

國立交通大學

電子工程學系 電子研究所碩士班

碩士論文

低功率時間共享多線程暫存器



Low Power Timing Sharing Multithreaded Register File

研究生：郭于玄

指導教授：黃威 教授

中華民國九十七年八月

低功率時間共享多線程暫存器

Low Power Timing Sharing Multithreaded Register File

研究生：郭于玄

Student : U-Chan Kuo

指導教授：黃 威教授

Advisor : Prof. Wei Hwang

國立交通大學

電子工程學系電子研究所

碩士論文

A Thesis

Submitted to Department of Electronics Engineering & Institute of Electronics
College of Electrical Engineering and Computer Engineering
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in
Electronics Engineering
August 2008
Hsinchu, Taiwan, Republic of China

中華民國九十七年八月


低功率時間共享多線程暫存器

學生：郭于玄

指導教授：黃威教授

國立交通大學電子工程學系電子研究所

摘 要



本論文提出一個低功率多線程的暫存器設計。此暫存器被切成數個小區塊，並且應用了時間共享的機制去增加它的效能。為了節省功率的消耗，提出了免充電位元線和切割小位元線的方法。而且，此暫存器可以正確操作在大範圍的工作電壓下，可依效能和功率的要求去做電壓上的調整。此二線程的暫存器包括了四個存／取埠，每個線程有 64x64 個位元大小，並以 TSMC 90nm CMOS 的製程技術做設計，實現在 426 x 219 μm^2 的面積上。它的工作電壓範圍在 0.5 伏到 1.0 伏。當時脈為 50MHz，它工作所消耗的功率在 215.51 微瓦 和 197.77 微瓦之間。

Low Power Timing Sharing Multithreaded Register File

Student : U-Chan Kuo

Advisors : Prof. Wei Hwang

Department of Electronics Engineering & Institute of Electronics
National Chiao-Tung University

ABSTRACT

A low-power multithreaded register file architecture is proposed. Banking architecture and timing sharing access scheme are adopted to reduce the register file area and increase its performance. Floating bitline scheme and divide bitline is also presented to reduce its active power. Furthermore, the register file architecture can be operated at a wide voltage range, and processors would have more strategies to adjust their power/performance. A dual-thread 4W/4R 64x64-bit register file which occupies **426 x 219 μm^2** silicon area is implemented in UMC 90nm CMOS technology. Its operating voltage range is between 0.5v and 1.0v. Its active power is around 215.28 μW to 197.87 μW when operating frequency is 50MHz at 0.5v.

誌 謝

如果沒有許多人的幫忙，這篇論文將無法順利完成，感謝指導教授黃威老師，提供了研究的方向，並在研究過程中，給了許多重要且關鍵的建議及指導，並讓我們學習到做研究該有的態度及方法。

感謝指導學長楊皓義，提供了許多的創意及想法，並且在研究遇到困難時，能不吝嗇的給予最大的協助。此外，要感謝黃柏蒼，張銘宏、謝維致學長們適時的幫助，讓研究得以順利進行。感謝實驗室的同學和朋友們為單調的研究生生活增添了色彩。

最後，感謝家人對我的支持和鼓勵，有了家人，才有今天的我。



Contents

Chapter 1	Introduction	1
Chapter 2	Overview of recent low-power- register-file technology	3
2.1	The leakage of a register file cell	4
2.2	Low power register file design	8
2.3	Conventional Register file Architecture	13
2.3.1	The Read Port Design	14
2.3.2	Low Power Write Port Design	15
2.4	Banked Register file Architecture	17
2.4.1	Register bank structure	18
2.4.2	Methods to decrease access conflicts.....	19
2.5	Other kind of multibanked Architecture.....	22
2.5.1	Customization of Register File Banking Architecture	23
2.5.2	Asymmetrically Banked Value-Aware Register Files	24
2.6	Conclusion	25
Chapter 3	Low power Multithreaded Register File Design	26
3.1	Multithreaded Banking Architecture.....	26
3.1.1	Register Bank structure.....	26
3.1.2	Timing sharing access scheme.....	27
3.1.3	Thread switching.....	29
3.2	Register file cell.....	30
3.2.1	Read Stability and Write-Ability.....	30
3.2.2	The 8T cell.....	34
3.2.3	Other register file cells.....	35
3.3	Low Power Floating Read Bitline Access Scheme.....	38
3.3.1	Floating Read Scheme.....	38
3.3.2	Divided Bitline.....	40
3.3.3	Push-Pull Write Scheme.....	43
3.4	Conclusion.....	44
Chapter 4	Decoder and Control Circuit Design.....	46
4.1	Decoder design.....	47
4.2	Logical effort model.....	52
4.2.1	Logical effort and gate sizing.....	53
4.2.2	Logical effort design steps.....	56
4.3	Timing Control Circuit.....	57
4.3.1	Write Replica Circuit.....	58
4.3.2	Read Replica Circuit.....	61
4.4	Comparison with conventional register file.....	64
4.5	The post-simulations.....	66
4.6	Conclusion.....	67

Chapter 5	Multithreading and Multi-core systems.....	68
5.1	Different type of Register file organizations.....	68
5.1.1	Clustered architecture.....	68
5.1.2	Duplicated Register File.....	70
5.1.3	Multilevel Register File.....	71
5.1.4	One-Level Less-Port register file Architecture.....	73
5.2	Multithreading.....	75
5.2.1	Fine-grained multithreading.....	77
5.2.2	Coarse-grained multithreading.....	77
5.2.3	Simultaneous multithreading.....	78
5.3	Multiprocessors.....	81
5.3.1	Multicore architecture and communication.....	82
5.4	Conclusion	86
Chapter 6	Conclusion	87
Bibliography	89
Vita	98



List of Figures

Fig. 2.1	Transistor gate length versus technology nodes for high-performance and ULP CMOS technologies.	5
Fig. 2.2	Thickness of transistor gate oxide versus technology nodes for high performance and ULP CMOS technologies.	6
Fig. 2.3	Leakage model in 6T cell.	7
Fig. 2.4	The possible nodes used to decrease the leakage current.	9
Fig. 2.5	VGND control schemes (a) Sleep-transistor (b) Diode-connected PMOS bias transistor. (c) Programmable bias transistors. (d) Op-Amp-based -feedback control.	10
Fig. 2.6	Multi-VCC designs. (a) Register file on a higher supply voltage. (b) Register file on a higher supply voltage than the wordline driver (c) Dynamic multi-VCC.	12
Fig. 2.7	Using inverter to isolate memory cells and pass transistors.....	14
Fig. 2.8	Using read buffers to sink read current.....	15
Fig. 2.9	Instantaneous short current in write operation.....	16
Fig. 2.10	Strong inverter	17
Fig. 2.11	An 8-read, 4-write port register file implemented using four two-read, one-write port banks.	18
Fig. 2.12	Structure of the processor pipeline.....	21
Fig. 2.13	Combine access to the same registers.....	22
Fig. 2.14	Heterogeneous Register File Banking.....	23
Fig. 2.15	A microarchitecture-level comparison among (a) a conventional banked register file and (b) the AB-VARF register file.	24
Fig. 3.1	The proposed dual-thread 4W/4R 64 x 64 bit register file architecture. ...	26
Fig. 3.2	Waveforms illustrating concept of timing sharing access scheme.....	28
Fig. 3.3	Thread switching scheme of Montecito microprocessor.....	29
Fig. 3.4	The location of registers.....	30
Fig. 3.5	SNM definition. The two DC noise voltage sources V_n are placed in series with the cross-coupled inverters.	31
Fig. 3.6	Static Noise Margin.....	32
Fig. 3.7	The 6T cell.....	32
Fig. 3.8	The write margin is defined by the write-trip point.....	33
Fig. 3.9	8T cell structure.....	34
Fig. 3.10	The Monte Carlo simulation result of 8T's noise margin at 0.5v. The sigma of V_t is 30mV (a) Write margin distribution. (b) Read noise margin distribution.	35
Fig. 3.11	Schematic of the 10 T bitcell.	36
Fig. 3.12	RF cell with conventional read circuit. An additional transistor, P_{gate} , is added to aid write margin in subthreshold.	37
Fig. 3.13	The auto-gating low-leakage cell.....	38
Fig. 3.14	Floating bitline access scheme.....	39
Fig. 3.15	Divide bitline connection (a) By pass transistor (b) By read buffer	40

Fig. 3.16	Access time with different word numbers per bank (a) Write access time. (b)Read access time.	42
Fig. 3.17	write scheme (a) conventional write scheme (b)floating write scheme (c)Push-pull write scheme.....	43
Fig. 4.1	Block diagram of proposed register file.....	46
Fig. 4.2	Write row decoder and block decoder.....	48
Fig. 4.3	Write block decoder.....	49
Fig. 4.4	Waveforms illustrating concept of block decoder.....	50
Fig. 4.5	The row decoder design (a) Conventional design. (b) The scheme with fewer logic gate.	51
Fig. 4.6	A logical path with branching.....	54
Fig. 4.7	Calculating the electrical effort of a path.	55
Fig. 4.8	(a)Write replica circuit. (b) Signal waveforms of this circuit.....	59
Fig. 4.9	State diagram of write replica circuit.....	60
Fig. 4.10	The organization of ARC.....	61
Fig. 4.11	(a)Read replica circuit. (b) Signal waveforms of this circuit.....	62
Fig. 4.12	State diagram of Read replica circuit.....	63
Fig. 4.13	A 4W/4R register file cell.	65
Fig. 4.14	Layout photograph of register file cell. (a) A 1W/1R register file cell. (b) A 4W/4R register file cell.	65
Fig. 4.15	The comparison between this work and conventional design (a) area (b) power consumption.	66
Fig. 4.16	Layout photograph of the dual thread 64 x 64 bits register file.	67
Fig. 5.1	Monolithic versus clustered register file organization.....	69
Fig. 5.2	4-thread, 2-read, 6-write, full-duplicate register file architecture.	70
Fig. 5.3	Multilevel register file (register file cache)	71
Fig. 5.4	Very Wide Register Organization.....	74
Fig. 5.5	How four threads use the issue slots of a superscalar processor in different approaches.	76
Fig. 5.6	The structure of a centralized shared-memory multiprocessor.	83
Fig. 5.7	The structure of a distributed-memory multiprocessor.	83
Fig. 5.8	Processor block diagram of Niagara2 SPARC processor.....	84
Fig. 5.9	The Niagara2 Crossbar.....	85

List of Tables

Table 4.1	Register file simulation result.....	66
-----------	--------------------------------------	----



Chapter 1

Introduction

A register file is one of the most important components of a multithreaded processor. Not only its access time dominates the processor speed, but also its area and power are the critical part of a processor. Conventionally, the area of a register file is positive to the port number of its cell. Power and access time of a register file become worse when its area is increasing. Therefore, many technologies have been proposed to reduce the port number of a cell, such as banking architecture [1.1].

Otherwise, register allocation for different threads is another issue of multithreaded register file design. In some design, such as [1.2], multiple cells share the same access port by using a multiplexer. It leads to the capacitance and access time of a storage node increasing. A storage node is also affected by noise easily. These drawbacks become more serious when the thread number increases.

The proposed multithreaded register file adapts banking architecture. By using this architecture, the port number of a cell is reduced, and area of the register file is also decreasing. However, banking architecture leads to access conflicts, and processor performance is degraded. In order to ease this drawback, timing sharing access scheme is proposed. Additionally, in order to prevent drawbacks induced by several cells sharing an access port on a bitline, every cell has its own private access port, and additional address bit is added as a thread switching signal during access operations.

In chapter 3, the architecture of the proposed register file and

the low power circuit design techniques are discussed. The stability of register cell is also shown in this chapter. In chapter 4, the decoder and timing control circuit design are presented. The simulation result is shown in the end of chapter 4. Different register file designs are discussed in chapter5 before the concluding in chapter 6.



Chapter 2

Overview of recent low-power-register-file technology

Because more and more portable devices are desired, to reduce power consumption is an important topic for discussion. A major factor in the weight and size of portable devices is the amount of batteries. Battery is directly impacted by the power dissipated of the electronic circuits.

Deep sub-micrometer or nanometer CMOS technologies limit dynamic energy dissipation. Scaling down the supply voltage and the threshold voltage offer a continuously higher level of integration and assure high speed. However, the significantly increasing subthreshold leakage is a drawback. Therefore, reducing the leakage power dissipation, without decreasing performance, is one of the major research topics in VLSI design. The design of fast and power efficient register file structures has become especially crucial.

Register file represent a substantial portion of power and area in modern processors, and are growing rapidly when the instruction issue width become wider. The trend toward simultaneous multithreading increases register count further. It is a serious problem that the area of conventional register file go more than quadratic with issue width. When in a conventional multi-port register file several data are accessed at the same time, for example, N addresses in one cycle, a processor needs N -port memory cells with N word-lines and N bitlines.

In this case, the chip size becomes huge because the quantity of wiring increases by the square of the number of ports. Therefore, design of a register file with many-ports causes problems such as

enlargement of chip size, deterioration of register access speed and high power consumption. In order to avoid the serious problem of area, power, and delay of conventional multiported register file design, many techniques have been proposed previously.

One approach divides the physical register file into several interleaved banks with fewer ports per bank and retains a centralized microarchitecture at the same time. Provided that the number of simultaneous accesses to any bank is less than the number of ports on each bank, this structure can provide the aggregate bandwidth needs of a superscalar machine and significantly reduce area compared to a fully multiported register file.

Even though bank conflicts cause a small performance penalty, which decrease IPC only by less than 5 % [2.4], the dramatic reductions in register file delay and power can potentially be used to increase the clock rate and lead to a more complexity-effective design. Banked register files are a natural solution to the increasing register file demands of simultaneous multithreading processors.

2.1 The leakage of a register file cell

When the size of the silicon technology is scaled below 100 nm, the transistor OFF-state leakage has started to take away a significant portion of the overall power budget in today's VLSI system [2.1].

In the modern low power chip design, one of the major challenges is the growing static leakage power. In order to provide improved performance and reduced power, transistor threshold voltage (V_T), gate oxide thickness (T_{ox}), and channel length have been scaled along with operating voltage over the last few decades.

Scaling of gate oxide thickness also increases the gate leakage. Gate leakage is no longer negligible to the overall chip power consumption. The power constraint due to the rising gate leakage has slowed down the scaling of gate oxide, which has made the control of 2-D short-channel effect even more challenging.

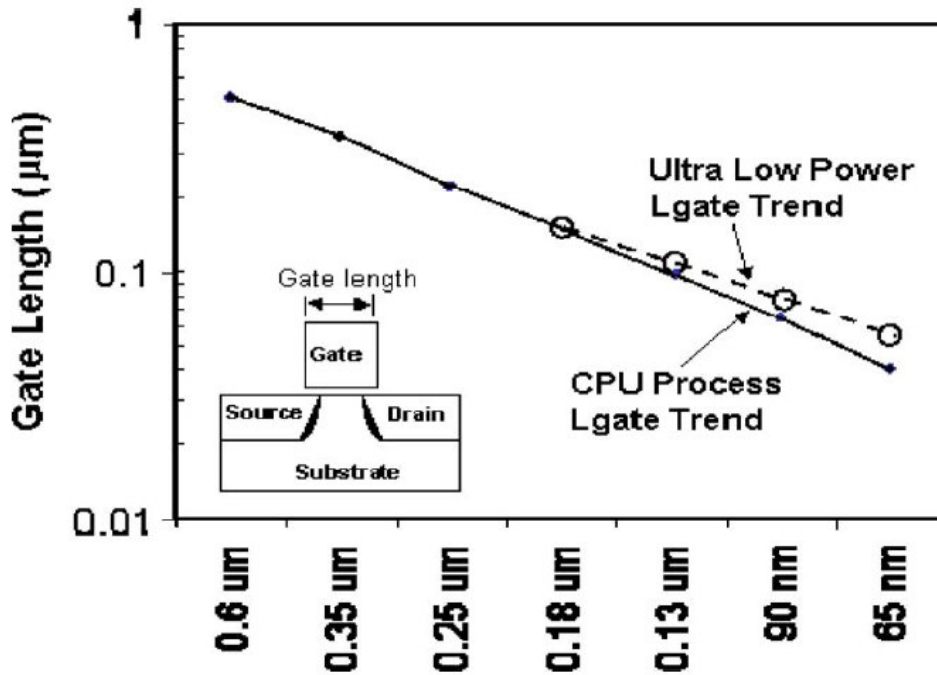


Fig. 2.1. Transistor gate length versus technology nodes for high-performance and ULP CMOS technologies.

Many advanced channel engineering techniques with optimized halo implants have been developed to achieve superior short-channel VT control. Traditionally, channel doping concentration is increased to raise VT in order to maintain low subthreshold leakage and good short-channel control. Unfortunately, the high doping concentration near the source and drain areas can lead to a very high junction leakage due to the direct band-to-band tunneling [2.5].

The increasing of transistor leakage had fueled the need of

developing low-power process technologies for mobile and handheld applications. The allowable power dissipation and, hence, the allowable leakage current are limited by battery life. The scaling of transistor gate length and T_{ox} of ultra low power technology has diverged from high-performance technology in recent generation. Figs. 2.1 and 2.2 show the scaling of transistor gate length and gate oxide thickness (T_{ox}) versus technology nodes for high-performance and ultra low-power (ULP) CMOS technologies.

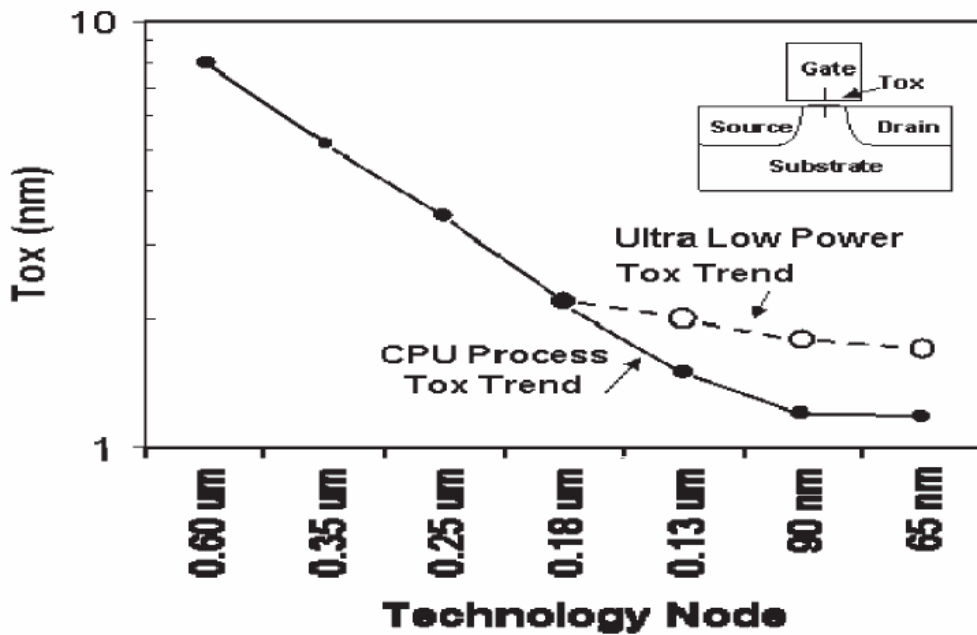


Fig. 2.2. Thickness of transistor gate oxide versus technology nodes for high performance and ULP CMOS technologies.

In order to achieve the low-power requirement of memory cell for portable applications, the ULP-technology platform needs to be adopted for its ultra low transistor leakage. Each leakage component, including the gate, subthreshold, and junction leakage, is optimized simultaneously to achieve best overall cell leakage. Fig. 2.3 shows schematically the various leakage paths in a 6T cell. When the cell is inactive, the word line WL is low level ("L") and bit lines BL and BLB are high level ("H"). One storage node of the cell is "H" and the other is "L."

Gate-leakage reduction is done mostly through the oxide-thickness optimization and gate nitridation. The gate leakage current increases exponentially as the physical thickness of gate insulation film becomes thinner in keeping with scaling. The gate leakage current of NMOS is 4-10 times greater than that of PMOS of the same thickness. The leakage mechanisms of junction and subthreshold current are influenced by many common processing parameters such as source-drain spacers, T_{ox} , well and halo implants, and doping profiles [2.2], [2.6].

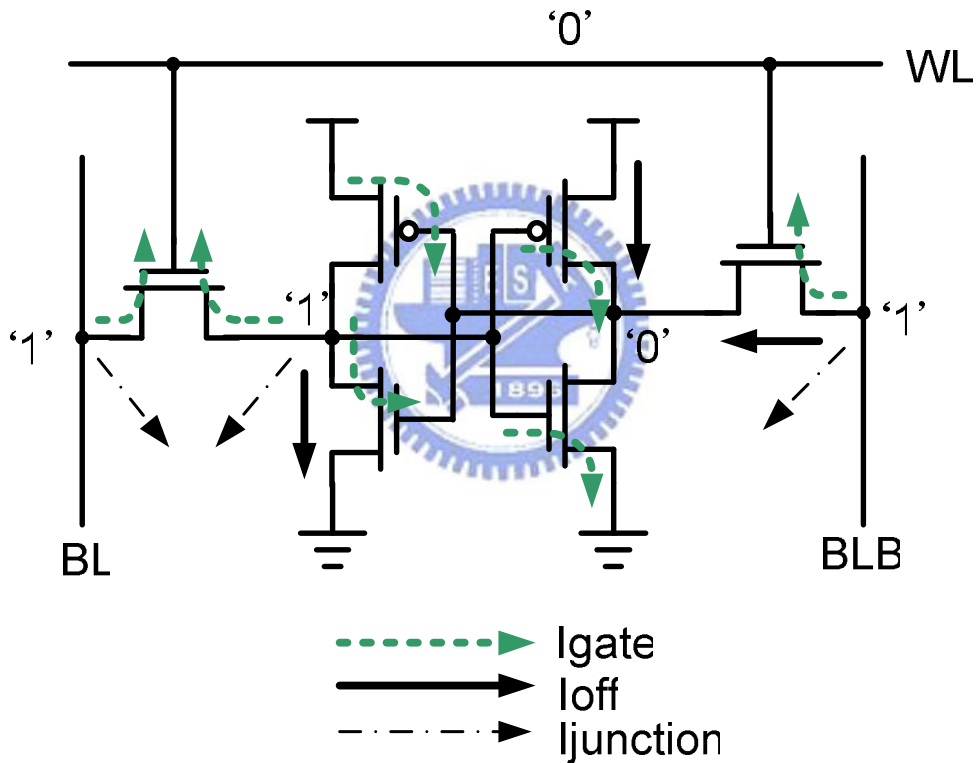


Fig. 2.3. Leakage model in 6T cell.

The subthreshold leakage current is one of the main contributors to the standby leakage. When supply voltage is scaled down, the subthreshold leakage current tends to increase due to decreased threshold voltage of the transistors. A dual- V_{th} technique is generally used to reduce the subthreshold leakage current. Using

low- V_{th} transistors in the critical to improve speed. On the other hand, in order to save power, high- V_{th} transistors is implemented in the noncritical path. The dual- V_{th} technique is easier than the multi-gate-oxide technique, so that the subthreshold leakage current can be reduced by optimizing V_{th} .

When the main source of junction leakage is contributed by trap-assisted leakage and direct band-to-band tunneling, the dopant species and doping profiles are carefully optimized to reduce the defects generated during the implantation process and to achieve a more graded source-drain junction doping profile to minimize the junction tunneling current. The process optimization is able to lower the cell leakage by two-three orders of magnitude in comparison to cell leakage from high-performance CMOS process.

2.2 Low power register file design

In order to reduce the standby leakage power of a register file, many techniques are proposed to act on internal nodes of the cell, as shown in Fig. 2.4.

In [2.14] and [2.15], it is shown how reverse body biasing of a memory cells is dynamically varied. The off drain-to-source current is exponentially dependent on V_{th} . It is applied by raising the node n-well (NW) and lowering the node p-well (PW) during the sleep mode.

This technique can be exploited only in double well technologies which allow the bulk of the nMOS and pMOS devices to be independently biased.

The need to drive the parasitic capacitances of the substrates leads to access time and dynamic energy consumption higher than the conventional SRAM structure. The leakage current can be reduced

by acting on the nodes BL and BLB or on the node WL visible in Fig. 2.4. The approach described in [2.17] reduces the leakage current by leaving BL and BLB floating during the idle time. In [2.18], it applies a negative voltage to the node WL to save leakage power. This technique does not impact on the SER, but the modified word line driver causes energy overhead for generating a negative voltage during the standby mode.

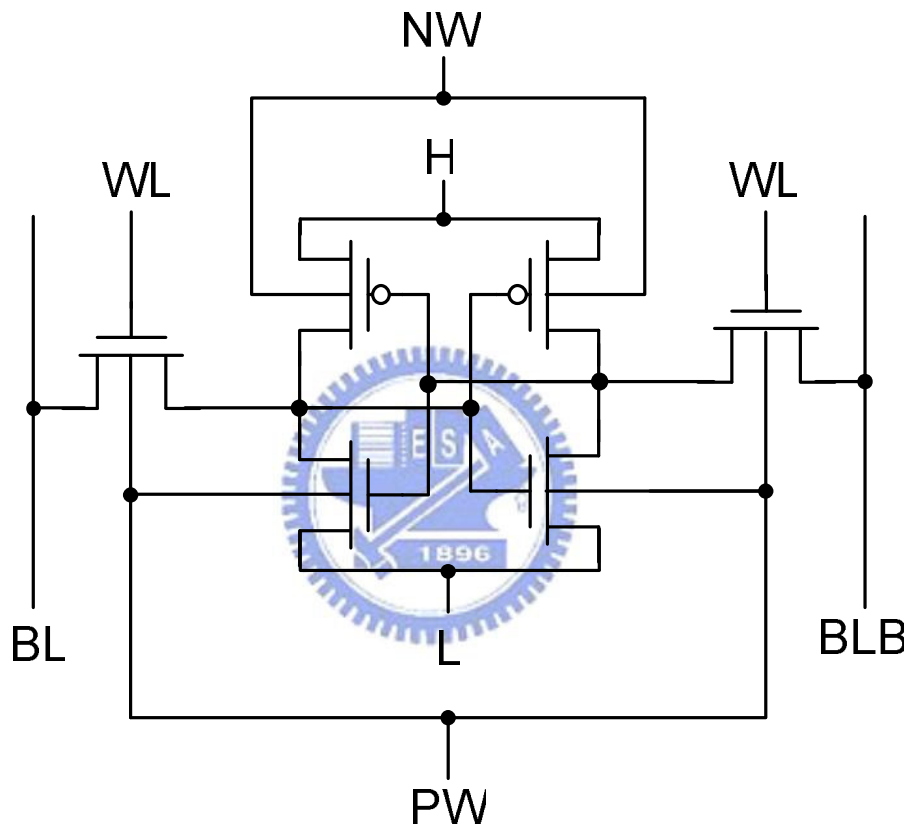


Fig. 2.4. The possible nodes used to decrease the leakage current.

In [2.19], an electric-field-relaxation (EFR) scheme is presented in which both the word line and the bit lines voltages are properly set during the idle time. In order to do this, a dc level converter is required. The application of this approach is not straightforward. In fact, designers must take into account that the bit lines are

shared by the SRAM cells of an entire column and that these cells belong to different rows.

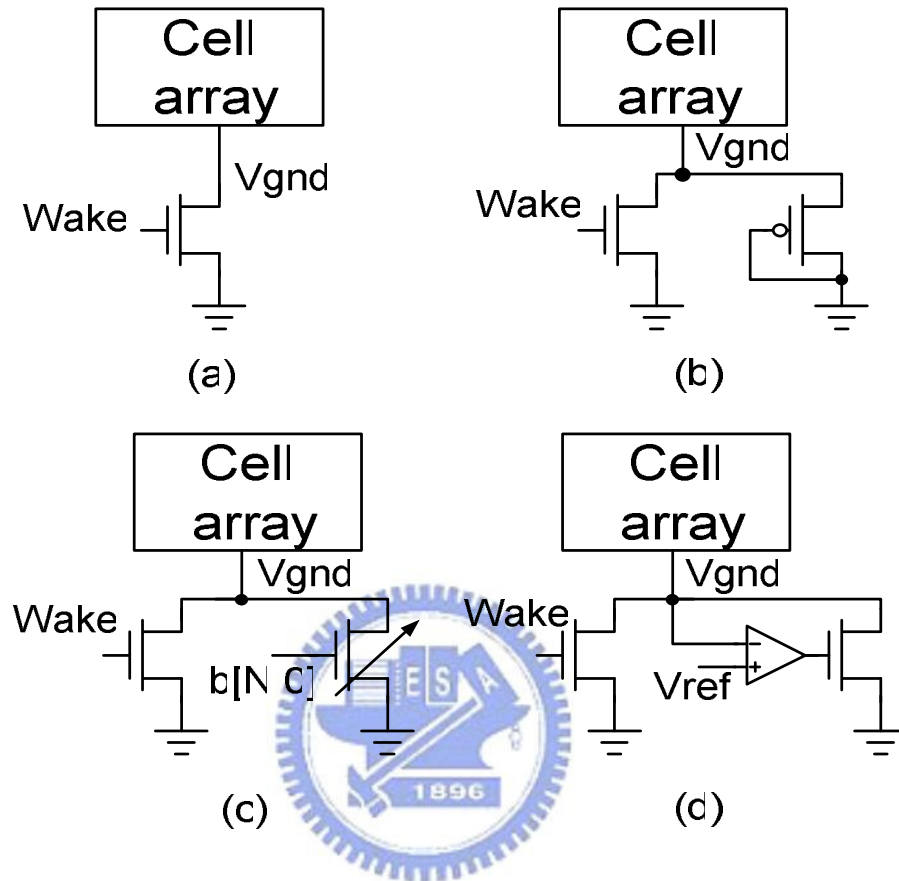


Fig. 2.5. Power-gating schemes (a) Sleep-transistor (b) Diode-connected PMOS bias transistor. (c) Programmable bias transistors. (d) Op-Amp-based-feedback control.

[2.7], [2.8], [2.9] act on the L node. It is connected to the ground usually and this assures stored data stability and full rail. If the voltage of L increases during the sleep mode, the signal rail $V_H - V_L$ and the subthreshold leakage current are reduced. In order to do this, a power-gating transistor is inserted in the pull down path. During idle or standby state, this transistor is off and the leakage mechanisms inside the cell are responsible for charging L.

When an array is on standby mode, the cell rail-to-rail voltage must be kept above the minimum retention voltage. Several techniques are proposed to achieve this. One approach is to use the “sleep” transistor to modulate the VGND [2.19], as shown in Fig. 2.5(a). The challenge is that the sleep transistor needs to be sized properly to meet the wake-up timing requirement and to maintain a low enough IR droop in the current path during an active mode.

A diode-connected PMOS bias transistor is proposed [2.20], as shown in Fig. 2.5(b). The PMOS clamp the VGND to one threshold voltage above the GND. It prevents the VGND from going too high to corrupt the data in the array. There are two shortcomings in this technique. First, leakage reduction will be suboptimal at high VCC applications when the VGND-VCC tracking is not one-to-one. Second, the V_t variation will impact the accuracy of the VGND control.

Programmable bias transistors, shown in Fig. 2.5(c), can overcome these problems [2.21]. The scheme has two benefits. First, the VGND voltage can be optimized based on the actual silicon results to achieve maximum leakage reduction. Second, different bias settings can be dynamically chosen at different supply voltages.

A control scheme with an active feedback based on Op Amp is also proposed [2.22], as shown in Fig. 2.5(d). The major drawback of this scheme is the dc consumed by the Op Amp that has to be replicated along each data bank to provide the needed granularity.

The method used in [2.10]-[2.13] dynamically vary the voltage of the node H. The subthreshold current is reduced by reducing the signal rail. However, the reduced signal rail leads to lower noise margins and a higher SER. Extra peripheral circuitry, such as a dc level controller or a dc-dc converter are required and additional time and dynamic energy are introduced with respect to the

conventional register file cell to enter and exit the sleep mode.

Voltage scaling is the most effective design knob in power management. However, with the memory-cell scaling, it has become difficult to maintain stability margin and write margin.

The register file array can be the voltage scaling limiter for low-power operation. By introducing different power supplies into the register arrays, the voltage-scaling difficulty for the register file cell can be alleviated.

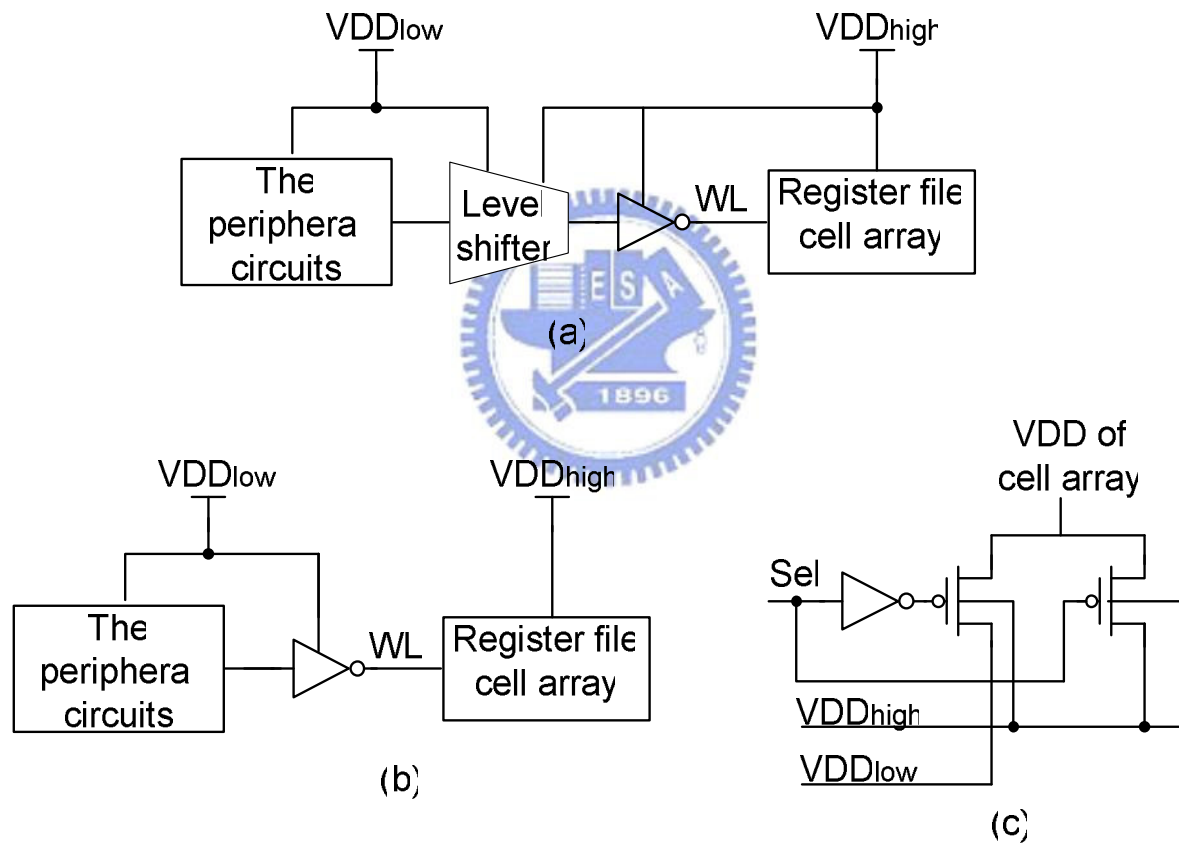


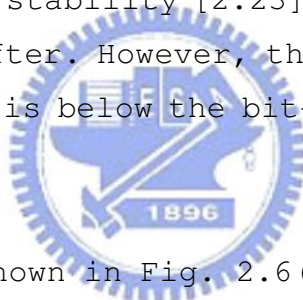
Fig. 2.6. Multi-VCC designs. (a) Register file on a higher supply voltage. (b) Register file on a higher supply voltage than the wordline driver (c) Dynamic multi-VCC.

To improve the write margin, a lower voltage level can be applied to the bit cell than the wordline voltage, which allows the bit-cell

flip more easily during the write operation. There are several different implementation schemes that have been published so far based on this concept. Here, a brief description on each scheme along with the tradeoffs is given.

A simple static multi-VCC design was proposed [2.26], which is shown in Fig. 2.6(a), where both SRAM-VCC and wordline driver are on a separate and higher supply than the rest of the system. Both register file stability and writ ability can be kept. This implementation requires voltage-level shifters at the boundary between the array and the rest of the chip.

The design, shown in Fig. 2.6(b), is to put the register file bit cells on a higher supply voltage than the wordline driver voltage in order to improve read stability [2.23]. This implementation can eliminate the level shifter. However, the write margin is reduced as the wordline voltage is below the bit-cell power supply during the write operation.



The other approach, shown in Fig. 2.6(c), uses a mux to switch the bit-cell power supply based on the operating condition. In the case of read stability improvement, the unselected columns during write and all the columns during read are switched to the higher supply voltage. The selected column during write is switched to the lower voltage; hence, the write margin is maintained. The wordline driver is always on the lower supply. The sleep transistor can also be used to lower the effective power supply to the bit cell, achieving a lower leakage.

2.3 Conventional Register file Architecture

One of the different between register file and SRAM is that the port number of register file is much more than that of SRAM. In order to support multi-accesses, each storage cell has the fully

ports in conventional scheme. However, this will cause a lot of area, power, and access time overhead. The size of area is positive to the port number of bit cell. As the port number of a register file is increased, the peripheral circuitry, such as write drivers and precharge circuits would increase at the same time and cause a lot of power consumption. Access time also become worse when larger area make longer wire delay.

2.3.1 The Read Port Design

It is possible that multiple read/write operation access the same register file cell. In the traditional SRAM, read port is composed of pass transistor. During the read operation, large sink current will pass through n-transistor in the SRAM cell and degrades the static noise margin. Therefore, isolation between sink current and the register file cell is necessary in multiple read port register files as shown in Fig. 2.7 and Fig. 2.8.

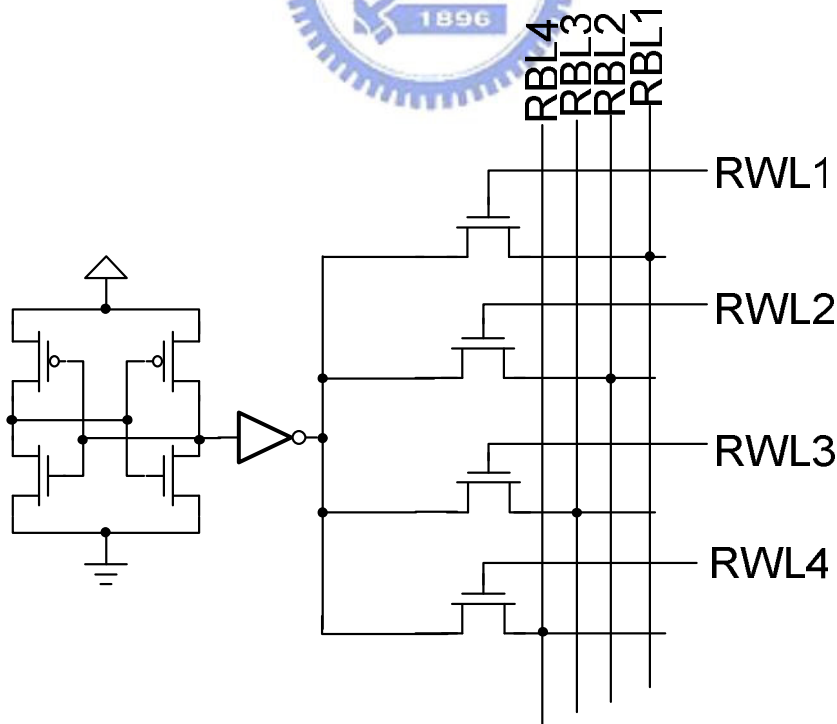


Fig. 2.7 Using inverter to isolate memory cells and pass transistors

Another feature of conventional register file is its heavy metal routing around register file cell. To alleviate the routing efforts, single ended access port is desired throughout the register file design.

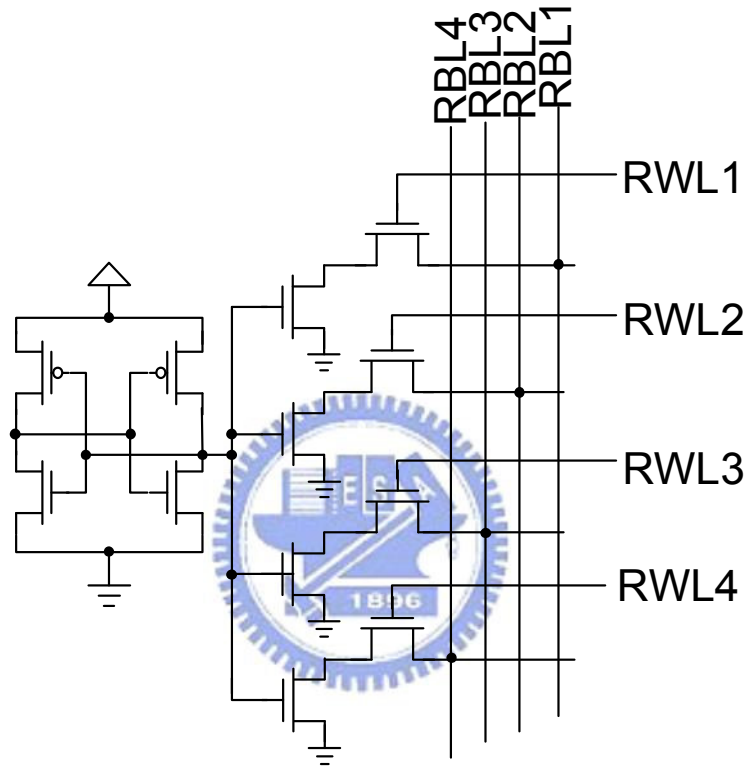


Fig. 2.8 Using read buffers to sink read current

2.3.2 Low Power Write Port Design

The basic component of a register file cell is a latch which is connected by two inverters and can hold the data. Because of such feedback mechanism, writing '1' to a register file cell storing '0' or writing '0' to a memory cell storing '1' (Fig. 2.9) causes an instantaneous short current from V_{dd} to ground. In order to minimize or suppress such power consumption, strong inverter is

proposed to reduce energy consumption while maintaining performance.

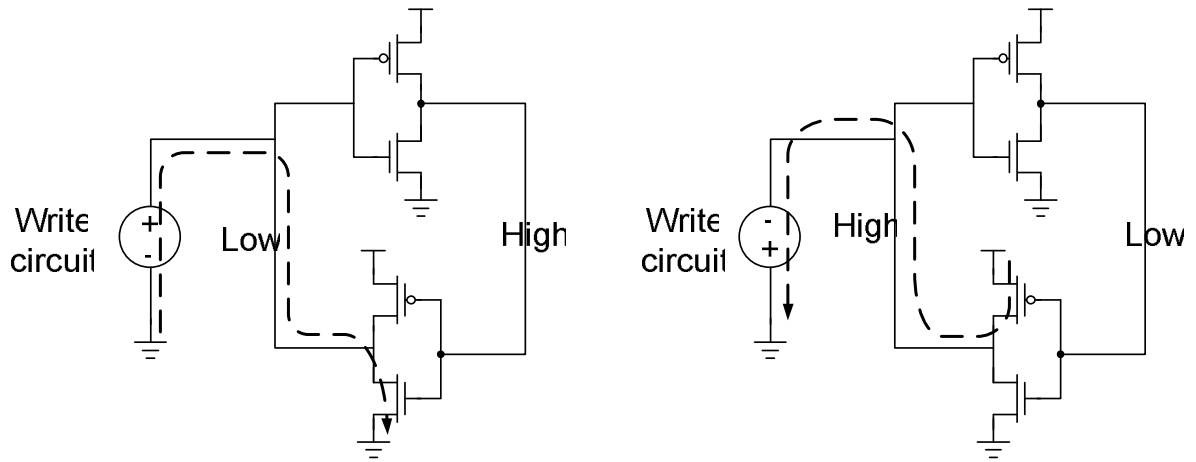


Fig. 2.9 Instantaneous short current in write operation

Strong inverter technique speeds up the transition process which will definitely reduce the duration of short current. It is good to use asymmetric inverters in single ended write scheme because multiple read ports requires sufficient driving ability to drive all the gate capacitances and the weak inverter. The weak inverter acts as feedback path to hold data, as shown in Fig. 2.10. Due to single-ended access, transmission gates are used as write port to guarantee the value written into the register file cell.

Due to the fast transition of the inverters, short current is suppressed significantly while maintaining the speed. Different port numbers that cause different loading at the output of strong inverter will have a different optimum size of the strong inverter.

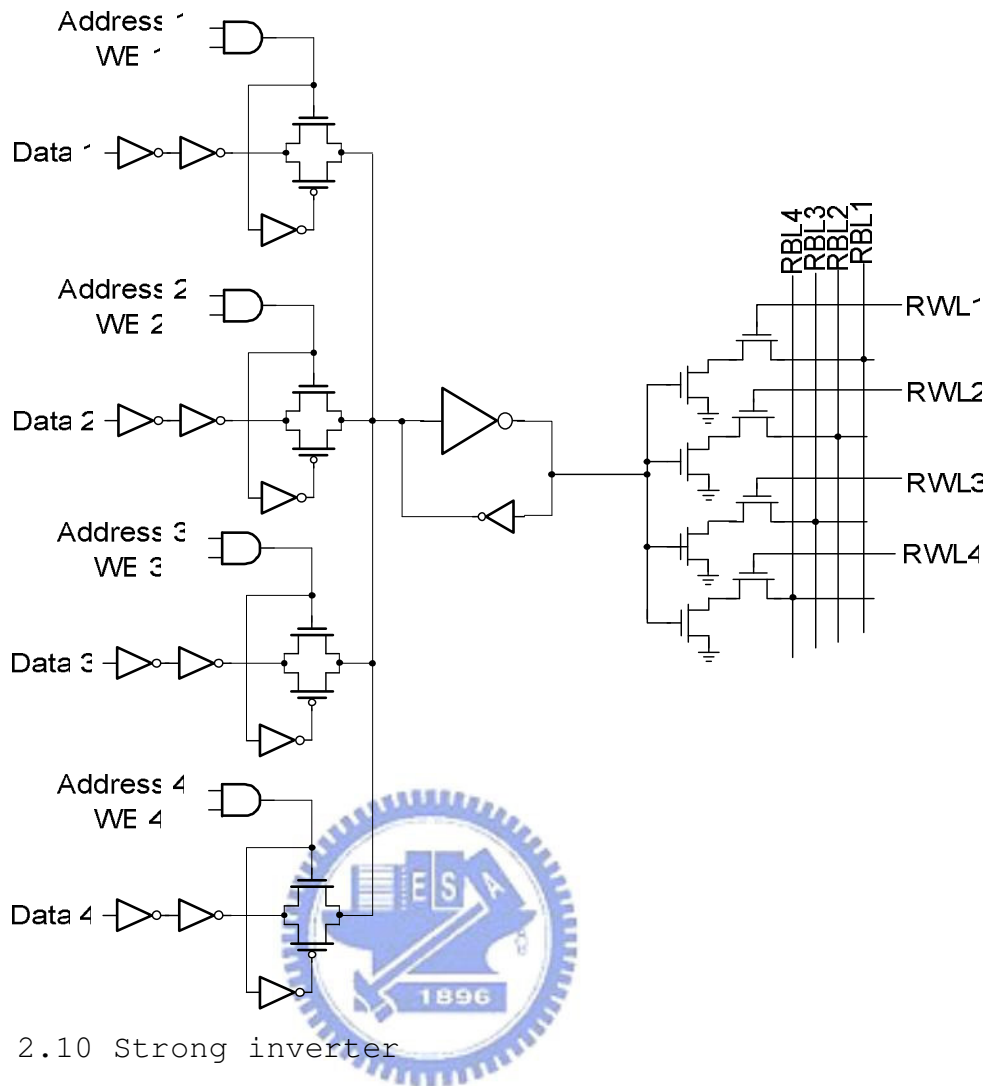


Fig. 2.10 Strong inverter

2.4 Banked Register file Architecture

Multibanked register files realize multi-port access by using less port memory cells instead of multi-port memory cells to reduce the quantity of wiring. Therefore the multi-bank register file can realize higher speed, smaller size, and lower power consumption than conventional multi-port register files. Fig. 2.11 show a example of an 8-read, 4-write port register file implemented using four one-write, two-read port banks.

However, when this scheme is used in a processor, it causes several

problems. If the processor requires a read or to write operation to a multibanked register file, the access to this register file is restricted to one access per bank. So the processor is not able to access several data in the same bank at once.

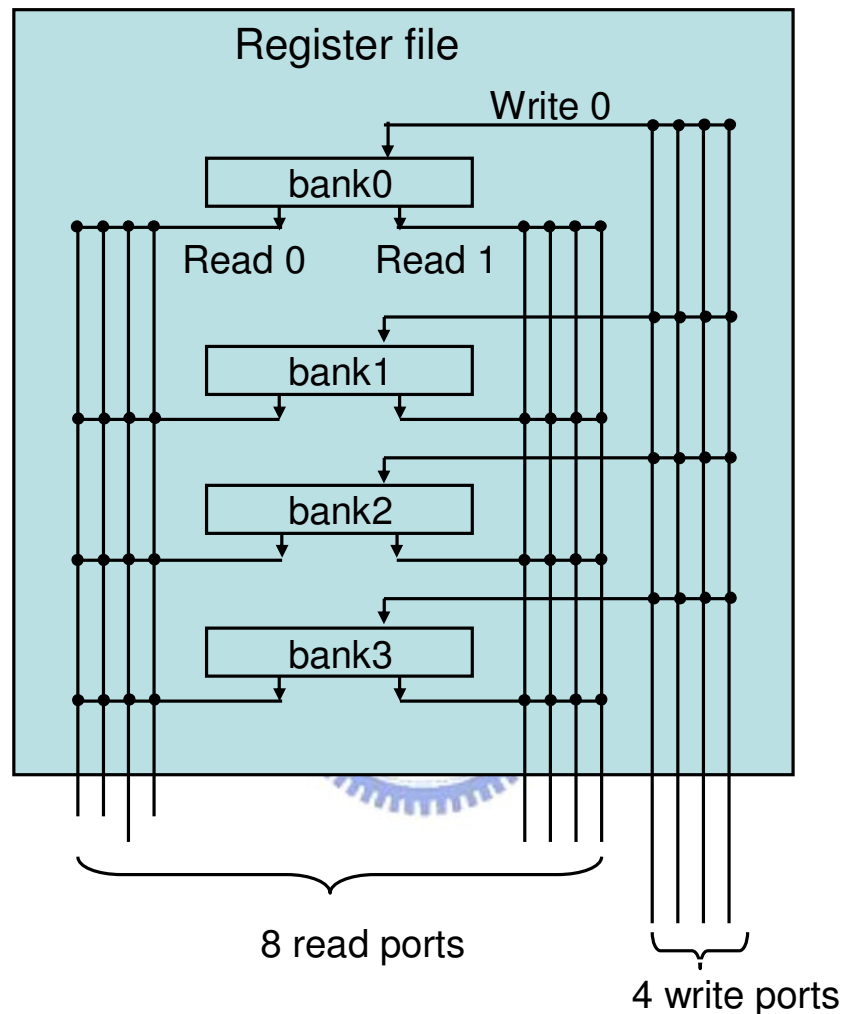


Fig. 2.11. An 8-read, 4-write port register file implemented using four two-read, one-write port banks.

2.4.1 Register bank structure

The multibanking approach adopted in [2.27], [2.28] constructs a register file from multiple interleaved register banks. The

challenge is managing the complexity and added latency of the control logic needed to handle read and write bank conflicts and the mapping of register ports to functional units. A banking scheme that uses the bypass network to reduce unnecessary read port contention and usage is described in [2.28], but no description of the bypass check or read conflict resolution logic is given.

Write conflicts are handled by delaying physical register allocation until write back, at which point registers are mapped to non-conflicting banks. The primary motivation for this delayed allocation was to limit the size of the physical register file, but this can lead to a deadlock situation requiring a complex recovery scheme.

The scheme presented in [2.27] handles read bank conflicts by only scheduling groups of instructions without conflicts. This reduces the IPC penalty, but adds significant logic into the critical wakeup-select loop. A design with single-ported read banks is evaluated; however, this requires complex issue logic and functional unit datapaths to allow instructions where both operands originate from the same bank to be issued across two successive bank read cycles.

Multiplexing circuits dominate the area of few-ported multibanked designs. Moving from a single read port to split dual read ports per bank has minimal area impact. In [2.27], write port conflicts are handled by buffering conflicting writes, which increases the size of the bypass network. Functional unit pipelines must also be stalled when conflicting writes queue up.

2.4.2 Methods to decrease access conflicts

In [2.4], a banked multiported register file design is presented and analyzed together with a control scheme. This scheme is suitable

for a deeply pipelined dynamically scheduled processor. It does not place any register bank arbitration in the critical wakeup-select loop, but instead speculatively issues potentially conflicting instructions. If any conflicts are found after issue, a pipelined recovery scheme quickly repairs the issue window and reissues conflicting instructions.

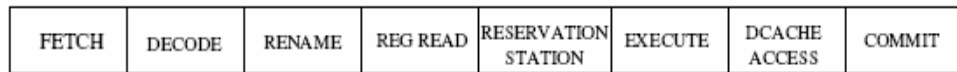
In contrast to previous work, all conflicts are detected and resolved in one pipeline stage. Therefore, no write buffering or pipeline stalls are required. The main drawback of the scheme is that both bank conflicts and the extra pipeline stage used for port arbitration can impact processor performance. Bank conflicts add penalty cycles to repair the pipeline and delay the issuing of dependent instructions, while the additional pipeline stage causes an increase in branch misprediction latency.

The conventional structure of the pipeline is the case of Fig. 2.12(a). After instructions are decoded, renamed and the register accesses are carried out, the instructions are registered in a buffer called reservation station. Register accesses are thus done per instruction. In this case, if register accesses are blocked by conflicts, the processor has to stall decode and rename for the following instructions until the conflict is resolved. So queue access per instruction does not work effectively.

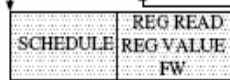
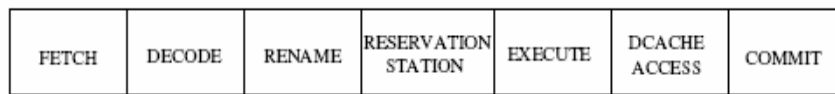
To remove this problem, [2.30] propose a method for register access per operand. It changes the structure as shown in Fig. 2.12(b). This new structure allocates the scheduling stage of register access to the same stage as the reservation station. Instructions are decoded and decomposed into accesses per operand, e.g. as two register accesses and one operation for a two-operand instruction.

Operations and tags of the source registers are registered in the reservation station, which receives the value for each source

register after several clock cycles. Since execution of the pipeline and register accesses are operated independently, register accesses per operand and decoding and renaming of instructions occur without stalling of the processor.



(a) Conventional structure of pipeline



(b) Structure of pipeline with out-of-order of register access

Fig. 2.12. Structure of the processor pipeline



Reduction of the number of register accesses is also a good method to avoid access conflict [2.31]. When a register has true dependency with an instruction, the processor uses the result of an instruction as the value of an operand register of the following instruction. Therefore, this value doesn't need to be buffered in the register access queue, so that forwarding is able to reduce the number of register accesses.

When multiple reading of the same register at the same time causes an access conflict, treating these accesses as one access to reduce the number of register accesses can avoid this conflict, as shown in Fig. 2.13.

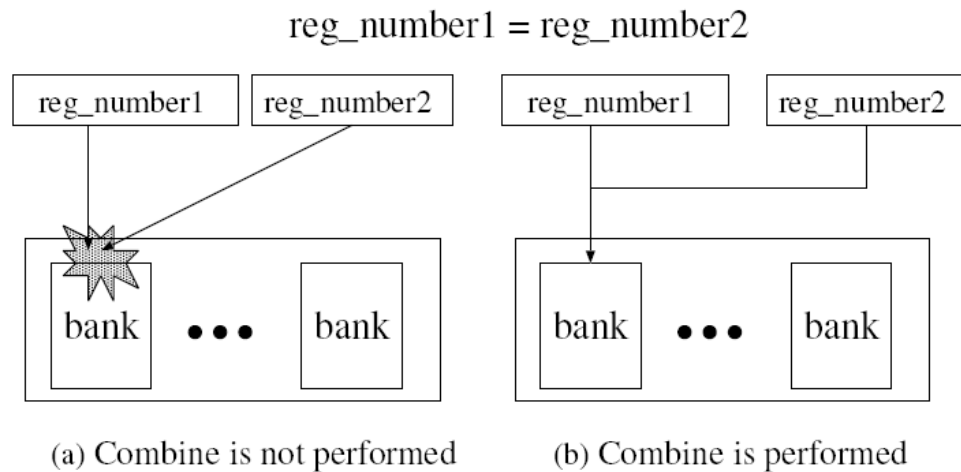


Fig. 2.13. Combine access to the same registers

In conventional register renaming schemes, both register allocation and releasing are conservatively done, the former at the rename stage, before registers are loaded with values, and the latter at the commit stage of the instruction redefining the same register, once registers are not used anymore. [2.32] introduces a renaming scheme that allocates registers later and releases them earlier than conventional schemes.

Specifically, physical registers are allocated at the end of the execution stage and released as soon as the processor realizes that there will be no further use of them. This approach enhances register utilization.

2.5 Other kind of multibanked Architecture

There are several different banked scheme adopted in order to save more power and decrease access time. [2.34] propose the scheme that the register bit-widths of different banks in the register files are different. In [2.33], different banks have different register

numbers and port numbers.

2.5.1 Customization of Register File Banking Architecture

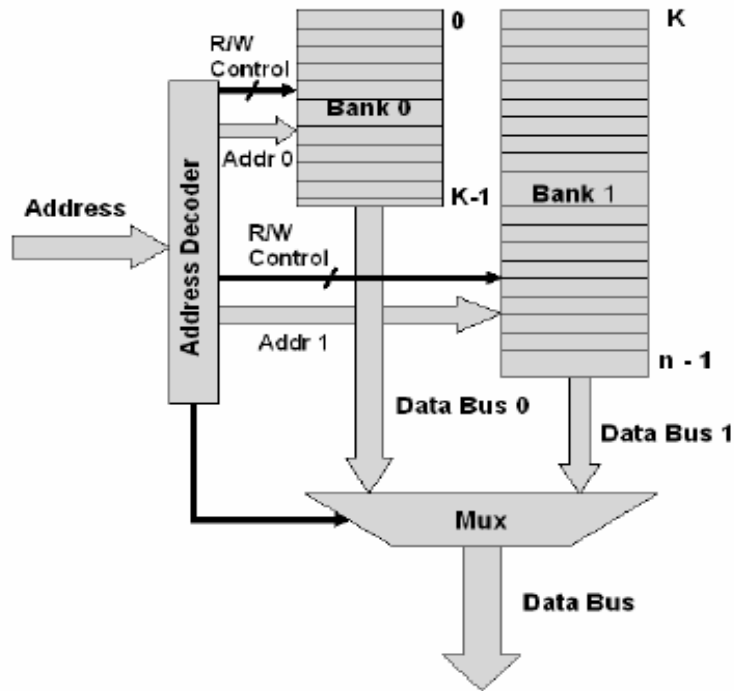


Fig. 2.14. Heterogeneous Register File Banking

[2.33] reduces register file power consumption by allocating variables in frequently accessed basic blocks to separate appropriately sized register file bank of active registers.

Fig.2.14 shows the heterogeneous banking of register file with n registers in a single issue RISC processor. Only the read path for one port is shown in the figure for clarity. It consists of two banks, one bank with a small number of registers (0 to $k-1$) and a second bank with a larger number of registers (k to $n-1$), with $k < n/2$.

If the placement of registers is such that most of the accesses occur to the smaller bank, there will be a significant reduction in the overall power dissipation as the smaller bank has a relatively smaller bit-line switching capacitance. The asymmetric banking structure makes the address decoder and output selection logic more complex, which introduces overheads in area, delay, and power.

2.5.2 Asymmetrically Banked Value-Aware Register Files

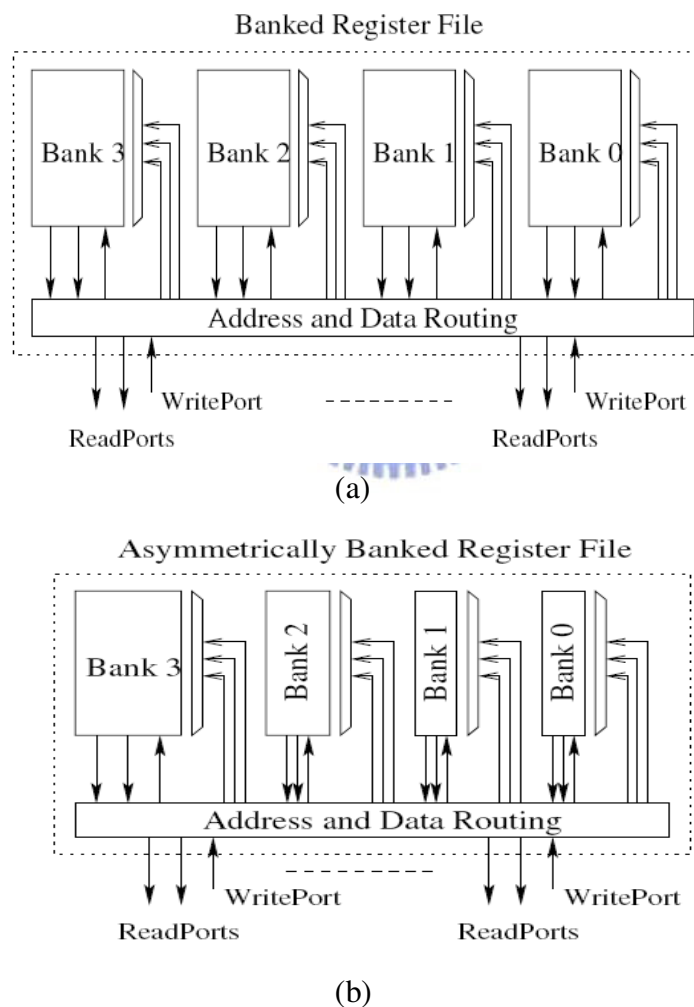


Fig. 2.15. A microarchitecture-level comparison among (a) a conventional banked register file and (b) the AB-VARF register file.

[2.34] propose a new microarchitecture, the asymmetrically-banked value-aware register file (AB-VARF), as shown in Fig. 2.15(b), to exploit the prevailing narrow width register values for low-latency and power-efficient register file designs. The register bit-widths of different banks in the AB-VARF register files are specifically customized to capture different narrow-width values. Augmented with a value width predictor, the register renaming logic is slightly tuned to rename predicted narrow-width registers to the corresponding narrow-width banks.

2.6 Conclusion

The design of register file is known to scale poorly with increasing numbers of ports and registers. In order to implement a large and fast multiported register file, banked architectures have explored alternative designs.

Even though that banked register file exhibits a small performance penalty, the reductions in register file delay and power are respectable. Consequently, banked register files are a natural solution to the increasing register file demands of SMT processors.

Chapter 3

Low power Multithreaded Register File Design

3.1 Multithreaded Banking Architecture

In this chapter, the multibanked register structure is described first. The timing access scheme is then characterized in detail before the discussion of thread switching.

3.1.1 Register Bank structure

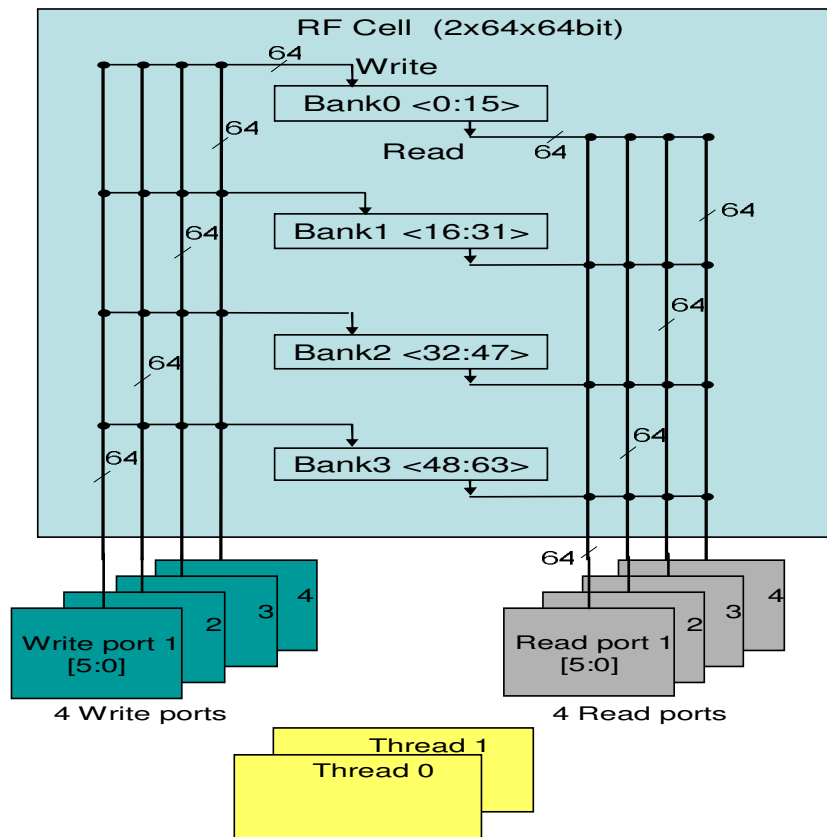


Fig. 3.1 The proposed dual-thread 4W/4R 64 x 64 bit register file architecture.

The proposed dual-thread register file has 4 Write ports and 4 Reads ports, and there are 64x64 bits for a thread. The proposed

design is also divided into four interleaved banks, and each bank has one Read port and one Write port, as shown in Fig. 3.1.

Compared to a conventional register file scheme, each storage cell has fewer ports and the storage cell size become smaller appreciably. A single local port must connect to all global ports, so local-global crossbar complexity is required. Bank conflicts would happen when too many global ports attempt to read or write the same bank. The register file can operate under voltage range from 1.0v to 0.5v.

3.1.2 Timing sharing access scheme

The area overhead of a register file can be reduced when a bank has only one local Read port and one Write port; however, access conflicts would happen if more than one word of a bank are selected in a clock. It leads to performance degradation of a conventional banking scheme. Thus, timing sharing access scheme is proposed to ease access conflict overhead in our design.

In the proposed timing sharing access scheme, local Read and Write ports can be accessed twice in a clock. In other words, a clock is divided into two time slots, and an access operation can be finished in one slot. Fig. 3.2 makes an example to explain this concept.

In cycle 1, the processor writes two data, D1 and D2, into the same bank. Thus, local Write bitline, WBL, are accessed twice. D1 is written into the bank in the first slot, and D2 is written into the bank in the second slot. The proposed multithreaded register file can do at most 4 Read operations and 4 Write operations in one cycle. Nevertheless, access conflicts still happen when more than two word of a bank are accessed during Read or Write operations.

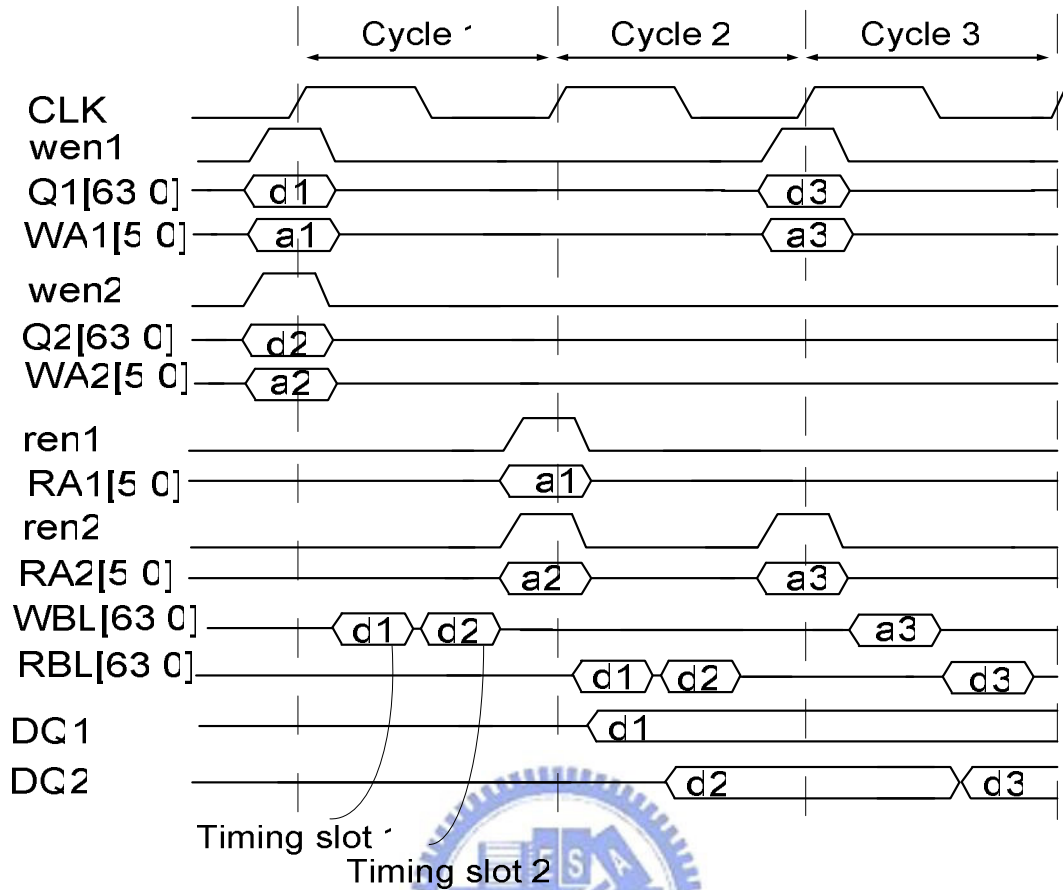


Fig. 3.2 Waveforms illustrating concept of timing sharing access scheme

The register file should arbitrate which two operations should be executed first when access conflicts happen, and all ports of this register file are defined as different priority levels. The priority of Read (Write) port1 is higher than Read (Write) port3 respectively, and Read (Write) port2 is higher than Read (Write) port4 as well. Port1 and port 3 work in the first time slot, and port2 and port4 work in the second time slot, as shown in Fig. 3.2. For example, when four Read ports want to access the same bank, port1 and port2 can operate successfully while both port3 and port4 would receive Read fail signals.

3.1.3 Thread switching

Different thread needs its own storage cells, and there are many methods to switch between different threads. The two threads of Montecito microprocessor is switching through a multiplexer which is connected to the storage node of memory cell [3.1], as shown in Fig. 3.3. This method would enlarge the capacitance of the storage node then make the noise margin and speed worse. If the thread number increases, Montecito microprocessor needs more levels of multiplexer and makes the problem worse.

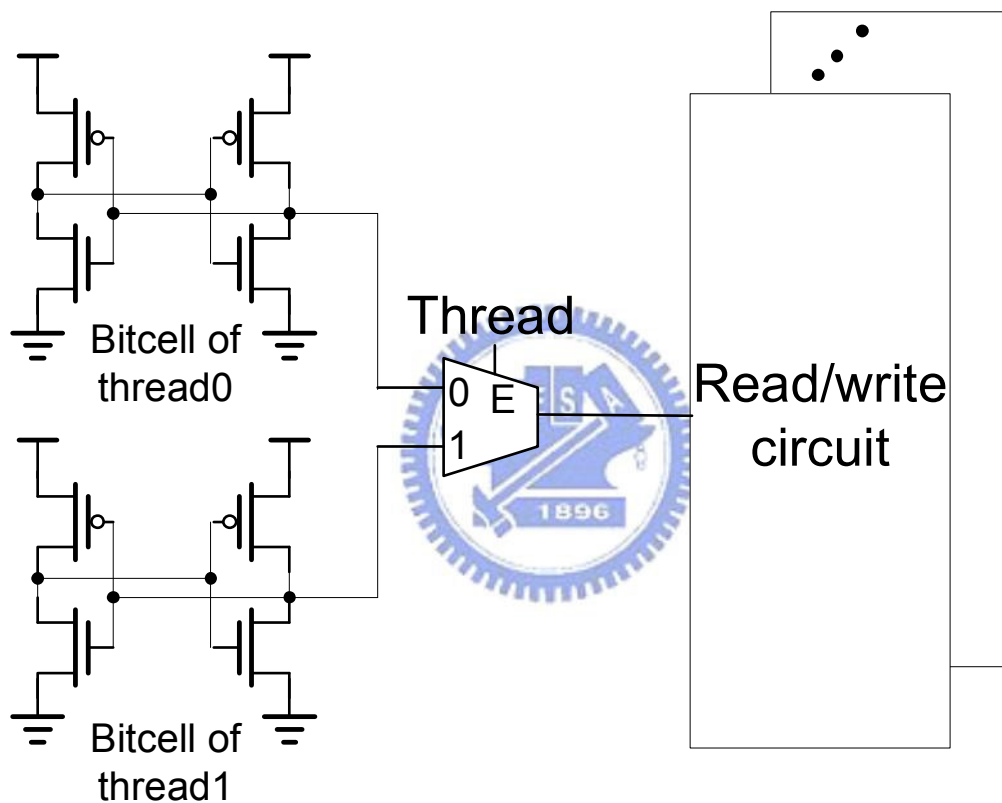


Fig. 3.3 Thread switching scheme of Montecito microprocessor

Additional address bits as thread ID are used here to switch threads. If the amount of thread is n , we need additional $\log_2 n$ address bit for each decoder. This method would increase some area penalty, but impact performance slightly.

Each decoder of this proposed register file has 7 bit. Bit[6:1] chose the register number when bit[0] is as thread number. LSB is

used as thread number instead of MSB, so the thread0's registers and thread1's registers are interleaved with each other, as shown in Fig. 3.4. The way scatter the location of registers from the same thread and can decrease the bank conflict.

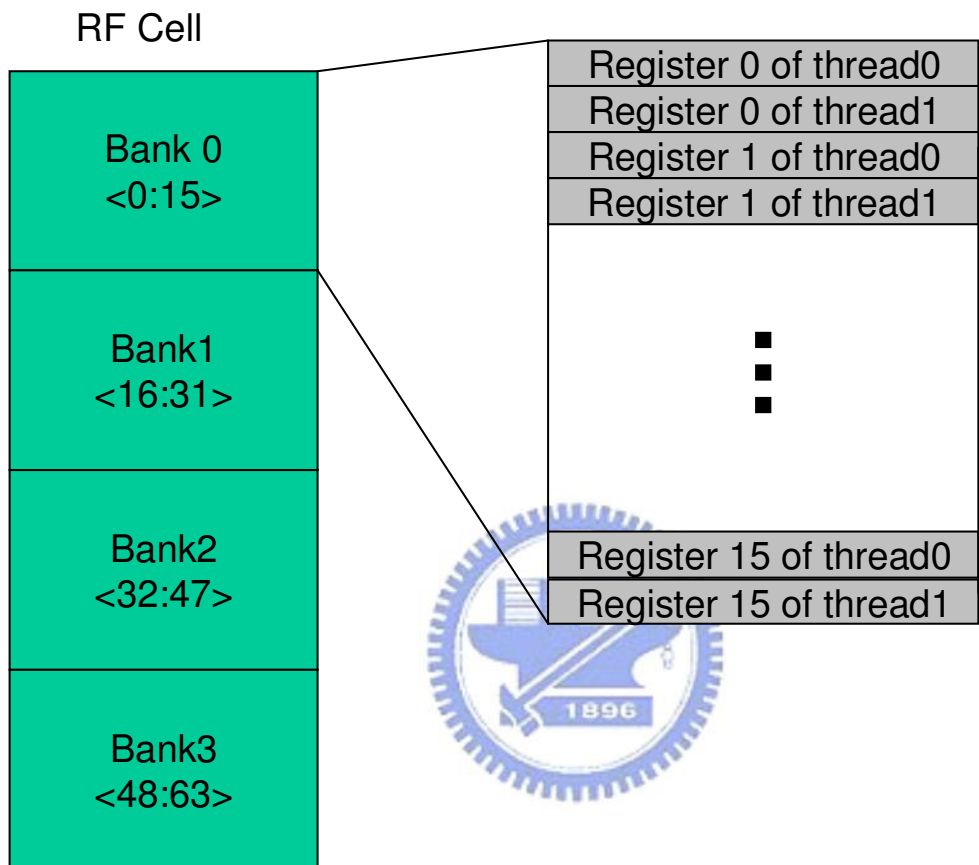


Fig.3.4 The location of registers

3.2 Register file cell

3.2.1 Read Stability and Write-Ability

The read stability and write ability of register file cells determines its process tolerances. Many new SRAM cell circuit designs have been developed to maximize the cell stability for future technology nodes.

As dimensions scale down to nanometer regime, the variations in CMOS transistor parameters, e.g., the threshold voltage, increase steadily due to random dopant density fluctuations in channel, source and drain. Therefore, two closely placed, supposedly identical transistors, have important differences in their electrical parameters as and make the design of the SRAM less predictable and controllable. Moreover, the stability of the SRAM cell is seriously affected by the increase in variability and by the decrease in supply voltage.

Data retention of the register file cell is an important functional constraint in advanced technology nodes. The cell becomes less stable when supply voltage decrease, leakage currents become larger or variability increase. The stability is always defined by the SNM as the maximum value of DC noise voltage that can be tolerated by the SRAM cell without changing the stored bit.

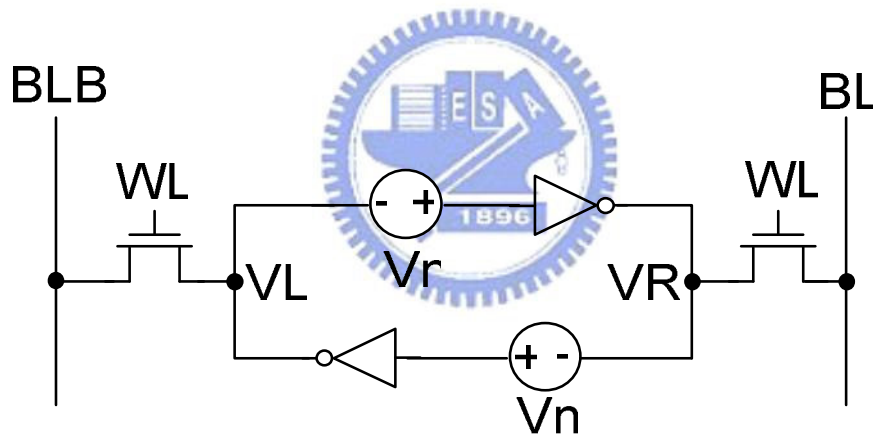


Fig. 3.5. SNM definition. The two DC noise voltage sources V_n are placed in series with the cross-coupled inverters.

The circuit definition for the SNM of conventional 6T cell is shown in Fig. 3.5. The two DC noise voltage sources are placed in series with the cross-coupled inverters and with worst-case polarity at the internal nodes of the cell. Locating the smallest square between the two largest ones delimited by the eyes of the butterfly curve determines graphically the SNM (Fig. 3.6). When the DC noise voltage is equal to the SNM, the VTCs move horizontally and/or vertically until the stable point A and the meta-stable point B coincide.

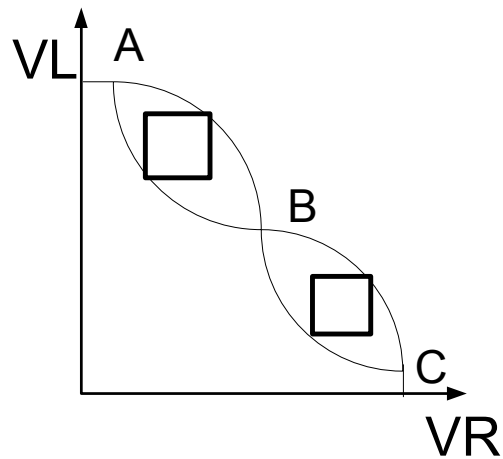


Fig. 3.6 Static Noise Margin

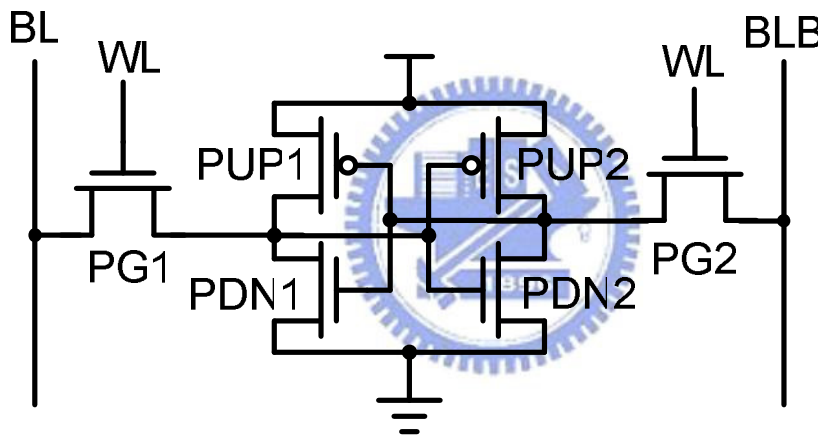


Fig. 3.7 The 6T cell

The cell is most vulnerable to noise during a read access since the "0" internal storage node rises to a voltage higher than ground. Due to this voltage division on, the SNM is primarily determined by the ratio of the pull down (PDN) to pass gate (PG) transistor, known as the cell beta ratio. In an ideal case, each of the two cross-coupled inverters in the SRAM cell has an infinite gain. As a result, the butterfly curves delimit a maximal square side of maximum, being an asymptotical limit for the SNM. Therefore, scaling limits the stability of the cell.

In the SRAM cell shown in Fig. 3.7, when the BL signal is set to

0, the NMOS transistor (PG1) is turned ON, which results in a voltage drop in the storage node holding data 1. This is the trigger for write operations to begin. Stable write operations require that the current of PG1 be higher than that of PMOS transistor PUP1. Because of the importance of PG1 in write operations, there should be a correlation between the WM value and the PG1 drivability under all conditions.

Besides the read stability for the register file cell, a reasonable write-trip point is also important to guarantee the write ability of the cell without spending too much energy in pulling down the bit-line voltage to 0 V. The write-trip point defines the maximum voltage on the bit-line, needed to flip the cell content (Fig. 3.8). The write-trip point is mainly determined by the pull-up ratio of the cell while the read stability is determined by the cell ratio of cell.

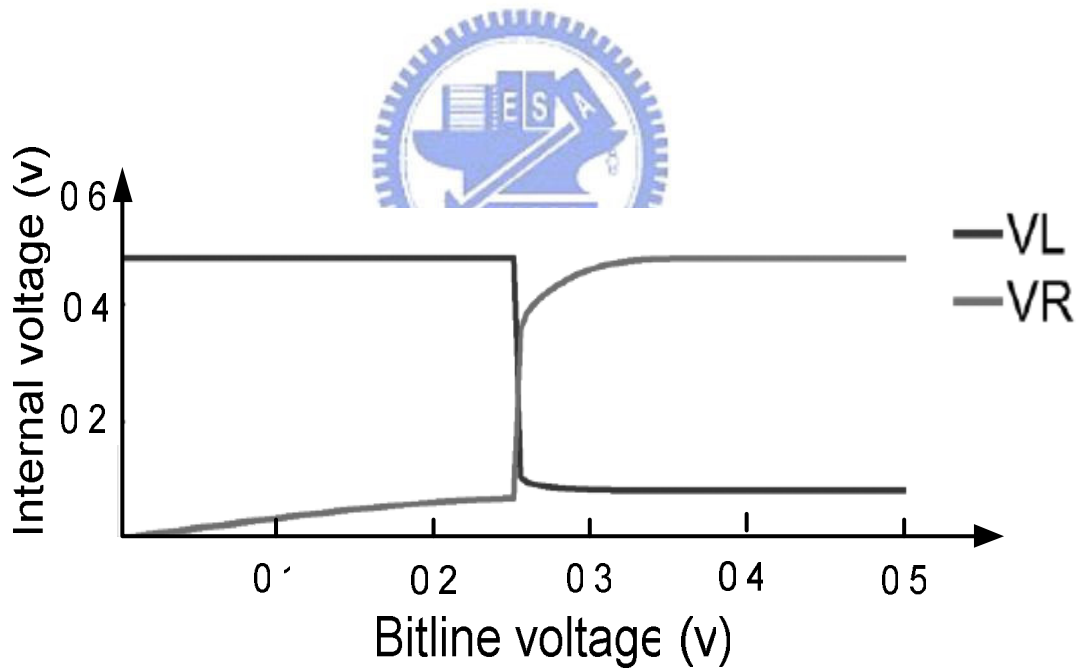


Fig. 3.8 The write margin is defined by the write-trip point.

3.2.2 The 8T cell

In the proposed architecture, an 8T cell, shown in Fig. 3.9, is adopted instead of conventional 6T cell. The conventional 6T cell shown in Fig. 3.7 has switch type read access transistors. The storage data is affected at the read operation. On the other hand, the read port and write port of 8T cell is separated. The read port of 8T cell is connected to storage node through a read buffer. The gate electrode of the read-port-drive-transistor receives the storage-node-voltage directly. Therefore, the read margin of 8T cell is better than that of 6T cell.

Because storage data is not affected at the read operation, the nMOS gate width of latch-inverters is reduced a lot compared with a conventional 6T cell.

Fig. 3.10 shows the Monte Carlo simulation results of 8T's Write margin and Read noise margin at 0.5v. According to the results of Fig. 3.10, 8T cell can work correctly at 0.5v.

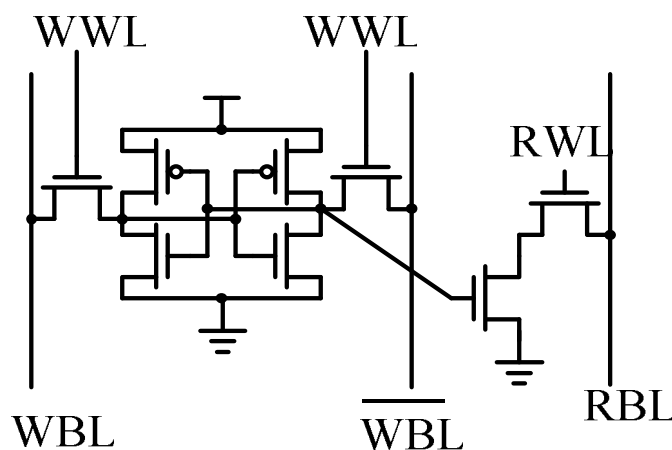


Fig.3.9 8T cell structure

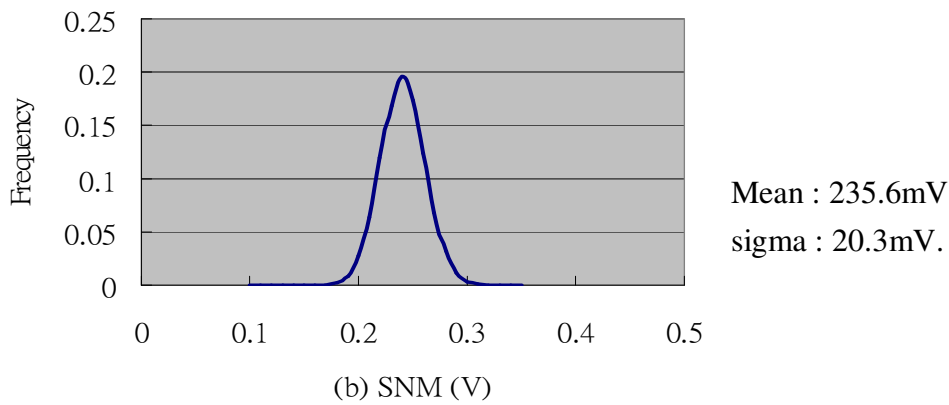
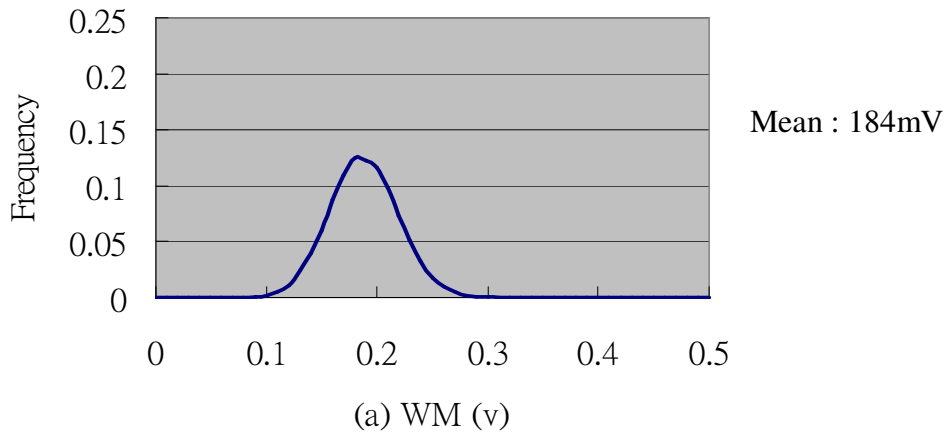


Fig. 3.10 The Monte Carlo simulation result of 8T's noise margin at 0.5v. The sigma of V_t is 30mV (a) Write margin distribution. (b) Read noise margin distribution.

3.2.3 Other register file cells

Fig. 3.11 shows the schematic of the 10 T sub-threshold bitcell [3.2]. M3 and M6 tie to a virtual supply voltage rail. The technique of weakening the cross-coupled inverters by gating their supply voltage or ground node, applied by previous works primarily to improve speed, can dramatically improve write margin. Transistors M7 through M10 implement a buffer used for reading.

One key advantage to separating the read and write wordlines and bitlines is that a memory using this bitcell can have distinct read and write ports. M7-M10 to remove the problem of Read SNM by buffering the stored data during a read access.

M10 is valuable to the bitcell because it reduces leakage current and allows more bitcells to share a bitline. The reduction in subthreshold leakage through M8 reduces the impact of leakage from unaccessed cells and gives the additional advantage of allowing more cells on a bitline during read.

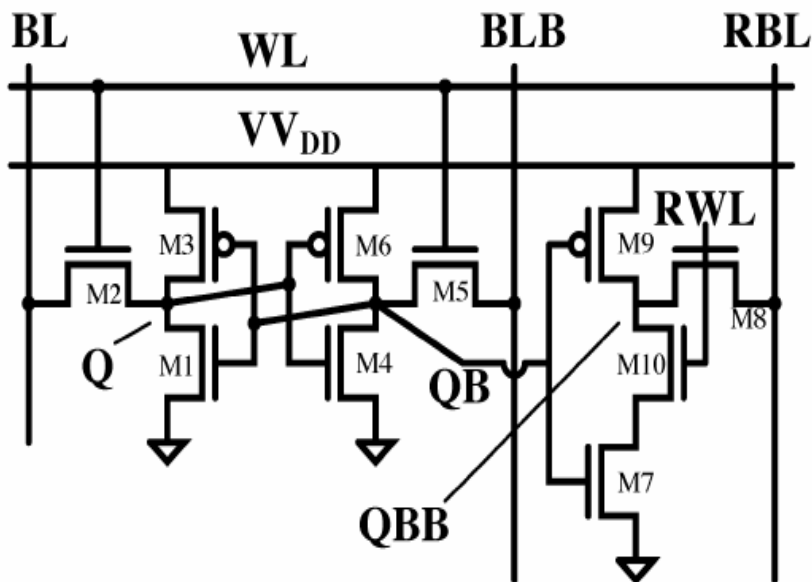


Fig. 3.11 Schematic of the 10 T bitcell.

In [3.3], N3 and N4 are added to keep the read current from affecting the cell value, as shown in Fig 3.12. The latch feedback loop is open during the write operation, and closed when the write is complete. This is accomplished by gating one of the feedback path transistors, Pgate, as shown in Fig. 3.12, which improves the write margin in subthreshold. The input node transitions slowly due to the open FB loop. When the loop is closed by the WWL, the cell value is quickly updated.

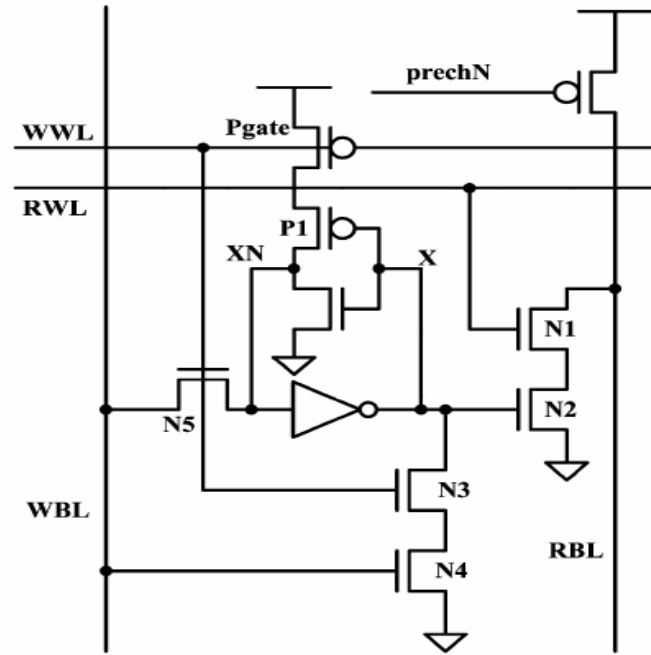
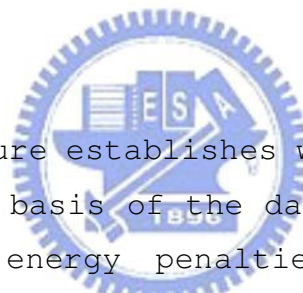


Fig.3.12 RF cell with conventional read circuit. An additional transistor, Pgate, is added to aid write margin in subthreshold.



[3.4] proposed structure establishes when to enter and to exit the sleep mode, on the basis of the data stored in it, without introducing time and energy penalties with respect to the conventional 6T cell. The low V_{th} MOS transistors M7 and M8 (Fig 3.13), gated by the signals Q and Q_b, respectively, are introduced to minimize the leakage power dissipation during the idle mode. When a 1 is written, BL and BL_b are forced to 1 and 0. The signal WL becoming high discharges the node Q_b and turns on the transistor M3. This makes the source terminal of the transistor M7 electrically connected to Q. Thus, M7 is turned off.

During the standby mode, let us suppose that it stores a 0 data. In this situation, the transistors M7 and M8 are turned off and the cell is disconnected from Vdd. The leakage paths through M6 and M8 are charging Q_b, whereas the leakage path through M2 forces Q_b to discharge towards ground. The data retention capability of the cell is assured by making the subthreshold current flowing

through the transistor M8 greater than that flowing through M2. This is obtained by using a low threshold voltage device that leads to a resultant charging effect on the node Q_b.

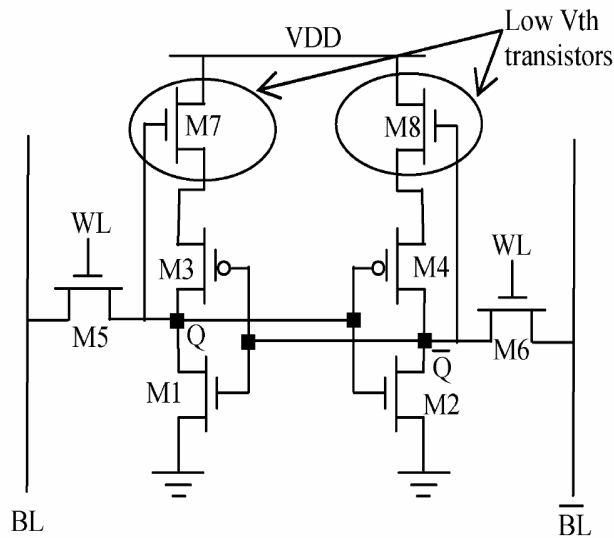


Fig 3.13 The auto-gating low-leakage cell



3.3 Low Power Floating Read Bitline Access Scheme

Precharging bitline would consume a lot of power every cycle. Thus, floating bitline access scheme is proposed. Following starts to explain the operations of floating bitline access scheme.

3.3.1 Floating Read Scheme

Floating read bitline architecture is also shown in Fig. 3.14. The read bitline need not be precharged. If the voltage of bitline is a strong '0' for a read operation, the node 'inx' of SA will be pulled down to '0'. If the voltage of bitline is a floating '0', the SA will charge the bitline slowly instead of causing a read fault.

Therefore, the enable signal of SA must rise slowly in order to prevent the situation that node 'inx' of SA would be discharged by floating bitline. The signal 'reset' and 'enable' of sense amplifier are come from timing control circuit, which is described in chapter 4.

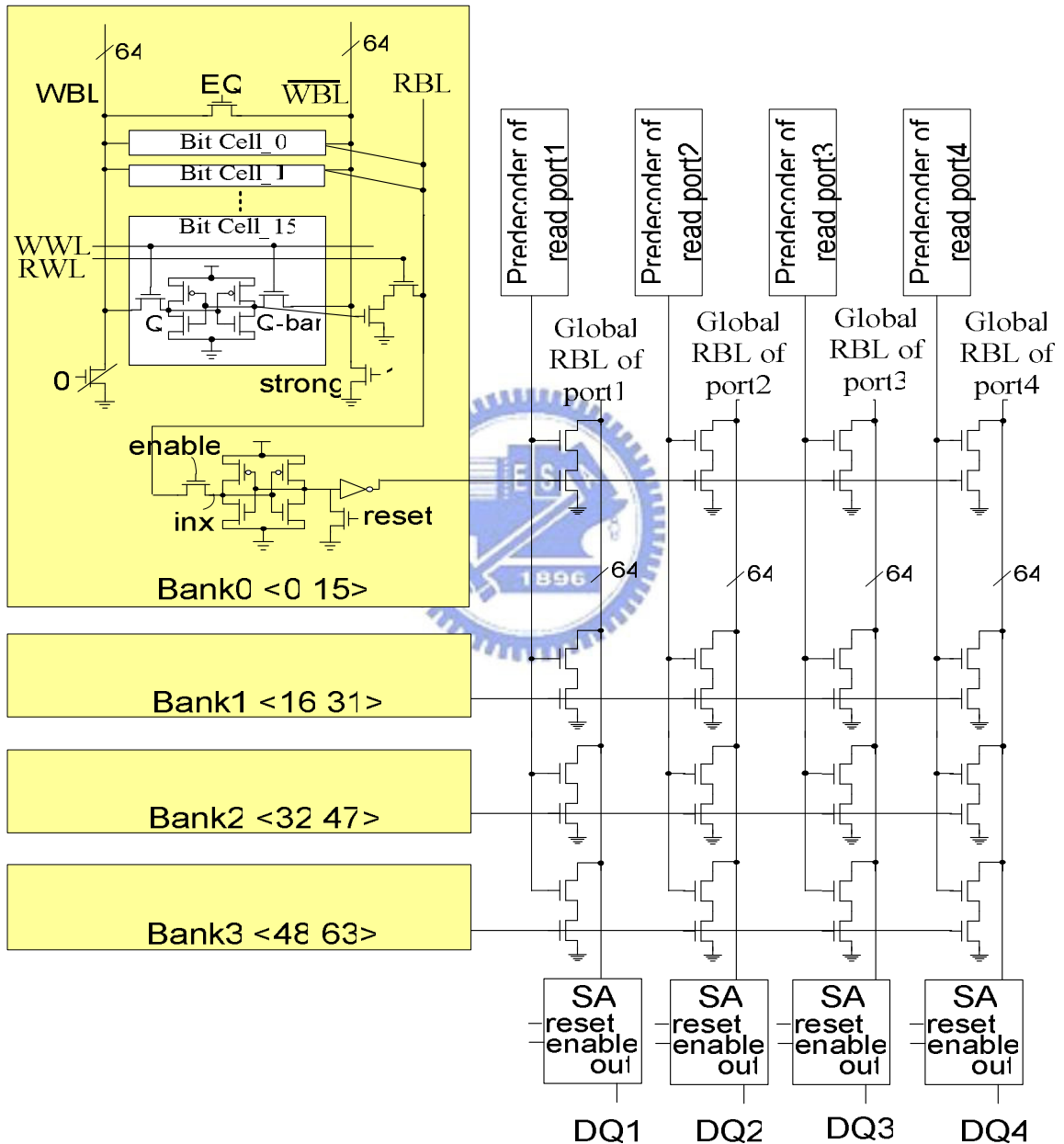


Fig . 3.14 Floating bitline access scheme

3.3.2 Divided Bitline

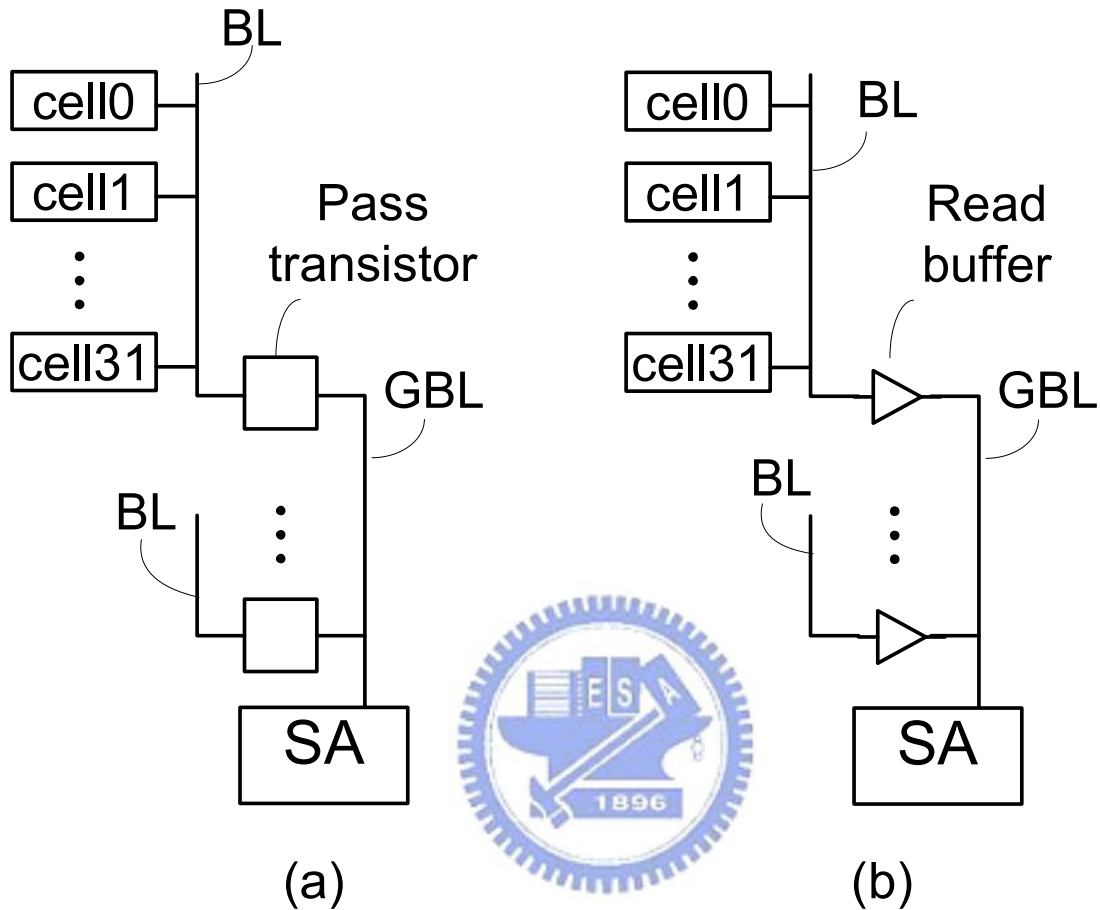


Fig . 3.15 Divide bitline connection (a) By pass transistor (b) By read buffer

In low power memory cell designs, the cell read current must create enough voltage different on the bitlines in a read operation. But this step spends a lot of memory delay time when the bitline capacitance is very large. To rescue this problem, one way is to make the read current as large as possible. However, this way results in high leakage currents because a large read current need large transistor widths, high supply voltage, or low threshold voltages for the cells.

The other way is to reduce the bitline capacitance. Access delay time would decrease when the amount of charge that the cell must draw from the bitline is reduced. An original bitline is divided into several shorter local bitlines, and a global bitline is designed to connect these local bitlines.

Often, local bitline connect to global bitline through a simple transistor, as shown in Fig. 3.15(a). This way adds little area overhead. However, the cell still needs to sink all the charge from global bitline. When a read buffer is inserted between local bitline and global bitline, the cell just need to sink the current from local bitline, as shown in Fig. 3.15(b). When the local bitline uses a large voltage swing, a simple inverter is allowed to be used as a sensing element for the buffer.

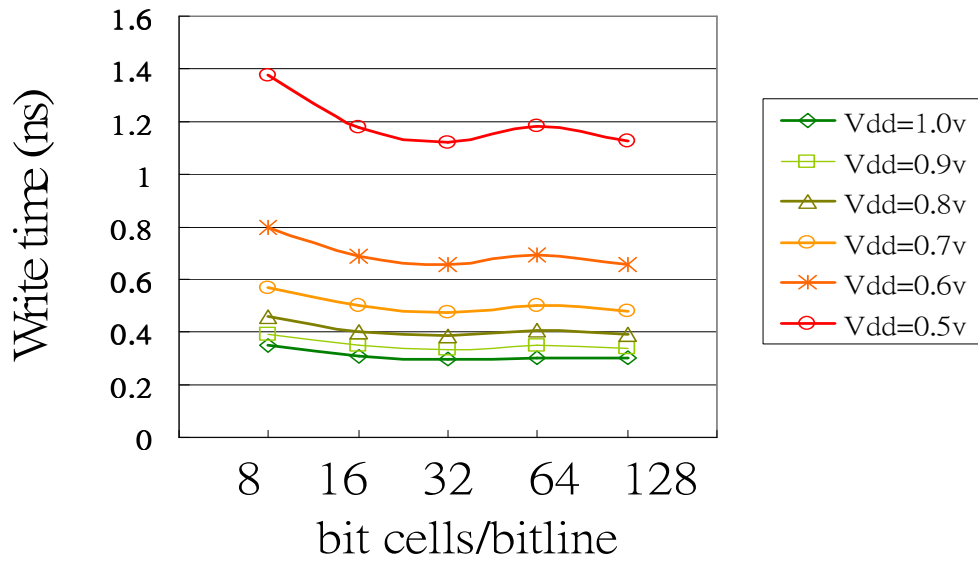
A inverter is not used as a sensing amplifier in the proposed register file, because the floating bitline read scheme is implemented. There are not a complete logic 1 in the scheme. The voltage of floating bitline is a strong 0 or a floating voltage which can not sensed correctly by a simple inverter.

A single end sensing amplifier is inserted between the local bitline and global bitline in our scheme, as shown in Fig. 3.14. The output of local sensing amplifier is connected to four global bitline through read buffers.

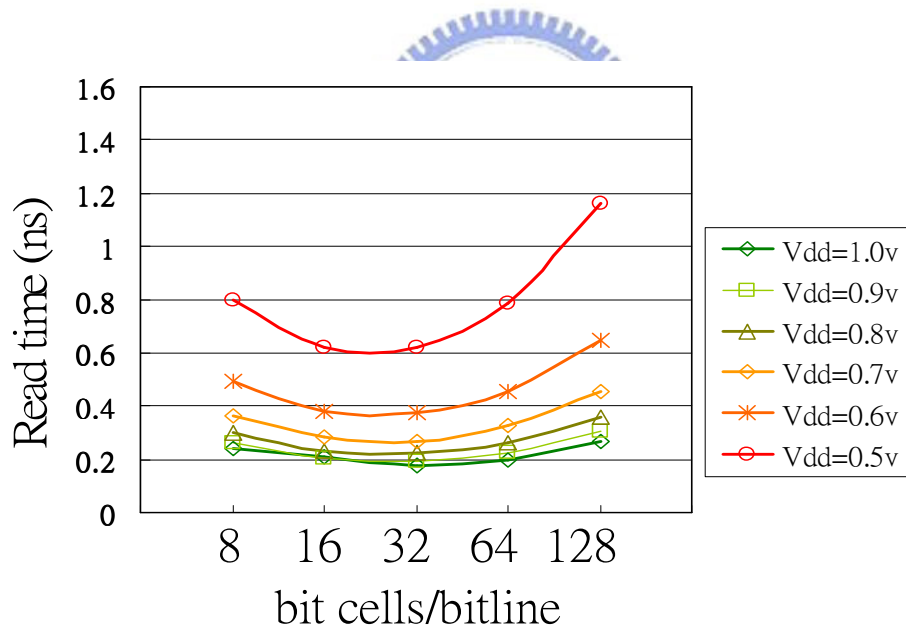
A shorter bitline has smaller bitline capacitance; however, it leads to a global bitline capacitance increasing. Some simulations should be done to find the optimized bitline length. Fig. 3.16 shows the simulation results about access time with different bank sizes when operating voltage is between 0.5V and 1.0V. It clearly shows that the best performance happens when bank size is 32 words.

In other words, the access delay is shortest when the capacitance

of a local bitline is approximate equal to that of a global bitline.

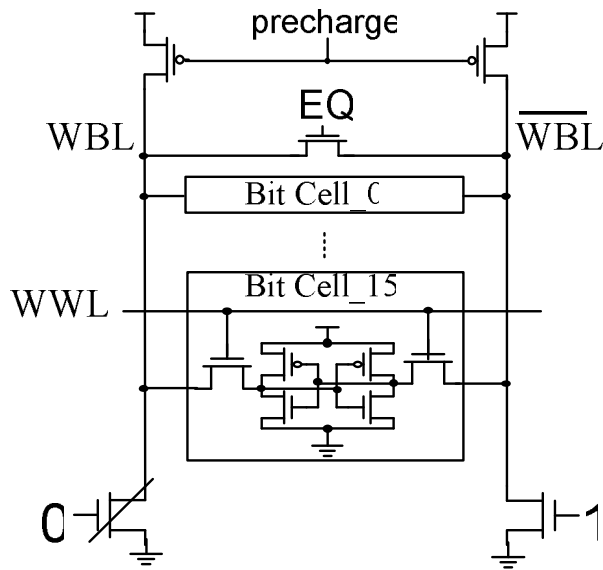


(a)

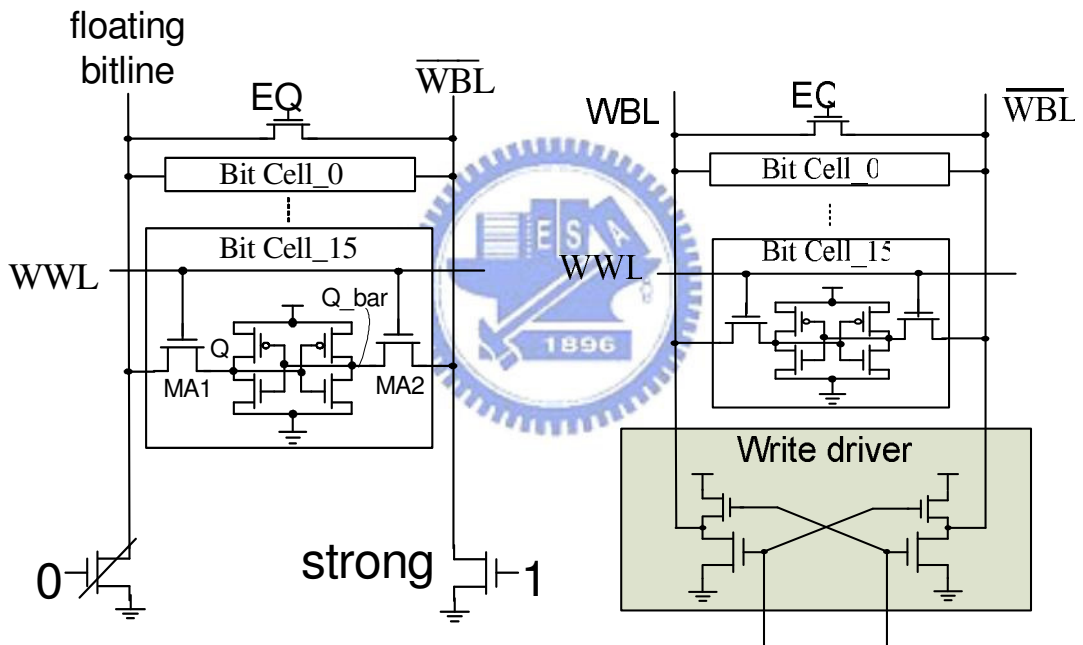


(b)

Fig. 3.16 Access time with different word numbers per bank
 (a) Write access time. (b) Read access time.



(a)



(b)

(c)

Fig . 3.17 write scheme (a) conventional write scheme
 (b)floating write scheme (c)Push-pull write scheme

3.3.3 Push-Pull Write Scheme

In conventional write scheme, the bitline pairs need to precharge before the write driver is turned on, as shown in Fig. 3.17(a).

Fig. 3.17(b) shows the floating write bitline architecture. During Write cycles, only one side of a bitline pair is connected to ground while the other side is floating. After one storage node is discharged approximately to zero, wordline is deactivated. Then, the cell acts like a latch, and the small difference between Q and Q-bar would be quickly amplified. Finally, Q and Q-bar would become complete logic signals. When Write bitline pairs are no longer precharged, access time would become slightly longer but power can be reduced a lot.

However, the write ability of this scheme is weaker than conventional write architecture. Node Q may be discharged by floating bitline, when the threshold voltage of MA1 is lower than that of MA2 under technology fluctuation.

To prevent the write fault discussed above, the push-pull write scheme is implemented here, as shown in Fig. 3.17 (c). When one of the bitline pair is discharged to ground, another one would be pulled up to $V_{dd} - V_{tn}$ by clamping NMOS in a write operation. Push-pull write scheme consumes larger power than floating write scheme, but has better write ability at the same time.

3.4 Conclusion

This register file is divided into four interleaved banks. Each bank has one Read port and one Write port. Therefore, it can save a lot of power and area overhead. However, bank conflicts would happen if there are too many accesses to the same bank. To ease access conflict in this design, timing sharing access scheme is proposed.

To operate the circuit correctly under process variation and wide range of supply voltage, the design of timing control circuit is a critical issue. It is discussed in the next chapter.

Several low power techniques are proposed in this chapter, such as floating read bitline, push-pull write scheme, and the divide bitline scheme.



Chapter 4

Decoder and Control Circuit Design

In this chapter, the design of control circuit for timing shared, multithreaded 4W/4R register file will be presented. Fig 4.1 is the block diagram of proposed register file. Multiple access ports with multiple decoders are connected to less port cell through a multiplexer. Multiported register files realize multi-port access by using less port memory cells instead of fully-port memory cells to relax the heavy routing wires.

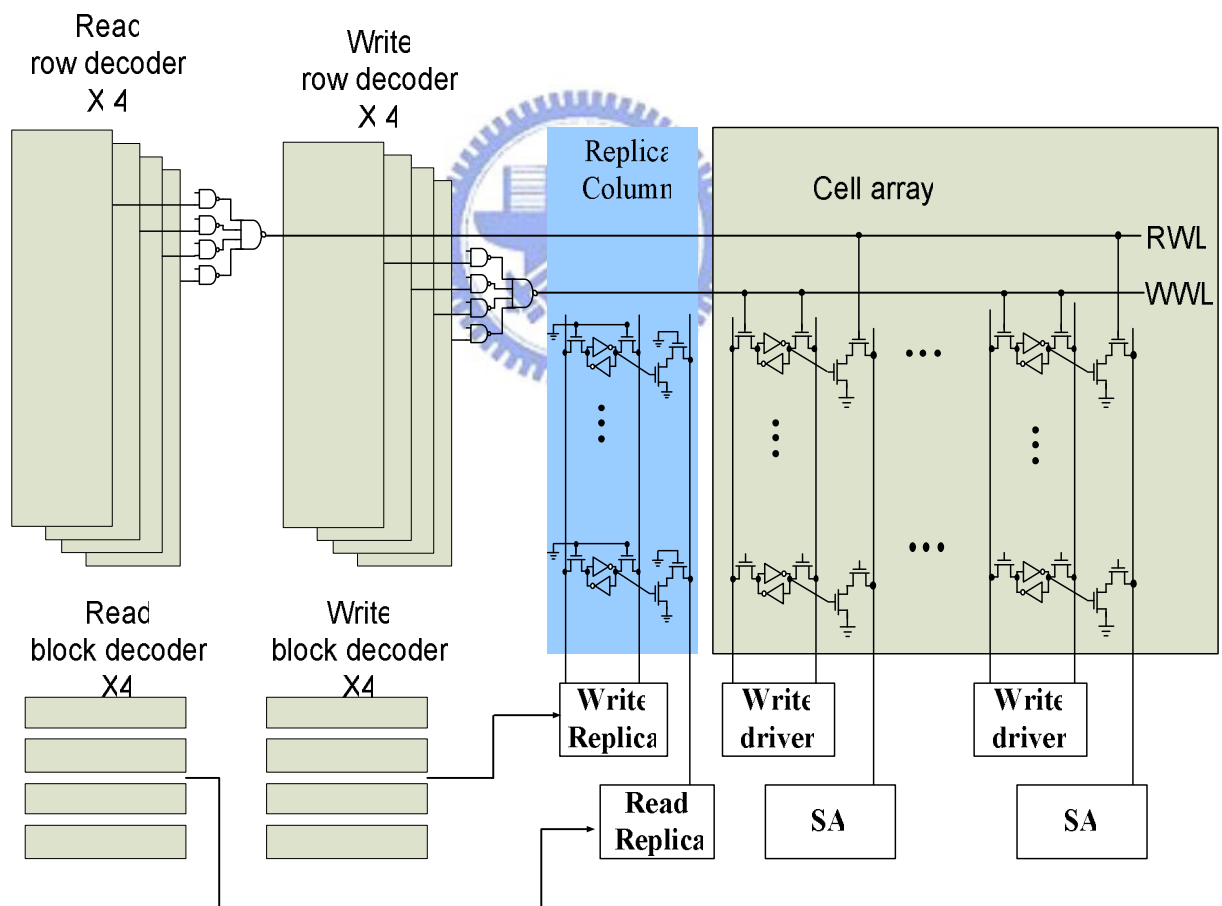


Fig.4.1 Block diagram of proposed register file

The cell array content the storage elements holding the data. Symbolic addresses are converted to physical addresses through the

decoders. Block decoders and Write/Read replica circuits control the write/read operation to work correctly. Replica column, connected to Write/read replica circuits, traces the bitline delay under all PVT variations.

4.1 Decoder design

The Design of a register file is generally divided into two parts, the decoder and the sense and column circuits. Decoder is the circuitry from the address input to the wordline. The sense and column circuits include the bitline to the data input/output circuits. For a read operation, the decoder contributes up to half of the access time and a significant fraction of the total power consumption.

While the logical function of the decoder is simple, it is equivalent to 2^n n-input AND gates, there are a large number of options for how to implement this function. Modern RAMs typically implement the large fan-in AND operation in an hierarchical structure. The decoder designer has two major tasks: choosing the circuit style and sizing the resulting gates, including adding buffers if needed.

As shown in Fig. 4.2, the symbolic addresses are separated into two groups and decoded separately. Upper addresses are decoded by block decoder when lower addresses are decoded by row decoder. Because of such partitioning, register banks can be disabled when other banks are accessed. The block decoder divided into three parts, including block select, priority decision and pulse width control, as shown in Fig 4.3.

A nand gate is inside the Block select. Upper addresses are decoded through this nand gate. The Priority decision arbitrate which two operations should be executed when access conflicts happen. Each

port of this register file has different priority levels. The priority of Read (Write) port1 is higher than Read (Write) port3 respectively, and Read (Write) port2 is higher than Read (Write) port4 as well. Signal 'W_TS1' and 'W_TS12' come from the write-timing-control circuit, which would be discussed later.

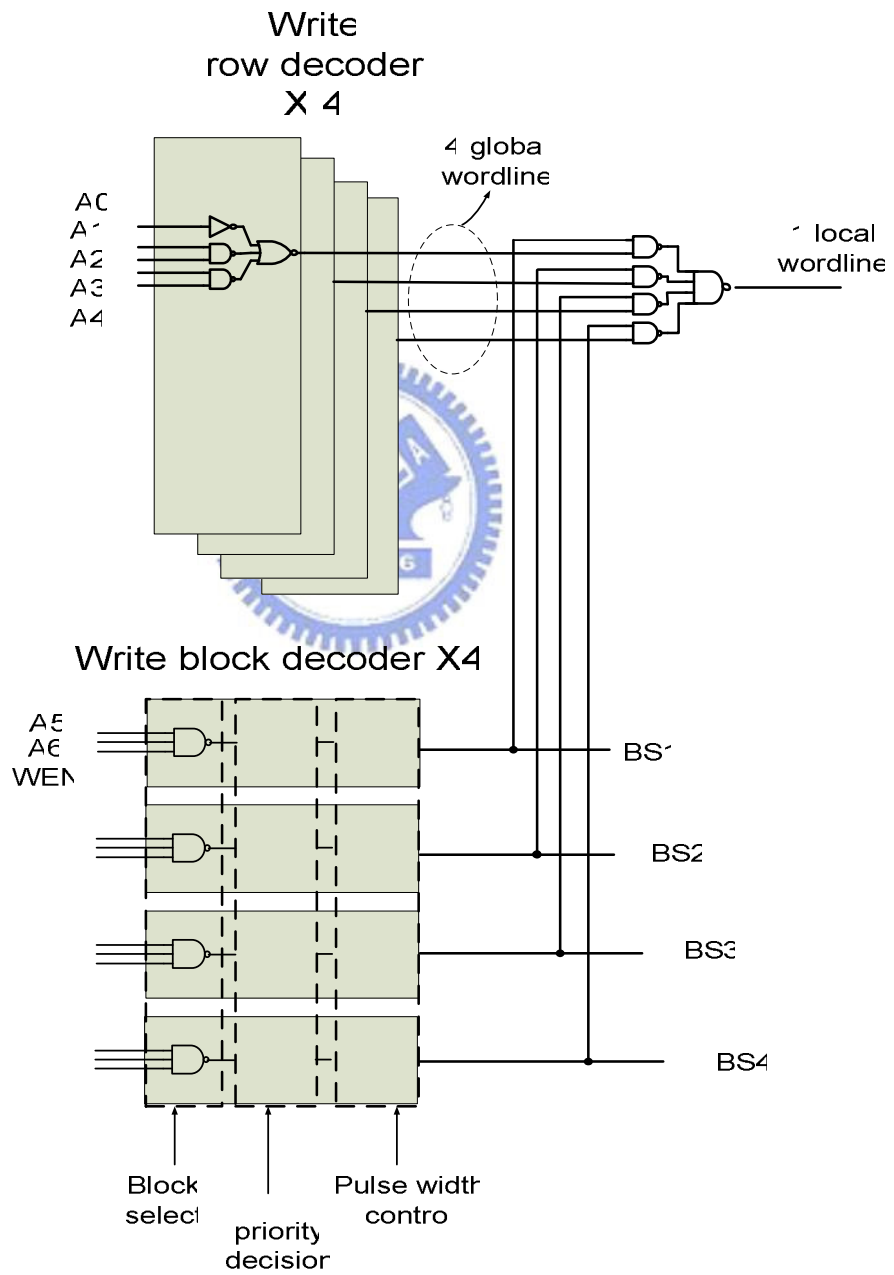


Fig. 4.2 Write row decoder and block decoder

The Pulse width control circuit control the width to be just enough for a write operation and divide the pulse into two working timing slot. An example is shown in Fig 4.4 to explain this concept.

Write block decoder X4

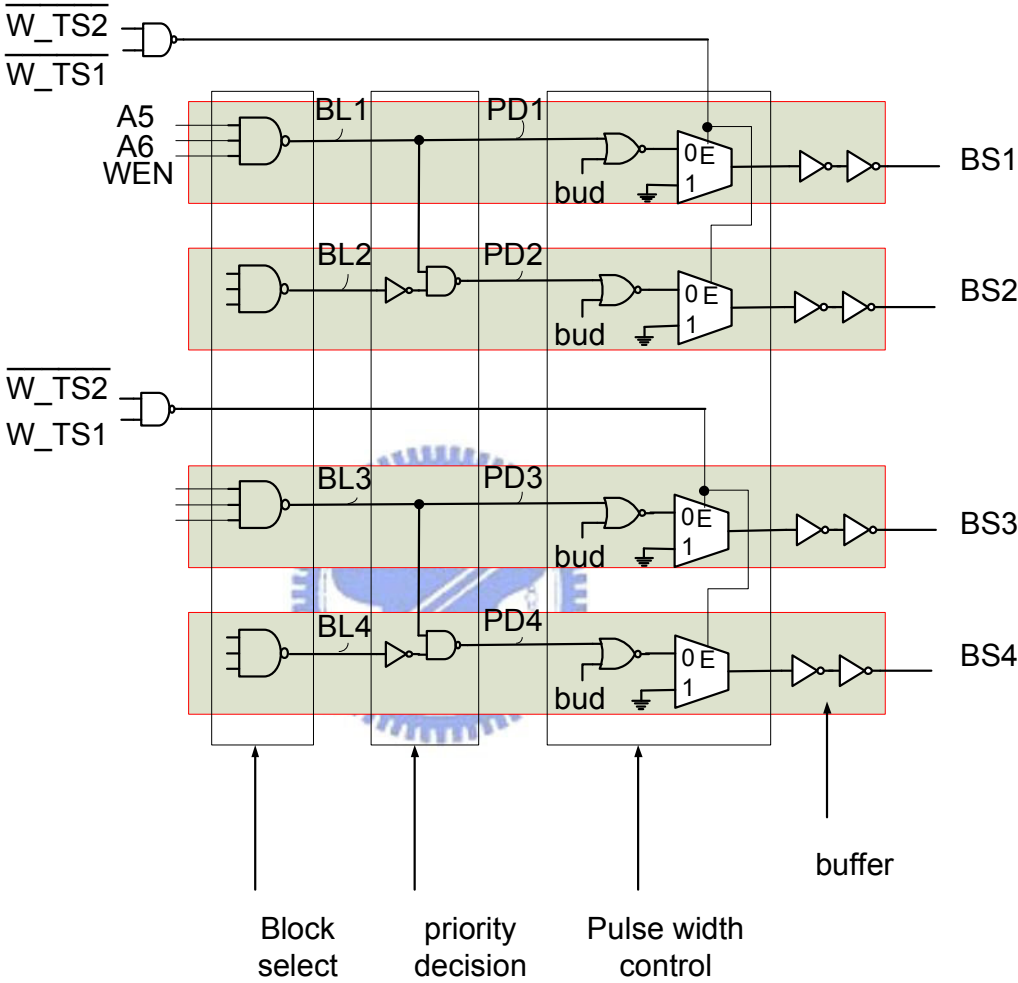


Fig. 4.3 write block decoder

In cycle 1, all the four ports want to access this bank, so the voltage of signal BL1, BL2, BL3 and BL4 are pulled up to logic '1'. However, one bank only can be accessed twice in one clock cycle. After the operation of the Priority decision circuit, port1 and port2 can operate successfully while both port3 and port4 would receive Read fail signals. Therefore, signal PD1 and PD2 would be pulled high when PD3 and PD4 are still logic '0'.

Port1 and port 3 work in the first time slot, and port2 and port4 work in the second time slot. The Pulse width control circuit control signal BS1 and BS2 to be just enough for a write operation. BS1 is pulled high at first working timing slot, when BS2 pulled high at second working timing slot.

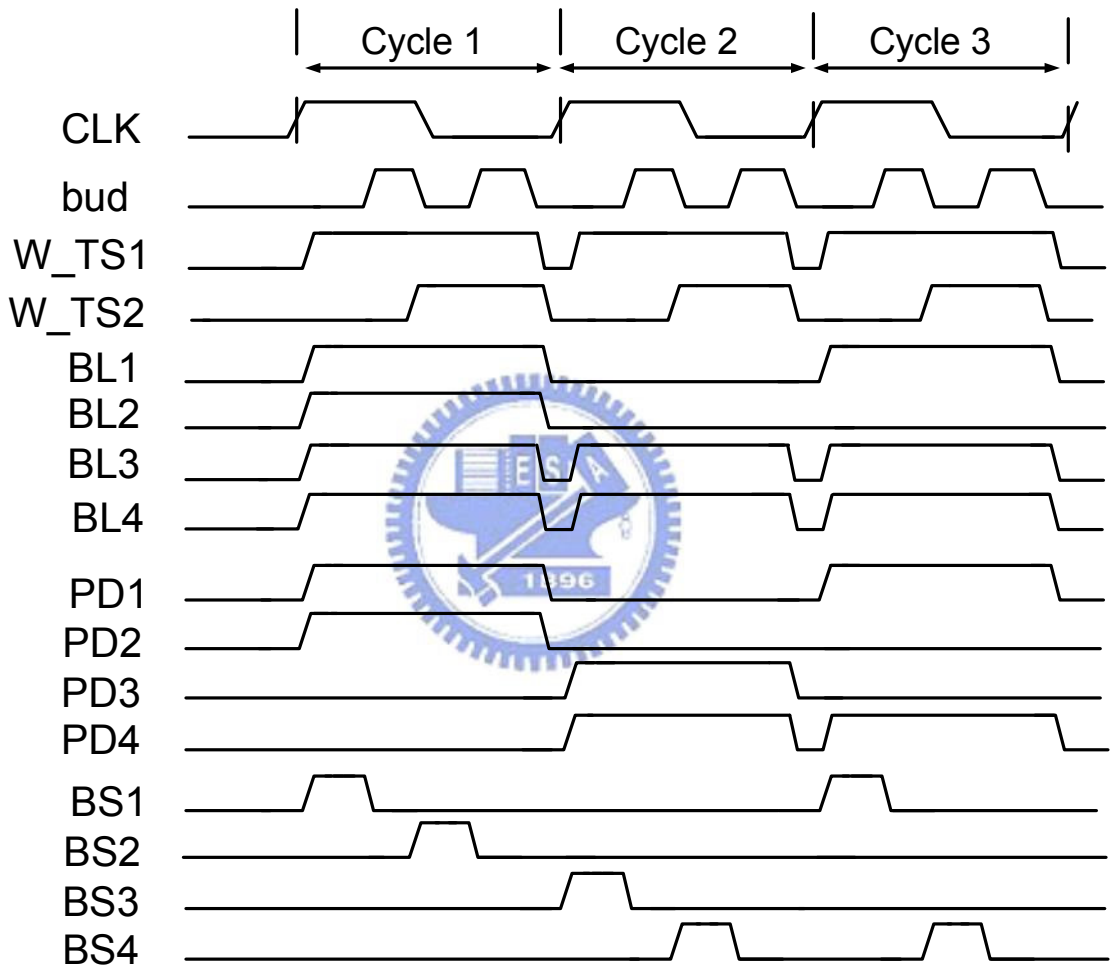


Fig. 4.4 Waveforms illustrating concept of block decoder

In conventional, to decode one word need a group of logic gate, as shown in Fig 4.5(a). However, some logic gate of near by wordlines are the same. In order to save more power and area, some logic gate are shared by the two neighboring words, as shown in Fig. 4.5(b).

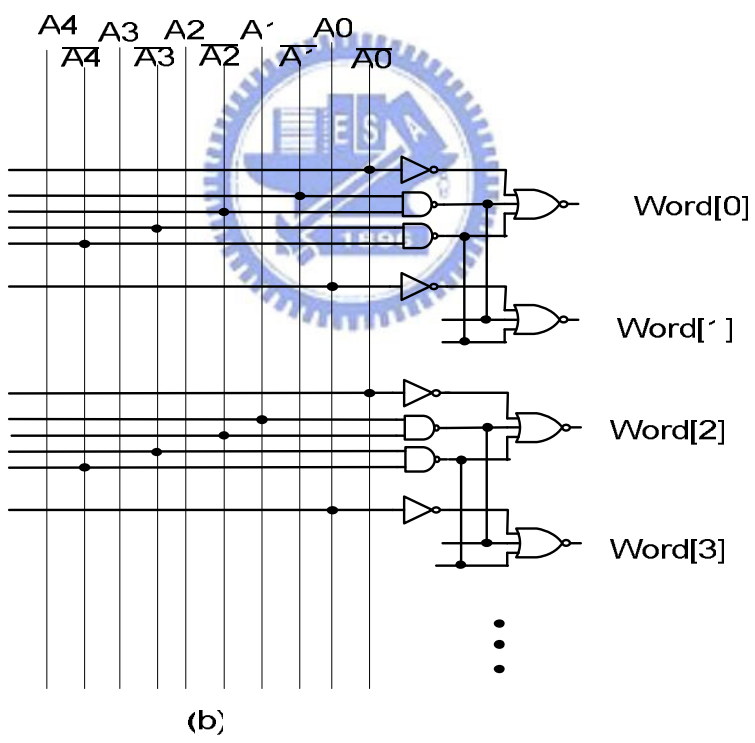
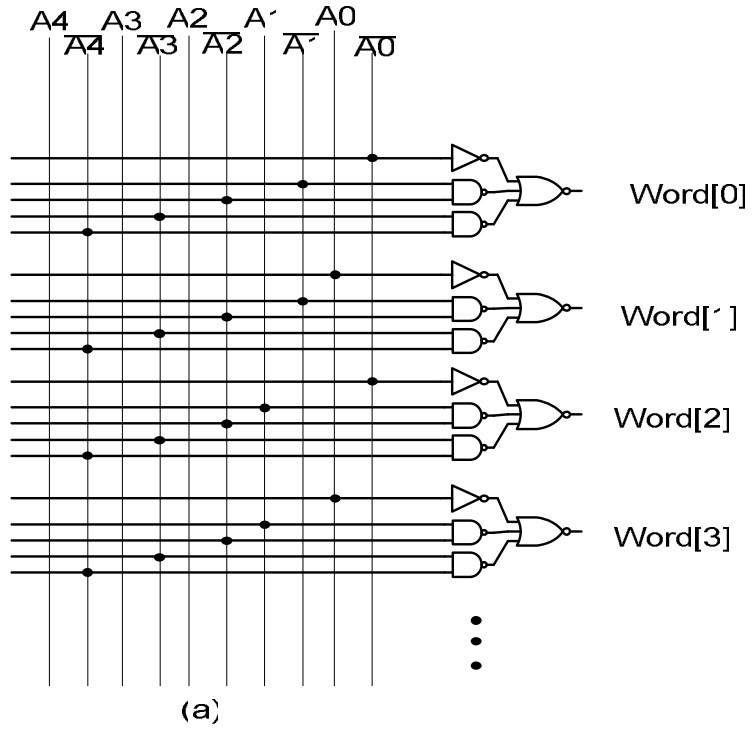


Fig. 4.5 The row decoder design (a) Conventional design. (b) The scheme with fewer logic gate.

4.2 Logical effort model

The convention of a register-transfer level description of a design into an implementation in silicon starts with logic synthesis, which consists of technology independent optimization, followed by technology mapping. In the latter step, the design is mapped to cells belonging to the target library while optimizing one or more performance metrics, such as delay, area, or power.

High-performance designs use rich libraries, with multiple instances of each cell, which have various delay, area, and drive capabilities. Technology mapping has to identify not only the best logic functionalities of cells to be used to implement some logic but also the best instance of each selected cell.

Since the library cells may be repeated thousands of times during the digital design process, their quality determine the final product performance. The number of driving strengths available for each cell also, have a crucial impact on the design performance. For instance, when a design is implemented by a single driving strength library, its performance degrades by up to 27%. This is compared to its performance when it is implemented using a library that uses three levels of driving strengths.

Hence, there is a need for multiple libraries for each technology process, which is impractical. The situation is exacerbated when there is a need for a diversity of libraries from different suppliers where each one has its own tools and documentations. As a result, virtual library concept has emerged as a solution for this problem.

Virtual library or library-free mapping terminology means, mapping the design's Boolean functions to the transistor level directly instead of using pre-characterized cells. Usually, the Boolean functions in this mapping technique are realized using

Static CMOS Complex Gates (SCCGs). The number of SCCGs (Boolean functions) in a virtual library is determined by the allowed number of serially connected transistors.

The logical effort method is widely recognized as a pedagogical way allowing designers to quickly estimate and optimize single paths by modeling equivalently propagation delay and transition time.

4.2.1 Logical effort and gate sizing

In the method of logical effort, the delay of a gate is estimated by modeling it as a linear function of the load being driven as

$$D = g \times \frac{c_l}{c_i} + p = g \times h + p = f + p \quad (4.1)$$

where g is the logical effort, $h = C_L/C_i$ is the electrical effort, C_L is the path load capacitance, $f = gh$ is the effort delay and p is the parasitic delay of the gate. This formulation separates the different components that contribute to the delay of a gate. More importantly, it leads to a natural extension for estimating the minimum delay, \hat{D} , of a path of logic as

$$\hat{D} = NF^{1/N} + P \quad (4.2)$$

where $F = GBH$ is referred to as the path effort, P as the path parasitic delay, and N as the number of gates on the path under consideration. The path logical effort, G , is the product of the logical efforts of the gates on the path, and the path electrical effort, H , is the product of the gate electrical efforts. The minimum delay of (2) is obtained by distributing the path effort F equally to each gate on the path.

For the networks with loads off logical path, as shown in Fig. 4.6, branching effort b should be introduced. The branching effort b at the output of a logical cell.

$$b = \frac{C_{on-path} + C_{off-path}}{C_{on-path}} \quad (4.3)$$

where $C_{on-path}$ is the load capacitance of a logical gate along the considered path, and $C_{off-path}$ is the load capacitance of a logical gate(s) off the path. The branching effort along an entire path B is the product of the branching effort at each of the stages along the path.

$$B = \prod_{i=1}^N b_i \quad (4.4)$$

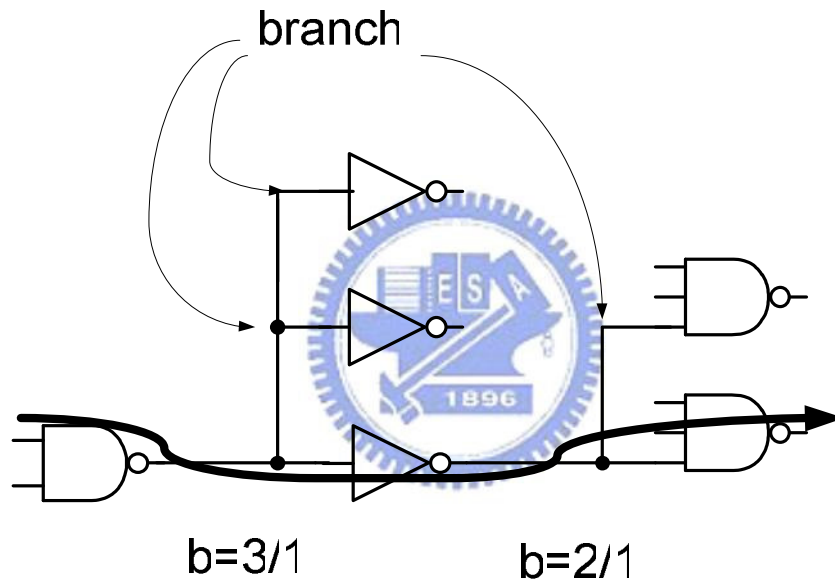


Fig. 4.6: A logical path with branching

The path electrical effort can also be calculated as the ratio of output and input capacitances of the path. Consider Fig. 4.7, which shows a simple path of four gates—A, B, C, and D. Each of these gates have input capacitances C_{inA} , C_{inB} , C_{inC} , and C_{inD} and drive output capacitances C_{outA} , C_{outB} , C_{outC} , and C_{outD} , respectively. The input capacitance of the path C_{in} is the input capacitance of gate A, and the output capacitance of the path C_L is the output capacitance of gate D. The path electrical effort, H , the product of the gate electrical efforts, telescopes, since

the input capacitance of each gate is the load capacitance of its input (e.g., $c_{inC} = c_{outB}$). Thus

$$\begin{aligned}
 H &= \frac{c_{outA}}{c_{inA}} \times \frac{c_{outB}}{c_{inB}} \times \frac{c_{outC}}{c_{inC}} \times \frac{c_{outD}}{c_{inD}} \\
 &= \frac{c_{outA}}{c_{in}} \times \frac{c_{outB}}{c_{inB}} \times \frac{c_{outC}}{c_{inC}} \times \frac{C_L}{c_{inD}} \\
 &= \frac{c_{outA}}{c_{in}} \times \frac{c_{outB}}{c_{inB}} \times \frac{c_{outC}}{c_{inC}} \times \frac{C_L}{c_{inD}} = \frac{C_L}{c_{in}}.
 \end{aligned}
 \tag{4.3}$$

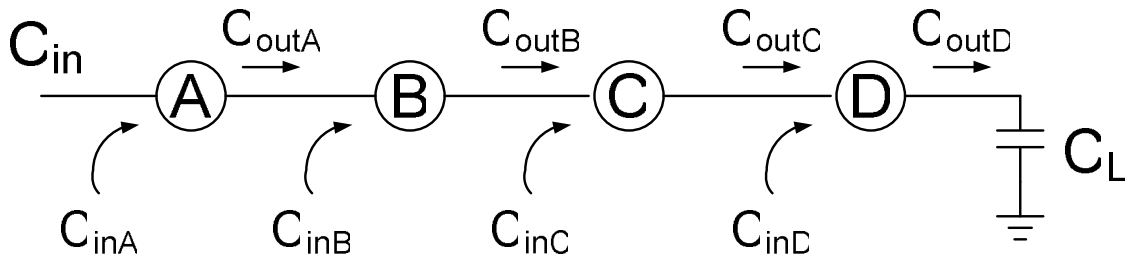


Fig. 4.7. Calculating the electrical effort of a path.

The logical effort approach is well suited for estimating the minimum delay that can be achieved by sizing a path of logic if the electrical effort, H , of the path is known. The individual gate sizes that are required to achieve this minimum delay can be calculated as follows: Each gate is assigned a gate effort of $f = F^{1/N}$. Starting with the gate at the output that drives a known load of C_L , the size of each gate is successively determined. Since the logical effort g of a gate is fixed, if an effort delay f is assigned to a gate, the input capacitance c_{in} that meets this effort delay can be calculated as

$$c_{in} = \frac{g \times c_l}{f}
 \tag{4.4}$$

where c_l is the load being driven by the gate under consideration.

4.2.2 Logical effort design steps

There are a three-step process to size a logical path to a achieve a required time with minimized area.

First, determine the input capacitance of the logical path and calculate H and, F , and the optimal number of gates N . If N is greater than the actual number of gates, add buffers to the path to match N .

Second, Calculate p_i and g_i for each gate. Third, calculate and roughly estimate fanout for each gate. The calculation should start from the last gate towards the first gate (at the input). This allows a rough estimation of the transistor sizes for each gate where C_i is the input capacitance of the current gate and C_{i+1} is the input capacitance of the next gate that is closer to the path output. q_i can also be calculated at this stage.

Cell- or library-based technology mapping is the process of binding a technology-independent logic level description of a circuit to a library of gates in the target technology. A dynamic programming algorithm based on tree covering has served as the basis of later technology mapping algorithms. This is a two-step algorithm.

In the matching step, matches for all gates are generated in an input-to-output traversal of the circuit, and the optimum match (based on its cost and the cost at its inputs) and the corresponding matches at the inputs are stored as the solution for that gate.

In the covering step, the solution for the entire circuit is generated by an output-to-input traversal of the circuit. At the primary outputs, the best match is selected, and the covering recurses on the inputs of this match.

4.3 Timing Control Circuit

Various operating voltage, frequency, and configurations are required for mobile applications. For mobile devices, reduction in operating voltage is strongly required in order to reduce power consumption. And they need at least 27MHz operation to synchronize with the frequency of base band. Thus the register file need to operate under low and wide Vdd.

In order to prevent too large subthreshold leakage, the threshold voltage doesn't be scaled down as fast as supply voltage. Therefore, the gate overdrive for the transistors is reduced. At the same time, the fluctuations of threshold voltages are not able to decrease in future technology. The delay variability of all low power circuits across process corners and various operating voltage will become larger and larger in the furture.

To operate the register file correctly at all process corners and the wide range of Vdd from 0.5v to 1.0v, the design of delay control circuit for the access timing is the most critical issue. Therefore, the Write replica Circuit and Read replica Circuit are designed to operate the register file correctly and save power at the same time.

The large delay spreads across process corners will necessitate bigger margins in the design of the bitline path in a register file, and will result in larger bitline power dissipation and loss of speed. This problem can be mitigated by using a self-timed approach to designing the bitline path, based on delay generators which track the bitline delays across operating conditions.

Controlling the wordline pulse width to be just wide enough to guarantee the minimum bitline swing development can further minimize bitline power. This type of bitline swing control circuit

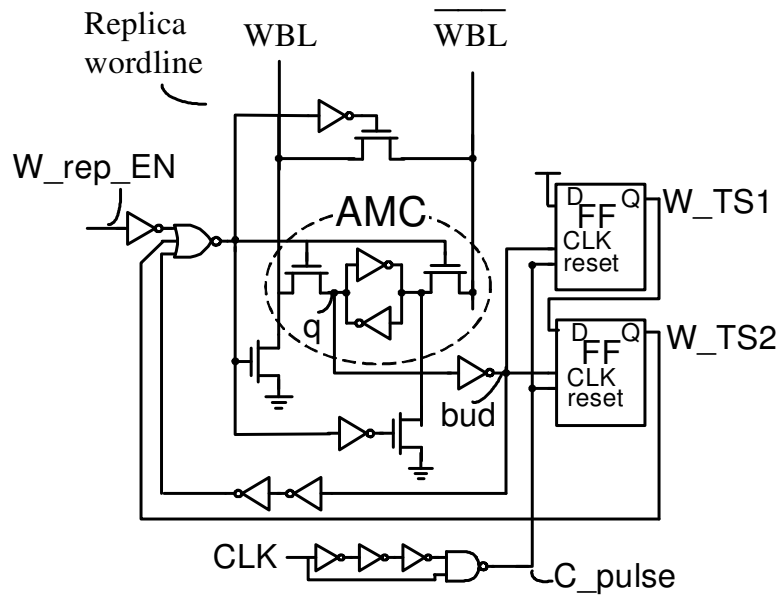
can be achieved by a precise pulse generator that can match the bitline delay. The sense clock starts the amplification, and hence the sense clock needs to track the bitline delay to ensure correct and fast operation. To save the sense power, the sense-amplifier-activating period can also be designed to just wide enough for sensing successfully. Therefore, a replica SA is needed to track the sensing speed of SA.

Fundamentally, the clock path needs to match the data path to ensure fast and low-power operation. The data path starts from the local block select and/or global wordline, and goes through the wordline driver, memory cell, and bitline to the input of the sense amps. The clock path often starts from the local block select or some clock phase, and goes through a buffer chain to generate the sense clock. The delay variations in the former are dominated by the bitline delay since the memory cells are made out of minimum sized devices and are more vulnerable to process variations.

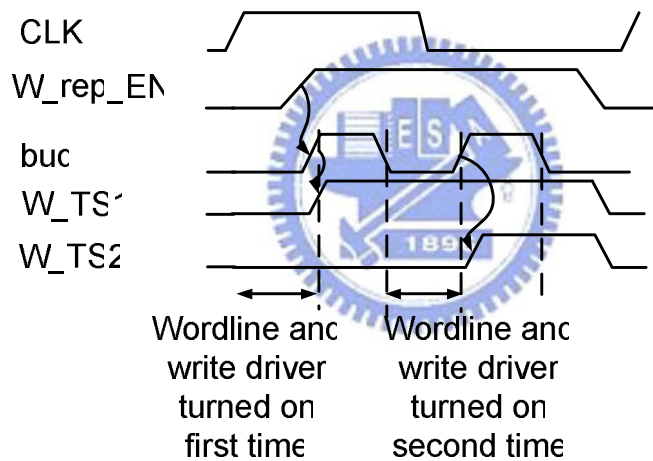
Therefore, the delays of the two paths do not track each other very well over all process and environment conditions. Enough delay margin has to be provided to the sense clock path for worst case conditions, which reduces the average case performance.

4.3.1 Write Replica Circuit

It is important to control the wordline-activating period to reduce the power consumption in write-access operation. Fig. 4.8 shows the Write replica circuit and its signal waveforms. Because there are two working timing slot in one clock cycle, it need two flip-flops, W_TS1 and W_TS2, to control the replica circuit operating twice in one clock cycle. In order to improve replica circuit's performance, all flip-flops inside Read/Write replica circuit are TSPC D flip flop which has very small clock-to-Q delay.



(a)

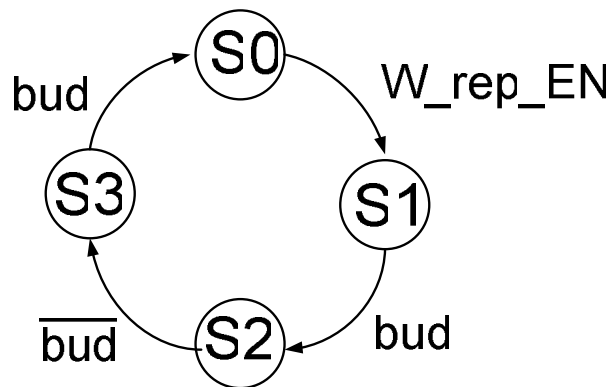


(b)

Fig.4.8 (a)Write replica circuit. (b) Signal waveforms of this circuit

The storage node 'q' of replica cell would be precharged to vdd. When one of the four global Write ports want to access this one subbank, the signal 'W_rep_EN' would be pull up to start the the write replica circuit. The replica write bitline(RWBL) is then discharged to ground. As the storage node of replica cell is discharged to 0v, replica circuit would turn off the wordline of

cell array and 'Replica wordline' in order to precharge node 'q' of replica cell for the second operation. The second operation starts when 'q' is reseted to Vdd.



Within one clock cycle

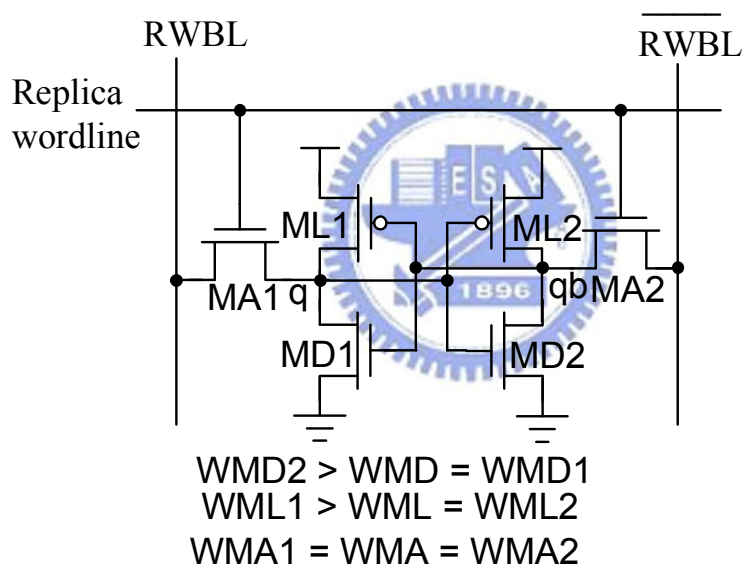
- S0 : standby , charge node 'q' to Vdd
- S1 : WL1 on , WL2 off
- S2 : WL1 off , WL2 off , charge node 'q' to Vdd
- S3 : WL1 off , WL2 on

Fig. 4.9. State diagram of write replica circuit

Fig. 4.9 show the state diagram of this Write replica circuit. There are four states within one clock cycle. S0 is the standby state. When signal 'W_rep_EN' is high, the state transit from S0 to S1. State S1 is the first working time slot. The wordline and write driver of cell array are turned on at state S1. When a write operation completes, the state transit to s2 and turn off wordline and write driver. At state S2, node 'q' of replica cell would be precharge to Vdd for the second operation. When q is pull up to vdd, the state transit to S3. State S3 is the second working time slot. The state transit to S0 and turn off wordline and write driver as the second write operation complete.

To write a data into register file correctly even with serious

timing fluctuation, the write access delay to asymmetrical replica cell (ARC) must be adjusted to replicate the slowest register file cell. Fig. 4.10 show ARC scheme. Its organization is the same as a 6T cell. The transistors of ARC are arranged that the storage node 'q' is hard to hold the logic low. The width of the drive-transistor WMD1 is smaller than the typical width WMD. And the width of load-transistor WML1 is larger than the typical width WML. The size of D2 and L2 are also adjusted in the opposite way. The Id of the drive-transistor MD2 and the load-transistor ML1 are increased. The sizes of the access-transistor MA1 and MA2 are the same as the typical transistor of register file cell.



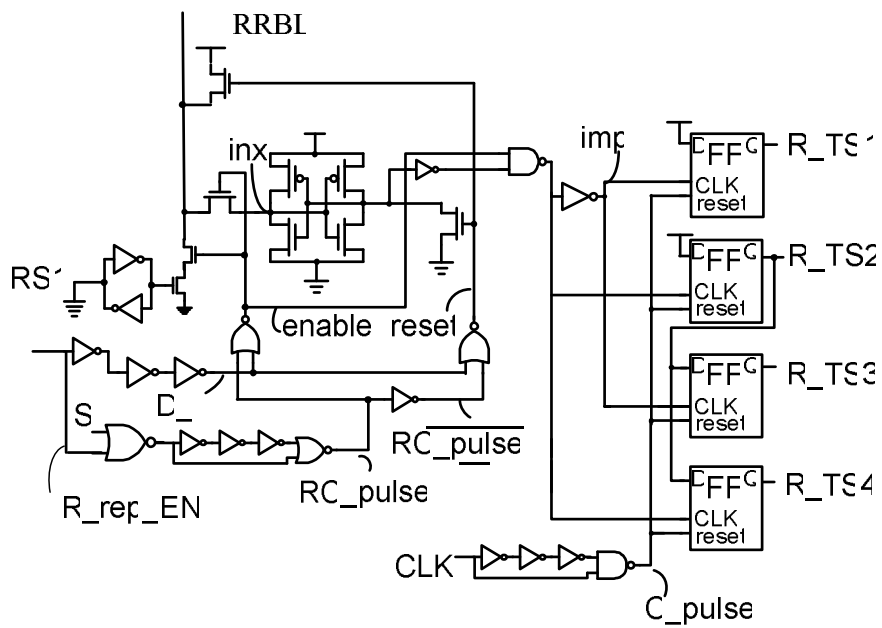
WMD, WML, WMA :Typical transisotr width in cell array

Fig. 4.10. The organization of ARC

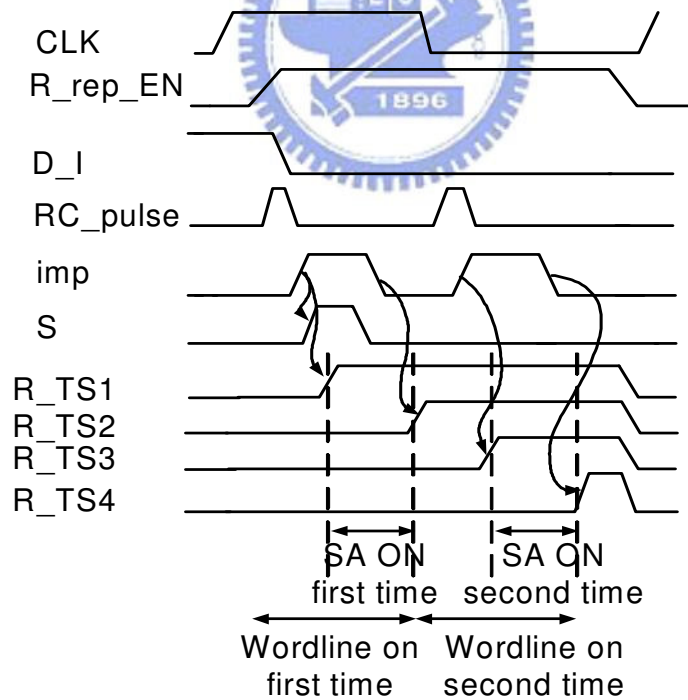
4.3.2 Read Replica Circuit

Fig. 4.11 shows the Read replica circuit and all signal waveforms. In order to save power, the turn-on time of the sensing amplifier and wordline should be minimized. The Read replica circuit detects which timing the SA and wordline should be turned on and turned

off.



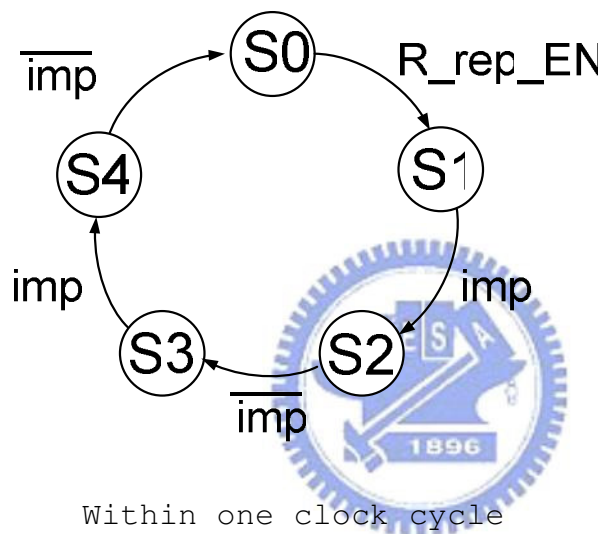
(a)



(b)

Fig. 4.11 (a) Read replica circuit. (b) Signal waveforms of this circuit

The storage node RS1 of replica cell is connected to ground in order to discharge the Replica BL. In order to detect the worst case delay time, the replica read bitline(RRBL) need to be precharged to $V_{dd}-V_{tn}$ before replica SA turned on. When one of four Read port starts to work, the Read replica circuit would be turned on by pulling up the signal 'R_rep_EN'. After node 'inx' of replica SA is reseted to 1, signal 'enable' will be turned on, then node 'inx' starts to be discharge. One Read operation completes when 'inx' is pulled down to '0'.



- S0 : standby
- S1 : WL1 on , WL2 off , SA off
- S2 : WL1 on , WL2 off , SA on
- S3 : WL1 off , WL2 on , SA off
- S4 : WL1 off , WL2 on , SA on

Fig. 4.12. State diagram of Read replica circuit

The circuit would work twice in one clock period. Therefore, there are several signals, 'R_TS1', 'R_TS2', 'R_TS3', and 'R_TS4', to control memory cell operating twice in one clock cycle. The 'reset' and 'enable' signal of local sense amplifier are come from signal 'RC_pulse' and 'imp', shown in Fig. 4.11, through a buffer. At the same time, signal 'reset' and 'enable' of global sense amplifier are produced by signal 'TS5', 'TS6', 'TS7', and 'TS8' through some

logic operation.

The state diagram of the Read replica circuit is shown in Fig. 4.12. In one clock cycle, there are four states. S0 is the standby state. When signal 'R_rep_EN' is high, the state transit from S0 to S1. State S1 and S2 are the first working time slot when state S3 and S4 are the second working time slot. In first working time slot, wordline(WL1) would be turned on at state S1 and S2 when sensing amplifier(SA) only is turned on at state S2. State S2 and S4 trace the sensing speed of SA.

4.4 Comparison with conventional register file

A conventional dual-thread 4W/4R 64 x 64 bit register file is implemented to compare with this work. Each register file cell of the conventional register file is a fully ported cell, as shown in Fig 4.13. The bit cell uses single ended write scheme, whose area overhead is much smaller than double ended write scheme. However, the single-ended-write cell has worse write ability. The lowest operation voltage of this cell is 0.77v. Writing '1' to a cell storing '0' with supply voltage smaller than 0.77v will cause a write fault.

Fig 4.14(b) is the layout photograph of a conventional register file cell, whose area is much bigger than ours, shown in Fig 4.14(a). The total area of a conventional register file including cell array, decoders, and control circuit is about $623 \times 508 \mu\text{m}^2$. The total area of this work is $426 \times 219 \mu\text{m}^2$. It can save 70.4% area comparing with conventional register file, as shown in Fig. 4.15(a).

In pre-simulation, the average active power of this work is 6.7mW at 500MHz with 1.0v when conventional register file consumes 14.9

uW. Comparing with conventional register file, this work can save 55.1% power consumption, as shown in Fig. 4.15(b).

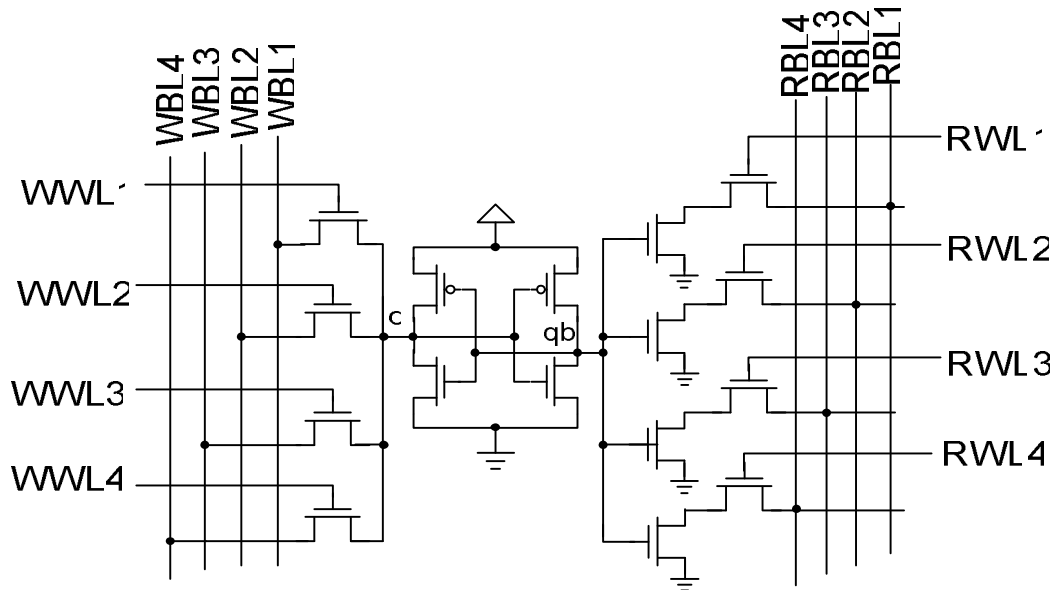


Fig. 4.13 A 4W/4R register file cell.

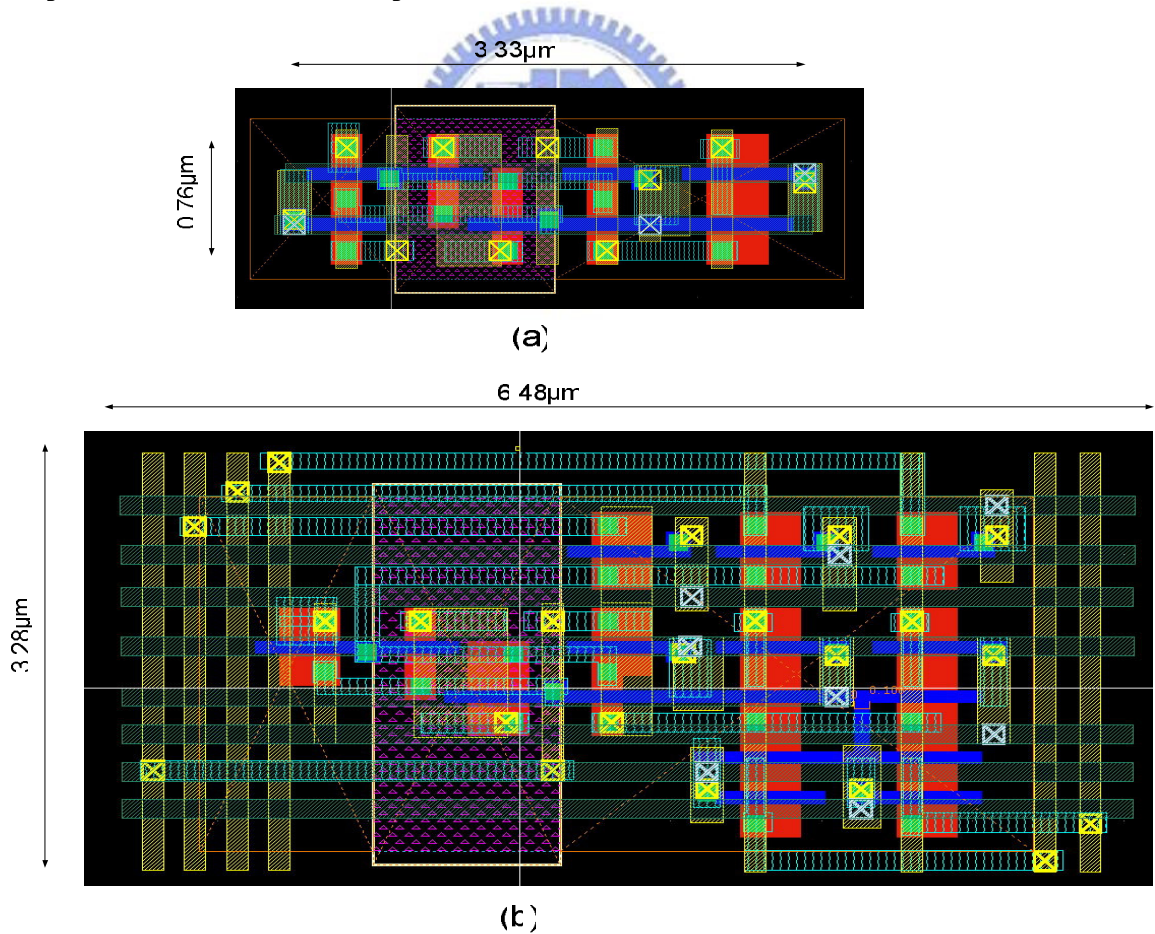


Fig 4.14. Layout photograph of register file cell. (a) A 1W/1R register file cell. (b) A 4W/4R register file cell.

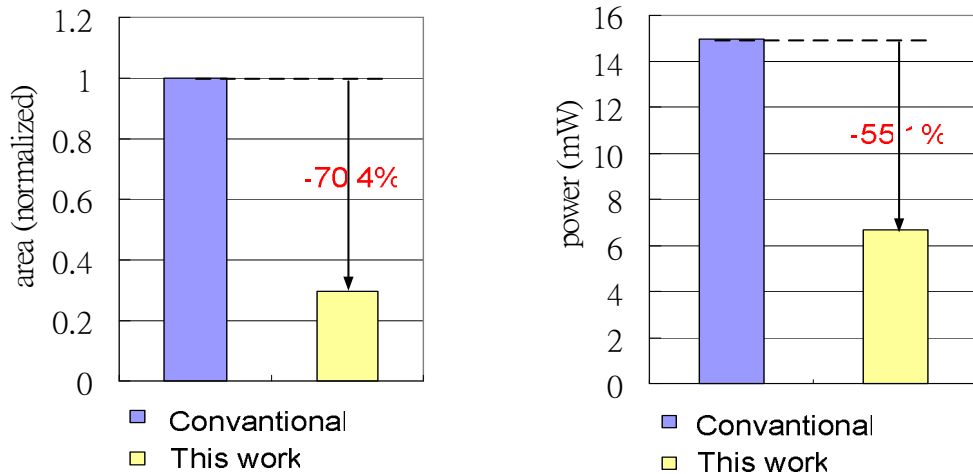


Fig 4.15 The comparison between this work and conventional design (a) area (b) power consumption.

4.5 The post-simulations

A dual thread 64 x 64 bits register file with the proposed low power techniques is implemented in UMC 90nm CMOS technology. Its simulation result is shown in Table 4.1. Operating voltage range is between 1.0v and 0.5v. It can operate up to 204MHz at 0.5v and consumes 197.51 μ W read power and 175.77 μ W write power at 50MHz with 0.5v. It consumes 3.62mW read power and 3.04mW write power at 250MHz with 1.0v. Fig. 4.16 shows the layout photograph of the proposed register file.

Technology	90nm UMC CMOS	
Configuration	Dual thread 4W/4R 64 x 64 bits	
area	426 x 219 μ m ²	
Power supply	0.5v	1.0v
Frequency	50MHz	250MHz
Read power	197.51 μ W	3.62mW
Write power	175.77 μ W	3.04 mW
Access time	10.42ns	2.70ns

Table 4.1 Register file simulation result

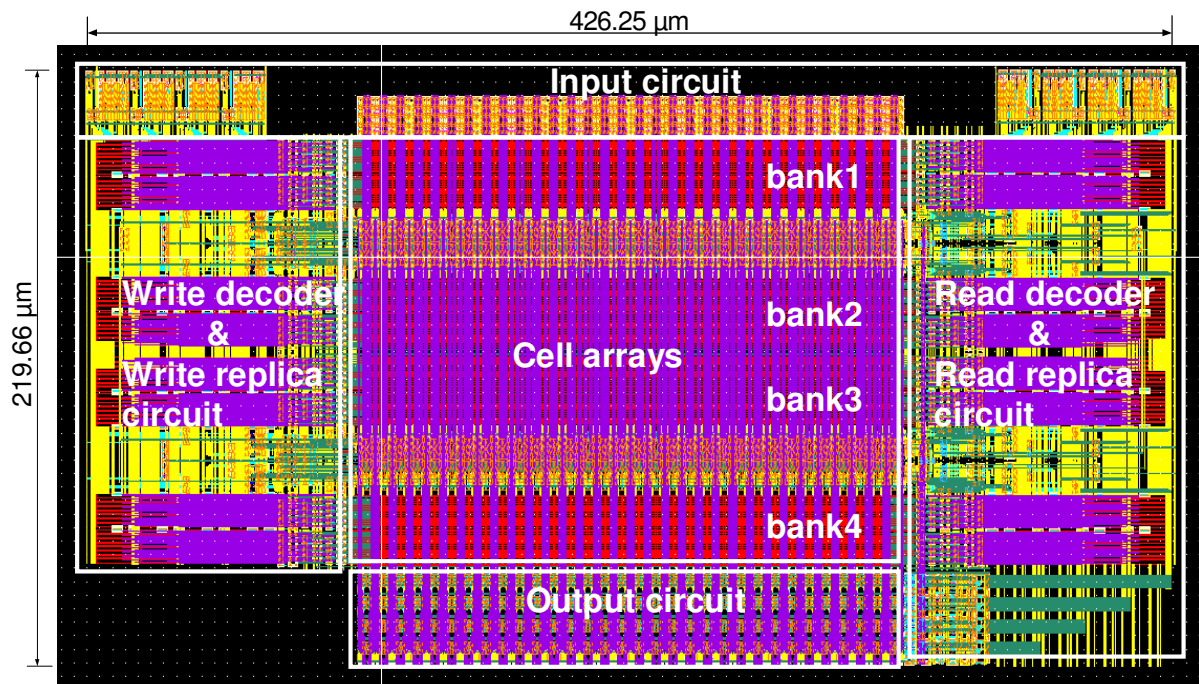


Fig. 4.16 Layout photograph of the dual thread 64 x 64 bits register file.

4.6 Conclusion



The decoder is separated into a row decoder and a block decoder. When a block is unused, it can be disabled by turning off the switch of block decoder to save power. Logical Effort technique helps determine transistor sizes for speed being an objective function.

The timing control circuit design which control the register file to operate rightly at all process corners and the wide range of V_{dd} from 0.5v to 1.0v is discussed in this chapter.

The dual thread 64 x 64 bits register file implemented in UMC 90um CMOS technology consumes around 197.51μW to 175.77μW at 50MHz with 0.5v and consumes around 3.62mW to 3.04mW at 250MHz with 1.0v.

Chapter 5

Multithreading and Multi-core systems

5.1 Different type of Register file organizations

Register files are not only the storage elements but also the communicational component. For multi-port register files above a threshold size, the area of the communication switch dominates the area of the register file. This section recognizes the ways to rearrange and decouple the storage and communication of register files.



5.1.1 Clustered architecture

The scheme, used in the Alpha 21264 [5.1] and 21464 [5.2] designs, consists of dividing the functional units among two clusters and providing a copy of all registers in each cluster. This approach halves the number of read ports required on each copy of the register file, but requires the same number of write ports on both register files to allow values produced in one cluster to be made available in the second cluster.

An extension of this approach is to develop a clustered architecture that divides the registers among a number of clusters [5.3], [5.4], [5.5], [5.6], [5.7]. Clustered architectures also allow the instruction window to be divided among clusters and have the potential to scale to larger issue widths at high clock frequencies. The number of write and read ports on each individual physical register and the overall complexities of the physical

register file, the bypass network and the wake-logic are decreased.

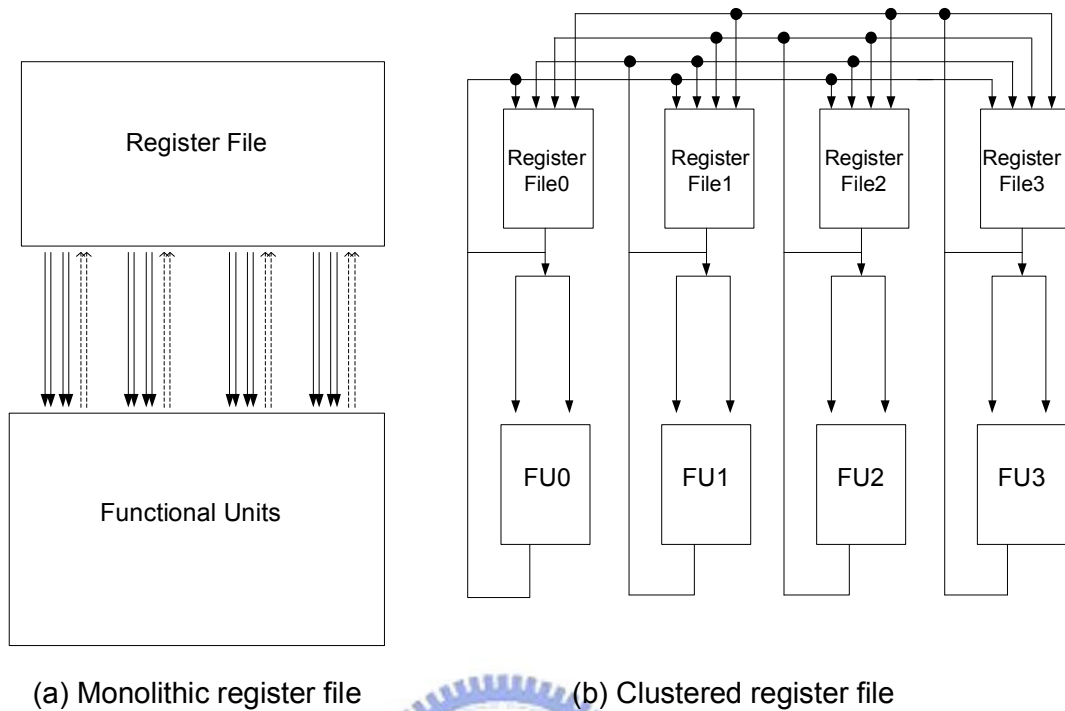


Fig. 5.1 Monolithic versus clustered register file organization

For example, a 4 -cluster architecture is shown in Fig. 5.1. Compared with a conventional superscalar architecture (Fig. 5.1(b)), the 4-cluster architecture presents a major difference: any physical register is connected with only half of the functional unit entries and can be written by only one fourth of the functional units.

However, Clustered architecture requires inter-cluster communication when a value is needed from a different cluster. The primary disadvantages of a clustered architecture are the complexity of the inter-cluster control logic and the additional area required to achieve performance similar to a centralized architecture.

5.1.2 Duplicated Register File

In SMT microprocessor, access time of register file is crucial part in instruction latency. It will increase as the size and ports of register file increase.

In [5.8], a new kind of Duplicated register file architecture is proposed for embedded SMT microprocessor. The Duplicated register file architecture distributes read ports to each local function unit, which reduce access time by reducing read ports of each Duplicated register file. Each copy of Duplicated register file has the same size, the same number of ports and the same contents. Each function unit writes its results to all Duplicated register files simultaneously and does not need to synchronize the different Duplicated register files.

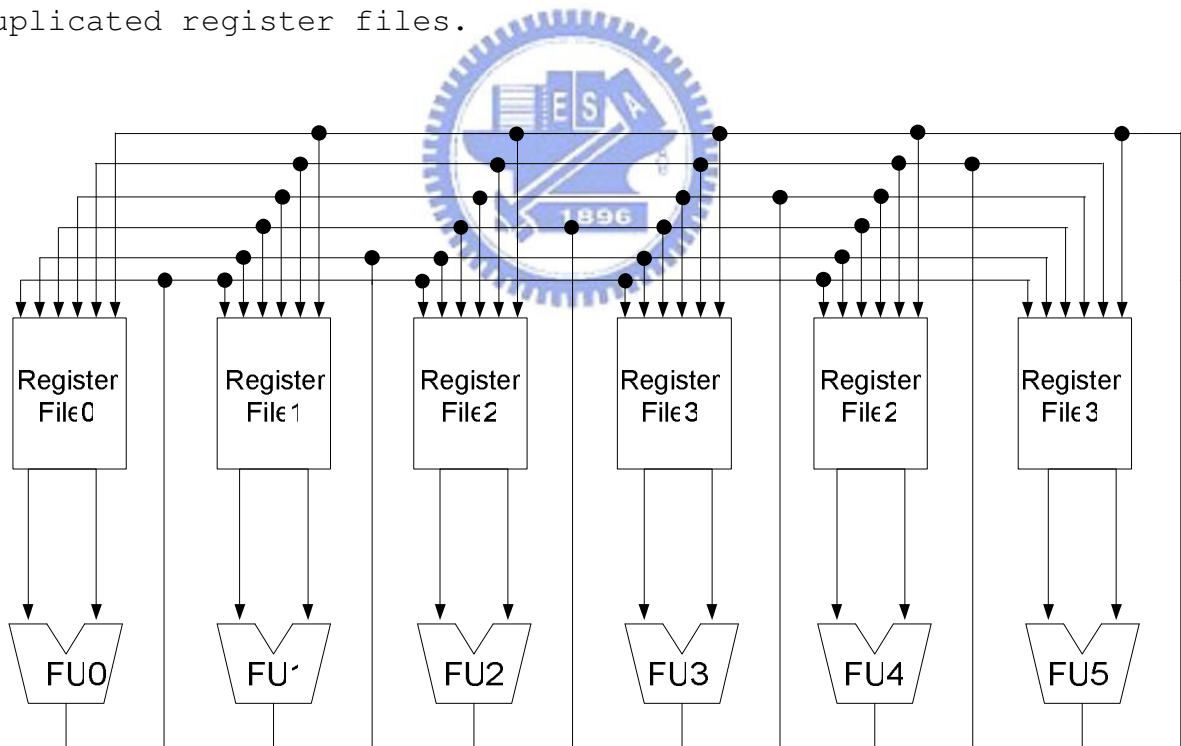


Fig. 5.2 4-thread, 2-read, 6-write, full-duplicate register file architecture.

As a result, it does not need communication between different clusters if some function unit tries to use value generated by other

function units. So, this kind of Duplicated register file architecture has dual functions: storage and communication.

Fig. 5.2 shows 6-duplicate (full-duplicate) register file architecture. Total area of all is larger than a central register file, but the access time become lesser.

The access time of Duplicated register files become lesser. However, total area and power consumption of all Duplicated register files is larger than a central register file.

5.1.3 Multilevel Register File

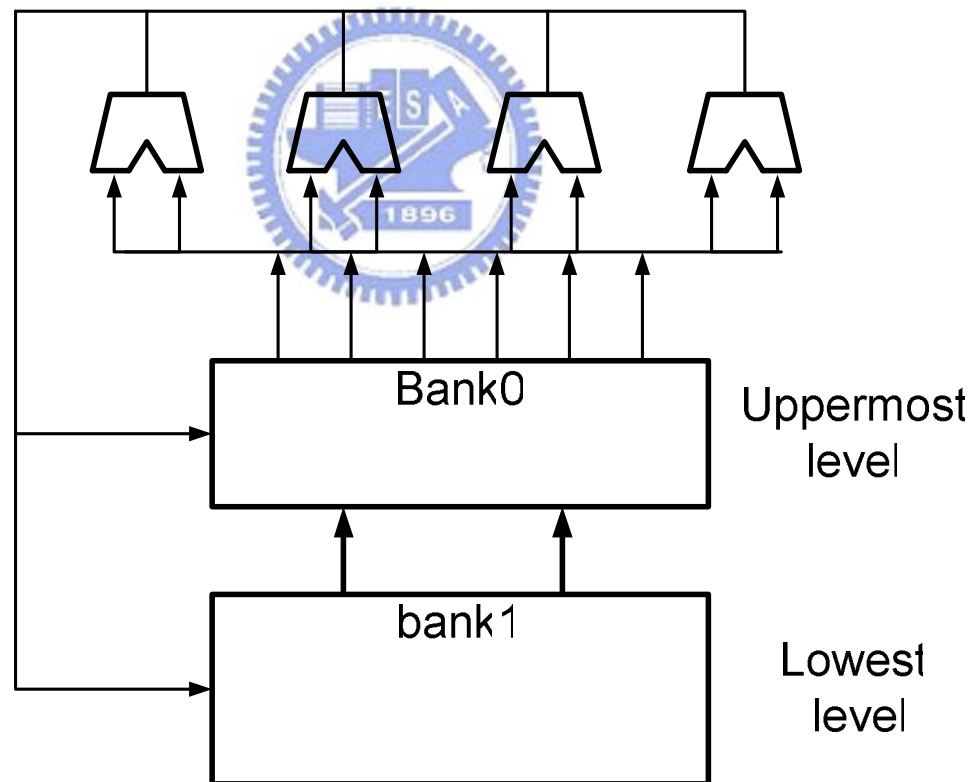


Fig. 5.3 Multilevel register file (register file cache)

Registers are cached to reduce average access latency in [5.9], [5.10]. A processor needs many physical registers. However, a very small number are actually required from a register file at a given moment.

A multilevel register file architecture consists of several levels of physical registers with a heterogeneous organization. Each level may have a different number of registers, a different number of ports and a different access time.

In a multi-level organization, the functional units can only obtain the source operands from the uppermost level directly. A subset of registers in the lower levels are cached in the upper levels depending on the expectations of being required in the near future. Results are always written to the lowest level, which contains all the values, and optionally to upper levels if they are expected to be useful in the near future.

A bank at the upper level of a register file cache can have many ports but few registers, which may result in a single-cycle access time. Banks at the lower levels have many more registers, a somewhat lower number of ports, and may have an increased latency. A more aggressive fetching mechanism could prefetch the values before they are required. Like in cache memories, prefetching must be carefully implemented to prevent premature or unnecessary fetching from polluting the upper levels. In general, prefetching can be implemented by software or hardware schemes.

It is a critical issue for the approach to deciding which values are cached in the upper level of the hierarchy. Like in cache memories, upper levels should contain those values that are more likely to be accessed in the near future. However, the locality properties of registers and memory are very different. First of all, registers have a much lower temporal re-use. In fact, most

physical registers are read only once, and there is even a significant percentage that are never read. Spatial locality is also rare, since physical register allocation and register references are not correlated at all.

Register caches have much worse locality than conventional data caches. Therefore, register caching can add considerable control complexity to an architecture and determining the appropriate values to cache is nontrivial.

5.1.4 One-Level Less-Port register file Architecture

Using a less-ported structure and only allowing necessary register file read accesses reduce the register file's area, energy, and access time. The designs in [5.11], [5.12], [5.13] do not use banked reads to avoid increasing the complexity of the select logic.

[5.12] propose two techniques to reduce the number of register ports without impacting performance. First, a small memory structure is added, the delayed write-back queue. To access the write-back queue instead of accessing the register file can reduce the access frequency of register file. In addition, the results is written back both in the register file and the write-back queue concurrently to avoid consistency problems during renaming.

Second, it proposed the technique to reduce the number of read ports by pre-fetching ready operands employs an operand pre-fetch buffer to store the pre-fetched operands, and a status bit, the pre-fetch flag, in the instruction queue entry to specify whether the operand is in the pre-fetch buffer or the register file.

There are two options for reducing demand for read ports in [5.11]. The first option is straightforward and identifies bypass operands

in an extra pipeline stage inserted between out-of-order issue and register read. Second, a novel technique, bypass hint, is proposed.

However, the select logic still has to select no more instructions than the number of available read ports after considering the bypass hint bits [5.11] or the prefetch flags [5.12]. [5.13] presents a novel register file architecture, which has single ported cells and asymmetric interfaces to the memory and to the datapath.

A high number of ports has a negative impact on the energy efficiency of register files. Traditionally, this problem is addressed through various clustering techniques that partition (or bank) the RF. However, as partitions get smaller the cost of inter-cluster copies quickly grows and the resulting register files are still multi-ported. For high energy efficiency, it is preferable that the registers be single ported.

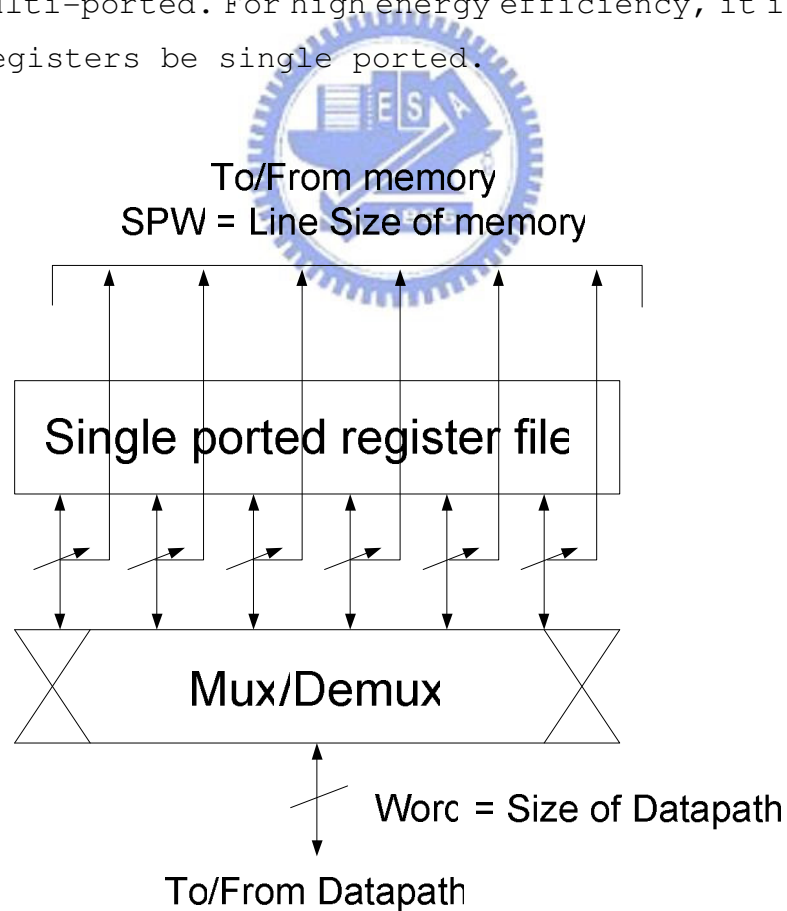


Fig. 5.4. Very Wide Register Organization

By making wide memories, related blocks of data can be loaded in parallel, thereby reducing the decoder overhead. This requires the bus between the memories and the register file to be wide as well.

Three aspects are important in the proposed organization: the interface to the memory, single ported cells and the interface to the datapath. The interface of this foreground memory organization is asymmetric: wide towards the memory and narrower towards the datapath.

A set of Very Wide Registers (VWR), with a single port each is used to replace a traditional register file. Every single VWR is made of single ported cells and it has no pre-decode circuit. A post-decode circuit consisting of a multiplexer is provided to select the appropriate word(s).

The asymmetric interface of the VWR, having a wide connection to the memory (width is complete row of the scratchpad) and a narrow connection of one word wide to the datapath, results in the following mode of operation: a complete row of the scratchpad is copied to the VWR at once, using a LOAD row., this scheme can save a lot of power in compared to a clustered VLIW register file.

5.2 Multithreading

Servers equipped with more powerful and power-hungry processors to meet higher computational demands are pushing the power and cooling capabilities of these datacenters to their limits, resulting in increased operating costs and decreased system reliability. Therefore, achieving high performance while maintaining existing power and thermal envelopes requires that microprocessor designs focus not only on performance but rather on the aggregate performance per watt.

Multithreading allows multiple threads to share the functional units of a single processor in an overlapping fashion. To permit this sharing, the processor must duplicate the independent state of each thread. For example, a separate copy of the register file, a separate PC, and a separate page table are required for each thread. The memory itself can be shared through the virtual memory mechanisms, which already support multiprogramming. In addition, the hardware must support the ability to change to a different thread relatively quickly; in particular, a thread switch should be much more efficient than a process switch, which typically requires hundreds to thousands of processor cycles.

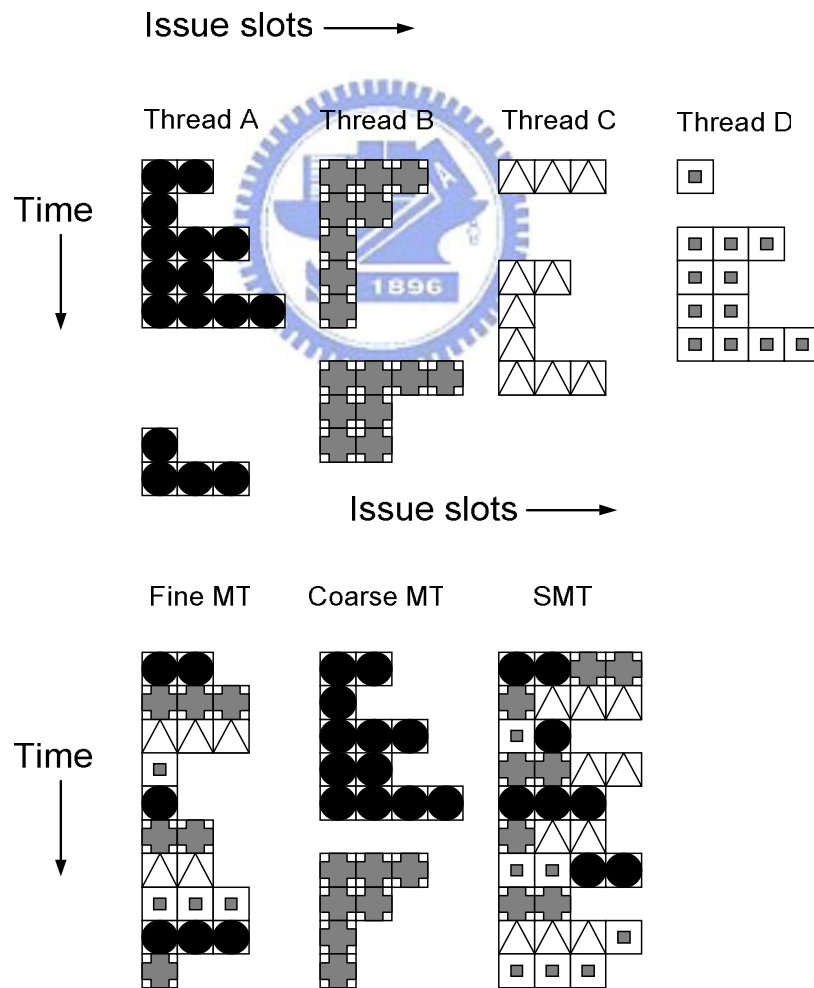


Fig. 5.5 How four threads use the issue slots of a superscalar processor in different approaches.

The top portion of Fig. 5.5 shows how four threads would execute independently on a superscalar with no multithreading support. In the superscalar without multithreading support, the use of issue slots is limited by a lack of instruction-level parallelism. In addition, a major stall, such as an instruction cache miss, can leave the entire processor idle. The bottom of Fig. 5.5 shows the three multithreaded categories including of Fine-grained, Coarse-grained, and Simultaneous multithreading.

5.2.1 Fine-grained multithreading

Fine-grained multithreading switches between threads on each instruction, resulting in interleaved execution of multiple threads. This interleaving is often done in a round-robin fashion, skipping any threads that are stalled at that time. To make fine-grained multithreading practical, the processor must be able to switch threads on every clock cycle.

One key advantage of fine-grained multithreading is that it can hide the throughput losses that arise from both short and long stalls, since instructions from other threads can be executed when one thread stalls. The primary disadvantage of fine grained multithreading is that it slows down the execution of the individual threads, since a thread that is ready to execute without stalls will be delayed by instructions from other threads.

In the fine-grained case, the interleaving of threads eliminates fully empty slots. Because only one thread issues instructions in a given clock cycle, however, instruction-level parallelism limitations still lead to a significant number of idle slots within individual clock cycles.

5.2.2 Coarse-grained multithreading

Coarse-grained multithreading was invented as an alternative to fine-grained multithreading. Coarse-grained multithreading switches threads only on costly stalls, such as level 2 cache misses. This change relieves the need to have thread switching be essentially free and is much less likely to slow down the execution of an individual thread, since instructions from other threads will only be issued when a thread encounters a costly stall.

Coarse-grained multithreading suffers, however, from a major drawback: It is limited in its ability to overcome throughput losses, especially from shorter stalls. This limitation arises from the pipeline start-up costs of coarse-grained multithreading. Because a CPU with coarse grained multithreading issues instructions from a single thread, when a stall occurs, the pipeline must be emptied or frozen. The new thread that begins executing after the stall must fill the pipeline before instructions will be able to complete. Because of this start-up overhead, coarse-grained multithreading is much more useful for reducing the penalty of high-cost stalls, where pipeline refill is negligible compared to the stall time.

In the coarse-grained multithreaded superscalar, the long stalls are partially hidden by switching to another thread that uses the resources of the processor. Although this reduces the number of completely idle clock cycles, within each clock cycle, the instruction-level parallelism limitations still lead to idle cycles. Furthermore, in a coarse-grained multithreaded processor, since thread switching only occurs when there is a stall and the new thread has a start-up period, there are likely to be some fully idle cycles remaining.

5.2.3 Simultaneous multithreading

Simultaneous multithreading (SMT) is a variation on multithreading that uses the resources of a multiple-issue, dynamically scheduled processor to exploit thread-level parallelism at the same time it exploits instruction-level parallelism.

The key insight that motivates SMT is that modern multiple-issue processors often have more functional unit parallelism available than a single thread can effectively use. Furthermore, with register renaming and dynamic scheduling, multiple instructions from independent threads can be issued without regard to the dependences among them; the resolution of the dependences can be handled by the dynamic scheduling capability.

In the SMT case, thread-level parallelism (TLP) and instruction-level parallelism (ILP) are exploited simultaneously, with multiple threads using the issue slots in a single clock cycle. Ideally, the issue slot usage is limited by imbalances in the resource needs and resource availability over multiple threads. In practice, other factors—including how many active threads are considered, finite limitations on buffers, the ability to fetch enough instructions from multiple threads, and practical limitations of what instruction combinations can issue from one thread and from multiple threads—can also restrict how many slots are used. Although Fig. 5.5 greatly simplifies the real operation of these processors, it does illustrate the potential performance advantages of multithreading in general and SMT in particular.

As mentioned earlier, simultaneous multithreading uses the insight that a dynamically scheduled processor already has many of the hardware mechanisms needed to support the integrated exploitation of TLP through multithreading. In particular, dynamically scheduled superscalar processors have a large set of registers that can be used to hold the register sets of independent

threads (assuming separate renaming tables are kept for each thread).

Because register renaming provides unique register identifiers, instructions from multiple threads can be mixed in the data path without confusing sources and destinations across the threads. This observation leads to the insight that multithreading can be built on top of an out-of-order processor by adding a per-thread renaming table, keeping separate PCs, and providing the capability for instructions from multiple threads to commit. There are complications in handling instruction commit, since we would like instructions from independent threads to be able to commit independently. The independent commitment of instructions from separate threads can be supported by logically keeping a separate reorder buffer for each thread.

There is a variety of other design challenges for an SMT processor. First, dealing with a larger register file needed to hold multiple contexts. Second, maintaining low overhead on the clock cycle, particularly in critical steps such as instruction issue, where more candidate instructions need to be considered, and in instruction completion, where choosing what instructions to commit may be challenging. Third, ensuring that the cache conflicts generated by the simultaneous execution of multiple threads do not cause significant performance degradation.

In viewing these problems, two observations are important. First, in many cases, the potential performance overhead due to multithreading is small, and simple choices work well enough. Second, the efficiency of current super scalars is low enough that there is room for significant improvement, even at the cost of some overhead. SMT appears to be the most promising way to achieve that improvement in throughput.

5.3 Multiprocessors

Computer performance has been driven largely by decreasing the size of chips while increasing the number of transistors they contain. In accordance with Moore's law, this has caused chip speeds to rise and prices to drop. This ongoing trend has driven much of the computing industry for years.

However, transistors can't shrink forever. Even now, as transistor components grow thinner, chip manufacturers have struggled to cap power usage and heat generation, two critical problems. Even performance-enhancing approaches like running multiple instructions per thread have bottomed out.

For these reasons, processor performance increases have begun slowing. Chip performance increased 60 percent per year in the 1990s but slowed to 40 percent per year from 2000 to 2004, when performance increased by only 20 percent.

Manufacturers are building chips with multiple cooler-running, more energy-efficient processing cores instead of one increasingly powerful core. The multicore chips don't necessarily run as fast as the highest performing single-core models, but they improve overall performance by handling more work in parallel.

Current transistor technology limits the ability to continue making single processor cores more powerful. For example, as a transistor gets smaller, the gate, which switches the electricity on and off, gets thinner and less able to block the flow of electrons.

Thus, small transistors tend to use electricity all the time, even when they aren't switching. This wastes power. Also, increasing clock speeds causes transistors to switch faster and thus generate more heat and consume more power. However, this approach can't keep

pace with processors' increasing power and heat build up.

These and other challenges have hurt manufacturers' plans for new, faster single-core processors. For example, Intel cancelled two next-generation Pentium 4 processors last year, noted Jeff Austin, the company's desktop product manager. Intel also postponed and then cancelled a 4-GHz, current generation Pentium. And IBM could build so few of its G5 chips that Apple Computer had to delay last year's introduction of its new iMac G5 desktop, which uses the processor.

Commercial multiprocessors and clusters usually define high performance as high throughput for independent tasks. This definition is in contrast to running a single task on multiple processors. The term parallel processing program is used to refer to a single program that runs on multiple processors simultaneously.

5.3.1 Multicore architecture and communication

To parallel processors share data, processors with a shared-memory offer the programmer a single memory address space that all processors share, as shown in Fig. 5.6. Processors communicate through shared variables in memory, with all processors capable of accessing any memory location via loads and stores.

As processors operating in parallel will normally share data, they also need to coordinate when operating on shared data; otherwise, one processor could start working on data before another is finished with it. This coordination is called synchronization. When sharing is supported with a single address space, there must be a separate mechanism for synchronization. One approach uses a lock. Only one processor at a time can acquire the lock, and other processors interested in shared data must wait until the original processor unlocks the variable.

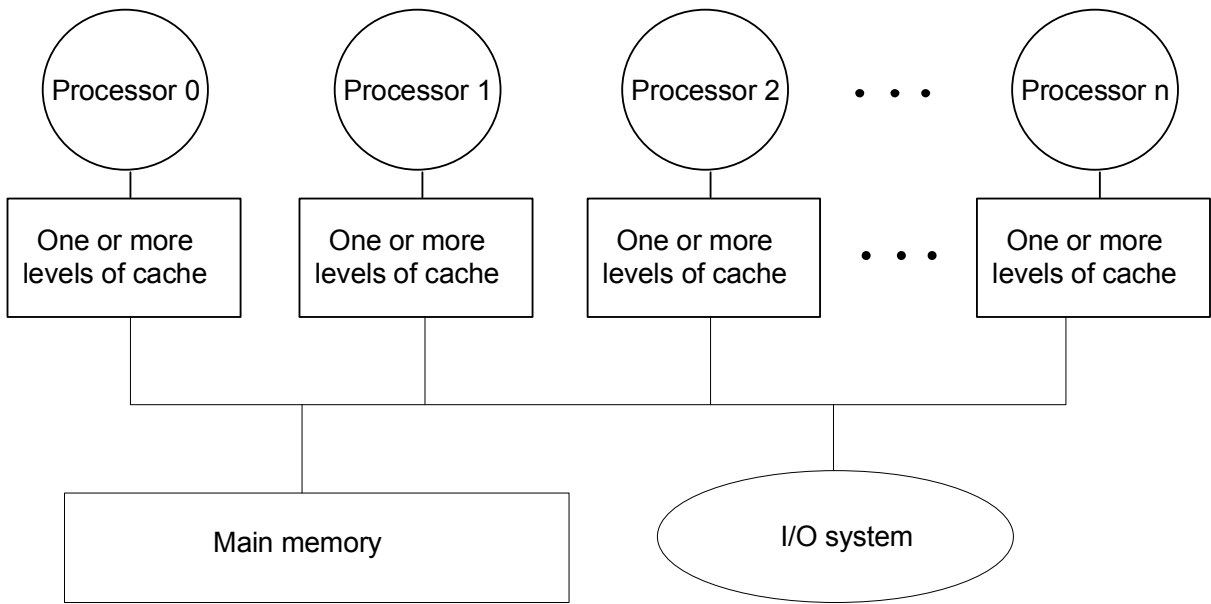


Fig. 5.6 The structure of a centralized shared-memory multiprocessor.

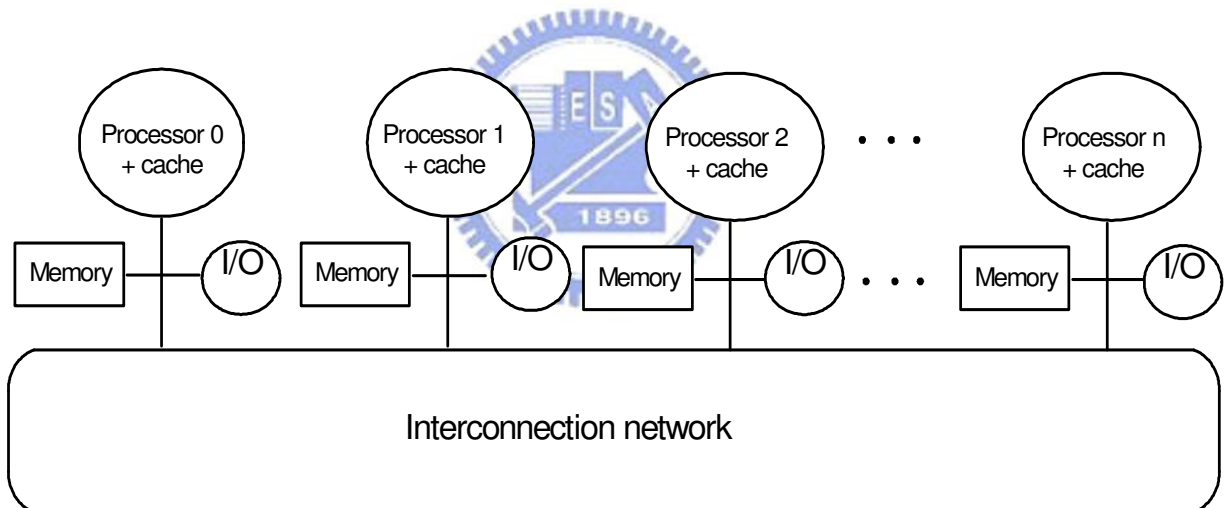


Fig. 5.7 The structure of a distributed-memory multiprocessor.

Single address space multiprocessors come in two styles. The first takes the same time to access main memory no matter which processor requests it and no matter which word is requested. Such machines are called uniform memory access (UMA) multiprocessors or symmetric multiprocessors (SMP). In the second style, some memory accesses are faster than others depending on which processor asks for which

word. Such machines are called nonuniform memory access (NUMA) multiprocessors. As you might expect, the programming challenges are different for a NUMA multiprocessor versus a UMA multiprocessor, but NUMA machines can scale to larger sizes and hence are potentially higher performance.

The alternative model for communicating uses message passing for communicating among processors. Message passing is required for machines with private memories, in contrast to shared memory. One example is a distributed-memory scheme, as shown in Fig. 5.7. The processors in different desktop computers communicate by passing messages over a local area network. Provided the system has routines to send and receive messages, coordination is built in with message passing since one processor knows when a message is sent, and the receiving processor knows when a message arrives. The receiving processor can then send a message back to the sender saying the message has arrived if the sender needs that confirmation.



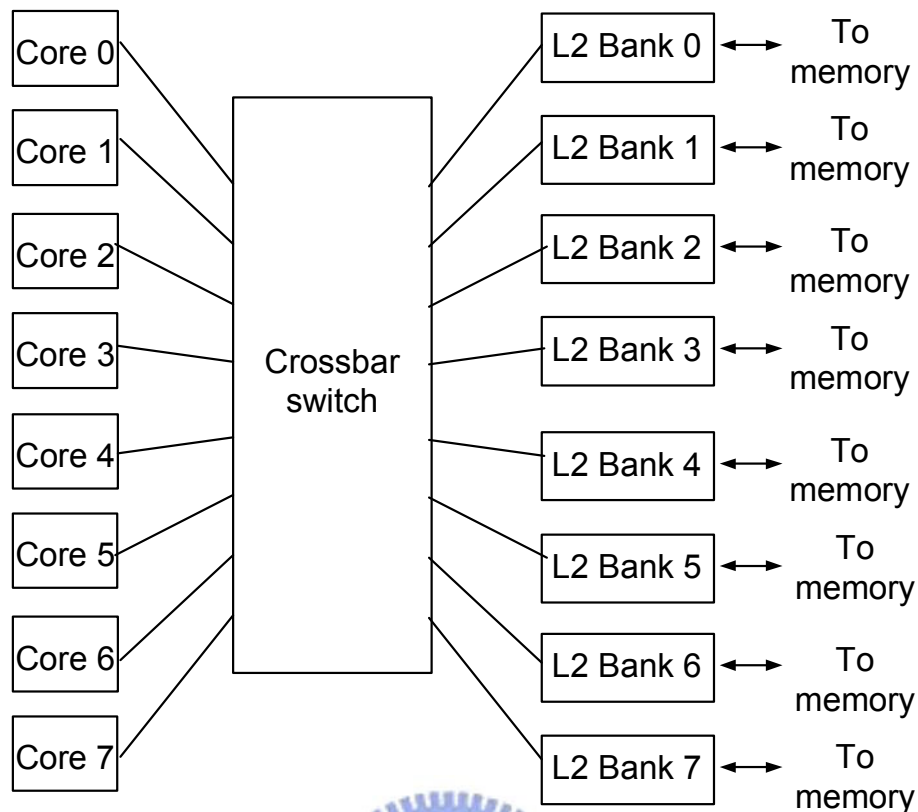


Fig. 5.8 Processor block diagram of Niagara2 SPARC processor

The Niagara2 from Sun Microsystems is the processor based on the multi-threading architecture [5.14]. The chip has eight SPARC Cores, a 4 MB shared Level2 cache, and supports concurrent execution of 64 threads. The Level2 cache is divided into eight banks. The SPARC Cores communicate with the Level2 cache through a crossbar. The block diagram of Niagara2 SPARC processor is shown in Fig. 5.8.

Fig. 5.9 shows the Niagara2 Crossbar (CCX) which serves as a high bandwidth interface between the eight SPARC Cores and the eight L2 cache banks, and the non-cacheable unit (NCU). CCX consists of two blocks: PCX and CPX. PCX (Processor-to-Cache -Transfer) is a multiplexer which transfers data from the eight SPARC cores to the eight L2 cache banks and the NCU. CPX (Cache-to-Processor Transfer) transfers data in the reverse direction. The PCX and CPX combined provide a Read/Write bandwidth of 270 GB/s. All crossbar data transfer requests are processed using a four-stage pipeline. The pipeline stages are: Request, Arbitration, Selection, and Transmission.

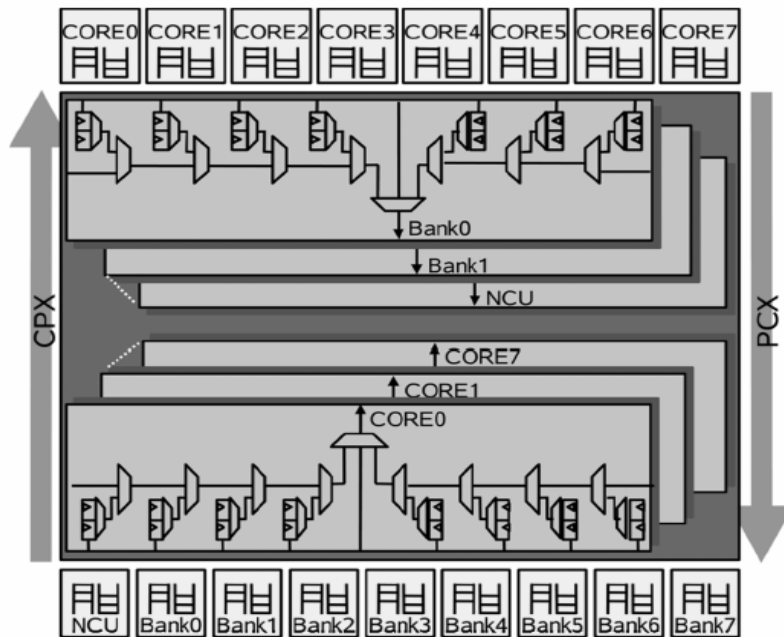


Fig. 5.9 The Niagara2 Crossbar



5.4 Conclusion

There are several register file organizations according to different system architecture. Different architectures of register file are discussed in this chapter. Clustered architecture can decrease port number but need complex control logic. Register file cache reduces access latency. However, Register caches have much worse locality than conventional data caches. Therefore, register caching can add considerable control complexity to architecture.

To increase the utility rate of function units and to achieve high performance while maintaining existing power, the concept of multithreading and multiprocessors is introduced.

Chapter 6

Conclusion

To reduce power consumption is an important topic for discussion when more and more portable devices are desired. Register file represent a substantial portion of power and area in modern processors. However, the conventional design of a register file with fully-ports causes problems such as enlargement of chip size, high power consumption and deterioration of register access speed. To avoid the serious problem, a low power banking multithread register file implemented by four interleaved banks with lesser ports is presented. Timing sharing access scheme is used to ease the performance degraded by bank conflict.

We use additional address bit as thread number to switch between two threads. This method would increase some area penalty of decoders, but impact performance slightly.

In order to design this low power multiple port register file, several low power techniques are proposed, such as floating bitline architecture, and divide bitline architecture. The register file can operate correctly in wide range of voltage supply from 0.5v to 1.0v. To operate correctly under all process corners and the wide range of Vdd, the design of timing control circuit is very important.

The dual thread 64 x 64 bits register file implemented in UMC 90um CMOS technology consumes around 215.51 μ W to 197.77 μ W at 50MHz with 0.5v and consumes around 3.62mW to 3.04mW at 250MHz with 1.0v.

In the future, this work can use dual-Vdd technology, using Higher Vdd in critical path to better performance and using lower Vdd in

non-critical path to save power. The lowest supply voltage of this work is 0.5v, a subthreshold register file can be implemented hereafter for ultra low power application. A power management circuit can also be added to this work.



Bibliography

Reference of Chapter 1

- [1.1] Jessica H. Tseng, and Krste Asanovic, “A Speculative Control Scheme for an Energy-Efficient Banked Register File ” , IEEE Transactions on Computers, Vol. 54, No. 6, pp. 741 - 751,June 2005.
- [1.2] Eric S. Fetzer, David Dahle, Casey Little, and Kevin Safford,” The Parity Protected, Multithreaded Register Files on the 90-nm Itanium Microprocessor,” J. Solid-State Circuits, vol. 41, no. 1, Jan. 2006.



Reference of Chapter 2

- [2.1] M. Horowitz et al., "Scaling, power, and the future of CMOS," in IEDM Tech. Dig., Dec. 2005, pp. 9–15.
- [2.2] C.-H. Jan et al., "A 65 nm ultra low power logic platform technology using uni-axial strained silicon transistors," in IEDM Tech. Dig., Dec. 2005, pp. 60–63.
- [2.3] S. Narendra, S. Borkar, V. De, D. Antoniadis, and A. Chandrakasan, "Scaling of stack effect and its application for leakage reduction," in Proc. IEEE Int. Symp. Low Power Electron. Des., 2001, pp. 192–200.
- [2.4] Jessica H. Tseng, and Krste Asanovic, "A Speculative Control Scheme for an Energy-Efficient Banked Register File" , IEEE Transactions on Computers, Vol. 54, No. 6, pp. 741 - 751, June 2005.
- [2.5] T. Ghani et al., "Scaling challenges and device design requirements for high performance sub-50 nm gate length planar CMOS transistors," in VLSI Symp. Tech. Dig., Jun. 2000, pp. 174–175.
- [2.6] Y. Taur and E. Nowak, "CMOS devices below 0.1 μm : How high will performance go?" in IEDM Tech. Dig., Dec. 1997, pp. 215–218.
- [2.7] A. Agarwal, L. Hai, and K. Roy, "A single-V low-leakage gated-ground cache for deep submicron," IEEE J. Solid-State Circuits, vol. 38, no. 2, pp. 319–328, Feb. 2003.
- [2.8] K. Kanda, K. Sadaaki, and T. Sakurai, "90% write power-saving SRAM using sense-amplifying memory cell," IEEE J. Solid State Circuits, vol. 39, no. 6, pp. 927–933, Jun. 2004.
- [2.9] K. Zhang, U. Bhattacharya, Z. Chen, F. Hamzaoglu, D. Murray, N. Vallepalli, Y. Wang, B. Zheng, and M. Bohr, "SRAM design on 65-nm CMOS technology with dynamic sleep transistor for leakage reduction," IEEE J. Solid State Circuits, vol. 40, no. 4, pp. 895–901, Apr. 2005.
- [2.10] K. Flautner, N. S. Kim, S. Martin, D. Blaauw, and T. Mudge, "Drowsy caches: Simple techniques for reducing leakage power," in Proc. 29th IEEE Ann. Int. Symp. Comput. Arch., 2002, pp. 148–157.
- [2.11] H. Qin, Y. Cao, D. Markovic, A. Vladimirescu, and J. Rabaey, "SRAM leakage suppression by minimizing standby supply voltage," in Proc. 5th IEEE Int. Symp. Quality Electron. Des., 2004, pp. 55–60.
- [2.12] K. Nii, Y. Tsukamoto, T. Yoshizawa, S. Imaoka, Y. Yamagami, T. Suzuki, A. Shibayama, H. Makino, and S. Iwade, "A 90-nm low-power 32-kB embedded SRAM

- with gate leakage suppression circuit for mobile applications,” *IEEE J. Solid-State Circuits*, vol. 39, no. 4, pp.684–693, Apr. 2004.
- [2.13] K. Kanda, T. Miyazaki, M. K. Sik, H. Kawaguchi, and T. Sakurai, “Two orders of magnitude leakage power reduction of low voltage SRAM’s by row-by-row dynamic V control (RRVD) scheme scheme,” in *Proc. 15th IEEE Ann. Int. ASIC/SOC Conf.*, 2002, pp. 381–385.
- [2.14] H. Kawaguchi, Y. Itaka, and T. Sakurai, “Dynamic leakage cut-off scheme for low-voltage SRAM’s,” in *Dig. Tech. Papers IEEE Symp. VLSI Circuits*, 1998, pp. 140–141.
- [2.15] C. H. Kim and K. Roy, “Dynamic SRAM leakage tolerant cache memory for low voltage microprocessors,” in *Proc. IEEE Int. Symp.Low Power Electron. Des.*, 2002, pp. 251–254.
- [2.16] K. Nii, H. Makino, Y. Tujihashi, C. Morishima, Y. Hayakawa, H.Nunogami, T. Arakawa, and H. Hamano, “A low power SRAM using auto-backgate-controlled MT-CMOS,” in *Proc. IEEE Int. Symp. Low Power Electron. Des.*, 1998, pp. 293–297.
- [2.17] S. Heo, K. Barr, M. Hampton, and K. Asanovic, “Dynamic fine-grain leakage reduction using leakage-biased bitlines,” in *Proc. 29th IEEE Ann. Int. Symp. Comput. Arch.*, 2002, pp. 137–147.
- [2.18] K. Itoh, A. R. Fridi, A. Bellaouar, and M. I. Elmasry, “A deep sub-V, single power-supply SRAM cell with multi-V_T, boosted storage node and dynamic load,” in *Dig. Tech. Papers IEEE Symp. VLSI Circuits*, 1996, pp. 132–133.
- [2.19] A. Agarwal, H. Li, and K. Roy, “A single-V_t low-leakage gated-ground cache for deep submicron,” *IEEE J. Solid-State Circuits*, vol. 38, no. 2, pp. 319–328, Feb. 2003.
- [2.20] A. Bhavnagarwala, S. V. Kosonocky, M. Immediato, D. Knebel, and A.-M. Haen, “A pico-joule class, 1 GHz, 32 kB × 64 b DSP SRAM with self reverse bias,” in *Proc. Symp. VLSI Circuits Dig. Tech. Papers*, Jun. 2003, pp. 251–252.
- [2.21] K. Zhang et al., “SRAM design on 65-nm CMOS technology with dynamic sleep transistor for leakage reduction,” *IEEE J. Solid-State Circuits*, vol. 40, no. 4, pp. 895–901, Apr. 2005.
- [2.22] M. Khellah et al., “A 256-Kb dual-VCC SRAM building block in 65-nm CMOS process with actively clamped sleep transistor,” *IEEE J. Solid- State Circuits*, vol. 42, no. 1, pp. 233–242, Jan. 2007.
- [2.23] M. Yamaoka, K. Osada, and K. Ishibashi, “0.4-V logic-library- friendly SRAM array using rectangular-diffusion cell and delta-boosted-array voltage scheme,” *IEEE J.*

- Solid-State Circuits, vol. 39, no. 6, pp. 934–940, Jun. 2004.
- [2.24] K. Zhang et al., “A 3-GHz 70-Mb SRAM in 65-nm CMOS technology with integrated column-based dynamic power supply,” *IEEE J. Solid-State Circuits*, vol. 41, no. 1, pp. 146–151, Jan. 2006.
- [2.25] M. Yamaoka, K. Osada, K. Itoh, R. Tsuchiya, and T. Kawahara, “Dynamic-Vt, dual-power-supply SRAM cell using D2G-SOI for low-power SoC application,” in *Proc. IEEE Int. SOI Conf.*, Oct. 2004, pp. 109–111.
- [2.26] M. Khellah et al., “A 256-Kb dual-VCC SRAM building block in 65-nm CMOS process with actively clamped sleep transistor,” *IEEE J. Solid-State Circuits*, vol. 42, no. 1, pp. 233–242, Jan. 2007.
- [2.27] R. Balasubramonian, S. Dwarkadas, and D.H. Albonese, “Reducing the Complexity of the Register File in Dynamic Superscalar Processors,” *Proc. 34th Ann. IEEE/ACM Int’l Symp. Microarchitecture (MICRO-34)*, Dec. 2001.
- [2.28] S. Wallace and N. Bagherzadeh, “A Scalable Register File Architecture for Dynamically Scheduled Processors,” *Proc. Int’l Conf. Parallel Architectures and Compilation (PACT)*, Oct. 1996.
- [2.29] E. Borch, E. Tune, S. Manne, and J.S. Emer, “Loose Loops Sink Chips,” *Proc. High Performance Computer Architecture (HPCA)*, pp. 299–310, Feb. 2002.
- [2.30] Tadashi Saito, et al, “Design of superscalar processor with multi-bank register file,” *ISCAS*, Vol.4, 2005.
- [2.31] T. Saito, M. Maeda, T. Hironaka, K. Tanigawa, T. Sueyoshi, K. Aoyama, T. Koide, H.J. Mattausch, “Design of superscalar processor with multi-bank register file” *ISCAS*, vol. 4 pp.3507 – 3510, 2005.
- [2.32] T. Monreal, V. Vinals, J. Gonzalez, A. Gonzalez, M. Valero, “Late allocation and early release of physical registers,” *IEEE Transactions on Computers*, Vol. 53, no. 10, pp.1244–1259, Oct. 2004.
- [2.33] Rakesh Nalluri, Rohan Garg, Preeti Ranjan Panda, “Customization of Register File Banking Architecture for Low Power,” *20th VLSID*, pp. 239–244, 2007.
- [2.34] Shuai Wang, Hongyan Yang, Jie Hu, Sotirios G. Ziavras, “Asymmetrically banked value-aware register files for low-energy and high-performance,” *Microprocessors & Microsystems*, Vol. 32, no. 3, pp. 171–182, 2008.
- [2.35] Kevin Zhang, Fatih Hamzaoglu, “Low-Power SRAMs in Nanoscale CMOS Technologies,” *IEEE Transactions on Electron Devices*, vol. 55, no. 1, Jan 2008.
- [2.36] Fabio Frustaci, Pasquale Corsonello, Stefania Perri, and Giuseppe Cocorullo,

“Techniques for Leakage Energy Reduction in Deep Submicrometer Cache Memories,” IEEE Transactions on Very Large Scale Integration (VLSI) SYSTEMS, Vol. 14, No. 11, Nov. 2006.

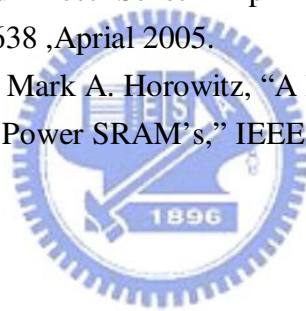


Reference of Chapter 3

- [3.1] Eric S. Fetzner, David Dahle, Casey Little, and Kevin Safford,” The Parity Protected, Multithreaded Register Files on the 90-nm Itanium Microprocessor,” J. Solid-State Circuits, vol. 41, no. 1, Jan. 2006.
- [3.2] Benton Highsmith Calhoun,, and Anantha P. Chandrakasan,” A 256-kb 65-nm Sub-threshold SRAM Design for Ultra-Low-Voltage Operation,” J. Solid-State Circuits, Vol. 42, no. 3, March 2007.
- [3.3] Jinhui Chen, Lawrence T. Clark, and Tai-Hua Chen,” An Ultra-Low-Power Memory With a Subthreshold Power Supply Voltage,” J. Solid-Sate Circuits, Vol. 41, no. 10, Oct 2006.
- [3.4] Fabio Frustaci, Pasquale Corsonello, Stefania Perri, and Giuseppe Cocorullo, “Techniques for Leakage Energy Reduction in Deep Submicrometer Cache Memories,” IEEE Transactions on Very Large Scale Integration (VLSI) SYSTEMS, Vol. 14, No. 11, Nov. 2006.
- [3.5] Terence B. Hook, Matt Breitwisch, Jeff Brown, P. Cottrell,Dennis Hoyniak, Chung Lam, and Randy Mann, “Noise Margin and Leakage in Ultra-Low Leakage SRAM Cell Design,” IEEE Transactions on Electron Devices, Vol. 49, no. 8, August 2002.
- [3.6] EVERT SEEVINCK, FRANS J. LIST, AND JAN LOHSTROH,” Static-Noise Margin Analysis of MOS SRAM Cells,” J. Solid-Sate Circuits, vol. SC-22, no. 5, Oct 1987.
- [3.7] Evelyn Grossar, Michele Stucchi, Karen Maex, Member, and Wim Dehaene,” Read Stability and Write-Ability Analysis of SRAM Cells for Nanometer Technologies,” J. Solid-Sate Circuits, Vol. 41, no. 11, Nov. 2006.
- [3.8] Satoshi Ishikura, et al, “A 45 nm 2-port 8T-SRAM Using Hierarchical Replica Bitline Technique With Immunity From Simultaneous R/W Access Issues,” J. Solid-Sate Circuits, Vol. 43, no. 4, Apr. 2008.
- [3.9] Kanak Agarwal, and Sani Nassif,” The Impact of Random Device Variation on SRAM Cell Stability in Sub-90-nm CMOS Technologies,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 16, No. 1, Jan. 2008.
- [3.10] Stefan Cosemans, Wim Dehaene, and Francky Catthoor,” A Low-Power Embedded SRAM for Wireless Applications,” J. Solid-Sate Circuits, Vol. 42, No. 7, July 2007.

Reference of Chapter 4

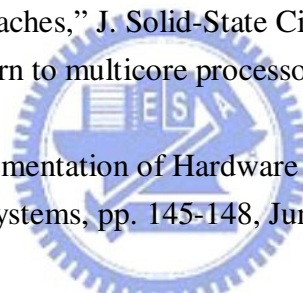
- [4.1] Shirrang K. Karandikar and Sachin S. Sapatnekar, "Technology Mapping Using Logical Effort for Solving the Load-Distribution Problem" IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 27, no. 1, pp.45-58, January 2008.
- [4.2] A. Kabbani, D. Al-Khalili, and A. J. Al-Khalili, "Logical Path Delay Distribution And Transistor Sizing," IEEE-NEWCAS Conference, pp. 391 - 394, June 2005.
- [4.3] Benoit Lasbougues, Sylvain Engels, Robin Wilson, "Logical Effort Model Extension to Propagation Delay Representation," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 25, no. 9, SEPTEMBER 2006.
- [4.4] Toshikazu SUZUKI, Yoshinobu YAMAGAMI, Ichiro HATANAKA, Akinori SHIBAYAMA, Hironori AKAMATSU, and Hiroyuki YAMAUCHI, "0.3–1.5V Embedded SRAM Core with Write-Replica Circuit Using Asymmetrical Memory Cell and Source-Level-Adjusted Direct- Sense-Amplifier", IEICE TRANS. ELECTRON., vol.E88–C, no.4, pp. 630-638, April 2005.
- [4.5] Bharadwaj S. Amrutur and Mark A. Horowitz, "A Replica Technique for Wordline and Sense Control in Low-Power SRAM's," IEEE J. Solid-State Circuits, vol. 33, no. 8, August 1998.



Reference of Chapter 5

- [5.1] R.E. Kessler, "The Alpha 21264 Microprocessor," IEEE Micro, vol. 19, no. 2, pp. 24-36, Mar./Apr. 1999.
- [5.2] R.P. Preston et al., "Design of an 8-Wide Superscalar RISC Microprocessor with Simultaneous Multithreading," ISSCC Digest and Visuals Supplement, Feb. 2002.
- [5.3] G. Sohi, S. Breach, and T.N. Vijaykumar, "Multiscalar Processors," Proc. 22nd Int'l Symp. Computer Architecture (ISCA-22), 1995.
- [5.4] S. Palacharla, N. Jouppi, and J.E. Smith, "Complexity- Effective Superscalar Processors," ISCA-24, pp. 206-218, June 1997.
- [5.5] K.I. Farkas, P. Chow, N.P. Jouppi, and Z.G. Vranesic, "The Multicluster Architecture: Reducing Cycle Time through Partitioning," Proc. 30th Ann. IEEE/ACM Int'l Symp. Microarchitecture (MICRO-30), pp. 149-159, 1997.
- [5.6] V.V. Zyuban and P.M. Kogge, "Inherently Lower-Power High- Performance Superscalar Architectures," IEEE Trans. Computers, vol. 50, no. 3, pp. 268-285, Mar. 2001.
- [5.7] A. Sez nec, E. Toullec, and O. Rochecouste, "Register Write Specialization Register Read Specialization: A Path to Complexity- Effective Wide-Issue Superscalar Processors," Proc. 35th Ann. IEEE/ACM Int'l Symp. Microarchitecture (MICRO-35), Nov. 2002.
- [5.8] Chengjie Zang, S. Imai, S. Kimura, "Duplicated register file design for embedded simultaneous multithreading microprocessor," ASIC, 2005. pp. 90-93, Oct. 2005.
- [5.9] J.L. Cruz, A. Gonzalez, M. Valero, and N.P. Topham, "Multiple- Banked Register File Architectures," Proc. Int'l Symp. Computer Architecture (ISCA-27), pp. 316-325, 2000.
- [5.10] E. Borch, E. Tune, S. Manne, and J.S. Emer, "Loose Loops Sink Chips," Proc. High Performance Computer Architecture (HPCA), pp. 299-310, Feb. 2002.
- [5.11] I. Park, M.D. Powell, and T.N. Vijaykumar, "Reducing Register Ports for Higher Speed and Lower Energy," Proc. 35th Ann. IEEE/ ACM Int'l Symp. Microarchitecture (MICRO-35), Nov. 2002.
- [5.12] N.S. Kim and T. Mudge, "Reducing Register Ports Using Delayed Write-Back Queues and Operand Pre-Fetch," Proc. 17th Ann. ACM Int'l Conf. Supercomputing (ICS), pp. 172-182, 2003.
- [5.13] p. Raghavan, A. Lambrechts, M. Jayapala, F. Catthoor, D. Verkest, H. Corporaal, "Very Wide Register: An Asymmetric Register File Organization for Low Power Embedded Processors," DATE '07, pp. 1-6, April 2007.
- [5.14] Umesh Gajanan Nawathe, Mahmudul Hassan, King C. Yen, Ashok Kumar, Aparna Ramachandran, and David Greenhill, "Implementation of an 8-Core, 64-Thread,

- Power-Efficient SPARC Server on a Chip,” J. Solid-State Circuits, vol. 43, no. 1, January 2008.
- [5.15] John L. Hennessy, David A. Patterson, Andrea C. Arpaci-Dusseau, Computer Architecture: A Quantitative Approach, 4th ed., Andrea C. Arpaci-Dusseau, Morgan Kaufmann, 2007.
- [5.16] David A. Patterson, John L. Hennessy, Computer Organization and Design: The Hardware/software Interface, 3rd ed., John L. Hennessy, Peter J. Ashenden, Elsevier/Morgan Kaufmann, 2004.
- [5.17] Chunqing Wu, Xiangquan Shi, Xuejun Yang, Jinshu Su, “The Impact of Parallel and Multithread Mechanism on Network Processor Performance,” Grid and Cooperative Computing (GCC), pp. 236-240, Oct. 2006.
- [5.18] Xiaoqi Yang, Qilong Zheng, Guoliang Chen, Shujuan Liu, Jun Luan, “Transactional Memory Execution for Parallel Multithread Programming without Lock,” PDCAT, pp. 209-216 , 2007.
- [5.19] Chang-Hyo Yu, Kyusik Chung, Donghyun Kim, and Lee-Sup Kim, ” An Energy-Efficient Mobile Vertex Processor With Multithread Expanded VLIW Architecture and Vertex Caches,” J. Solid-State Circuits, vol. 42, no. 10, Oct. 2007.
- [5.20] D. Geer, “Chip makers turn to multicore processors,” Computer, pp. 11-13, May 2005.
- [5.21] Naraig Manjikian, “Implementation of Hardware Multithreading in a Pipelined Processor,” Circuits and Systems, pp. 145-148, June 2006.



Vita

PERSONAL INFORMATION

Name: U-Chan Kuo

Birth Date: April. 23, 1984

Birth Place: Taipei, Taiwan, R.O.C.

Address: Department of Electronics Engineering
National Chiao Tung University
1001 Ta-Hsueh Road
Hsin-chu, Taiwan 30010, R.O.C.

E-Mail Address: kuo001.ee95g@nctu.edu.tw

EDUCATION

B.S. [2006] Department of Electronics Engineering, National Chung-Hsing University.

M.A.[2008] Institute of Electronics, National Chiao-Tung University.

