AMBA Bus with Multiple Masters Using VLSI

[1]Miss. Dhage Naiyna Kashinath, [2]Prof. S.I. Nipanikar,
[1]ME (VLSI and Embedded System), [2]Assistant Professor
Department of Electronics and Telecommunication,
[1]G.S.Moze College of Engineering, Savitribai Phule University of Pune
[2]PVPIT, Bavdhan, Savitribai Phule University of Pune

***Abstract* - Implementation of DMA controller of AMBA Bus with two masters is described in this paper. DMA controller is connected to the AMBA AHB Bus. The Direct Memory Access (DMA) Controller is a hardware feature. It enables movement of blocks of data from peripheral to memory, memory to peripheral, memory to memory and peripheral to peripheral. This movement of data reduces the load on the processor. A DMA controller save power in a system by putting the CPU in a low power state. DMA controller is used to move the data. Architecture of DMA controller for AMBA bus consists of DMA system, Host and Arbiter. Arbiter gives response to DMA system and Host. Three buses are defined within AMBA specification i.e. The Advanced High-Performance Bus (AHB), The Advanced System Bus (ASB), The Advanced peripheral Bus (APB). The proposed architecture provides bus access to any one master at a time for improved speed and performance.**

***Keywords* - DMA, AMBA, SOC, power state.**

I. INTRODUCTION

Microprocessor is the most important part of system. Microprocessor is able to access peripherals via special bus called Advanced Microprocessor Bus architecture (AMBA). AMBA bus is simpler in architecture than any other buses. The AMBA bus is applied easily to small scale SOCs. Three buses are defined within the AMBA specification. Advanced High-Performance Bus (AHB), Advanced System Bus (ASB), Advanced Peripheral Bus (APB).

AMBA BUS

In Advanced Microcontroller Bus Architecture (AMBA) specification three buses are defined. Advanced High- Performance Bus (AHB), Advanced System Bus (ASB), Advanced Peripheral Bus (APB). The ASB is the older form of system bus, with AHB being introduced later to improve support for higher performance. The APB is generally used as a local secondary bus which appears as a single slave module on the AHB or ASB. Typically AMBA bus structure is shown in Fig.1.

1. Advanced High- Performance Bus (AHB)

The high-performance bus which is the main system 'backbone'. This bus is also able to sustain the data rates required by the external bus interface. The CPU and other bus masters (such as a DMA controller), and high-speed local memory are normally connected to this bus. AHB is also specified to ensure ease of use in an eff1cient design flow using synthesis and automated test techniques. AHB multiple bus masters and provides high bandwidth operation, and 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, and wider data bus con- figurations (64/128 bits).

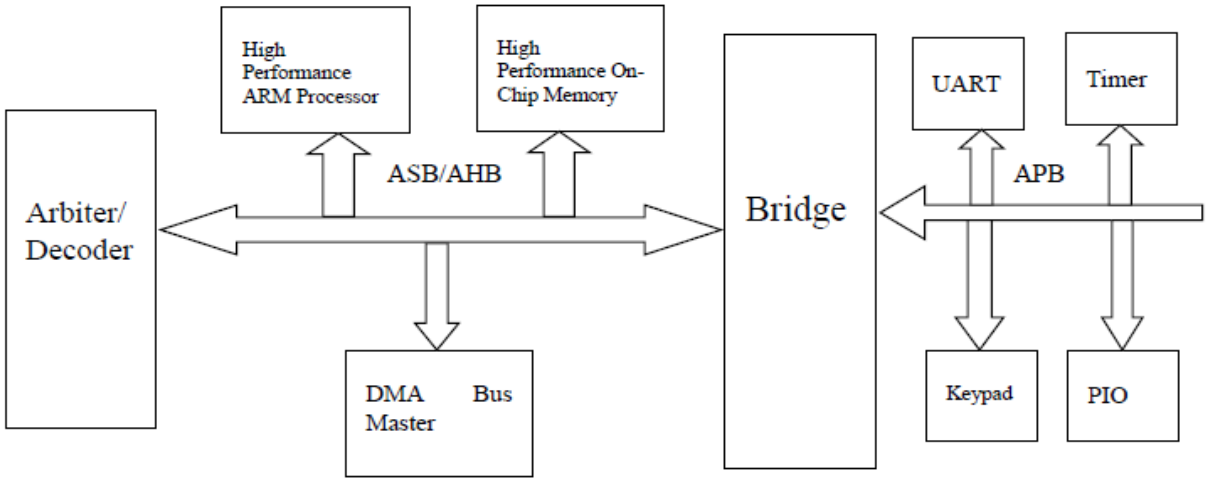

Fig.1: A typical AMBA BUS architecture

2. Advanced System Bus (ASB)
The ASB is designed for high-performance, high bandwidth usage. It is non-multiplexed (i.e. separate) address and data buses. Support for pipelined operation (including arbitration). Support for multiple bus masters and multiple slave devices, including a bridge to the peripheral bus (APB).

3. Advanced Peripheral Bus (APB)
The simple, low-speed, low-power peripheral bus. This is often, but not always, a narrower bus and is designed to be simple (i.e.unpipelined) for connecting many common peripherals such as timers, parallel I/O ports, UARTs, etc. (By placing these infrequently accessed peripherals on the APB, and partitioning them away from the AHB, loading on the AHB is reduced and allows maximum performance on the AHB to be more readily achieved.)

II. FEATURES OF AMBA AHB BUS

DMA controller is connected to the AMBA AHB bus. An AMBA AHB design may include one or more bus masters. Typically a system would contain at least the processor and the test interface. Direct Memory Access (DMA) or Digital Signal Processor (DSP) is the most common AHB bus masters. The external memory interface, the Advanced Peripheral Bus (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. A typical AMBA AHB system design includes the following components:

- AHB Master
- AHB Slave
- AHB Arbiter
- AHB Decoder

AHB Master- A bus master is able to initiate read and write operations by providing an address and control information. At any one time only one bus master is allowed to actively use the bus .
AHB Slave - A bus slave responds 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.
A DMA controller save power in a system by putting the CPU in a low power state. DMA controller reduces the use of CPU at the time of data transmission. So that at this time, CPU kept free for doing any other operation. For simulation we are using Xilinx 14.3.

III. PROPOSED ARCHITECTURE

The main idea of this paper is to design DMA controller for AHB of AMBA Bus with multiple masters. In our design, we are implementing DMA controller of AMBA Bus with two masters. A bus master is able to initiate read and write operation by providing address and control information. Only one bus master is allowed to actively use the bus at any one time. In our design, we are including Direct Memory Access (DMA) as one of the master. Another master is Host which is included just for demo purpose.
A direct memory access (DMA) is an operation in which data is copied (transported) from one resource to another resource in a computer system without using CPU.
Fig.2 shows the proposed architecture of DMA controller for AMBA bus. It consists of DMA system, Host and Arbiter. The DMA system consists of DMA, Peripheral 1, Peripheral 2 and arbiter. DMA consists of memory and control unit.
The DMA controller, which consists of memory and control unit is the basic module of our project. The control unit will control the reading and writing operation. When peripheral 1 want to read data from memory or write data into memory, peripheral 1 will send enable signal high. When peripheral 1 wants to read the data from memory, it will make read write bar signal high. Similarly when peripheral 1 wants to write data into memory, it will make read write bar signal low. This is same with peripheral 2. When peripheral 2 want to read the data from memory, it will make read write bar signal high and it will make read write bar signal low to write the data into memory.
But at a time only one peripheral can work, that means only one peripheral can take the access of BUS for reading or writing the data. So that we are also designing arbiter to give priority between peripheral 1 and peripheral 2. Arbiter is used for priority. The arbiter will take the request from peripheral 1 and peripheral 2 and will grant the response to peripheral 1 and peripheral 2 respectively. But when both the peripherals will give the request at a time then arbiter will give response to peripheral 1 and peripheral 2 alternatively. One more arbiter is used to give priorities between DMA system and the host. When DMA system and host give the request at the same time then arbiter will give response to DMA system and host alternatively. After application of clock signal, peripheral 1 and peripheral 2, both are requesting bus for writing operation to the control unit. The control unit will grant bus access alternatively to both the peripherals

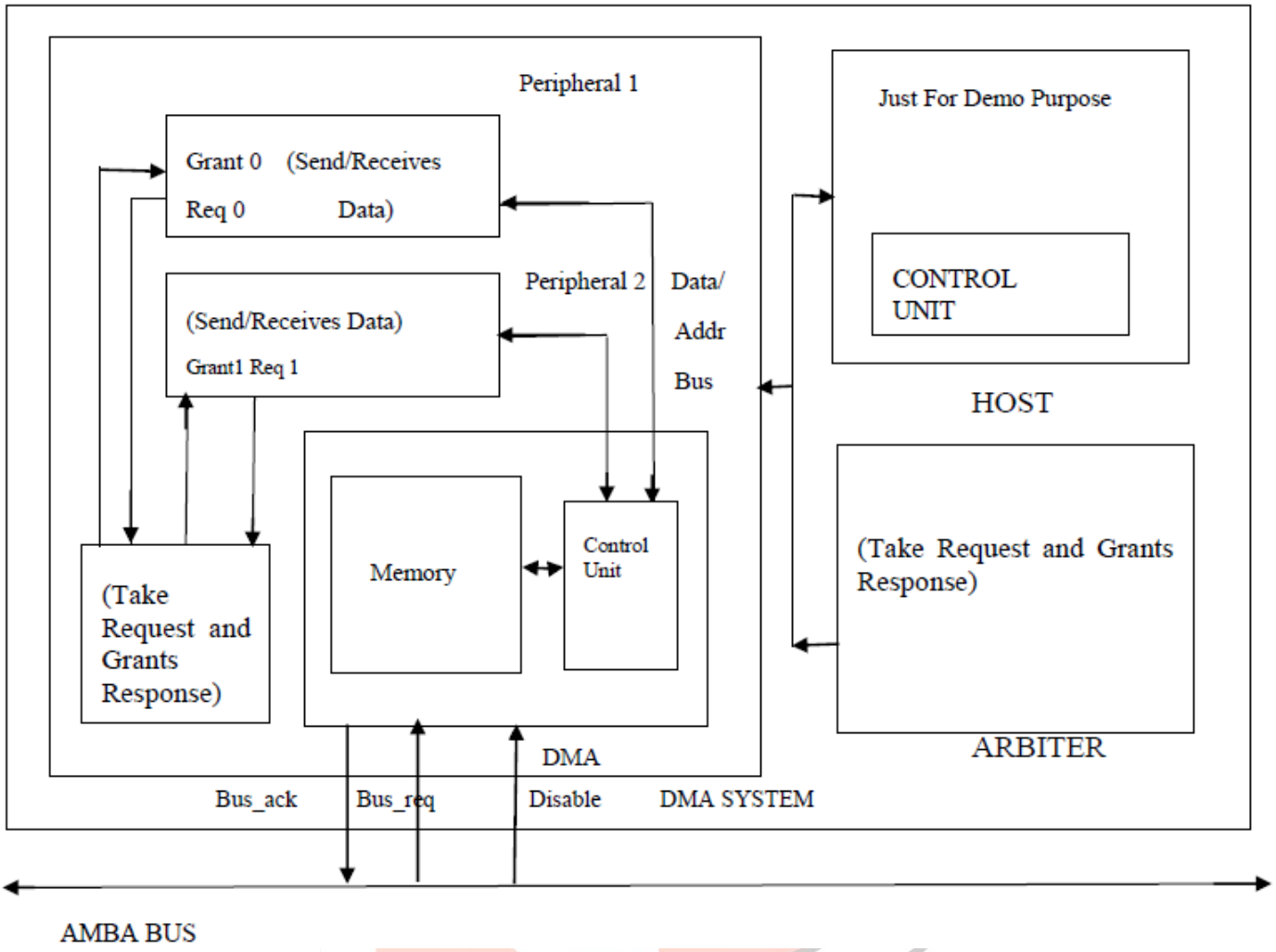

Fig.2: Proposed Architecture of DMA Controller for AMBA Bus

After getting bus access, peripheral sends address, data, and read /write signal to control unit. The data given by the peripheral is then written in the memory location whose address is also provided by the peripheral. The data which is given by peripheral 1 and peripheral 2 is written in the memory location whose address is also given by the peripheral 1 and peripheral 2 respectively.
After writing operation, reading operation starts where data is read from memory. The data from selected memory location is read by the peripheral.

IV. ALGORITHM

We can transfer data from memory to memory, memory to peripheral, peripheral to memory and also from peripheral to peripheral. Now we will see the algorithms.

1. Algorithm for memory to memory transfer
1. Start
2. Send bus request
3. Bus grant for transfer
4. Ready for data transfer
5. Send data to destination address
6. Wait for data transfer
7. Ready for next transfer
8. End

2. Algorithm for Peripheral to memory transfer
1. Start
2. Send bus request
3. Bus grant for transfer
4. Send data to internal buffer
5. Send data from internal buffer to memory

6. Wait for data transfer
7. Ready for next transfer
8. End

3. Algorithm for Peripheral to peripheral transfer

1. Start
2. Send bus request
3. Bus grant for transfer
4. Send data to internal buffer
5 Internal buffer transfer data to peripheral
6. Wait for data transfer
7. Ready for next transfer
8. End

V. RESULT

1. **Memory to memory transfer**

Fig.3 shows memory to memory transfer. Here we transfer data 00000001, 00000010, 20000000, 20002000 from one memory location to another.

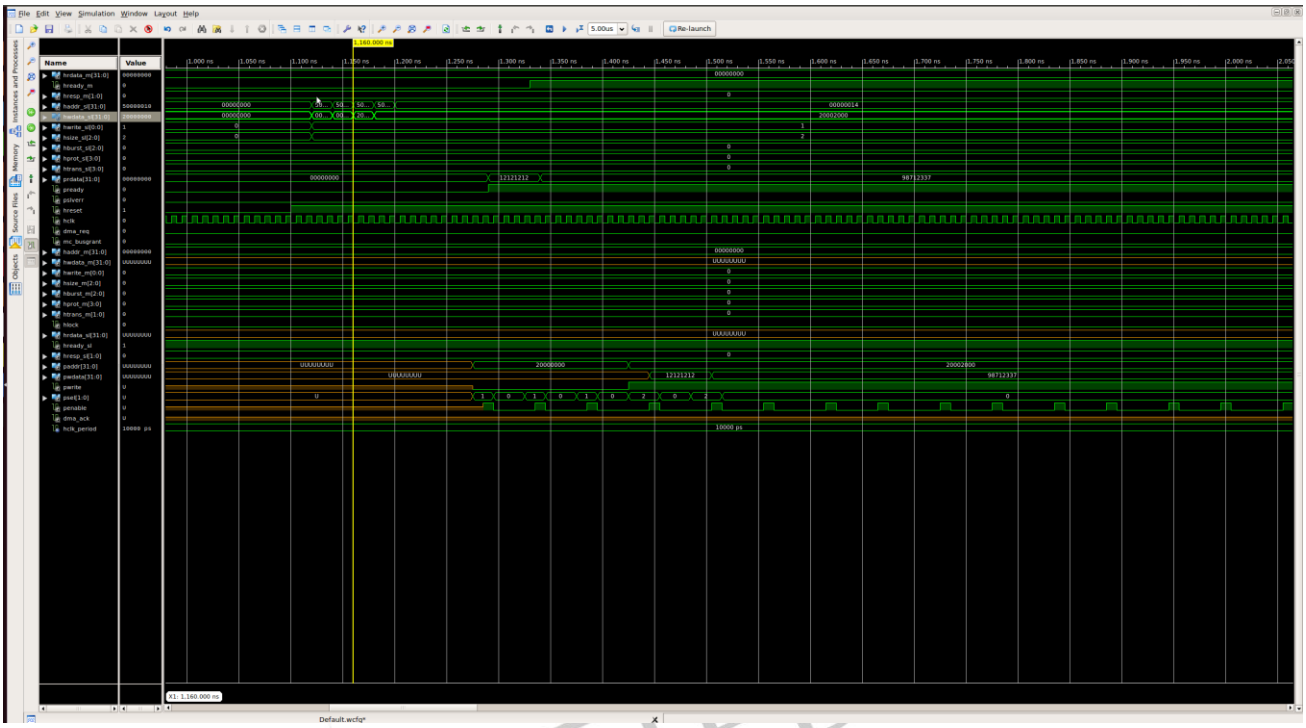

Fig.3: Memory to memory transfer

2. Memory to peripheral transfer

Fig.4 shows memory to peripheral transfer. Here data 12341234 is transferred from memory to peripheral.

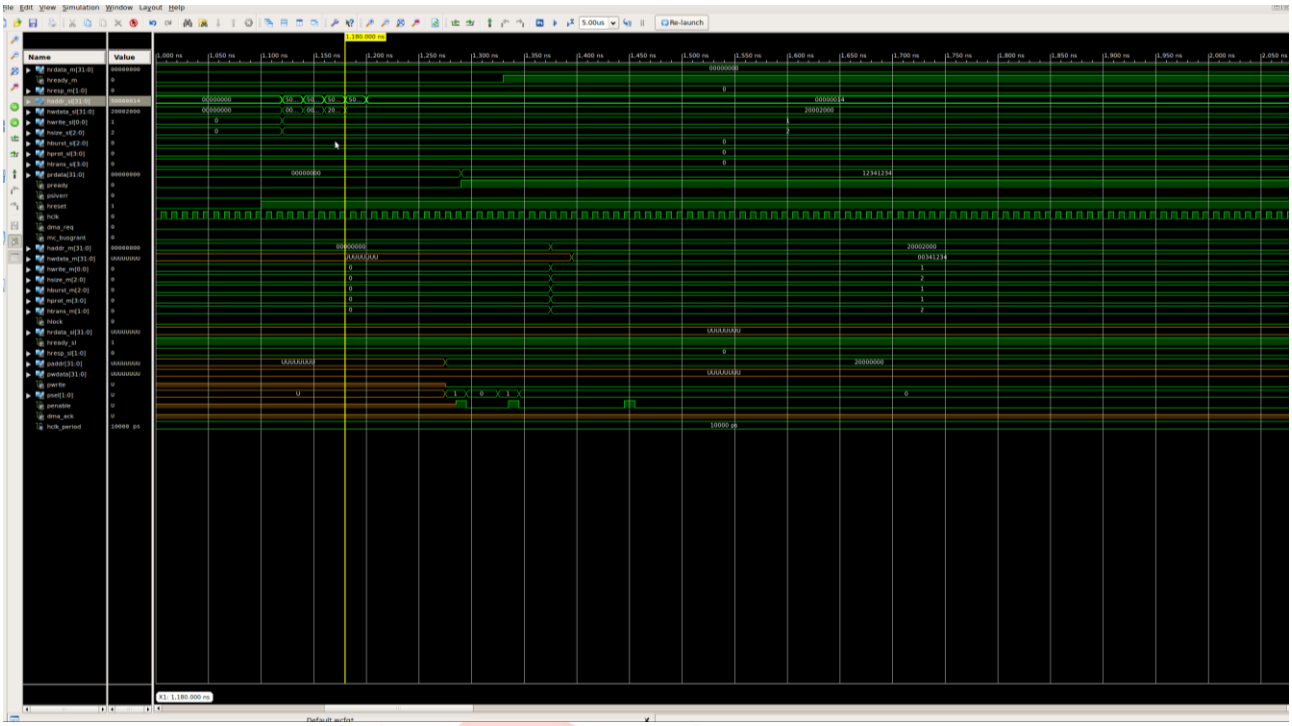

Fig.4: Memory to peripheral transfer

3. Peripheral to memory transfer

Here data 12341234, 77777777, 98712337 is transferred from peripheral to memory as shown in fig.5.

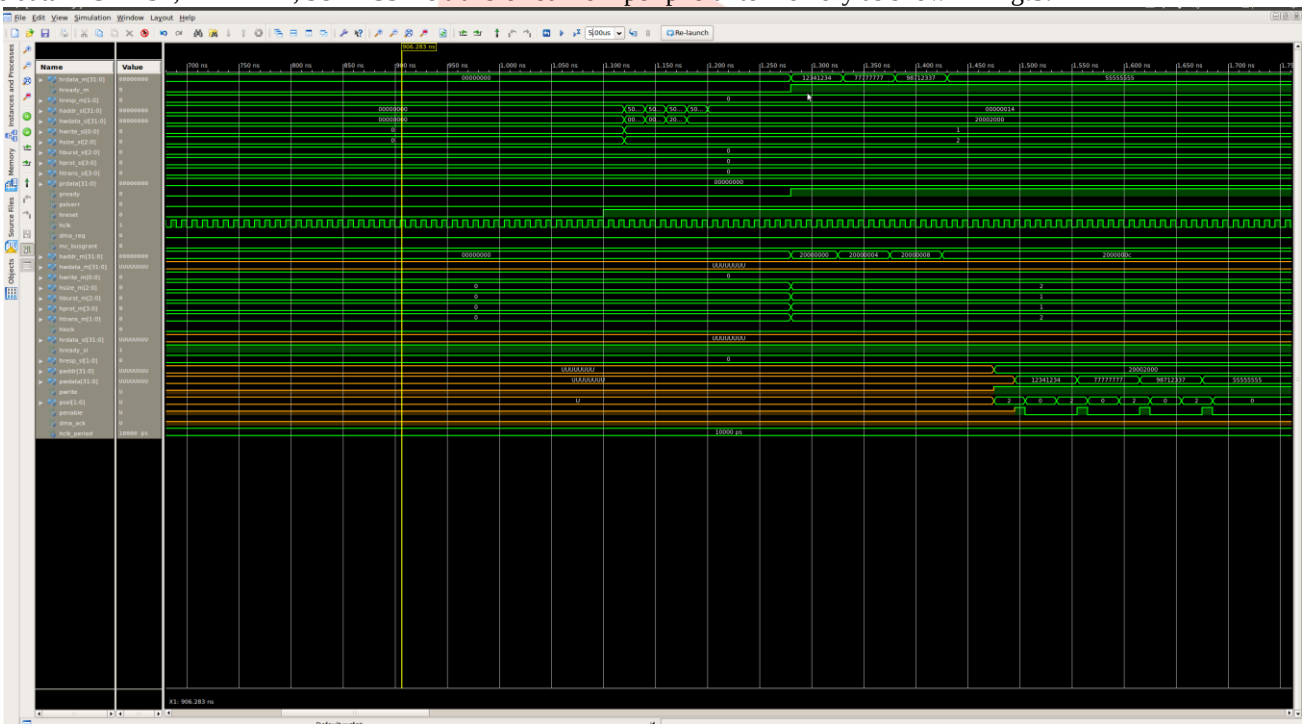

Fig.5: Peripheral to memory transfer

4. Peripheral to peripheral transfer

Fig.6 shows Peripheral to peripheral transfer. Data 12344321, 78787878, 98712337 transferred from Peripheral to peripheral transfer.

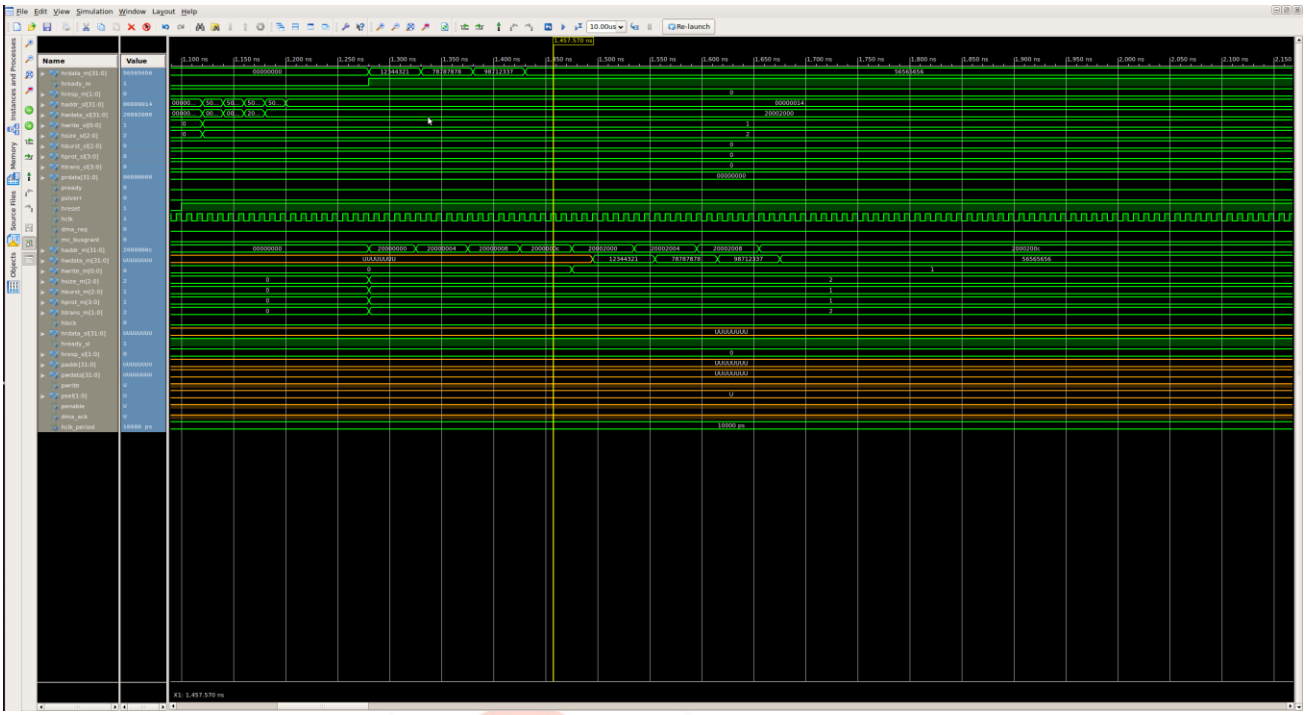

Fig.6: Peripheral to peripheral transfer

VI. ADVANTAGES

1. DMA allows for faster processing since the processor can be working on some other operations while the peripheral is populating the memory.
2. The DMA works in parallel with the CPU, thereby simulating a multi-processing environment and effectively increasing the CPU's bandwidth.
3. The longer the CPU runs, the more power is consumed. Using DMA to offload the CPU thereby reduces power consumption.
4. Offloading the CPU results in more idle time for the CPU. This frees up available processing capacity for future product enhancements.

VII. CONCLUSION

In this paper we present design of DMA controller AMBA bus with multiple masters one of the master is direct memory-access i.e. DMA and another master is host. Only one bus master either DMA or Host is allowed to actively use the bus at any one time. Arbiter is used to give response to DMA system and Host. Simulation is done by Xilinx (14.3). We can transfer the data from memory to memory, memory to peripheral, peripheral to memory and peripheral to peripheral. A DMA controller save power in a system by putting the CPU in a low power state. DMA controller reduces the use of CPU at the time of data transmission.

REFERENCES

[1] International Journal of Information Technology and Knowledge Management January-June 2011, Volume 4, No. 1, pp. 285-288

[2] F. Polmi,D. Bntazzi, L. Bmini, and A. Bogliolo., "Psrformancc Aoalysirr of Arbibation Policics forsd: Communication Architecfurcs", Kluwa Jolnnal on Design Automation for Embedded Systmu, 8, np. 2, pp.189-210, 2003.

[3] AMBA Bus DMA Controller Specification, Draft 1.03, Gareth Morris 2000.

[4] J. Liang, Swaminathan, S, "ASOC: a Scalable, Singlechip Communications Architecture", Tessier, R. Parallel Architectures and Compilation Techniques, 2000, Proceedings.International Conference, pp.37-46, 2000.

[5] AMBA Specification (rev2.0) and Multi Layer AHB Specification, Arm: http://www.arm.com, 2001.

[6] www.iaeme.com/ijecet.asp