International Journal of Electrical & Electronics Research. (IJEER)
Volume 4, Issue 1, Pages 1-4, March 2016, ISSN: 2347-470X

ASIC Implementation of DMA Controller

Aman Agarwal
School of Electronics Engineering,
VIT University, Vellore, India
aman.agrawal01@gmail.com

Arjun J Anil
School of Electronics Engineering,
VIT University, Vellore, India
arjun44anil@gmail.com

Rahul Nair
School of Electronics Engineering
VIT University, Vellore, India
rahulnair.nair9@gmail.com

K. Sivasankaran
School of Electronics Engineering
VIT University, Vellore, India
ksivasankaran@vit.ac.in

ABSTRACT- Direct Memory Access is a method of transferring data between peripherals and memory without using the CPU. It is designed to improve system performance by allowing external devices to directly transfer information from the system memory. We generally use asynchronous type of DMA as they respond directly to input. The DMA controller issues signals to the peripheral device and main memory to execute read and write commands. In this paper DMA controller was designed using Verilog HDL and simulated in Cadence NC Launch. The design was synthesized using low power constraints. Through this design we have decreased the power consumption to 69%**.**

Keywords- Direct Memory Access, Asynchronous, Synchronous, MIPS, TSMC.

1. INTRODUCTION

Direct Memory Access (DMA) is a computerized system feature in which the peripheral will access the main system memory independently of the CPU. External peripheral will take the control of system bus from CPU. It is normally used for high speed data transfer from /to mass storage peripherals. In the absence of DMA CPU will be engaged in the read write cycle and find it hard to manage the other works. DMA starts the data transfer by CPU initiation. CPU will perform other works at this time. After data transfer DMA will interrupt the CPU. DMA transfer is require when CPU cannot match with the data transfer or when CPU have to wait for a slow data transfer.

During a bus cycle any one peripheral will be connected to the system bus. Then this component becomes the master and the other device it is communicating with becomes the slave. Generally CPU is the master having bus control. But some specially designed components can send a bus access request to the CPU. After the bus cycle which is currently running, CPU will grant the bus control to the requested device and the device becomes the master. This process is called bus stealing. This component must be able to take the control of bus by placing the address and they are called processors because of this application.

DMA controller designed in this project is such a processor. Whenever an external peripheral requires a data transfer, it will send DREQ (DMA Request) to the DMA controller. In response to this, DMA will send a HREQ (Hold Request) signal to the Central Processing Unit. After receiving the HREQ signal, CPU will suspend the execution and turn its data, address and control buses in high impedance state. Then CPU sends back HLDA (Hold Acknowledgment) [2] signal. There after DMA takes the control of Data, Address and control buses and performs the data transfer from external memory to system memory. CPU overhead is very less during DMA controlled data transfer.

DMA is a speedy synchronization method and shows much improvement over interrupts with respect to latency and throughput [1]. Sometimes the external mass storage cannot keep up with data rate of the core processor and DMA allows the peripheral to access the memory directly without using the core. In processing applications, data transfer is most crucial factor. Therefor DMA can improve the performance of the system.

The organization of this paper is describes by following section. Section 2.shows the problem formulation of DMAC and its stage. Section 3.describes the architecture of the DMA control system in which design of each block given and explained. Section 4.describes result of DMA.

In this paper, the main objective is to reduce the power consumption by DMA controller by achieving this parallel data transfer mechanism [3]. The functional verification has been done by RTL coding and the results were simulated using NC Launch. Synthesis was done using RTL compiler, formal verification was completed. The physical design was completed using SoC Encounter using 180 nm technology.

2. PROBLEM FORMULATION

DMA design is proposed for high speed data transfer between peripheral and memory to reduce CPU overhead. The core is prototyped such that it provides HOLD acknowledgement signal which facilitates DMA transfer. According to the peripheral request, there are two cycles:

- DMAR (DMA Read cycle): Corresponds to memory to IO device data transfer.
- DMAW (DMA Write mode): corresponds to IO device to memory data transfer.

3. ARCHITECTURE

The basic block diagram illustrating the functioning of DMA data transfer is given below. The main blocks are DMA controller, CPU, DMA, MEMORY, PERIPHERAL DEVICE and ADDRESS LATCH

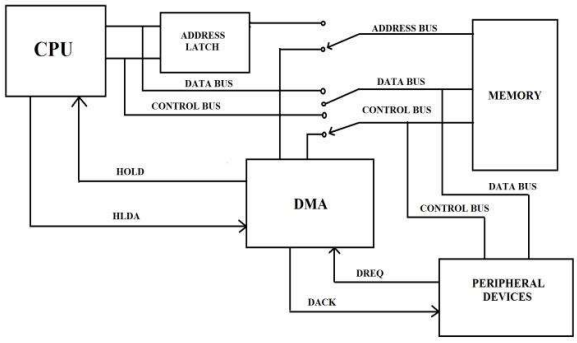

Fig.1 Block Diagram of DMA Controller

DMA: DMA access the data, address and control buses from CPU and reduces the CPU overhead over the data transfer and thus improving the transfer rate.

CPU: CPU grants the data, address and control buses to DMA controller when a request for the access is received. CPU will takes the control of buses when data transfer is over.

MEMORY: This block acts as the system memory which reads from or writes into the external memory.

PERIPHERAL DEVICE: Acts as the external mass storage which wants the access to system memory. It sends a request to DMA for data transfer initially.

3.1 CPU:

In this paper, CPU is prototyped by a module having internal memory of 16 bit width and two ports. ALU is given as a sub-block of CPU itself. Whenever Write Enable (we) signal is high, it access data from system memory. These data are the operands for the ALU. By the user input opcode, the corresponding operation is performed on these operands and result are given out. CPU will perform its operation until DMA is in idle state or when the terminal count is one.

3.2 MEMORY:

Main memory is having registers for storing 16 bit data controlled by reset input. By default the memory values will be invalid values. If reset signal is deactivated we can provide 16 bit input values to main memory.

3.3 PERIPHERAL:

Peripheral is designed same as CPU memory. The mass storage devices are prototyped using 16 bit wide registers. According to DMA control signals such as DMA READ and DMA WRITE, the peripheral can access the internal memory.

3.4 DMA CONTROLLER:

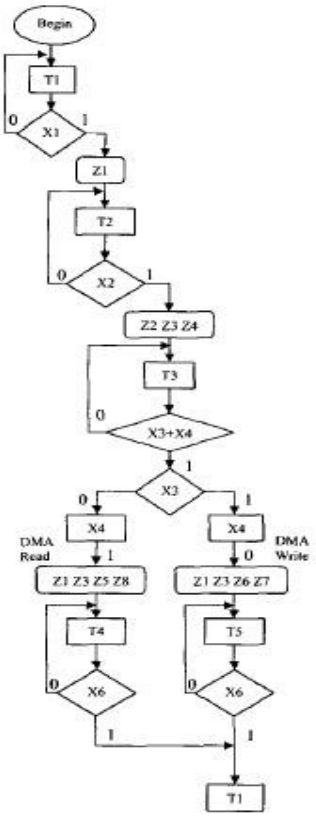

Fig.2 ASM Chart For DMA

The initial state of controller is T1. Whenever an error is occurred in DMA operation, the processor resets the controller state to T1. The peripheral which want to perform data transfer send DREQ(X1) signal to DMA. In response to this DMA sets HREQ (Z1) output to 1 informing the CPU about peripheral request. This is T2 state. CPU respond by placing the buses in high impedance states and giving HLDA(X2) signal. DMA gets the access and sends DMA acknowledgement/DACK (Z2) in state T3. DMA raises AEN/Address Enable (Z3) and ADSTB (Z4). Peripheral-controller-microprocessor request/ acknowledgement handshaking is completed.

3.5 DMA READ:

Memory device to input output device data transfer. Corresponds to MEMORY READ (MR) and INPUT OUTPUT WRITE (IOW). The mode select pins MS1 and MS2 determines the operation to be read or write.

MS1	MS2	Operation
0	0	Do nothing
0	1	DMA Write
1	0	DMA Read
1	1	Do nothing

Table 1. Mode Select Operation

For DMA read, X3=0, X4=1. T4 is the new state. The HRQ (ZI), AEN *(23)*. MEMR (Z5) and IOW outputs are activated. The MEMR and IOW signals enable data transfer from memory to the *I/O* Device while the HRQ signal is used to hold the buses and AEN latches the particular addresses. When Terminal count (X6) signal is received, DMA read stops. The state returns to TI.

3.6 DMA WRITE:

For DMA write, X3=1, X4=0. T5 is the new state. The HRQ (ZI), AEN (23). MEMW (Z5) and IOR outputs are activated. The MEMW and IOR signals enable data transfer from memory from the I/O Device while the HRQ signal is used to hold the buses and AEN latches the particular addresses. When the Terminal count (X6) signal is received, DMA write stops. The state returns to TI.

4. RESULTS

The DMA controller was designed using Verilog HDL and simulated in Cadence NC Launch. The design was synthesized using low power constraints and physical design was developed using Cadence Encounter using 180 nm Technology. The delay of the critical path is 7.2 ns, which is the maximum frequency is 139MHz. The layout of DMA read and write mode is given in *Table I and* In Table II area and power values of synthesis are listed. Physical design obtained is given in *Figure 3*. Area and power values after physical design are listed in Table III.

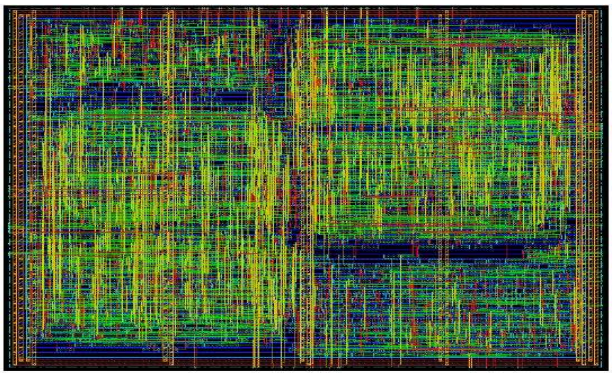

Fig.3 Physical Design Layout

Fig. 4 Simulation Result

TABLE II: Post Synthesis Results

	Post Synthesis Values		
Stages	**Area(μm^2)**	**Timing(ps)**	**Power(mW)**
Post Synthesis	21090	95	3.051

TABLE III: Physical Design Results

	Post Synthesis Values		
Stages	**Area(μm^2)**	**Timing(ps)**	**Power(mW)**
Post Synthesis	84653.6	Hold Time = 231 Setup Time = 99	3051.972 7.946

5. CONCLUSION

DMA controller design for high speed and synchronous data transfer between external peripheral and memory using Verilog HDL is accomplished successfully. The proposed specification DMA controller was designed and implemented using TSMC 180nm technology. The simulation, synthesis and physical design was carried out. The signals from the CPU are generated using code. In the synthesis the design was optimised for the low power and speed. Also the physical design was carried out using 6 metal layer process and all the signals were routed with minimum congestion and effective area.

REFERENCES

[1] Aghdasi, F., "Self-Clocked Asynchronous Controllers", University of Zimbabwe Publications, Harare, 1996, pp 34-58.

[2] Brzozowski, J.A., "Asynchronous Circuits", Springer-Verlag, New York, 1995 pp 315-318.

[3] Vibhu Chinmay, Shubham Sachdeva "A Review Paper on Design of DMA Controller Using VHDL"

[4] Clement A. 'The Principle of Computer Hardware", ***Td*** Edition, Oxford University Press, 1996.

[5] Oommen, A., Murudkar, F., Aghdasi, F. "Self-Clocked Asynchronous Sequential Circuits",Botswana institution of Engineen-Conference Proceedings 1999

[6] Brzozowski, J.A., "Asynchronous Circuits", Springer-Verlag, New York, 1995

[7] F. Aghdasi and A. Bhasin "DMA Controller Design using Self-Clocke