International Journal of Computer Applications Technology and Research
Volume 12–Issue 04, 08-11, 2023, ISSN:-2319–8656
DOI:10.7753/IJCATR1204.1003

Design and Implementation of an Accelerated DMA Controller based on AXI Bus

Deqing Cai
School of Communication Engineering
Chengdu University of Information Technology
Chengdu, China

Muyao Ge
School of Communication Engineering
Chengdu University of Information Technology
Chengdu, China

Hashi Wang
School of Communication Engineering
Chengdu University of Information Technology
Chengdu, China

Abstract: Because Direct Memory Access (DMA) hardly consumes processor resources when carrying high-speed data, an accelerated DMA controller based on Advanced eXtensible Interface（AXI）bus protocol is proposed in this paper. The accelerable part of the controller is to replace the CPU for descriptor splitting processing by hardware, which greatly improves the CPU computing power. At the same time, the controller is equipped with eight deeply configured channels, which are used to process different types of tasks. The design supports single transmission of data and linked list transmission type, which can verify the transmitted data and ensure the security of data transmission. In this design, the working frequency of the controller can reach 500M, the power consumption is 1.3mw, and the data throughput rate can reach 40Gbps, which greatly improves the data moving efficiency of the system.

Keywords: AXI bus; High bandwidth; Accelerable; DMA controller; Data verification; Low power consumption;

1. INTRODUCTION

Direct Memory Access is a high-speed data transmission technology that hardly consumes processor resources. In this technology, data is directly read and written without additional intervention of the Central Processing Unit (CPU). The traditional microprocessor data transfer is usually controlled by the CPU [1-3]. The CPU first copies the data of the source address to the internal register, and then writes the data of the register to the destination register, and the amount of data copied has an obvious length limit [4]. With the rapid development of large-scale integrated circuits and the rapid increase of market demand, the above methods severely limit the performance of CPU. In the existing SoC products, the performance of CPU has been greatly improved, which is no longer the key factor for the low data exchange rate in SoC design [5-6]. Therefore, how to improve the data transmission efficiency of the bus and the working efficiency of each slave device has become another key factor affecting the design of a SoC [7].

DMA technology is more and more widely used in the existing SoC design. This technology is to carry out data transfer through hardware, and does not require CPU to participate all the time. During the data transfer of hardware, CPU can handle other tasks [8]. In this way, the data migration mode can be flexibly adjusted and the CPU performance can be greatly exercised. After the CPU transfers the bus right to DMA controller, DMA controller can independently complete the data transfer and processing, and the bus right can be returned to the CPU after the transfer is completed. In SoC chip design, IP core reuse technology has been widely used in order to reduce design cost and design cycle. In order to better use the IP core reuse technology to achieve the enhancement and expansion of the chip, the way of using the on-chip bus in the design process is gradually recognized by the designer [9]. AMBA bus is performance microcontroller communication standard proposed by ARM company. Due to its characteristics of strong universality and high performance and low power consumption, AMBA bus has become a widely used data transmission bus protocol in SOC design.

In this paper, the design of an accelerated DMA controller based on AXI bus adopts high performance, low power consumption, low delay, high bandwidth in the data transmission [10]. The design realizes single transmission and linked list transmission data types, and the data can be checked in the process of data transmission. Multiple channels are designed inside the controller to store different commands sent by the CPU. Hardware is used instead of CPU to accelerate the analysis of DMA descriptors in the channel. The data transmission granularity can be configured in the DMA descriptor, which well meets the requirements of versatility and flexibility of DMA controller.

2. OVERALL STRUCTURE DESIGN OF ACCELERATED DMA CONTROLLER

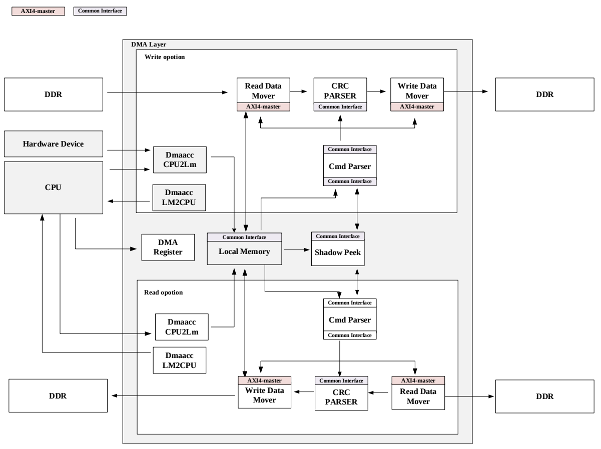

Figure 1. Overall structure of DMA controller

The DMA controller in this paper is an accelerated circuit based on AXI bus, which mainly implements hardware acceleration processing on the process of CPU parsing descriptor, which replaces the traditional idea of using software to process descriptor and further improves the efficiency of data moving. As shown in the figure 1, the controller can be divided into write operation and read operation according to command types. It can be divided into external interface circuit and internal logic circuit of controller by function. Therefore, the accelerated DMA controller peripheral interface circuit designed in this paper includes master interface module based on AXI bus, which is used for request and transmission in the process of data moving. The internal circuit of the controller mainly includes the hardware module which can accelerate the parsing descriptor, the multi-channel monitoring and arbitration module, the DMA descriptor processing module, the data request and processing module, the data verification module, the data move status information module and the register module.

3. CIRCUIT STRUCTURE DESIGN

3.1 THE CIRCUIT OF ACCELERATED PARSER DESCRIPTOR

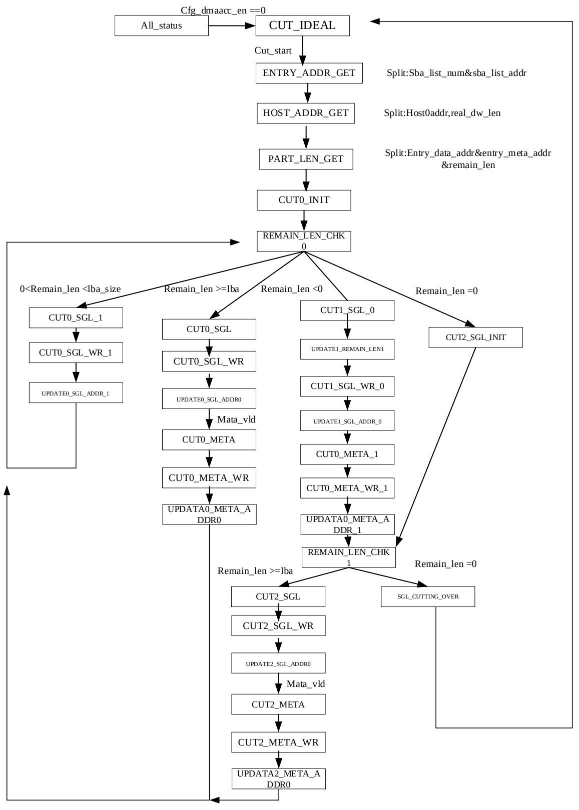

Figure 2. Split the DMA command state transition graph

As shown in Figure 2, this module is the core circuit for accelerating analytic descriptors in the accelerable controller, and the logic for accelerating processing descriptors is implemented by the state machine logic in the figure above, and corresponding command processing is carried out according to different command types. We can find in Figure 2, the core idea of this module is to split and reassemble DMA descriptors by hardware on behalf of the CPU, allowing the CPU to handle other tasks in the meantime. The module has a FIFO data cache module, which is used to synchronize DMA requests from software or other hardware modules. Then, the internal state machine is triggered to read, parse, split and store the descriptors inside the FIFO. There are 8 independent channels in the module, which can be deeply configurable, used to store the DMA descriptors and Entry information after splitting, in order to meet the needs of processing different types of DMA commands. The number of DMA requests received by each channel and the response after processing are maintained by the pointer inside the channel.

3.2 MULTI-CHANNEL MONITORING CIRCUIT

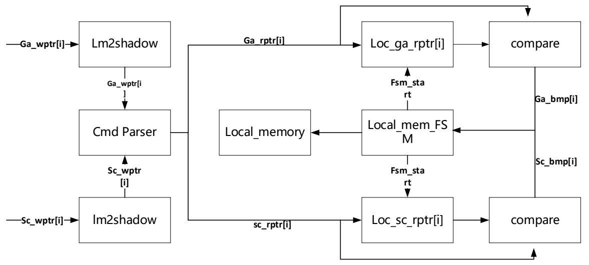

Figure 3. Channel monitoring module design diagram

As shown in Figure 3, the core of this module is to design multiple monitoring circuits to work in parallel, which is used to monitor the change of all channel read and write Pointers in DMA controller. If the pointer changes, the state machine will be triggered to acquire tasks in the corresponding channel, and then the DMA controller will be further triggered to work, and the acquired commands and channel pointers will be analyzed and processed. We can find in Figure 3, at the same time, the monitoring circuit further calculates the remaining DMA commands in the channel according to the changes of the pointer, compares the pointer data maintained locally with that in the channel, and determines the state of all channels. If the Pointers maintained by both are consistent, the channel is in idle state; on the contrary, the corresponding channel is undergoing data processing. Finally, the read pointer of the channel is updated to Local Memory module.

3.3 READ DATA REQUEST CIRCUIT

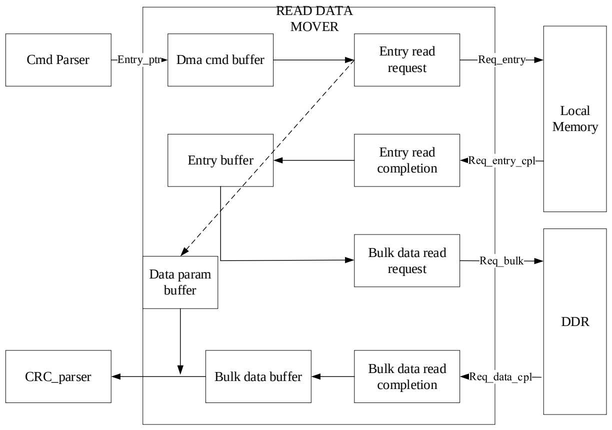

Figure 4. Read data request state transition diagram

As shown in Figure 4, this module mainly caches the command information from other module into the internal cache, then parses the command information in the cache area, and initiates an Entry information request to Local Memory according to the storage address of the Entry information contained in the parsed information. Store the requested Entry information in the local Entry Buffer. When the internal state machine detects new data in the Entry Buffer, we can find in Figure 4, it will parse and process the entry in the local buffer. The internal logic of the module will obtain the actual address and data length of the data, and then trigger the control circuit of the state machine to send a request for origin data to the DDR. When the data request is complete, the data and control information are synchronized, and then transmitted to the others module for processing.

3.4 DATA VERIFICATION CIRCUIT

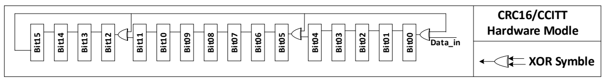

Figure 5. CRC16-CCITT check hardware model

As shown in Figure 5, the data verification method used in this design is Cycle redundancy Check (CRC) algorithm. The module through the hardware way to describe the circuit. The basic idea of CRC check is to use linear coding theory, that is according to the K-bit binary sequence transmitted at the sending end of data, to generate a set of check codes for check according to certain check rules, whose bit width is r, that is CRC codes. In the process of data transmission, CRC codes are added to the tail of user data to form a new set of binary sequences. At the receiving end, according to the data information and CRC code verification rules, to determine whether the data transmission in the process of error. If there is no remainder in the result of data verification at the receiving side, it indicates that the data transmission is correct. Otherwise, the error location of the transmitted data can be inferred according to the remainder. The figure above shows the hardware circuit model under the CRC16-CCITT protocol.

3.5 WRITE DATA TRANSMISSION CIRCUIT

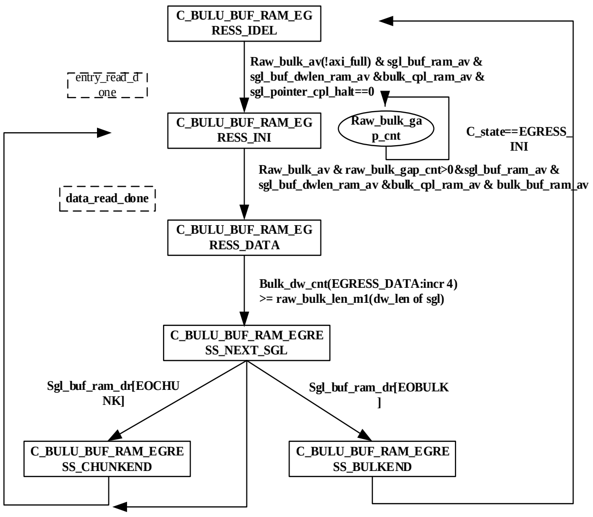
C_BULU_BUF_RAM_EGRESS_IDEL
Raw_bulk_av(!axi_full) & sgl_buf_ram_av & sgl_buf_dwlen_ram_av &bulk_cpl_ram_av & sgl_pointer_cpl_halt==0
entry_read_done
C_BULU_BUF_RAM_EGRESS_INI
Raw_bulk_gap_cnt
C_state==EGRESS_INI
Raw_bulk_av & raw_bulk_gap_cnt>0&sgl_buf_ram_av & sgl_buf_dwlen_ram_av &bulk_cpl_ram_av & bulk_buf_ram_av
data_read_done
C_BULU_BUF_RAM_EGRESS_DATA
Bulk_dw_cnt(EGRESS_DATA:incr 4) >= raw_bulk_len_m1(dw_len of sgl)
C_BULU_BUF_RAM_EGRESS_NEXT_SGL
Sgl_buf_ram_dr[EOCHUNK]
Sgl_buf_ram_dr[EOBULK]
C_BULU_BUF_RAM_EGRESS_CHUNKEND
C_BULU_BUF_RAM_EGRESS_BULKEND

Figure 6. Write data transfer state transition diagram

The working principle of this module is the same as that of the data reading request module. we can find in Figure 6, The core logic of this circuit is mainly to cache the command information from the command parser module into the internal cache area of this module, and then parse the command in the cache area. Then a request for Entry information is sent to the Local Memory according to the destination address information of entry stored in the DMA command. The verified data and the entry information returned by the request are synchronized and matched through the state machine inside the module, and the information is written to the destination through the AXI bus. The above state machine used for data transmission is shown in Figure 6. The data transmission process of the state machine can be roughly divided into two parts. First, according to the control information in the entry, the data within each entry is transmitted; then, according to the effective identifier of the control information, the data transmission of the entire command is completed by the state machine.

3.6 FIXED PRIORITY ARBITRATED CIRCUIT

The monitoring circuit of this module will monitor the update of internal commands of all channels at the same time. If multiple channels receive new data moving commands at the same time, and the DMA controller only processes commands of a specific channel each time, the preemption of hardware resources by multiple channels will be involved. In this case, the quorum module is required to determine which channel uses the hardware resources. Because in this scenario, different channels correspond to different hardware devices, the arbitration module in this design adopts the arbitrator with fixed priority, that is, the priority of each channel is fixed. If multiple channels simultaneously initiate resource occupation requests, the channel with higher priority will be granted the channel use right.

4. ANALYSIS OF SIMULATION RESULTS

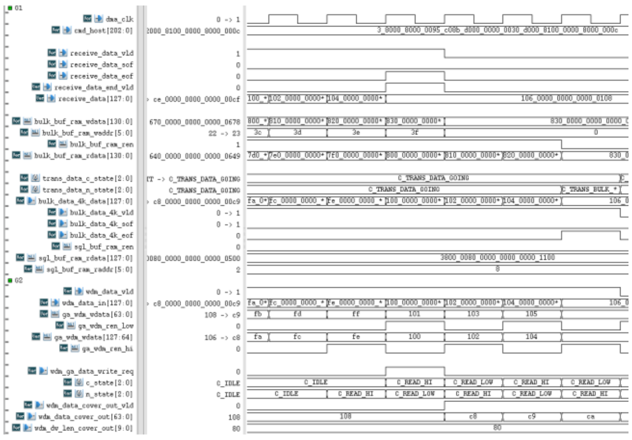

Figure 7. Data simulation results

The design follows the collaborative verification method of hardware and software, and uses the VCS simulation software to build the verification environment. It mainly verifies the single data transmission mode, linked list data transmission mode, accelerated analytic descriptor function, multi-channel monitoring function, arbitration function and interface transmission function of DMA controller. Through the report, we can find that the above functions are consistent with the expected hardware behavior. The data can be moved normally. This Design is based on SMIC's 0.18um process library and synthesizes the controller with Design compiler software. The working frequency of the controller can reach 500M, the power consumption is 1.3mw, and the data throughput rate can reach 40Gbps.

5. CONCLUSIONS

In this paper, an accelerated DMA controller interface circuit based on AXI bus is designed, which has the characteristics of high bandwidth and low power consumption. In the read and write transmission circuit, there are 8 independent channels, and the channel depth can be flexibly configured. In order to meet different command requirements, and based on the SMIC 180nm process comprehensive implementation. The performance of design satisfies the expected index.

6. REFERENCES

[1] Ahmed M A,Aljumah A,Ahmad M G．Design and Implementation of a Direct Memory Access Controller for Embedded Applications．International Journal of Technology,2019,10(2)：309-319 .

[2] Qiao Lufeng,Wang Zhigong. Design of DMA controller for multichannel PCI bus frame engine and data link manager [C].New York,USA.2002.P1481-1485.

[3] Zou Y Y,Chen M,Wei K L．Design of Custom AXI4 IP Based on AXI4 Proto-col.Applied Mechanics and Materials,2014, (3634):2326-2330.

[4] Markatos, E.P. and Katevenis,M.G.H. “User-level DMA without operating system kernel modification” High-Performance Computer Architecture, 1997. Third International Symposium on 1-5 Feb. 1997 P322-331.

[5] Kim, D.and Managuli, R.and Kim, Y.“Data cache and direct memory access in programming mediaprocessors”, Micro,IEEE,Volume:21 Academic Publishers, 1998.

[6] Kidav J U,Sivamangai N M,Pillai MP,et al．FPGA and prototype of cycle stealing DMA array signal processor for ultrasound sector imaging sys-tems Microprocessors and Microsystems,2019,64(64)：53-72.

[7] Saluja H,Grover N.Memory Controller and Its Interface using AMBA 2.0．IJEM-International Journal of Engineering and Manufacturing ,2019,9(4)：33-44.

[8] A. F. Harvey, *National Instruments. DMA Fundamentals on Various PC Platforms Application Note 011*, 1991.

[9] J. Corben, A. Rubini and G. Kroah-Hartman, "Linux device drivers", *O’Reilly Media*, 2005.

[10] C. Bohm, H. Kavianipour, D. Nygren, C. Robson, C. Wernhoff and G. Wikstrom, "A low energy muon trigger for icecube", *Proc. Conf. Rec. 2008 IEEE Nucl. Sci. Symp.*, pp. 2784-2787.