An IP Core of AMBA Bus Interface in HDL

Sakshi Nagesh[1*], *D K* Mishra[1], *Rajesh* Khatri[1], and *Amit* Naik[1]

[1]Department of Electronics and Instrumentation Engineering, Shri G.S. Institute of Technology and Science, Indore, India

Abstract. The AMBA on-chip bus architecture is a well-known open specification that explains how to connect and manage the functional units that make up a System-On-Chip (SoC). The design and implementation of an AHB Master, RAM, ROM, FIFO and Memory Controller implementation is proposed in this paper. It is primarily divided into two categories: operation initiator (AHB MASTER) and AHB SLAVE. Furthermore, AHB master generate the operation in burst mode, single transfer according to interface requirement and Address generator, generates the address in increment or wrap mode, as well as completing data transfers with an asymmetric asynchronous FIFO with variable data widths for read and write. A bridge between an AHB Master and an AHB slave will be demonstrated using a memory controller, and their outcome in terms of area and speed will be address ed. A finite state machine will be used to design the control framework. Xilinx Virtex 2 XC2VP40 will be used to implement the AHB Master and Slave IP.

1 Introduction

Embedded real-time systems and systems-on-chip (SoC) often contain numerous computational units. These components perform various functions in the processing of an overall solution. Take, for example, a set-top box for televisions [1]. Reduced device size and increased silicon area enable for the creation of very complex integrated circuits with billions of transistors. This allows for new degrees of integration, with several components on a single chip. One or more programmable mechanisms, such as digital signal processors (DSPs), general-purpose processors, or IP blocks specifically developed for certain applications, may be included in SoCs (PI). Analog interfaces, memory, I/O devices, and other components can be included. SoC is a type of integrated circuit that can handle the majority of the functions of an electronic system. General-purpose processors or digital signal processors are the computational elements. Several of these are now incorporated into a System-On-Chip solution. To execute a task, a CPU must communicate with other processors, memory, and I/O devices. Buses are now utilised to connect these IP blocks [2]. The AMBA standard was established and modified throughout time by ARM, one of the world's major producers of processors and microcontrollers for embedded systems, and has a wide range of applications

* Corresponding author: sakshinagesh1999@gmail.com

in electronics (multimedia players, mobile phones, and so on). It has made it the industry's de facto standard with the support of a business like ARM, which is a worldwide recognised forerunner in AMBA IP (Intellectual Property) Compliance and Standard Delivery. In addition to the libraries provided by ARM or the ISV community, another benefit of the AMBA standard is the absence of licencing fees, which can affect development costs and standard interfaces for others. The memory controller and the interconnection should be combined quickly and efficiently. Missing these deadlines costs money, which is displayed on the screen as black boxes or heard as noise. This is unacceptable, hence strict real-time deadlines must always be met. General-purpose processors or digital signal processors are examples of computational elements. Several of them are now combined into a single System-On-Chip solution. In order to execute a task, a CPU must communicate with other processors, memory, and I/O devices. Interconnecting these IP blocks is currently done using buses. Because NoCs provide more flexibility than buses, recent research in the field advocates employing them to interconnect IP blocks [3]. However, according to, there are still open research questions in getting NoCs acknowledged as a communication paradigm in SoCs. How to deal with large amounts of data is an example. Dynamic memory is used to store large amounts of data off-chip. Memory manufacture differs from ordinary logic production hence the separation is required. High data speeds and storage capacities are provided by dynamic memory. Dynamic memories, on the other hand, lack the flexibility to effectively access random memory locations and require refresh cycles to maintain the material. A memory controller is required to facilitate efficient communication between IP blocks and shared memory. Requestors are IP blocks that talk to the memory. Memory accesses are managed by the memory controller, which arbitrates between requestors. The purpose of this thesis is to develop a memory controller that gives strong real-time guarantees and efficient communication between IP blocks and memory. In addition, the memory controller and the interconnection should be simple and efficient to integrate [4-10].

1.1 The AMBA Based Microcontroller Architecture

A high-speed system bus serves the CPU's external devices DMA bandwidth, on-chip memory, and other DMA technologies in an AMBA-based microcontroller. Figure 1 depicts the AMBA. system architecture [9]. The bandwidth of this high-speed system bus is quite huge.

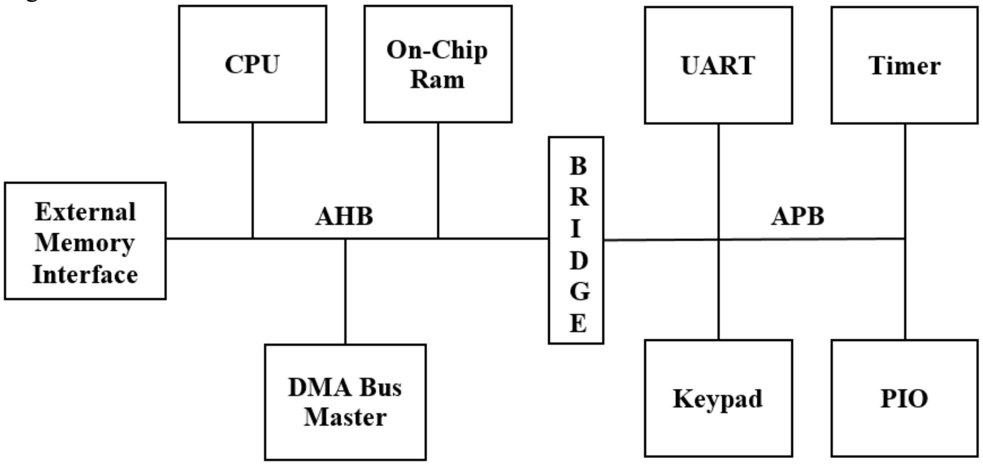

Fig. 1. AMBA based microcontroller [13]

2 Proposed Design

Todays to support system on chip solutions more companies are adopting AMBA bus communication hence it has become more demanding standard of interconnections between processors and other coprocessor. AMBA promotes the reusable IP core design need for SoC components [11]. This study proposes the IP core implantation and its digital aspects of implementation of find trade-off between area and speed. The role-specific operations are the focus of AMBA interaction. There are two main categories: operation initiator (AHB MASTER) and AHB SLAVE. Additionally, AHB master generate the operation in burst mode, single transfer according to interface requirement [12] and Address generator, generates the address in increment or wrap mode, as well as data transfer with read and write data widths. To develop the control framework, a finite state machine will be employed. The controller interface will be constructed and tested, as well as the AHB Master and AHB Slave along with other memory interface.

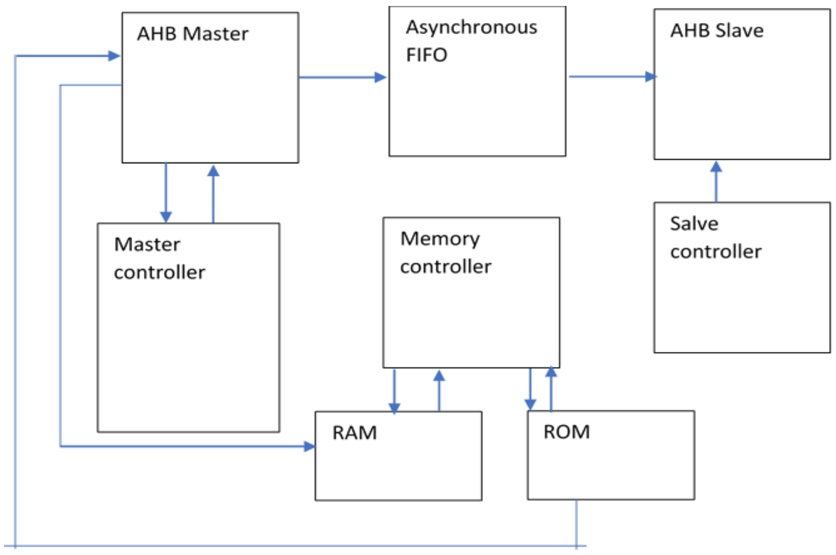

Fig. 2. Proposed system architecture of AMBA with Memory interface

Figure 2 shows the top architecture of an AHB compliant master. FIFO is employed for data and control buffering in the suggested technique. This is done with the intention of assigning separate clock cycles (CLK) to the slave and memory, resulting in error-free communication and a reduction in complexity. Data is then transferred by RAM or ROM, depending on the READ and WRITE processes. The memory controller is based on a finite state machine. It has a "idle" status at first, which means the system is in a "reset state" where no operations are being performed. The FSM is triggered when the operation start signal is received, and the command instruction state determines the next operation based on the instructions given. It advances to RAM READ, RAM WRITE, and ROM READ operations based on the Read/Write instructions.

2.1 AHB Master

The AHB bus master shown in Figure 3. It initiates read and write operations and sends the necessary control signals to the slave devices that provide the data transfer.

Figure 3 illustrates the module of AHB master. There are majorly five modules Address generator, Multiplexer, counter, controller and Interface of various control. Address generator module generates the next address according to command for increment or wrap operation. Controller will take care of sequence of operation including start and end of address generation. It initiates the operation according to HTRANS signal, counter module keeps tract of the clock cycle, and it counts the number of address generated, it can count up to 16 clock cycles. Interface of various control is registers and input coming from controller to pass the control signals for other connecting blocks.

The AHB Master is a data transfer interface that allows the processor to send data to the AHB slave. The master is in charge of transferring data and addresses between the processor and memory. As an HGRANTx signal is activated by an arbitration unit, it creates the relevant control signal. It can operate in burst mode and generates a series of actions based on all handshaking signals. The bus master is in charge of all four types of slave responses as well as their wait state, which is HREADY. It generates four HTRANS transfer types to finish handshaking with the slave's other interfaces. The pin diagram of the AHB Master Interface, which is required for establishing the interface between the AHB Master and the AHB, is shown in Figure 4.

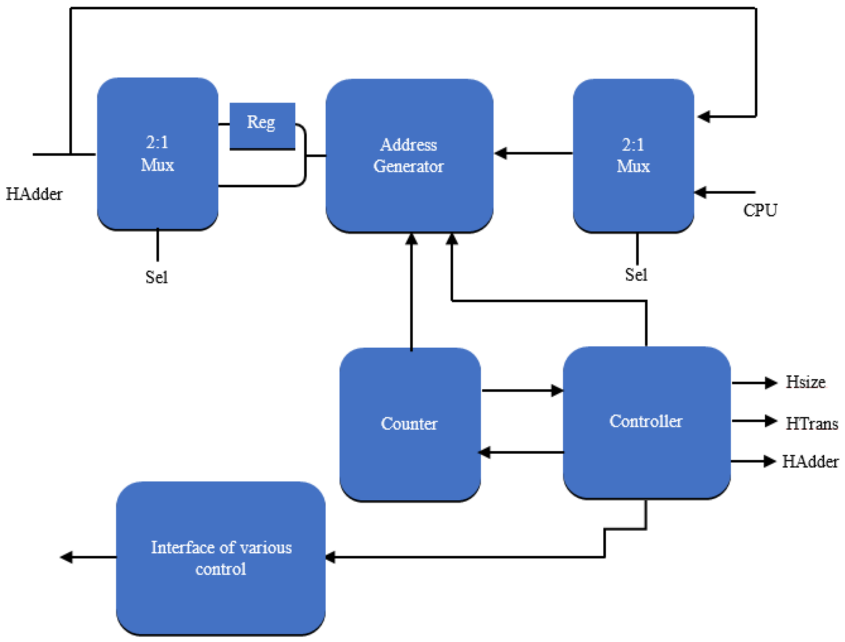

Fig. 3. AHB master architecture

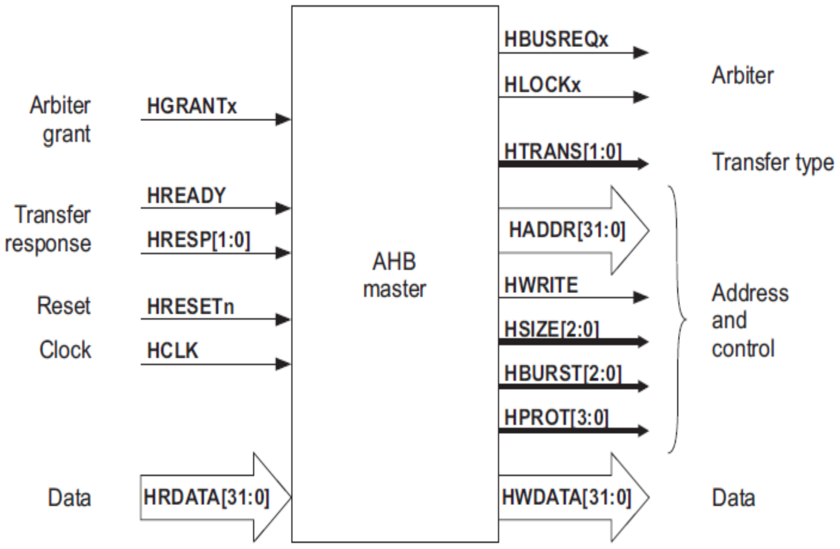

Fig. 4. AHB Bus Master Interface (AMBA™., 1999).

2.2 Slave and Slave Controller

AHB bus slave ensures the transfer or operation completion and generates the handshaking commands to master for successful completion of transfer. The slave generates the response to a bus transfer using an HSELx select signal from the decoder. The bus master creates any other signals according to HSElx, if it high then the read or write address and other control signals is processed. Slave controller is implemented with FSM and generate the handshaking signal to operate the transfer operation. The AHB bus slave presented in Figure 5 replies to a transmission initiated by a bus master in the system.

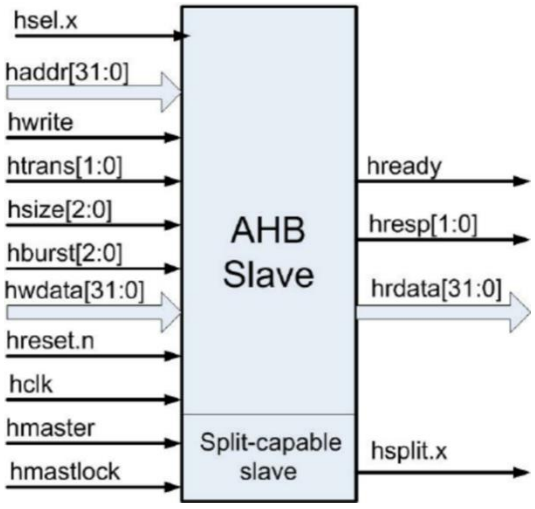

Fig. 5. AHB bus slave interface

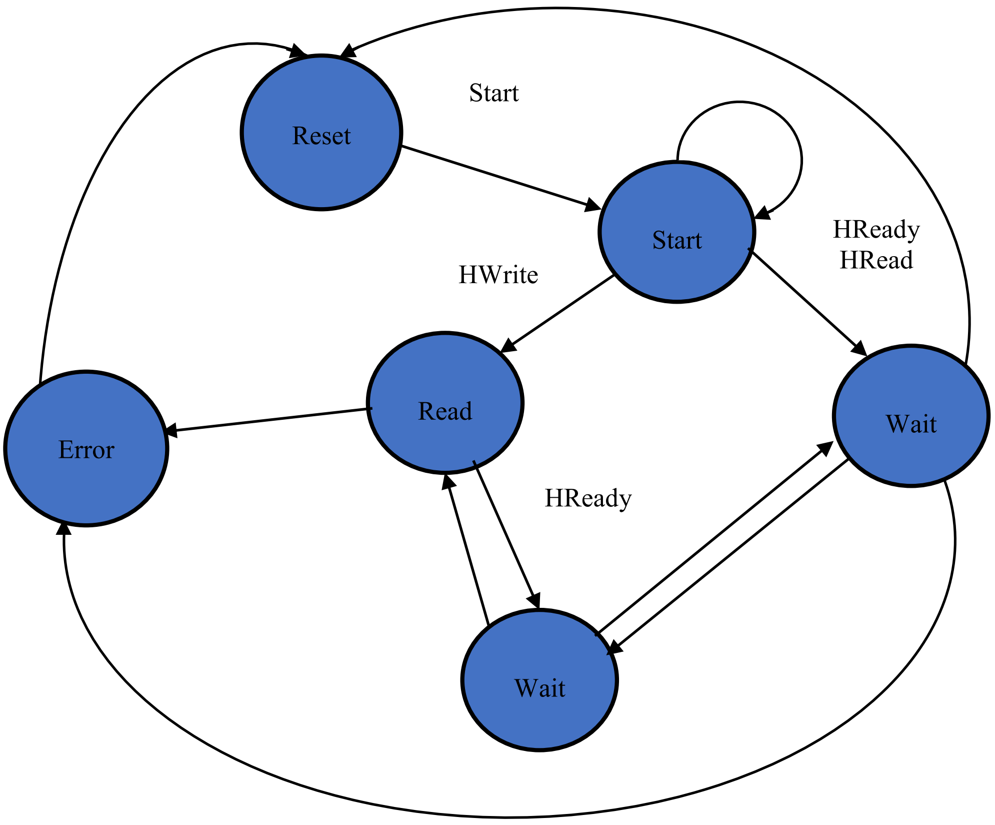

Fig. 6. State diagram for slave interface

Figure 6 presents the state diagram for slave interface. AHB master and slave both works on different protocol so for transferring of signal from AHB master to slave or memory there is a need of interface called slave interface. FSM diagram of slave interface start its operation at start state, it waits for hready signal if it is high, then it performs read operation. It will continuously monitor hready signal, if it goes low than FSM switch its state to wait state, and it will rest in that state unless until hready goes high. If fault is coming than state moves to reset state to begin with operation from start.

2.3 Memory Controller

It is a FSM based implementation. The initial condition is the reset state, where operation waits till valid start signal occurs. It is the ideal state where operation can enter after completion of operation or any stop command comes in the picture. There is command state where operation rom read and ram read or write operations as instructed. The core memory controller's two main responsibilities first to establish the synchronization between various commands and other is to track the current operating condition.

For the primary memory controller, these two main tasks include managing all delays between commands and keeping track of which row is active. It takes time for a row to activate, so the central memory controller has a redirect feature to allow the referee to specify which instruction to execute after the current instruction completes. This allows an early digit to be activated if the subsequent instruction does not reach the similar chip as the existing instruction.

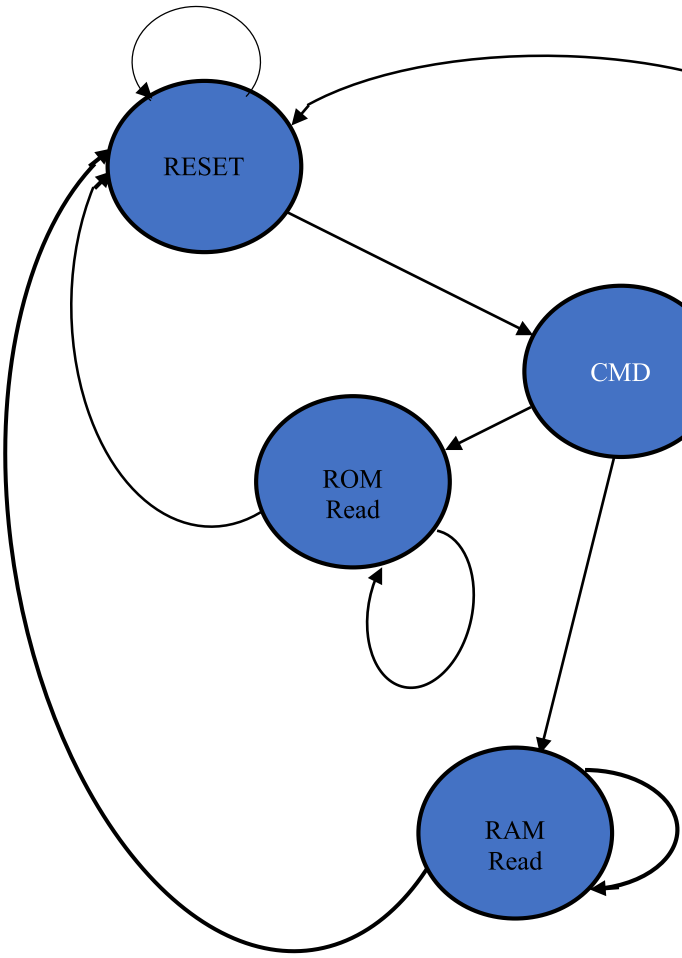

Fig. 7. State diagram for memory controller

2.4 FIFO

FIFO stands for first-in, first-out. It is acting as buffer when read and write operation are happening under different clock frequency. In asynchronous FIFO write operation is perform in one clock frequency and read out with other clock frequency. Order of write and read is defined, the first written will be read out first. The main function of FIFO is to process and retrieve the data. In a FIFO system, the items entered first are excluded first.

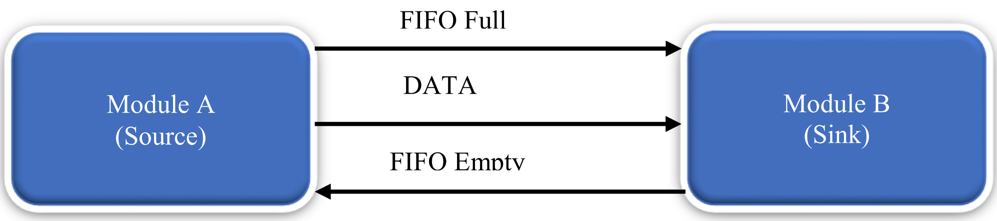

Fig. 8. Asynchronous FIFO

Figure 8 shows the connection between two modules (the source and the sink). The source is the master data and sink is the slave data. The sink is the module that receives the information.

FIFO Full, FIFO Empty, and Data The data wire is the one that carries data from the source to the sink. Handshake signals such as FIFO Full and FIFO Empty are the command signal used by controller to allow communicating the data. The FIFO Full is indication of the FIFO is full, and legitimate data should be placed on the write data channel.

2.5 Designing of RAM

Random access memory (RAM) is also an integrated circuit. The content of each cell must be readable or writable, so it may vary. Unlike ROM, the output corresponding to the bit string of the input address is not fixed immediately after creation, it can change depending on the program used and the data entered into it. A read or write operation is in progress. In a read operation, the bits that make up the address are received by the address decoder, which finds the desired location. Depending on whether this cell contains 0 or 1, the data will appear as 0 in the read/write line or 1 in the read/write cell line and will be used to redirect if you want to write 0 or 1. The data in the desired cell by read/write line 0 or read/write line 1.

Static RAM and RAM Dynamics. We distinguish two kinds of random-access memory: statics and dynamics. In a living memory, a bistable formed of two transistors is utilized to represent a storage element. Without intervention from the outside, the bistable maintains an electrical state representing a binary information. At each of the states, one of the two transistors is saturated and the other is blocked, and only the application of an electrical voltage can cause the bistable to pass from one state to the other. It is the fact that a stable state is maintained without external intervention which makes it called static memory. On the other hand, the so-called dynamic memories are based on the use of a capacitor which maintains between its electrodes a voltage 5 V or 0 V equivalent, which is equivalent to states 1 and 0. However, the capacitor discharges and a periodic refreshing of the memory is required. This means that the reader is read and then rewritten Memory regularly. Therefore, a regular external intervention, refresh, is necessary to maintain the state of a dynamic memory. Despite this drawback, the simplicity of the dynamic RAMs makes it possible to integrate them in greater number on the same silicon chip than their static competitors. It is this characteristic which has contributed to generalization.

2.6 Designing of ROM

ROM is basically an integrated circuit. This means that it is a series of circuits connected together to perform a specific function. Its purpose in this situation is to store particular information and, if necessary, restore it. Therefore, the ROM packet contains various blocks (bits) and circuits necessary to provide the necessary information when the requested information cell address is requested.

3 Simulation Results

This heading describes the simulation and synthesis results obtained by Modelsim and Xilinx respectively. Proposed system architecture of AMBA with memory interface is simulated and their area and speed is analysed in term of MHz and power utilized in terms of mW.

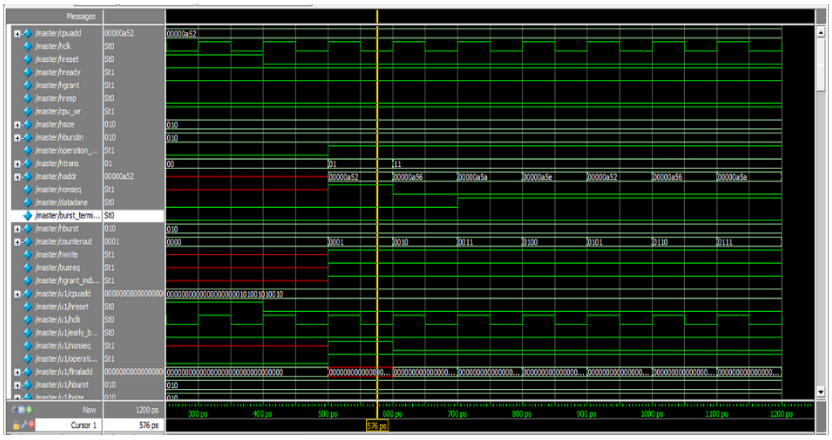

Fig. 9. Simulation waveform of Burst mode operation in AHB master

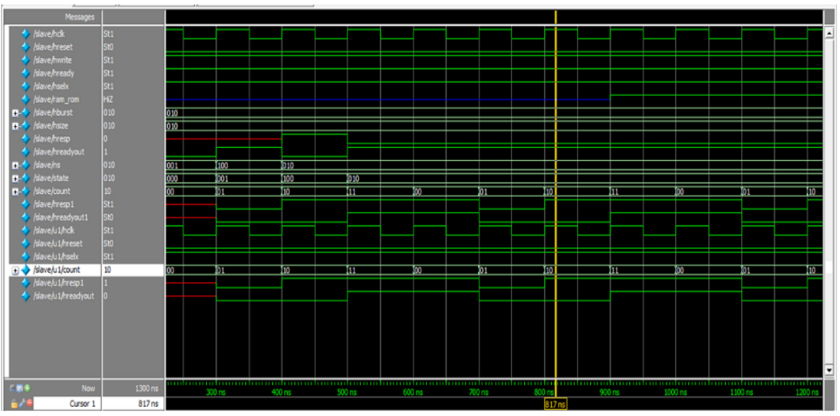

Fig. 10. Simulation waveform of slave interface

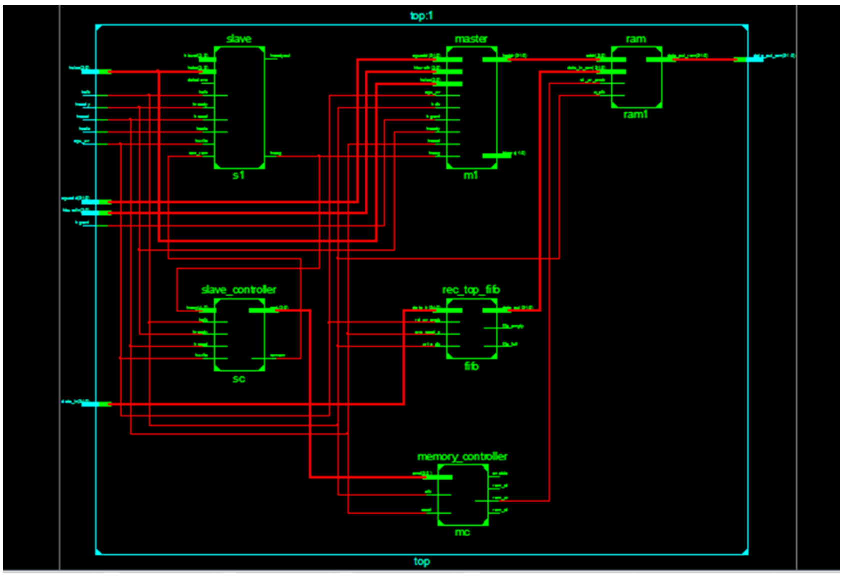

Fig. 11. Top architecture of AHB system

Table 1. Device Utilization Summary.

Count of Slices	163 out of 768 (20%)
Slice Flip-Flops Count	115 out of 1536 (7%)
LUTs with four inputs	269 out of 1536 (17%)
Logic based on numbers	1158
The number of RAMs used	92
Total number of IOs	108
The total number of bonded IOBs	79 out of 124 (63%)
Flip Flops by IOB	32
GCLKs in number	3 out of 8 (37%)

Table 2. Comparative analysis of proposed implementation with previous research works.

Methods	**Speed**	**Power**
[8]	175 Mhz	412 MW
[4]	155.4 MHz	27.34mw
[6]	114.83Mhz	15 mW
Proposed	183.67 MHz	23 mW

Table 1 shows the device utilization summary for the proposed research work with percentage if devices used in simulation, and its comparative analysis with previous research works is presented in Table 2. It was found that the proposed research work outperforms the previous works on the basis of speed.

4 Conclusion

AHB-based Memory Controller is designed in this research. The VHD for SOC solution was used to implement the design. The balance of area and speed has been considered in the design. Master IP has been installed, and its address generating block is constructed with a single adder rather than a 64-bit adder, allowing us to reduce the area by using correct design. Every implementation follows a structural approach, resulting in a well-documented framework. The slave and slave interfaces are implemented separately, and the memory controller is interfaced further. We used FIFO for the memory controller interface and slave interface to avoid the handshaking complexity. Even though it increases latency, it simplifies design and eliminates bottleneck issues.

References

1. Bevoor, S. and Gowda, P.N., 2017, May. *Design of high Performance master/slave memory controller for grey scale image transfer*. In 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT) (pp. 611-614). IEEE.
2. Bhatt, P., and Shah, D., 2018. *Design AMBA based AHB to APB Bridge using Verilog HDL.* Journal of Emerging Technologies and Innovative Research (JETIR), 5(4), pp.1080-1084.

3. Bhowmik, B., Biswas, S., Deka, J.K. and Bhattacharya, B.B., 2019. *A low-cost test solution for reliable communication in networks-on-chip*. Journal of Electronic Testing, 35(2), pp.215-243.
4. S. Ramagundam et al., "Design and implementation of high Performance master/slave memory controller with microcontroller bus architecture," Conf. Rec. - IEEE Instrum. Meas. Technol. Conf., no. May, pp. 10–15, 2014.
5. D. Simon and U. Guruprasad, "*Design and implantation of AMBA-Memory controller for image transfer applications*," IJRET Int. J. Res. Eng. Technol., vol. 5, no. 4, pp. 290–293, 2016.
6. S. Rao and A. S. Phadke, "*Testing of AMBA Compliant Memory Controller using Pattern Generator / Logic Analyser*," Int. J. Adv. Res. Electr. Electron. Instrum. Eng., vol. 2, no. 6, pp. 2227–2233, 2013.
7. A. B. Nithin Joe, "*D Esign and a Nalysis of a N Ovel Digital*," Int. J. Adv. Eng. Technol., vol. 4, no. pp. 2053–2063, 2013.
8. R.S. Kurmi and A. Somkuwar, "*Design of AHB Protocol Block for Advanced Microcontrollers*," Int. J. Comput. Appl. (0975, vol. 32, no. 8, pp. 23–29, 2011.
9. D. Jayapraveen and T. G. Priya, "*Design of memory controller based on protocol*," Elixir Comp. Sci. Engg. 51A, vol. 2, pp. 11115–11119, 2012.
10. P. S. Shete and S. Oza, "*Design of an Efficient FSM for an Implementation of AMBA AHB Master*," Int. J. Adv. Res. Comput. Sci. Softw. Eng., vol. 4, no. 3, pp. 267–271, 2014.
11. Z. Khan, T. Arslan, and A. T. Erdogan, "A novel bus encoding scheme from energy and crosstalk efficiency perspective for AMBA based generic SoC systems," Proc. IEEE Int. Conf. VLSI Des., no. January, pp. 751–756, 2005.
12. M. K. Kumar, A. Sajja, and F. Noorbasha, "Design and FPGA Implementation of AMBA APB Bridge with Clock Skew Minimization Technique," vol. 7, no. 3, pp. 42–45, 2017.
13. A. R. M. Ihi, "AMBA TM Specification."