configuration. For the testability of this chip, this data path can be operated in three different modes. They are WRITE mode, EXECUTION mode, and TEST mode. When the chip is going to execute a program, the chip will operate in the order of WRITE, EXECUTION, and TEST. The action of the three modes and the test method will be listed in the following.

- **WRITE mode:** In this mode, the user can insert instructions into the instruction memory and the data into the data memory for execution from the input port data_in. With the combination of the input port IS_sselect, address_mode, I_RM, R_M, user can decide which memory element is the writing target.

- **EXECUTION mode:** After inserting the instruction and the required data into the chip, the data path can begin to execute the program. In this mode, VI_ALU, VI_LS, VI_MAC, will show the state saved in the power management units of voltage separation. User can use logic analyzer to analyze the three outputs, and compare if the outputs are identical to the result of compiler. When the data path is in execution mode, user can also use power meter to measure the power dissipation of the five power supply for this chip.
- **TEST mode:** After the program execution is finished, user can test the data in the data memory and the register file in this chip. With the combination of the input port address_mode, I_RM, and R_M, user can read out the data from the output port data_out and use a logic analyzer to exam the output if the execution result is correct.



4.3 Simulation Result

In this section, we will show the functionality simulation of this chip. These simulations include the clock gating operation and a program execution of two matrixes addition. In section 4.3.1, the clock gating operation in five stage pipeline is shown. In section 4.3.2, a test bench of two matrix addition is executed and the three operation mode simulation of this chip is shown. All simulations are post-layout simulation and the operation frequency is 40Mhz.

4.3.1 Clock Gating Simulation

In this section, we will simulate the functionality of clock gating and the control signal for voltage separation in the data path to make sure that after implementing these two dynamic power management, the instruction execution and the pipeline operation are correct. Firstly, the following instructions are executed.

1. SUBI R5 R4 #1
2. ADDI R1 R0 #1
3. LW R28 R4 #0
4. STALL
5. LW R29 R5 #0
6. STALL
7. STALL

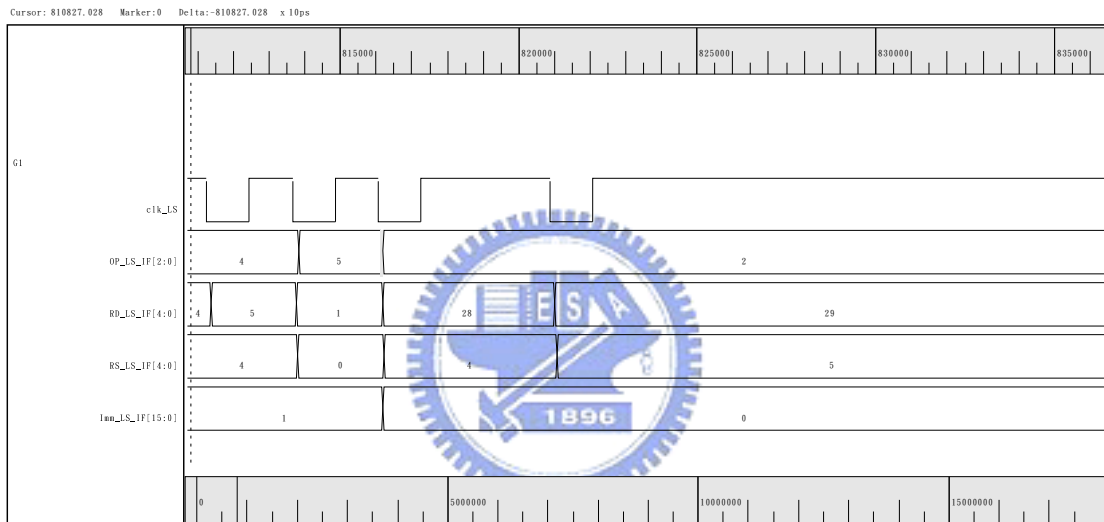


Figure 4.3.1. Simulation of instruction fetch

Figure 4.3.1 shows the post-layout simulation result of the instruction fetch and present the operation of clock gating in this stage. OP_LS_IF, RD_LS_IF, RS_LS_IF and imm_LS_IF are the pipeline registers for functional unit L/S in instruction fetch stage. They are used to save the instruction fetched from instruction memory. The more detail definition of the signals in Figure 4.3.1 is summarized in Table 4.3.1.

Table 4.3.1. Definitions of the signals in Figure 4.3.1

clk_LS	The clock signal of the pipeline registers for functional unit L/S in IF stage
OP_LS_IF	The pipeline registers used to save the OP code of instruction. Number 4 for instruction SUBI. Number 5 for ADDI and number 2 for LW
RD_LS_IF	The pipeline registers used to save the destination register address
RS_LS_IF	The pipeline registers used to save the source register address
imm_LS_IF	The integer used to calculate data address in memory

Only when the instruction fetched is not a stall instruction, the clock signal switches, or it remain in high state. The simulation result is identical to that shown in Figure 3.3.2. The clock gating operate correctly, and the pipeline register only reads the instructions that really be executed.

The clock gating operates correctly in single pipeline stage. In the next, the simulation of pipeline operation with clock gating and the state detection for voltage separation will be presented in Figure 4.3.2. An example of the functional unit L/S is shown. The data flow in all the pipeline registers will also be shown. The signals in G1 represent the pipeline registers in IF stage. G2 represent the pipeline registers in ID stage. G3 represent the pipeline registers in MEM stage. G4 represent the pipeline registers in WB stage, and the detail definitions of the signals in Figure 4.3.2 is summarized in table 4.3.2.

Table 4.3.2 Definitions of the signals in Figure 4.3.2

G1	clk_LS	The gated clock signal for pipeline registers in IF stage
	RD_LS_IF	The pipeline registers used to save the destination register (RD) address
	RS_LS_IF	The pipeline registers used to save the source register (RS) address
	imm_LS_IF	The integer used to calculate data address in memory
	OP_LS_IF	The pipeline registers used to save OP code of the instruction fetched
G2	clk_LS	The gated clock signal for pipeline registers in EX stage
	WB_LS_EX	A part of the instruction decoded result which controls the access to register file at WB stage
	MEM_LS_EX	A part of the instruction decoded result which controls the access to MEMORY at MEM stage.
	EX_LS_EX	A part of the instruction decoded result which decides the operation at EX stage.
	RDdata_LS_EX	The pipeline registers which save the data at destination address. If the instruction is STORE, the data here will be written to MEMORY.
	RSdata_LS_EX	The pipeline registers which save the data at source address and is used to calculate the data address in memory
	imm_LS_EX	The integer used to calculate data address in memory,

		and the data in this part is directly form imm_LS_IF
G3	clk_LS	The gated clock signal for pipeline registers in MEM stage
	WB_LS_MEM	The data in this part is directly form WB_LS_EX
	MEM_LS_MEM	That control the access to memory and the data in this part is directly from MEM_LS_EX
	RDdata_LS_MEM	The data in this part is directly form RDdata_LS_EX. If the instruction is STORE, the data here will be written to MEMORY
	result_LS_MEM	The result of the functional execution and the data in this register maybe the data address in memory or the data which will be written back to RD
	RD_LS_MEM	The data in this part is directly from RD_LS_EX.
G4	clk_LS	The gated clock signal for pipeline registers in WB stage
	WB_LS_WB	The data in this part is directly form WB_LS_MEM and controls the access to register file
	result_LS_WB	The data in this part is directly from result_LS_MEM. If the instruction is ADDI or SUBI, the data will be written back to RD
	MEMdata_LS_WB	The data in this part is from memory. If the instruction is LOAD, the data will be written back to RD.
	RD_LS_WB	The data in this part is directly from RD_LS_MEM and represents the address of RD

G5	voltage_separation_LS	That represents the condition of the functional unit. “1” means that the functional unit is busy now. “0” means that the functional unit is idle and the power supply can be turned off
----	-----------------------	--

In Figure 4.3.2, following instructions are executed :

1. STALL
2. ADDI R4 R0 #15
3. STALL
4. STALL
5. STALL
6. STALL
7. STALL
8. STALL
9. SUBI R3 R1 #6
10. STALL



Only the instruction fetched is not a stall instruction, the gated clock to each pipeline register will switch. In this case, all the instructions between instruction ADDI R4 R0 #15 and instruction SUBI R3 R1 #6 are all stall. In this period, the functional unit is under idle situation. So after instruction ADDI R4 R0 #15 is executed, the signal voltage_separation_LS is set to low. This means that the power supply to this functional unit could be turned off. Before instruction SUBI R3 R1 #6 is fetched, the signal voltage_separation_LS is set to high. This means that the functional unit need to be charge immediately.

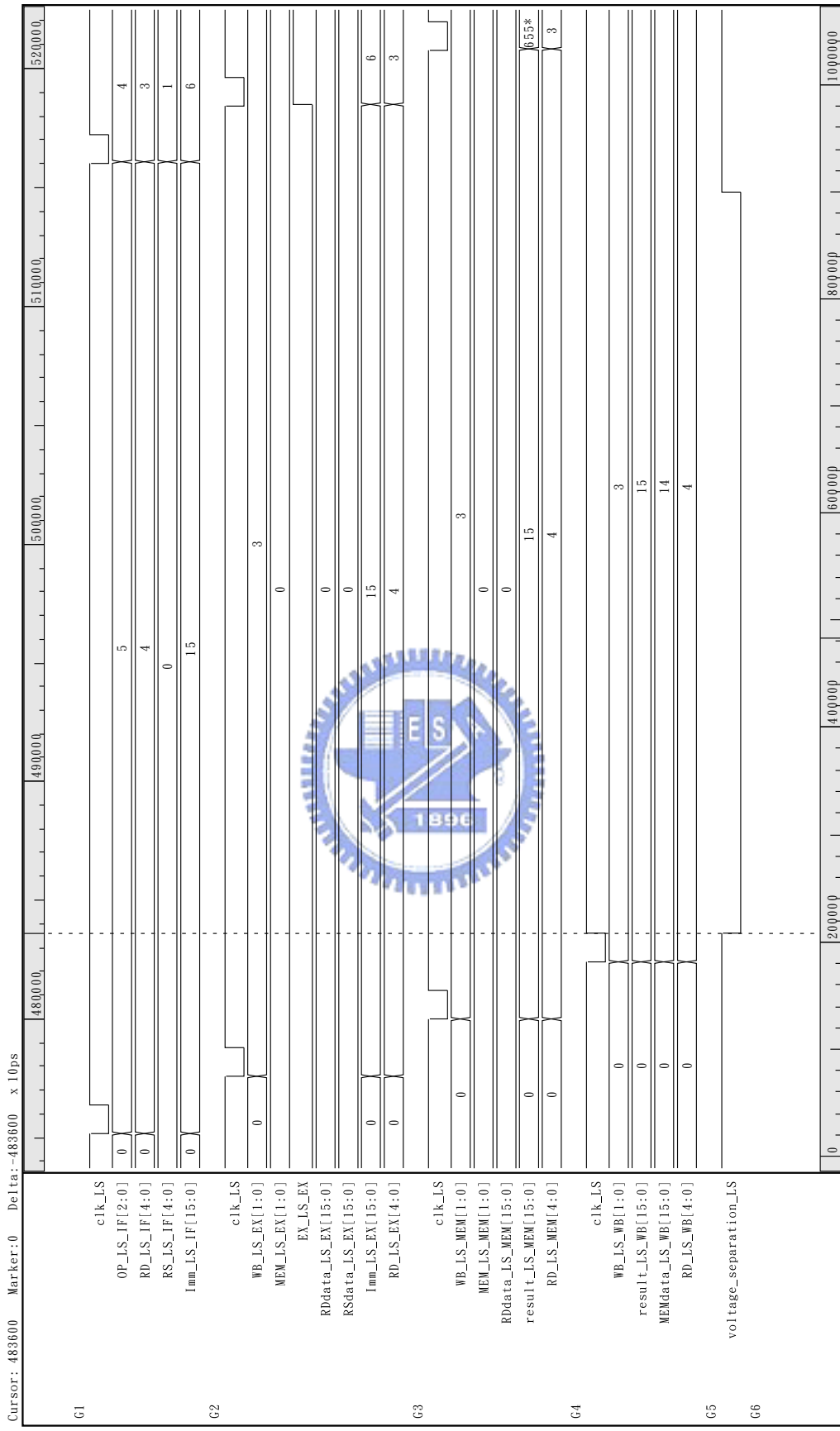


Figure 4.3.2. The simulation of pipeline operation with power management units

4.3.2 Test Bench Simulation

In the following we will show a simulation of a program execution. The program is the addition of two matrixes : $A = A + B$, in which A and B are 5x5 matrixes as shown in Figure 4.3.3. The elements in matrix A and B are individually saved in data memory from address 0 to 24 and address 25 to 49. The assemble language of this program is shown in table 4.3.3, and the program will use the functional unit ALU and L/S.

$$A = \begin{bmatrix} 0 & 5 & 10 & 15 & 20 \\ 1 & 6 & 11 & 16 & 21 \\ 2 & 7 & 12 & 17 & 22 \\ 3 & 8 & 13 & 18 & 23 \\ 4 & 9 & 14 & 19 & 24 \end{bmatrix} \quad B = \begin{bmatrix} 25 & 30 & 35 & 40 & 45 \\ 26 & 31 & 36 & 41 & 46 \\ 27 & 32 & 37 & 42 & 47 \\ 28 & 33 & 38 & 43 & 48 \\ 29 & 34 & 39 & 44 & 49 \end{bmatrix}$$

Figure 4.3.3. Matrix A and matrix B

Table 4.3.3. The assemble language of the program $A=A+B$

	ALU	LS	MAC
0		ADDI R31 R0 #24	
1			
2			
3		LW R21 R31 #0	
4		LW R2 R31 #25	
5		LW R3 R31 #1	
6		LW R4 R31 #26	
7		LW R5 R31 #2	
8		LW R6 R31 #27	
9		LW R7 R31 #3	
10		LW R8 R31 #28	
11	ADD R21 R21 R2	LW R9 R31 #4	
12	ADD R3 R3 R4	LW R10 R31 #29	
13	ADD R5 R5 R6		
14	ADD R7 R7 R8	SW R21 R31 #0	
15	ADD R9 R9 R10	SW R3 R31 #1	
16		SW R5 R31 #2	

17		SW R7 R31 #3	
18		SW R9 R31 #4	
19		SUBI R31 R31 #1	
20	BEQ R31 # -19		
21			
22	HALT		
23	HALT		

As mentioned in Section 4.2, the first step is to insert the instructions shown in Table 4.3.3 into the instruction memory, and the elements of A and B into the data memory inside the chip. The next is to execute the instruction which just inserted. After the execution is finished, the test mode starts. The result of the program will be read out from the output port data_out. This operation flow is shown in Figure 4.3.4. When the signal WRITE is accessed, the chip is under the WRITE mode and the signal data_in is the instruction and matrix elements which are inserted. When the signal EX is accessed, the chip is under EXECUTION mode. And when the signal TEST is high, the chip is under TEST mode.

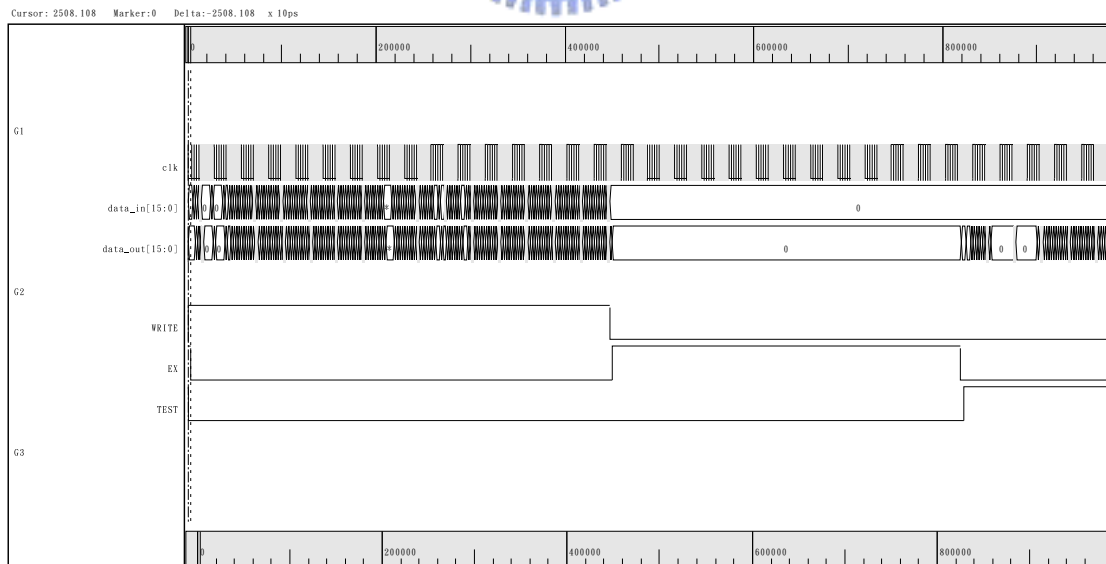


Figure 4.3.4 The operation flow of this chip

When this chip is under WRITE mode, the instructions and the required data are inserted. Figure 4.3.5 shows the operation of WRITE mode. When the signal I_RM is

accessed, the instruction is inserted and the signal IS_select means which instruction memory will save the instruction form the input port data_in. The signal address_mode means the address of the specified instruction memory that will save the input instruction. When the signal I_RM is low, the input data will be saved into the data memory. In this period, the elements of A and B are inserted and the insertion of matrix A and matrix B are shown in Figure 4.3.6 and Figure 4.3.7 respectively. The elements of matrix A will be saved in the data memory from address 0 to address 24, and the elements of matrix B will be saved in the data memory form address 25 to address 49.

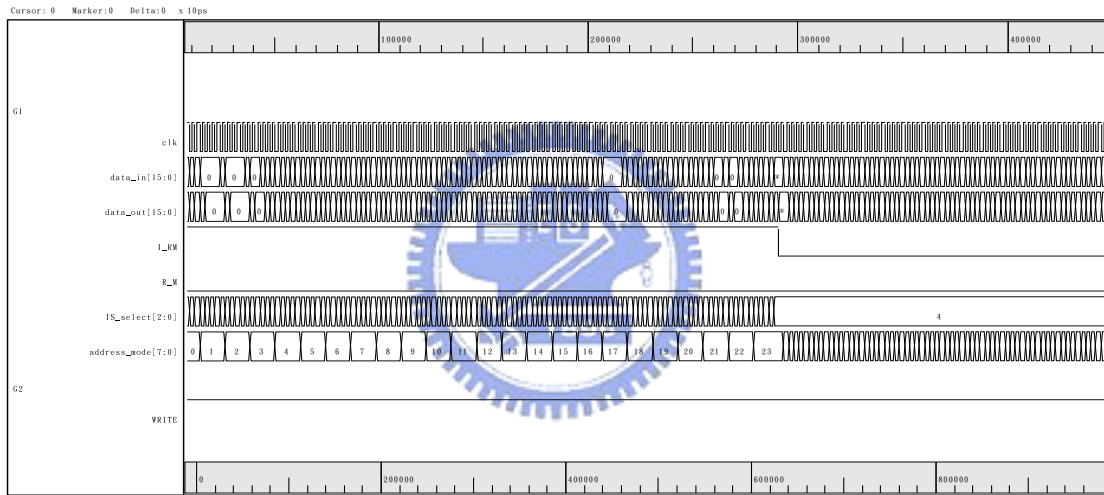


Figure 4.3.5 The operation of WRITE mode

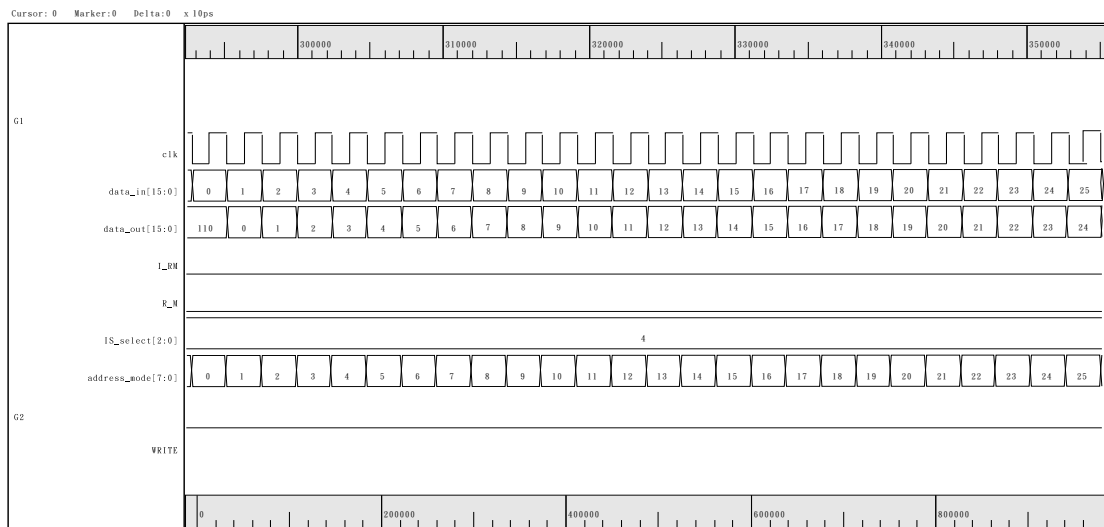


Figure 4.3.6 Insertion of matrix A

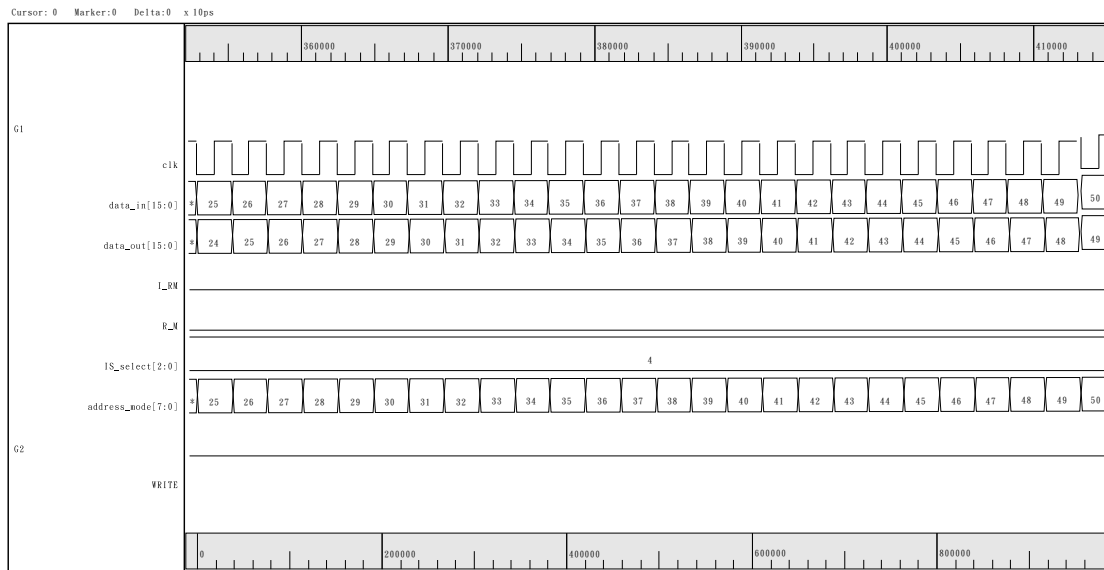


Figure 4.3.7 Insertion of matrix B

After WRITE mode, the EXECUTION mode begins. Figure 4.3.8 shows the voltage_separation_ALU, voltage_separation_LS, and voltage_separation_MAC signals in EXECUTION mode. The three signals are generated from the power management unit for voltage separation in this chip. The program only use ALU and LS, so the voltage_separation_ALU and voltage_separation_LS will be set to high, and voltage_separation_MAC is set to low. This means that the power control outside the chip should supply voltage to the functional units ALU and L/S, and the power supply of MAC can be turned off.

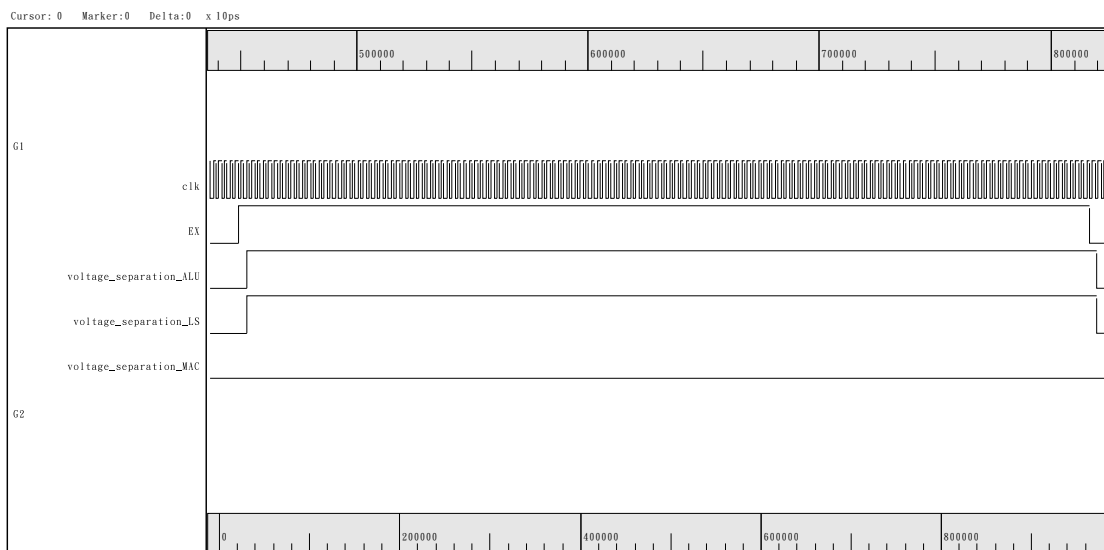


Figure 4.3.8. EXECUTION mode

The last step is to test the result of program execution. Figure 4.3.9 shows matrix A after execution, and these elements will be saved in the data memory from address 0 to address 24. User can read the elements out in TEST mode, and this operation is shown in Figure 4.3.10. The signal address_mode means that the address of the data in the data memory will be read out by the output port data_out. As shown in Figure 4.3.10, the execution result is correct.

$$A = \begin{bmatrix} 25 & 35 & 45 & 55 & 65 \\ 27 & 37 & 47 & 57 & 67 \\ 29 & 39 & 49 & 59 & 69 \\ 31 & 41 & 51 & 61 & 71 \\ 33 & 43 & 53 & 63 & 73 \end{bmatrix}$$

Figure 4.3.9. Matrix A after execution

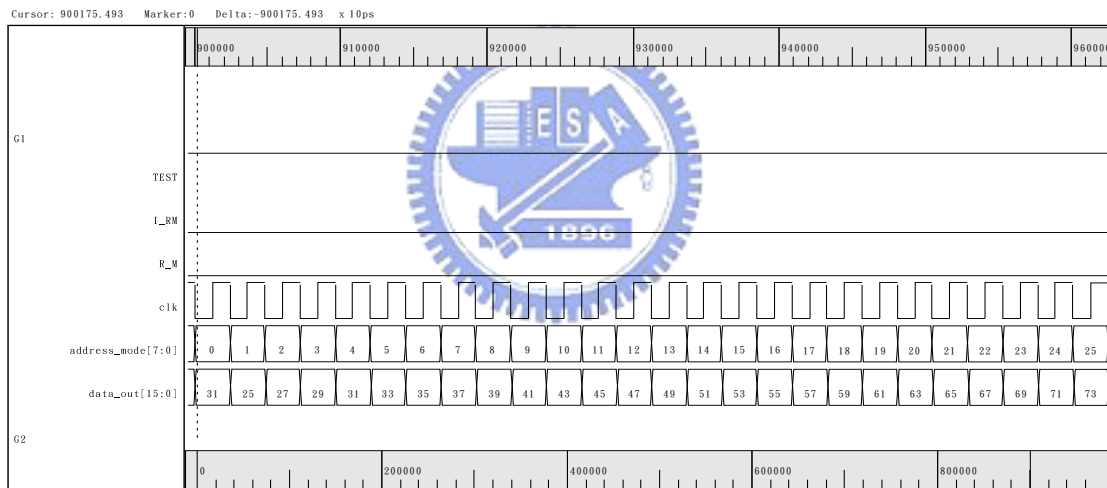


Figure 4.3.10. TEST mode

Chapter 5 Power Analysis

This chapter presents the power analysis of the parallel data path with dynamic power management according to the verification flow which is mentioned in Chapter 4. There are five benchmarks simulated. The first is matrix addition. We will compare the difference of power dissipation with and without power management. The second is idle process. In this case, all the clock for pipeline register are disabled and all the functional units can be turned off. This is also the best case for power management simulation. The third, fourth, and the last benchmark will all execute the same program. In the third, all the functional units are used. In the fourth, only MAC and L/S are used. In the last, only ALU and L/S are used. In the last three benchmarks, the power dissipation, performance, and energy consumption of the three cases are shown, compared, and discussed. In Section 5.1, the benchmarks are defined. In Section 5.2, the results and comparison of simulation are presented. Finally, a summary is presented in Section 5.3. All the simulation is completed with Synopsys Nanosim and the operation frequency is 40MHz.

5.1 Benchmark Definition

In this Section, each benchmark will be defined and the assemble language of each benchmark will also be shown.

5.1.1 Matrix Addition

The first benchmark is addition of two matrixes : $A = A + B$, in which A and B are 10x10 matrixes. The assemble language of this program is shown in Table 5.1.1. The stall instructions occupy about 56%. This benchmark use two functional units that are ALU and L/S, and the execution time is 7812 ns.

Table 5.1.1 The assemble language of Matrix Addition

	ALU	LOAD/STORE	MAC
0		ADDI R31 R0 #180	
1			
2			
3		LW R21 R31 #0	
4		LW R2 R31 #1	
5		LW R3 R31 #2	
6		LW R4 R31 #3	
7		LW R5 R31 #4	
8		LW R6 R31 #5	
9		LW R7 R31 #6	
10		LW R8 R31 #7	
11		LW R9 R31 #8	
12		LW R10 R31 #9	
13		LW R11 R31 #10	
14		LW R12 R31 #11	
15		LW R13 R31 #12	
16	ADD R21 R21 R2	LW R14 R31 #13	
17	ADD R3 R1 R4	LW R15 R31 #14	
18	ADD R5 R1 R6	LW R16 R31 #15	
19	ADD R7 R1 R8	LW R17 R31 #16	
20	ADD R9 R1 R10	LW R18 R31 #17	
21	ADD R11 R1 R12	LW R19 R31 #18	
22	ADD R13 R1 R14	LW R20 R31 #19	
23	ADD R15 R1 R16	SW R21 R31 #0	
24	ADD R17 R1 R18	SW R3 R31 #1	
25	ADD R19 R1 R20	SW R5 R31 #2	
26		SW R7 R31 #3	
27		SW R9 R31 #4	
28		SW R11 R31 #5	
29		SW R13 R31 #6	
30		SW R15 R31 #7	
31		SW R17 R31 #8	
32		SW R19 R31 #9	
33		SUBI R31 R31 #20	
34	BEQ R31 #1		

35			
36	HALT		
37	HALT		

5.1.2 Idle Process

The case of Idle Process is to execute nothing, and the whole data path is under stand by situation. This is the best case for power management simulation. Because the clock of the pipeline registers can all be turned off, so as the power of the three functional units. We will execute the case for 5000 ns, and presented the difference of with and without power management.

5.1.3 Matrix Calculation with All Functional Units

The benchmark is matrix calculation of $C = 4A + 3B$, $D = 4A - 3B$. In which A, B, C, D are all 5 X 5 matrixes. The assemble language of this benchmark is shown in Table 5.1.2. All the function units are used, and this is the best instruction scheduling of the program in this parallel data path. The execution time of this case is 2912ns and the stall instructions occupy about 35%.

Table 5.1.2. The assemble language of matrix calculation with all functional units

	ALU	LOAD/STORE	MAC
0		ADDI R31 R0 #40	
1		ADDI R30 R0 #4	
2		ADDI R29 R0 #3	
3		LW R21 R31 #0	
4		LW R2 R31 #1	
5		LW R3 R31 #2	
6		LW R4 R31 #3	MUL R22 R21 R30
7		LW R5 R31 #4	MUL R23 R2 R29
8		LW R6 R31 #5	MUL R24 R3 R30
9		LW R7 R31 #6	MUL R25 R4 R29
10	ADD R11 R22 R23	LW R8 R31 #7	MUL R26 R5 R30
11	SUB R12 R22 R23	LW R9 R31 #8	MUL R27 R6 R29

12	ADD R13 R24 R25	LW R10 R31 #9	MUL R28 R7 R30
13	SUB R14 R24 R25	SW R11 R31 #50	MUL R2 R8 R29
14	ADD R15 R26 R27	SW R12 R31 #51	MUL R3 R9 R30
15	SUB R16 R26 R27	SW R13 R31 #52	MUL R4 R10 R29
16	ADD R17 R28 R2	SW R14 R31 #53	
17	SUB R18 R28 R2	SW R15 R31 #54	
18	ADD R19 R3 R4	SW R16 R31 #55	
19	SUB R20 R3 R4	SW R17 R31 #56	
20		SW R18 R31 #57	
21		SW R19 R31 #58	
22		SW R20 R31 #59	
23		SUBI R31 R31 #10	
24	BEQ R31 #1		
25			
26	HALT		
27	HALT		

5.1.4 Matrix Calculation with Functional Unit MAC

The program is the same with that in Section 5.1.3. But only functional units L/S and MAC are used. The assemble language of this benchmark is shown in table 5.1.3. The execution time of this case is 3060 ns and the stall instructions occupy about 46%.

Table 5.1.3 The assemble language of matrix calculation with MAC

	ALU	LOAD/STORE	MAC
0		ADDI R31 R0 #40	
1		ADDI R30 R0 #4	
2		ADDI R29 R0 #3	
3		ADDI R28 R0 # -3	
4		LW R21 R31 #0	
5		LW R3 R31 #1	
6		LW R5 R31 #2	
7		LW R7 R31 #3	MUL R21 R21 R30
8		LW R9 R31 #4	MUL R3 R3 R30
9		LW R2 R31 #5	MUL R5 R5 R30

10		LW R4 R31 #6	MUL R7 R7 R30
11		LW R6 R31 #7	MUL R9 R9 R30
12		LW R8 R31 #8	MAC R21 R2 R29
13		LW R10 R31 #9	MAC R21 R2 R28
14			MAC R3 R4 R29
15		SW R21 R31 #50	MAC R3 R4 R28
16		SW R21 R31 #51	MAC R5 R6 R29
17		SW R3 R31 #52	MAC R5 R6 R28
18		SW R3 R31 #53	MAC R7 R8 R29
19		SW R5 R31 #54	MAC R7 R8 R28
20		SW R5 R31 #55	MAC R9 R10 R29
21		SW R7 R31 #56	MAC R9 R10 R28
22		SW R7 R31 #57	
23		SW R9 R31 #58	
24		SW R9 R31 #59	
25		SUBI R31 R31 #10	
26	BEQ R31 #1		
27			
28	HALT		
29	HALT		



5.1.5 Matrix Calculation with Functional Unit ALU

The program is the same with that in Section 5.1.3. But only functional units L/S and ALU are used. The assemble language of this benchmark is shown in Table 5.1.4. The execution time of this case is 4812ns and the stall instructions occupy about 54%.

Table 5.1.4. The assemble language of calculation with ALU

	ALU	LOAD/STORE	MAC
0		ADDI R31 R0 #40	
1			
2			
3		LW R21 R31 #0	
4		LW R2 R31 #1	
5		LW R3 R31 #2	
6	ADD R21 R21 R21	LW R4 R31 #3	
7	ADD R22 R2 R2	LW R5 R31 #4	

8	ADD R3 R3 R3	LW R6 R31 #5	
9	ADD R24 R4 R4	LW R7 R31 #6	
10	ADD R5 R5 R5	LW R8 R31 #7	
11	ADD R26 R6 R6	LW R9 R31 #8	
12	ADD R7 R7 R7	LW R10 R31 #9	
13	ADD R28 R8 R8		
14	ADD R9 R9 R9		
15	ADD R30 R10 R10		
16	ADD R21 R21 R21		
17	ADD R3 R3 R3		
18	ADD R5 R5 R5		
19	ADD R7 R7 R7		
20	ADD R9 R9 R9		
21	ADD R22 R22 R2		
22	ADD R24 R24 R4		
23	ADD R26 R26 R6		
24	ADD R28 R28 R8		
25	ADD R30 R30 R10		
26	ADD R11 R21 R22		
27	SUB R12 R21 R22		
28	ADD R13 R3 R24		
29	SUB R14 R3 R24	SW R11 R31 #50	
30	ADD R15 R5 R26	SW R12 R31 #51	
31	SUB R16 R5 R26	SW R13 R31 #52	
32	ADD R17 R7 R28	SW R14 R31 #53	
33	SUB R18 R7 R28	SW R15 R31 #54	
34	ADD R19 R9 R30	SW R16 R31 #55	
35	SUB R20 R9 R30	SW R17 R31 #56	
36		SW R18 R31 #57	
37		SW R19 R31 #58	
38		SW R20 R31 #59	
39		SUBI R31 R31 #10	
40	BEQ R31 #1		
41			
42	HALT		
43	HALT		

5.2 Simulation Result

In this section, the simulation results are shown. In each benchmark, we will show the difference in power dissipation of with and without power management. Furthermore, the comparison and discussion for matrix calculation with different functional units are also presented.

5.2.1 Matrix Addition

The execution of matrix addition takes 7812 ns, and the power dissipation of the data path and the three functional units are listed in Table 5.2.1 and Table 5.2.2 respectively. The numerical values in Table 5.2.1 are the simulation result which the power management is turned off, and these in Table 5.2.2 are the simulation result with clock gating.

Table 5.2.1. Power distribution of matrix addition (without power management) :

Part	Power (mW)	Percentage
Rest elements in data path	4.49517	98.2 %
ALU	0.04676	1 %
L/S	0.03334	0.7%
MAC	0.00011	< 0.001%
Total	4.57528	

Table 5.2.2. Power distribution of matrix addition (with clock gating) :

Part	Power (mW)	Percentage
Rest elements in data path	3.53560	97.8%
ALU	0.04552	1.2 %
L/S	0.03092	0.9%
MAC	0.00011	< 0.001%
Total	3.61240	

Figure 5.2.1 shows the total power dissipation comparison of with and without power management. This program doesn't use the functional unit MAC, and the static power dissipation of MAC can be saved by the power management of voltage

separation. But the scale of the static power of MAC is too small in comparison with the total power, so that the difference between with and without voltage separation is not very obviously, and the power saved by clock gating and voltage separation is about 21 %.

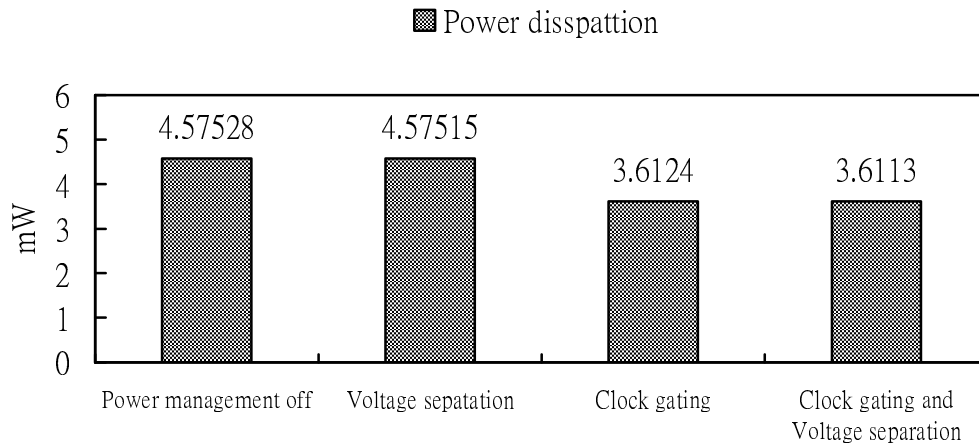


Figure 5.2.1. Total power dissipation of matrix addition

Figure 5.2.2 shows the difference in power dissipation of the sum of the three functional units with and without power management. The result of the simulation are listed in Table 5.2.3.

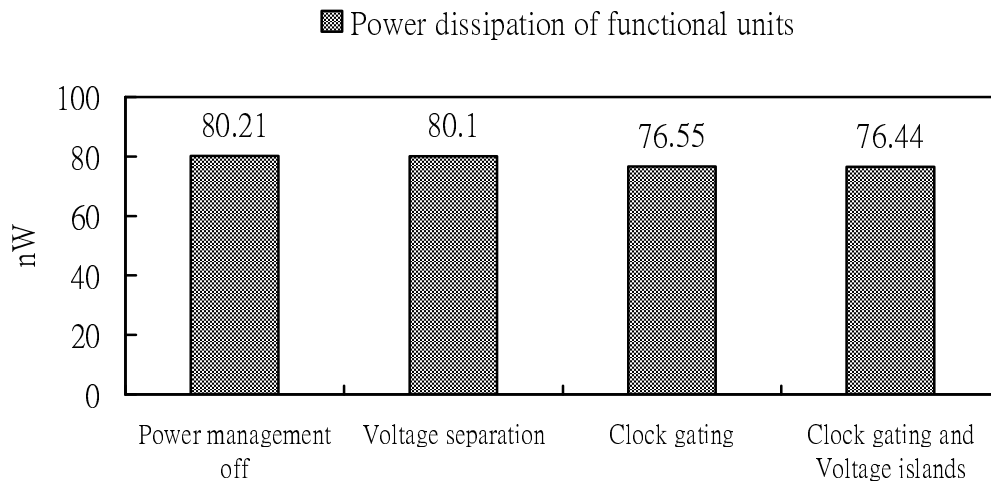


Figure 5.2.2. The power dissipation of the three functional units

Table 5.2.3. The power analysis result of matrix addition

Program	A = A + B, A and B are 10x10 matrix.
Functional units used	ALU, L/S
Execution time	7812 ns
Percentage of stall instruction	56 %
Total power dissipation without P.M	4.57528 mW
Total power dissipation with P.M	3.61229 mW
Power saved	21 %

5.2.2 Idle Process

The simulation time of idle process takes 5000 ns. The data path is standing by, and no functional units is used. The power dissipation is listed in Table 5.2.4 and Table 5.2.5. Table 5.2.4 shows the power dissipation without power management, and Table 5.2.5 shows the case with clock gating. The power consumed by data path almost occupy all the power.

Table 5.2.4. Power distribution of idle process (without power management) :

Part	Power (mW)	Percentage
Rest elements in data path	3.76610	99.9%
ALU	0.00001	<0.001%
L/S	0.00001	<0.001%
MAC	0.00011	<0.001%
Total	3.76623	

Table 5.2.5. Power distribution of idle process (with clock gating) :

Part	Power (mW)	Percentage
Rest elements in data path	1.99429	99.9%
ALU	0.00001	<0.001%
L/S	0.00001	<0.001%
MAC	0.00011	<0.001%
Total	1.99442	

Figure 5.2.3 shows the power dissipation of idle process. It's the same with matrix addition that the scale of the static power dissipation of the three function units is too small in comparison with the one of the rest elements in data path. So that the difference between with and without voltage separation is not very obviously. Figure 5.3.4 shows the power dissipation of the three functional units. The power management of voltage separation can save all the static power of the functional units, and the power saved by power management of clock gating and voltage separation is about 47% in the case of idle process. The simulation result of idle process is listed in Table 5.2.6.

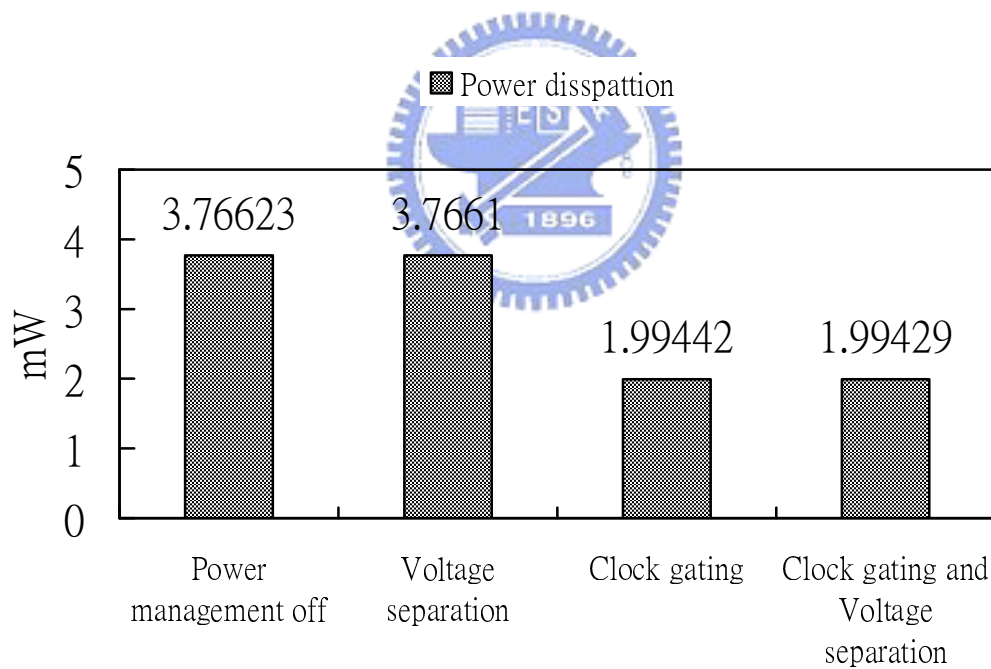


Figure 5.2.3. Total power dissipation of idle process

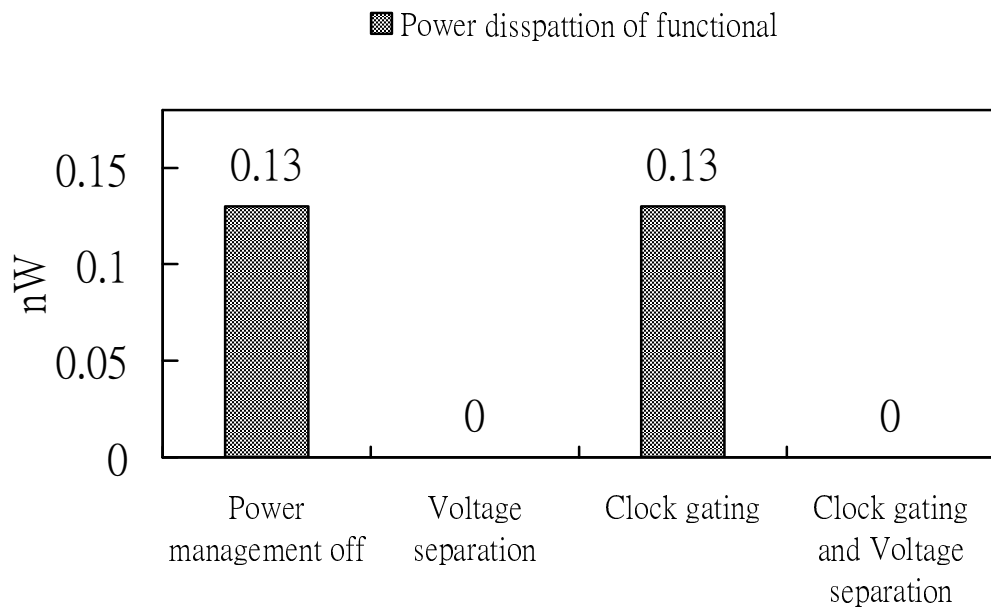


Figure 5.2.4. Total power dissipation of the three functional units

Table 5.2.6. The power analysis result of idle process

Program	Stand by
Functional units used	Non of all
Execution time	5000 ns
Percentage of stall instruction	100 %
Total power dissipation without P.M	3.76623 mW
Total power dissipation with P.M	1.99429 mW
Power saved	47 %

5.2.3 Matrix Calculation with All Functional Units

The execution of matrix calculation with all functional units takes 2910 ns, and the power dissipation of the three functional units and the rest elements in data path are listed in Table 5.2.7 and Table 5.2.8. The numerical values in Table 5.2.7 are the simulation result which the power management is turned off, and these in Table 5.2.8 are the simulation result with clock gating.

Table 5.2.7. Power distribution of matrix calculation (without P.M.) :

Part	Power (mW)	Percentage
Rest elements in data path	4.94905	94.8 %
ALU	0.14443	2.7 %
L/S	0.04484	0.8%
MAC	0.07850	1.5%
Total	5.21681	

Table 5.2.8. Power distribution of matrix calculation (with clock gating) :

Part	Power (mW)	Percentage
Rest elements in data path	4.27484	94.2%
ALU	0.14218	3.1 %
L/S	0.04302	0.9%
MAC	0.07500	1.7%
Total	4.53503	

Figure 5.2.5 shows the total power dissipation comparison of with and without power management for matrix calculation with all functional units. All the functional units are used when program execution. We assume that the idle period of ALU and MAC are not long enough for power control to switch power supply and the power supply of the three functional never be turned off. So that the power dissipation with and without voltage separation are the same. The power dissipation of the three functional units is shown in Figure 5.2.6, and the power saved by clock gating is about 13.1 %. The simulation result of this case is summarized in Table 5.2.9.

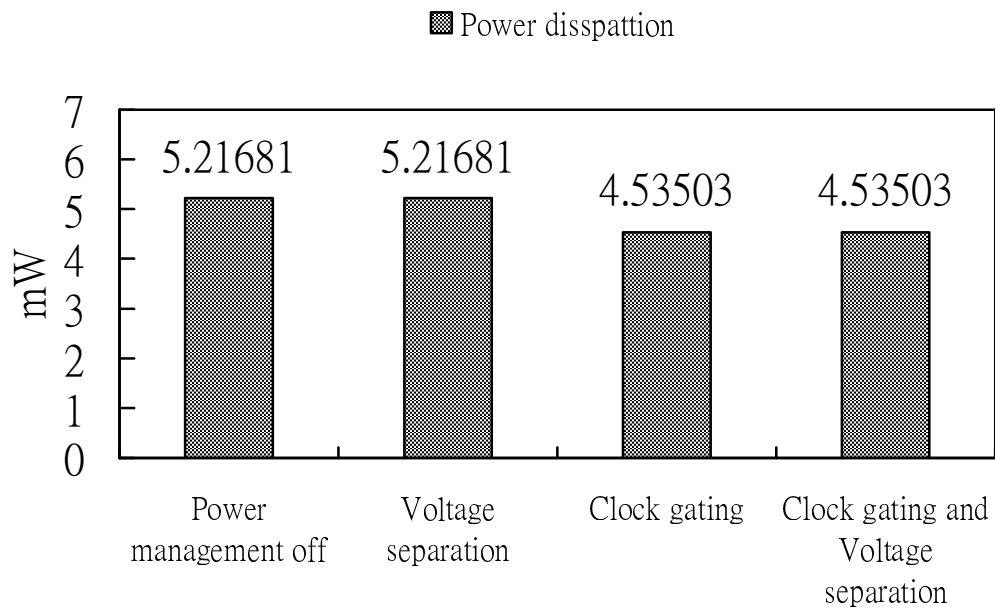


Figure 5.2.5 Total power dissipation of matrix calculation with all functional

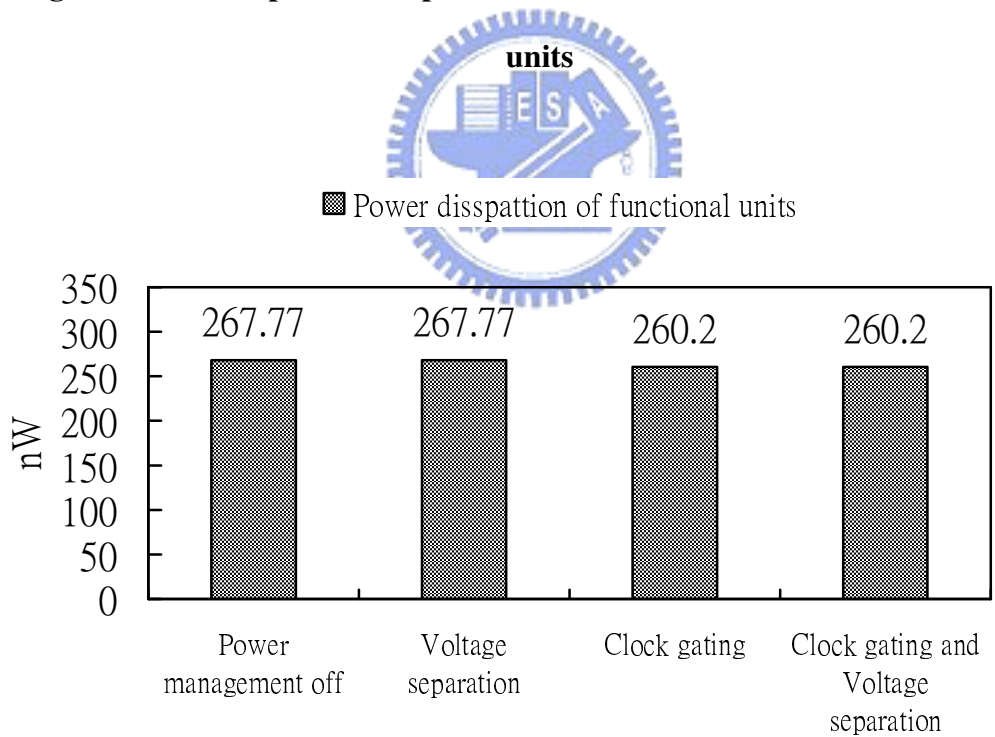


Figure 5.2.6 Total power dissipation of the three functional units

Table 5.2.9. The power analysis of matrix calculation with all functional units

Program	$C = 4A + 3B, D = 4A - 3B$. A,B,C,D are 5x5 matrix
Functional units used	All
Execution time	2910 ns
Percentage of stall instruction	35 %
Total power dissipation without P.M	5.21681mW
Total power dissipation with P.M	4.53503 mW
Power saved	13.1 %

5.2.4 Matrix Calculation with Functional Unit MAC

The execution of matrix calculation with MAC takes 3060 ns, and the power dissipation of the three functional units and the rest elements in data path are listed in Table 5.2.10 and Table 5.2.11. The numerical values in Table 5.2.10 are the simulation result which the power management is turned off, and these in Table 5.2.11 are the simulation result with clock gating (C.G.).

Table 5.2.10 Power distribution of matrix calculation with MAC (without P.M.) :

Part	Power (mW)	Percentage
Rest elements in data path	4.93033	92.4%
ALU	0.00529	0.1 %
L/S	0.04674	0.8%
MAC	0.35182	6.6%
Total	5.33419	

Table 5.2.11 Power distribution of matrix calculation with MAC (with C.G.) :

Part	Power (mW)	Percentage
Rest elements in data path	4.07782	91.2%
ALU	0.00529	0.1 %
L/S	0.04460	0.9%
MAC	0.34409	7.7%
Total	4.47181	

Figure 5.2.7 shows the total power dissipation comparison of with and without power management, and Figure 5.2.8 shows the one of the three functional units. The total power saved by clock gating and is about 16.1 %. In this case, the major functional units used are L/S and MAC, and the static power dissipation of ALU could be saved by voltage separation. But ALU still be needed when the branch instruction comes. The power may be switched often and the overhead may be larger than the static power saved, so that we assume that the power supply of ALU never be turned of during execution. This is why the power dissipation voltage separation shown in Figure 5.2.7 and Figure 5.2.8 are the same. Finally the simulation result is listed in Table 5.2.12.

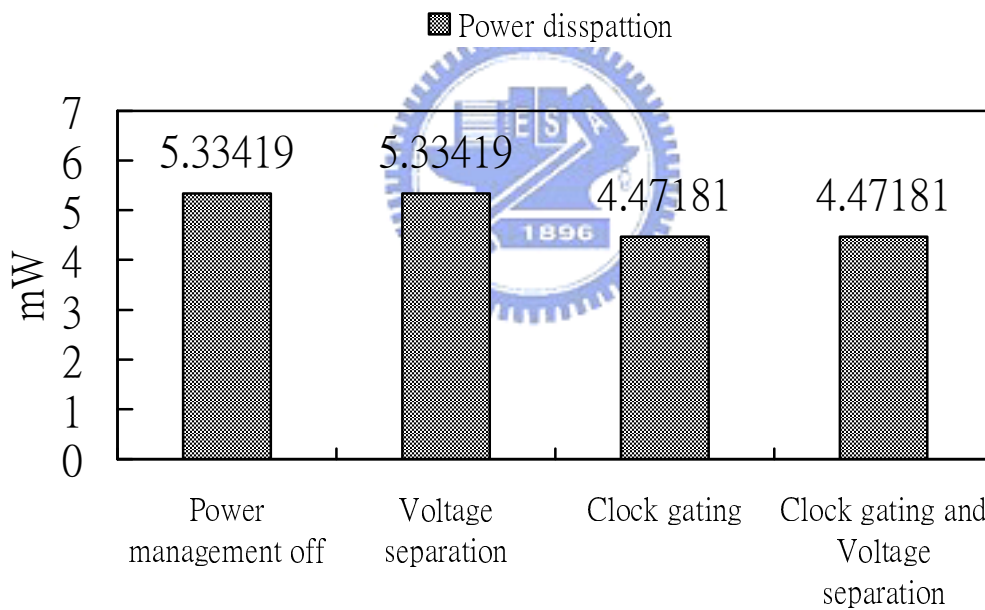


Figure 5.2.7. Total power dissipation of matrix calculation with MAC

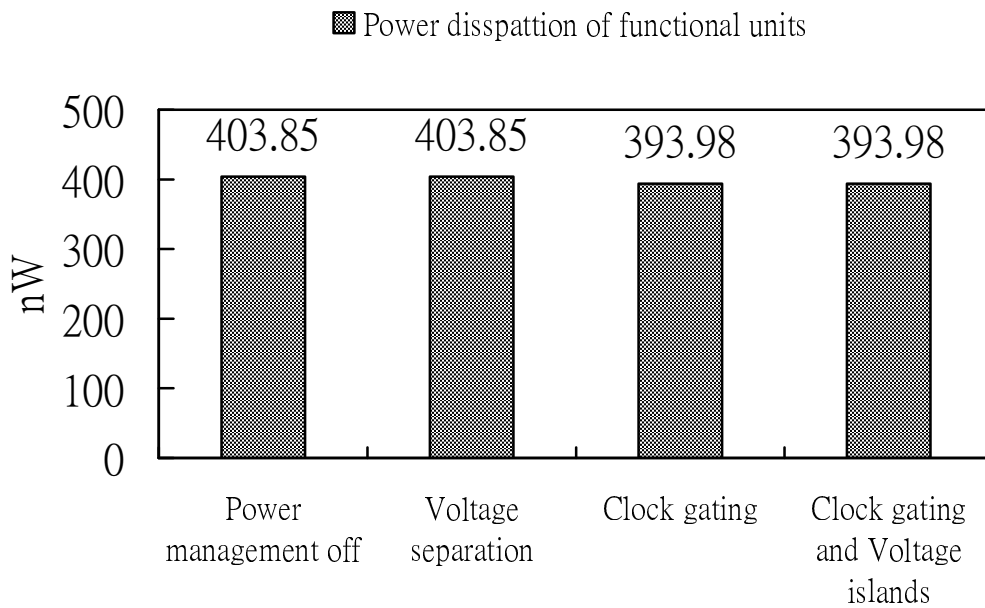


Figure 5.2.8. Total power dissipation of the three functional units

Table 5.2.12. The power analysis of calculation with MAC

Program	$C = 4A + 3B, D = 4A - 3B$. A,B,C,D are 5x5 matrix
Functional units used	MAC, L/S
Execution time	3060 ns
Percentage of stall instruction	46 %
Total power dissipation without P.M	5.33419mW
Total power dissipation with P.M	4.47181mW
Power saved	16.1 %

5.2.5 Matrix Calculation with Functional Unit ALU

The execution of matrix calculation with ALU takes 4814 ns, and the power dissipation of the three functional units and the rest elements in data path are listed in Table 5.2.13 and Table 5.2.14. The numerical values in Table 5.2.13 are the simulation result which the power management is turned off, and these in Table 5.2.14 are the simulation result with clock gating.

Table 5.2.13. Power distribution of matrix calculation with ALU (without P.M.)

Part	Power (mW)	Percentage
Rest elements in data path	4.79300	96.3%
ALU	0.15397	3.1%
L/S	0.02734	0.5%
MAC	0.00011	<0.001%
Total	4.97536	

Table 5.2.14. Power distribution of matrix calculation with ALU (with C.G)

Part	Power (mW)	Percentage
Rest elements in data path	3.83297	95.5%
ALU	0.15136	3.8 %
L/S	0.02572	0.6%
MAC	0.00011	<0.001%
Total	4.01114	

Figure 5.2.9 shows the total power dissipation comparison of with and without power management. This program doesn't use the functional unit MAC, and the static power dissipation of MAC can be saved by voltage separation. But the scale of the static power of MAC is too small in comparison with the total power, so that the difference between with and without voltage separation is not very obviously. The power saved by clock gating and voltage separation is about 19.4 %.

Figure 5.2.10 shows the difference in total power dissipation of the three functional units with and without power management. The simulation result is summarized in Table 5.2.15.

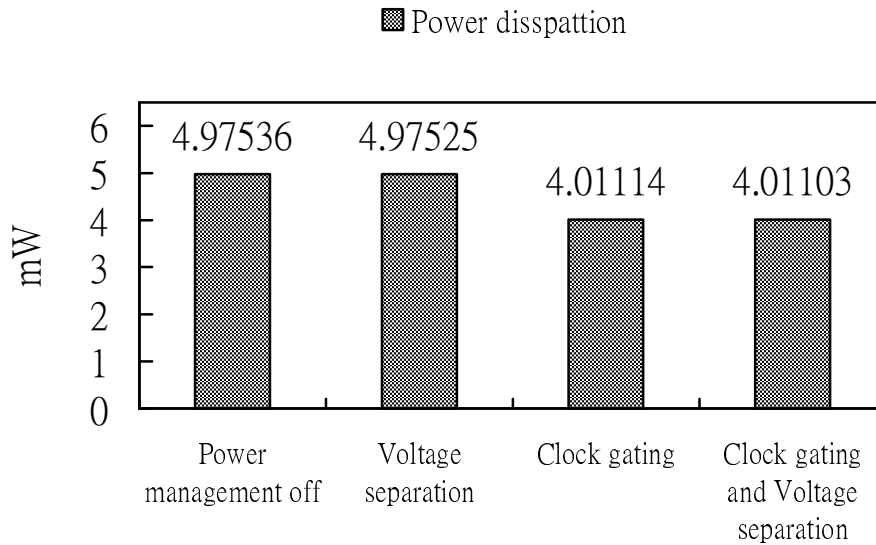


Figure 5.2.9. Total power dissipation of matrix calculation with ALU

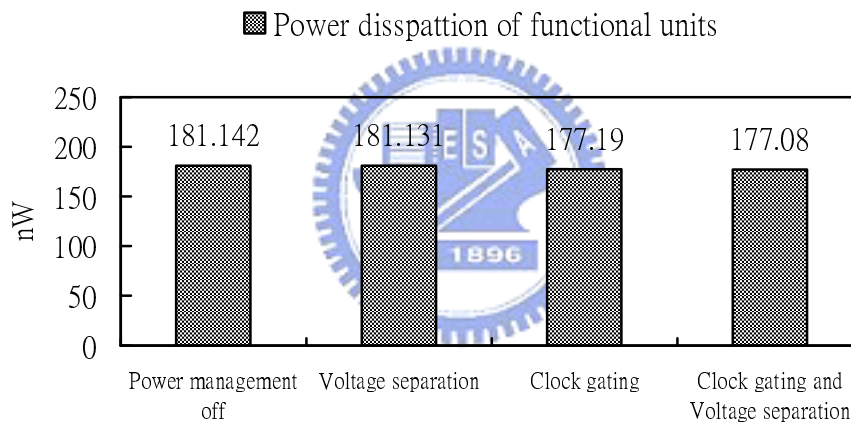


Figure 5.2.10. Total power dissipation of the three functional units

Table 5.2.15. The power analysis of matrix calculation with ALU

Program	$C = 4A + 3B, D = 4A - 3B$. A,B,C,D are 5x5 matrix
Functional units used	ALU, L/S
Execution time	4814 ns
Percentage of stall instruction	54 %
Total power dissipation without P.M	4.97536mW
Total power dissipation with P.M	4.01103mW
Power saved	19.4 %

5.2.6 Comparison

In this section, we will show the comparison of power dissipation, execution time and energy consumption of matrix calculation with different functional units. Figure 5.2.11 shows the power dissipation of the three cases. While the power management is turned off, the power dissipation of the case with MAC is the largest. This is because the major functional unit used in this case is MAC and the code density in this functional unit column is higher. When the clock gating is turned on, the power dissipation of the case with all functional units become the largest, the case with MAC is the second, and the case with ALU is the smallest. This is because the percentage of stall instruction in the case with ALU is the largest, and is the smallest in the case with all functional units. While the clock gating is applied, the power dissipation saved by clock gating in the case with MAC is much larger than the power consumption in MAC. So the power dissipation of the case with MAC is smaller than the case with all functional units. When the power management of voltage separation is applied, only the power dissipation of the case with ALU is reduced. This is because we assume that the static power of MAC is saved. But the scale of the power saved by voltage separation is too small in comparison with total power, so that the difference between with and without voltage separation is not obviously.

Figure 5.2.12 shows the required time for the three cases. The execution time of the case with all functional units is the shortest and the case with MAC is the second. In the case with ALU only functional units ALU and L/S are used, and require the longest execution time.

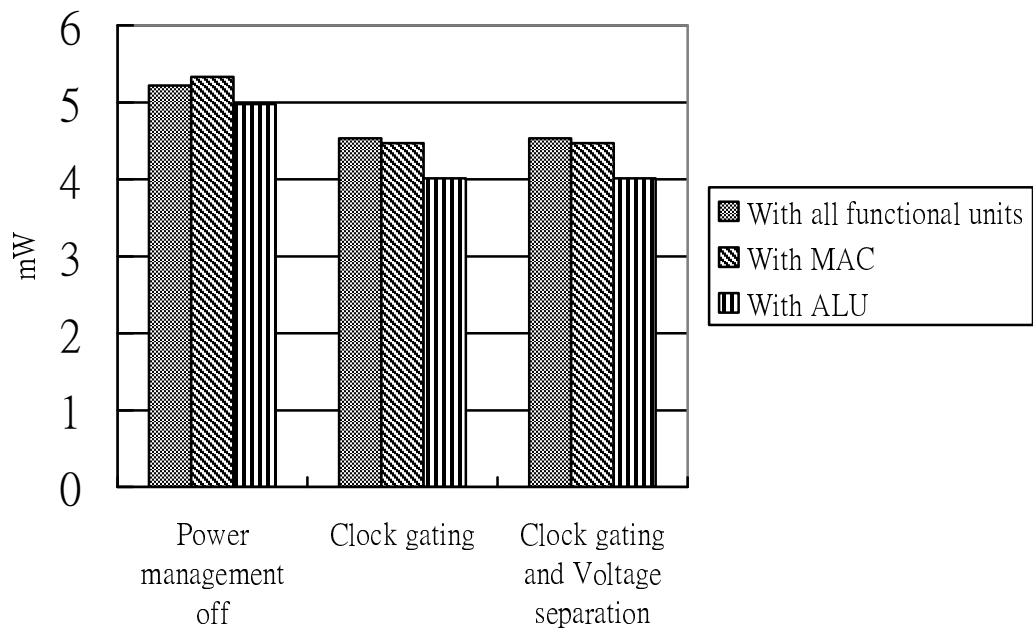


Figure 5.2.11. The power dissipation of matrix calculation

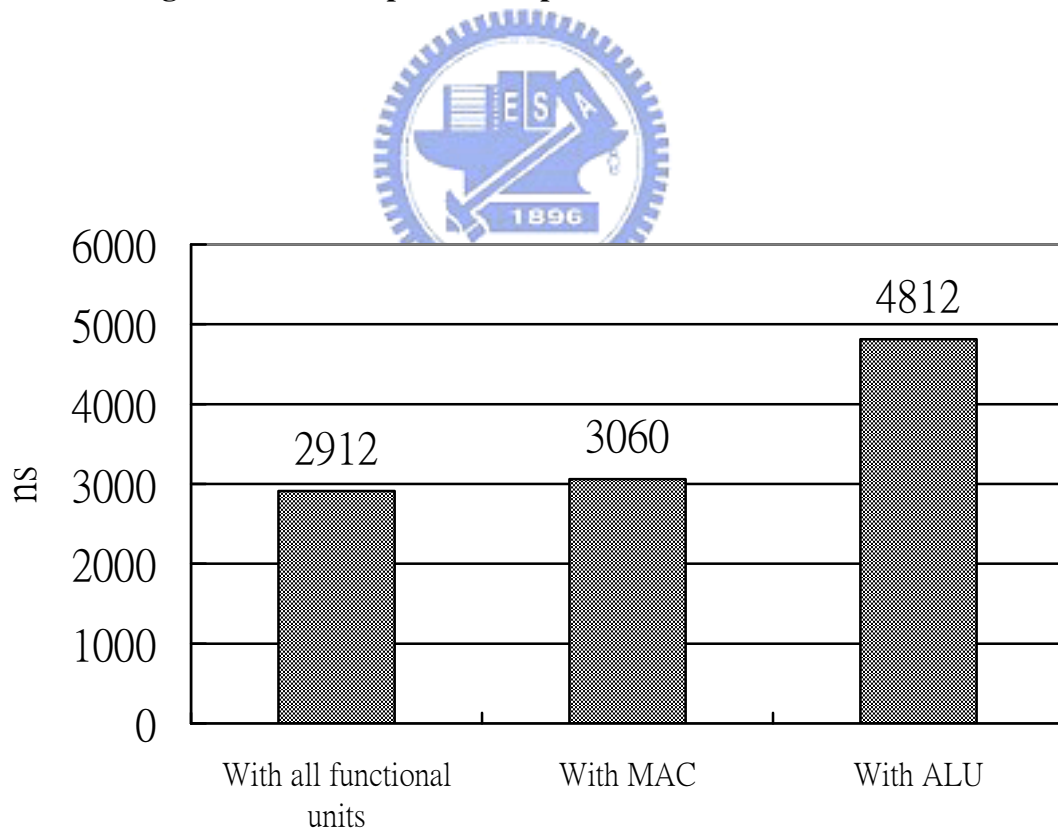


Figure 5.2.12. The execution time of the three cases

The energy consumption of the three cases is shown in Figure 5.2.13. The case with all functional units consumes the least energy in any power management mode. Although the power dissipation of this case is the largest, but the required execution time is the shortest. The power dissipation of the case with ALU is the smallest, but the required execution time is much longer than the other cases. So the case with ALU cost the largest energy. The power dissipation of the case with MAC is smaller than the case with all functional units, but the execution time is longer and the difference between execution time is larger than that in power dissipation. So that the energy consumed by the case with MAC is still larger than the case with all functional units.

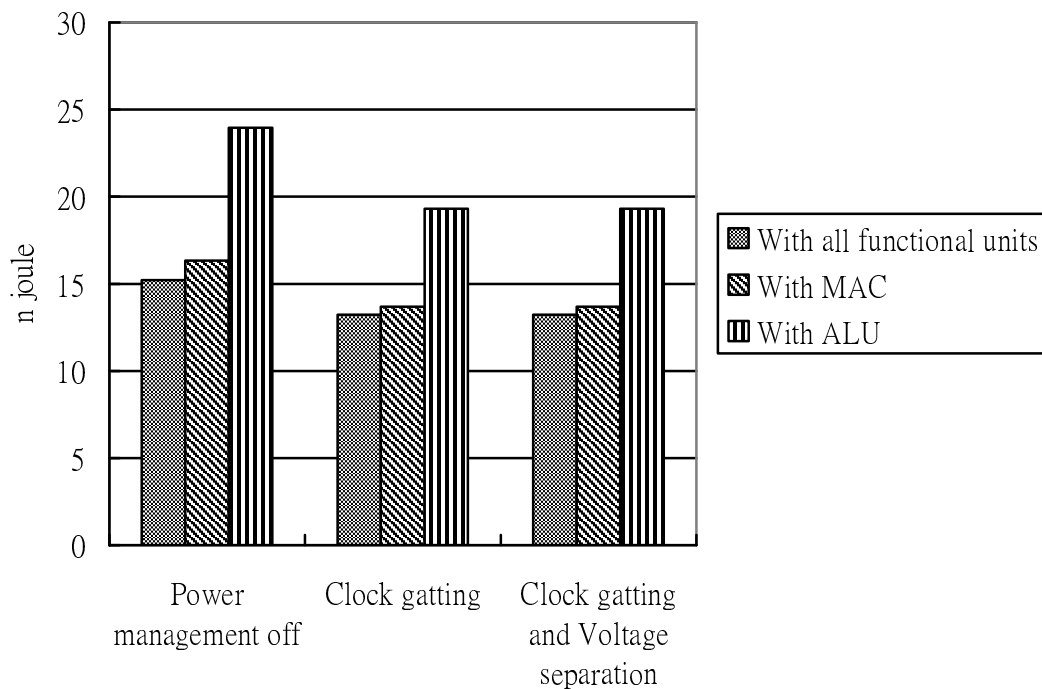


Figure 5.2.13. The energy consumption of the three cases

When executing the same program, the execution time will increase as the number of the functional units decrease. But there will not be large difference in power dissipation while there is no power management. Even in some case the power dissipation become larger. However, while power management of clock gating and voltage separation is applied in this data path, the power dissipation will decrease as

the number of the functional units decrease, furthermore the power dissipation and the performance becomes scalable.

The energy consumption of the data path will depend on both the power dissipation and required execution time. From the simulation result shown in Figure 5.2.13, the required execution time and energy consumption will be the largest if the functional unit MAC is not contained in the parallel data path and the program can only be completed with ALU. Thus, if more functional units are included in the parallel data path and the functionality of the data path can become more powerful, than a program execution could be completed more rapidly and the total energy consumption could also be decreased since the execution time can be largely reduced. Furthermore, if the system have to lower the power dissipation of the data path, than only use a portion of the functional units and turn off the idle functional units and the related elements with dynamic power management. Although that will increase execution time, yet the power dissipation can be reduced. This characteristic of power and performance scalable will be suitable for a data path of a powerful VLIW processor in a reconfigurable architecture.

5.3 Summary

In this chapter, we have shown the power simulation results of the five benchmarks. As shown in this chapter, the power dissipation of the parallel data path is strongly related to the percentage of stall instruction. If the percentage of stall instruction is 100 %, 47 % of power dissipation is saved. If the percentage of stall instruction is 56 %, 21 % of power dissipation is saved. If the percentage of stall instruction is 54 %, 19.4 % of power dissipation is saved. If the percentage of stall instruction is 46 %, 16.1 % of power dissipation is saved. If the percentage of stall

instruction is 35 %, 13.1 % of power dissipation is saved. Clock gating technique can save a significant amount of power in this chip. However, power saved by voltage separation is not very obviously. Dynamic power can be largely reduced by dynamic power management and the static power consumption is still not a serious problem in the present 0.18 um process. Furthermore, we execute a program of matrix calculation with different functional units. From the simulation result, the power dissipation and performance of the parallel data path become scalable when dynamic power management is applied.



Chapter 6 Conclusion

Platform-based design and reconfigurable architecture are useful design methods for SoC design. The microprocessor is the most important element in these two design methods. In recent research, VLIW processor can provide general purpose DSP calculation such as DCT, motion estimation, FIR filter...etc. And these DSP calculations are common blocks for communication and media applications. If VLIW processor is applied on these two design methods, the general purpose DSP functional units can be replaced by VLIW processor, and the design complexity in hardware can be reduced. However, the power issue in VLIW processor due to low code density and hardware utilization can cause a great overhead. So the power management is necessary.

In this thesis, we have successfully applied the clock gating and voltage separation with the parallel data path which is similar to a data path in a three-issue VLIW processor. An appropriate design flow to implement these two dynamic power managements with current EDA tool for cell-base design is explored. A complete power analysis is accomplished and the analysis result of the chip is identical to our expectation. From the analysis result, three conclusions have been made. First, the static power dissipation is not a serious problem in the current 0.18 um process. Second, clock gating can reduce significant power dissipation which is strongly related to the percentage of stall instruction. More stall instructions, more power dissipation can be saved. In the best case, 47 % of power dissipation is saved. Third, the power and performance of the parallel data path is scalable and that is identical to that we expected.

Furthermore, as mentioned in Chapter 3, the penalty of circuit charge and discharge for voltage separation will be measured with this chip, and then design the

compiler according to the measurement. In the further, the static power consumption is superposed to be a serious problem in more advanced process. So the voltage separation with MTCMOS technology will be integrated into the parallel data path in 0.13 um or 0.09 um process and the effect of the static power reduction will be measured in future project.



Bibliography:

- [1] L. Hennessy and A. Patterson, *Computer Architecture: A Quantitative Approach*, Third Edition, Morgan Kaufmann Publishers, 2003.
- [2] Fabio Campi, Mario Toma, Andrea Lodi, Andrea Cappelli, Roberto Cangeallo, Roberto Guerrieri, "A VLIW Processor with Reconfigurable Instruction Set for embedded Applications," 2003 IEEE International Solid-State Circuits Conference.
- [3] Rohini Krishnan, O.P.Gangwal, Jos.v.Eindhoven, and Anshul Kumar, "Design of a 2D DCT/IDCT Application Specific VLIW Processor Supporting Scaled and Sub-sampled Blocks," Proceeding of the 16th International Conference on VLSI Design (VLSI'03), 2003 IEEE.
- [4] K. Heutzer, S. Malik, A. R. Newton, J. M. Rabaey, and A. Sangiovanni-Vencentelli, "System Level Design: Orthogonalization of Concerns and Platform-based Design," invited paper, IEEE Transactions on Computer-Aided Design, Vol. 19, No. 12, December 2000.
- [5] A. Sangiovanni-Vincentelli, L. Edgar, H. Buttner, "Platform-based Design," University of California at Berkeley.
- [6] M. Keating and P. Bricaud, *Reuse Methodology Manual for System-on-a-Chip Designs*, Third Edition, Kluwer Academic Publishers, 2002.
- [7] <http://www.arm.com/>
- [8] <http://www.semiconductors.philips.com/products/nexperia/>
- [9] Tay-Jyi Lin, Chin-Chi Chang, Chen-Chia Lee, and Chein-Wei Jen, "An Efficient VLIW DSP Architecture for Baseband Processing," Proceedings of the 21 st International Conference on Computer Design (ICCD'03), 2003 IEEE.
- [10] T. Inukai, T. Hiramoto, T. Sakurai, "Variable threshold voltage CMOS (VTCMOS) in series connected circuits" International Symposium on, 2001, Page(s): 201 -206.
- [11] R. Fujioka, K. Katayama, R. Kobayashi, H. Ando, T. Shimada, " A preactivating mechanism for a vt-cmos cache using address prediction, " . ISLPED '02. Proceedings of the 2002 International Symposium on , 2002 Page(s): 247 -250.
- [12] S. Mutoh, T. Douseki, Y. Matsuya, T. Aoki, S. Shigematsu, and J. Yamada, "1-V power supply high-speed digital circuit technology with multi-threshold-voltage CMOS," IEEE J. Solid-State Circuits. 847-854, Aug. 1995.
- [13] S. Shigematsu et al, "A 1V High-Speed MTCMOS Circuit Scheme for Power-Down Applications," IEEE J. Solid-State Circuits, vol. 32, no. 6, June 1997, pp. 861-869.

- [14] Mohab H. Anis and Mohamed W. Allam and Mohamed I. Elmasry
 “Energy-Efficient Noise-tolerant Dynamic Styles for scaled-Down CMOS and MTCMOS Technologies,” IEEE Transactions on very large scale integration system, VOL. 10, NO. 2, April. 2002.
- [15] Benton H. Calhoun, Frank A. Honore, and Anatha P. Chandrakasan, Fellow, IEEE, “A Leakage Reduction Methodology for Distributed MTCMOS,” IEEE J. Solid-State Circuits, vol. 39, NO. 5, May 2004.
- [16] Qing Wu, Massoud Pedram, and Xunwei Wu, “Clock-Gating and Its Application to Low Power Design of Sequential Circuits,” IEEE Transactions on Circuits and Systems—I: Fundamental Theory and Applications, vol. 47, NO. 103, March 2000.
- [17] Wu Ye and Mary Jane Irwin, “Power Analysis of Gated Pipeline Registers,” Computer Science and Engineering Department, University of Pennsylvania State, 1999.
- [18] T. Burd, T. Pering, A. Stratakos, R. Brodersen, “A Dynamic Voltage-Scaled Microprocessor System,” 2000 IEEE International Solid-State Circuits Conference Digest of Technical papers, Feb 2000.
- [19] T.D. Burd, R.W. Brodersen, “Design issues for Dynamic Voltage Scaling,” 2000. ISLPED '00. Proceedings of the 2000 International Symposium on , 2000 Page(s): 9 -14.
- [20] V. Gutnik, A. Chandrakasan, “An efficient controller for variable supply-voltage low power processing,” 1996. Digest of Technical Papers, 1996 Symposium on , 13-15 Jun 1996 Page(s): 158 -159.
- [21] David E. Lackey, Paul S. Zuchowski, Thomas R. Bednar, Douglas W. Stout, Scott W. Gould, John M. Cohn “Managing Power and Performance for System-on-Chip Designs using Voltage Islands,” IBM Microelectronics Division Essex Junction, Vermont 05452, USA.
- [22] Juan-Antonio Carballo, Jeffery L. Burns, Seung-Moon Yoo, Ivan Vo, and V. Robert Norman, “A Semi-Custom Voltage-Island Technique and Its Application to High-Speed Serial Links,” IBM Microelectronics 3039 Cornwallis Road Raleigh, NC 27709, USA.
- [23] J. Tschanz, J. Kao, S. Narendra, R. Nair, D. Antoniadis, A. Chandrakasan, Vivek De, “Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage,” ISSCC. IEEE International , Volume: 1 , 2002 Page(s): 422 -478.
- [24] J.T. Kao, M. Miyazaki, A.R. Chandrakasan, “A 175-MV multiply-accumulate unit using an adaptive supply voltage and body bias architecture,” Journal of Solid-State Circuits, IEEE Journal of , Volume: 37 Issue: 11.

- [25] M. Miyazaki, J. Kao, A.P. Chandrakasan, "A 175mV multiply-accumulate unit using an adaptive supply voltage and body bias (ASB) architecture, " Solid-State Circuits Conference, 2002. Digest of Technical Papers. ISSCC. 2002 IEEE International , Volume: 2 , 2002 Page(s): 40 -391.
- [26] <http://www.trimedia.com/>
- [27] <http://www.transmeta.com/>
- [28] Chip Implementation Center, Taiwan, <http://www.cic.org.tw>.

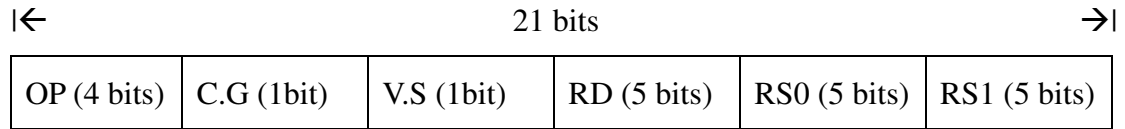


Appendix A Instruction set summary

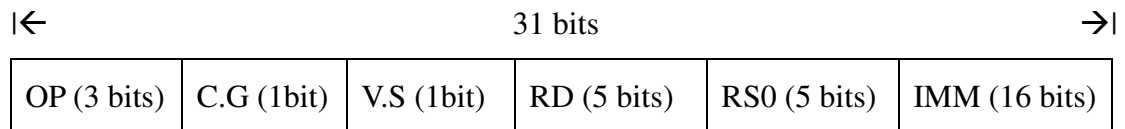
Functional Unit	Instruction	Example	Action	OP code
ALU	ADD	ADD RD RS0 RS1	$RD = RS0 + RS1$	1000
	SUB	SUB RD RS0 RS1	$RD = RS0 - RS1$	1001
	AND	AND RD RS0 RS1	$RD = RS0 \& RS1$	1010
	OR	OR RD RS0 RS1	$RD = RS0 RS1$	1011
	XOR	XOR RD RS0 RS1	$RD = RS0 \wedge RS1$	1100
	SLL	SLL RD RS0 3	$RD = RS0 \ll 3$	1101
	SRL	SRL RD RS0 3	$RD = RS0 \gg 3$	1110
	SRA	SRA RD RS0 3	$RD = \text{sign extend}\{RS0 \gg 3\}$	1111
	JR	JR RD	$PC = RD$	0100
	JUMP	JUMP 50	$PC = PC + 50$	0110
	JAL	JAL RD 50	$RD = PC+1, PC = PC + 50$	0101
	BEQ	BEQ RD 50	If ($RD = R0$) go to $PC+50$	0111
LOAD/STORE	ADDI	ADDI RD RS IMM	$RD = RS + IMM$	010
	SUBI	SUBI RD RS IMM	$RD = RS - IMM$	011
	LOAD	LOAD RD RS IMM	$RD = \text{MEM}(RS + IMM)$	100
	STORE	STORE RD RS IMM	$\text{MEM}(RS + IMM) = RD$	101
MAC	MAC	MAC RD RS0 RS1	$RD = RD + (RS0 \times RS1)$	10
	MUL	MUL RD RS0 RS1	$RD = RS0 \times RS1$	11

Appendix B Instruction format

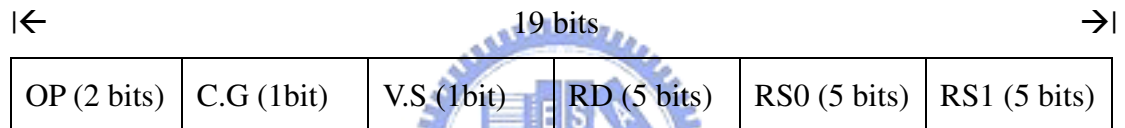
ALU :



LOAD/STORE:



MAC:



OP	Operation code
C.G.	Clock gating control bit
V.S	Voltage separation control bit
RD	Destination register
RS	Source register
IMM	Immediate data