# VLSI Implementation of Direct Memory Access (DMA) Controller

Ankur Changela
Assistant Professor
Electronics and Communication
Indus University

*Abstract:* This paper aims at implementation of Direct Memory Access (DMA) controller which is a part of all modern computers that allows peripheral devices to be able to move large blocks of data without any interaction with the processor. While the device transfers the block of data, the processor is free to continue running the software without worry about the data being transferred into memory, or to another device, or vice versa. The objective of this work is to make Direct Memory Access (DMA) controller schedule the task of peripherals such that they can perform at their own. In this work, DMA controller is designed using Verilog-HDL and testbench is developed to verify the same. Icarus and Design Vision EDA tool have been used to synthesis and simulate the implemented DMA controller.

***IndexTerms* – Direct Memory Access (DMA), Peripherals, EDA Tools**

## I. INTRODUCTION

In computer based data acquisition applications, data transfer through computer I/O devices must often be managed at high speed or in large quantities. Microprocessor has to perform more important task than just a data transfer. Involving Microprocessor to just transfer the data between I/O devices and Memories is inadequate utilization. In such situation DMA controller can be used to communicate between peripherals whereas Microprocessor will be left free to perform other tasks. There are three well known data transfer methods are practiced.

1. Polling-Based Transfer
2. Interrupt-Based Transfer
3. DMA-Based Transfer

A Polling-based transfer continuously polls or checks the peripheral status whether it is ready to transmit or receive data or not. In the polling-based transfer, when the peripheral is ready, the related flag in one of its status registers is set. In order to find out this new status, the computation module should continually checks the status registers of the peripheral in tight polling loops. If the status shows the peripheral is ready to transmit/receive data, the computation module should drop the loops immediately and start to perform a data transfer. After transferring the data item, the computation module computes new data. The microprocessor starts a new loop. The polling-based transfer is simple and requires less hardware. But from the description of the execution, it is clear that whenever data are to be received or transmitted, the status of the peripheral have to be checked first. This checking process can consume a lot of loop executions; therefore take a large amount of time. And during the checking execution, the computation module is not able to do anything else.

An Interrupt-based transfer is to stop the current process of the computation module with an interrupt event so that the computation module is able to participate in another task, indicated by the event. In data handling, a peripheral interrupt is used to handle the data transfer task with the cooperation from interrupt controller. On receiving the INT signal, the computation module reacts to the controller with an Interrupt Acknowledge signal to start the interrupt service. During the interrupt service, the computation module will do the following operations: First, the computation module checks which device gives the interrupt. After the sending peripheral is found, the computation module checks the status that caused the interrupt. If the status is ready for transferring data, the computation module starts the processes that are related to the peripheral and start a data transfer. At the end of the data transfer, the computation module clears the interrupt and resumes the previous process. The interrupt-based transfer is efficient, because the processing operation of microprocessor is only suspended during the data transfer. And it relieves the computation module from having to continually poll for the peripheral status. But since the interrupt function requires a particular hardware, which is called Interrupt Controller, overhead of interrupt must be accepted. If it takes a long time for the computation module to perform data transfer, other processes have to wait until the task is finished.

### 1.1 DMA-Based Transfer:

Direct Memory Access (DMA) is a transfer that allows devices to transfer data from/to system memory to/from system peripheral automatically without intervention of the computation module. When a number of data elements need to be transferred from data memory, the computation module initiates the DMA Controller and grants the bus access to the DMA Controller. The DMA Controller starts a data transfer over the bus system. By the end of the transfer, DMA Controller interrupts the computation module to inform the completion of the data transfer task. The computation module takes control of the bus access from the DMA Controller. The block diagram of the DMA-based transfer is shown in Figure1.

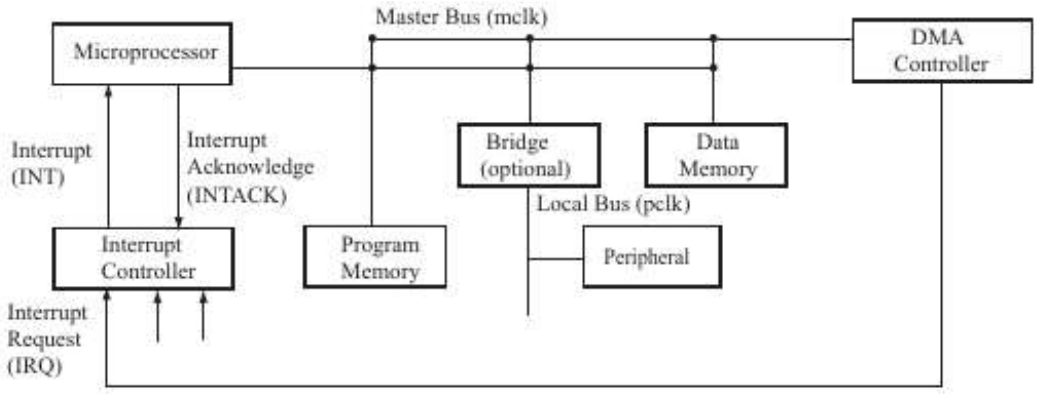


Fig1. DMA based data transfer

Using the DMA-based transfer, a DMA Controller, which leads to extra size and power consumption for the system, has to be introduced into the system. When there are a large number of data elements, considering the less power consumption compared with the computation module and the advantage of offloading the computation module, the DMA-based transfer is the best choice among all the transfers. Otherwise, if the amount of data elements is small (less than 10), it won't be beneficial for the extra size. In this situation, the interrupt-based transfer can be adopted.

## II. DMA ARCHITECTURE AND IMPLEMENTATION.

A DMA Controller can be seen as an interface block among microprocessor, peripherals and bus system. This interface block is a circuit of DMA transfer mechanism built on an interrupt block. Therefore, the DMA Controller consists of an interrupt part and a DMA part. Figure 2 shows a simple structure of the DMA Controller.

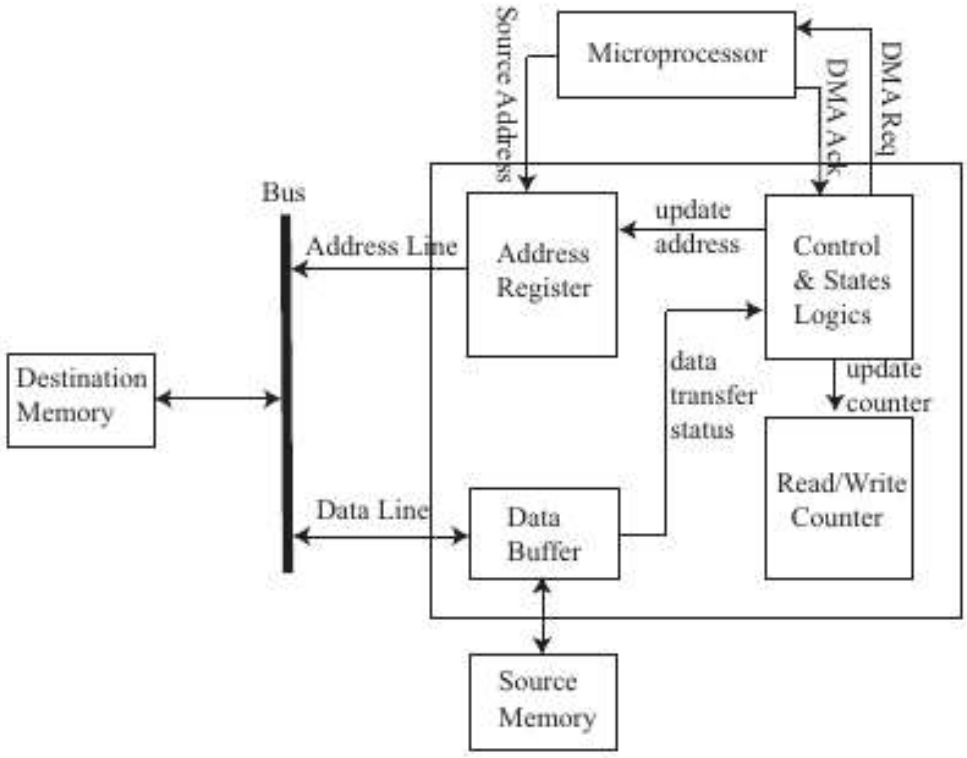


Fig2. Simple Structure of DMA Controller

**Address Register:** Address Register is used to store the addresses of the required data elements. Before DMA transfer, the microprocessor must send the source address to this Address Register. During DMA transfer, at the end of each single data element transfer, the value of the register will increase or decrease by '1'. Thus, the addresses are provided in an incrimination or defragmentation form.

**Read/Write Counter (Word Count) Register:** Read/Write Counter(R/W Counter) is used to record the length (No. of Data words) of the transferred data elements. The initial value of the counter is also set by the microprocessor before DMA transfer. The value equals to the length of the required data elements. During DMA transfer, the value of the counter decrease '1' right after one data

element is transferred. So the R/W Counter performs in a decrement form. When the counter turns to '0', the DMA transfer of the current data group is finished and the DMA Controller send an interrupt signal to the microprocessor.

**Control and State Logics:** This part of the circuit is composed of Control and Timing Logics and Status Flag. It is used to modify the value of Address Register and R/W Counter, set read or write process, and so on. So this section is basically consist of mode register in this work.

## III. FUNCTION MODES OF DMA CONTROLLER.

According to the description of DMA transfer, some basic functions of DMA Controller can be summarized. A DMA Controller can be programmed by microprocessor in order to set the initialization information. A DMA request signal is sent to microprocessor. After microprocessor accepts the DMA request, the DMA Controller starts DMA operation. During the DMA operation period, the DMA Controller can send address signals to address lines, and send read/write control signal to control lines. The size of data elements is also controllable by the DMA Controller. When the data transfer task is finished, the DMA Controller is able to send a complete signal and release the bus. DMA Controller is able to perform either as the system controller or as a peripheral unit which is controlled by microprocessor. Therefore, there are two functional modes for the DMA Controller existing in the system. They are Active Mode and Passive Mode.

**3.1 Active Mode:** In active mode DMA transfers the data between I/O and system memory. So DMA is read data either from memory or from I/O and write it to either I/O or to a memory. So in active mode DMA became a bus master. In this mode DMAC is generates control signals for the read and write for both memory and I/O devices.

**3.2 Passive Mode:** In this mode the internal registers of DMAC can be written or read by processor. So this is basically programming mode because DMA can be programmed by processor in this mode. When CS and HLDA is low at that time DMAC is going in programming mode.

## IV. DATA TRANSFER TYPES

There are three transfer types during the DMA transfer. They are single word transfer, block transfer and demand transfer.

**Single Transfer Type:** In single transfer mode, the DMA Controller is programmed to perform one transfer only. The read/write counter will decrease by one and the address will increase /decrease following each transfer. After the transfer, the DMA Controller is halted and release the bus. When next word of DMA transfer is needed, the DMA Controller has to request again.

**Block Transfer Type:** In this type of transfer DMAC is transfer the blocks of the data until the word count becomes zero. Here the transfer is done sequentially means single block is read from the source and after write to the destination after that word count is decremented by one and address is incremented /decremented by one and then new transfer is takes place.

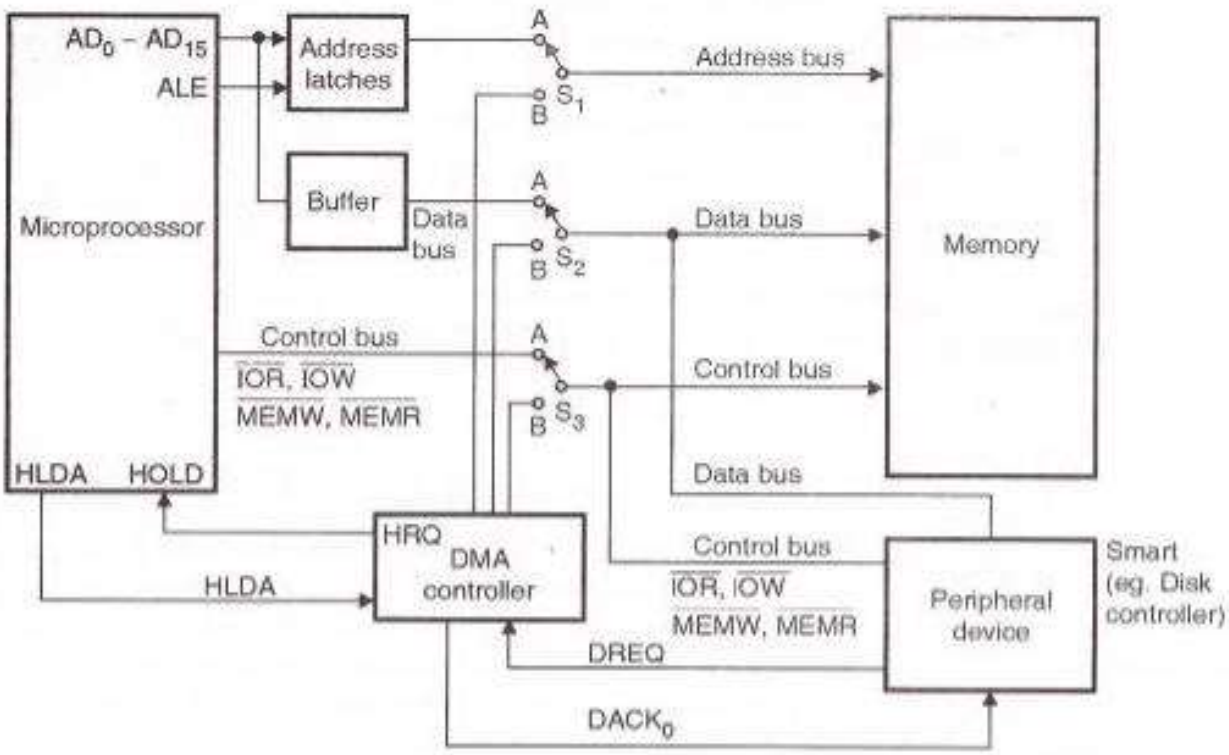


Fig3. DMA Controller connection

**Demand Transfer Type**: Demand transfer type is similar to the block transfer type, except that after each data element is transferred, the DMA Controller checks whether the DMA request is valid. Therefore, it's a way of polling as well. If the request goes inactive, the DMA Controller will release the system bus. Otherwise, the DMA Controller continues to make transfer until the block is completely transferred. And in this case, the DMA Controller performs as the block transfer type. Therefore, by controlling the active or inactive of the DMA request signal, a data block can be divided into several parts, and transferred separately each time period sequentially when the I/O device is ready.

## V. Feature of Implementation and Synthesised Design

Implemented DMA controller has four channels. DMA channels shows the no. of I/O peripherals can be interfaced with DMAC. Each Channel consist of pair of pins DREQx and DACKx for handling request of I/O devices. This DMAC have four DMA channels so it have 8 pins e.g. DREQ0, DREQ1, DREQ2, DREQ3, DACK0, DACK1, DACK2, DACK3. Implemented DMA controller has also auto reload mode. In auto reload mode, initial values of these register is stored in their base register so after the completion of transfer if auto reload mode is selected then this register reload initial values from its respective base registers so that if any peripheral send request then transfer takes place with previous programmed registers. Fig.3 shows the basic connections of DMA controller with peripherals and microprocessor. Fig4 shows the gate level netlist of synthesized design.

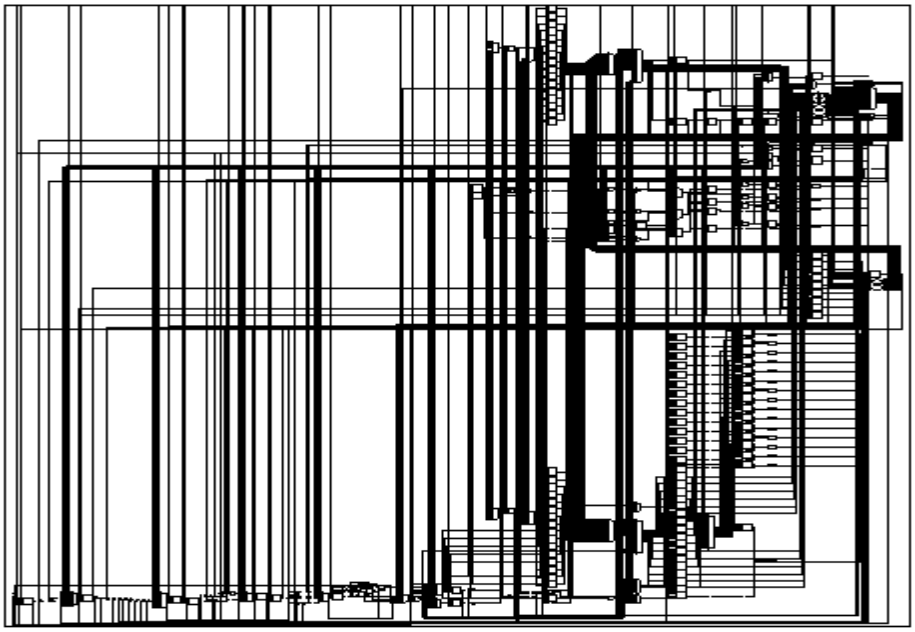

Fig.4 Gate level net list

## VI. SIMULATION RESULTS AND CONCLUSION

Implemented design is simulated with selecting different mode and peripherals. Fig.5 shows the simulation result for Memory read and I/O Write with Auto reload and Address increment Mode. Fig.6 shows the simulation result for Effect of Auto reload mode in next transfer and Fig.7 I/O read and Memory Write without Auto reload, Address decrement.

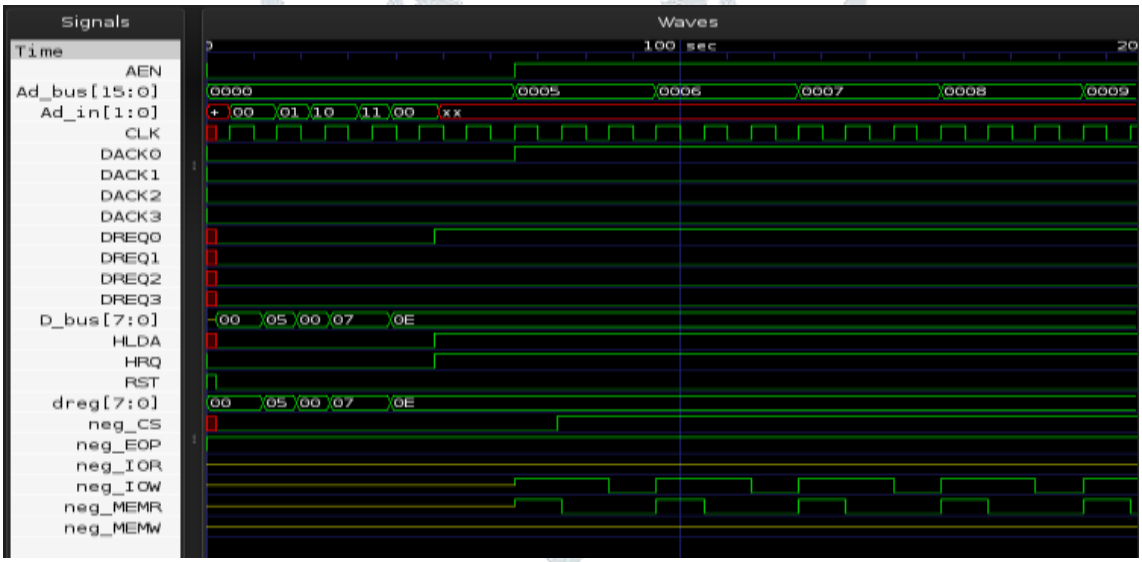


Fig.5 Memory read and I/O Write with Auto reload and Address increment Mode


Fig.6 Effect of Auto reload mode in next transfer


Fig6. I/O read and Memory Write without Auto reload, Address decrement.

## VII. CONCLUSION

The RTL Simulation of DMA controller has been verified and validated by suitable test benches. The Synthesis of DMA controller has been successfully completed by the extraction of Synthesized Net-list with unit delays. And we have also obtained desired gate level net list to implement a DMAC using Verilog with help of Icarus and Design Vision EDA tool. The Output generated was also cross verified with outputs represented in the DMAC specification document.

## REFERENCES

**[1]** S. Palntikar. Verilog HDL Guide to Digital System Design and Synthesis, 2nd Edition, Pearson.

**[2]** C. H. Yu, C. K. Liu, C. H. Kang, T. H. Wang, C. C. Shen and S. Y. Tseng, "An Efficient DMA Controller for Multimedia Application in MPU Based SOC," *2007 IEEE International Conference on Multimedia and Expo*, Beijing, 2007, pp. 80-83.

**[3]** D. Comisky, S. Agarwala, C. Fuoco, "A scalable highperformance DMA architecture for DSP applications", *Proc. of International Conference Computer Design IEEE*, pp. 414-419, Sept. 2000.

**[4]** *TI 0MAP5912 Multimedia Processor Direct Memory Access (DMA) Support Reference Guide*, March 2005

**[5]** Guoliang Ma and Hu He, "Design and implementation of an advanced DMA controller on AMBA-based SoC," *2009 IEEE 8th International Conference on ASIC*, Changsha, Hunan, 2009, pp. 419-422.

**[6]** Haihua Liu, Xinhao Chen, "Weighted Priority Rotational Algorithm on PCI Bus Arbitration", *Computer Engineering and Applications*, vol. 36, 2003.

**[7]** 8237A High Performance Programmable DMA Controller – Data Sheet of 8237A by Intel.