Vol 07, Article 06678; June 2016
http://ijves.com

International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

Design and Analysis of DMA Controller for System on Chip based Applications

A MURALI[1], BONU RAMA KRISHNA[2], KUNA DURGA PRASAD[3]
[1]Research Scholar, Department of ECE, KLEF, K.L University, Green fields, Guntur, A.P, INDIA
[1]Associate Professor, Department of ECE, RAGHU Engineering College, Visakhapatnam, INDIA
[2]B.Tech Final year, [3]Assistant Professor, Department of ECE, PYDAH College of Engineering & Technology, Visakhapatnam, INDIA
[1]amurali3@gmail.com, [2]pavanram1505@gmail.com, [3]durgaprasad.kuna@gmail.com

ABSTRACT

In this paper, the design of Direct Memory Access (DMA) Controller is proposed using Verilog. Direct memory access (DMA) is a feature of modern computers that allows certain hardware subsystems within the computer to access system memory for reading and/or writing independently of the central processing unit. Computers that have DMA channels can transfer data to and from devices much smaller CPU load than computers without a DMA channel. Memory-to memory transfer capability is also provided. Intel 8237 DMA ip core design which is using a very different design technique. So by this project we will trying and use various modelling techniques for designing ip cores than it may be used in various power level circuits and processors. So these IP cores are used in ASICs to control its power, speed and size. So an IP core design is a part of a main embedded circuits and control the working of the circuit and process of high speed data transfer rate and much suitable in SOC products. Its achieved the maximum frequency of 306.24MHz and minimum time period of 3.265nsec Simulation is done on Mentor graphics tool modelsim simulator and synthesis is done on Xilinx tool Xilinx 13.2 ISE synthesizer.

Keywords: *DMA Controller, IP Core, SOC,*

1. INTRODUCTION

Many system-on-chip integrated circuits contain embedded cores with different scan frequencies. Many IP core are design software like Xilinx, Leonardo Spectrum, and Modelsim etc. which can used to design IP core like DMA, Interrupt Controller etc. These IP core can be Power aware an Implement on SoC by choosing different design technique and various modelling techniques .These all modelling technique and tolls like Xilinx ISE also provide RTL view which will help to make IP cores to use in any Processor design. Today's SoCs are composed of a wide variety of modules, such as microprocessor cores, memories, peripherals, and customized blocks directly related to the targeted application. To effectively perform simulation-based design verification of peripheral cores, it is necessary to stimulate the description in a broad range of behaviour possibilities, checking the produced results. Different strategies for generating suitable stimuli have been proposed by the research community to functionally verify these modules and their interconnection when embedded in a SoCs. Direct Memory Access allows devices to transfer data without subjecting the processor a heavy overhead. Otherwise, the processor would have to copy each piece of data from the source to the destination. This is typically slower than copying normal blocks of memory since access to I/O devices over a peripheral bus is generally slower than normal system RAM. During this time the processor would be unavailable for any other tasks involving processor bus access. But it can continue to work on any work which does not require bus access. DMA transfers are essential for high performance embedded systems where large chunks of data need to be transferred from the input /output devices to or from the primary memory.

2. IP CORE (Intellectual Property)

An IP (intellectual property) core is a block of logic or data that is used in making a field programmable gate array (FPGA) or application-specific integrated circuit (ASIC) for a product. As essential elements of design reuse, IP cores are part of the growing electronic design automation (EDA) industry trend towards repeated use of previously designed components. Ideally, an IP core should be entirely portable - that is, able to easily be inserted into any vendor technology or design methodology. Universal Asynchronous Receiver/ Transmitter (UART's), central processing units (CPU's), Ethernet controllers, and PCI interfaces are all examples of IP cores. IP cores fall into one of three categories: *hard cores*, *firm cores*, or *soft cores*. Hard cores are physical manifestations of the IP design. These are best for plug-and-play applications, and are less portable and flexible than the other two types of cores. Like the hard cores, firm (sometimes called *semi-hard*) cores also carry placement data but are configurable to various applications. The most flexible of the three, soft cores exist either as a net-list (a list of the logic gate s and associated interconnections making up an integrated circuit) or hardware description language (HDL) code. In electronic design a semiconductor intellectual property core, IP block, IP core or logic core is a reusable unit of logic, cell, or chip layout design that is the intellectual property of one party. IP cores may be licensed to another party or can be owned and used by a single party alone. The

Vol 07, Article 06678; June 2016
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

term is derived from the licensing of the patent and source code copyright intellectual property rights that subsist in the design. IP cores can be used as building blocks within ASIC chip designs or FPGA logic designs.

3. DMA CONTROLLER

DMA controller is a device, usually peripheral to a CPU that is programmed to perform a sequence of data transfers on behalf of the CPU. A DMA controller can directly access memory and is used to transfer data from one memory location to another, or from an I/O device to memory and vice versa. A DMA controller manages several DMA channels, each of which can be programmed to perform a sequence of these DMA transfers. Devices, usually I/O peripherals, that acquire data that must be read (or devices that must output data and be written to) signal the DMA controller to perform a DMA transfer by asserting a hardware DMA request (DRQ) signal. A DMA request signal for each channel is routed to the DMA controller. This signal is monitored and responded to in much the same way that a processor handles interrupts. When the DMA controller sees a DMA request, it responds by performing one or many data transfers from that I/O device in to system memory or vice versa. Channels must be enabled by the processor for the DMA controller to respond to DMA requests. The number of transfers performed, transfer modes used, and memory locations accessed depends on how the DMA channel is programmed. A DMA controller typically shares the system memory and I/O bus with the CPU and has both bus master and slave capability. Fig 3.1.

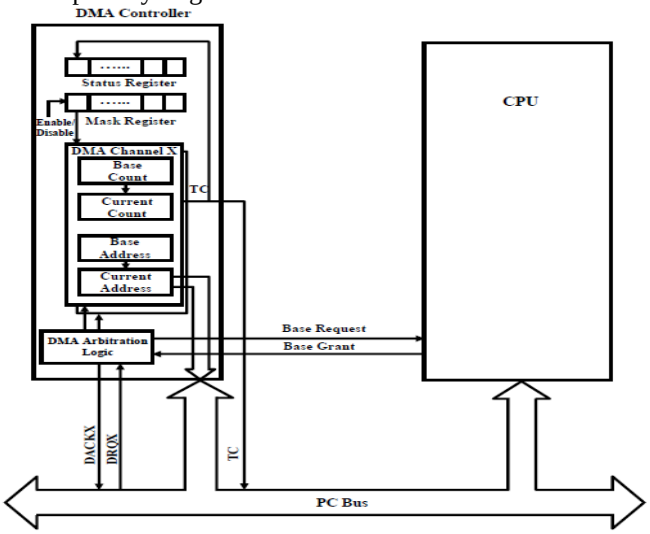


Figure 3.1 shows the DMA controller architecture and how the DMA controller interacts with the CPU.

i) In bus master mode, the DMA controller acquires the system bus (address, data, and control lines) from the CPU to perform the DMA transfers. Because the CPU releases the system bus for the duration of the transfer, the process is sometimes referred to as cycle stealing.

ii) In bus slave mode, the DMA controller is accessed by the CPU, which programs the DMA controller's internal registers to setup DMA transfers. The internal registers consist of source and destination address registers and transfer count registers for each DMA channel, as well as control and status registers for initiating, monitoring, and sustaining the operation of the DMA controller.

3.1 The Principle of the DMA

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 requires only 5 clock cycles. Nowadays, 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. A DMA controller interfaces with several peripherals that may request DMA. DMA is an essential feature of all modern computers, as it allows devices to communicate without subjecting the CPU with a heavy head. Otherwise, the CPU would have to copy each piece of data from source to destination. DMA, usually used with the ISA bus transfer is done using the DMA controller, which is typically part of the motherboard chipset. More advanced designs, such as the PCI bus, as a rule, use bus mastering DMA, where it takes the control of the bus and performs the translation. A typical use of the DMA copies a block of memory from RAM or from a buffer to the device. Such an operation does not stall the processor, which as a result can be scheduled to perform other tasks. DMA is essential to high performance embedded systems.

DMA Controller Operation Steps in a Typical DMA cycle

Device wishing to perform DMA asserts the processors bus request signal.

i. Processor completes the current bus cycle and then asserts the bus grant signal to the device.

Vol 07, Article 06678; June 2016
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

ii. The device then asserts the bus grant ack signal.
iii. The processor senses in the change in the state of bus grant ack signal and starts listening to the data and address bus for DMA activity.
iv. The DMA device performs the transfer from the source to destination address.
v. During these transfers, the processor monitors the addresses on the bus and checks if any location modified during DMA operations is cached in the processor. If the processor detects a cached address on the bus, it can take one of the two actions:
   - Processor invalidates the internal cache entry for the address involved in DMA write operation
   - Processor updates the internal cache when a DMA write is detected
vi. Once the DMA operations have been completed, the device releases the bus by asserting the bus release signal.
vii. Processor acknowledges the bus release and resumes its bus cycles from the point it left off.

3.2 DMA 8237 IP Core:

The 8237 programmable DMA controller core is a peripheral interface circuit for microprocessor systems. The core is designed to be used in conjunction with an external 8-bit address latch. It contains four independent channels and may be expanded to any number of channels by cascading additional controller chips. Each channel has a full 64-K address and word count capability.

3.2.1 Some Important Signal Pins

DREQ (DMA request): Used to request a DMA transfer for a particular DMA channel.
DACK (DMA channel acknowledge): Acknowledges a channel DMA request from a device.
HRQ (Hold request): Requests a DMA transfer. HLDA (Hold acknowledge): signals the 8237 that the Micro-processor has relinquished control of the address, data and control buses.
ADSTB (Address strobe): Functions as ALE to latch address during the DMA transfer.
EOP (End of process): Signals the end of the DMA process.
IOR (I/O read): Used as an input strobe to read data from the 8237 during programming and used as an output strobe to read data from the port during a DMA write cycle.
IOW (I/O write): Used as an input strobe to write data to the 8237 during programming and used as an output strobe to write data to the port during a DMA read cycle.
MEMW (Memory write): Used as an output to cause memory to write data during a DMA write cycle.
MEMR (Memory read): Used as an output to cause memory to read data during a DMA read cycle.

3.2.2 Internal Register

The current address register (CAR) is used to hold the 16-bit memory address used for the DMA transfer. The current word count register (CWCR) programs a channel for the number of bytes (up to 64K) transferred during DMA action. The base address (BA) and base word count (BWC) registers are used when auto-initialization is selected for a channel. In this mode, their contents will be reloaded to the CAR and CWCR after the DMA action is completed. Each channel has its own CAR, CWCR, BA and BWC. The command register (CR) programs the operation.

3.3 Basic DMA Operation

The DMA I/O technique provides direct access to the memory while the microprocessor is temporarily disabled. A DMA controller temporarily borrows the address bus, data bus, and control bus from the microprocessor and transfers the data bytes directly between an I/O port and a series of memory locations. The DMA transfer is also used to do high-speed memory-to memory transfers. Two control signals are used to request and acknowledge a DMA transfer in the microprocessor-based system. The HOLD signal is a bus request signal which asks the microprocessor to release control of the buses after the current bus cycle. The HLDA signal is a bus grant signal which indicates that the microprocessor has indeed released control of its buses by placing the buses at their high-impedance states. The HOLD input has a higher priority than the INTR or NMI Interrupt input.

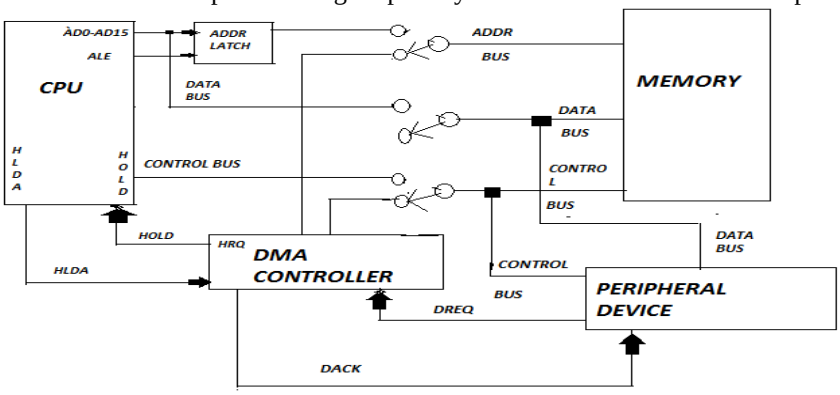


Figure 3.2: When DMA Operates

Vol 07, Article 06678; June 2016
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

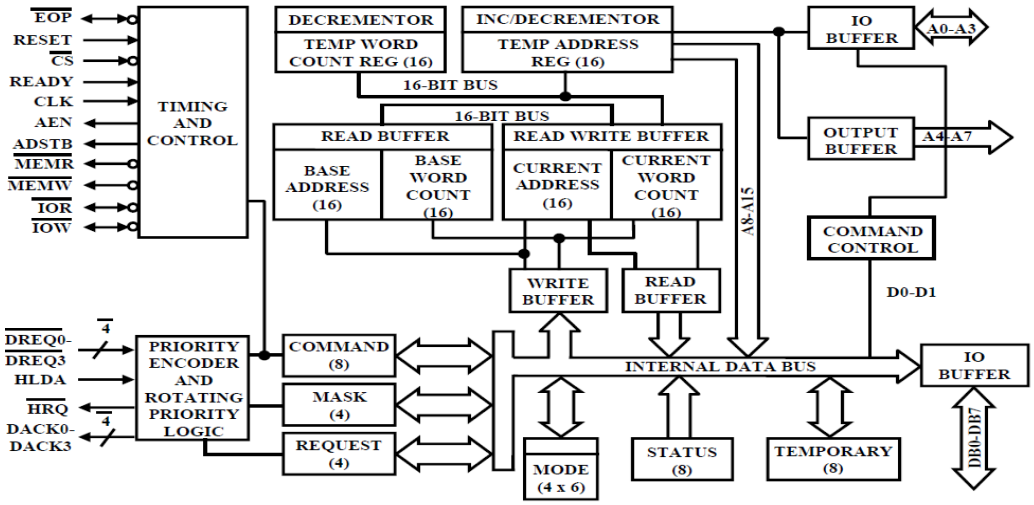


Figure 3.3 8237 Internal Architecture

4. DESIGN AND IMPLEMENTATION

The Method of high speed data transfer by Advanced Microprocessor Bus Architecture (AMBA) based Direct Memory Access (DMA) controller using asynchronous first in first out (FIFO). Direct memory access controller (DMAC) is designed to read/write data from/to dual ram. The DMA controller is a synthesizable intellectual property(IP) core for easy integration into system on chip (SOC) implementation and it enables transferring the blocks of data from memory to peripheral and vice versa. This transfer of data appreciably reduces the load on the processor. The design is implemented in Verilog Hardware Description Language (HDL) and it may use in microprocessor based SOC for high speed data transfer. It increases the data transfer speed as compared to original Microprocessor Unit (MPU) based architecture and provides more data transfer rate compared to conventional DMA method. With 162 Slice Look -Up Tables (LUTS) the design has achieved 306.24 Mhz maximum frequency. The proposed architecture is especially suitable for MPU based SoC for high speed large data transfer.

METHOD OF IMPLEMENTATION

There are three cases of data transference:

- From a peripheral interface to the external memory
- From the external memory to a peripheral interface
- From one memory location to another.

Data transfer from a peripheral interface to the external memory: When data is to be transferred from one peripheral interface, it is first stored into 64 byte FIFO buffer. When this buffer fills up to half its size, i.e. 32 bytes, the DMA controller is notified and the data in the FIFO is being transferred to the external memory. If one of these memory addresses recently have been used by the CPU, the data is stored in the internal cache memory as well. The transfer of data from a FIFO to the memory is done in bursts, and the length of these bursts is either 16 or 32 bytes as chosen by the software programmer in an internal register.

Data transfer from the external memory to a peripheral interface: When the DMA controller receives notice of that a FIFO buffer is becoming half empty, i.e. less than or equal to 32 bytes, it begins a transfer of data from the external memory. When data is transferred from the external memory it is read in bursts of 16 or 32 bytes (as chosen by the software programmer). If one of the memory addresses where the data is located has recently been in use by the CPU, the data is read from the internal cache memory.

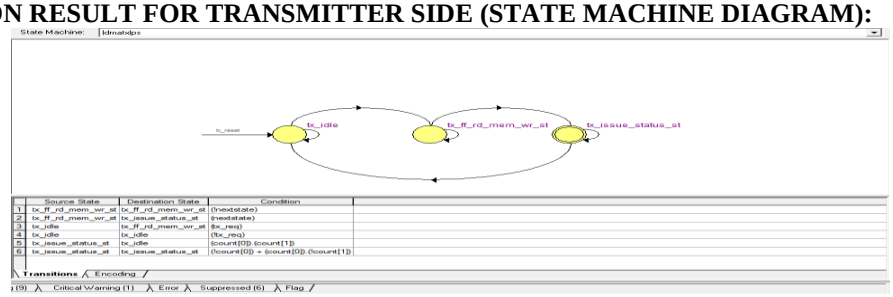
Data transfer from one external memory location to another: In order to increase the performance of the DMA, as with previous transfers the data is read from the external memory when the FIFO buffer is becoming half empty. The data is either read from, or written to the cache - depending on the direction of the data flow - if one of the memory addresses have been in use by the CPU recently.

5. RESULT ANALYSIS

DMA design consists of asynchronous FIFO, DMA engine in both transmitted side (when peripherals want to write) and received side (when peripheral want to read) and a common dual port RAM. The whole design implemented in Verilog HDL for easy integration in SoC. Simulation has done for individual blocks and a complete top-level block architecture using Mentor Graphics tool Modelsim ISE by Mentor Graphics. The simulation result of AMBA based DMAC has achieved and indicates reading just after writing in a single clock.

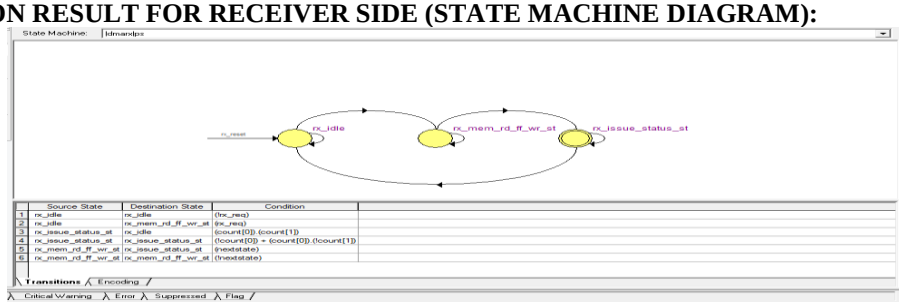
Vol 07, Article 06678; June 2016
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

SIMULATION RESULT FOR TRANSMITTER SIDE (STATE MACHINE DIAGRAM):



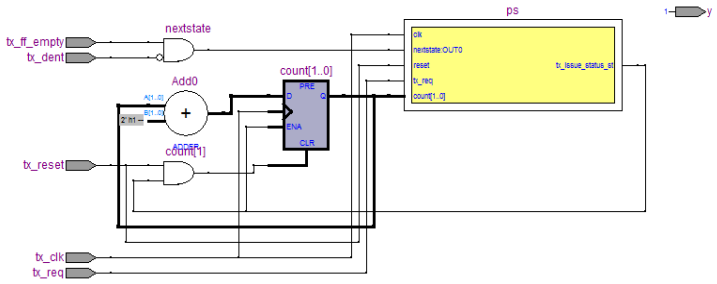
Screen.5.1: Result for transmitter side (state machine)

SIMULATION RESULT FOR RECEIVER SIDE (STATE MACHINE DIAGRAM):



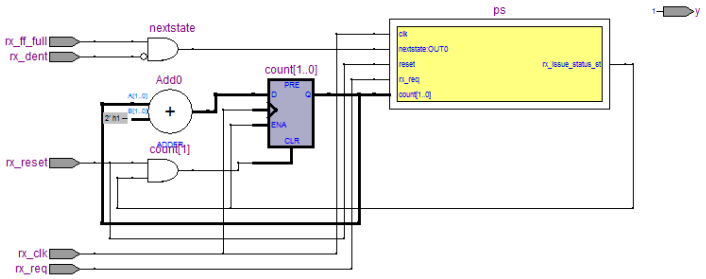
Screen.5.2: Result for Receiver side (state machine)

RTL SCHEMATIC DIAGRAM FOR TRANSMITER:



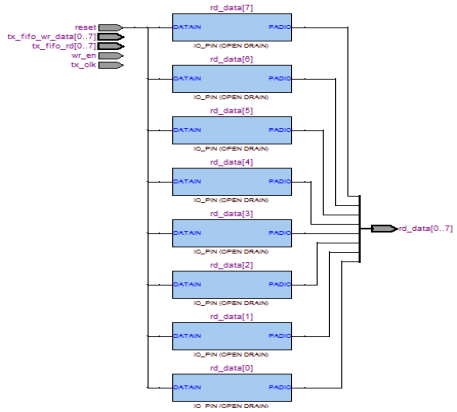
Screen.5.3: RTL Schematic Diagram for Transmitter

RTL SCHEMATIC DIAGRAM FOR RECEIVER:



Screen.5.4: RTL Schematic Diagram for Receiver

TRANSMITTER FIFO INTERNAL DIAGRAM

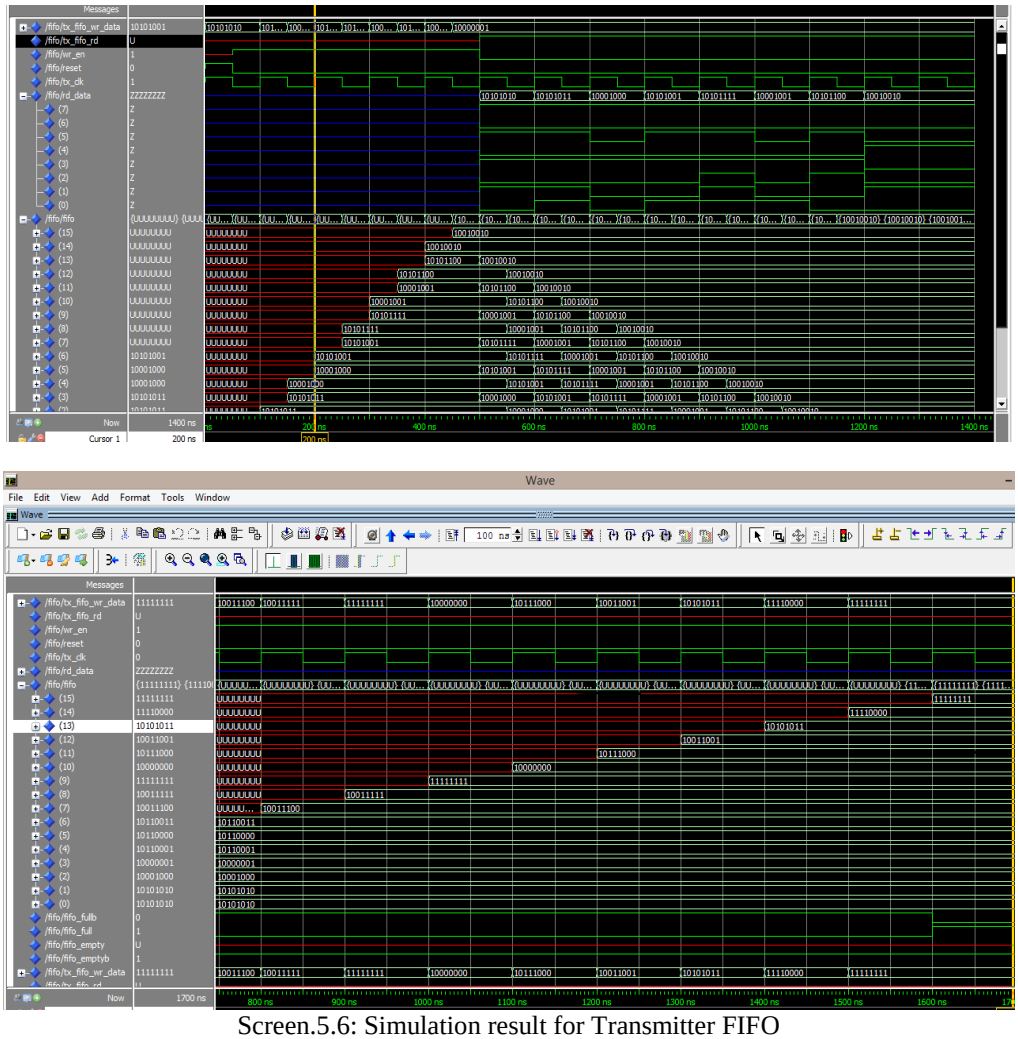


Screen.5.5: Transmitter FIFO internal diagram

Vol 07, Article 06678; June 2016
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

SIMULATION RESULT FOR TRANSMITTER FIFO:

FIFO READ/WRITE:



Screen.5.6: Simulation result for Transmitter FIFO

6. CONCLUSION

AMBA based DMAC module is implemented. Simulation is done on Mentor graphics tool modelsim simulator and synthesis is done on Xilinx tool Xilinx 13.2 ISE synthesizer. The design has achieved AMBA based fast data rate DMAC core for easy integration into SoC product. This design describes how peripherals access data from dual RAM and how it controls address, data and control signals independently and achieves AHB bus and APB bus to run parallel. The DMAC could adapt buffered data transferred mode according to the speed of the peripheral. Experimental results show DMAC has the advantage of high-speed transfer rate and is much suitable for integration in SoC products. It achieved the maximum frequency of 306.24 MHz and minimum time period of 3.265 nsec.

REFERENCES

[1] Ma,G., 2009, Design and implementation of an advanced DMA controller on AMBA-based SoC, Proceedings of the 8th International Conference on ASIC, October 20-23,2009,Changsha,Hunan, pp: 419- 422.
[2] Olugbon, A., S.Khawam, T.Arslan, I.Nousias and I.Lindsay, 2005. An AMBA AHB-based reconfigurable SoC architecture using multiplicity of dedicated flyby DMA blocks. Proceedings of the Asia and South Pacific Conference on Design Automation, Volume 2, January 18-21,2005, New York, pp:1256-1259.
[3] Intel 8237 data sheet
[4] www.arm.com, www.amba.com, www.intel.com