A

DISSERTATION REPORT

on

Design and Implementation of AMBA-AHB based memory controller

by

RAKESH GEHALOT

2015PEV5103

Under the supervision of

Mr. RAKESH BAIRATHI

Associate professor, ECE Department
MNIT, Jaipur

Submitted in partial fulfillment of requirement of degree of

MASTER OF TECHNOLOGY

to the

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

JULY – 2017

Department of Electronics and Communication Engineering
Malaviya National Institute of Technology, Jaipur

Certificate

This is to certify that this Dissertation report entitled ***"Implementation and design of AMBA-AHB based memory controller"*** by **Rakesh Gehalot** (2015PEV5103), is the work completed under my supervision and guidance, hence approved for submission in partial fulfillment for the award of degree of ***Master Of Technology*** in ***VLSI Design*** to the Department of Electronics and Communication Engineering, Malaviya National Institute of Technology, Jaipur in the academic session 2016-2017 for full time post graduation program of 2016-2017. The contents of this dissertation work, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Mr. RAKESH BAIRATHI
Associate professor, ECE Department
MNIT, Jaipur

DECLARATION

We hereby declare that the Dissertation work entitled Implementation and design of AMBA-AHB based memory controller, is an authentic record of work carried out as requirements of Dissertation for the award of degree of M.Tech in VLSI Design from Malaviya National Institute of technology Jaipur, under the guidance of Mr. Rakesh Bairathi, during July, 2016 to July, 2017.

(***Signature of Student)***

RAKESH GEHLOT
(2015PEV5103)

Acknowledgement

I take this opportunity to present my vote of thanks to all those guidepost who really acted as lightening pillars to enlighten my way throughout this project that has led to successful and satisfactory completion of this study. I am really grateful to my HOD for providing me with an opportunity to undertake this project in this Institute and providing all facilities. I am highly thankful to my mentor **Mr. Rakesh Bairathi** for his active support, valuable time and advice, whole-hearted guidance, sincere cooperation and pains-taking involvement during the study and in completing the assignment of preparing the said project within the time stipulated. Lastly, I am thankful to all those, particularly the various friends , who have been instrumental in creating proper, healthy and conductive environment and including new and fresh innovative ideas for me during the project, their help, it would have been extremely difficult for me to prepare the project in a time bound framework.

RAKESH GEHALOT

(2015PEV5103)

Abstract

Memory controller is a part of digital system. Memory controller works as a bridge between microprocessor and memory. It is used to control the data flow (read and write) between microprocessor as master and memory as slave. Traditionally memory controllers were placed outside the microprocessor. External memory controller will introduce more latency than on chip memory controller. Nowadays memory controller is on same die as microprocessor which makes a system on chip and it reduce the latency. A design of AMBA-AHB based memory controller is presented here. The Memory controller is based on ARM Advance Bus Protocol and the concept of interleaving is used to improve the throughput of memory controller. The result of interleaving is compared with FIFO buffer for same purpose. The concept of interleaving is to use memory access time or latency to access the next address from another bank of memory. The controller given here significantly improves memory access time also. The memory controller is designed and synthesized using Xilinx design tool ISE. The Verilog language is used for design purpose. Detail synthesis report is also generated by Xilinx tool which includes the maximum frequency and delay path.

Content

List of Figures

List of tables

Glossary

AMBA: Advanced Microcontroller Bus Architecture is a popular bus protocol using in system on chip and design by ARM.

AHB: Advanced High-performance Bus Protocol is developed by ARM. It is used in high bandwidth devices.

APB: Advanced Peripheral Bus protocol is developed by ARM for low bandwidth devices in SoC.

AXI: Advanced extensible Interface protocol.

ACE: AXI Coherency Extensions.

CHI: Coherent Hub Interface

CPU: Central processing Unit

DMA: Direct memory access

RTL: Resister Transfer Level, Hardware synthesized code used in hardware description language.

SoC: System on chip, A large number of functional blocks on a single die.

VLSI: Very Large Scale Integration, A large number of transistor combined on a single die.

CRC: cyclic redundancy check is an error-detecting code generally used in storage devices and digital networks to detect changes in stored data.

SSD: solid state drive, It is a secondary storage device.

HDD: Hard disk drive, It is a secondary storage device.

RAM: Random access memory.

ROM: Read only memory.

DDR: Double data rate, It is a feature of RAM to sample data at both positive and negative edge of system clock.

MCU: Memory Control Unit, Memory control unit is used for efficient use of memory.

NOC: Network on chip.

IP: Intellectual property

Chapter – 1

INTRODUCTION

1.1 Introduction

Memory is a physical device capable of storing data in the form of digit 1 and 0. This data can be used for future actions.
Memory is used to store data or information for immediate use or for a long term use of data. There are two broad categories of memory in computing first is the primary memory and second is the secondary memory.

1.2 Memory hierarchy

In computer architecture design concept of discussing issues related to performance is known as memory hierarchy. The memory hierarchy is divided into different levels based on response time. Since complexity, capacity and response time are related, these levels of memory hierarchy can also be distinguished by their controlling technology and performance. In figure 1.1 memory hierarchy is shown.

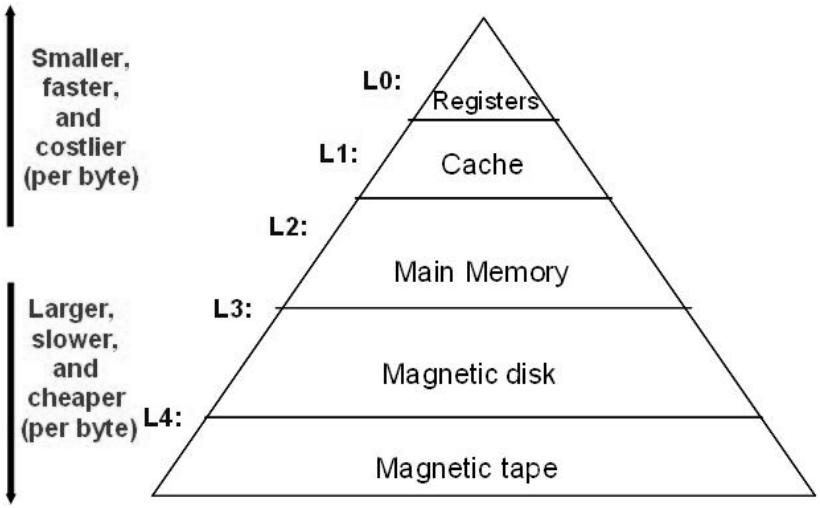

Figure 1.1: Memory hierarchy [21]

1.2.1 Primary Memory

The computer uses the memory to store the program which will be executed in near future. These programs are stored in primary memory by a computer to use frequently in near future. Primary memory also called main memory or physical memory. In computer speed of processor is higher than the memory device. To access the data from memory will lead to slow down the overall throughput. The speed of primary memory is higher than the speed of secondary memory. Computer continuous read and write data from and to primary memory.

Main memory is connected to processor directly or via cache memory or registers. It has two buses one for data transfer (DATA BUS) and another for an address (ADDR BUS). The first CPU sends the address, the location where it wants to write or read data. Then CPU sends data as information if it is a write cycle or if it is a read cycle then it will retrieve the data from memory. In addition, we can use memory management unit for translating virtual address to physical address which is used as virtual memory.

1.2.1.1 Cache Memory

Cache memory is most tightly connected memory with computer processor. It is the fastest memory but also more costly than any other category of memory. This memory is used in fewer amounts inside the CPU because it is expansive. Most frequently used information is copied from main to cache for increasing the performance. There can be multilevel cache which has different locality with the processor.

1.2.2.2 Main Memory

Main memory is the memory used in the computer actively for storing the instructions

of multiple programs. When CPU needs data to perform the action it will copy that data from secondary memory to main memory so that it can access it fast. A copy of that data will be there in secondary memory also.

We cannot use secondary memory as main memory because access time of secondary memory is more as compare to main memory. As we know main memory is volatile memory if power is down then data stored in main memory will not retain. Main memory is used for temporary data storage by the central processing unit. After CPU finishes executing with that data it will clear that part of data from main memory and load another required data for execution. We cannot copy all required data of all application which are currently in action because of the limited size of RAM. So to use the RAM in an efficient manner developer make their applications or program to use fewer amounts of memory by copying only needed data from secondary storage to main memory. In this method, data are swapped in between RAM and secondary storage according to requirement. This technique called swapping. Deferent algorithms are used to make room for important data and swap in main memory. Main memory also called as RAM which stands for Random access memory or Physical memory as the term used in the concept of virtual memory.

Types of main memory

There are two types of random access memory in the modern era of technology.

1.2.2.1.1 SRAM (static random access memory)

SRAM uses six transistors to store one bit of data. There is no need for any refresh circuitry to retain data in a cell because it stores the data as long as power is switched on. There is no leakage charge in SRAM. SRAM is more expansive than DRAM and it takes more area per bit than DRAM but it is faster than DRAM. In computers SRAM used as cache memory.

Static random access memory uses 6 transistors per cell. But there are 4T, 8T SRAM also available. The SRAM with six transistors is shown in figure 1.2. There are four transistors crossed coupled making inverters. Other two transistors are used for read/write operation. There are two stable states 0 and 1. In transistors, there is a layer of poly-silicon used for high resistance pull-up resistors.

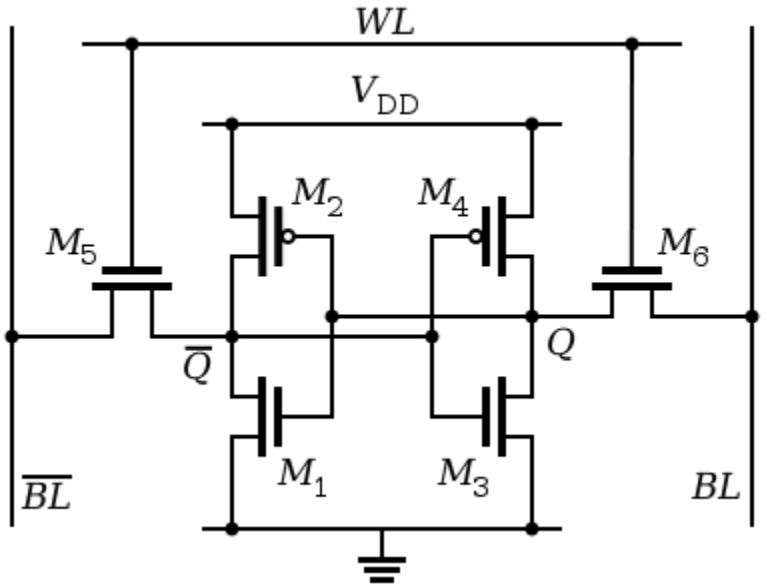

Figure 1.2: SRAM with six transistors. [12]

1.2.2.2.2 DRAM (Dynamic random access memory)

DRAM is a volatile memory used in computer and other devices. This memory uses one transistor and one capacitor to store one bit of data. Because of the only one transistor is used in DRAM. It will take less amount of area per cell as compare to SRAM. DRAM is less expensive and higher density of bit cell than SRAM. There is leakage of charge in capacitors of RAM so refresh circuitry is used to retain data in memory. There is different state for recharging the cells in RAM.

DRAM consumes a large amount of power because it needs to refresh every cell in

each clock cycle. The sense amplifier is used to detect the level of charge stored in a capacitor.

1.2.1.3 Read only memory

Read only memory is a non-volatile memory use to store data for a long time in the computer or other electronic devices. Data can be stored in ROM multiple times but at slow speed. This memory is a hard wired memory such as memory made of diode or mask memory. Generally, data are written one time only inside the ROM. This is a disadvantage in many application because there can be a need for an update of codes or BIOS for security issues. But nowadays ROM can be rewritten many times.

In modern technology, there are ROMs in which data can be modified even after writing multiple times. These ROMs called programmable read only memories. This programmable ROM can be programmed multiple times but at slow speed.

ROMs are used for storing initial startup programs like BOOT sequence or factory settings in electronic devices. There are many error check algorithms are used to detect any failure in read only memory. For example: checksum.

ROM can be manufactured at low cost than the cost of a RAM. There is no write operation so ROM is simpler than RAM and has low cost than RAM. In modern computer, Programmable ROM is used so that BIOS can be upgraded if there is a need. But for other electronics device like the keyboard, mouse the one time programmable ROM is used.

There are many types of read only memories on the basis of number of rewriting or technology used in fabrication.

1. Mask read only memory:

   MROM is a one-time programmable memory. The Information is stored at the time

of manufacturing of memory. To write a bit in MROM, the mask is diffused. One advantage of MROM is its cost. Per bit cost of this memory is significantly lesser than other semiconductor memories. But if there is any error while writing the data then memory is a waste. It cannot be used again. [6]

2. Programmable read only memory:

   This memory is one time programmable memory (OTP). Special PROM programmer is used to write data in this memory. This programmer will destroy the fuse inside the memory to write a bit.

3. Erasable programmable read only memory

   This memory can be programmed multiple times. To erase the written data inside memory UV light is used. Then rewriting process begins with high voltage to write inside the memory.

4. Electrical erasable programmable read only memory

   This memory is similar to EPROM. But it uses different way to erase the data from memory. A block of data can be deleted from memory. There is no need of removing the memory from CPU. High voltage is used to delete a bit from memory. The process of erasing memory is very slow.

1.3 Memory transfer cycle

Memory transfer cycle is the transfer of addresses, control signals and data in single clock cycle of system. CPU reads the instructions from memory by sending address and control signals to memory. Memory will respond to CPU by providing the required data after the access time of that particular memory. Time from sending address and control signal to retrieving data from memory is called memory transfer cycle.

1.3.1 Memory Read cycle

In memory read cycle the CPU will first send the address of required data on address bus. After that it will send the control signals like chip select, write enable, output enable. Memory will drive the required data on data bus and assert the data strobe signal to indicate that data is valid on data bus. Figure 1.3 shows a memory read cycle.

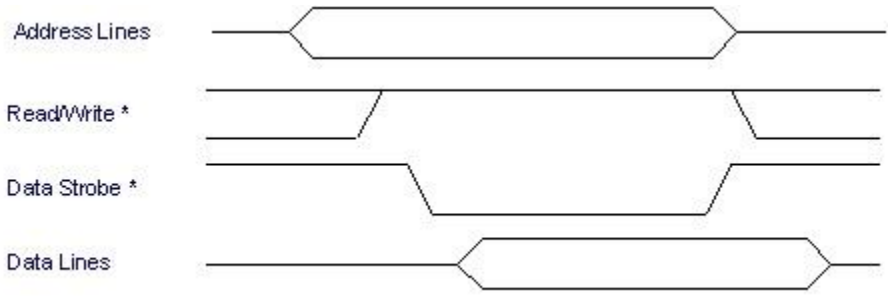

Figure 1.3: Memory read cycle [13]

1.3.2 Memory write cycle

In memory write cycle the CPU writes the data to a particular location in memory. CPU first sends the address on address bus. After that it will send the control signals like write enable and output enable. Data are driven on data bus. Memory will send data acknowledgement signal for verification of successful data write in the memory. In figure 1.4 showing a data write cycle.

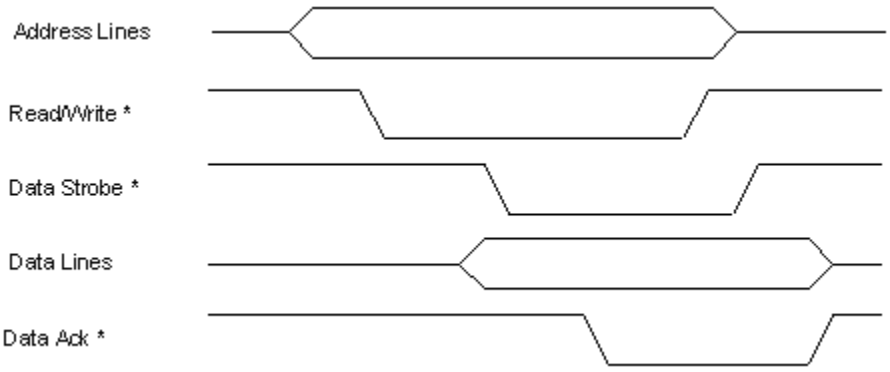

Figure 1.4: Memory write cycle [13]

1.4 Main Memory and its influence on system performance

It has been seen that by increasing of a computer memory will increase the performance. But if memory is lesser to accommodate all required information at a time then computer will use the virtual memory. The virtual memory is a space reserved in secondary memory to simulate additional RAM in the system. This concept of using secondary memory as RAM is called swapping. Method of swapping makes the system slower. The average time to access the RAM is 200ns and the average time to access secondary memory is 12000000ns. This is equivalent to a normal tasks of 3 ½ minute which will now take 4 ½ months to complete. [11]

Technically RAM does not affect the performance of computer. RAM does not have the power to increase system performance. The computer uses the RAM to store the instruction of a program. Processor will search the instruction to be executed, in main memory. If main memory is absence in the system then processor has to search the instruction in secondary memory which will take more time to access the instruction. A large amount of RAM means more programs can fit at time in memory. All present computer system have multitasking concept where multiple programs can run at a time. For example we can run notepad and browser at same time.

There are many situations when a program is larger than total available RAM at that time system will store the extra amount of data into a swap file in hard disk. The required data or instruction to be executed will be loaded into main memory from swap area. This will introduce more latency in the system. The transfer rate of RAM is roughly 800 MB/s as compared to hard disk which has transfer rate 30-100 MB/s. So every time computer exchanges the data between swap and RAM it takes time. This will affect the performance of the system. [11]

1.5 Memory interfacing method

Interfacing the memory with the processor is a way of communicating between two devices. Interfacing is of two types.

1. Memory mapped interfacing:

In memory mapped interfacing the address bus is shared with memory and other peripheral devices. The address bus is used to decode chip select. Extra hardware is required for decoding purpose. Arithmetic and logic operations can be directly performed with memory and I/O devices.

2. I/O mapped interfacing:

In this interfacing, I/O devices are connected to a memory device. Generally, 8 bit of address bus is used for interfacing. The processor uses same control signals and instructions to access I/O as those of memory, here RD and WR signals are activated indicating memory bus cycle. For address bus of width 8 bit, 256 of input or output devices can be connected.

1.6 Memory access method

There are two type of memory access method

1. Synchronous memory access

In synchronous memory access, method reference clock is used to access the memory. The processor can access data from memory at any time with the reference clock. There is no need of waiting for arbiter time. Data will be accessed with raising edge or falling edge of the clock.

2. Asynchronous memory access

In asynchronous memory access data is accessed without reference clock. There can be

a random wait time of data arrival from memory. This random time is the latency of the memory. If the CPU has transferred data while a window is open, and if a subsequent clock cycle occurs while that window remains open, the CPU cannot transfer additional data until the next window opens, thereby wasting that clock cycle. Asynchronous operation forces the CPU to conform to a fixed schedule for transferring data, rather than doing so whenever it wishes.

1.7 Memory controller

The memory controller is digital circuit used to manage the read and write operation with system memory. Memory controller can be integrated on same die with processor or it can be a separate chip. Memory controller also called memory control unit (MCU).

In past, memory controllers are designed separately from processor and it has off chip latency to access the data from RAM. Integrated memory controllers are manufactured with processor to reduce the latency and increase the system performance. The problem is that memory controller will not be portable. Because of that we need to redesign the memory controller for new type of RAM.

The concept of integrating memory controller with the same die of microprocessor is not a new concept. In 1990s there are some microprocessors like Alpha 21066 had the memory controller integrated with same die of microprocessor. This concept is used to reduce cost of system, not to reduce the latency of the system. The external memory controller will increase the cost of system. [14]

Memory controllers are designed to control the read, write, refreshing of RAM. Without refresh circuit the data stored in RAM will lose with time (time to refresh is 63 millisecond according to standard). Without memory controller CPU will spends all its

time in refreshing and deciding when to read or write the data to a RAM. So memory controller allows CPU to know when it can write or read a data from RAM and it also refreshes the memory each time at negative clock edge.

Reading and writing operation are done by selecting rows and columns of RAM. Data are access on the data bus and address is provided on address bus. Data bus width can be 8-bit, 16-bit, 32-bit depends on type of RAM. Address width is also 32-bit or 16-bit depends on RAM type and processor. Number of bits in address can tell how much amount of data can be accessed. For example if the address bus is 32 bit then 4 GB of address can be accessed by this system. Sometime there is second type of address translation used by memory controller as first translation is done by memory management unit.

There can be multiple master wants to access the RAM for data. A memory controller allows multiple masters to get it's turn to access the data from RAM where hand shake and bus grant signal are used to provide the access. A master can be CPU, Camera, and LCD etc.

Different variant of memory controller

1.7.1 Double data rate memory controller

Double data rate MC is used in DDR RAM. In DDR RAM data are sampled at both raising edge and falling edge of each clock cycle. DDR ram controller is more complex than simple ram controller because of sampling the data at both negative and positive edge. But have advantage of transferring data at double speed than simple ram controller.

1.7.2 Dual channel memory controller

Dual channel memory controller is used when multiple masters are accessing the same

RAM simultaneously. It needs multiple data transfer channels for data transfer. More channels can be added to increase the bandwidth of system. The length of every channel should be equal so access time for each channel will be equal. So it's more difficult to increase the number of channel because it will introduce the line capacitance and wire count.

1.7.3 Fully buffered memory controller

Fully buffered memory controllers are used, where buffer is placed at each memory module unlike simple memory controller. Instant of parallel link, in buffered memory controller serial link is used for data transfer, this will decrease the number of wire used for buses. This will help to accommodate more memory controller on a single die because less area is required for this type of memory controller. But this will also increase latency. Memory controller will convert the serial link to parallel link at master side.

1.7.4 Flash memory controller

Nowadays flash memory is for large data transfer in many devices like USB pen drive, camera, mobile, memory card etc. Flash memory controller is used in flash memories to transfer the large amount of data. Flash memory is generally slower than RAM.

1.7.5 Direct memory access controller

Direct memory access is the concept of accessing the RAM (read/write operations) without interpreting the CPU for each read and write. Without DMA to access the data from RAM, CPU will spend its all-time in reading and writing the data and cannot get the bandwidth for other work. So DMA will help CPU to focus on other work while it will handle the data transfer with RAM without interrupting again and again. CPU will first initiate the data transfer sequence and then gives the command to DMA to transfer the whole data. DMA will start transferring the data to ram to other place. After DMA

finishes it will interrupt the CPU to tell the data transfer is finished. This will give more bandwidth to CPU for other important tasks.

Direct memory access has many applications in network cards, graphic cards, disk drive, sound cards and cameras. DMA also used in intra chip data transfer. Computer which has DMA channel can transfer data at minimum overheads with processor then computer without DMA channel. Using DMA data can be transferred from memory to memory without interrupting the processor.

In typical on chip system or embedded systems, the bus architecture is a complex system used to connect all functional blocks like AMBA AHB protocol. There are two types of components in AMBA AHB master and slave. A typical slave can be memory chip or other peripherals. A master can be a processor or other controller like DMA controller. A DMA will use interface for transaction between memory and system without disturbing CPU.
A system which needs to transfer large amount of data like network devices can be considered as master and salve. AHB do not support tri state buses.

1.7.6 Interleaved memory controller

In interleaved memory controller, memory is organized in multiple memory banks and each bank is accessed in pipelined way. It will improve the overall throughput of data transfer. Memory controller can n-ways interleaved if memory is divided in n number of banks.
If there is n number of memory banks are present then memory location A will be at memory bank A/n. In memory interleaving each consecutive address is at different memory bank. For example, if we have four memory bank then memory address 00 will be at bank 0, memory address 01 will be at bank 1, memory address 10 is at bank 2 and memory address 11 is at memory bank 3.

1.8 Bus Protocols

Bus protocols are set of rules used to transfer the data between multiple devices in an efficient way. Every bus protocol has some control signals, data bus, address bus. Generally bus protocol need to perform read and write operations between different devices.

1.8.1 ARM bus protocol

The ARM Advanced Microcontroller Bus Architecture (AMBA) is an open Source bus protocol used in system on chip for connection and management of functional block. ARM AMBA is designed for multiple masters and slave configuration for efficient communication. This bus protocol used in microcontroller for accessing the memory and many other peripherals. Nowadays AMBA is very popular bus protocol used on different range of SOC and Application specific ICs for example smart phones, cameras, embedded systems etc.

ARM AMBA was first developed by ARM in 1996. There are many AMBA protocols used in system on chip. The first AMBA bus protocol is Advanced Peripheral Bus (APB) and Advanced system bus (ASB). AMBA version 2 is introduced in 1999 which includes AMBA High performance Bus (AHB). The AMBA version 3 protocol is introduced in 2003 which is a third generation of AMBA protocol. This generation also includes Advance Extensible Interface Protocol (AXI) and Advance Trace Bus (ATB) for debug and trace solutions. The AMBA and AXI version 4 is introduced in 2011. [16]

In system on chip there are number of components are designed as a block to increase the portability and reuse of these functional blocks. Now to connect those large number of blocks needs a good algorithm or protocol for communication in between them. AMBA bus protocol gives solution for interfacing these functional blocks.

Features of AMBA protocol:

- Facilitate right-first-time development of embedded microcontroller products with one or more CPUs, GPUs or signal processors.
- Be technology independent, to allow reuse of IP cores, peripheral and system macro cells across diverse IC processes.
- Encourage modular system design to improve processor independence, and the development of reusable peripheral and system IP libraries.
- Minimize silicon infrastructure while supporting high performance and low power on-chip communication. [17]

In Advanced Microcontroller Bus Architecture, there are multiple microcontrollers masters along with multiple salve devices like RAM, ROM, external memory, DSP, other peripherals like UART, USB, I2C, PCI are connected according to AMBA specification.

Main motive of ARM bus protocol is to develop an efficient and standard way for interconnecting the functional block with reusability capability in different design. ARM AHB (Advanced High performance) or ASB (Advanced System Bus) protocol are used for higher speed functional block like processor and an APB (Advanced Peripheral Bus) protocol is used for low speed device like peripherals for example slow RAM, UART etc.

AMBA bus protocols are introduced first time in 1996 by ARM. The evolution of AMBA protocol till now the latest version is AMBA 5 protocol. [17]
Following are the most popular AMBA protocols used nowadays and specification of these protocols can be downloaded from ARM official website for free.

1.8.1.1 APB (Advanced Peripheral Bus)

The Advanced Peripheral Bus (APB) is used to connect slow peripherals like RAM, UART etc. In this protocol pipeline is not used. This protocol used as a bridge between the fast processor and slow peripheral devices. Latest version of Advanced Peripheral Bus is APB 2.0 and available at ARM official website.

1.8.1.2 AHB (Advanced High-performance Bus):

The Advanced High-performance Bus (AHB) is used with higher bandwidth devices like processor. For example it can be a fast memory or DMA. AHB have many features for example burst mode, protection, master lock, user privilege mode, split mode multi master support etc.

There is also simpler AHB version AHB-lite. This protocol can be found at ARM official website. The latest version is AHB 5.0. [17]

1.8.1.3 AXI (Advanced Extensible interface)

The Advanced Extensible interface (AXI) is most advance protocol which is more complex than AHB protocol. It is used for low latency and high bandwidth interconnects. AXI protocol doesn't share the bus like AHB, there is point to point interconnection between multiple masters in AXI. AXI also have many features like burst mode, separate read write, split mode etc.

a. AXI-lite protocol is a simplified version of AXI protocol and it will not support burst transfer of data.

b. AXI-stream protocol is another type of AXI protocol that supports only streaming of data from master to slave. In this protocol there is no separate read and write unlike a AXI-lite or full AXI. Multiple data streams can be transferred across mater and slave. And we can use interleaving method also for that.

The specification of full AXI and Lite AXI can be downloaded from ARM official website for free.

For AXI stream there is another website to download specification given below. [17]
http://infocenter.arm.com/help/topic/com.arm.doc.ihi0051a/index.html

1.8.1.4 ACE (AXI Coherence extension)

AXI Coherence extension protocol is an extension of AXI4 protocol. It is used in multicores which have cache also in same chip. In this protocol as it is extension of AXI the read and write of data channels by introducing separate snoop address, snoop response and snoop data channels. [17]
Specification of this protocol can also be found at ARM official website for free.

1.8.1.5 CHI (Coherent Hub Interface)

The ACE protocol is an extension to AXI to support coherent interconnects.
CHI protocol is introduced in AMBA version 5 as a complete redesign of AXI Coherence extension protocol for large number of coherent clusters on system on chip like server SoC designs. CHI uses packets as a message and implements protocol/physical/link layers based communication. This protocol is currently not freely available. [17]

1.9 Terminology

Defining some important terms.

1.9.1 Interleaving

Interleaving is way of increasing the system bandwidth by accessing the memory banks simultaneously. This will significantly reduce the latency of memory.

1.9.2 Burst

Burst is the sequence of similar or consecutive data transfer in a single go without any interrupt.

1.9.3 Protocol

Protocol is a set of rule defined for a design. This will increase the portability and reusability of that design.

1.10 Outline

Chapter -2 is literature review. This chapter is study of AMBA AHB based memory controller in which buffer (FIFO) used. In chapter-3 AMBA AHB protocol is discussed. In chapter-4 the memory controller is designed and implemented. Chpater-5 is simulation and results. Chapter-6 is conclusion and future work.

Chapter – 2

Literature Review

2.1 Objective

The objective for reviewing the literature is to find a method to improve the memory controller. Here it is tried to improve the performance of memory controller in term of area and latency and make the design more portable and reusable with different type of master and salve in vast area of system on chip. Constrains are area, speed/latency and power. All three constrains are in trade off situation. AHB protocol is most popular bus protocol nowadays used in system on chip. This protocol will make the design portable and reusable at different functional blocks. This will also reduce time to market because of already available verification IPs for the same.

2.2. Literature Review of memory controller

Ramagundam, Shashisekhar et al. "Design And Implementation Of High-Performance Master/Slave Memory Controller With Microcontroller Bus Architecture". ***2014 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings*** **(2014).**

The author of this paper says that memory controller is an intermediate device for a processor as a slave. It will help processor to communicate with slow memory. Memory controller will take commands from master and decode it in readable format which can be understood by slow memory. Nowadays these all systems are not separate chips or boards but these all are integrated on a single die which is called system on chip. In this system there can be many processor, memories, input output devices, communication

RF devices etc.

Shobha R Hadimani ,Panchami , (2015) " Verilog Based Design Of High Performance Data Access Amba Memory Controller " , *International Journal of Management and Applied Science (IJMAS).*

In this paper the author designed a memory controller based on FIFO. The memory controller is consisting of SRAM, ROM and CACHE. As microprocessor performance has improved in recent years, it has become important to provide a high-bandwidth, low-latency memory subsystem to achieve the full performance of these processors. The memory controller is part of system and it controls the memory data transfer. The aim is develop an architecture, design, and test AMBA AHB compliant Memory Controller for ARM based EMBEDDED platforms. Memory access time is reduced to a great extent by using AHB protocol thereby increasing the overall performance of the memory controller.

Rishabh Singh Kurmi, Shruti Bhargava and Ajay Somkuwar. Article:Design of AHB protocol block for Advanced Microcontrollers. *International Journal of Computer Applications* 32(8):23-29, October 2011.

The design of an AMBA advanced high performance bus (AHB) protocol IP block is presented in this paper. The AHB (Advanced High-performance Bus) is a high performance bus in AMBA (Advanced Microcontroller Bus Architecture) family. This AHB can be used in high clock frequency system modules. The AHB acts as the high performance system backbone bus. AHB supports the efficient connection of processors, on-chip memories and off-chip external memory interfaces with low-power peripheral macro cell functions. In this work, the design of the Advanced High Performance Bus Protocol is developed which has the basic blocks such as Master and Slave.

Nowadays systems are made in different functional blocks and they are integrated on a board. These blocks are called intellectual property (IP). These IPs need to connect using bus protocol which should not affect the functionality of IP. Now with standing memory device is slower than processor which will degrade the performance of processor. Day by day speed of processor is getting increase. But the improvement in memory is not matching with the performance of processor. Memory controller is responsible for communication between memory and processor. The challenge is interfacing memory controller with RAM in an efficient manner with less foot print in SoC and minimum latency in system. There is need of an efficient way for communication between memory and processor. The solution for this is to make an efficient memory controller which can also be used with wide range IPs. This will lead to make an AHB based memory controller. The FIFO based memory controller uses the buffer to store data and commands in FIFO and wait for raising edge of memory clock cycle to complete a transaction. This idle or wait is cause for system latency. [4]

There are many approaches are used to design memory controller for example Buffered memory controller and memory controller with interleaving concept.

Chapter – 3

AMBA AHB Protocol

3.1 Introduction

The Advanced microcontroller bus architecture is an open source bus protocol used to interconnect and manage multiple functional blocks in system on chip. It is most popular protocol used in system on chip and application specific designs nowadays. In system on chip there are large number of functional blocks like processor, memories and peripherals. These all functional blocks should meet required constrain like area and speed. Solution is to use a good connectivity protocol like ARM AHB. AMBA AHB protocol is designed for high bandwidth synthesizable system. [5]

AMBA is introduced in 1996 by ARM. The first AMBA bus was advanced system bus and advance peripheral bus. After that in 1999 the second version came, AMBA version 2 which have high performance bus which is a single clock edge protocol. After this in 2003 ARM developed AMBA version 3 which have AXI and ATB bus protocol which have even more bandwidth than AHB. In 2010 AMBA version 4 came. In 2013 AMBA version 5 came. These protocols are nowadays used for wide range of 32 bit system. We can find all these protocol available at ARM official website for free. [5]

3.2 AHB signals [5]

In AMBA, all signals have naming convention to make it more readable and distinguish between other AMBA protocols. This will also help while reading and analyzing the wave form of simulation results for input and output. In naming convention each signal have a letter for that protocol. For example

AHB: H
ASB: B
APB: P

A lower case letter n in the signal name indicates that the signal is active low signal otherwise signal name is always in uppercase letters.
Test signals have a prefix T regardless of the bus type.

H prefix indicates that it is AHB signal. For example: HBURST, HSIZE, HTRANS etc. These signals are active high signals. HRESETn is active low signal because it includes the lower case n letter in signal name.

The following signals are AHB signals shown in table 3.1.

Name	Source	Description
HCLK (Bus clock)	Clock source	This clock times all bus transfers. All signal timings are related to the rising edge of HCLK.
HRESETn (Reset)	Reset controller	The bus reset signal is active LOW and is used to reset the system and the bus. This is the only

		active LOW signal.
HADDR[31:0] (Address bus)	Master	The 32-bit system address bus.
HTRANS[1:0] (Transfer type)	Master	Indicates the type of the current transfer, which can be NONSEQUENTIAL, SEQUENTIAL, IDLE or BUSY.
HWRITE (Transfer direction)	Master	When HIGH this signal indicates a write transfer and when LOW a read transfer.
HSIZE[2:0] (Transfer size)	Master	Indicates the size of the transfer, which is typically byte (8-bit), halfword (16-bit) or word (32-bit). The protocol allows for larger transfer sizes up to a maximum of 1024 bits.
HBURST[2:0] (Burst type)	Master	Indicates if the transfer forms part of a burst. Four, eight and sixteen beat bursts are supported and the burst may be either incrementing or wrapping.
HPROT[3:0] (Protection control)	Master	The protection control signals provide additional information about a bus access and are primarily intended for use by any module that wishes to implement some level of protection. The signals indicate if the transfer is an opcode fetch or data access, as well as if the transfer is a privileged mode access or user mode access. For bus masters with a memory management unit these signals also indicate whether the current access is cacheable or bufferable.

HWDATA[31:0] (Write data bus)	Master	The write data bus is used to transfer data from the master to the bus slaves during write operations. A minimum data bus width of 32 bits is recommended. However, this may easily be extended to allow for higher bandwidth operation.
HSELx (Slave select)	Decoder	Each AHB slave has its own slave select signal and this signal indicates that the current transfer is intended for the selected slave. This signal is simply a combinatorial decode of the address bus.
HRDATA[31:0] (Read data bus)	Slave	The read data bus is used to transfer data from bus slaves to the bus master during read operations. A minimum data bus width of 32 bits is recommended. However, this may easily be extended to allow for higher bandwidth operation.
HREADY (Transfer done)	Slave	When HIGH the HREADY signal indicates that a transfer has finished on the bus. This signal may be driven LOW to extend a transfer. Note: Slaves on the bus require HREADY as both an input and an output signal.
HRESP[1:0] (Transfer response)	Slave	The transfer response provides additional information on the status of a transfer. Four different responses are provided, OKAY, ERROR, RETRY and SPLIT.

Table 3.1: AHB signals [5]

AHB also supports multiple bus masters for that extra signals are require for communication between these masters. There are many arbitration signals which are point to point. Suffix x indicates that this signal is related to master-x.

There will be a number of HBUSREQx signals in a system, for example HBUSREm1, HBUSREQm2 and HBUSREQm3 related to master 1, master 2, master 3 respectively. These signals are shown in table 3.2.

Name	Source	Description
HBUSREQx (Bus request)	Master	A signal from bus master x to the bus arbiter which indicates that the bus master requires the bus. There is an HBUSREQx signal for each bus master in the system, up to a maximum of 16 bus masters.
HLOCKx (Locked transfers)	Master	When HIGH this signal indicates that the master requires locked access to the bus and no other master should be granted the bus until this signal is LOW
HGRANTx (Bus grant)	Arbiter	This signal indicates that bus master x is currently the highest priority master. Ownership of the address/control signals changes at the end of a transfer when HREADY is HIGH, so a master gets access to the bus when both HREADY and HGRANTx are HIGH.
HMASTER[3:0] (Master number)	Arbiter	These signals from the arbiter indicate which bus master is currently performing a transfer and is used by the slaves which support SPLIT transfers to determine which master is attempting an access.

		The timing of HMASTER is aligned with the timing of the address and control signals.
HMASTLOCK (Locked sequence)	Arbiter	Indicates that the current master is performing a locked sequence of transfers. This signal has the same timing as the HMASTER signal.
HSPLITx[15:0] (Split completion request)	Slave	This 16-bit split bus is used by a slave to indicate to the arbiter which bus masters should be allowed to re-attempt a split transaction. Each bit of this split bus corresponds to a single bus master.

Table 3.2: AHB signals for multi master [5]

AHB is a new generation bus protocol used to work with high speed processor as well as with slow memory also for communication. For slow device we use AMBA APB protocol over the AMBA AHB protocol. This will work as bridge between higher bandwidth devices and low bandwidth devices. There are lots of features of AMBA AHB which support for higher frequency system.

Features AHB supports

- Burst transfers
- Split transactions
- Single cycle bus master handover
- Single clock edge operation
- Non-tristate implementation
- Wider data bus configurations (64/128 bits).

3.3 A typical AMBA AHB-based microcontroller [5]

A typical AMBA AHB based microcontroller system is shown in figure. Which consist of high bandwidth memory interface, high frequency ARM processor, on chip RAM, DMA bus master, a bridge between slow memory and high speed blocks, UART, Keypad, Timer, external RAM and ROM. The AMBA based system is shown in figure 3.1.

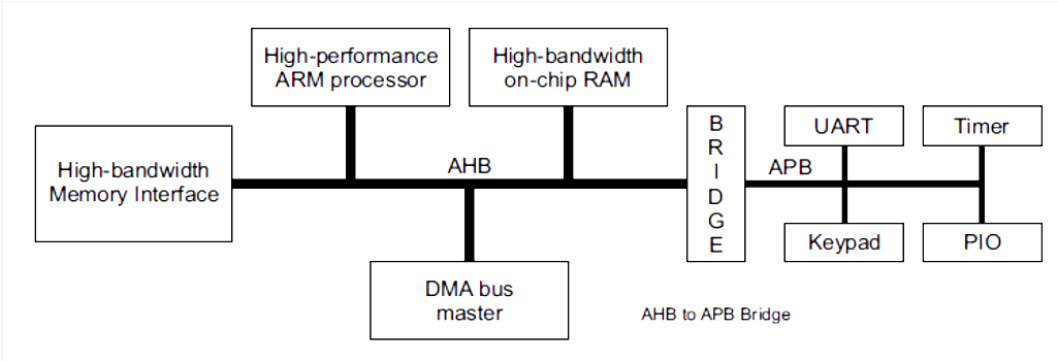

Figure 3.1: AMBA AHB based system [5]

Bus in AMBA AHB there is multiple master are connected with slaves via multiplexer, arbiter and decoder. Multiplexer is used to connect multiple masters with multiple slaves. Arbiter will decide which master should send the address and control signal which indicates the transfer they want to perform.

Master should assert bus request before send address and command signal. If bus is available then arbiter will grant the bus to that master for transfer. Otherwise master will wait for bus grant signal. In AMBA based system buses are multiplexed for multi-masters. These multiplexers are shown in figure 3.2.

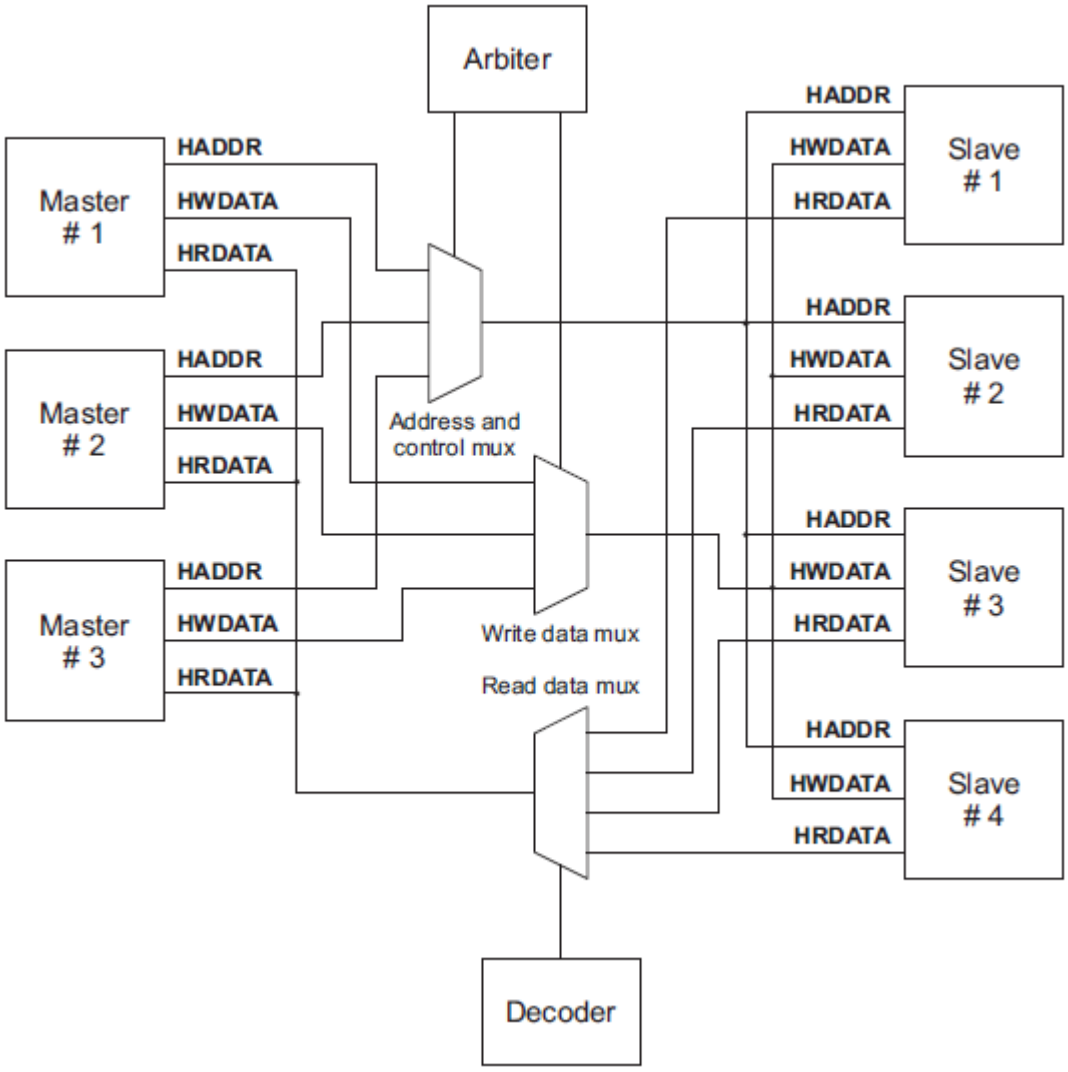

Figure 3.2: Multiplexed Signals in AMBA AHB system [5]

A decoder in system is used to controller the multiplexer for read. Control signal and address provides the information about which slave should receive the data. These signals also have information about size of transfer, type of burst etc.

There are two data bus one write data bus transfer the data from master to slave and another read data bus read data from slave and transfer it to master.

Every transfer consists of:

- An address and control signals.

- One more cycles for the data.

The address cannot be extended so all slave should sample the address at every clock cycle. We can extent data cycle by HREADY signal. When HREADY is low then this is a wait state allow slave extra time to sample the data in next clock cycle.

3.4 Basic transfer [5]

In AHB the master will start transaction after bus is granted to it. Master will first send the address and then data in second cycle or after wait if HREADY is low. So there are two sections one is address phase and another is data phase.

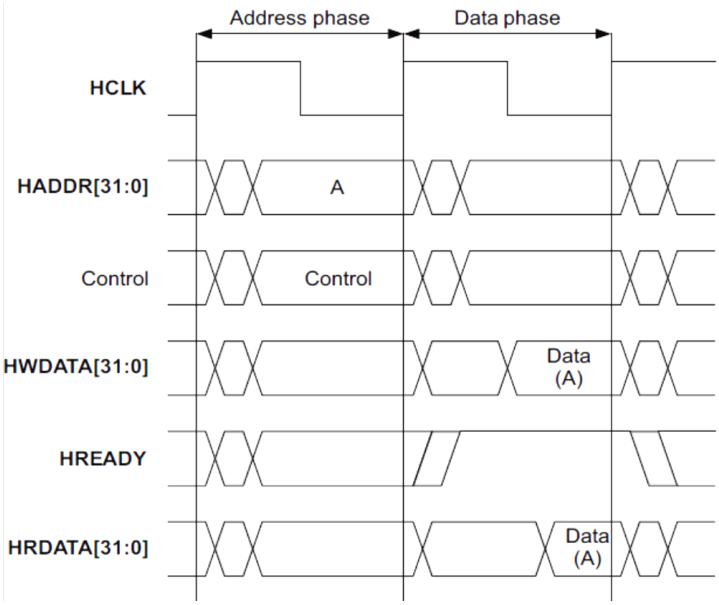

Figure 3.3: Basic transfer in AHB [5]

The simple transfer without wait state is shown in figure 3.3

- Address and control signals are send by master after raising edge of HCLK.
- The slave sample these signal at next raising edge of HCLK.
- After sampling the address and control signal salve will generate the response according to those singles and send it to HRESP bus.
- If slave is busy it will send HREADY low to master to extent the data phase.

3.5 Transfer with wait state [5]

In AHB if slave is with other task then it will send the HREADY low. This is the indication for master to wait for one cycle. Master will extend its transfer with same address and control signals. A transfer with wait state is shown in figure 3.4.

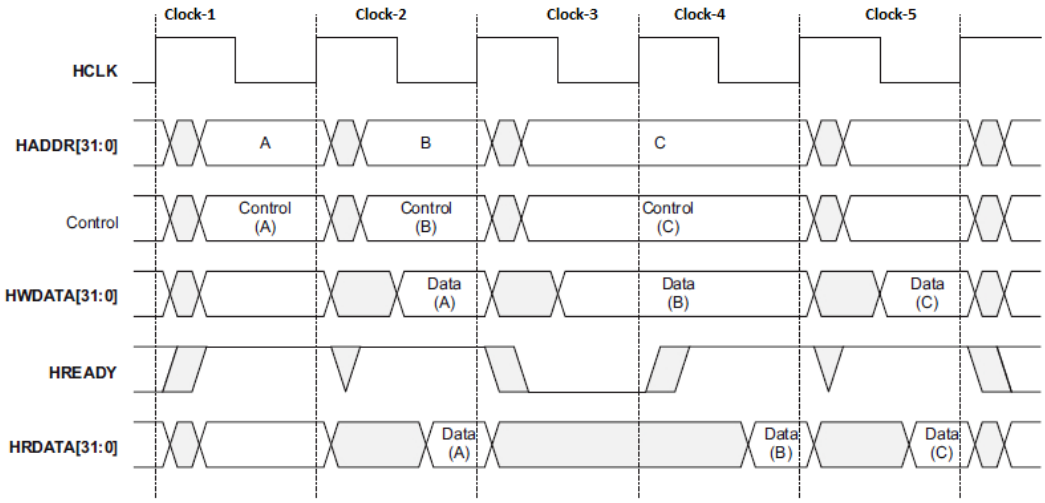

Figure 3.4 Transfer with wait state [5]

- First address A is send by master a raising edge of HCLK. In next clock slave will sample the address and control signals. If it is a read transfer then data will be read by master in this cycle (clock-1) otherwise data will be written in slave.

- After that address B will be sampled by slave with control signals. Slave will sample the address but there is busy signal asserted by master so slave will assert HREADY low indicating master to extend its transfer for next clock (clock-4) also.

- So data will be provided in next clock (clock-5) after HREADY is high.

- Address C will be sampled by slave in next clock cycle (clock-4) and further next cycle (clock-5) data will be sampled by salve without any wait state.

3.6 Control Signals [5]

3.6.1 Transfer Type

In AHB protocol transfer sends with transfer type which will be sampled by slave for indication of burst-finish or wait state or idle state. There are four types of transfer. The description of these signals is given in table 3.3.

HTRANS[1:0]	Type	Description
2'b00	IDLE	Indicates that no data transfer is required. The IDLE transfer type is used when a bus master is granted the bus, but does not wish to perform a data transfer. Slaves must always provide a zero wait state OKAY response to IDLE transfers and the transfer should be ignored by the slave.
2'b01	BUSY	The BUSY transfer type allows bus masters to insert IDLE cycles in the middle of bursts of transfers. This transfer type indicates that the bus master is continuing with a burst of transfers, but the next transfer cannot take place immediately. When a master uses the BUSY transfer type the address and control signals must reflect the next transfer in the burst. The transfer should be ignored by the slave. Slaves must always provide a zero wait state OKAY response, in the same way that they respond to IDLE transfers.
2'b10	NONSEQ	Indicates the first transfer of a burst or a single transfer. The address and control signals are unrelated to the previous transfer. Single transfers on the bus are treated as bursts of one and therefore the transfer type is NONSEQUENTIAL.
2'b11	SEQ	The remaining transfers in a burst are SEQUENTIAL and the address is related to the previous transfer. The control information is identical to the previous transfer. The address is equal to the address of the previous transfer plus the size (in bytes). In the case of a wrapping burst the address of the transfer wraps at the address boundary equal to the size (in bytes) multiplied by the number of beats in the transfer either 4, 8

Table 3.3: Transfer signals in AHB [5]

3.6.2 Transfer Size

Transfer size indicates the size of transfer or significant data on data bus. This is 3 bit long. Detail of these signals is given in table 3.4.

HSIZE[2:0]	Size	Description
3'b000	8 bits	Byte
3'b001	16 bits	Half-word
3'b010	32 bits	Word
3'b011	64 bits	-
3'b100	128 bits	4-word line
3'b101	256 bits	8-word line
3'b110	512 bits	-
3'b111	1024 bits	-

Table 3.4: Transfer size in AHB [5]

3.6.3 Burst signals

In burst transfer data will be sent in a continuous burst. This signal is 3 bit long. In table 3.5 it is shown the burst type.

Wrapping burst: In wrapping type burst the address will wrap at a particular address when boundary reached.

Incremental burst: In incremental burst will finish after number of beats completed. The address of all beats is sequential.

HBURST[2:0]	Type	Description
3'b000	SINGLE	Single transfer
3'b001	INCR	Incrementing burst of unspecified length
3'b010	WRAP4	4-beat wrapping burst
3'b011	INCR4	4-beat incrementing burst
3'b100	WRAP8	8-beat wrapping burst
3'b101	INCR8	8-beat incrementing burst
3'b110	WRAP16	16-beat wrapping burst
3'b111	INCR16	16-beat incrementing burst

Table 3.5: Burst type in AHB [5]

3.6.4 Response Type

During a transfer the slave sends the response signals to indicate success or failure of transfer. This is 2 bit long signal. Description of these signals is given in table 3.6.

This signal name is HRESP [1:0].

HRESP[1:0]	Response	Description
2'b00	OKAY	When HREADY is HIGH this shows the transfer has completed successfully. The OKAY response is also used for any additional cycles that are inserted, with HREADY LOW, prior to giving one of the three other responses.
2'b01	ERROR	This response shows an error has occurred. The error condition should be signaled to the bus master so that it is aware the transfer has been unsuccessful. A two-cycle response is required for an error condition.
2'b10	RETRY	The RETRY response shows the transfer has not yet completed, so the bus master should retry the transfer. The master should continue to retry the transfer until it completes. A two-cycle RETRY response is required.
2'b11	SPLIT	The transfer has not yet completed successfully. The bus master must retry the transfer when it is next granted access to the bus. The slave will request access to the bus on behalf of the master when the transfer can complete. A two-cycle SPLIT response is required.

Table 3.6: Response signals in AHB [5]

Chapter - 4

Design and implementation of Memory controller

4.1 Introduction

Memory controller is small digital device which controls the data transfer between two systems. This can be separate digital chip or can be embedded with other circuit on same chip. If a memory controller is on same die then it is called integrated memory controller. The reason to use on same die is to reduce the data transfer latency between master and salve. A master can be a typical processor and a salve can be a memory for example RAM or ROM. In computers the memory controller is at North Bridge of mother board. This is now replaced with integrated memory controller used with CPU to reduce memory latency. But this also make it memory depended controller which need to redesign the controller for new technology memory. [14]

Memory controller generally handles the read and write of data with a memory which can be an internal memory or external memory. The speed of memory can different from speed of processor which will make it difficult to design. DRAM controller need to refresh memory in every clock cycle which will allow processor to focus on important task. Otherwise processor will spend its all time with accessing and refreshing the memory. To access the memory we need to select memory chip according to address and control signals. Then memory will decode it to rows and columns for particular byte or depends on width of data bus. There can be a memory response for telling that data is written successfully or there is an error while writing

data to that address for example if it is ROM address then data cannot be altered or there can be an address which is not connected to any memory device.

4.2 Concept of interleaving

Main memory is typically divided in series of memory banks. The number of banks is in the range of power of 2 in the most of system. Access a data from flat memory is requires time equal to the memory bank access time. If we access the memory banks simultaneously then bandwidth can be increase by number of memory bank and speed of accessing single data from one bank. If there is K number of banks then K request can be send simultaneously to memory bank. Here K is 2^n. [18]

For example if there is two banks in memory. If interleaving is not used then we first access the data from first address and then we send second address to memory. This will introduce the latency in system. If we use interleaving then data can be accessed simultaneously from bank while another bank is preparing data. This will increase the bandwidth of system and reduce the latency. Addresses are continuously asserted with raising edge of system clock to memory banks so each bank will receive the address at different clock and provide the data after memory clock cycle.

Comparing with old simpler memory controller used in past where data are sending one by one. This is a bottleneck condition sometime because master works at higher frequency than the memory device. But in AHB based memory controller there is feature of burst mode which will send contiguous data in a single burst. AHB also have error handling features, split mode, protection mode, bus lock etc. Combining AHB with interleaving will significantly improve the performance of system.

However if there is 32 bit for addresses then we can access 4 GB of memory location which is a large amount of memory. In past, when CPU only supports less than 4 GB of

location then we need to send a single back to CPU for invalid address. This is handled as error for AHB master.

Let's assume that our CPU can only access 8K of RAM. The amount of memory is quite small, but makes it easy to illustrate the concept of interleaved memory.

In table 4.1 Memory addresses are distributed in a non-interleaved system.

Bank-0	Bank-1	Bank-2	Bank-3	Bank-4	Bank-5	Bank-6	Bank-7
0	1024	2048	3072	4096	5120	6144	7168
1	1025	2049	3073	4097	5121	6145	7169
…	…	…	…	…	…	…	…
1023	2047	3071	4095	5119	6143	7167	8192

Table 4.1: Memory address distribution in non interleaved memory

To find the bank for an address

If address is A and number of bank is N, then bank number:-

$$\text{Bank Number} = A / 1024$$

Flat memory distribution: In this memory address are continuous in a memory bank.

The higher order address is used for chip select of memory banks. The higher order memory distribution is shown in figure 4.1.

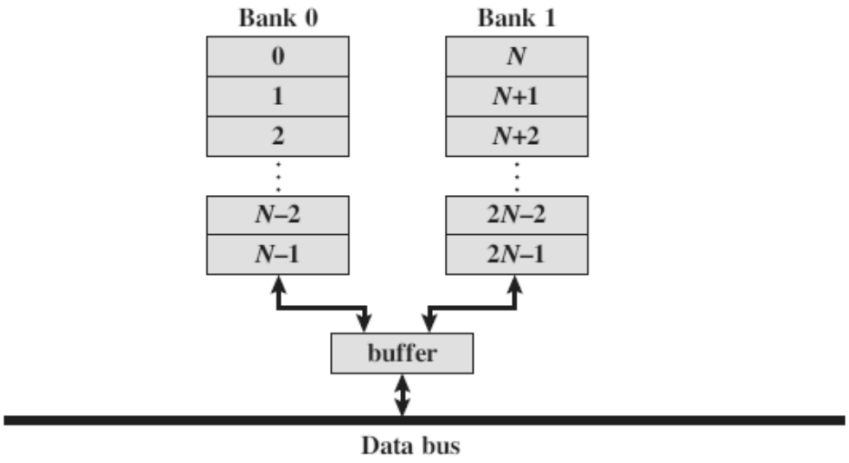

Figure 4.1: Memory bank distribution in flat memory [18]

Interleaved memory distribution: In this memory address is alternative in both memory banks. This is a two way memory interleaving.

Memory address divided in interleaved system is shown in table 4.2.

Bank-0	Bank-1	Bank-2	Bank-3	Bank-4	Bank-5	Bank-6	Bank-7
0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
…	…	…	…	…	…	…	…
8185	8186	8187	8188	8189	8190	8191	8192

Table 4.2: Memory address distribution in interleaved memory

We divided this 8K memory in 8 banks. The address is accessible from A[0] to A[8192].

If address is A and number of bank is N, then bank number:-

Bank Number = A % N

For example address 123 will be in bank 123 % 8 = Bank-3.

The lower order memory address bits are used for chip select logic of memory banks. It is shown in figure 4.2.

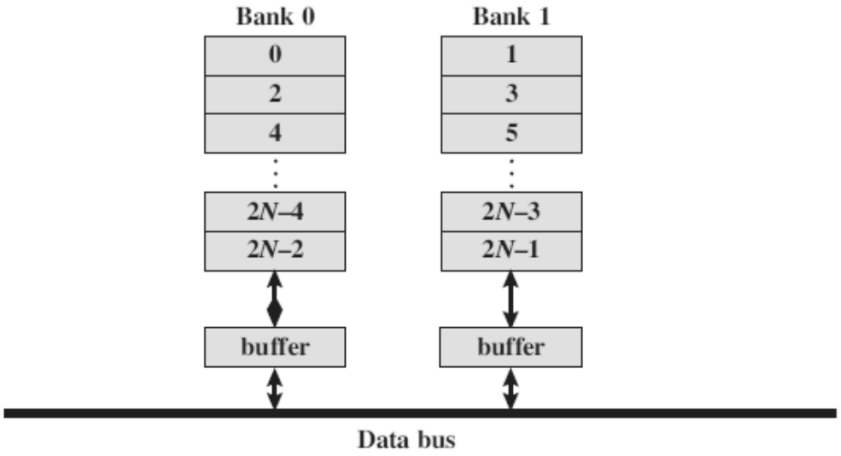

Figure 4.2: Memory bank distribution in interleaved memory [18]

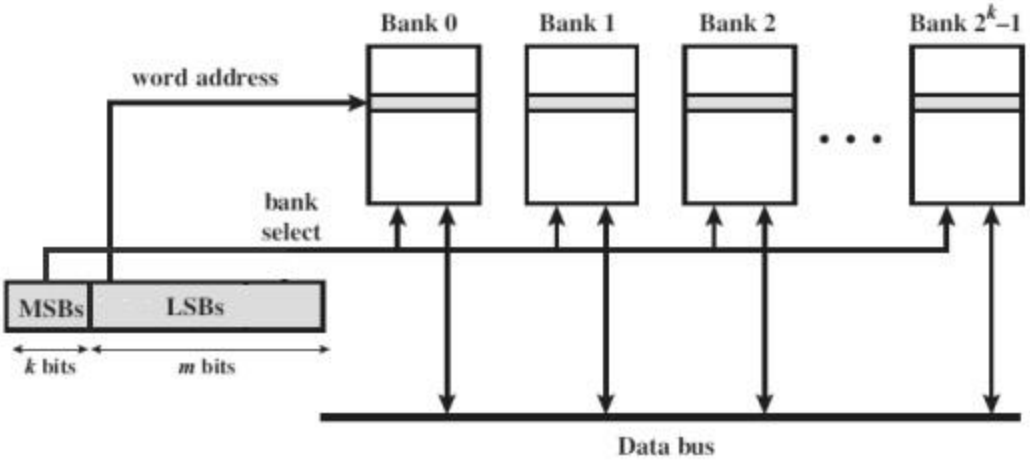

Figure 4.3: High order address decoding in flat memory distribution [18]

In figure 4.3 a higher order memory distribution is shown. In this memory distribution higher memory address bits are used for chip select of memory banks.

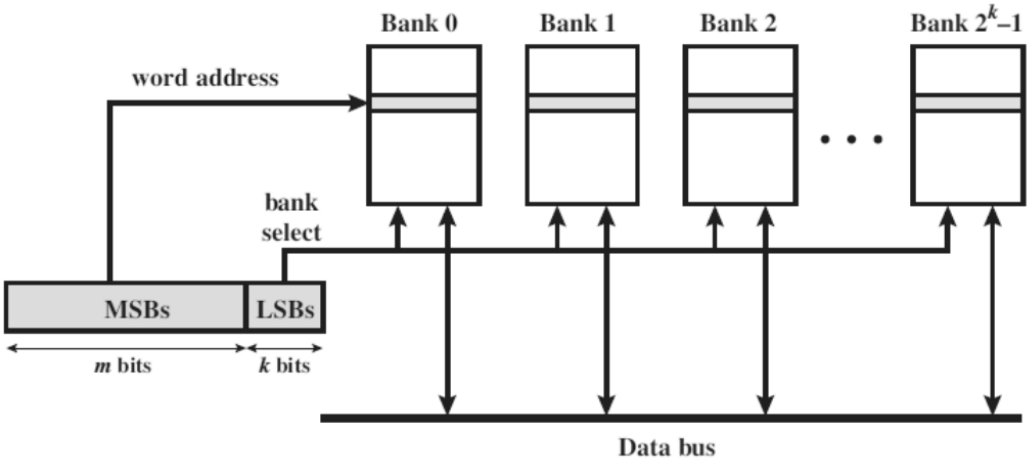

Figure 4.4: Lower order address decoding in interleaved memory distribution [18]

In figure 4.4 a lower order memory distribution is shown. In this memory distribution lower address bits are used for chip select. This is called interleaved memory.

In interleaving the chip select logic is decoded using LSB of address bus. So that alternative address is spread over different memory bank. Even addresses are in memory bank-0 and odd addresses are in memory bank-1. There can be more number of memory banks like 4, 8 or 16. The memory latency can be hide by using interleaving where time to access is less than the one clock cycle of system clock multiple by number of banks.

4.3 Finite state machine of memory controller

Finite state machine for memory controller based on AHB and concept of interleaving is implemented. FSM is the heart of memory controller. It will change the state according to data, address and control signal send by master and memory. After receiving these all signal it will generate appropriate responses.

There are total six states in memory controller. Concept of interleaving introduced with AHB bus protocol. AHB master will send a transaction to memory controller to process and provide required information and data. Memory controller will change its state according to that transaction. If there is no transaction then memory controller will be in idle state and sends okay response. The Finite state machine of memory controller is shown in figure 4.5.

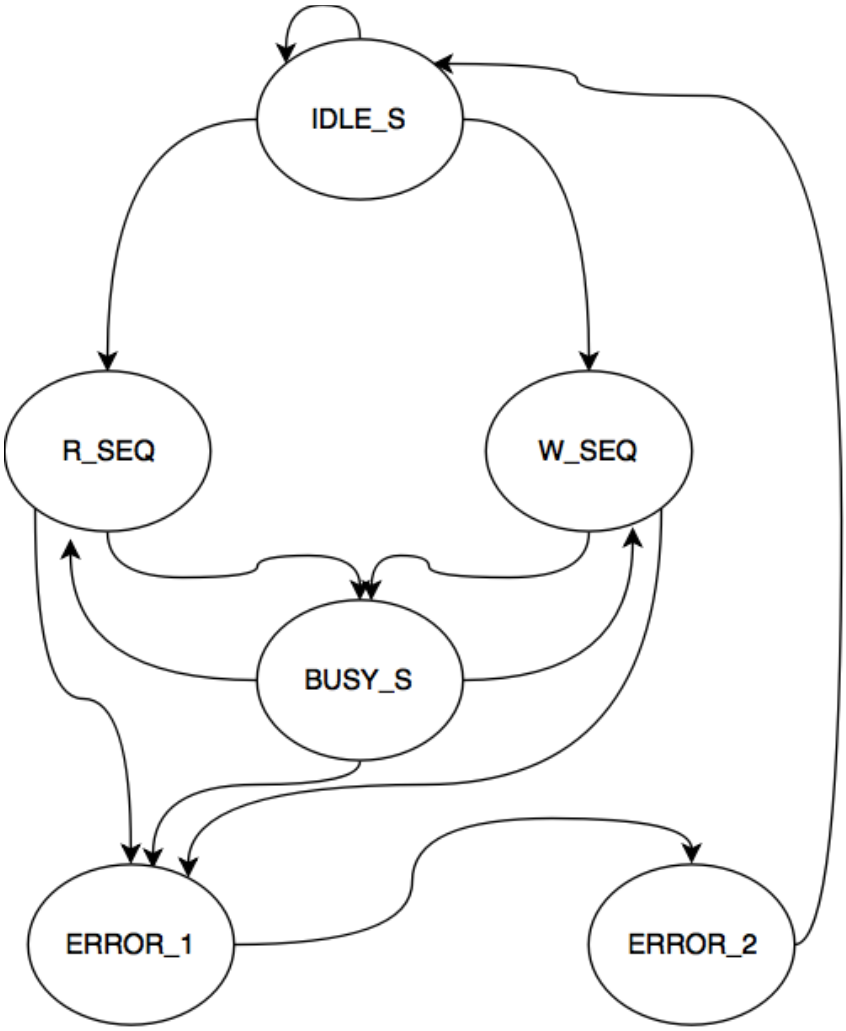

Figure 4.5: Finite state machine for memory controller

IDEL_S state: In this state memory controller will not transfer any data. Memory controller will send OKAY response to master in this state. It's will change its state according to control signals. If there is a NONSEQ or SEQ transfer then memory controller will go to next state in R_SEQ or W_SEQ according to HWRITE signal. Otherwise it will be in same state.

W_SEQ state: In this state memory controller will write the data in memory. It will also generate control signals for memory in this state. If master is busy in between a

burst transfer then master will send BUSY signal on HTRANS bus. So memory controller will change its state to IDEL_S. If there is any mismatch with upcoming address and raising edge of corresponding memory bank then memory controller will change its state to BUSY_S. If there is an address which is not available in memory then memory controller will change its state to Error_1. Otherwise it will be in the same state.

R_SEQ state: In this state memory controller will read the data from memory. It will also generate control signals for memory in this state. If master is busy in between a burst transfer then master will send BUSY signal on HTRANS bus. So memory controller will change its state to IDEL_S. If there is any mismatch with upcoming address and raising edge of corresponding memory bank then memory controller will change its state to BUSY_S. If there is an address which is not available in memory then memory controller will change its state to Error_1. Otherwise it will be in the same state.

BUSY_S state: In this state memory controller will send HREADY low. This indicates that memory controller is in busy state and it cannot transfer the data. If there is an address which is not available in memory then memory controller will change its state to Error_1.

ERROR_1 state: In this state memory controller will sent HREADY signal low. And it will also send HRESP signal with ERROR. This will be inferred by master that there is an error in transfer. Master will reinitiate its transfer. It will change its state to Error_2. Because in AHB a slave will send two clock cycle response if there is an error.

ERROR_2 state: In this state if there is a SEQ or NONSEQ transfer then it will change it change its state to R_SEQ or W_SEQ according to HWRITE signal. If there is error then it will change its state to Error_1 state. Otherwise it will go in IDEL_S state.

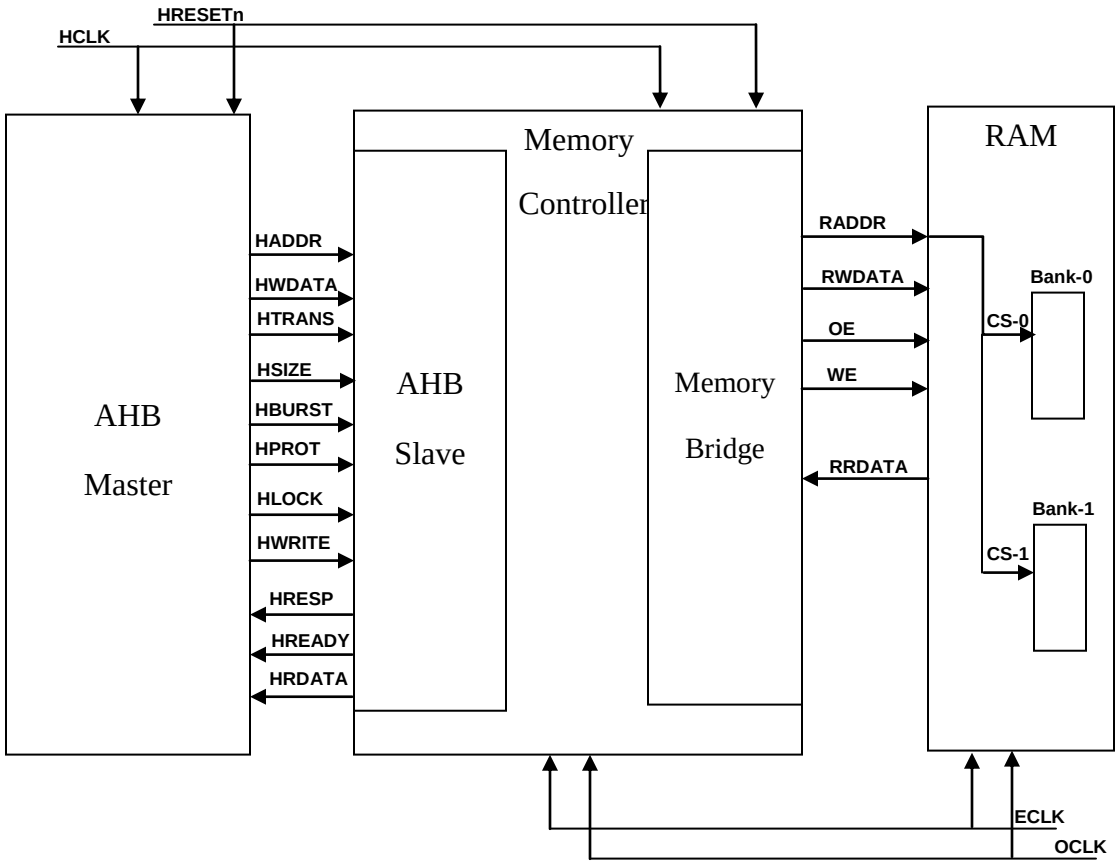

Figure 4.6: Block diagram of AHB based memory controller

In figure 4.6 block diagram of memory controller is shown. Memory Bridge will convert the AHB signals to memory readable format. This will also introduce the interleaving. In read and write state, if sampled address is synced with clock of RAM then there is no need of inserting HREADY low otherwise need to insert HREADY low which will send memory controller in BUSY_S state. For example if address is even and clock cycle for even clock is low then we can't send data in next clock because of that we need to insert one wait state. For that master will extend its transfer to next clock. If it is an odd address then next clock is in favor of transfer so memory controller will go into R_SEQ or W_SEQ according to HWRITE signal. After first clock, it can

continuously read and write for single burst but if data it in single transfer then there can be wait state. This will introduce the concept of interleaving.

The specification of RAM is used from IBM SRAM Document. In which reading from memory will be done by sending appropriate control signals to RAM. To read a data from memory we need to first select correct memory bank. The address, write enable and output enable should be provided before clock edge. The data will be sample by memory controller after next raising edge of memory clock. In figure 4.7 a memory read cycle is shown.

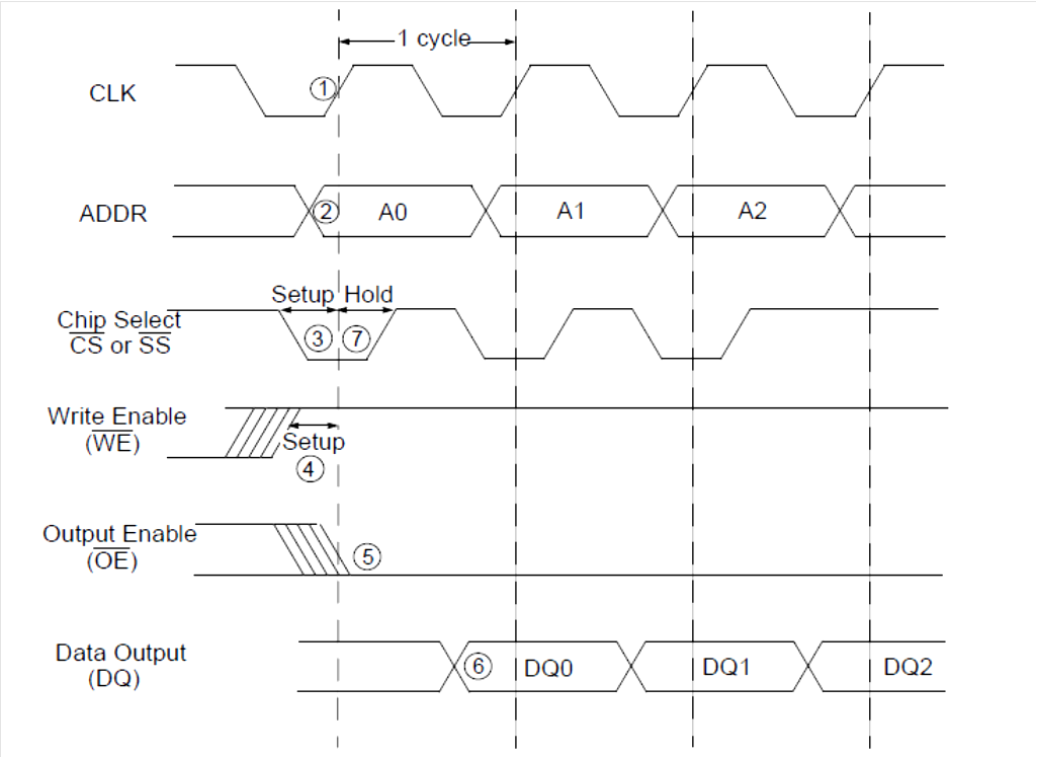

Figure 4.7: Memory Read cycle [19]

In write cycle, we need to first select correct memory bank according to sampled memory address. Data, address and control signals will be sampled by memory at raising edge of memory clock. Memory needs at least one memory clock cycle to properly write the data in memory bank. We cannot send another data and control signals in between the memory cycle. The output enable signal should be high because it is an active low signal. The write enable signal should be low because it is an active low signal. In figure 4.8 a memory write cycle is shown.

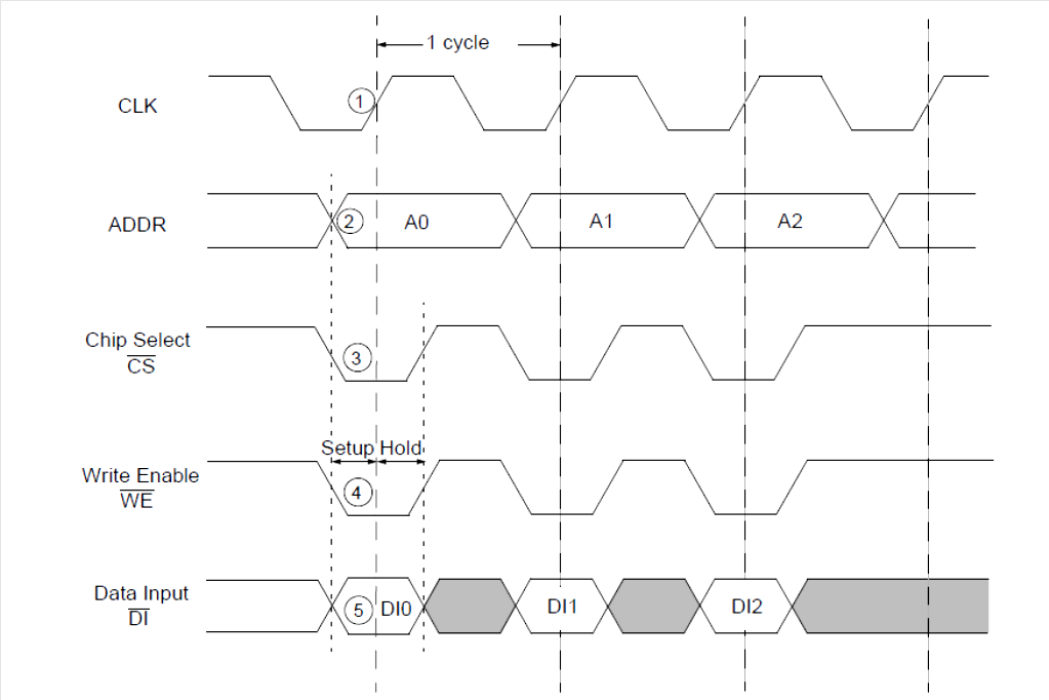

Figure 4.8 Memory Write cycle [19]

Chapter - 5

Simulation and Synthesis Report

5.1 Simulation and Wave forms

The simulation of AHB based memory controller is done in Xilinx ISE software (ISE v 12.1 April 19, 2010 student version) which is freely available for students. The Xilinx version used for simulation and synthesis is. This Xilinx tool is used to synthesis and analysis of timing performance of RTL design written in Verilog or VHDL. There is ISIM simulator used for waveform analysis.

Write transaction wave forms (simulation results):

In figure 5.1 the first NONSEQ burst started with address 0x14(green color). Slave will sample the address and data after one clock cycle (Blue color). Data value 0x48 will be written in memory. The address 0x14 is related to memory bank-1. So it will be sampled by memory at raising edge of clke.

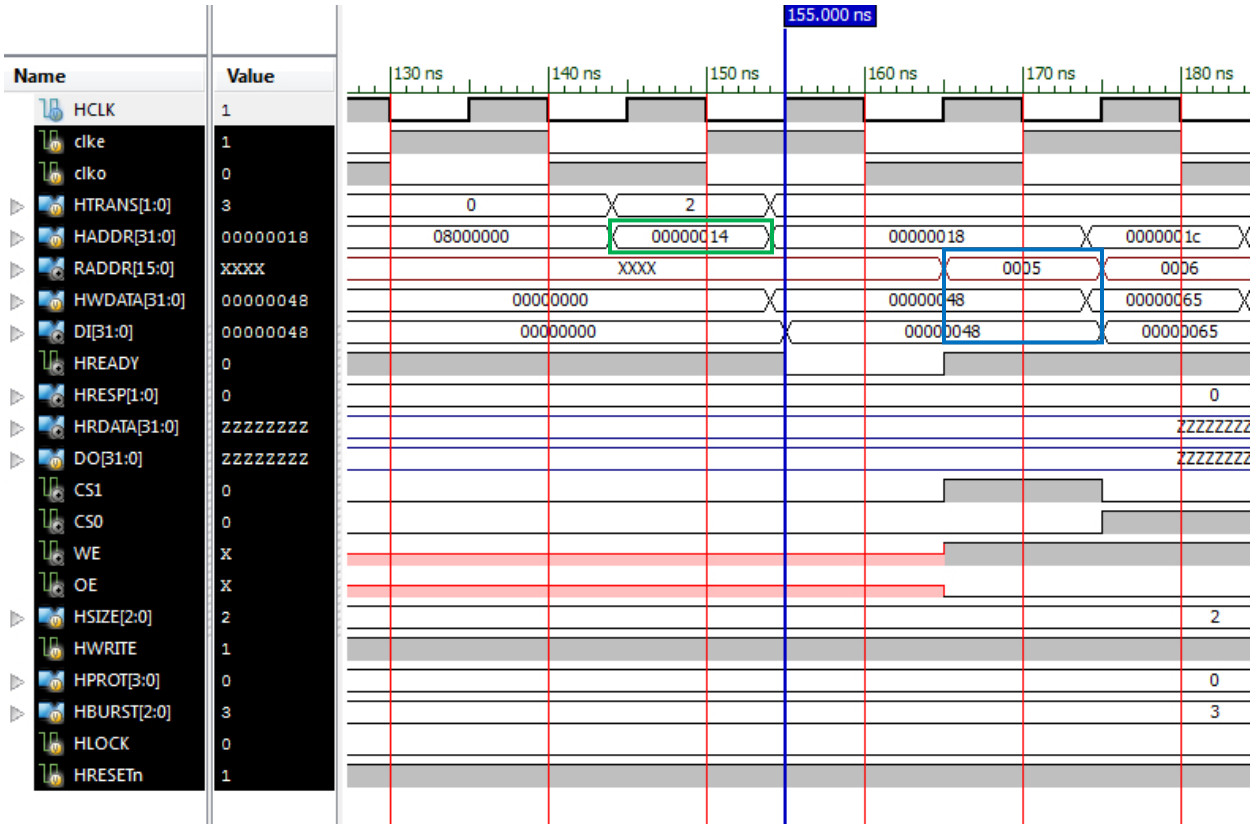

Figure 5.1: Write transaction wave forms – 1(2-way MC)

In figure 5.2 SEQ-burst is sending continuous addresses (green color). Slave will sample the address 0x20 and 0x24 at 185ns and 195ns. Data will be written in memory at next cycle (Blue color).

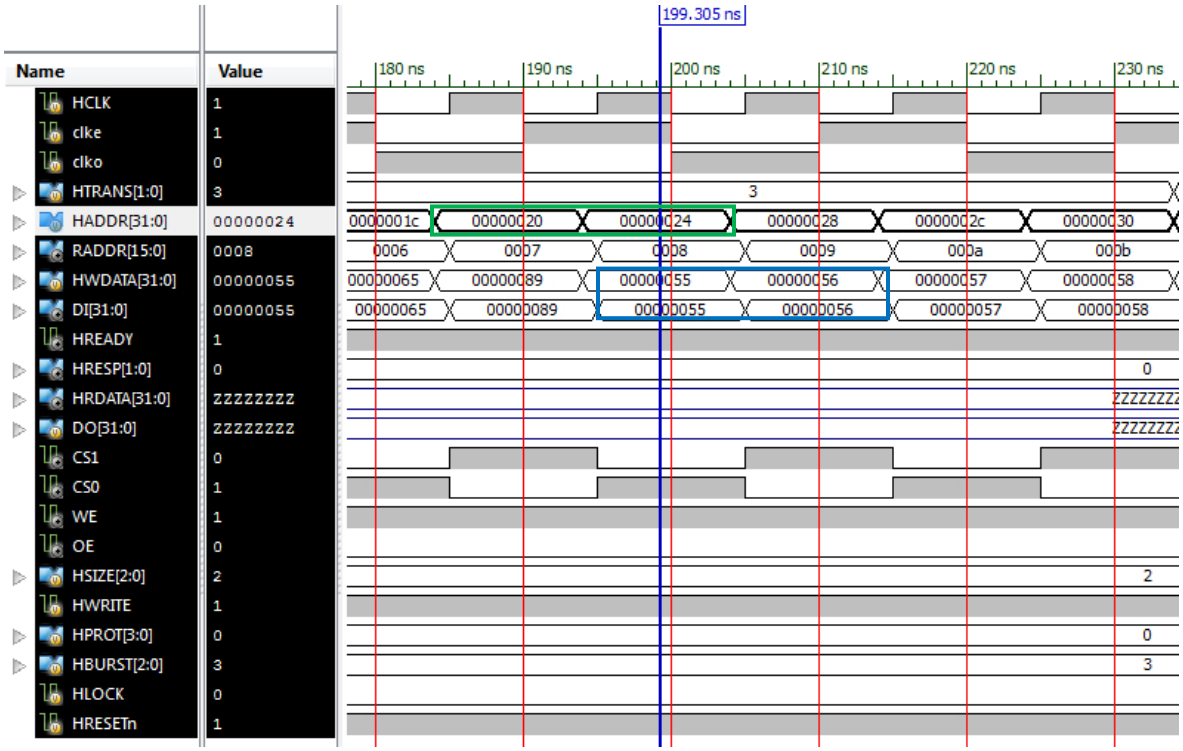

Figure 5.2: Write transaction wave forms – 2 (2-way MC)

In figure 5.3 there is an IDLE cycle (Red color). In IDLE cycle chip select is no activated (Green color).

Figure 5.3: Write transaction wave forms – 3 (2-way MC)

In figure 5.4, data write cycle is shown in 4-way memory controller. (Red Color) Data will be written in memory at 180 ns. The address 0x14 is related to memory bank-1. So it will be sampled by memory at raising edge of clk1. This is shown in yellow color.

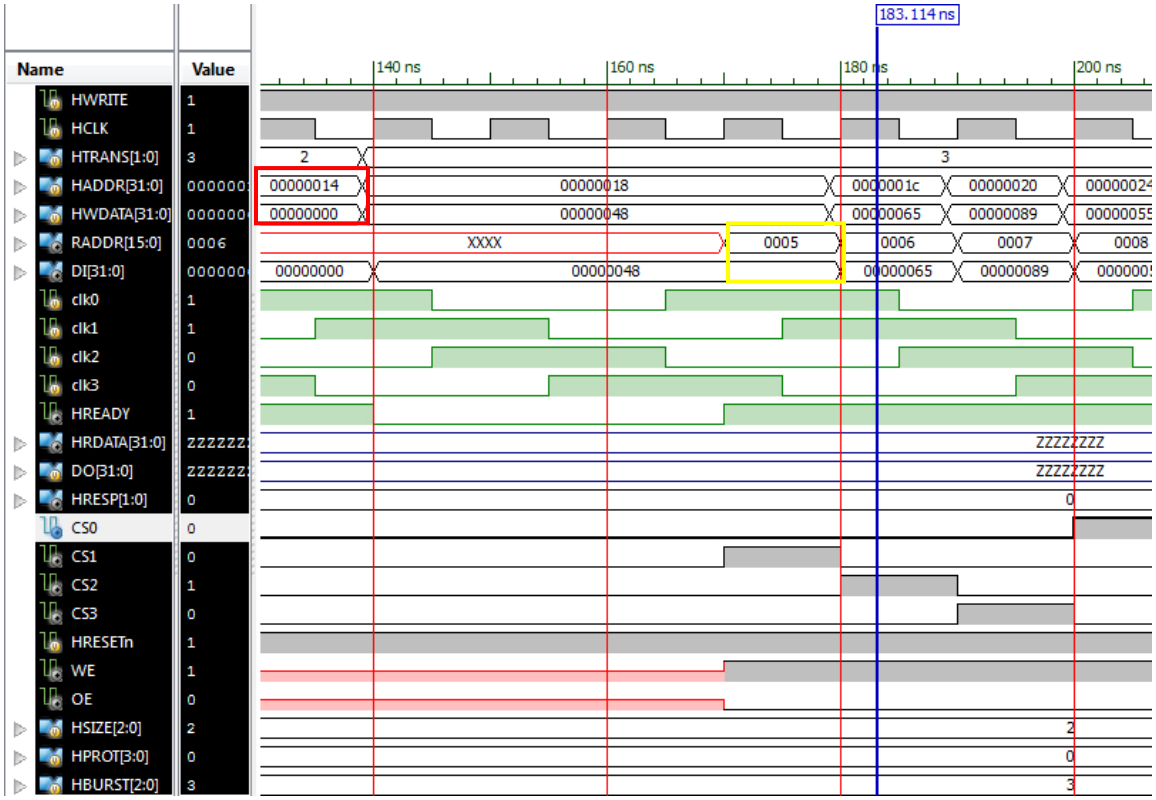

Figure 5.4: Write transaction wave forms – 5 (4-way MC)

Read transaction wave forms (simulation results):

In figure 5.5 Read burst is started with address 0x14. Data will be sampled by master at 555 ns (Green color). The addresses are 0x14 and 0x18 and corresponding data are 0x48 and 0x65 read from memory.

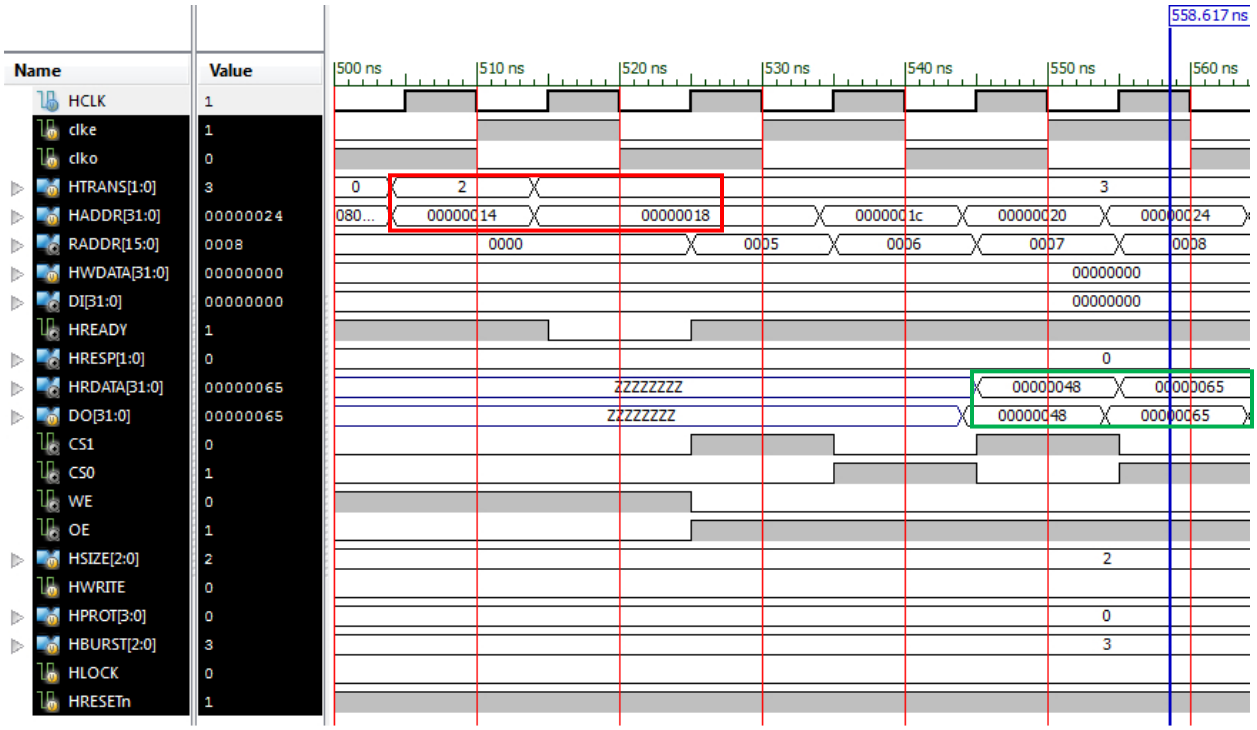

Figure 5.5: Read transaction wave forms – 1 (2-way MC)

In figure 5.6 read burst is shown. (Red color) Sequentially data are read from memory in 4-way memory controller. (Blue color). The first address 0x14 is sampled at 560 ns by memory controller. Then next three more addresses are received at 570ns, 580ns and 590ns. Corresponding data are read at 610ns, 620ns, 630ns and 640ns respectively.

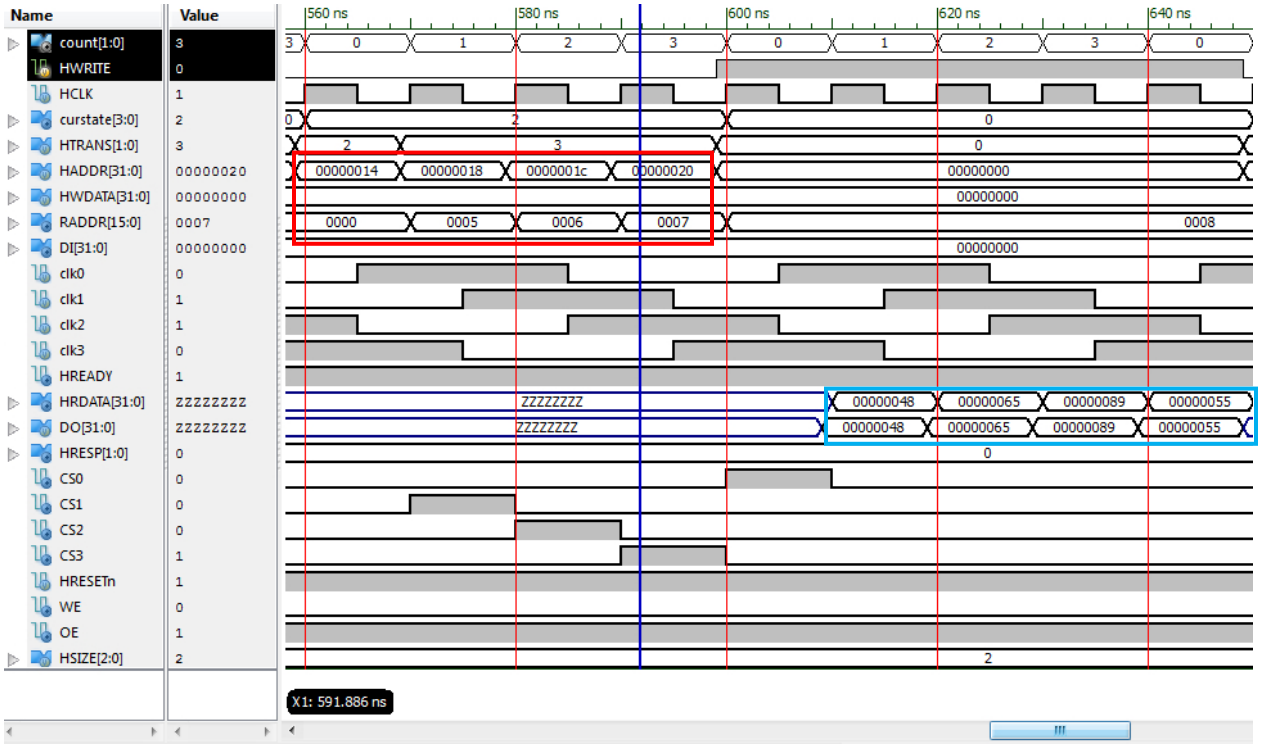

Figure 5.6: Read transaction wave forms – 2 (4-way MC)

5.2 Synthesis Report

Xilinx standard synthesis report for RTL design shown in table 5.1 and table 5.2.

Project File:	mem.xise	**Parser Errors:**	No Errors
Module Name:	memc	**Implementation State:**	Synthesized
Target Device:	xc3s1000-4fg320	**Errors:**	No Errors
Product Version:	ISE 12.1	**Warnings:**	
Design Goal:	Balanced	**Routing Results:**	
Design Strategy:	Xilinx Default	**Timing Constraints:**	
Environment:	System Settings	**Final Timing Score:**	

Table 5.1: Project status

Synthesis summary of Memory controller is given in table 5.2. The target device used for this is Spartan-3 (3s50pq208-5). This device is also used in FIFO based memory controller. But this device is not suitable for this design because numbers of available IOBs are less then number of used IOBs. So target device used for this design is xc3s1000-4fg320.

Logic Utilization	Used	Available	Utilization
Number of Slice Registers	86	768	11%
Number of Slice LUTs	122	1536	7%
Number of fully used LUT-FF pairs	156	1536	10%
Number of bonded IOBs	**191**	**124**	**154%**
Number of BUFG/BUFGCTRLs	1	8	12%

Table 5.2: Synthesis summary of 2-way MC (Device: 3s50pq208-5)

Device xc3s1000-4-fg320 have 221 bonded IOBs. This is enough for this design.

Logic Utilization	Used	Available	Utilization
Number of Slices	86	7680	1%
Number of Slice Flip Flops	122	15360	0%
Number of 4 input LUTs	156	15360	1%
Number of bonded IOBs	191	221	86%
Number of GCLKs	1	8	12%

Table 5.3: Synthesis summary of 2-way MC (Device: xc3s1000-4-fg320)

Logic Utilization	Used	Available	Utilization
Number of Slices	136	7680	1%
Number of Slice Flip Flops	165	15360	1%
Number of 4 input LUTs	247	15360	1%
Number of bonded IOBs	197	221	89%
Number of GCLKs	2	8	25%

Table 5.4: Synthesis summary of 4-way MC (Device: xc3s1000-4-fg320)

5.2.1 Detailed Synthesis Report of 2 – Way Memory controller

```
Release 12.1 - xst M.53d (nt64)

=========================================================================
*                            Final Report                               *
=========================================================================
Final Results
RTL Top Level Output File Name     : memc.ngr
Top Level Output File Name         : memc
Output Format                      : NGC
Optimization Goal                  : Speed
Keep Hierarchy                     : NO
Design Statistics
# IOs                              : 201
Cell Usage :
# BELS                             : 164
#      GND                         : 1
#      INV                         : 3
#      LUT2                        : 67
#      LUT3                        : 37
#      LUT3_D                      : 1
#      LUT3_L                      : 16
#      LUT4                        : 30
#      LUT4_L                      : 2
#      MUXCY                       : 5
#      MUXF5                       : 1
#      VCC                         : 1
# FlipFlops/Latches                : 154
#      FD                          : 84
#      FDC                         : 68
#      FDS                         : 2
# Clock Buffers                    : 1
#      BUFGP                       : 1
# IO Buffers                       : 190
#      IBUF                        : 103
#      OBUF                        : 87
```

```
==========================================================================
Timing Summary:
---------------
Speed Grade: -4

   Minimum period: 5.570ns (Maximum Frequency: 179.520MHz)
   Minimum input arrival time before clock: 7.571ns
   Maximum output required time after clock: 8.337ns
   Maximum combinational path delay: No path found

Timing Detail:
--------------
All values displayed in nanoseconds (ns)

==========================================================================
Timing constraint: Default period analysis for Clock 'HCLK'
  Clock period: 5.570ns (frequency: 179.520MHz)
  Total number of paths / destination ports: 286 / 75
--------------------------------------------------------------------------
Delay:               5.570ns (Levels of Logic = 3)
  Source:            hready_rnm0 (FF)
  Destination:       curstate_FSM_FFd3 (FF)
  Source Clock:      HCLK rising
  Destination Clock: HCLK rising

  Data Path: hready_rnm0 to curstate_FSM_FFd3
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDS:C->Q             53   0.720   2.041  hready_rnm0 (hready_rnm0)
     LUT3:I2->O            1   0.551   0.827  curstate_FSM_FFd3-In20
(curstate_FSM_FFd3-In20)
     LUT4_L:I3->LO         1   0.551   0.126  curstate_FSM_FFd3-In28
(curstate_FSM_FFd3-In28)
     LUT4:I3->O            1   0.551   0.000  curstate_FSM_FFd3-In95
(curstate_FSM_FFd3-In)
     FDC:D                     0.203          curstate_FSM_FFd3
    ----------------------------------------
    Total                      5.570ns (2.576ns logic, 2.994ns route)
                                       (46.2% logic, 53.8% route)

==========================================================================
Timing constraint: Default OFFSET IN BEFORE for Clock 'HCLK'
  Total number of paths / destination ports: 243 / 132
--------------------------------------------------------------------------
Offset:              7.571ns (Levels of Logic = 5)
  Source:            HTRANS<1> (PAD)
  Destination:       curstate_FSM_FFd1 (FF)
  Destination Clock: HCLK rising

  Data Path: HTRANS<1> to curstate_FSM_FFd1
```

```
                                  Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O             68   0.821   2.398  HTRANS_1_IBUF
(HTRANS_1_IBUF)
     LUT4:I0->O             1   0.551   0.000  curstate_FSM_FFd1-In461
(curstate_FSM_FFd1-In461)
     MUXF5:I0->O            1   0.360   1.140  curstate_FSM_FFd1-In46_f5
(curstate_FSM_FFd1-In46)
     LUT2:I0->O             1   0.551   0.996  curstate_FSM_FFd1-In63_SW0
(N50)
     LUT4:I1->O             1   0.551   0.000  curstate_FSM_FFd1-In75
(curstate_FSM_FFd1-In)
     FDC:D                      0.203          curstate_FSM_FFd1
    ----------------------------------------
    Total                       7.571ns (3.037ns logic, 4.534ns route)
                                        (40.1% logic, 59.9% route)

=========================================================================
Timing constraint: Default OFFSET OUT AFTER for Clock 'HCLK'
  Total number of paths / destination ports: 86 / 86
-------------------------------------------------------------------------
Offset:              8.337ns (Levels of Logic = 1)
  Source:            hready_rnm0 (FF)
  Destination:       HREADY (PAD)
  Source Clock:      HCLK rising

  Data Path: hready_rnm0 to HREADY
                                  Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDS:C->Q              53   0.720   1.973  hready_rnm0 (hready_rnm0)
     OBUF:I->O                  5.644          HREADY_OBUF (HREADY)
    ----------------------------------------
    Total                       8.337ns (6.364ns logic, 1.973ns route)
                                        (76.3% logic, 23.7% route)

=========================================================================
Total REAL time to Xst completion: 7.00 secs
Total CPU time to Xst completion: 6.45 secs
-->
Total memory usage is 244956 kilobytes
```

5.2.2 Detailed Synthesis Report of 4 – Way Memory controller

```
Release 12.1 - xst M.53d (nt64)

=========================================================================
*                            Final Report                               *
=========================================================================
```

```
Final Results
RTL Top Level Output File Name     : memc.ngr
Top Level Output File Name         : memc
Output Format                      : NGC
Optimization Goal                  : Speed
Keep Hierarchy                     : NO

Design Statistics
# IOs                              : 211

Cell Usage :
# BELS                             : 271
#      GND                         : 1
#      INV                         : 4
#      LUT2                        : 104
#      LUT2_L                      : 1
#      LUT3                        : 98
#      LUT3_L                      : 2
#      LUT4                        : 25
#      LUT4_D                      : 4
#      LUT4_L                      : 9
#      MUXCY                       : 5
#      MUXF5                       : 17
#      VCC                         : 1
# FlipFlops/Latches                : 197
#      FD                          : 84
#      FDC                         : 104
#      FDR                         : 9
# Clock Buffers                    : 2
#      BUFGP                       : 2
# IO Buffers                       : 195
#      IBUF                        : 100
#      OBUF                        : 95
=========================================================================

Timing Summary:
---------------
Speed Grade: -4

   Minimum period: 5.848ns (Maximum Frequency: 170.999MHz)
   Minimum input arrival time before clock: 7.550ns
   Maximum output required time after clock: 8.223ns
   Maximum combinational path delay: No path found

Timing Detail:
--------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default period analysis for Clock 'HCLK'
  Clock period: 5.848ns (frequency: 170.999MHz)
```

```
  Total number of paths / destination ports: 476 / 120
--------------------------------------------------------------------------------
Delay:               5.848ns (Levels of Logic = 3)
  Source:            curstate_0 (FF)
  Destination:       curstate_1 (FF)
  Source Clock:      HCLK rising
  Destination Clock: HCLK rising

  Data Path: curstate_0 to curstate_1
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDC:C->Q             16   0.720   1.576  curstate_0 (curstate_0)
     LUT3:I0->O            1   0.551   0.869  curstate_mux0003<1>51 (N8)
     LUT4:I2->O            1   0.551   0.827  curstate_mux0003<0>21_SW1
(N30)
     LUT4:I3->O            1   0.551   0.000  curstate_mux0003<2>1
(curstate_mux0003<2>)
     FDC:D                     0.203          curstate_1
    ----------------------------------------
    Total                      5.848ns (2.576ns logic, 3.272ns route)
                                       (44.0% logic, 56.0% route)

================================================================================

Offset:              5.470ns (Levels of Logic = 2)
  Source:            HRESETn (PAD)
  Destination:       st_FSM_FFd2 (FF)
  Destination Clock: clk0 rising

  Data Path: HRESETn to st_FSM_FFd2
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O             1   0.821   0.801  HRESETn_IBUF (HRESETn_IBUF)
     INV:I->O            105   0.551   2.271  HRESETn_inv1_INV_0
(HRESETn_inv)
     FDR:R                     1.026          st_FSM_FFd2
    ----------------------------------------
    Total                      5.470ns (2.398ns logic, 3.072ns route)
                                       (43.8% logic, 56.2% route)

================================================================================
Timing constraint: Default OFFSET OUT AFTER for Clock 'HCLK'
  Total number of paths / destination ports: 94 / 94
--------------------------------------------------------------------------------
Offset:              8.223ns (Levels of Logic = 1)
  Source:            hready_rnm0 (FF)
  Destination:       HREADY (PAD)
  Source Clock:      HCLK rising
```

```
  Data Path: hready_rnm0 to HREADY
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDR:C->Q             33   0.720   1.859  hready_rnm0 (hready_rnm0)
     OBUF:I->O                 5.644          HREADY_OBUF (HREADY)
    ----------------------------------------
    Total                      8.223ns (6.364ns logic, 1.859ns route)
                                       (77.4% logic, 22.6% route)

==========================================================================


Total REAL time to Xst completion: 7.00 secs
Total CPU time to Xst completion: 6.98 secs

-->

Total memory usage is 247452 kilobytes
```

5.3 Gate level schematics

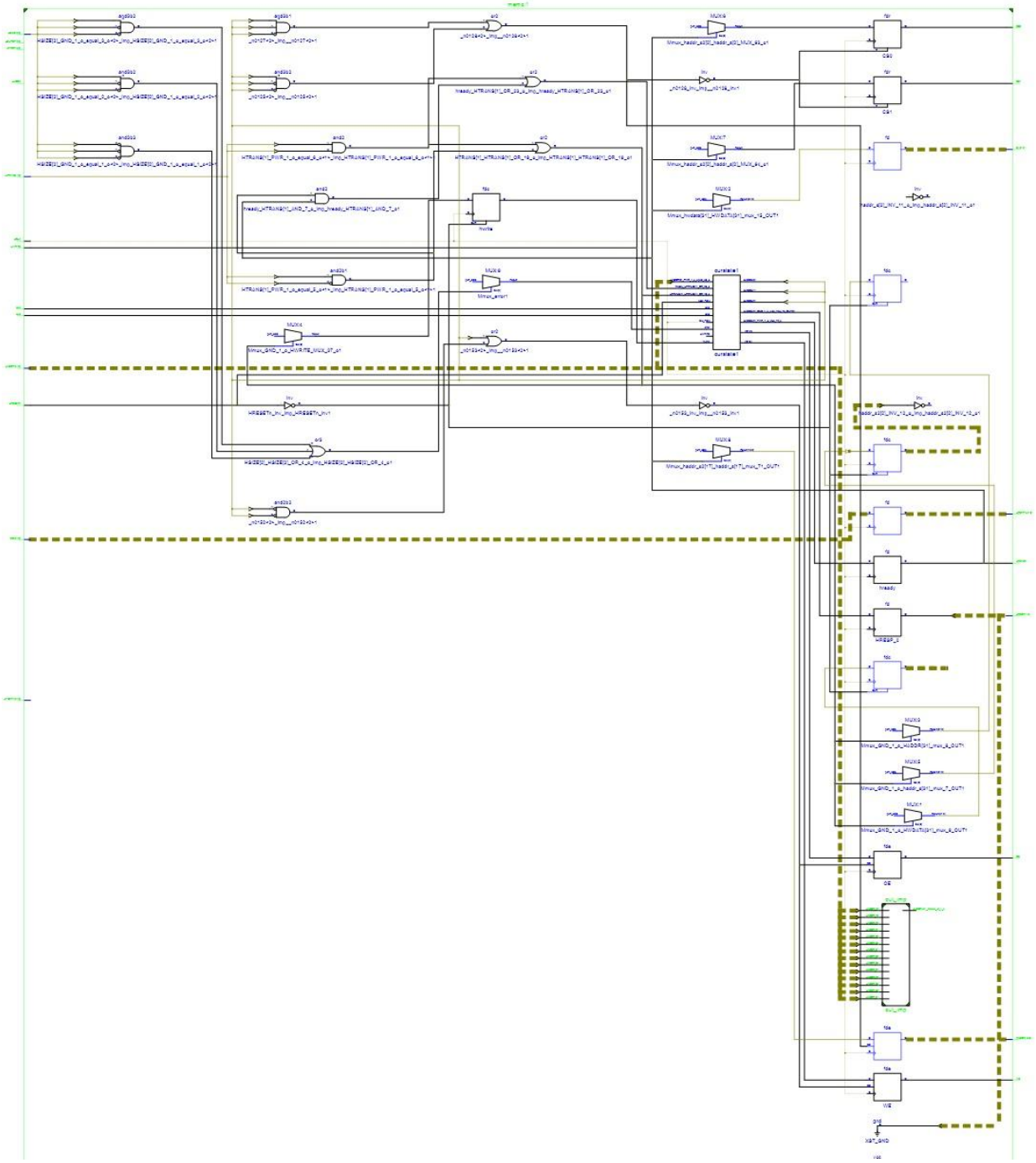

Figure 5.7: Gate level schematics for 2-way MC.

Chapter – 6

Conclusion and future work

6.1 Conclusion

The memory controller is successfully designed. AMBA AHB bus protocol is used for this design. This protocol is most popular protocol used nowadays in the industry. Previously used FIFO based memory controller has latency. But introducing concept of interleaving will reduce the latency in system and increase the throughput. This will significantly improve the performance of memory controller. The wave form for read and write show that memory controller is working according to its specification. The critical path delay can be found in synthesis report generated from Xilinx design tool ISE. Number of LUTs used in design can also found in synthesis report which will be indication for area used for design on a die.

Logic Utilization	Available			Used			Utilization		
	FIFO Based MC	Interleaved MC 2-Way	Interleaved MC 4-Way	FIFO Based MC	Interleaved MC 2-Way	Interleaved MC 4-Way	FIFO Based MC	Interleaved MC 2-Way	Interleaved MC 4-Way
Number of Slice Registers	768	7680	7680	160	86	136	20%	1%	1%
Number of Slice LUTs	1536	15360	15360	111	122	165	7%	0%	1%
Number of fully used LUT-FF pairs	1536	15360	15360	273	156	247	17%	1%	1%
Number of bonded IOBs	124	221	221	79	191	197	63%	86%	89%
Number of BUFG/BUFGCTRLs	8	8	8	3	1	2	37%	12%	25%

Table 6.1: Comparison between results of different memory controllers

In FIFO based AHB memory controller initially FIFO is empty after transfer begins master will start filling data and commands in FIFO. After some time FIFO will be full. Master will wait for FIFO to become empty. This will introduce bubble or stall in system. [1] After comparing with FIFO based AHB memory controller it has been seen that there is need of extra chip area to accommodate FIFO circuit in SoC. This will increase area of die. And the maximum frequency for FIFO based memory controller is 155.67MHz (Critical path delay is 6.425ns). [1] Now comparing with 2-way Interleaved AHB based memory controller, the maximum frequency is 179.520 MHz and critical path delay is 5.570 ns can be referred in synthesis report in chapter 5. Comparing with 4-way interleaved AHB based memory controller the maximum frequency of 4-way memory controller is 170.999 MHz and critical path delay is 5.848 ns. This is significantly improvement in terms of speed as compared to FIFO based memory controller.

6.2 Future work

The memory controller can be made n-way interleaved by increasing number of banks in memory. We can also make parameterized memory controller. Instant of using AHB protocol we can use AXI bus protocol which can increase more bandwidth for system. Memory control can also be made independent of types of memory.

Bibliography

[1] Ramagundam, Shashisekhar et al. "Design And Implementation Of High-Performance Master/Slave Memory Controller With Microcontroller Bus Architecture". *2014 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings* (2014): n. pag. Web. 27 June 2017.

[2] Shobha R Hadimani ,Panchami , (2015) " Verilog Based Design Of High Performance Data Access Amba Memory Controller " , *International Journal of Management and Applied Science (IJMAS)* , pp. 88-91, Volume-2, Issue-5

[3] Rishabh Singh Kurmi, Shruti Bhargava and Ajay Somkuwar. Article:Design of AHB protocol block for Advanced Microcontrollers. *International Journal of Computer Applications* 32(8):23-29, October 2011. Full text available. BibTeX

[4] C. Sharma, Archana. "Construct High-Speed SDRAM Memory Controller Using Multiple FIFO's For AHB Memory Slave Interface". *International Journal of Emerging Technology and Advanced Engineering* volume 3.Issue 3 (2013): 1-10. Web. 28 June 2017.

[5] https://soc.eecs.yuntech.edu.tw/Course/SoC/doc/amba.pdf

[6] "Read-Only Memory". *En.wikipedia.org*. N.p., 2017. Web. 28 June 2017.

[7] Khalifa, Khaled et al. "Memory Controller Architectures: A Comparative Study". *2013 8th IEEE Design and Test Symposium* (2013): n. pag. Web. 11 June 2017.

[8] Rao, Shilpa. "Testing Of AMBA Compliant Memorycontroller Using Pattern Generator/ Logicanalyser". *Rroij.com*. N.p., 2017. Web. 26 June 2017.

[9] "What Is Hard Disk Drive (HDD)? - Definition From Whatis.Com". *SearchStorage*. N.p., 2017. Web. 26 June 2017.

[10] "What Is SSD (Solid-State Drive)? - Definition From Whatis.Com". *SearchSolidStateStorage*. N.p., 2017. Web. 26 June 2017.

[11] "Memory (RAM) And Its Influence On Performance". *Computermemoryupgrade.net*. N.p., 2017. Web. 27 June 2017.

[12] "Static Random-Access Memory". *En.wikipedia.org*. N.p., 2017. Web. 28 June 2017.

[13] "Read Write Processor Bus Cycles". *Eventhelix.com*. N.p., 2017. Web. 28 June 2017.

[14] "Memory Controller". *En.wikipedia.org*. N.p., 2017. Web. 28 June 2017.

[15] "ARM | Corelink DMC-520 Dynamic Memory Controller". *ARM | The Architecture for the Digital World*. N.p., 2017. Web. 28 June 2017.

[16] "Advanced Microcontroller Bus Architecture". *En.wikipedia.org*. N.p., 2017. Web. 28 June 2017.

[17] "AMBA Open Specifications - ARM". *Arm.com*. N.p., 2017. Web. 28 June 2017.

[18] https://www.cs.vassar.edu/~jones/Stallings/E-InterleavedMemory.pdf

[19] https://www.ece.cmu.edu/~ece548/localcpy/sramop.pdf

[20] En.wikipedia.org. (2017). *Memory hierarchy*. [online] Available at: https://en.wikipedia.org/wiki/Memory_hierarchy [Accessed 29 Jun. 2017].

[21] SUN, GUANGYU. *Exploring Memory Hierarchy Design With Emerging Memory Technologies*. [Place of publication not identified]: SPRINGER INTERNATIONAL PU, 2016. Print.

Design and Implementation of AMBA-AHB based memory controller

ORIGINALITY REPORT

%16	%16	%6	%10
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	opencores.org Internet Source	%2
2	www.ukessays.com Internet Source	%1
3	anysilicon.com Internet Source	%1
4	ensino.univates.br Internet Source	%1
5	www.mikrocontroller.net Internet Source	%1
6	Submitted to Liverpool John Moores University Student Paper	%1
7	access.ee.ntu.edu.tw Internet Source	%1
8	web.cecs.pdx.edu Internet Source	%1
9	blog.swanspace.org	

	Internet Source	%1
10	www.ijesmjournal.com Internet Source	%1
11	Submitted to CSU, Fresno Student Paper	<%1
12	www.xilinx.com Internet Source	<%1
13	documents.mx Internet Source	<%1
14	Submitted to BITS, Pilani-Dubai Student Paper	<%1
15	Submitted to Visvesvaraya Technological University Student Paper	<%1
16	wiki.scinethpc.ca Internet Source	<%1
17	share.pdfonline.com Internet Source	<%1
18	Submitted to University of Birmingham Student Paper	<%1
19	vlsi1.engr.utk.edu Internet Source	<%1
	en.wikipedia.org	

20	Internet Source	<%1
21	innovexpo.itee.uq.edu.au Internet Source	<%1
22	www.dtic.mil Internet Source	<%1
23	www.kanecomputing.co.uk Internet Source	<%1
24	dspace.thapar.edu:8080 Internet Source	<%1
25	csl.anthropomatik.kit.edu Internet Source	<%1
26	excamera.com Internet Source	<%1
27	S PASRICHA. "On-Chip Communication Architecture Standards", On-Chip Communication Architectures, 2008 Publication	<%1
28	Divekar, Shraddha, and Archana Tiwari. "Interconnect matrix for multichannel AMBA AHB with multiple arbitration technique", 2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE), 2014. Publication	<%1

29	diva-portal.org Internet Source	<%1
30	www.ijetch.org Internet Source	<%1
31	alexfreed.com Internet Source	<%1
32	www.gregwhaley.com Internet Source	<%1
33	univagora.ro Internet Source	<%1
34	www.ijsps.com Internet Source	<%1
35	l3.elfak.ni.ac.rs Internet Source	<%1
36	docplayer.net Internet Source	<%1
37	wiki.musl-libc.org Internet Source	<%1
38	Divekar, Shraddha, and Archana Tiwari. "Multichannel AMBA AHB with multiple arbitration technique", 2014 International Conference on Communication and Signal Processing, 2014. Publication	<%1

39	www.gaisler.com Internet Source	<%1
40	digital.lib.usf.edu Internet Source	<%1
41	www.slideshare.net Internet Source	<%1
42	Sungju Park. "Design of Test Access Mechanism for AMBA-Based System-on-a-Chip", 25th IEEE VLSI Test Symmposium (VTS 07), 05/2007 Publication	<%1
43	www-artist.imag.fr Internet Source	<%1
44	Vani, R.M., P.V. Hunagund, and M. Roopa. "UART controller as AMBA APB slave", National Conference on Challenges in Research & Technology in the Coming Decades (CRT 2013), 2013. Publication	<%1
45	developer.apple.com Internet Source	<%1
46	www.scribd.com Internet Source	<%1
47	github.com Internet Source	<%1

48	Udipi, Aniruddha N., Naveen Muralimanohar, Rajeev Balasubramonian, Al Davis, and Norman P. Jouppi. "Combining memory and a controller with photonics through 3D-stacking to enable scalable and energy-efficient systems", ACM SIGARCH Computer Architecture News, 2011. Publication	<%1
49	www.eng.auburn.edu Internet Source	<%1
50	www.elitebastards.com Internet Source	<%1
51	arm.com Internet Source	<%1
52	www.bg-informatika.com Internet Source	<%1
53	www.coursehero.com Internet Source	<%1
54	burgath.com Internet Source	<%1
55	Microcomputer Systems Using the STE Bus, 1989. Publication	<%1

EXCLUDE QUOTES	ON	EXCLUDE MATCHES	OFF
EXCLUDE BIBLIOGRAPHY	ON		