國 立 交 通 大 學

電機與控制工程學系

碩士論文

智能型 DMA 的 DSP 架構設計在雙核心上的應用

A Smart DMA-Based DSP Architecture for Dual-Core Application

研究生：　　郭昕展

指導教授：　　林進燈 教授

中華民國九十八年七月

智能型DMA的DSP架構設計
在雙核心上的應用

A Smart DMA-Based DSP Architecture for Dual-Core Application

研 究 生：郭昕展　　Student：Hsin-Chan Kuo

指導教授：林進燈　　Advisor : Dr. Chin-Teng Lin

國立交通大學

電機與控制工程學系

碩士論文

A Thesis

Submitted to Institute of Electrical and Control Engineering
College of Electrical Engineering
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in

Electrical and Control Engineering

July 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年七月

智能型DMA的DSP架構設計在雙核心上的應用

學生：郭昕展　　指導教授：林進燈 博士

國立交通大學電機與控制工程研究所

中文摘要

雙核心處理器雖具有平行處理的能力以達到較好的運算效能，卻會受限於記憶體資料傳輸上頻寬的限制。若透過處理器做資料傳輸，將變得非常沒有效率。而一般處理器的 DMA(Direct Memory Access) 雖能有效的利用記憶體頻寬用以減低處理器傳輸上的負擔，但無法提供特殊數位訊號處理功能。在現今的處理器已經開始重視 DMA 對數位訊號處理的能力設計，例如智原科技的 FTMCP020 以及 TI DSP 晶片等。針對數位訊號處理的一些特殊定址及運算，是傳統的處理器或 DMA 將不能發揮較好的效能，因此本論文將提出智慧型直接記憶體控制器的設計。

本論文提出一個智慧型直接記憶體存取（DMA），用以輔助雙核心處理器提升運算效能及傳輸效率。智慧型 DMA 控制器設計以傳統 DMA 傳輸模式設計加上支援五種定址模式，能夠有效選取傳輸資料區塊，降低傳輸的頻寬及處理器的負擔。本論文設計特色是具有（1）擁有內建 Dual-MAC 運算器搭配定址模式，可支援雙通道資料記憶體向量運算，協助處理器處理大量且具有規則與繁雜的數位訊號；（2）支援周邊輸出入匯流排，使得周邊擴充更有彈性；（3）減少約 75% 等待資料的時間；及（4）降低組語的程式碼。

本論文設計一個智慧型 DMA 控制器，並整合於一個通用雙核心處理器上，經實驗結果證明能大幅提升 FFT, DCT, FIR 等運算，特別是複數 FFT 運算。此晶片採用 UMC 90nm 製程，以 Cell-based 方式設計，晶片面積約 2.1x2.1 mm^2，預估最大操作頻率在 200MHz。

A Smart DMA-Based DSP Architecture for Dual-Core Application

Student：Hsin-Chan Kuo Advisor: Dr. Chin-Teng Lin

Department of Electrical and Control Engineering
National Chiao Tung University

Abstract

Although a dual-core processor has the ability of parallel processing and has a better performance, it is limited to memory bandwidth. If the processor is used as data transmission, it will become inefficiency. However, for a general-purpose processor DMA (direct memory access) is often used to improve the effective usage of memory bandwidth, but it can not offer special functions for digital signal processing. In recent years, the processor has been respected for DMA design in the ability of digital signal processing, such as Faraday's FTDMAC020 and TI's digital signal processors, etc. Because the traditional processor or DMA has not more efficiency at present, this thesis proposes a novel smart DMA controller design.

This thesis presents the SDMA controller in order to assist a dual-core processor improving performance and transmission efficiency. The SDMA supports five addressing modes compared with the design method of traditional DMA and four transmission types to select the region of valid data and to reduce the transmission bandwidth for the processor. The SDMA design has features as follows. (1) It has a built-in dual complex-valued multiplication-and-accumulation (Dual-MAC) to processes mass and regular data computation. Moreover, two channel can access two

memory banks and perform vector operations at the same time; (2) it supports the peripheral bus to expand I/O devices flexibly; (3) it can save about 75% time wasted on data transfer; and (4) the code size can be reduced.

This thesis proposes the smart DMA design is integrated into the dual-core architecture to be a DSP-like processor. By experimental results, the proposed design can achieve greatly efficiency at FFT, DCT, and FIR computation, especially in complex operations. The chip has been integrated in the total area of 2.1 × 2.1 mm^2 by using UMC 90nm CMOS technology and has fabricated via the National Chip Implementation Center (CIC). The maximum clock frequency is at 200MHz with a single 1.0V supply.

誌謝

兩年的研究所生涯隨著論文的完成劃上了句號，這兩年間，要感謝許多人的鼓勵和幫忙，使我獲得充實的專業能力並順利完成研究所的學業。

首先要感謝的是我的指導教授-林進燈老師。林老師是國內十分傑出的一位教授，在不同領域內都有相當好的研究成果。感謝老師提供了很理想的研究環境、豐富的資源及正確的引導，使我在研究上非常順利。在老師悉心的指導下，讓我學習到解決問題的能力及做研究應有的態度，使我獲益良多。

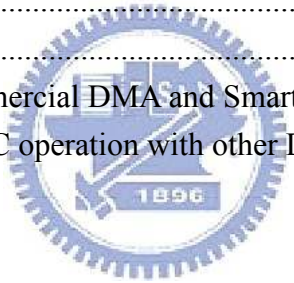
在實驗室裡，鍾仁峰博士給予我最直接的教導，不管遇到課業上或研究上的問題，常常去請教鍾仁峰博士，感謝鍾學長不厭其煩地教導，使我增進了對積體電路設計上的專業知識，開拓了我的視野。在學校裡，感謝范倫達教授時常關心我學業上的研究，時常與我討論論文方向及進度。還要特別感謝洪紹航學長對於我研究上的指導，幫解決了我許多的問題。另外也感謝實驗室所有的夥伴，毓廷、煒忠、建昇、孟修、寓鈞、舒愷、孟哲、俊彥、依伶等學長姐們。還有我的同學們，哲睿、介恩、家欣、有德。以及實驗室的學弟妹，感謝大家在研究上及生活上的互相扶持及鼓勵。

最後要感謝家人爸爸、媽媽、妹妹的支持，讓我能專心於學術上的研究，渡過所有難關，謝謝！人生值得感謝的人其實很多，感謝老天、感謝許多親人、朋友和同學，在生命的旅途中，因為有你們，因為我們彼此珍惜、相互扶持，才能有無比的力量。

CONTENTS

List of Figures

List of Tables



Chapter 1

Introduction

Media processing usually deals with large data streams of video, audio and graphics. In the media system, a large number of data should be transported. If the processor, which is good at data calculating, takes the charge of those data transporting, the performance of the media system will decrease greatly.

In the last century, the technology of direct memory access (DMA) was introduced into the DSP (Digital Signal Processor) design. The traditional DMA only supports increment/decrement addressing modes and four types of transferring modes [1], [2], [3]. Due to this fundamental mode, it could complete the data transporting among memories or peripherals without the aid of the processor. Most real-time scheduling algorithms demand that the worst-case execution time of each task is known in advanced. This is hardly satisfied if a task uses a DMA I/O method to transfer data between I/O devices and memories.

Several DMA controllers are proposed to support real-time systems. In 1988, B. Sprunt et al. [4] proposed a Preemptable I/O Controller (PIOC) to avoid priority inversion. The commercial product TMS320C621 DSP contains an Enhanced DMA (EDMA) controller [5], which prioritizes transfer requests and prefers serving higher-priority requests. Furthermore, the EDMA uses RAM to store transfer parameters and allows the new channel parameters immediately loaded via a linking mechanism. S. Srinivasam [6] proposed a PDMA (Pre-programmed DMA mechanism) that allow a DMA action to continue moving data even that the source or destination addresses are not consecutive. Although the PDMA can be used to execute a task

chain according to a predetermined schedule, it can not accept any unscheduled request and does not provide facility to circumvent unexpected delays to access I/O or memory.

For example, in Freescale MPC823e, a dedicated RISC core was used to take charge of data transporting [7]. In ADI Blackfin processor, a task-chain based two-dimensional DMA mechanism which was very suitable for video processing was developed [8]. It could efficiently resolve data transporting in the media system. But those above were developed only for the single-processor system which was working in a single clock domain. Faraday's FTDMAC020 [9] has a slave AHB (Advanced High-performance Bus) interface. The DMA is configurable up to an 8-channel DMA engine. Each channel can be assigned a group priority level and channels of the same group priority are serviced in the Round Robin fashion. To ensure real-time applications, the computing system requires guaranteed I/O throughput. Thus, this thesis will improve traditional DMA and integrate Smart DMA into a general-purpose dual-core processor.

1.1 Motivation

In recent years, the DSP applications are widely. Especially, the development of embedded systems and consumer electronics are rapidly, such as mobile phones and MP3, etc. Due to a large number of consumer electronic products, the output value of DSP is about 75%. Therefore the cost and performance will become an important indicator of a choice. If we choose a low-end processor, the performance will decrease because there are a lot of architecture and different addressing modes used in digital signal processing [10], such as circular and bit-reverse addressing. If the processor does not support special addressing modes, it will need to spend more execution

cycles. This means that the processor needs higher clock rate and wastes more power consumption.

The hardware design of digital signal processing has to consider the efficiency of architecture and data transmission. In most of researches, they only enhance the architecture and discarding the mass data transmission. Up to now, the DSP design begins emphasizing and discussing how to enhance DMA architecture in order to control transmission and computation of mass data.

For reducing the cost and complexity of developing a DSP, in this thesis a smart DMA (SDMA) is proposed and integrated into a general-purpose dual-core processor. It can assist the dual-core processor to handle mass and regular operations and to achieve a DSP-like processor. Thus, the traditional DMA architecture will be improved to support multiple addressing modes and built-in operation units at the same time. These multiple addressing modes include increment/decrement, circular, mirror, and bit-reverse addressing. For the different computing case such as FIR, DCT, and FFT [11], we can choose different addressing modes and collocate with dual-MAC architecture to perform different digital signal processing functions.

1.2 Organization of the Thesis

In this thesis, the SDMA architecture is introduced in Chapter 2. Chapter 3 describes system combination of SDMA and a dual-core processor. The experimental results and chip implementation are presented in Chapter 4, and conclusions are made in the last chapter.

Chapter 2

Smart DMA Controller

This chapter will introduce the design of smart direct memory access controller (SDMAC), involving with the SDMAC function, architecture design, and illustrating functions and setting of control register groups.

2.1 Function of Smart DMA

2.1.2 Transfer Modes

The SDMAC supports four transfer modes as memory-to-memory, memory-to-peripheral, peripheral-to-memory, and peripheral-to-peripheral operations. Their function is described as follows, respectively.

1. Memory-To-Memory:

The memory interface of SDMAC supports two embedded memories at the same time. In the mode, data transfer occurs between two memories or in one of memory. SDMAC can read and write data at the same time when the source and destination memory are different. When the source and destination memory are same, SDMAC begins to write destination data, if the internal buffer is full, after reading source data from one of memory. Fig. 2-1 shows the transfer mode under the memory-to-memory data path. The arrow with dotted line indicates all bus transfer paths, and the arrow with hard line indicates the memory-to-memory path through Channel 0.

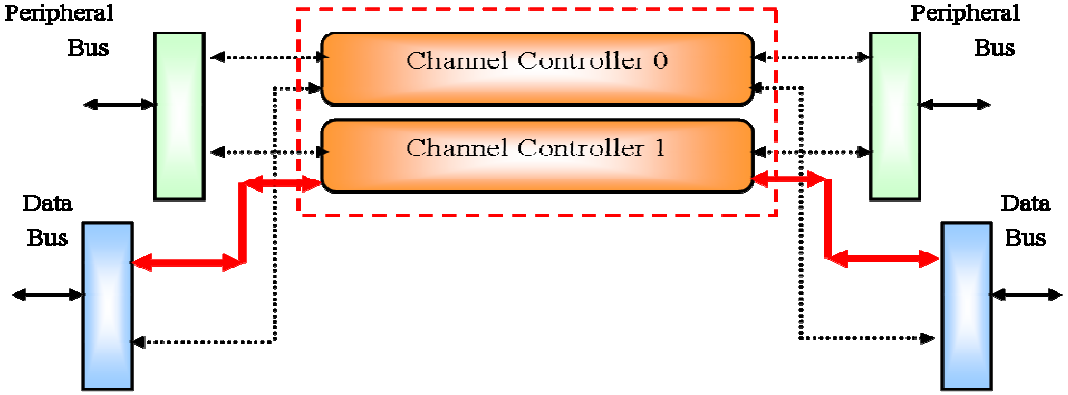


Fig. 2-1: Illustration of the memory-to-memory transfer mode.

2. *Memory-To-Peripheral:*

The mode can transfer data which is in particularly block of data memory to peripheral devices. Fig. 2-2 shows the data path of memory-to-peripheral transfer mode. The arrow with dotted line indicates all bus paths, and the arrow with hard line indicates the memory-to- peripheral path through Channel 0.

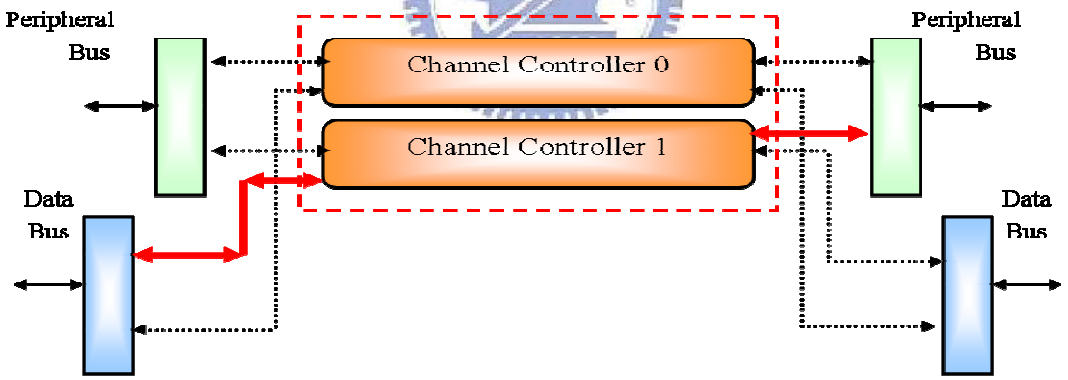


Fig. 2-2: Illustration of the memory-to- peripheral transfer mode.

3. *Peripheral -To- Memory:*

It can transfer the data form peripheral device to a specify memory block using SDMAC, then processor can processing the data in memory. Fig. 2-3 is the data path of peripheral-to-memory transfer mode. It can transfer data from I^2S to memory.

The arrow with dotted line indicates all bus transfer paths, and the arrow with hard line indicates the peripheral-to-memory transfer path through Channel 0.

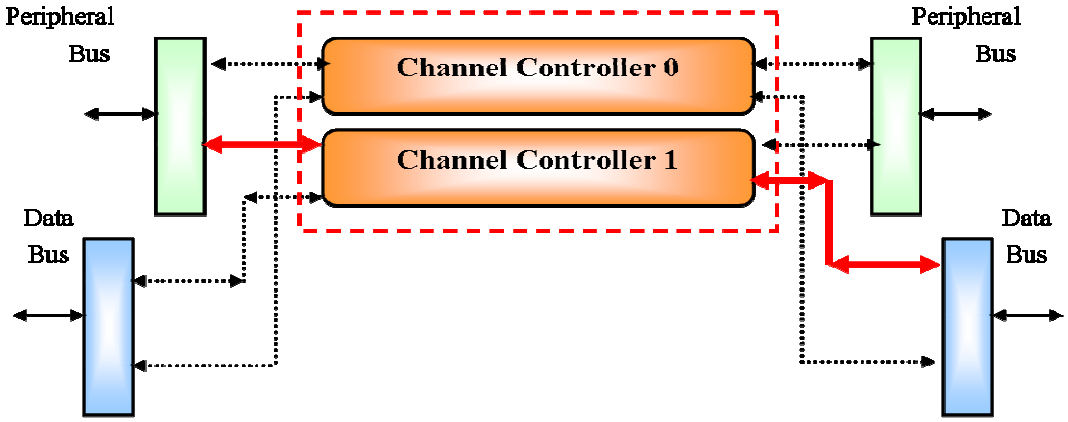


Fig. 2-3: Illustration of the peripheral-to-memory transfer mode.

4. Peripheral -To- Peripheral:

It can set the peripheral-to-peripheral transfer mode by SDMA when the peripheral with ability of processing or just transfer data between two peripherals. It can transfer without interrupt and do not need processor to handle this task. Fig. 2-4 is the data path of peripheral-to-peripheral transfer mode. The arrow with dotted line indicates all bus transfer paths, and the arrow with hard line indicates the peripheral-to-peripheral transfer path through Channel 0.

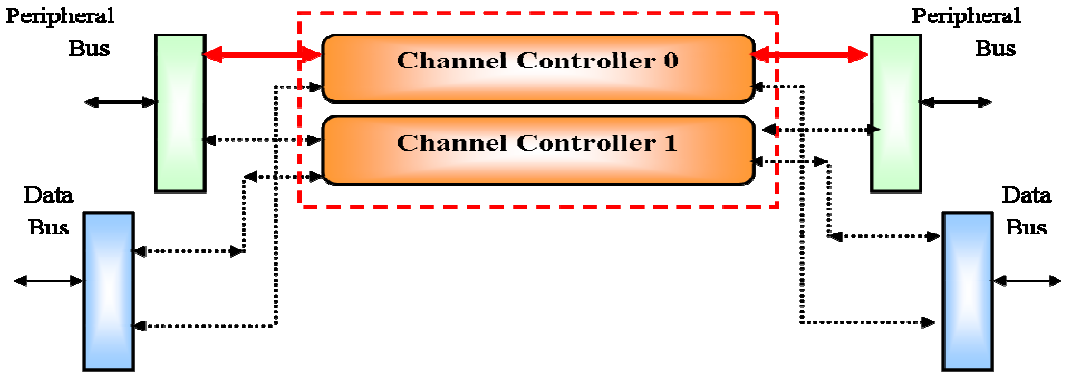


Fig. 2-4: Illustration of the peripheral-to- peripheral transfer mode.

2.1.3 Addressing and Operation Mode

The SDMA supports five addressing modes involving increasing/decreasing addressing, circular addressing, mirror addressing index-based addressing, and bit-reversed addressing. The flowing will illustrate each addressing mode:

1. Increasing/Decreasing Addressing Mode:

This mode is a basic mode and is suitable for the other four modes. Fig. 2-5 is the increasing/decreasing addressing mode. The data is copied from Memory A (RAM A) to Memory B (RAM B). When reading data from RAM A, it uses increasing addressing; when writing data to RAM B, it uses decreasing addressing.

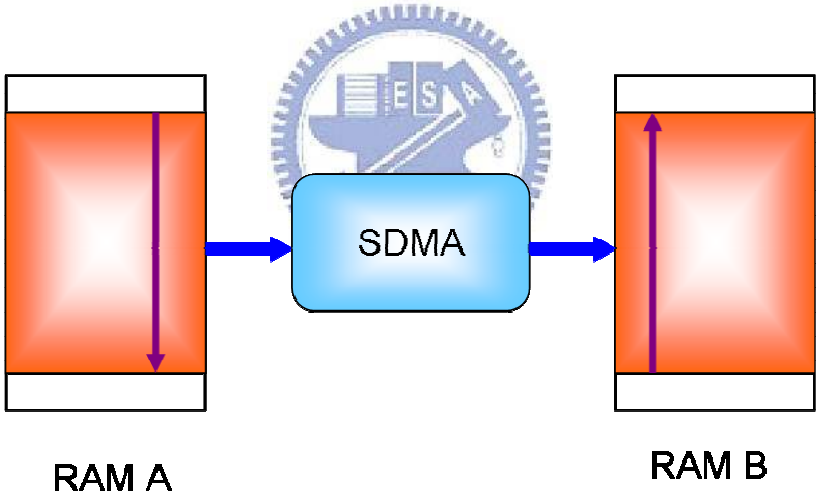


Fig. 2-5: Increasing/Decreasing Addressing Mode

In Fig. 2-6, increasing addressing with Dual-MAC can compute inner product. The data can from memory or peripheral and also can operate in a memory. It can reduce the effort of processor.

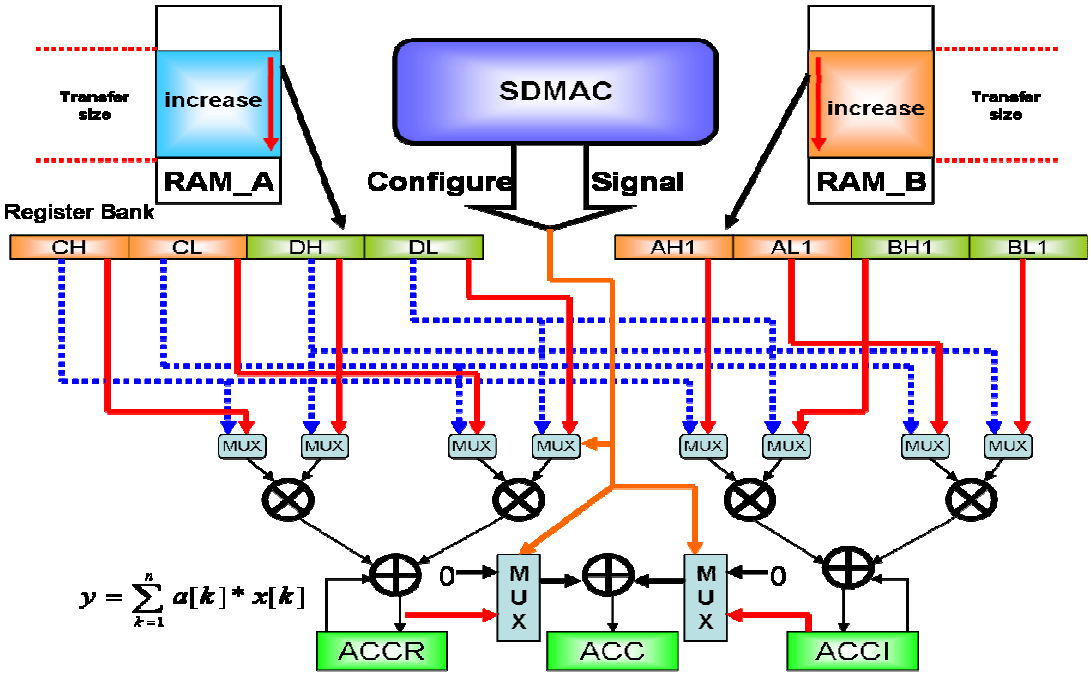


Fig. 2-6: Inner Product with Smart DMA

Increasing and decreasing addressing with Dual-MAC can perform convolution operation such as in Fig. 2-7. The data can from memory or peripheral and also can operation in a memory. It can reduce the effort of processor.

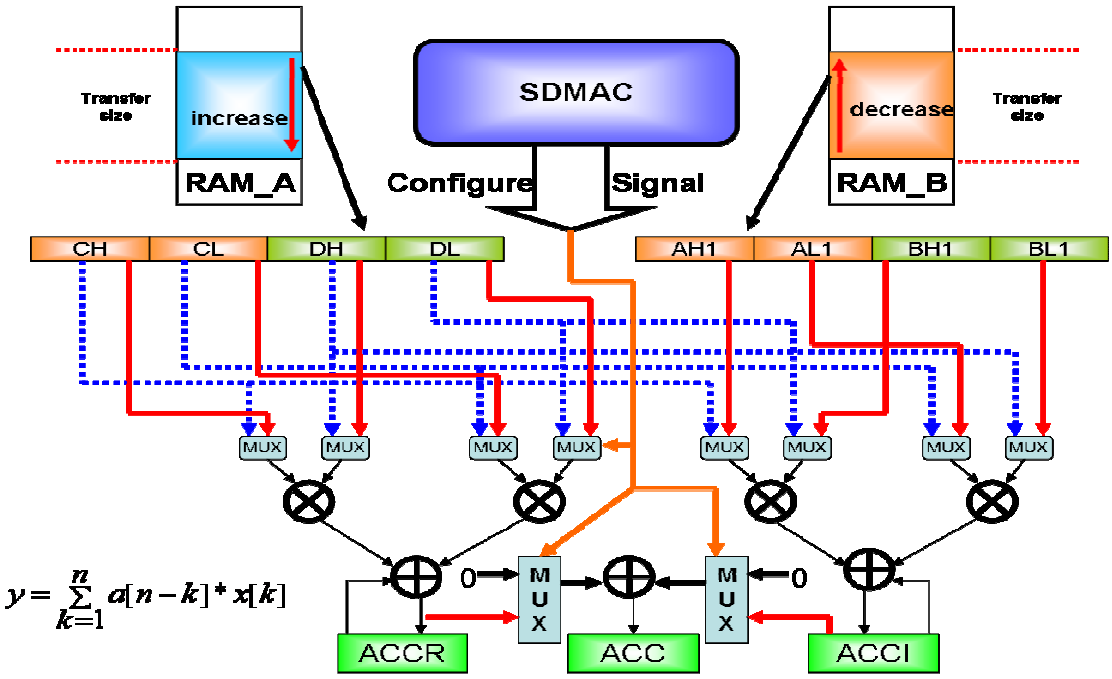


Fig. 2-7: Convolution with Smart DMA

2. *Circular Addressing Mode:*

User can define a block of memory when using circular addressing. When address excesses the boundary of block, it returns to the starting address of the block by the method of circular. To move in circles until the data transfer completely. This addressing can also use increasing or decreasing addressing according to different cases. In Fig. 2-8 , this addressing mode with Dual-MAC can perform circular convolution which is often used in digital signal processing; the X1 in RAM_A is a sequential data block and X2 in RAM_B is a circular data block. SDMAC read data from two memories to do the function of circular convolution.

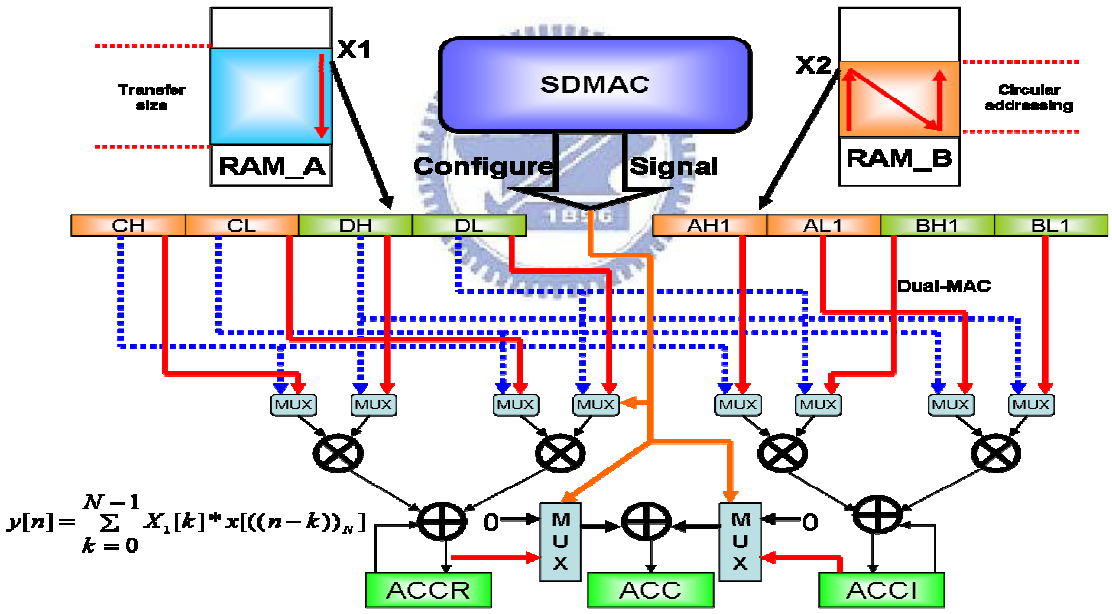


Fig. 2-8: Circular Convolution with Smart DMA

3. *Mirror Addressing Mode:*

User can define a block of memory when using circular addressing. When address excesses the boundary of block, it will increase or decrease from the boundary address by the method of mirror until the data transfer completely. In Fig. 2-9, this addressing mode with Dual-MAC can perform Discrete Cosine

Transform (DCT) which is often used in digital signal processing; X1 in RAM_A is a sequential data block, and X2 in RAM_B is a mirror data block. SDMAC read data from two memories to execute. It can achieve the operation of mirror addressing.

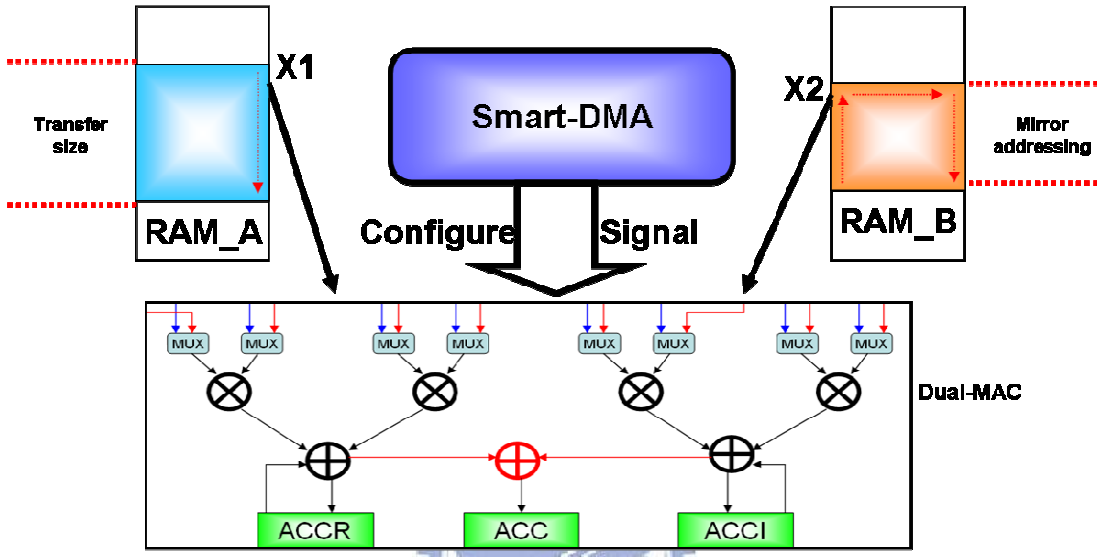


Fig. 2-9: Mirror addressing of Smart DMA

4. *Index-based Addressing Mode :*

User can set the index of increasing or decreasing address when use index-based addressing mode to transfer data. It can choose valid data to reduce the bandwidth when transfer. It can do a lot of digital signal processing functions such as FIR (Finite Impulse Response), DCT (Discrete Cosine Transform). In Fig. 2-10, RAM_A is a sequential data block and reads data from RAM_B by index-based addressing. SDMAC read data from two memories to perform automatically. It can achieve the operation of mirror addressing.

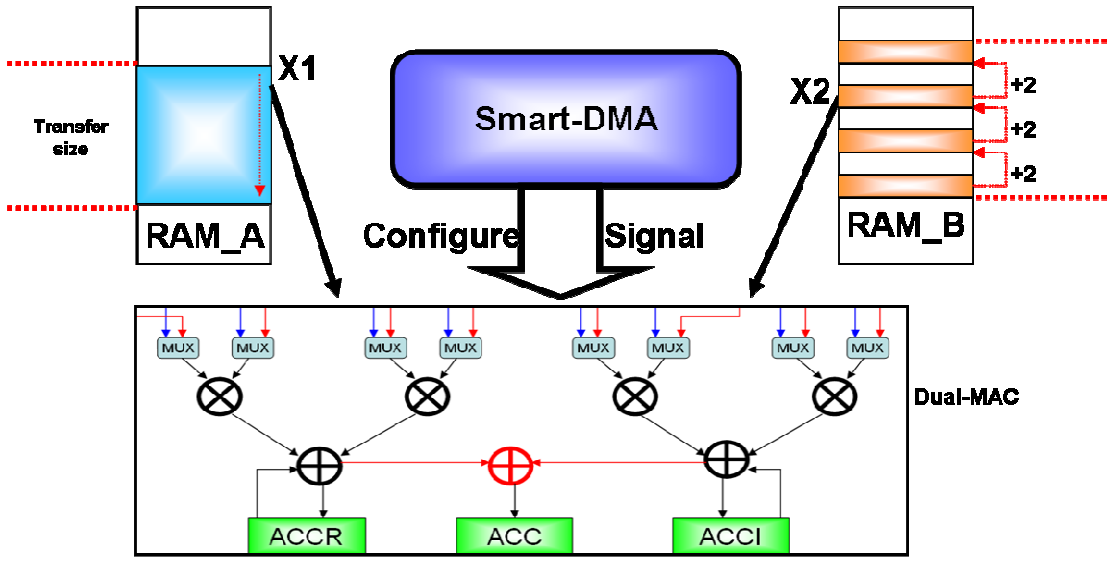


Fig. 2-10: Index-based addressing of Smart DMA

5. *Bit-Reversed Addressing Mode:*

By using bit-reverse addressing, we can reorder the coefficient before computing and control the data path of Dual-MAC by configure signal. This method can perform a butterfly unit in a cycle to speed up the FFT operation as show in Fig. 2-11.

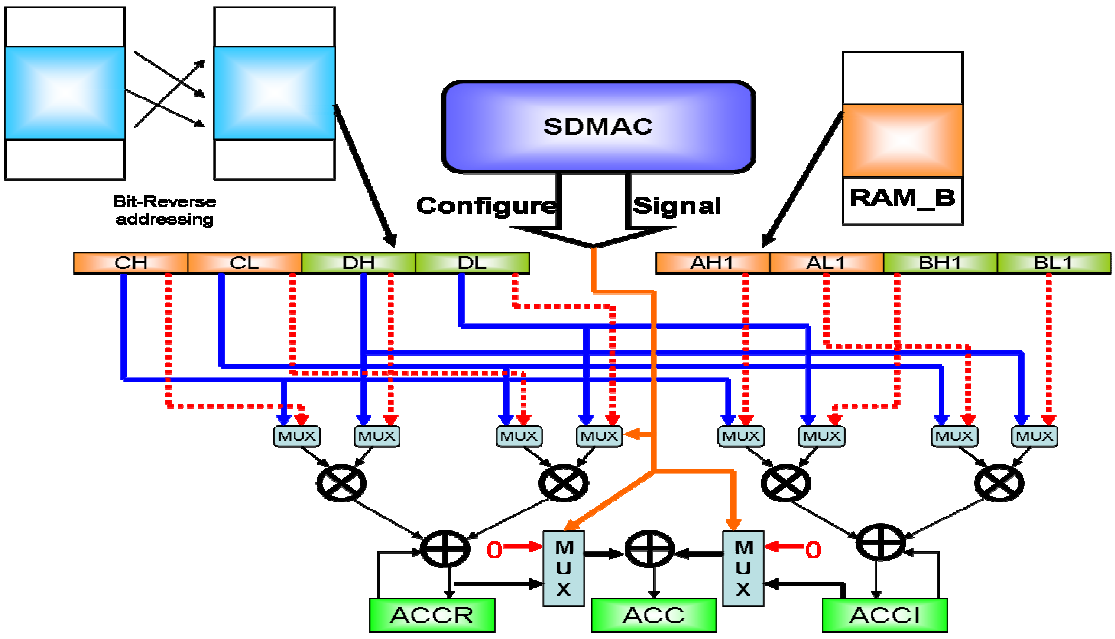
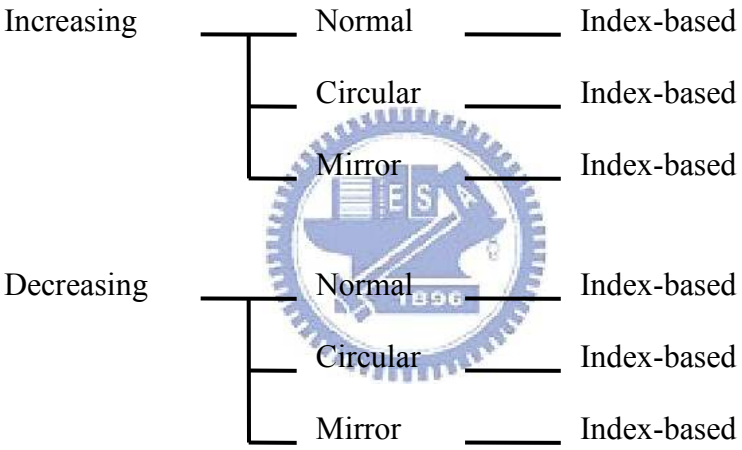


Fig. 2-11: Bit-Reverse addressing of Smart DMA

We can divide the five addressing mode into 4 types such as D-type, B-type, I-type, and bit-reverse as shown in Table. 2-1.

Table. 2-1: Addressing type of Smart DMA

Direction (D-type)	Block (B-type)	Index (I-type)	other
Increasing Decreasing	Circular Mirror Normal	Index-based	Bit-reverse



2.2 Architecture of SDMAC

The SDMAC is different from traditional DMA design because it has multiple addressing modes and built-in Dual-MAC. The SDMAC architecture is shown in Fig. 2-12. This includes two channel controllers, an arbiter, a register bank, an interrupt controller, memory interface, and a Dual-MAC. The following subsection will illustrate each component.

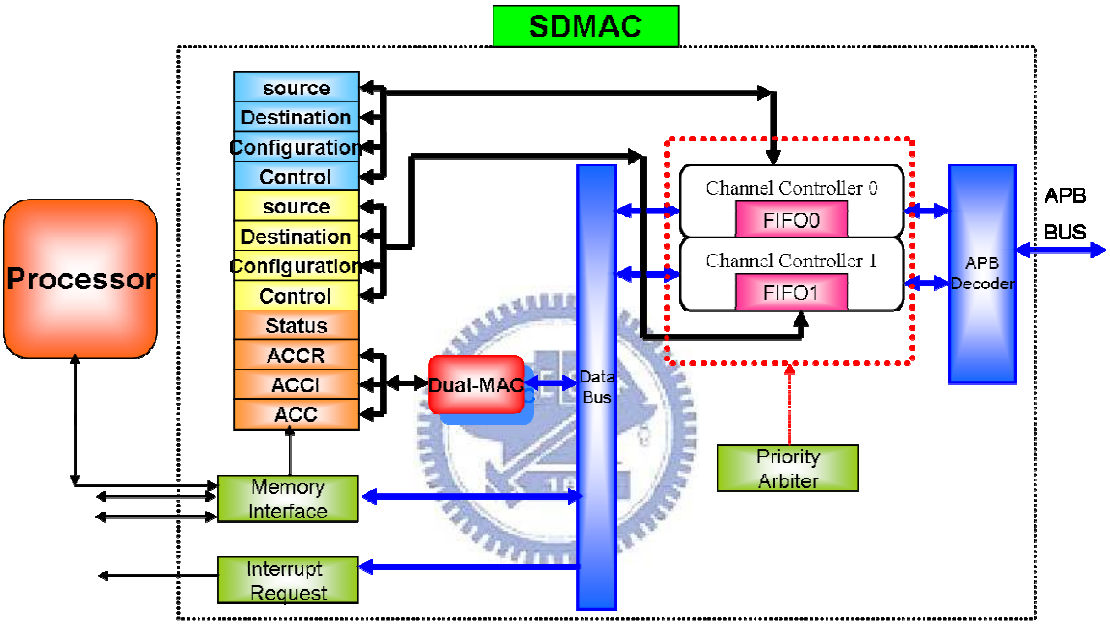


Fig. 2-12: Architecture of Smart DMA

2.2.1 Channel Controller

The SDMAC has two identical channel controllers. Each channel controller plays the most important role in SDMAC. It will control all operations of data transmission. According to the architecture of Fig. 2-12, when the data bus is not conflict, all channels can work at the same time. On the other hand the controller will decide channel priority by the arbiter. Of course, the number of channel controllers can be increased easily to enhance processor performance. The channel controller consists of two independent controllers. One is regarded as reading controller. The

other is regarded as writing controller. When the controller reads or writes data, first it has to check the FIFO (First In First Out) status. If FIFO is empty, the reading controller can read data from external devices to FIFO. If FIFO is full, the writing controller can write data from FIFO to external devices. Otherwise, the controller is disabled. The register bank connected to two channel controllers separately can decided all functions of SDMA. Both reading and writing controllers are implemented by FSM (Finite State Machine). The channel controller architecture is shown in Fig. 2-13.

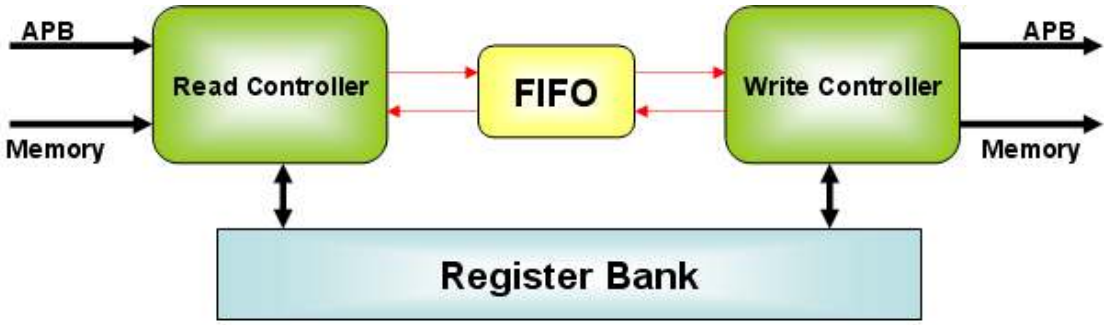


Fig. 2-13: Architecture of Channel Controller

The two channel controllers are disabled before using SDMAC. They will enter different modes according to the given transfer mode (Memory or Peripheral). Then SDMAC is enabled. The design flow for two channel controllers is explained in Fig. 2-14. First, the state stays in the idle mode. If the authority of reading or writing operation is confirmed, the controller will enter the setup mode. After the data of source or destination and FIFO are valid, the controller will enter the enable mode to do data transmission. If data is not yet transferred completely, the controller will enter the setup mode and wait next data. If data is transferred completely, the controller will return to the disabled mode and send the interrupt signal to the processor. Finally, data transmission is finished.

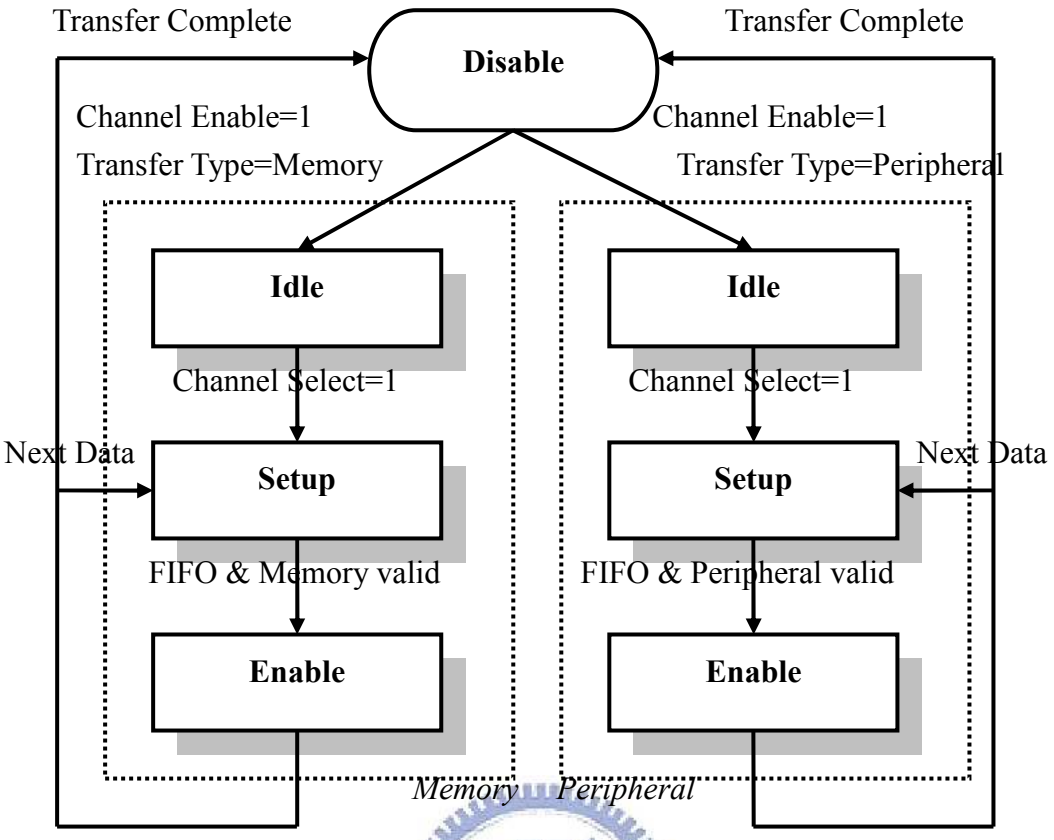


Fig. 2-14: Finite State Machine of read/write Controller

2.2.2 Prioritizing Arbiter

In Fig. 2-12, these two channels (named Channel 0 and Channel 1) share an internal data bus and a peripheral bus. If these two channels use the same bus at the same time, the bus conflict will occur. So, a prioritizing arbiter is designed to avoid data conflict between the same buses. The arbiter shown in Fig. 2-15 decides use of bus in order of priority. The priority of Channel 0 is higher than Channel 1. When these two channels obtain data from the same bus, the arbiter will halt operations of Channel 1 due to a lower priority. If Channel 0 finishes the jobs, the arbiter will return the control authority to Channel 1. When the lower priority channel is transferring data, if data conflict occurs, it must finish the present data transmission first and then release the control authority. When transfer the data from memory, it is usually a mass

and not interrupted. To avoid transferring low-speed data by peripheral devices can not get the bus control authority. Hence, we suggest setting a lower priority when transfer the data from memory under the arbiter architecture.

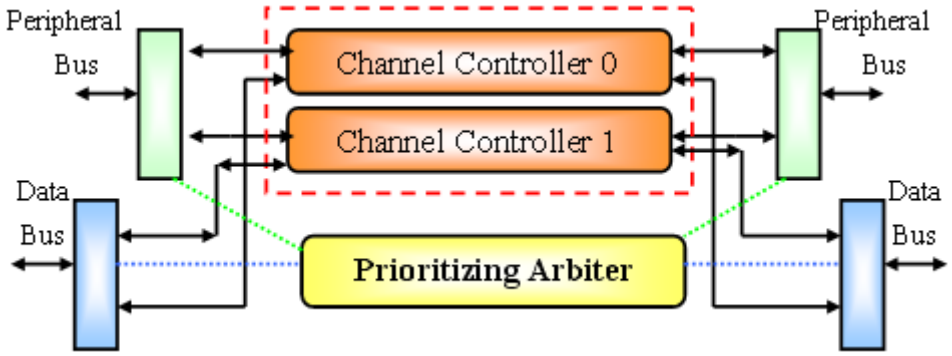


Fig. 2-15: Prioritizing Arbiter of Smart DMA

2.2.3 FIFO (First In First Out)

The channel controller needs a buffer such as FIFO to save data. The FIFO structure can read, write, and check own state. FIFO control signals are shown in Table. 2-2. They are independent and used to design the controller read and write of the channel controller.

There are two FIFO pointers (PT_PUSH and PT_POP) shown in Fig. 2-16. They individually point to the PUSH address and POP address. Data will push into the buffers when push is happened and the pointer PT_PUSH will move forward. When PT_PUSH excess the end of FIFO, it will return to the start point of this FIFO. Data will pop out from the buffers when pop is happened and the pointer PT_POP will move forward. When PT_POP excess the end of FIFO, it will return to the start point of this FIFO.

Table. 2-2: FIFO Control Signals

Read Control		
Signal	Bit width	Description
data_in	32 bits	The data push into the FIFO.
Push	1 bit	When clock at rising edge and this signal is high Data will push into the FIFO.
Write Control		
Signal	Bit width	Description
data_out	32 bits	The data pop out from FIFO.
Pop	1 bit	When clock at rising edge and this signal is high Data will pop out from FIFO.
State Check		
Signal	Bit width	Description
Full	1 bit	Data in FIFO is full.
Empty	1 bit	No data in FIFO.
Half	1 bit	Data in FIFO is over than half.

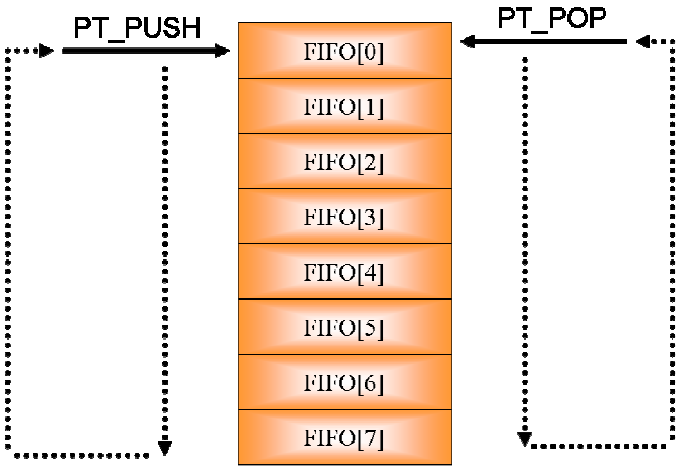


Fig. 2-16: Pointer of FIFO

There is a counter in the FIFO structure. It counts the number of data saved in FIFO. The current state of FIFO from the counter can be obtained directly, such as shown in Fig. 2-17.

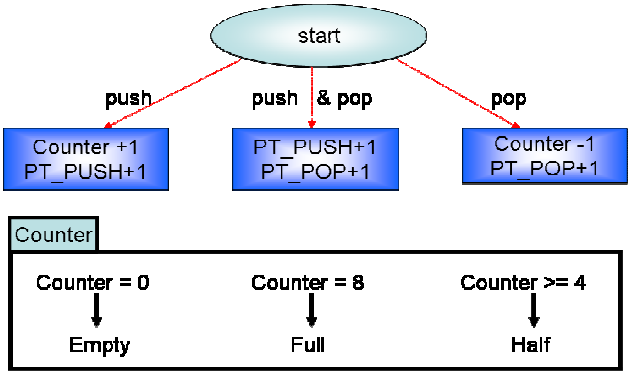


Fig. 2-17: Pointer and counter of FIFO

2.2.4 Register Bank

The SMDAC has 12 control registers called a register bank, where each register is 32 bits. The register bank seems a 16 × 16 memory array in the processor. Hence, the processor can directly read/write the register bank as access memory. This method is called as memory-mapped I/O. The use of the control register is illustrated in next subsection in details.

For register reading and writing in SDMAC, the timing diagram is shown in Fig. 2-18 and Fig. 2-19. Reading or writing operations are enabled by the rising edge clock and CEN=0. When WEN is high, the register bank can be read according to the given address. When WEN is low, data can be written to the register bank of SDMAC according to the given address.

Fig. 2-20 shows the register bank in SDMAC. The left side is connected to the part of processor's memory. Another side is connected to two channel controllers and a dual-MAC unit.

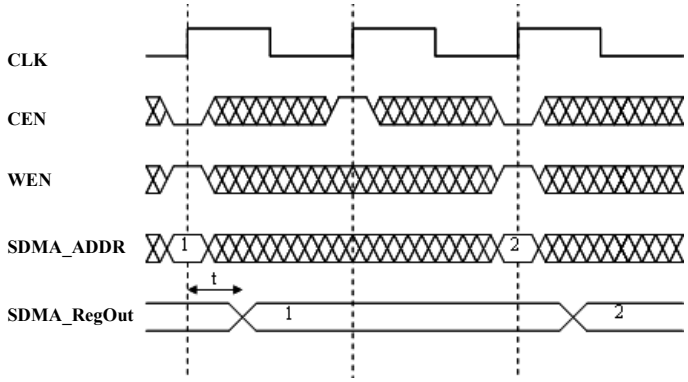


Fig. 2-18: Timing Diagram of reading data from register in Smart DMA

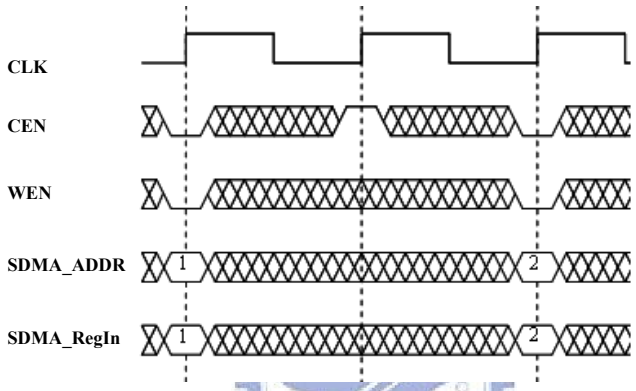


Fig. 2-19: Timing Diagram of writing data from register in Smart DMA

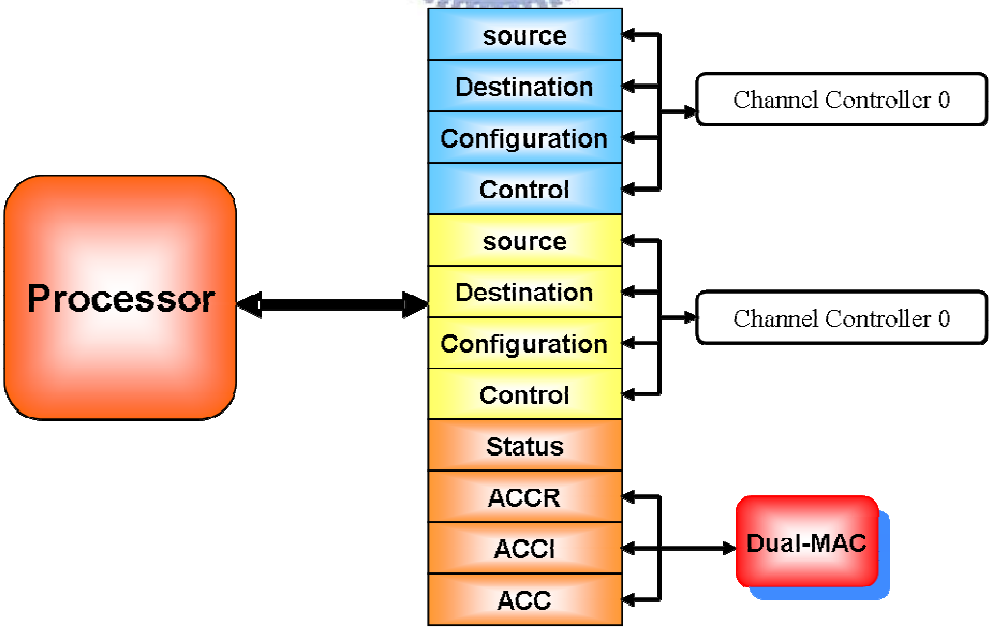


Fig. 2-20: Register Bank of Smart DMA

2.2.5 Interrupt

The SDMAC can enable interrupt via setting configuration register. After the interrupt is set, if the channel controller finishes assigned works, an interrupt signal is sent to the processor immediately. The channel controller is closed at the same time. The interrupt signal is active low and holds on for two clock cycles. The channel controller is connected to the interrupt controller. When data transmission is finished, an interrupt signal is generated and written to the status register as shown in Fig. 2-21 .

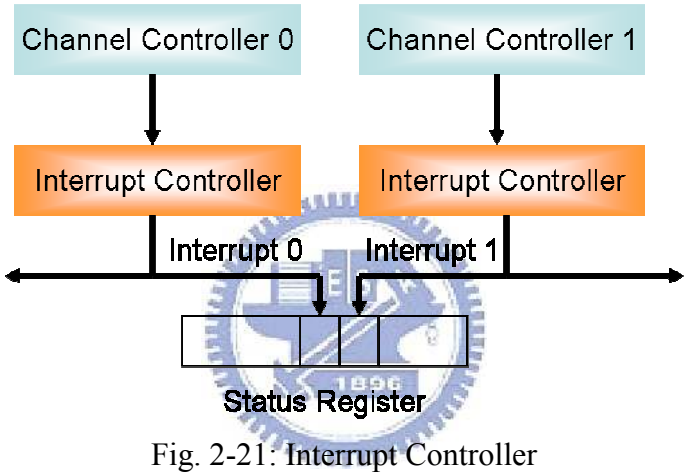


Fig. 2-21: Interrupt Controller

2.2.6 Memory Interface

The SDMAC supports two synchronous high-density and single-port SRAM. Each memory has a 9-bit address line, a 32-bit data line, and read/write control lines. Fig. 2-22 shows the data transmission from RAM_A to RAM_B and the memory interface. If the processor wants to access internal memory, it also uses the same memory interface.

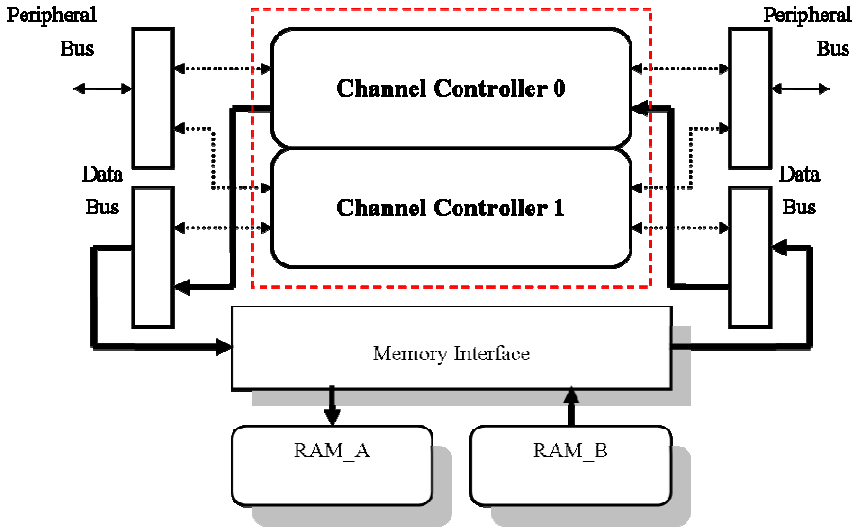


Fig. 2-22: Data transmission via internal memory interface

The memory symbol is shown in Fig. 2-23. The pin assignments are shown in Table. 2-3.

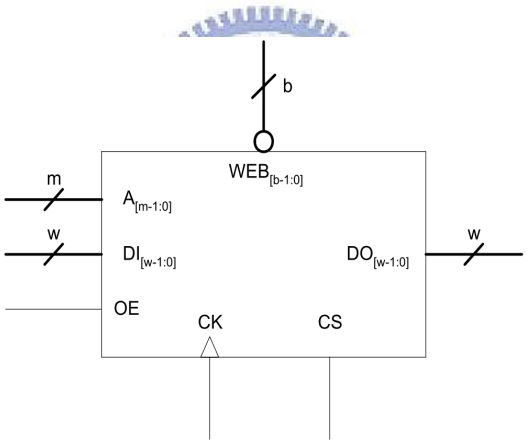


Fig. 2-23: Memory symbol.

Table. 2-3: Pin descriptions of memory

Signal	Bit Width	Description
CK	1	Clock signal
A	m	Address signal
D	w	Input data
OE	1	Output enable signal, active high
CS	1	Chip select, active high
WEB	1	Write enable signal, active high
DO	w	Output data

Programmers set SDMA control registers to select one of memory banks which want to read or to write. So the memory interface has to conform to the timing diagram of Fig. 2-24 and Fig. 2-25.

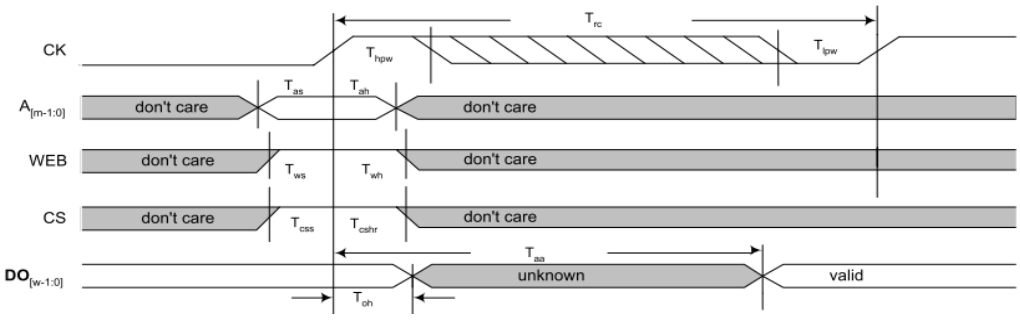


Fig. 2-24: Read cycle timing diagram of memory

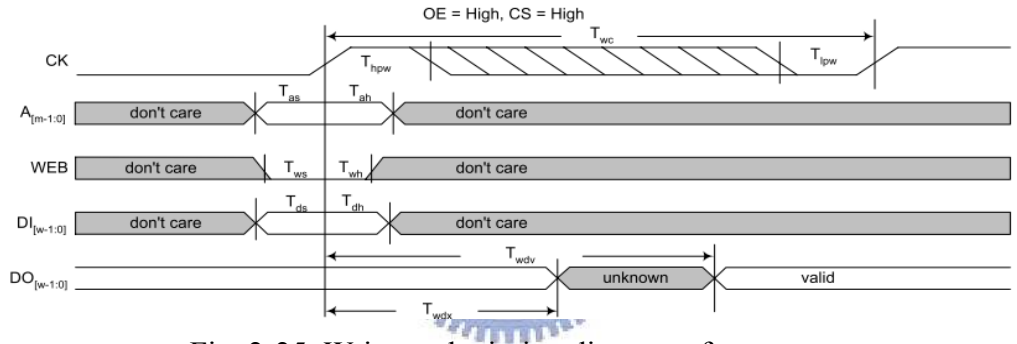


Fig. 2-25: Write cycle timing diagram of memory

2.2.7 APB(Advanced Peripheral Bus)

APB is a part of AMBA [12] (Advanced Microcontroller Bus Architecture), the control signals are shown as Fig. 2-26. It has low power and easy to use. The channel controller in Smart DMA supports the interface of APB directly. The timing diagram of reading cycle of APB shows in Fig. 2-27. The timing diagram of writing cycle of APB shows in Fig. 2-28.

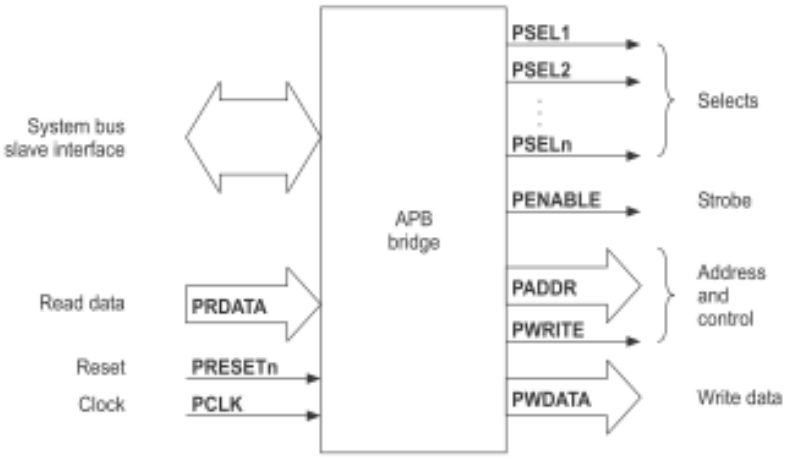


Fig. 2-26: APB bridge interface

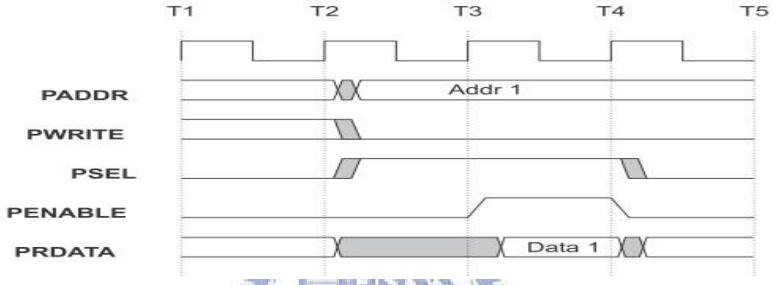


Fig. 2-27: Read cycle timing diagram of APB

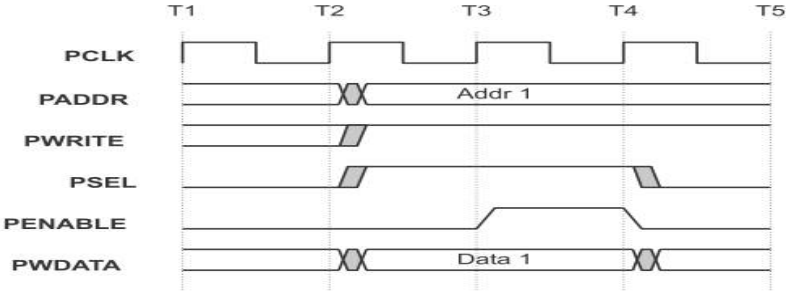


Fig. 2-28: Write cycle timing diagram of memory

2.2.8 Dual-MAC

The MAC operation is one of the most important features for a DSP. This is not only used for multimedia applications but also used for communication applications. Traditional DSP with a real-valued MAC data path cannot efficiently execute arithmetic operations on complex-valued signals. Accordingly, recent DSP are designed with a view to improve the efficiency of the complex-valued MAC operation

[13], [14], [15], [16]. The complex-valued MAC operation consists of four real-valued multiplications and two accumulations as Eq. (2.1), where ACCR is the real part of the accumulation register, and ACCI is the imaginary part of the accumulation register.

$$\begin{aligned}&(ACCR + jACCI) + X * Y\\&= (ACCR + jACCI) + (X_R + jX_I) * (Y_R + jY_I)\\&= (ACCR + jACCI) + (X_R Y_R - X_I Y_I) + j(X_R Y_I + X_I Y_R)\end{aligned} \tag{2.1}$$

The dual-MAC architectures, such as DSP16000 [14] and MDSP-II [16], use two parallel 16×16 multipliers to calculate the product and to sum the real and imaginary parts, respectively. Thereby, it spends two cycles for completing one complex-valued MAC operation. The swap multiplexer in DSP16000 and MDSP-II is used to select a pair of inputs from four operands and to generate four partial products during the complex-valued multiplication operation. The other dual 16×16 MAC architectures in LODE [13] operate by a similar strategy except that it employs a delay register to store the input operands for the next MAC operation. This approach also executes one complex-valued MAC within two cycles. Straightforward the extension of the previous architectures to a 16×16 complex-valued MAC architecture using four parallel real MAC units is possible but unpopular, primarily because of its lack of hardware efficiency. Although the four-parallel-MAC architecture can perform one complex-valued MAC operation within one cycle, three MAC units sit idle in a real-valued MAC operation. An efficient four-parallel MAC architecture is proposed in the proposed SDMAC design.

The proposed dual-MAC architecture is shown in Fig. 2-29. We can configure the data path by the control register to perform different DSP applications. The complex-valued MAC is composed of four real multipliers and two accumulators as

shown in Fig. 2-29. (ACC is not used in this case.) Table. 2-4 lists an example of the execution of a 3-tap complex FIR filter to compute one output samples within three cycles. In this case, the signal Mode is switched to the complex mode. The data path can perform a *k*-tap real-valued FIR within *k*/2 cycles, and it can perform a butterfly unit within one cycle.

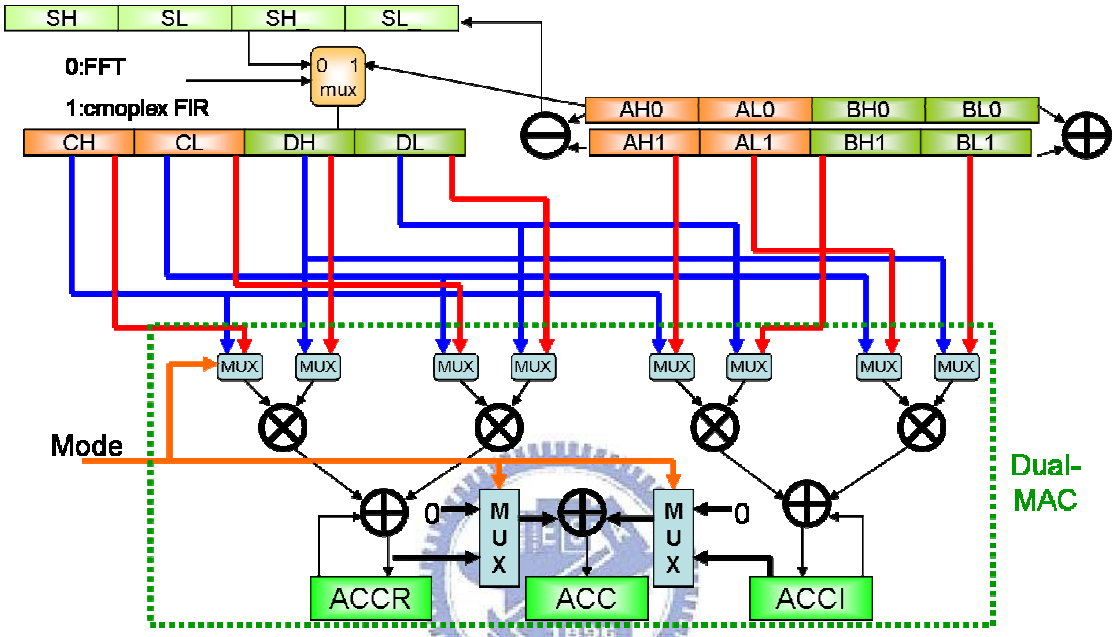


Fig. 2-29: Architecture of dual-MAC

Table. 2-4: Processing schedules of a complex 3-tap FIR convolution

Cycle	MAC Input		Product	
	CH	CL	P0	P2
	DH	DL	P1	P3
1	CR(0)	CI(0)	CR(0)XR(0)	CR(0)XI(0)
	XR(0)	XI(0)	CI(0) XI(0)	CI(0)XR(0)
2	CR(1)	CI(1)	CR(1)XR(1)	CR(1)XI(1)
	XR(1)	XI(1)	CI(1) XI(1)	CI(1)XR(1)
3	CR(2)	CI(2)	CR(2)XR(2)	CR(2)XI(2)
	XR(2)	XI(2)	CI(2) XI(2)	CI(2)XR(2)

2.3The register bank of Smart DMA

To achieve the best performance of SDMAC, users must set the configuration register. The following will illustrate these functions and the method how to set the register bank in SDMAC.

2.3.1 Source Register

The source register is 32 bits in the register bank, but is seems to two 16 bits for processor. After setting the source register by processor, the value will preserve. But the source address will be changed according to the addressing mode. In the source register, the high 16-bit part specifies the addressing mode, and the low 16-bit part is defined as source address as shown in Table. 2-5. The following will illustrate the arrangement for each bit.

Table. 2-5: Source Register

Bit	**High Source Register**	
15	Source Circular	0:circular / 1:mirror
14:7	Source Base	Address number of increasing or decreasing
6:0	Source Offset	The start position of block
Bit	**Low Source Register**	
15	Source Device	0:Ram_A / 1:Ram_B
14:0	Source Address	Source address

(1) Source circular: decide circular addressing or mirror addressing. If it does not need both of them, set the block size equal to zero.

(2) Source Base: to decide the index of increase or decrease of the index-based addressing.

(3) Source offset: to decide the distance of first data form the block boundary as shown in Fig. 2-30.

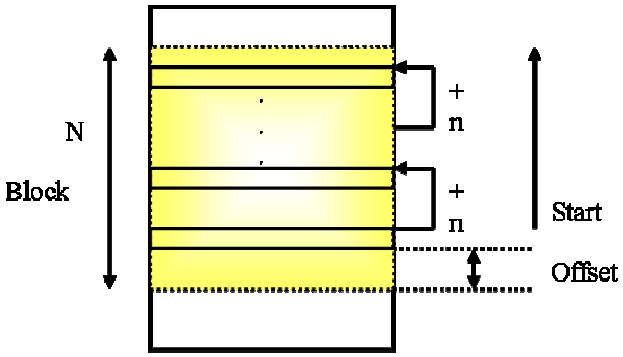


Fig. 2-30: Source offset

2.3.2 Destination Register

The setting method is same as the source register. The register is applied to a target device. After setting the destination register by processor, the value will preserve. But the destination address will be changed according to the addressing mode. The high 16-bit part specifies the addressing mode, and the low 16-bit part is defined as destination address as shown in Table. 2-6.

Table. 2-6: Destination Register

Bit	**High Destination Register**	
15	Destination Circular	0:circular / 1:mirror
14:7	Destination Base	Address number of increasing or decreasing
6:0	Destination Offset	The start position of block
Bit	**Low Destination Register**	
15	Destination Device	0:Ram A / 1:Ram B
14:0	Destination Address	Destination address

2.3.3 Control Register

The control register seems two 16-bit registers. After setting the control register by processor, the value will preserve. But the number of data transmission accompanying with writing data to target device actually will be reduced. The high 16-bit part indicates the source and destination region. The low 16-bit part is control signals of SDMAC as shown in Table. 2-7.

(1) Source/destination region: indicate the source or destination region of mirror or circular addressing.

(2) Source increment: increase source address (increase=1).

(3) Source decrement: decrease source address (increase=0, decrease=1).

(4) Destination increment: increase destination address (increase=1).

(5) Destination decrement: decrease destination address (increase=0, decrease=1).

Table. 2-7: Control Register

	Source/Destination Region Register	
Bit	**Signal**	**Description**
15:8	SrcReg	Block size of source
7:0	DestReg	Block size of destination
	Control Register	
Bit	**Signal**	**Description**
15	SrcInc	Source Increase
14	SrcDec	Source Decrease
13	DestInc	Destination Increase
12	DestDec	Destination Decrease
11	SrcWidth	Source Width(Default =32 bits)
10	DestWidth	Destination Width(Default =32 bits)
9:0	TransferSize	Transfer Size

2.3.4 Configuration Register

The configuration register seems one 16-bit register which can access from the processor. After setting the configuration register, the value will preserve. But the reset signal of accumulators (high-active) will become low after one cycle by setting the configuration register. After channel enable and data transfer completely, the channel controller will close automatically. The following will illustrate the arrangement of bit as shown in Table. 2-8.

(1) Halt: channel access stop. It will stop after write out the data from FIFO.

(2) Interrupt enable.

(3) Source peripheral: supporting 8 source peripheral devices.

(4) Destination peripheral: supporting 8 destination peripheral devices.

(5) Transfer type: transfer Type (0：RMA/ 1：I/O device).

(6) Sequence transfer: continuously transfer, the transfer data are infinite (ST=1).

(7) Function: operation functions (normal: 000 MAC: 001 CFIR: 010 FFT: 100).

(8) ACC clear: accumulator clear (AC = 1).

(9) Channel enable.

Table. 2-8: Configuration Register

Bit	**Configuration Register**	
	signal	**description**
15	Halt	Channel stop
14	IntEn	Channel interrupt enable
13:11	SrcPer	Source Peripheral
10:8	DestPer	Destination Peripheral
7:6	TransferType	Transfer Type
5	SeqTran	Sequence Transfer
4:2	Func	Function
1	ACClr	ACC Clear
0	ChEn	Channel Enable

2.3.5 Status Register and ACC Register

The status register seems one 16-bit register which can access from the processor. It consist the information of two channels. The higher bits are channel 1 and the lower bits are channel 0. The following will illustrate the arrangement of bit as shown in Table. 2-9.

(1) Interrupt：Low-Active interrupts.

(2) Full: The state of FIFO.

(3) Half: The state of FIFO.

(4) Empty: The state of FIFO.

(5) Channel Select: Arbiter decides which channel is working.

(6) Error: accumulator overflow.

Table. 2-9: Status Register

Bit	**Status Register**	**[15:8] Channel 1 / [7:0] for Channel 0**
	Signal	**Description**
7	Interrupt	Channel Interrupt
6	Full	DMA FIFO data full
5	Empty	DMA FIFO data empty
4	Half	DMA FIFO data more than half
3	ChSel	Channel is select in working state
2:0	Err	Accumulator overflow

The ACC register seems one 16-bit register which can access from the processor. It can only write data with a 16-bit format and only read data with a 32-bit sign-valued format. The ACC is 40 bits in register bank.

2.3.6 Operating Flow of SDMA

The setting method of SDMAC is shown in Fig. 2-31. First, set the source, destination, and control register and then set the configuration register to enable the SDMA channel. This moment it can do other tasks until interrupt occurred, then the SDMAC finish its jobs.

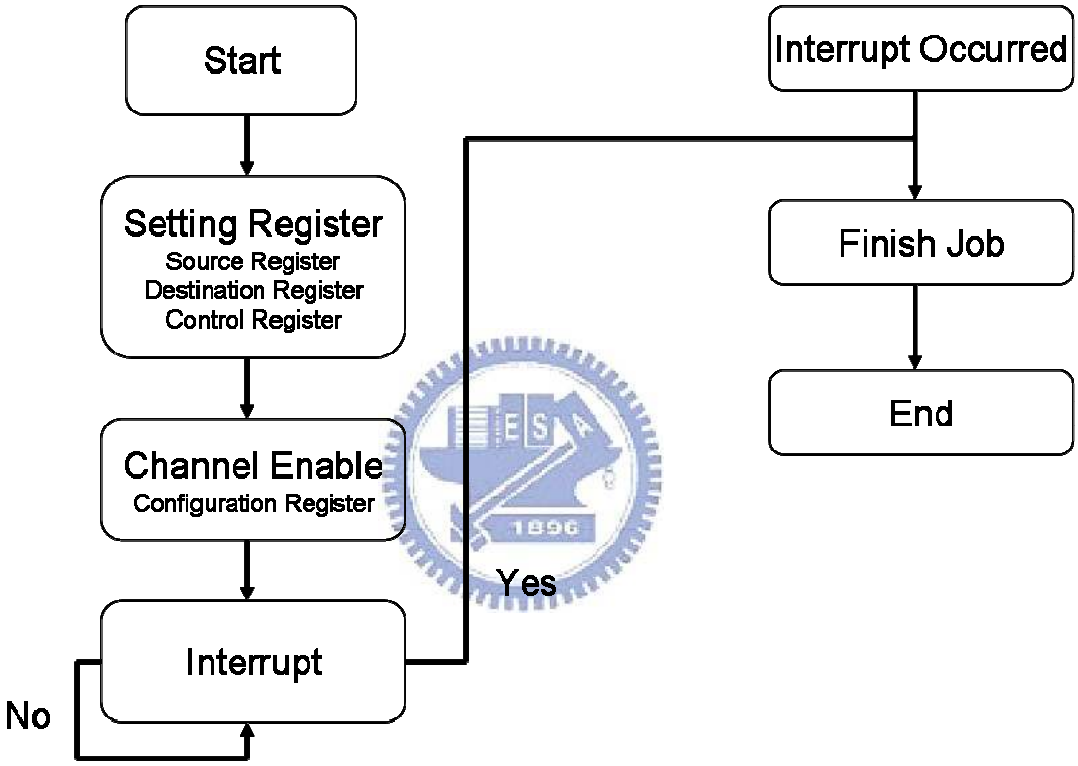


Fig. 2-31: Operating flow of SDMA

Chapter 3

SDMA Integrated with Dual-Core Processor

The SDMAC can not work alone so it needs to put into a processor and to use. In this thesis, we also develop a general-purpose dual-core processor. The following subsection will illustrate characteristics, instructions sets, assembler, and architecture of the dual-core processor.

3.1Architecture of Dual-Core Processor

The dual-core processor is RISC architecture and has five-stage pipeline [17], [18], where dual-core design can be a DSP unit or micro-control unit. The proposed SDMAC designed is also integrated into the dual-core signal processor [21]. We can set SDMA operations via defined assembly instructions. The SDMAC has a peripheral bus called APB so that some peripheral devices can be mounted. It can transfer data from peripheral to memory by SDMAC.

3.1.1 Processor Kernel

The processor which is a dual-core processor has a five-stage pipeline. The design of processor architecture shown in Fig. 3-1 includes a program counter (PC), an instruction fetch unit (IF), an instruction decode unit (ID), a register file, an execution unit. The following will illustrate their function for each component.

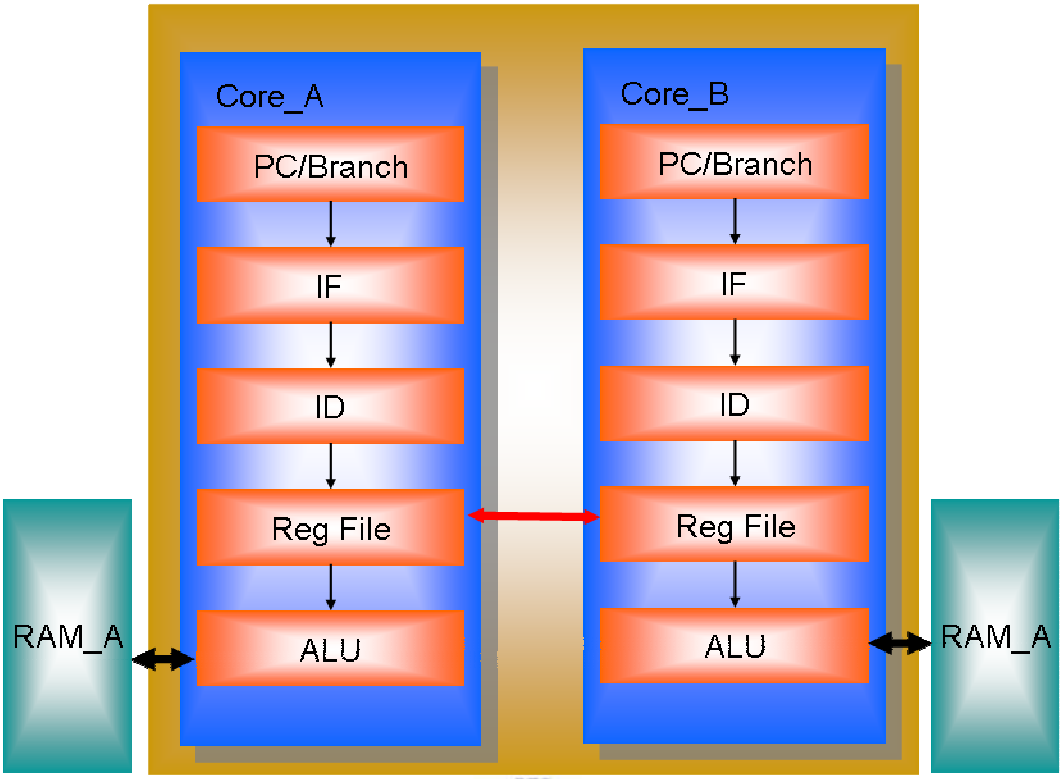


Fig. 3-1: Processor Kernel

(1) Program Counter (PC): PC is an internal register. When the value of PC is changed and sent to the next stage, the processor will jump correct instruction address to perform.

(2) Instruction Fetch: According to PC as a memory address, fetch one instruction from an external program ROM. The memory address line and data width is 16 bits and 32 bits, respectively. Because there are some instructions need the value of PC in the execution stage, passing it stage by stage. To avoid fetching an instruction after a branch instruction, two flags are set to stall the pipeline.

(3) Instruction Decode: The fetched instruction from ROM is decoded. In the instruction format, there are two source registers and one destination register. These addresses will pass to next stage for fetching operands and the write back address of the execution stage. Due to the dual-core architecture, it can

get register data for each other. Also this processor combines SDMAC design so a few instructions are defined. When the processor uses SDMAC, it needs a set of address bus to access the register bank and to set parameters.

(4) Register File: Each core has 32 registers. It has two read ports and one write port to offer operands for the execution and write-back stage. There are a lot of signals connected to the execution stage for data forwarding.

(5) Execution: The unit is to compute and to handle data forwarding. The forwarding unit is used to avoid a RAW hazard in the pipeline.

There are some special designs in the dual-core processor should be presented as follows.

(1) Operand isolation: We will know what kind of instructions in the decode stage. It can set an enable signal to control the execution stage. Then it can avoid perform addition, subtraction, multiplication at the same time to reduce power consumption. With this method, the processor can save about 30% power consumption.

(2) A MAC instruction completed in one cycle.

(3) Bit-reverse addressing for FFT operation.

3.1.2 Instruction set architecture

The instructions can be divided into 5 groups including data moving, arithmetic and logic, branch and jump, SDMA, and others.

(1) Arithmetic and logic instructions are listed in Table. 3-1.

Table. 3-1: Instructions of logic and arithmetic

Instruction	Opcode	Example
NOP	000000	NOP
ADD	000001	ADD rd, rs, rt
SUB	000010	SUB rd, rs, rt
MUL	000011	MUL rd, rs, rt
MAC	000100	MAC acc, rs, rt
AND	000101	AND rd, rs, rt
OR	000110	OR rd, rs, rt
XOR	000111	XOR rd, rs, rt
INV	001000	INV rd, rs
ADDI	001001	ADDI rd, rs, imm
SUBI	001010	SUBI rd, rs, imm
MULI	001011	MULI rd, rs, imm
ANDI	001100	ANDI rd, rs, imm
ORI	001101	ORI rd, rs, imm
XORI	001110	XORI rd, rs, imm

(2) Jump and branch instructions are listed in Table. 3-2.

Table. 3-2: Instructions of jump and branch

Instruction	Opcode	Example
J	001111	J, imm
JR	010000	J, @rd
BEZ	010001	If(rs=0) go to address
BNEZ	010010	If(rs!=0) go to address
BGTZ	010011	If(rs>=0) go to address
BLTZ	010100	If(rs<=0) go to address
BEQ	010101	If(rs=rt) go to address
BNE	010110	If(rs!=rt) go to address
BGT	010111	If(rs>rt) go to address
BLT	011000	If(rs<rt) go to address

(3) Data instructions are listed in Table. 3-3.

Table. 3-3: Instructions of data moving

Instruction	Opcode	Example
LW	011001	LW, rd, address
SW	011010	SW, rs, address
MOVRC	011011	MOVRC, rt, imm
MOVRR	011100	MOVRR, rt, rs
MOVMRR	011101	MOVMRR, rt, rs
MOVRRM	011110	MOVRRM, rt, rs
MOVARR	011111	MOVRRM, rt@B, rs@A
MOVREVRM	100000	MOVREVRM, rt, rs
MOVREVRM	100001	MOVREVRM, rt, rs
MOVREVMRR	100010	MOVREVMRR, rt, rs
MOVREVRRM	100011	MOVREVRRM, rt, rs

(4) The instructions about setting SDMAC as shown in Table. 3-4.

Table. 3-4: Instructions of setting Smart DMA

Instruction	Opcode	Example
SDMAD	100100	SDMAD, data
SDMAR	100101	SDMAD, rs
DMAOK	100110	DMAOK
GDMA	100111	GDMA, address
GDMAR	101000	GDMAR, rt

(5) Other instructions

Table. 3-5: Other instructions

Instruction	Opcode	Example
SHR	101001	SHR, rs
SHL	101010	SHL, rs
SET	101011	SET, rs
GET	101100	GET, rs
ENDC	101101	ENDC

In the Table. 3-5, the instruction of ENDC means the end of program.

3.1.3 I^2S Bus

The I^2S [22] design contains three signals as SCK (serial clock), WS (word select), and SD (serial data). They are explained as follows.

(1) SCK: the transferring clock of each bit which is same as sampling rate.

(2) WS: the signal for switching channel, it indicates the channel being transmitted when WS= 0 (left channel) and WS = 1 (right channel).

(3) SD: the data for transmission.

The bus is a unidirectional design which has one transmitter and one receiver. The transmitter plays a slaver when transferring data, it is controlled by the signal of SCK and WS which sent by the receiver as shown in Fig. 3-2. The receiver plays a slaver when receiving data, it is controlled by the signal of SCK and WS which sent by the transmitter as shown in Fig. 3-3.

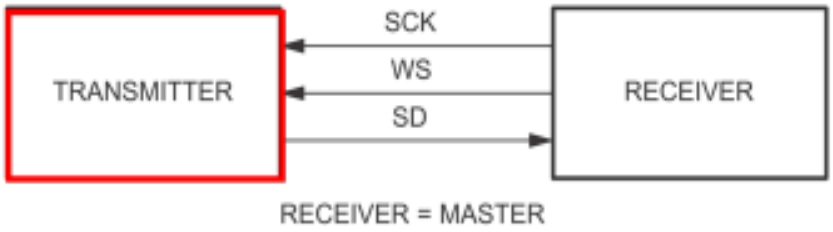


Fig. 3-2: The transmitter of I^2S

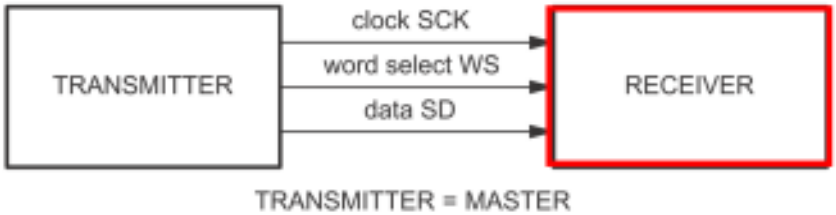


Fig. 3-3: The receiver of I^2S

The I^2S is implemented by a shift register. It has to follow the timing diagram as shown in Fig. 3-4 when transmitting or receiving data.

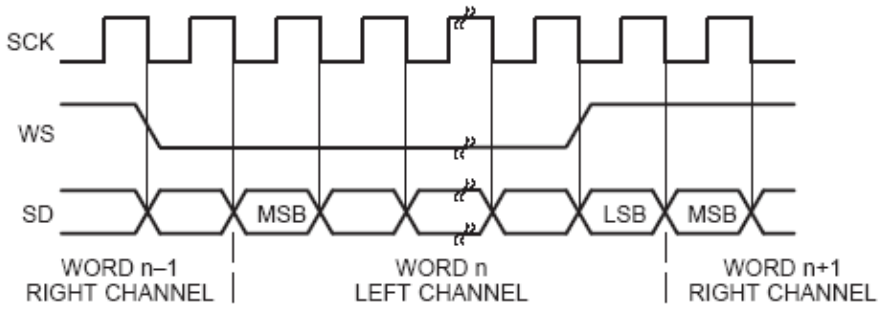


Fig. 3-4: Timing diagram of I^2S

I^2S conforms to the specification of the APB interface. They are seems to two registers at the bus as shown in Fig. 3-5.

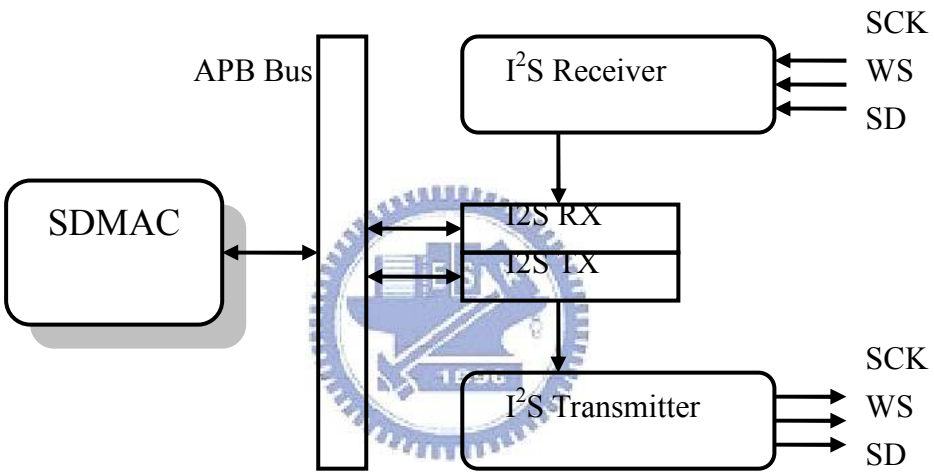


Fig. 3-5: I^2S at APB bus

3.2 Integration of SDMA and Dual-Core Processor

In this thesis, we integrate the proposed SDMAC and the general-purpose dual-core processor as shown in Fig. 3-6 to verify the correctness of SDAMC and the whole system. We also verify functions and performance of SDMA integrated with the dual-core processor. The program memory is put outside, and two data memories are put in the core together. These two data memories and SDMAC share the same data bus to connect each other. The SDMAC supports the ARM peripheral bus (APB).

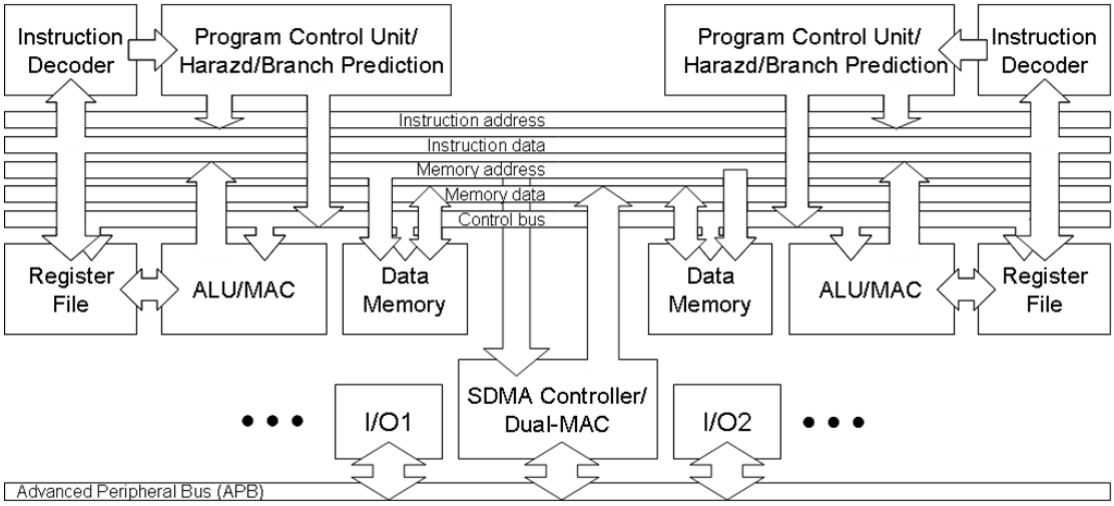


Fig. 3-6: Integration of Smart DMA and dual-core processor

There are four separately bus in the dual-core architecture as shown in Fig. 3-7. Both SDMAC and the dual-core processor share two data buses to access two data memories in the different core. The dual-core processor controls these two data buses to handle data conflict. When the processor does not use instructions of memory reference, a data bus will be released to SDMAC.

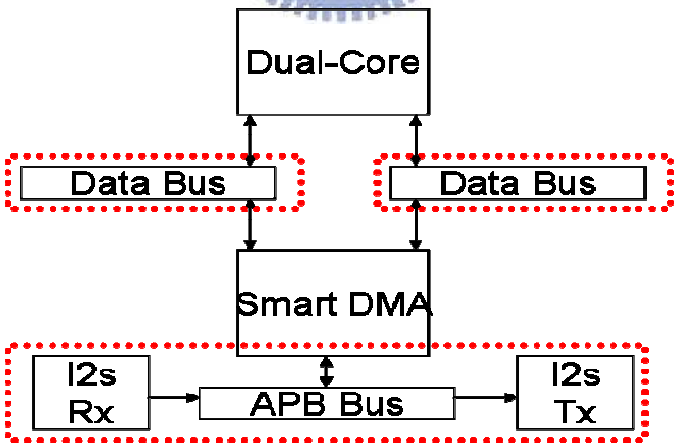


Fig. 3-7: Bus of this system

The processor can directly write the register of SDMAC to set parameters. There are two different ways to integrate two IPs. First, it can add some instructions for access the register of SDMAC. Second, it can use the method of memory mapping as

shown in Fig. 3-8.

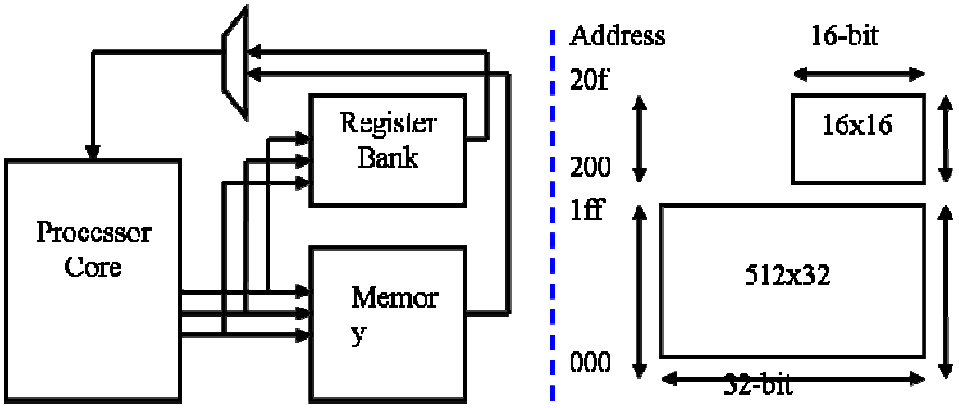


Fig. 3-8: Memory mapping

Chapter 4

Chip Implementation and Verification

This chapter will describe the function verification and experimental results of SDMAC and dual-core processor and also refer the chip realization. Finally, compared its performance with other commercial DSP is discussed in the last subsection.

4.1 Chip Implementation

4.1.1 Design Flow

In this thesis, the cell-based design flow shown in Fig. 4-1 to implement the proposed SDMAC architecture is adopted. First, we write corresponding RTL code in terms of proposed architecture to do functional verification. Then the design is synthesized by UMC 90nm process technology, and the scan chain and memory BIST elements for the testability are also added. If the function of synthesized circuits is correct, the design is automatic placed and routed (APR) by SOC encounter. After APR, the tool DRC and LVS is used to check layout correctness. Finally, post-layout simulation is performed by the tool called NanoSim to verify the whole function of chip design.

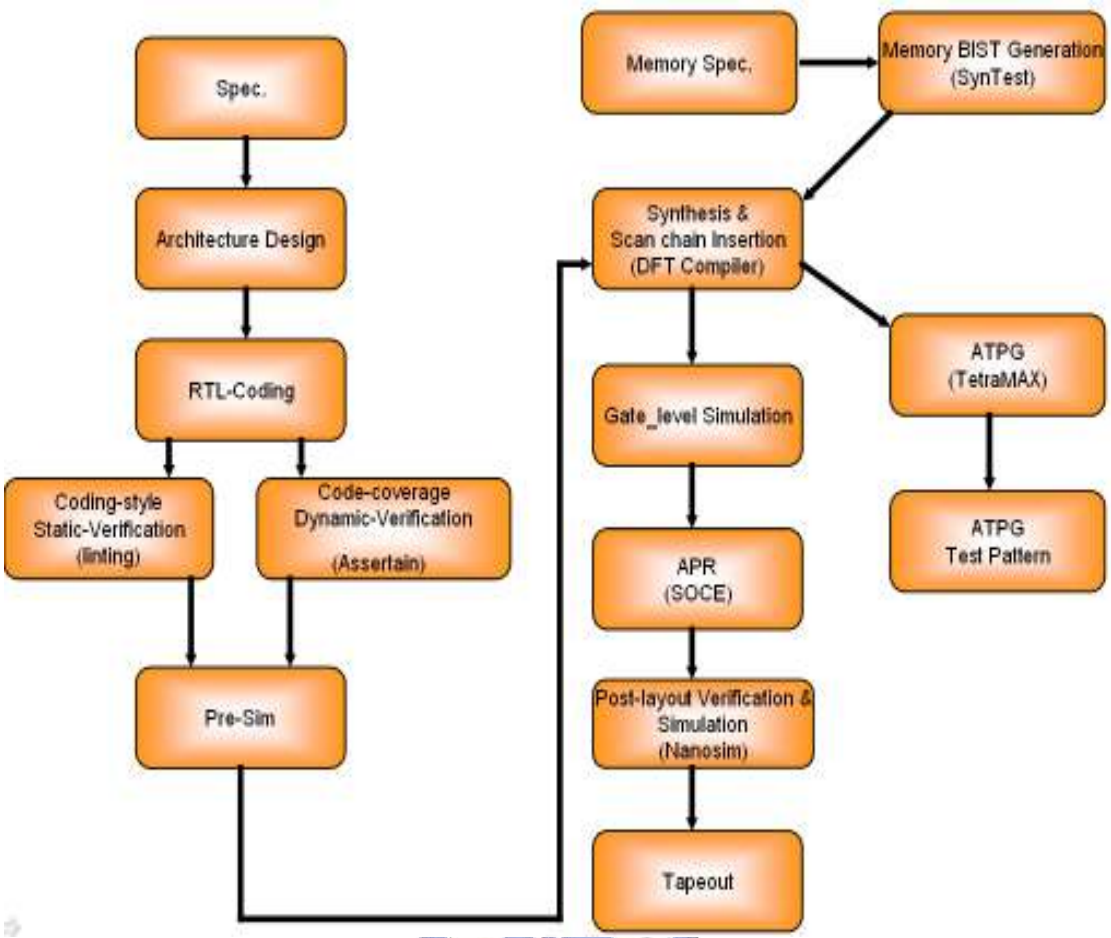


Fig. 4-1: Chip design flow

4.1.2 Synthesis Results

First, we write corresponding RTL code in terms of proposed architecture to verify the design function. Then the design is synthesized by UMC 90nm process technology, and the scan chain and memory BIST elements for the testability are also added. Table. 4-1 lists the synthesis result of the whole chip design.

Table. 4-1: Synthesis Results

ITEM	Area（㎜^{2}）	Timing	Total fault	Fault coverage
Dual-Core Processor+SDMA	2.1x2.1	5ns	317480	95.07 %

4.1.3 Layout and Package

The design is automatic placed and routed (APR) by the tool called SOC encounter. The APR result will be illustrated as follows.

(1) CHIP name：DCSP

(2) Technology：UMC 90nm 1P9M standard CMOS process

(3) Package：144 CQFP

(4) Chip Size：2.1× 2.1 mm^2

(5) Power Dissipation：~100mW

(6) Operation Frequency：200 MHz（5ns）

Using the tool called Prime Power to measures the power consumption which is about 100mW. The layout and the PAD allocation are shown in Fig. 4-2, and the package is shown in Fig. 4-3.

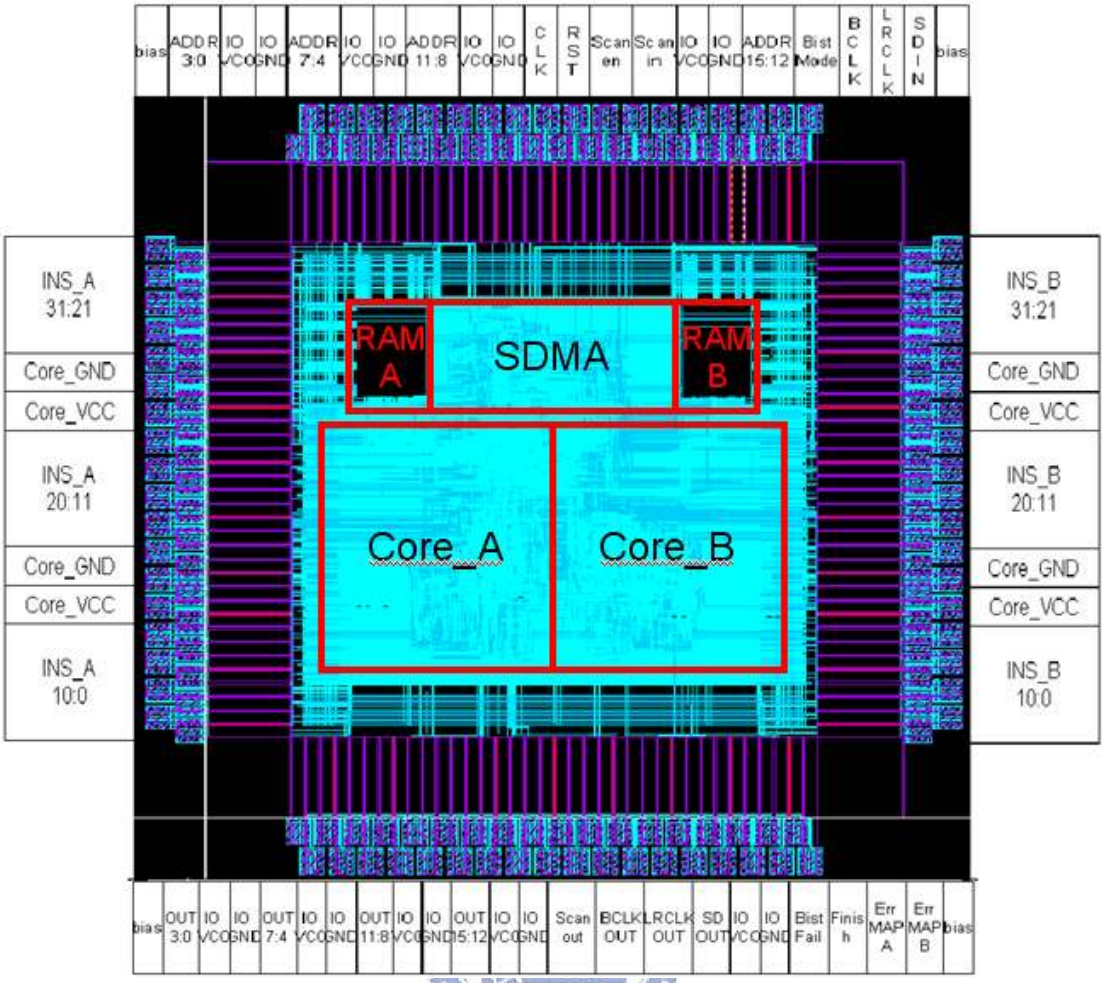


Fig. 4-2: Chip Layout and PAD allocation

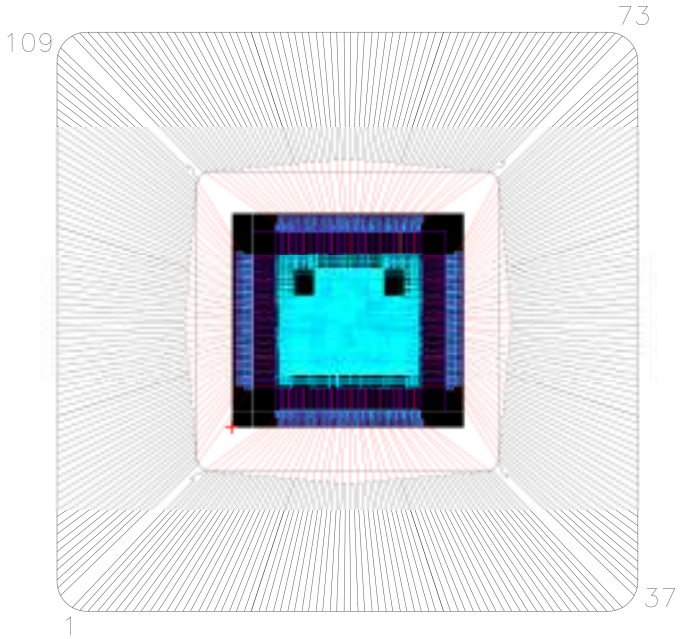


Fig. 4-3 Chip Package

To verify the final chip layout, both DRC (Design Rule Check) and LVS (Layout verse Schematic) are passed. The result is shown in Fig. 4-4 and Fig. 4-5.

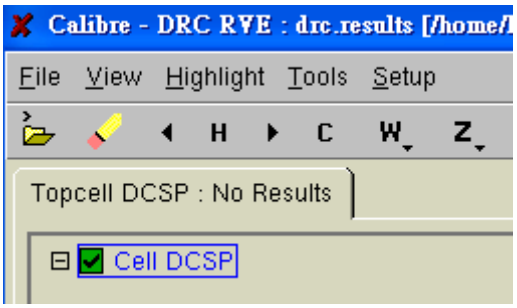


Fig. 4-4: DRC result

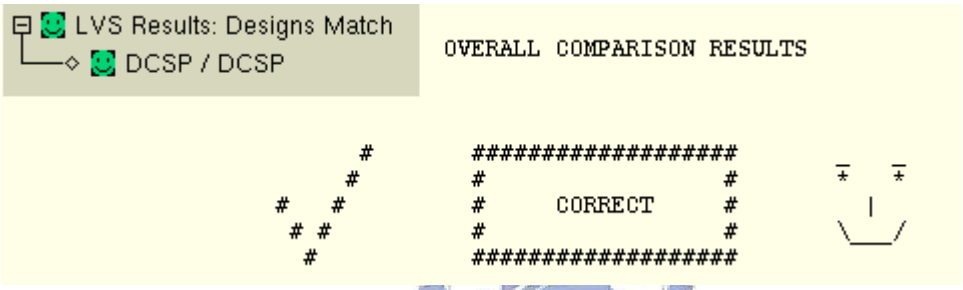


Fig. 4-5: LVS result

Finally, the detailed chip specification is listed in Table. 4-2.

Table. 4-2 Specification Table

Technology	Description
Process	UMC 90nm 1P9M Mixed Signal
Architecture	Dual-Core 5-stage pipeline
Synthesis	Synopsys Design Compiler
Gate Count	253K
Embedded Memory	RAM0，RAM1
Die size	1.85 × 1.85 mm^2
Supply	1.0V/3.3V ± 10%
Input Delay Time	Max 1ns/ Min 0.5ns
Output Delay Time	Max 2ns/ Min 1ns
Output Loading	30pf
Power consumption	100mW@200MHz
Operation Frequency	200MHz
DMA Design	
DMA Channel	2
DMA Request	8
DMA Gate Count	45K
Transfer Type	Memory-to-Memory Memory-to-Peripheral Peripheral-to-Memory Peripheral-to-Peripheral
Compliance with APB bus	Support 8 device
Hardware DMA channels priority	High: Ch0 / Low:Ch1
DMA Special Function	Inner Product Convolution Multiple Addressing Built-in I2S interface

4.1.4 Design for Testing Consideration

There are two types for chip testing. One is the error testing produced by manufacturing. The other is the correctness testing of functions. The previous is about the memory BIST (Built-In Self Test) and the scan-chain insertion of the whole circuit. The following will illustrate about BIST and scan-chain functions.

Because the process is more and more advanced, the gate counts increase in a unit area. To avoid errors that produced by manufacturing, it must add testing circuit to the proposed design.

At the part of memory, using the tool called Srambist to produce self-test circuit in memory and using the Moving Inversion algorithm as shown in Fig. 4-6. This method can test where the built-in memory is failed or not. Each memory has a BIST controller itself, and shares the BistMode signal. When BistMode is high, the two memories are running at the testing mode until testing is finished. The Finish signal will be set to high. If some errors occur, the BistFail signal will be set to high.

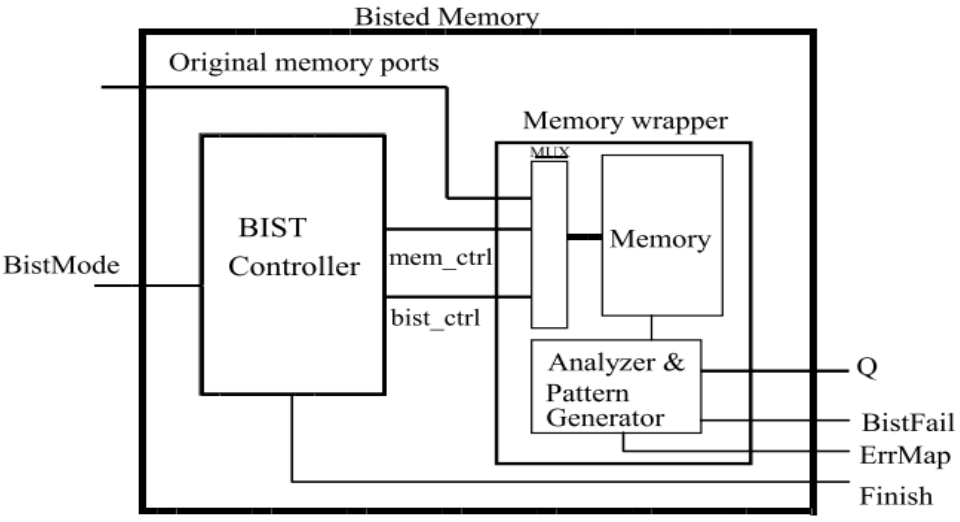


Fig. 4-6: Memory BIST

Using the tool called DFT Compiler adds testing circuit to the proposed design. As can be seen, a scan-chain line is inserted. Finally, the report is generated as shown in Fig. 4-7. The testing pattern data are 904, and the fault-coverage is 95.07%.

```
    Uncollapsed Stuck Fault Summary Report
-----------------------------------------------
fault class                     code   #faults
------------------------------  ----  ---------
Detected                         DT     301735
  detected_by_simulation         DS    (266011)
  detected_by_implication        DI     (35724)
Possibly detected                PT        218
  atpg_untestable-pos_detected   AP       (210)
  not_analyzed-pos_detected      NP         (8)
Undetectable                     UD       2871
  undetectable-tied              UT         (12)
  undetectable-blocked           UB          (2)
  undetectable-redundant         UR       (2857)
ATPG untestable                  AU      12223
  atpg_untestable-not_detected   AN     (12223)
Not detected                     ND        433
  not-controlled                 NC       (177)
  not-observed                   NO       (256)
-----------------------------------------------
total faults                             317480
test coverage                             95.94%
fault coverage                            95.07%
-----------------------------------------------
          Pattern Summary Report
-----------------------------------------------
#internal patterns                          904
    #basic_scan patterns                    886
    #full_sequential patterns                18
-----------------------------------------------
```

Fig. 4-7: Fault-Coverage

For verifying the correctness of functions, we support five patterns and translate to the format of the tester (Agilent 93K). We can see the results from PAD. The testing patterns are data moving, inner product, convolution, DCT, and FFT.

4.2 Chip Verification

We can write program in assembler to verify the functions and performance of SDMAC with a general-purpose dual-core processor. The testing functions involve in data transmission, inner product, convolution, DCT, and FFT. The following subsection will illustrate in details.

4.2.1 Data Transmission

Data transmission consists of four types. This can verify the correctness of transferring in different memories of the dual-core processor architecture.

(1) Architecture of data transmission in different memories as shown in Fig. 4-8.

(2) Architecture of data transmission in the same memory as shown in Fig. 4-8: Transmission in different memories .

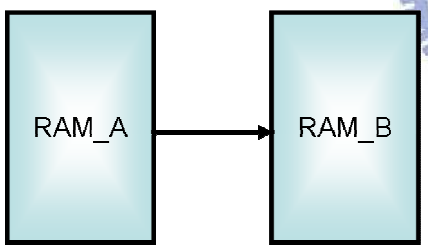


Fig. 4-8: Transmission in different memories

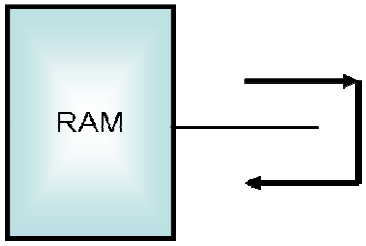


Fig. 4-9: Transmission in a same memory

(3) Architecture of data transmission in different memories with circular addressing as shown in Fig. 4-10.

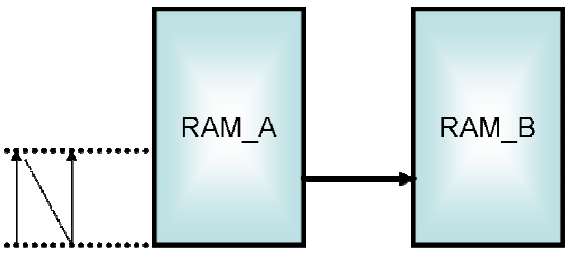


Fig. 4-10: Data transmission in different memories by circular addressing

(4) Architecture of data transmission in different memories with mirror addressing as shown in Fig. 4-11.

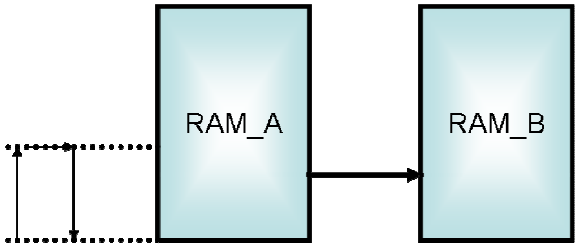


Fig. 4-11: Data transmission in different memories by mirror addressing

4.2.2 Inner product

In Fig. 4-12, the data in memory is the same as the memory address, and we set two memories (RAM_A and RAM_B) with increment addressing to compute the inner product. At this time, the dual-MAC operates at the real-valued-MAC mode. The hardware can automatically compute the result of Eq. (4.1), and post-layout simulation is shown in Fig. 4-13.

$$\sum_{i=1}^{510} i^2 = 44347135 \tag{4.1}$$

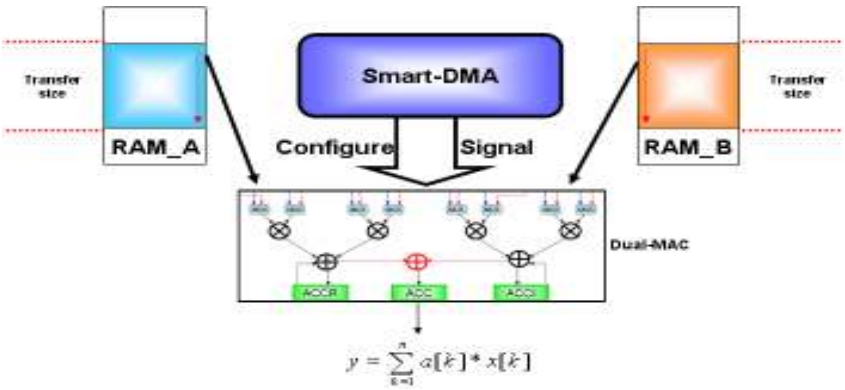


Fig. 4-12:Inner product

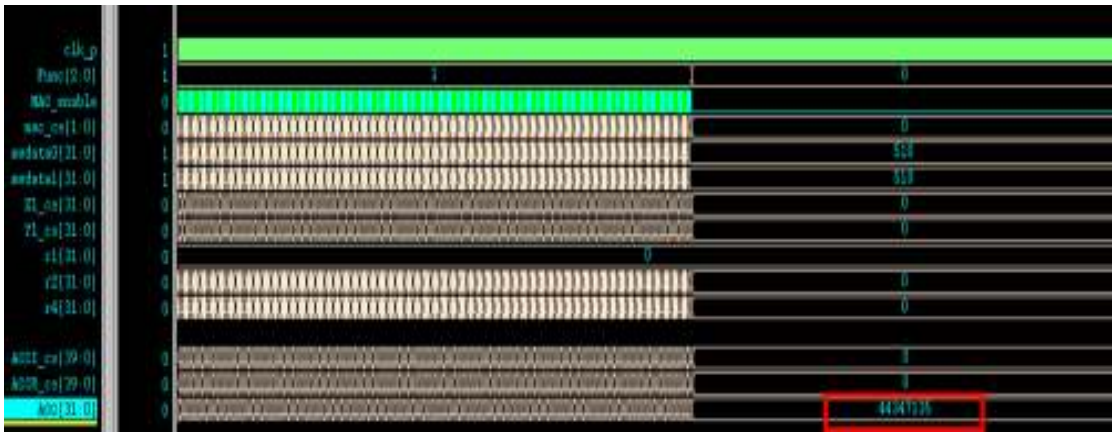


Fig. 4-13: Post-layout simulation of inner product

4.2.3 Convolution

In Fig. 4-14, the data in memory is the same as the memory address, and we set one memory with increment addressing and the other with decrement addressing to compute the linear convolution. At this time, the dual-MAC operates at the real-valued-MAC mode. The hardware can automatically compute the result of Eq. (4.2), and post-layout simulation is shown in Fig. 4-15.

$$y[n] = \sum_{k=1}^{510} (510-k) * k = 22108415 \tag{4.2}$$

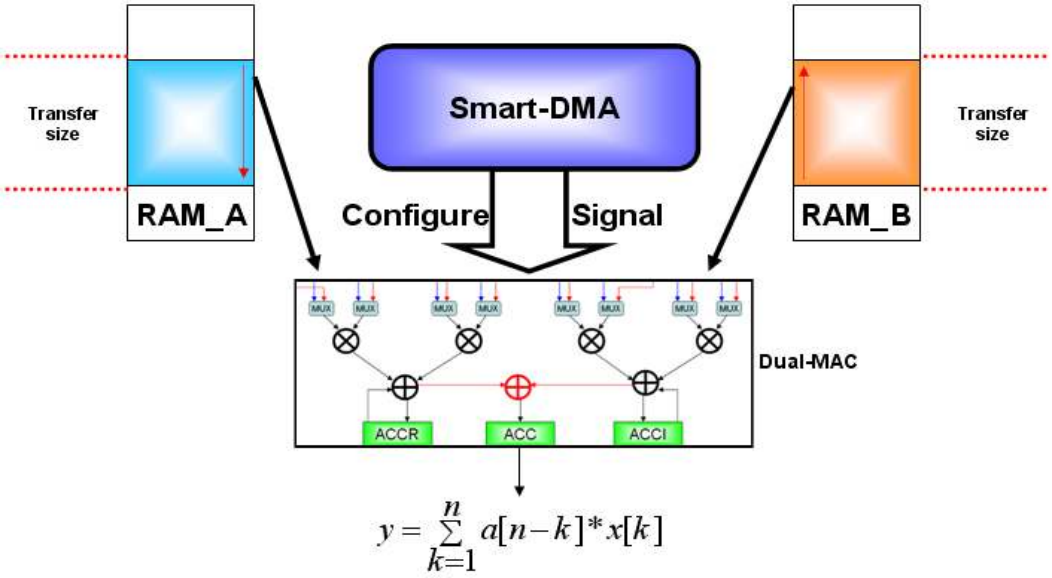


Fig. 4-14: Convolution

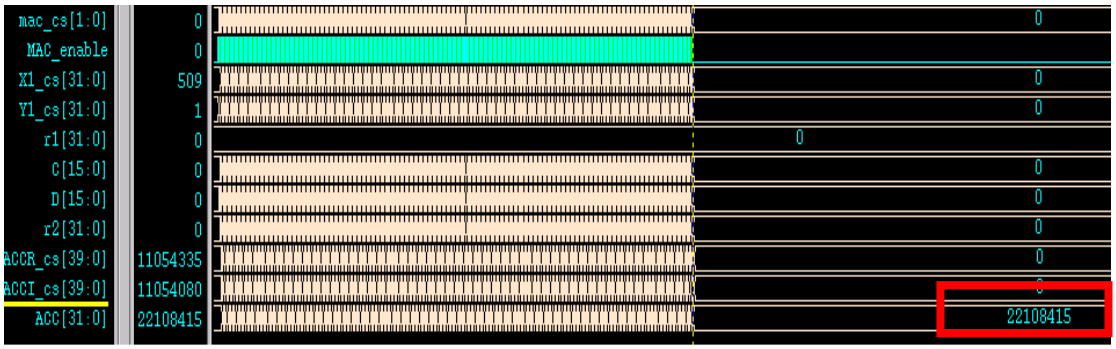


Fig. 4-15: Post-layout simulation of convolution

4.2.4 FFT

The SDMA computes 32-point FFT operation with a 16-bit fixed-point format. Fig. 4-16 and Fig. 4-17 show the post-layout simulation result and the MATLAB result, respectively.

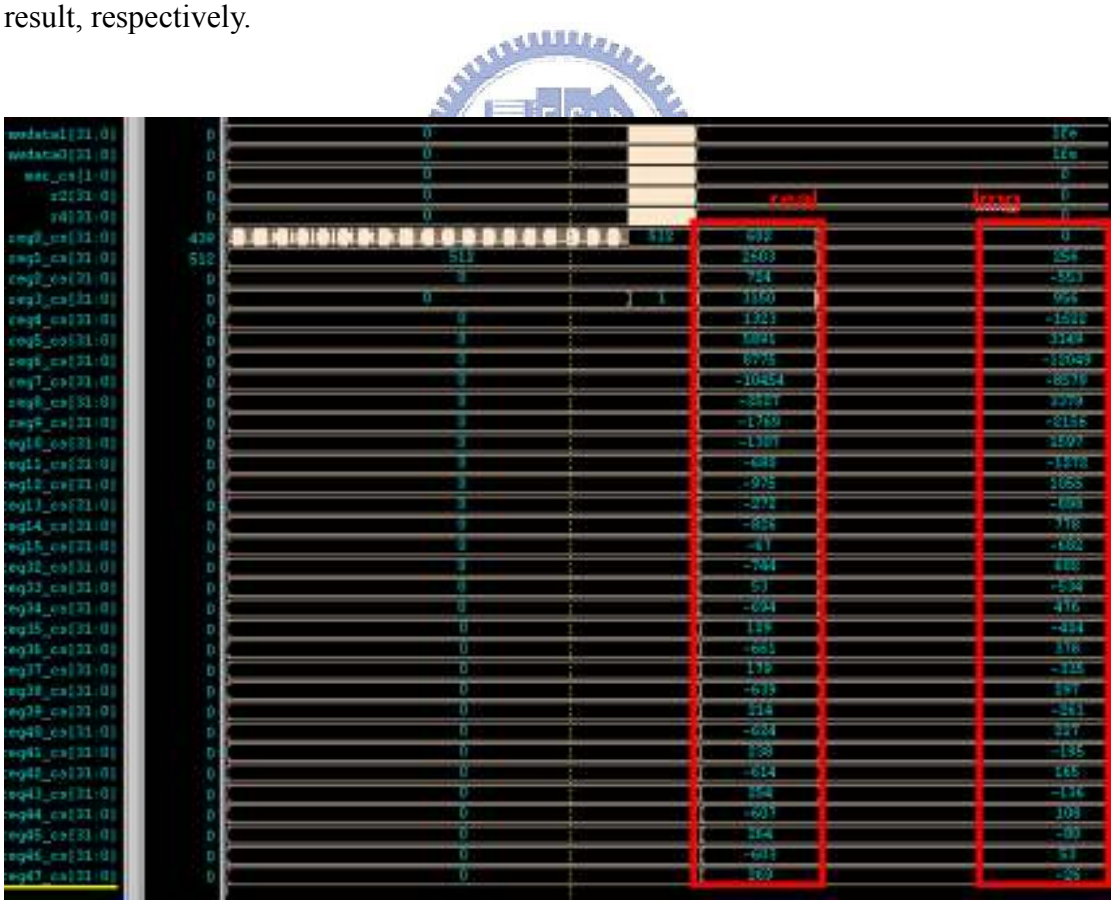


Fig. 4-16: Post-layout simulation of FFT

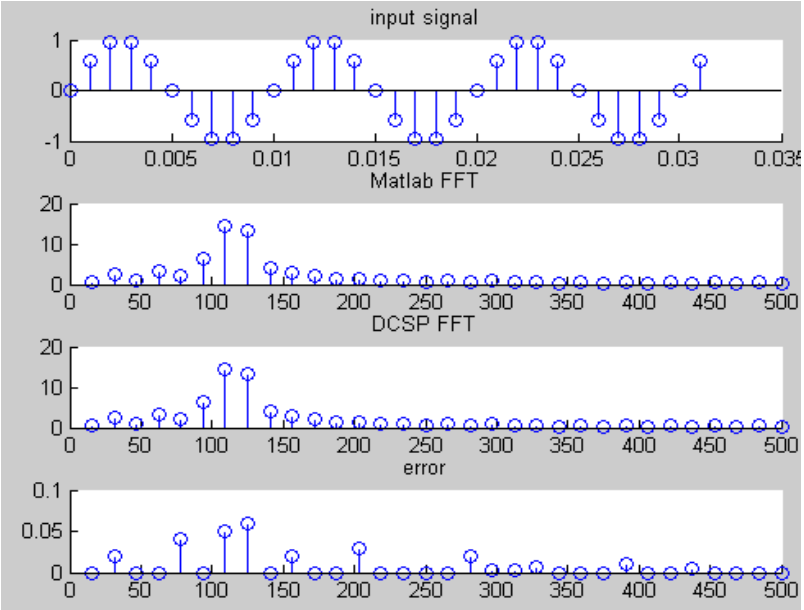


Fig. 4-17: MATLAB result of FFT

4.2.5 DCT

The SDMA computes 36-point DCT operation with a 16-bit fixed-point format. The input data is shown in Fig. 4-18. Fig. 4-19 and Fig. 4-20 show the post-layout simulation result and the MATLAB result, respectively.

```
x =

  Columns 1 through 18

    1     1    -1    -1     2     2    -2    -2    -7    -7    -7    -7     6     6    -6    -6     5     5

  Columns 19 through 36

    5     5     4     4     4     4     3     3     3     3     2     2     2     2     1     1     1     1
```

Fig. 4-18: Input data of DCT

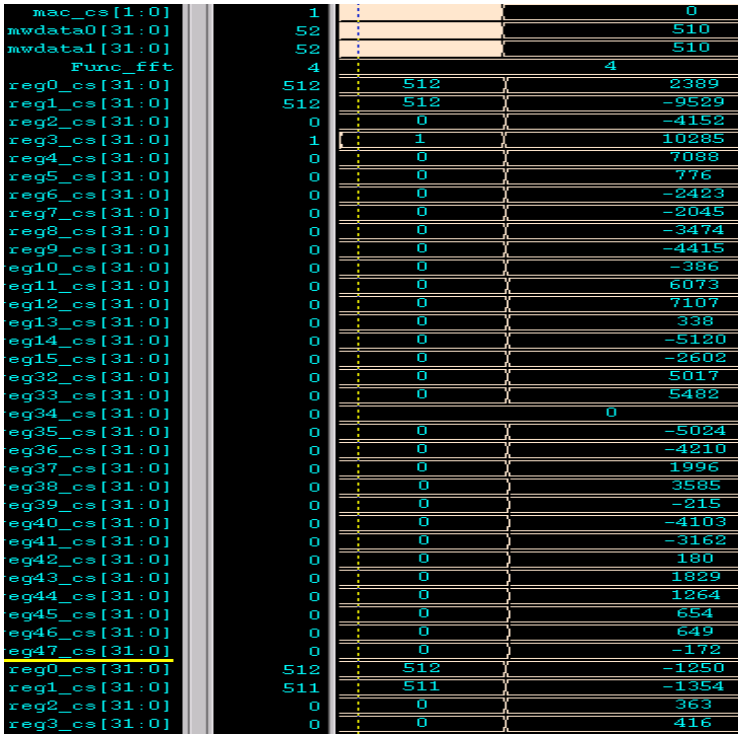


Fig. 4-19: post-layout simulation of DCT

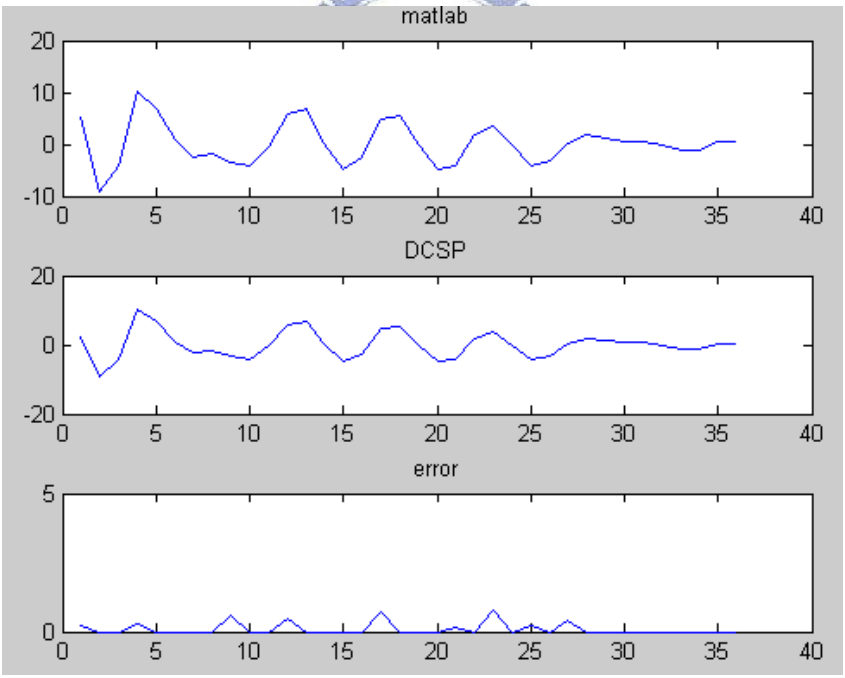


Fig. 4-20: MATLAB result of DCT

4.2.6 Peripheral Interface - APB

The peripheral interface bus called APB is often used in data transferring of peripheral devices, such as I^2S (Inter-IC Sound). The architecture of data transmission

form I^{2}S to memory is shown in Fig. 4-21. The post-layout simulation shows in Fig. 4-22.

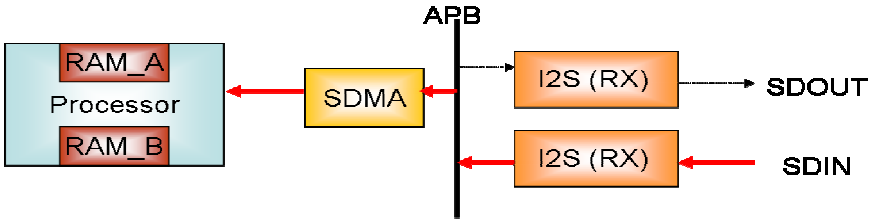


Fig. 4-21: Transferring data form I^{2}S to memory

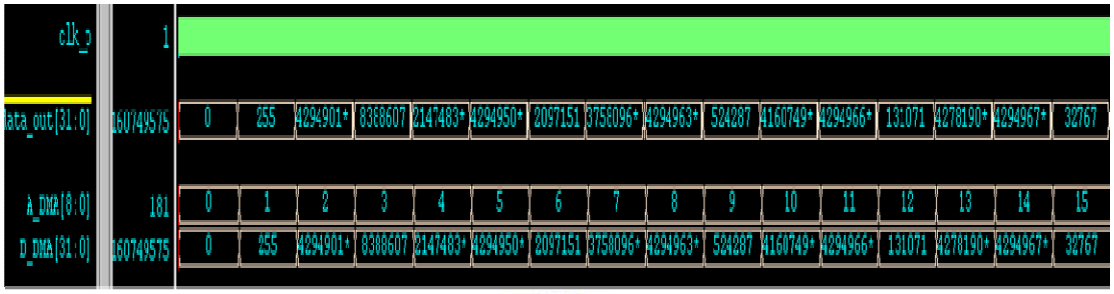


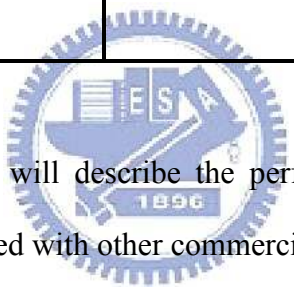
Fig. 4-22: Post-layout simulation of data transmission form I^{2}S to memory

4.2 Performance Comparison

In this thesis, the SDMAC, which is different from traditional DMA, is proposed and designed. The traditional DMA only supports fewer addressing modes and does not have built-in arithmetic units. For the proposed design, it has not only multiple addressing modes, but also a built-in dual-MAC. Hence, it is suitable for a lot of DSP operations. SDMA can compute at the same time during data transmission. So, this method can greatly improve performance of the general-purpose processor. As shown in Table. 4-3, the proposed design is compared with the commercial DMA such as Faraday [9] and Global Unichip [24].

Table. 4-3 Comparison of commercial DMA and Smart DMA

	FARADAY [9]	GLOBAL UNICHIP [24]	This work
	FTDMAC020	UAPC5110	DCSP
Channel	8	2(8)	2
Request	8	4(32)	8
Transfer Type	M-to-M M-to-P P-to-M	M-to-M M-to-P P-to-M P-to-P	M-to-M M-to-P P-to-M P-to-P
Addressing	Chain Transfer	Increase/Decrease	Increase/Decrease Circular Mirror Index-Base Bit-Reverse
Special Function	None	None	Dual-MAC



The following subsection will describe the performance on data moving and arithmetic computation compared with other commercial DSP.

4.3.1 Performance of Data Transmission

We compare the processor that has not SDMAC and has SDMAC, and the performance of data transmission is shown in Fig. 4-23, where A2A indicates from memory A to A, and A2B indicates from memory A to B. As can be seen, the SDMAC can help the general-purpose processor to do data transmission efficiently in lots of addressing.

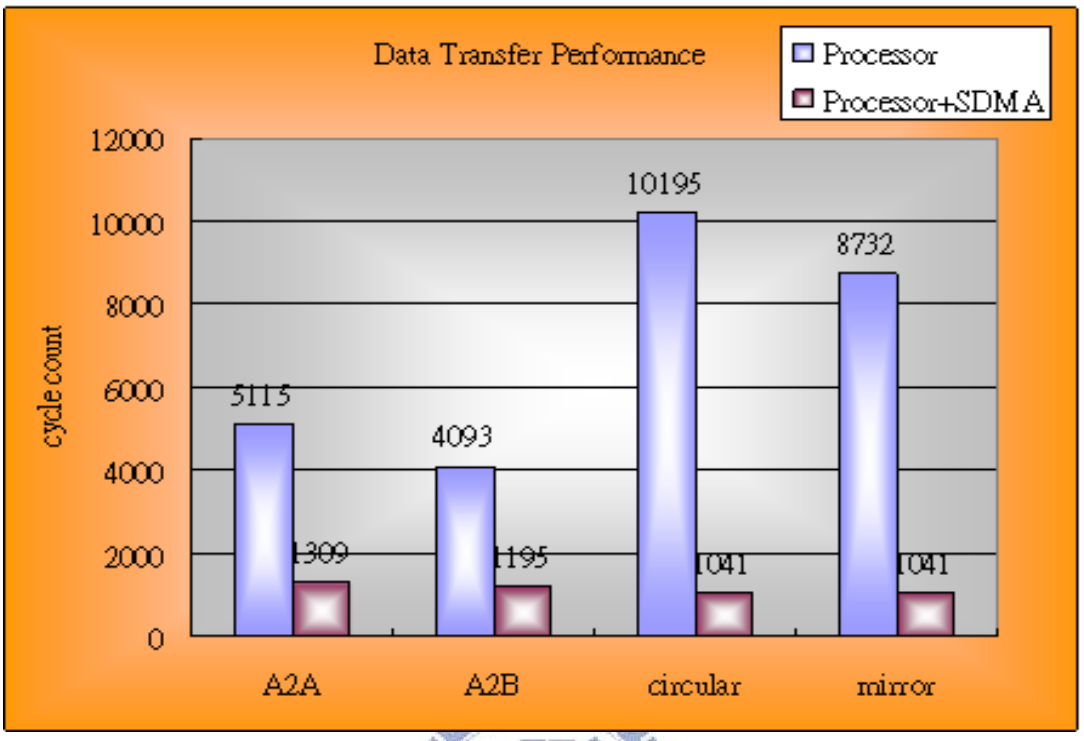


Fig. 4-23: Performance of data moving

4.3.2 Comparison the MAC performance with other DSPs

The MAC operation is a basic unit for DSP functions. This thesis presents a dual-MAC architecture put into SDMA. This architecture in a cycle can compute a complex-valued operation or a butterfly unit. As shown in Table. 4-4, we compare the execution cycles of dual-MAC with other DSPs.

Table. 4-4: Comparison of MAC operation with other DSPs

	TI C62X	LODE [13]	DSP1600 [14]	MDSP-II [15]	CDSP [16]	Hiroyuki JSSC98 [27]
Complex-MAC	No	No	No	No	Yes	Yes
Complex-MAC	2 cycles	2 cycles	2 cycles	2 cycles	4 cycles	2 cycles
Real-MAC	1/2 cycles	1/2 cycles	1/2 cycles	1 cycles	1/2 cycles	1/4 cycles
Butterfly	6 cycles	4 cycles	4 cycles	5 cycles	8 cycles	3 cycles

	Hinrichs JSSC2000 [28]	Ackland JSSC2000 [29]	Olofsson ISSCC2002 [30]	Agarwaral ISSCC2002 [31]	**This Work**
Complex-MAC	No	No	No	No	**Yes**
Complex-MAC	2 cycles	2 cycles	4 cycles	2 cycles	**1 cycles**
Real-MAC	1/2 cycles	1/2 cycles	1/2 cycles	1/2 cycles	**1/4 cycles**
Butterfly	3 cycles	3 cycles	8 cycles	3 cycles	**1 cycles**

4.3.3 Comparison DSP functions with other DSPs

For a traditional DMA of the general-purpose processor, it can not perform DSP functions efficiently, but it can be improved to be SDMA. The SDMAC can assist the dual-core processor to handle mass and regular operations such as FIR, DCT, FFT, etc. The computational performance of FIR, and DCT is compared and shown in Fig. 4-24 and, where the R-FIR means real-valued FIR, and C-FIR means complex-valued FIR.

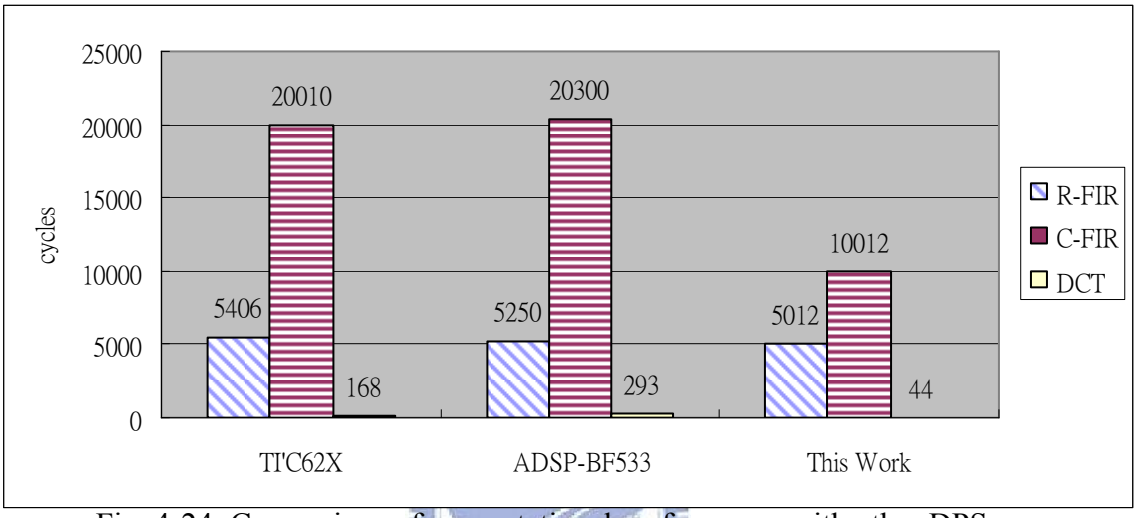


Fig. 4-24: Comparison of computational performance with other DPSs

Fig. 4-25 shows lower cycles than other processors computing in 256-point complex-valued FFT operations with using dual- MAC architecture. The complex FFT operation is a significant DSP algorithm which is often published for a variety of processors.

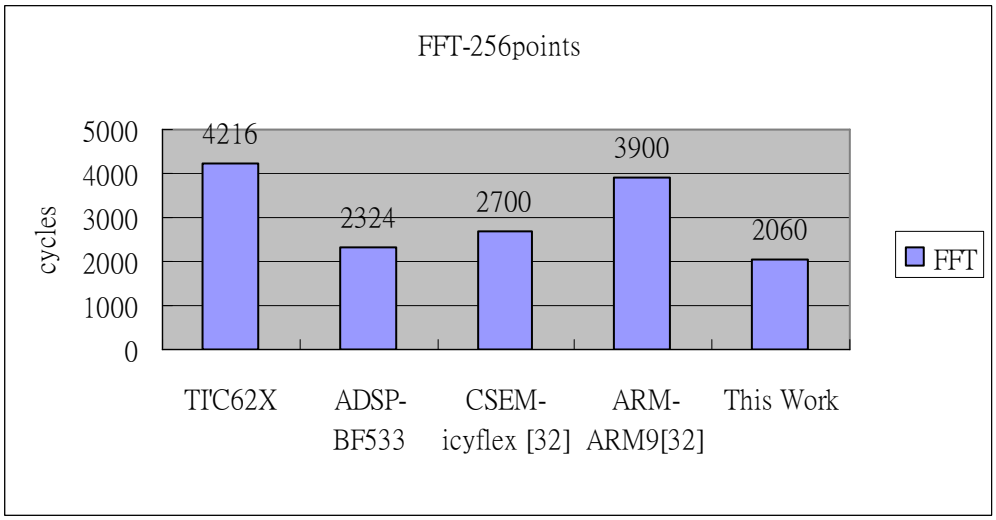


Fig. 4-25: FFT Execution Cycles

Chapter 5

Conclusions and Future Works

This thesis proposes an efficient SDMAC design which is integrated into the general-purpose dual-core processor in order to turn into a DSP-like processor chip. The main purpose is to increase the additional value when data transmitting, i.e., arithmetic computation.

The SDMA supports five addressing modes compared with the design method of traditional DMA and four transmission types to select the region of valid data and to reduce the transmission bandwidth for the processor. The SDMA design has features as follows. (1) It has a built-in dual complex-valued multiplication-and-accumulation (Dual-MAC) to processes mass and regular data computation. Moreover, two channel can access two memory banks and perform vector operations at the same time; (2) it supports the peripheral bus to expand I/O devices flexibly; (3) it can save about 75% time wasted on data transfer; and (4) the code size can be reduced.

By experimental results, the proposed dual-core processor design with SDMA can achieve greatly efficiency at FFT, DCT, and FIR computation, especially in complex operations. Compared with TI C62X series, when the processor is running real- or complex-valued FIR operations, it will improve efficiency about 7% to 50%. The dual-core processor with SDMA has been integrated in the total area of 4.41 mm^2 by using UMC 90nm standard CMOS technology and has fabricated via the National Chip Implementation Center (CIC). The maximum clock frequency is at 200MHz with a single 1.0V supply.

In the future, this chip can be integrated with AHB and other ASIC (application specify integrated circuit) or ADC to become a SoC (system-on-chip) as shown in

Fig. 5-1. Users can write a program on this chip according to the algorithm which they wanted. On the other hand, the proposed architecture has not a friendly I/O interface. In the next generation, we can also add other serial communication interfaces, for example, I2C, Uart, SPI, etc. We can also increase the channels to

enhance the peripheral transmission.

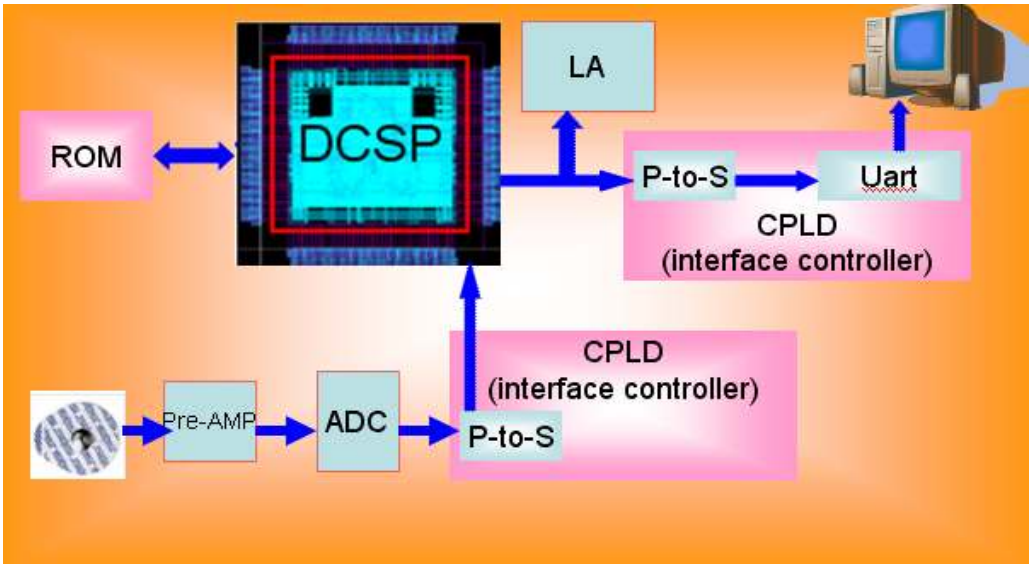


Fig. 5-1: System level design

References

[1] Dave Comiskey, Sanjive Agarwala, and Charles Fuoco, "A scalable high-performance DMA architecture for DSP application," *Proceedings of the IEEE International Conference on Computer Design (ICCD*), pp. 414-417, 2000.

[2] C. M. Yuen, K. F. Tsang, and W. H. Chan, "Direct memory access frequency synthesizer for channel efficiency improvement in frequency hopring communication," *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS 2000)*, pp.485-488, 2000.

[3] Mattias O'Nils and Axel Jantsch, "Synthesis of DMA controllers from architecture independent descriptions of HW/SW communication protocols," *Proceedings of IEEE International Conference on VLSI Design (ICVD)*, 1999.

[4] B. Sprunt, D. Kirk, L. Sha, "Priority-driven, preemptive I/O controllers for real-time systems," Proceedings of the 15th Annual International Symposium on Computer architecture, pp. 152-159, 1988.

[5] Texas Instrument, TMS320C6000 Peripherals Reference Guide.

[6] S. Srinivasan and D. B. Stewart, "High Speed Hardware-Assisted Real Time Interprocess Communication for Embedded Microcontrollers," Proceedings of the 21st Real-Time Systems Symposium, pp. 269-279, 2000.

[7] Motorola Inc, PowerPC™ MPC823e Reference Manual, Website at http://e-www.motorola.com/.

[8] Analog Devices Inc, ADSP-BF533 Blackfin™ Processor Hardware Reference, Website at http://www.analog.com/static/imported-files/data_sheets/.

[9] Faraday, "Direct Memory Access Controller: Faraday/UMC FTDMAC020," rev. 1.2, www.faraday.com.tw, 2003.

[10] Vijay K. Madisetti, VLSI Digital Signal Processors: An Introduction to Rapid Prototyping and Design Synthesis, IEEE Press, 1995.

[11] Alan V. Oppenheim, Ronald W. Schafer, and John R. Buck, Discrete Time Signal Processing, 2nd Edition, Prentice Hall, 1999.

[12] ARM Ltd, AMBA Specification, rev. 2.0, http://www.arm.com, 1999.

[13] I. Verbauwhede, M. Touriguian, K. Gupta, J. Muwafi, K. Yick, and G. Fettweis, "A low power DSP engine for wireless communications," *in Proc. Workshop VLSI Signal Processing*, vol. IX, pp. 471–480,1996.

[14] M. Alidina, G. Burns, C. Holmqvist, E. Morgan, D. Rhodes, S. Simanapalli, and M. Thierbach, "DSP16000: A high performance, low power dual-MAC DSP core for communication applications," *in Proc. IEEE Custom Integrated Circuits Conf.*, pp. 119–122, May 1998.

[15] T.S. Rappaport, Wireless Communications, Principles and Practice, 2nd Englewood Cliffs, NJ: Prentice Hall, 2002.

[16] B. W. Kim, J. H. Yang, C. S. Hwang, Y. S. Kwon, K. M. Lee, I. H. Kim, Y. H. Lee, and C. M. Kyung, "MDSP-II: A 16-bit DSP with mobile communication accelerator," *IEEE J. Solid-State Circuits*, vol. 34, pp.397–403, Mar. 1999.

[17] John L. Hennessy and David A. Patterson, Computer Architecture, 3rd Edition, Morgan Kaufmann, 2003.

[18] John L. Hennessy and David A. Patterson, Computer Organization & Design：The Hardware / Software Interface, 2nd Edition, Morgan Kaufmann Publishers, 1998.

[19] A. Kalavade and E. A. Lee, "A hardware-software codesign methodology for DSP application," *IEEE Design & Test of Computers*, vol. 10, pp. 16-28, Sept. 1993.

[20] Steve Fuber, ARM System-on-Chip Architecture, 2nd edition, Addison-Wesley Professional, 2000.

[21] Hans-Joachim Stolberg, Mladen Berekovic, Lars Friebe, Sören Moch, Mark Bernd Kulaczewski and Peter Pirsch, "HiBRID-SoC: A Multi-Core System-on-Chip Architecture for Multimedia Signal Processing Applications," *Proceedings of International Conference on Very Large Scale Integration of System-on-Chip*, pp. 155-160, 2003.

[22] Philips Semiconductor, I2S bus Specification, 1996. Website Available: http://www.semiconductors.philips.com.

[23] Donald E. Thomas and Philip Moorby, The Verilog Hardware Description Language, Kluwer Acadmic Publishers, 1994.

[24] GlobalUnichip, "UAPC-5110 DMA Controller Lite," 2002. Website Available: http://www.globalunichip.com.tw.

[25] TMS32C62x Technical Overview. Texas Instruments Inc.Dallas, TX. Website Available: http://www.ti.com/

[26] C. K. Chen, P. C. Tseng, Y. C. Chang, and L. G. Chen, "A digital signal processor

with programmable correlator array architecture for third generation communication system," *IEEE Trans. Circuits Syst. II*, vol. 48, pp. 1110–1120, Dec. 2001.

[27] H. Igura, Y. Naito, K. Kazama, I. Kuroda, M. Motomura, and M.Yamashina, "An 800-MOPS, 110-mW, 1.5-V, parallel DSP for mobile multimedia processing," *IEEE J. Solid-State Circuits*, vol. 33, pp. 1820–1828, Nov. 1998.

[28] W. Hinrichs, J. P. Wittenburg, H. Lieske, H. Kloos, M. Ohmacht, and P. Pirsch, "A 1.3-GOPS parallel DSP for high-performance image-processing applications," *IEEE J. Solid-State Circuits*, vol. 35, pp. 946–952, July 2000..

[29] B. Ackland et al., "A single-chip 1.6-billion 16 bMAC/s multiprocessor DSP," *IEEE J. Solid-State Circuits*, vol. 35, pp. 412–424, Mar. 2000.

[30] A. Olofsson and F. Lange, "A4.32 GOPS 1Wgeneral-purpose DSP with an enhanced instruction set for wireless communication," *in IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, vol. 2, pp. 36–389, 2002.

[31] S. Agarwala et al., "A 600 MHz VLIW DSP," *in IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, vol. 2, pp. 38–389, 2002.

[32] C. Arm et al., "Low-power 32-bit dual-MAC 120 μW/MHz 1.0 V icyflex DSP/MCU core," *34th European Solid-State Circuits Conference*, pp. 190-193, Sept. 2008.

Appendix

A Tapeout Review Form

1. 晶片概述：
   - *1-1* 專題名稱：___*智能型 DMA 的 DSP 架構設計在雙核心上的應用*___
   - 1-2 Top Cell 名稱：___*DCSP*___
   - 1-3 使用 library 名稱：
     - _*v*_ CIC_CBDK90
     - ___ CIC_CBDK18
     - CBDK 版本: _____
     - 是否使用 Core Cell: _*Yes*_ 若有使用 Core Cell 型號: □hvt V rvt □lvt
     - Core Cell 有無更改 Cellname: ____ (建議保留勿更改)
     - 是否使用 IO: _*Yes*_ 若有使用 IO, 採用形式是：□Linear V Staggered
   - 1-4 是否使用 CIC 提供之 Memory？ _*Yes*_ 若使用 Memory, 是否已上傳 spec 檔: _*Yes*_
     - 使用 Memory 之種類爲何？ ___*Synchronous Single-Port Register File*___
   - 1-5 是否使用 CIC 提供之 ARM CPU IP？ _____(若爲 Yes, 請務必填寫第 9 項)
     - 使用 CPU 之種類爲何？(ARM7TDMI or ARM926EJ) _____
   - 1-6 工作頻率：___*200Mhz*___
   - 1-7 功率消耗：___*100mW*___
   - 1-8 晶片面積：___*1.85mm*___X___*1.85mm*___

2. 設計合成：
   - 2-1. 使用之合成軟體？___*Synopsys Design Compiler*___
   - 2-2. 是否加入 boundary condition：
     - _*v*_ input drive strength、_*v*_ input delay、_*v*_ output loading、_*v*_ output delay
   - 2-3. 是否加入 timing constraint：
     - _*v*_ specify clock (sequential design)
     - _*v*_ max delay、_*v*_ min delay (combinational design)
   - 2-4. 是否加入 area constraint？ ___*No*___
   - 2-5. 合成後之 report 是否有 timing violation？ _*No*_
     - ___ 有 setup time violation、___ 有 hold time violation
   - 2-6. 合成後之 verilog 是否含有 assign 描述？ ___*No*___
   - 2-7. 合成後之 verilog 是否含有 *cell* 之 instance name？ ___*No*___
   - 2-8. 合成後之 verilog 是否含有反斜線 \ 之 instance name 或 net name？ ___*No*___

3. 可測試性設計(前瞻性晶片必填)：
   - 3-1. 使用之設計軟體？___*DFT Compiler*___
   - 3-2. 使用之 ATPG 軟體？___*TetraMAX*___
   - 3-3. 使用 Embedded memory 數量: SRAM___*2*___，ROM_____
     - Memory 大小:_*512x32 (Word x bit)*_
     - 測試方法: BIST___*Yes*___，or 其他測試方法_____
     - 若使用 BIST,其 Test Algorithm 爲何?___*Moving Inversion (13N March)*___
     - 同時有多個 memory，是否共用 BIST controller ___*No*___，BIST controller 數量_*2*_
   - 3-4. Scan Chain Information
     - Flip-Flop 共有多少個？___*4863*___
     - Scan chain 的數量共有多少條？___*1*___
     - Scan chain length (Max.) ？___*66087.41*___
   - 3-5. Uncollapsed fault coverage 是否超過 90% ？___*Yes*___，爲多少？___*95.07%*___

ATPG pattern 的數目爲多少？ ***904***

註：若使用 Synopsys TetraMAX 來產生 ATPG pattern，請使用 `set faults -fault_coverage` 指令指定 TetraMAX 產生 fault coverage information
若使用 SynTest TurboScan 之 asicgen 來產生 ATPG pattern，請以 atpg pessimistic fault coverage 的值爲準

4. 佈局前模擬
   4-1. gate level simulation 是否有 timing violation？ ***No***
   ___ 有 setup time violation、___ 有 hold time violation

5. 實體佈局
   5-1. 使用之 P&R 軟體？ ___ Apolo、***v*** SOC Encounter
   5-2. power ring 寬度？ ***8um*** 是否已考量 current density(1mA/1um)？ ***Yes***
   5-3. 是否考慮 output loading？ ***Yes***
   5-4. 是否加上 Clock Tree？ ***Yes***
   5-5. 是否加上 Corner pad？ ***Yes***
   5-6. IO Buffer 間是否加上 IO Filler: ***Yes*** IO Filler 寬度: ***12*** um (建議至少需 12um 寬)
   5-7. 是否加上 Core Filler？ ***Yes***
   5-8. 是否上加 Bonding Pad？ ***Yes***
   以下(A-1)爲使用 Apollo 者才須回答
   A-1. 是否執行 Fill Notch and Gap 步驟？ _____
   以下(S-1 至 S-2)爲使用 SOC Encounter 者才須回答
   S-1. power ring 上是否有 overlap vias？ ***No***
   S-2. 是否確定 IO Row 和 Corner Row 互相貼齊？ ***Yes***

6. 佈局後模擬
   6-1. 是否做過 post-layout gate-level simulation？ ***Yes***
   STA(static timing analysis) 軟體？ ***Primetime /ncverilog***
   6-2. 是否做過 post-layout transistor-level simulation？ ***Yes***
   6-3. 已針對以下環境狀態模擬：___ SS、***v*** TT、___ FF
   6-4. 晶片取得時將以何種方式進行測試？ ***CIC 測試機台（Agilent 93K）***
   6-5. 模擬時是否考量輸出負載影響？ ***Yes*** 若有輸出負載是: ***30*** pF (建議至少需 20pF)

7. DRC/LVS 驗證
   7-1. 是否有 DRC 錯誤？ ***Yes*** 錯誤原因： ***有 4.1.1GDEN_LT25 的錯誤, 這是因爲我們的 Standard Cell 是不完整的 Layout,所以會有 DIFF density 不足的錯誤***
   驗證 DRC 軟體？ ***Calibre***
   是否有不作 DRC 的區域？ ***No***
   7-2. 是否有 LVS 錯誤？ ***No***
   驗證 LVS 軟體？ ***Calibre***
   是否有非 CIC 提供的 BlackBox？ ***No***
8. MT Form 填寫
   8-1. 是否填上 ___ 系所單位、___ 設計者姓名、___ 聯絡電話(與手機) 、___ 日期
   8-2. 是否填上晶片上傳目錄？ _____
   8-3. 是否填上檔案名稱？ _____
   8-4. 是否寫上 top cell name？ _____
9. 使用 ARM926EJ or ARM7TDMI CPU IP
   9-1. 若有使用 ARM926EJ /ARM7TDMI CPU IP，請提供以下訊息以便向 ARM 原廠申請 Design ID。

使用的 CPU 種類 (ARM926EJ or ARM7TDMI) :_____
使用的 metal layers 的層數:_____
佈局中 ARM926EJ /ARM7TDMI Macro 的 cell name:_____
這個晶片是否爲修訂版本(revision,也就是之前曾下線過相同晶片)？_____
若是修訂版本，前一次下線的晶片編號:_____
修訂版本的原因是？(例如修正 bug) _____

設計者簽名:_____郭昕展_____ 指導教授簽名:_____林進燈_____