Design and Implementation of an Advanced DMA Controller on AMBA-Based SoC

Guoliang Ma* and Hu He

Abstract —*In this paper, we have proposed a design and implementation of an AMBA based advanced DMA controller. The DMAC has 8 channels which support hardware and software triggers, linking operation and channel chaining transfer and provides three dimensions transmission by parameter sets so as to perform data block moving, data sorting and subframe extraction of various data structures. Channel arbitration mechanism adopts hardware priority combined with weighted priority rotational algorithm. Moreover the DMAC supports incrementing and wrapping addressing modes and completes data transfer which the data width of read and write is different by asymmetric asynchronous FIFO. Furthermore the DMAC adopts dual-clock domain design so as to decrease the power consumption. The DMAC has the function of APB Bridge, and achieves AHB bus and APB bus to run in parallel. And the DMAC could adopt buffer and non-buffer data transfer mode according to the speed of equipments. With 0.18um library technology of SMIC, a working frequency of 408MHZ is achieved. Experimental results show that the DMAC has better performance than traditional DMAC and DMA PL081.*

Index Terms — DMA controller; AMBA bus; SoC; Dual-clock Domain; Asymmetric Asynchronous FIFO.

I. INTRODUCTION

Direct Memory Access controller (DMAC) is an important component of SoC architecture and Direct Memory Access (DMA) is an important technique to increase data transfer rate and MPU (microprocessor unit) efficiency in SoC system.

There are a few of on-chip bus standards, but AMBA Rev 2.0[4] (Advance Microcontroller Bus Architecture) has become popular industry-standard on-chip bus architecture. The design of DMAC is compliance to the AMBA specification for easy integration into SoC.

Many new SoC architectures about DMAC have been proposed in recent years, such as dedicated flyby DMAC blocks, multi-channel transmission capacity to support multi-user and a great variety of physical links, and dedicated channels for equipments which are large data throughout. Flyby DMAC and dedicated channels DMAC adopt non-buffer data transmission mode, and the advantage of this mode is improvement of the efficiency of data transfer, but when rapid data blocks transfer proceed in equipments which are on the same group of bus or data movement in the same port that is common in audio and video codec application, this mode is no longer applicable. Because non-buffer DMAC can not achieve write operation after reading in single-cycle.

In addition, when DMAC transfers data, it is as a master on AHB bus. When realizing data transfer between AHB slave and APB peripheral, DMAC must buffer data and apply to APB Bridge for visiting APB peripheral. The buffer mode leads to transfer rate not high.

We have proposed a new DMAC architecture which lies in between AHB bus and APB bus (Fig.1), with APB Bridge function, that is, DMAC controls directly data, address and control signals on APB bus. So it could achieve AHB operation and APB operation run in parallel. And data transfer mode can be buffer and non-buffer mode according to practical application by setting control register.

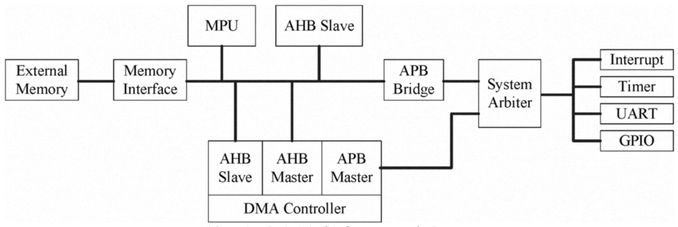

Fig. 1. DMAC Connectivity

II. DMAC ARCHITECTURE

A. Functional Overview

This section provides work process of the DMA controller, Fig. 2 shows the DMAC architecture.

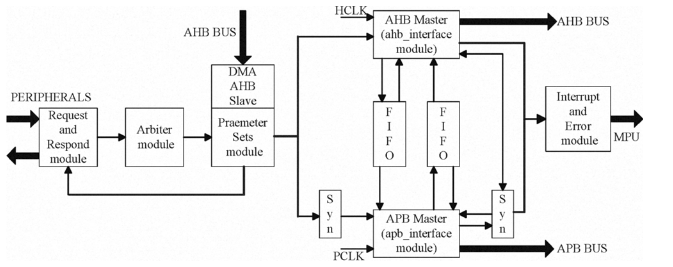
HCLK
AHB BUS
AHB Master (ahb_interface module)
AHB BUS
DMA AHB Slave
PERIPHERALS
Request and Respond module
Arbiter module
Praemeter Sets module
FIFO
FIFO
Interrupt and Error module
MPU
Syn
Syn
APB Master (apb_interface module)
PCLK
APB BUS

Fig. 2. DMAC Architecture

Firstly, MPU programs the parameter set associated with the channel by DMA AHB Slave interface. Request and Respond module accepts the request of data transfer from peripherals, software transfer request, or chaining transfer. Those requests enter arbiter module. The selected request finds its corresponding parameter set in parameter sets module which submits transfer parameters to AHB Master module and APB Master module pasting synchronizer. AHB Master module asserts bus request signal to get access to the AHB according to parameter set, and completes data transfer between AHB and FIFO. APB Master module asserts bus request signal to gain the control of APB after arbitration with

This work was supported by the National Natural Science foundation of China (No.60236020)

Guoliang Ma*, Master Candidate, Institute of Microelectronics of Tsinghua University, Beijing, China. (e-mail: mgl07@mails.tsinghua.edu.cn)

Hu He, Associate Professor, Institute of Microelectronics of Tsinghua University, Beijing, China. (e-mail: hehu@tsinghua.edu.cn)

978-1-4244-3870-9/09/$25.00 ©2009 IEEE

APB Bridge, and completes data transfer between APB and FIFO. AHB operation and APB operation are independent, and DMAC could achieve AHB operation is in parallel with APB operation. When data transfer completes or error occurs in the process of data transfer, interrupt and error module asserts interrupt signal or error signal to the MPU.

B. Dual-Clock Domain Design

The relation of clock frequency between AHB bus and APB bus should be the same or multiple is not be defined clearly in AMBA 2.0 specification, but the multiple clock is technical trend. Its advantages are as follows:

(1). Power consumption and frequency is linear relation, so we can decrease the power consumption of bus and peripheral by reducing the APB clock frequency.

(2). If the clock frequency of APB is lower, the design of peripheral module can be more focused on reducing the area rather than meeting the timing requirements.

But metastability could never been avoided in anytime when transmitting signals between asynchronous clock domains. We must use synchronous circuits to decrease it to an acceptable level. APB Master module uses PCLK frequency which times APB bus transfer and is half of HCLK frequency, and the other modules use HCLK. So all signals input and output APB module must be synchronized (Fig. 2 syn module).The channel parameters from parameter sets module are synchronized to APB Master module by two levels synchronizer. Pulse control signal transmission between AHB Master and APB Master, or between Interrupt and Error module and APB Master module are synchronized by pulse synchronizer (Fig. 3).When data transfer between AHB slave and APB peripheral, we should use asynchronous FIFO as an excellent solution.

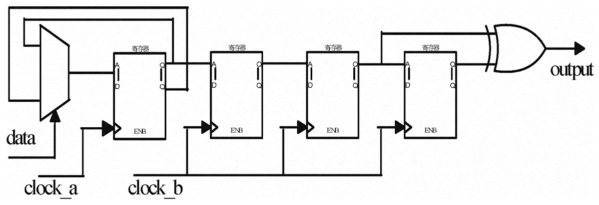

Fig. 3. Pulse Synchronizer

C. Multi-channel Design and Arbitration Mechanism

DMAC has 8 channels, providing 6 peripheral triggering channels and 2 software channels triggered by Parameter Sets module upon writing to a parameter set entry automatically. All channels can also be used for channel chaining transfer triggered automatically by Interrupt and Error module. The channel chaining capability for the DMAC allows the completion of a DMAC channel transfer to trigger another DMAC channel transfer. The purpose is to allow you the ability to chain several transfers through one transfer occurrence.

DMAC combines hardware- and software-programmable arbitration mechanism. In hardware channel priority, the lowest-numbered channel has the higher priority. And software-programmable arbitration mechanism adopts weighted priority rotational algorithm proposed in the paper "Weighted Priority Rotational Algorithm on PCI Bus Arbitration" [3].

User assigns different weights for each channel according to their performance request, and the weight (W_i) is the times of the channel gaining access of the bus in one rotation. The initial time (T_i) of each channel gaining access to bus in one rotation is W_i.

When the DMAC begins to run, all channels get access to bus in terms of hardware priority. If many different channels request to use the bus, the arbitration mechanism ensures the only one channel has access to the bus by observing which channel has the greatest weight. The greatest weighted channel will get access to the bus, and the time of getting access to bus for the channel decreases $T_i = T_i - 1$. When the time T_i of a channel decreases to zero, the channel will lose the right of access to bus until a new rotation begins. Arbiter will start the next rotation in the following circumstances:

(1). The times (T_i) of all channels decrease to zero.

(2). No channel requests to use bus, except the highest weight channel.

(3). The time (T_i) of the lowest weight channel decreases to zero.

The algorithm avoids the disadvantage of dominating bus exclusively in fixed arbitration algorithm and gaining access to bus too averagely in cycle priority arbitration algorithm. By assigning weights for different channels, the probability of each channel visiting to bus meets the actual needs.

D. Parameter Sets

DMAC has 16 parameter sets. Set from 0 to 7 corresponds to channel from 0 to 7 separately. Parameter sets from 8 to 15 are used for linking transfer. Each parameter set is organized into eight words (32bit), and for only contiguous data transfer, you can only configure four words, the remaining words use default values. Each parameter set includes source address, destination address, offset address index (SRCARY, DSTARY, SRCFRM, DSTFRM), data width, burst size(ACNT, BCNT, CCNT)and control information, such as address mode, transfer type, data flow control, link address, chaining transfer control, interrupt and error masking.

DMAC supports linking transfer which is useful for maintaining circular buffer and scatter transfers all with no MPU intervention.

DMAC transfer is defined in terms of three dimensions. This means the source and destination areas are independent index and don't have to occupy contiguous areas of memory. So the DMAC could perform easily data block moving, include bursting and non-bursting transfer, subframe extraction of various data structures and data sorting. The first dimension (Array) consists of ACNT contiguous bytes, the second dimension (Frame) consists of BCNT arrays. Each array transfer is separated from each other by index SRCARY and DSTARY. The third dimension (Block) consists of CCNT frames. Each transfer in the third dimension is separated from the previous by index SRCFRM and DSTFRM. Fig. 4 shows a transfer of 3(CCNT) frames of 4(BCNT) arrays of n (ACNT) bytes. Each array is submitted as one transfer request.

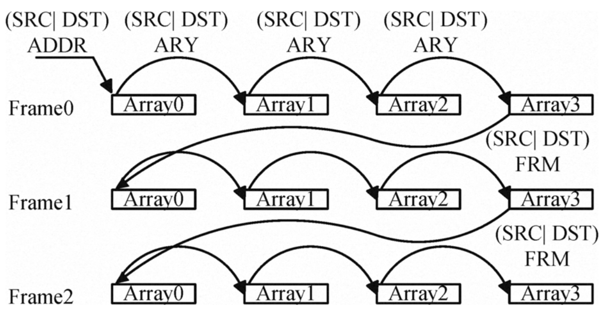

Fig. 4. Three Dimension Transfer

Parameter sets need to be written with transfer context and link parameter set for desired channel by MPU. In this case, DMAC is a slave on AHB bus and has AHB slave interface so that MPU configures the transfer parameter. If MPU programs parameter set 6 or 7, parameter sets module will trigger the channel 6 or 7 transfer request automatically.

Once a trigger event is recognized, the parameter set associated with the channel is read to determine the transfer details, and parameter sets module is responsible for checking the parameter whether is valid and submitting a valid transfer request to AHB Master module and APB Master module to complete transfer task. If the parameter is not valid, the channel request is abandoned.

Each transfer request conveys the transfer information for the first dimension. The first dimension contains 256 AHB bursts at most, and data width is 8, 16 or 32bit. When parameter sets module submits a transfer request, it is responsible for updating the parameter set of the channel, ensuring it can submit the next transfer request as soon as possible when the active transfer is completed. For non-final transfer, this includes address and count updates. When the parameter set is exhausted and needs linking transfer, a link update occurs. The current transfer parameters are reloaded with the parameter set pointed to by the 4-bit link address field.

E. AHB and APB Operation and Parallelism

DMAC as AHB master enables four transfer types: AHB slave-to-AHB slave, AHB slave-to-APB peripheral, APB peripheral-to-AHB salve, and APB peripheral-to-APB peripheral. And DMAC has incrementing and wrapping address modes for source and destination and supports for 8, 16 and 32 bit wide transactions.

When AHB Master module and APB Master module receive transfer request from Parameter Sets module, they can divide into numbers of bursts according the data size in first dimension, and implement appropriate burst transfer. AHB Master module is responsible for transfer data between AHB slave and APB peripheral or between FIFO and AHB slave. It complies with AMBA AHB bus protocol. APB Master module has the function of APB Bridge, complying with AMBA APB bus protocol. And it is responsible for transfer data between AHB slave and APB peripheral or between FIFO and APB peripheral. When DMAC visits APB peripheral, system arbiter will employ an internal algorithm to decide DMAC or APB Bridge will gain access to APB bus. DMAC could achieve AHB Master module and APB Master module get access to AHB bus and APB bus simultaneously so that read and write operation process in parallel.

In AHB slave-to-AHB slave, in order to meet the requirements of real-time, DMAC reads data from AHB slave for one burst, writing into FIFO, and asserts request bus signal for write operation. When one burst data is read completely and DMAC occupies bus again, DMAC writes data to AHB slave. Transfer operations carry out in order, until task terminates.

In AHB slave-to-APB peripheral, if the APB peripheral is slow equipment, user could adopt the way of transfer with FIFO buffer by set transfer mode register in parameter set, and the process is like AHB slave-to-AHB slave. If the frequency of APB peripheral is match to AHB bus frequency, DMAC can implement transfer without FIFO buffer. AHB read operation is in parallel with APB write operation, forming a two-stage pipeline, thus transfer speed is increased greatly.

The case of APB peripheral-to-AHB slave is like AHB slave-to-APB peripheral, and APB peripheral-to-APB peripheral is like AHB slave-to-AHB slave, which needs FIFO buffer.

F. Asymmetric Asynchronous FIFO Design

In order to realize data transfer in cross clock domain, DMAC need build in two FIFO. One is used for AHB slave-to-APB peripheral, the other is used for APB peripheral-to-AHB slave. And the data width of reading and writing may be different, so we should design asymmetric asynchronous FIFO on the basis of general FIFO. In the transfer type of AHB-to-AHB and APB-to-APB, data transfer proceeds in the same clock, so the FIFO is only as a data buffer in one clock domain. That is, in write port, we can not only write data to FIFO, but also read data from FIFO, and in read port, we only read data.

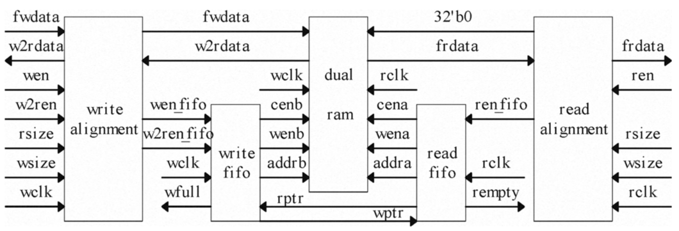

Fig. 5. Asymmetric Asynchronous FIFO

Fig. 5 shows the asymmetric asynchronous FIFO. dualram module is a 128*32bit asynchronous dual-port SRAM.

fifo_align module includes write and read alignment. When writing data width is equal to read data width and less than 32bit, data need duplicate to 32bit so that AHB salve and APB peripheral select data from correct byte lanes which complying with little-endian data bus in AMBA 2.0. For example, write data is 8 bits, the every byte of 32bit data written to FIFO is the 8-bit write data. When write data width is less than read data width, write alignment need save data until up to read data width, and meet the requirement of data bus. For example, the write data width is 8 bit and read data width is 16bit, write alignment module waits one period getting a 16bit data, and duplicates to 32bit data written to dual-ram. When write data width is more than read data width, read alignment need get one data from dual-ram, and output data partly, according to read data width. Then it reads next data from FIFO.

fifo_control module controls reading or writing dual-ram and generates write full signal and read empty signal. And it realizes to read data and write data in one port when AHB slave-to-AHB slave, or APB peripheral-to-APB peripheral data transfer.

G. Interrupt and Error System Design

If a transfer has been completed, indicated by the transfer count reaching 0 if the DMAC is performing flow control, or by the peripheral setting the LREQ (last request) signals if the peripheral is performing flow control.

When data transfer terminates corresponding to one channel, DMAC will generate interrupt signal to MPU and describe which channel request is complete. DMAC begins to finish the next channel data transfer immediately. If the parameter set corresponding to the channel has chaining transfer, the module will trigger another DMAC channel transfer automatically.

If one parameter set corresponding to the selected channel is error or error which is generated by slave to indicate some form of error condition occurs in process of data transfer, the data transfer request is abandoned, DMAC will generate error signal to MPU and describe which channel is error. DMAC begin to finish the next channel data transfer immediately.

If link parameter set is error, data transfer unrelated with link parameter set is complete. Data transfer corresponding to link parameter set is abandoned. DMAC will generate error signal to MPU and describe which link parameter set is error.

And interrupt signal and error signal could be masked by setting control resister in parameter set.

III. PERFORMANCE AND RESULTS

The DMAC was designed in Verilog language and successfully synthesized into the gate-level circuit. With 0.18um library technology of SMIC, we synthesized our design by Design Compiler. The delay of critical path is 2.45ns, that is, the maximum frequency is 408 MHZ.

We use ModelSim to simulate our DMAC, the performance of the DMAC is compared to that of the traditional DMAC (AN2548 designed by ST) which can not use burst mode for transfer data and PrimeCell Single Master DMAC Controller (PL081) which uses burst mode and has link operation, but does not have the function of APB Bridge designed by ARM.

Assume that there are six non-contiguous areas. Each area occupies contiguous address space containing 64 words. The source areas or destination areas is APB peripheral or AHB slave. HCLK and PCLK have the same frequency, and one data transfer on APB bus needs two cycles according to APB protocol. When data transfer between AHB and APB, our DMAC could adopt non-buffer mode and visit AHB bus and APB bus simultaneously. Read and write operate at the same time, thus constituting a two-stage pipeline. According to AN 2548 Application note[2], the DMAC performing a data transfer needs 4 cycles in AHB-to-AHB, 8 cycles between AHB and APB, and 10 cycles in APB-to-APB. So the time that those data transfer need is listed in TABLE I.We can see the performance of our DMAC is better. Each data transfer rate is increased by about 50%. Between AHB slave and APB peripheral, our DMAC adopts non-buffer mode to transfer data, and the rate is increased by 67%.

TABLE I.
Performance Comparison Unit: cycles

	AN2548 DMAC	Our DMA (buffer)	Our DMA (non-buffer)	PL081 DMAC
AHB-to-AHB	1920	989		989
AHB-to-APB	3072	1564	1012	1320
APB-to-AHB	3072	1564	1012	1320
APB-to-APB	3840	1883		1728

From TABLE I, if our DMAC uses buffer mode, the performance of PL081 is better than ours. But the time spent more than PL081 is used in signals synchronization. Our DMAC adopts dual-clock domain design, and PL081 only has one clock HCLK. When our DMAC uses non-buffer, even though signal synchronization will occupy much time, the performance of our DMAC is better than PL081, and the data transfer rate is increased by 23.3%.And our DMAC has more features than PL081.

V. SUMMARY AND CONCLUSIONS

In this paper, we propose a design and implementation of an AMBA based advanced DMAC controller. The DMAC has 8 channels which support hardware and software triggers, linking operation and channel chaining transfer to improve the real-time processing capability and provides three dimensions transmission so as to perform data block moving, data sorting and subframe extraction of various data structures. Channel arbitration mechanism adopts hardware priority combined with weighted priority rotational algorithm, so that meet the different requirements of fairness and the priority in different systems. And the DMAC supports incrementing and wrapping address modes and completes data transfer which the data width of read and write is different by asymmetric asynchronous FIFO. Moreover the DMAC adopts dual-clock domain design so as to decrease the power consumption. Furthermore the DMAC has the function of APB Bridge, and it can control address, data and control signals independently and achieve AHB bus and APB bus to run in parallel. And the DMAC could adopt buffer and non-buffer data transfer mode according to the speed of equipments. Non-buffer mode can enhance the data transfer rate significantly. Experimental results show that the DMAC has the advantage of high speed transfer rate and is much suitable to various application fields, such as multimedia processing.

REFERENCE

[1] PrimeCell Single Master DMA Controller(PL081) Revision:r1p2 Technical Reference Manual. http://www.arm.com

[2] AN2548 Application note. http://www.st.com.

[3] Liu Haihua, Chen Xinhao. “Weighted Priority Rotational Algorithm on PCI Bus Arbitration”. Computer Engineering and Applications 2003. 36

[4] AMBA Specification (Rev 2.0). http://www.arm.com

[5] TMS320DM643x DMP EDMA3 User’s Guide. SPRU987, January 2007.