Design and Implementation of an Advanced Direct Memory Access Controller on Advanced Microprocessor Bus Architecture

Ravitesh Mishra
Assistant Professor, Department of Electronics and Communication, RNTU, Bhopal.

Abstract: - In this paper, we have proposed a design and implementation of an AMBA based advanced DMA controller. The DMAC has 6 channels which support hardware 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 sub-frame extraction of various data structures. Channel arbitration mechanism adopts hardware priority. 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. And the DMAC could adopt buffer and non-buffer data transfer mode according to the speed of equipment. Experimental results show that the DMAC has better performance than traditional DMAC and DMA PL081.

Keywords: DMAC, DMA PL081, AMBA

I. INTRODUCION

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 (Advance Microcontroller Bus Architecture) has become popular industry-standard on-chip bus architecture. Design of DMAC is compliance to AMBA specification for easy integration into SoC [1].

Many new SoC architectures about DMAC have been proposed in recent years, such as dedicated flyby DMAC blocks, multi-channel transmission capacity to support multiuser and a great variety of physical links, and dedicated channels for equipment's 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 equipment's 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same group of bus or data movement in the same port that is common in audio and video codec application, this mode is no longer applicable.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 cannot achieve write operation after reading in single cycle [2].

To overcome data movement restrictions imposed by sharing resources with the core, the DMA system in the family contains its own dedicated internal address and data buses. Internal memory is partitioned so that the Program Control Unit (PCU) and DMA can both perform internal memory accesses in the same core clock cycle, as long they are accessing different memory partitions. Also, if one of these two controllers (PCU or DMA) is accessing internal memory, the other controller can perform an external memory access in the same core clock cycle In addition to data moves between I/O and internal or external memory, the DMA in the can perform memory-to-memory transfers (internal, external, or mixed) [3].

Traditionally, DMA uses the same internal address and data buses as the core. Consequently, when DMA performs one or more word transfers, it can cause the core to temporarily halt activity for one or more cycles while DMA moves the data. With this type of architecture, the core and DMA cannot both perform data moves in the same core clock cycle. The DMA unit contains the necessary counters, offset registers, and pointers to transparently handle one-, two-, and three-dimensional data matrix transfers. These registers can be given values that result in special addressing modes, for example, access to circular buffers and linear buffers with non-unit stride. The data structure dimensionality can be chosen independently for the source access versus the destination access involved in the data move. The DSP56300 contains six DMA channels that share buses and offset registers but are otherwise independent. Each DMA channel can be triggered by interrupt pins, peripheral actions, or other DMA events, and assigned a priority relative to other channels and relative to the core. The DMA unit contains the necessary counters, offset registers, and pointers to transparently handle one, two-, and three-dimensional data matrix transfers. These registers can be given values that result in special addressing modes, for example, access to circular buffers and linear buffers with non-unit stride [4, 5].

II. DIRECT MEMORY ACCESS

Direct memory access (DMA) is a process in which an external device takes over the control of system bus from the CPU. DMA is for high-speed data transfer from/to mass storage peripherals, e.g. hard disk drive, magnetic tape, CD-ROM, and sometimes video controllers. For example, a hard disk may boasts a transfer rate of 5 M bytes per second, i.e. 1 byte transmission every 200 ns, To make such data transfer via the CPU is both undesirable and unnecessary. The basic idea of DMA is to transfer blocks of data directly between memory and peripherals. The data don't go through the microprocessor but the data bus is occupied. "Normal" transfer of one data byte takes up to 29 clock cycles. The DMA transfer enquires only 5 clock cycles. DMA can transfer data as fast as 60 M byte per second. The transfer rate is limited by the speed of memory and peripheral devices [6].

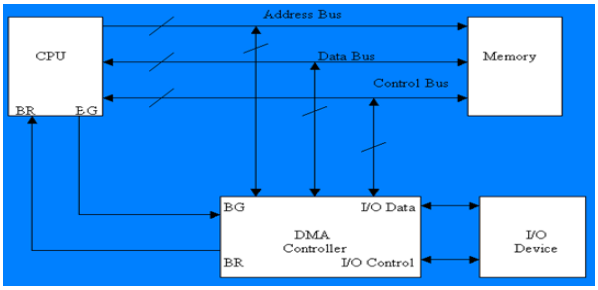

Figure 1: Block diagram of DMA Controller

Direct Memory Access (DMA) is one of several methods for coordinating the timing of data transfers between an input/output (I/O) device and the core processing unit. DMA is

one of the faster types of synchronization mechanisms, generally providing significant improvement over interrupts, in terms of both latency and throughput. An I/O device often operates at a much slower speed than the core. DMA allows the I/O device to access the memory directly, without using the core. DMA can lead to a significant improvement in performance because data movement is one of the most common operations performed in processing applications [7].

III. OPERATION OF DMA

- Each of the six DMA channels contains its own set of four operational registers, all of which are memory-mapped in the internal I/O memory space and all of which are 24-bit registers
- DMA Source Address Register (DSR):A read/write register that contains the source address for the next DMA transfer for its channel. Each DMA channel has one DSR: DSR0, DSR1, DSR2, DSR3, DSR4 and DSR5.
- DMA Destination Address Register (DDR):A read/write register that contains the destination address for the next DMA transfer for its channel. Each DMA channel has one DDR: DDR0, DDR1, DDR2, DDR3, DDR4 and DDR5.
- DMA Counter (DCO):A read/write register that contains the number of DMA data transfers to be performed by its channel. The DCO has five modes of operation determined by the DMA channel Address Generation [8, 9]
- Mode defined in the DMA channel's Control Register. Each DMA channel has one DCO: DCO0, DCO1, DCO2, DCO3, DCO4 and DCO5
- DMA Control Register (DCR): A read/write register that controls the operation of a DMA channel. Each DMA channel has one DCR: DCR0, DCR1, DCR2, DCR3, DCR4 and DCR5.

IV. PROPOSED METHODOLOGY

In addition, when DMAC transfers data, it is as a master on ARB bus. When realizing data transfer between ARB 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 ARB bus and APB bus (Fig. 2), with APB Bridge function, that is, DMAC controls directly data, address and control signals on APB bus. So it could achieve ARB 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.

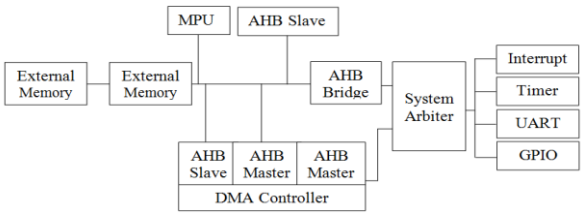

Figure 2: Proposed DMA

System On-chip:- A system on a chip or system on chip (SoC or SOC) is an integrated circuit (IC) that integrates all components of a computer or other electronic system into a single chip. It may contain digital, analog, mixed signal, and often radio frequency functions all n a single chip substrate. A typical application is in the area of embedded systems.

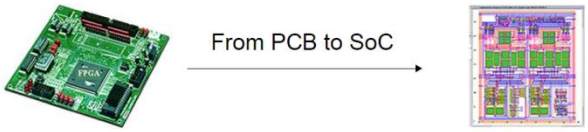

Figure 3: System on Chip

The contrast with a microcontroller is one of degree. Microcontrollers typically have under 100 kB of RAM (often just a few kilobytes) and often really are single-chip-systems, whereas the term SoC is typically used with more powerful whereas the term SoC is typically used with the more powerful processors, capable of running software such as the desktop versions Windows and Linux, which need external memory chips (flash, RAM) to be useful, and which are used with various external peripherals. In short, for larger systems system on a chip is hyperbole, indicating technical direction more than reality: increasing chip integration to reduce manufacturing costs and to enable smaller systems. Many interesting systems are too complex to fit on just one chip built with a process optimized for just one of the system's tasks. When it is not feasible to construct an SoC for a particular application, an alternative is a system in package (SiP) comprising a number of chips in a single package. In large volumes, SoC is believed to be more cost-effective than SiP since it increases the yield of the fabrication and because its packaging is simpler. Another option, as seen for example in higher end cell phones and on the Beagle Board, is package on package stacking during board assembly. The SoC chip includes processors and numerous digital peripherals, and comes in a ball grid package with lower and upper connections. The lower balls connect to the board and various peripherals, with the upper balls in a ring holding the memory buses used to access NAND flash and DDR2 RAM. Memory packages could come from multiple vendors.

AMBA Micro-controller: - An AMBA-based microcontroller typically consists of a high-performance system backbone bus (AMBA AHB or AMBA ASB), able to sustain the external memory band width, on which the CPU, on-chip memory and other Direct Memory Access (DMA) devices reside. This bus provides a high-bandwidth interface between the elements that are involved in the majority of transfers. Also located on the high performance bus is a bridge to the lower bandwidth APB, where most of the peripheral devices in the system are located.

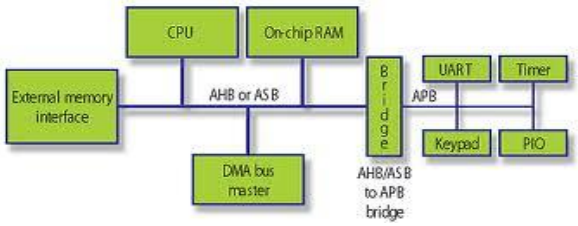

Figure 4: AMBA based microcontroller

AHB is a new generation of AMBA bus which is intended to address the requirements of high-performance synthesizable designs. It is a high-performance system bus that supports multiple bus masters and provides high-bandwidth operation.
AMBA AHB implements the features required for high-performance, high clock frequency systems including:

- Burst transfers
- Split transactions
- Single-cycle bus master handover
- Single-clock edge operation
- Non-tristate implementation
- Wider data bus configurations (64/128 bits)

Bridging between this higher level of bus and the current ASB/APB can be done efficiently to ensure that any existing designs can be easily integrated. An AMBA AHB design may

contain one or more bus masters, typically a system would contain at least the processor and test interface. However, it would also be common for a Direct Memory Access (DMA) or Digital Signal Processor (DSP) to be included as bus masters. The external memory interface, APB bridge and any internal memory are the most common AHB slaves. Any other peripheral in the system could also be included as an AHB slave. However, low-bandwidth peripherals typically reside on APB.

A typical AMBA AHB system design contains the following components: AHB master- A bus master is able to initiate read and write operations by providing an address and control information. Only one bus master is allowed to actively use the bus at any one time.

AHB slave: - Bus slaves respond to a read or write operation within a given address- space range. The bus slave signals back to the active master the success, failure or waiting of the data transfer.

AHB arbiter: - The bus arbiter ensures that only one bus master at a time is allowed to initiate data transfers. Even though the arbitration protocol is fixed, any arbitration algorithm, such as highest priority or fair access can be implemented depending on the application requirements. An AHB would include only one arbiter, although this would be trivial in single bus master systems.

AHB decoder: - The AHB decoder is used to decode the address of each transfer and provide a select signal for the slave that is involved in the transfer. A single centralized decoder is required in all AHB implementations.

V. SIMULATION RESULT

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 MRZ.

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 cannot 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.

Table 1: Performance table (Unit: - Cycles)

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

Read and write operate at the same time, thus constituting a two-stage pipeline. According to AN 2548 Application note!", 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 1.

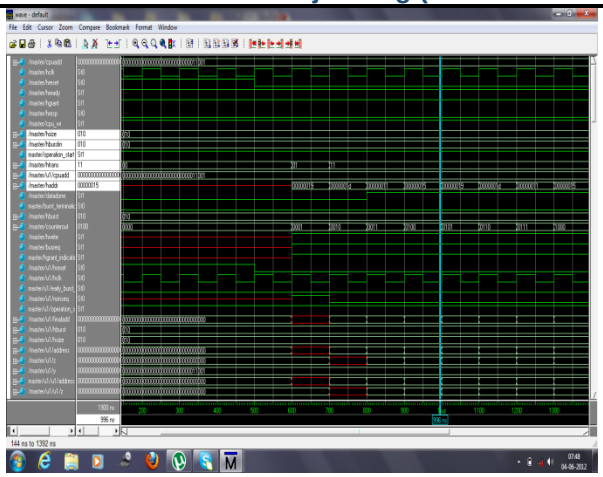

Figure 4: AHB Slave –to-AHB Slave

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%.

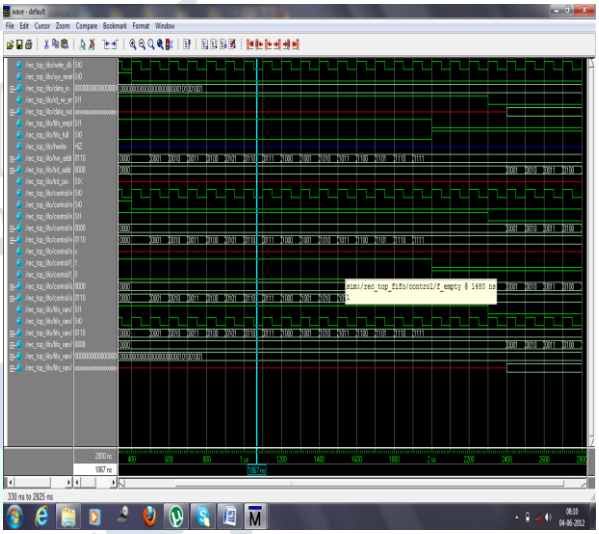

Figure 5: AHB Slave –to-APB Peripheral

VI. CONCLUSION

A design and implementation of an AMBA based advanced DMAC controller is proposed. 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 sub frame 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 ARB 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 equipment's. Non-buffer mode can enhance the data transfer rate significantly.

REFERENCE

[1] Guoliang Ma and Hu He, "Design and Implementation of an Advanced DMA controller on AMBA based SoC", IEEE 8th International Conference on Digital Object, pp.

419 – 422, IEEE 2009.

[2] Hessel, S.; Szczesny, D., Bruns, F., Bilgic, A.; Hausner and J.Vehicular, "Architectural analysis of a Smart DMA Controller for Protocol Stack Acceleration in LTE terminals, Technology", 72nd Digital Object, pp. 1-5, IEEE 2010.

[3] Jaehoon Song; Piljae Min; Hyunbean Yi; Design of Test Access Mechanism for AMBA-Based System-on-a-Chip,Sungju Park VLSI Test Symposium, 2007. 25th IEEE, pp. 375 – 380.

[4] Hang Yuan; Hongyi Chen; An improved DMA controller for high speed data transfer in MPU based SOC, Guoqiang Bai Solid-State and Integrated Circuits Technology, 2004. Proceedings. 7th International Conference on Vol. 2 Digital Object, pp. 1372 - 1375.

[5] Szczesny, D.; Hessel, S.; Traboulsi, S.; Optimizing the Processing Performance of a Smart DMA Controller for LTE Terminals, Bilgic, A. Embedded and Real-Time Computing Systems and Applications (RTCSA), 2010 IEEE 16th International Conference on Digital Object , Page(s): 309 – 315

[6] Chia-Hao Yu; Chung-Kai Liu; Chih-Heng Kang; Tsun-Hsien Wang; Chih-Chien Shen; An Efficient DMA Controller for Multimedia Application in MPU Based SOC, Shau-Yin Tseng Multimedia and Expo, 2007 IEEE International Conference on, Page(s): 80 – 83

[7] Prokin; DMA transfer method for wide-range speed and frequency measurement, M. Instrumentation and Measurement, IEEE Transactions on Volume: 42 , Issue: 4 , Page(s): 842 - 846

[8] Tai-Yi Huang; Liu, J.W.-S.; A method for bounding the effect of DMA I/O interference on program execution time, Hull, D.Real-Time Systems Symposium, 1996., 17th IEEE ,

[9] Osborne, S.; Erdogan, A.T.; Arslan, T.; Bus encoding architecture for low-power implementation of an AMBA-based SoC platform, Robinson, Volume: 149 , Issue: 4 , Page(s): 152 – 156

[10] Pockrandt, M.; Herber, P.; Model checking a System C/TLM design of the AMBAAHB protocol, Page(s): 66 – 75, 2011.