Design and Implementation of Multiple - Master Multiple-Slave AMBA AHB Protocol Block with and Without Split/Retry Transfer for Advanced Microcontroller in Verilog/VHDL

[1]Kajol Singh, [2]Shefali Verma, [3]Shobha Sharma
[1,2,3]Dept. of Electronics and Comm., Indira Gandhi Delhi Tech. Univ. for Women, New Delhi, India

Abstract
The on-chip interconnection system known as Advanced Microcontroller Bus Architecture (AMBA) is a well established open specification for the proper management of functional blocks comprising system-on-chips (SOCs). In this subject paper, the design and implementation details of AMBA advanced high bus (AHB) with split/ retry transfer and without splitting are shown. The AHB design contains basic blocks such as master and slave and the working of these blocks are based on arbitration scheme. Multiplexer and Decoders are used to select the appropriate signals between masters and slaves that are involved in the transfer. The SPLIT and RETRY responses provide a mechanism for slaves to release the bus when they are unable to supply data for a transfer immediately. Both mechanisms allow the transfer to finish on the bus and therefore allow a higher-priority master to get access to the bus. The designing of AMBA (AHB) is done on QUESTASIM tool by verilog language.

Keywords
Advanced Microcontroller Bus Architecture (AMBA), Advanced High Performance Bus (AHB), System-On-Chips (SOC).

I. Introduction

In recent days, the development of SOC chips and the reusable IP cores are given higher priority because of its less cost and reduction in the period of Time-to-market. So this enables the major and very sensitive issue such as interfacing of these IP cores. These interfaces play a vital role in SOC and should be taken care because of the communication between the IP cores property. There are many interconnect buses that are widely used in the industry like AMBA, Wishbone, Core Connect, Avalon etc. AMBA is most preferred among all of them because it has a hierarchy of buses like AHB(Advance high performance bus) that can be connected to high performance peripherals and APB (Advance Peripheral Bus) that can be connected to low performance peripherals.

II. Microcontroller Structure Based on AMBA-AHB

This type of microcontroller structure consists of High performance ARM processor, High bandwidth on-chip RAM, High bandwidth external memory, Direct memory access (DMA) device, Bridge as a converter , UART, Timer, Keypad, PIO and other devices based on application as shown in fig. 1. An AMBA based microcontroller typically consists of a high performance system backbone bus (AMBA-AHB), able to sustain the external memory bandwidth, on which the above given devices reside. This bus provides a high bandwidth interface between the elements that are involved in the majority of transfers. Also located on the high performance bus is a bridge to the lower bandwidth APB, where most of the peripheral devices in the system are located.

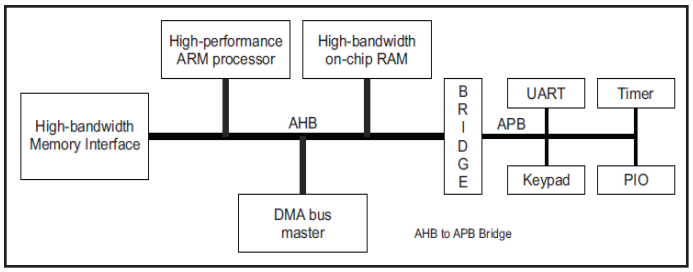

Fig. 1: AMBA Based Microcontroller

III. Advanced High Performance Bus (AHB) Protocol

The AHB 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 act as the high performance system backbone bus. It supports features such as:-

1. AHB is defined with a choice of several bus widths, from 8-bit to 1024-bit. The most common implementation has been 32-bit, but higher bandwidth requirements may be satisfied by using 64 or 128-bit buses.
2. AHB used the HRESP signals driven by the slaves to indicate when an error has occurred.
3. AHB also offers a large selection of verification IP from several different suppliers. The solutions offered support several different languages and run in a choice of environments.
4. AHB can be used at higher frequency along with separate data buses that can be defined to 128-bit and above to achieve the bandwidth required for high-performance bus applications.
5. AHB can access other protocols through the proper bridging converter. Hence it supports the bridge configuration for data transfer.

AHB offers burst capability by defining incrementing bursts of specified length.
It supports SEQ, NONSEQ, BUSY and IDLE transfers types. AHB also offers a fairly low cost (in area), low power (based on I/O) bus with a moderate amount of complexity and it can achieve higher frequencies.

IV. AHB Bus Interconnection

The AMBA AHB bus protocol is designed to be used with a central multiplexor interconnection scheme as shown in fig. 2. Using this scheme all bus masters drive out the address and control signals indicating the transfer they wish to perform and the arbiter determines which master has its address and control signals routed to all of the slaves. A central decoder is also required to control the read data and response signal multiplexor, which selects the appropriate signals from the slave that is involved in the transfer

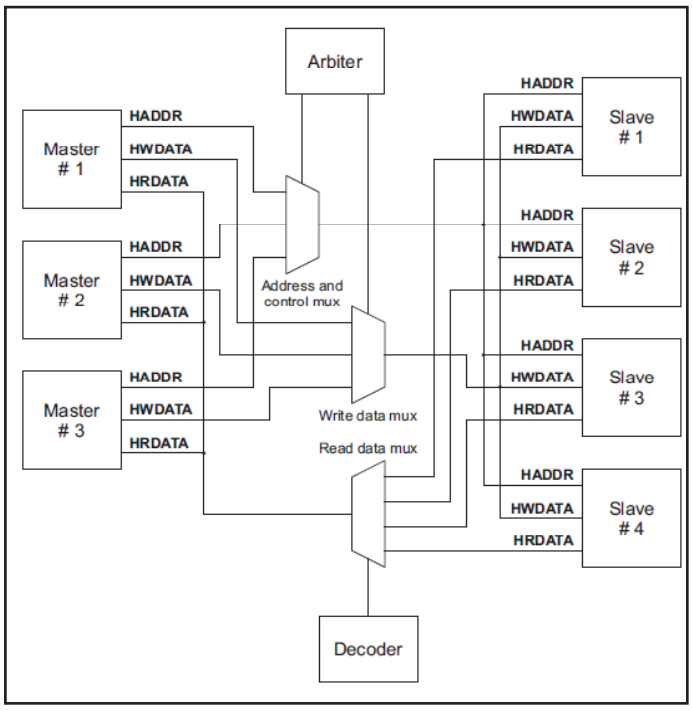

Fig. 2: Bus Interconnection of AHB

V. Design of AMBA-AHB

AMBA AHB uses **HTRANS[1:0]** signal whose width is 2-bit as a transfer type. This signal is driven by master who indicates the type of the current transfer happening. 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.

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. Slaves must always provide a zero wait state OKAY response. The NONSEQ transfer type indicates the first transfer of a burst or a single transfer. The address and control signals are unrelated to the previous transfer. 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 or 16). AMBA AHB uses **HBURST[2:0]** signal whose width is 3-bit. This signal is driven by master who indicates if the transfer forms part of a burst. Four, eight and sixteen-beat bursts are defined in the AMBA AHB protocol, as well as undefined-length bursts and single transfers. Both incrementing and wrapping bursts are supported in the protocol:

Incrementing bursts access sequential locations and the address of each transfer in the burst is just an increment of the previous address. For **wrapping bursts,** if the start address of the transfer is not aligned to the total number of bytes in the burst (size x beats) then the address of the transfers in the burst will wrap when the boundary is reached. In this paper, only incrementing bursts(4, 8 and 16-bit) is shown. An incrementing burst can be of any length, but the upper limit is set by the fact that the address must not cross a 1kB boundary.

A. Slave transfer responses

After a master has started a transfer, the slave then determines how the transfer should progress. Whenever a slave is accessed it must provide a response which indicates the status of the transfer. The **HREADY** signal is used to extend the transfer and this works in combination with the response signals, **HRESP[1:0],** which provide the status of the transfer. The slave can complete the transfer in a number of ways. It can complete the transfer immediately, insert one or more wait states to allow time to complete the transfer, signal an error to indicate that the transfer has failed or delay the completion of the transfer, but allow the master and slave to back off the bus, leaving it available for other transfers.

Every slave must have a predetermined maximum number of wait states that it will insert before it backs off the bus, in order to allow the calculation of the latency of accessing the bus. It is recommended, but not mandatory, that slaves do not insert more than 16 wait states to prevent any single access locking the bus for a large number of clock cycles.

A typical slave will use the **HREADY** signal to insert the appropriate number of wait states into the transfer and then the transfer will complete with **HREADY** HIGH and OKAY response, which indicates the successful completion of the transfer. The ERROR response is used by a slave to indicate some form of error condition with the associated transfer. Typically this is used for a protection error, such as an attempt to write to a read-only memory location. The SPLIT and RETRY response combinations allow slaves to delay the completion of a transfer, but free up the bus for use by other masters. These response combinations are usually only required by slaves that have a high access latency and can make use of these response codes to ensure that other masters are not prevented from accessing the bus for long periods of time.

B. Address Decoding

A central address decoder is used to provide a select signal, **HSELx,** for each slave on the bus as shown in figure 3. The select signal is a combinatorial decode of the high-order address signals. A slave must only sample the address and control signals and **HSELx** when **HREADY** is HIGH, indicating that the current transfer is completing. Under certain condition, it is possible that **HSELx** will be asserted when **HREADY** is LOW, but the selected slave will have changed by the time the current transfer completes. If a NONSEQUENTIAL or SEQUENTIAL transfer is attempted to a non-existent address location then the default slave should provide an ERROR response. IDLE or BUSY transfers to nonexistent locations should result in a zero wait state OKAY response.

C. Arbitration

The arbitration mechanism is used to ensure that only one master has access to the bus at any one time. The arbiter as shown in fig. 4 performs this function by observing a number of different requests to use the bus and deciding which is currently the highest priority master requesting the bus. The arbiter also receives requests from slaves that wish to complete SPLIT transfers. Any slaves which are not capable of performing SPLIT transfers do not need to be aware of the arbitration process, except that they need to observe the fact that a burst of transfers may not complete if the ownership of the bus is changed.

A brief description of each of the arbitration signals is given below:

HBUSREQx, the bus request signal is used by a bus master to request access to the bus. **HLOCKx,** indicates to the arbiter that the master is performing a number of indivisible transfers and

the arbiter must not grant any other bus master access to the bus once the first transfer of the locked transfers has commenced. **HGRANTx,** the grant signal is generated by the arbiter and indicates that the appropriate master is currently the highest priority master requesting the bus, taking into account locked transfers and SPLIT transfers. A master gains ownership of the address bus when **HGRANTx** is HIGH and **HREADY** is HIGH at the rising edge of HCLK. Through **HMASTER[3:0],** the arbiter indicates which master is currently granted the bus. The arbiter indicates that the current transfer is part of a locked sequence by asserting the **HMASTLOCK** signal, which has the same timing as the address and control signals. **HSPLIT[15:0],** the 16-bit Split Complete bus is used by a SPLIT-capable slave to indicate which bus master can complete a SPLIT transaction.

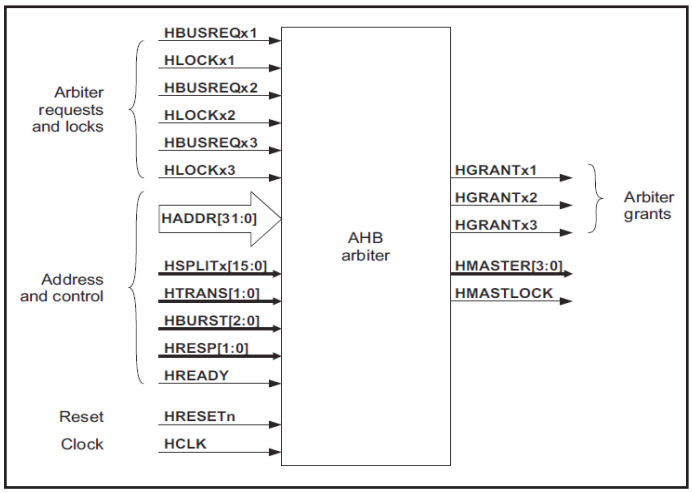

Fig. 4: AHB Arbiter

VI. Split Transfers

SPLIT transfers improve the overall utilization of the bus by separating (or splitting) the operation of the master providing the address to a slave from the operation of the slave responding with the appropriate data.

When a transfer occurs the slave can decide to issue a SPLIT response if it believes the transfer will take a large number of cycles to perform. This signals to the arbiter that the master which is attempting the transfer should not be granted access to the bus until the slave indicates it is ready to complete the transfer. Therefore the arbiter is responsible for observing the response signals and internally masking any requests from masters which have been SPLIT. During the address phase of a transfer the arbiter generates a tag, or bus master number, on **HMASTER[3:0]** which identifies the master that is performing the transfer. Any slave issuing a SPLIT response must be capable of indicating that it can complete the transfer, and it does this by making a note of the master number on the **HMASTER[3:0]** signals. Later, when the slave can complete the transfer, it asserts the appropriate bit, according to the master number, on the **HSPLITx[15:0]** signals from the slave to the arbiter. The arbiter then uses this information to unmask the request signal from the master and in due course the master will be granted access to the bus to retry the transfer. The arbiter samples the **HSPLITx** bus every cycle and therefore the slave only needs to assert the appropriate bit for a single cycle in order for the arbiter to recognize it. In a system with multiple SPLIT-capable slaves the **HSPLITx** buses from each slave can be ORed together to provide a single resultant **HSPLIT** bus to the arbiter. In the majority of systems the maximum capacity of 16 bus masters will not be used and therefore the arbiter only requires an **HSPLIT** bus which has the same number of bits as there are bus masters. However, it is recommended that all SPLIT-capable slaves are designed to support up to 16 masters.

A. Split Transfer Sequence

The basic stages of a SPLIT transaction are:

1. The master starts the transfer in an identical way to any other transfer and issues address and control information
2. If the slave is able to provide data immediately it may do so. If the slave decides that it may take a number of cycles to obtain the data it gives a SPLIT transfer response. During every transfer the arbiter broadcasts a number, or tag, showing which master is using the bus. The slave must record this number, to use it to restart the transfer at a later time.
3. The arbiter grants other masters use of the bus and the action of the SPLIT response allows bus master handover to occur. If all other masters have also received a SPLIT response then the default master is granted.
4. When the slave is ready to complete the transfer it asserts the appropriate bit of the HSPLITx bus to the arbiter to indicate which master should be regranted access to the bus.
5. The arbiter observes the HSPLITx signals on every cycle, and when any bit of HSPLITx is asserted the arbiter restores the priority of the appropriate master.
6. Eventually the arbiter will grant the master so it can re-attempt the transfer. This may not occur immediately if a higher priority master is using the bus.
7. When the transfer eventually takes place the slave finishes with an OKAY transfer response.

VII. Results

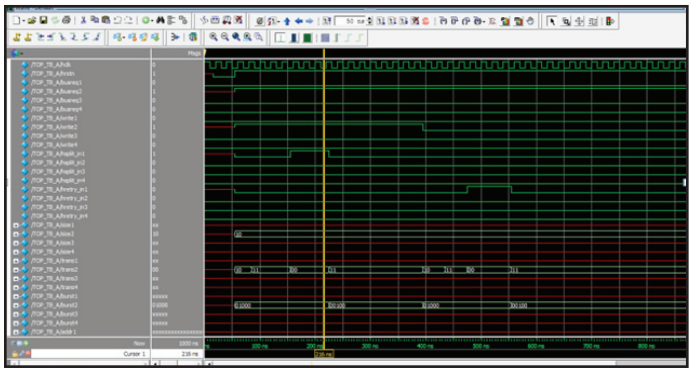

Fig. 5: Complete data transfer cycles with splitting; with hclk as clock signal given to design and active low signal hrstn used to reset the system. Here master 2 is making request to transfer data. Busreq2=1.

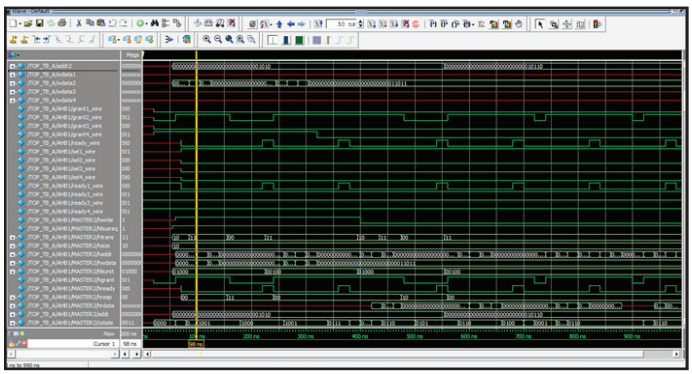

Fig. 6: Extended Signals of Design HBUSREQ2=1 and HGRANT2=1

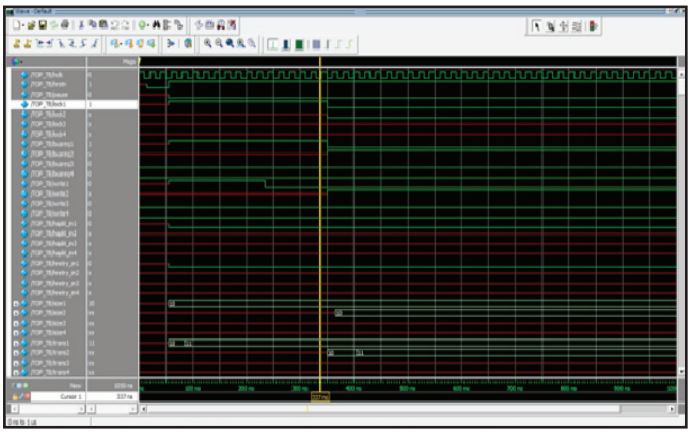

Fig. 7: Complete Data Transfer Cycles Without Splitting. Here master1 and master2 are making requests busreq1 and busreq2 with locked access to bus by using lock1 and lock2 signals.

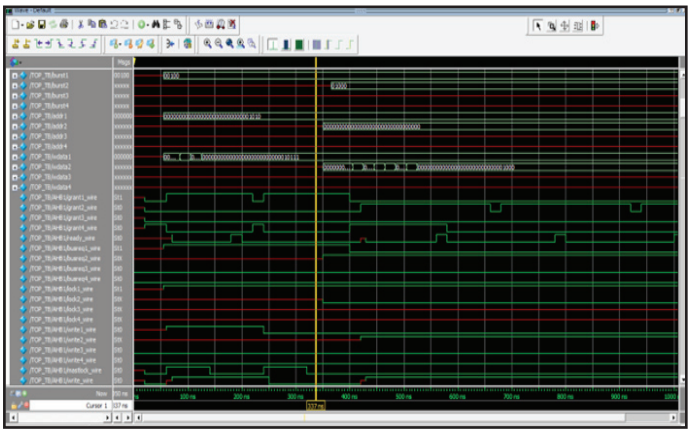

Fig. 8: Extended Signals of Design Without Splitting

VIII. Conclusion

We have designed the intellectual properties of master and slave depending upon the design specification, data transfer and various transfer modes that are supported by AMBA bus architecture. The various scenarios for each component in the AMBA-AHB bus design are verified effectively during the simulation with respect to its specification. The main goal of this work is to study unique design feature of AMBA – AHB i.e. split transfers that has increased the bus bandwidth. The SPLIT and RETRY response combinations has allow slaves to delay the completion of a transfer, but free up the bus for use by other masters. These response combinations are usually only required by slaves that have high access latency and can make use of these response codes to ensure that other masters are not prevented from accessing the bus for long periods of time. SPLIT response tells the Arbiter to give priority to all other masters until the SPLIT transfer can be completed. A SPLIT response is more complicated to implement than a RETRY, but has the advantage that it allows the maximum efficiency to be made of the bus bandwidth.

References

[1] "AMBA Specification Rev 2.0. ARM Ltd", 1999.

[2] [Online] Available: http://www.arm.com/pdfs/ahb_overview.pdf

[3] "AMBA Design Kit revision r3p0", Technical Reference Manual, ARM Inc.

[4] "AHB Example AMBA System", Technical Reference Manual, ARM Inc.

[5] A Verilog HDL test bench primer.

[6] Verilog HDL A guide to digital design and synthesis by Samir Palnitkar.

[7] Yashdeep Godhal, Krishnendu Chatterjee, Thomas A. Henzinger,"Synthesis of AMBA AHB from Formal Specification: A Case Study", In International Journal on Software Tools for Technology Transfer, July 2011.

[8] Yangyang Li, Wuchen Wu, LigangHou, Hao Cheng, "A Study on the Assertion-Based Verification of Digital IC", In Proc. of Second International Conference on Information and Computing Science, 2009, pp. 25-28.

Kajol Singh obtained her B.tech degree in electronics and communication from maharaja Surajmal institute of technology in 2014. She is currently pursuing M.tech in VLSI design from Indira Gandhi Delhi Technical University for Women, New Delhi, India. Her areas of interests include CMOS analog and Digital design.

Shefali Verma obtained her B.tech degree in electronics and communication from H.M.R Institute of technology and management in 2013. She is currently pursuing M.tech in VLSI design from Indira Gandhi Delhi Technical University for Women, New Delhi, India. Her areas of interests include Low Power Circuit design, CMOS analog and Digital design.

Shobha Sharma received her M.E degree in ECE from BITS Pilani. She is pursuing her Ph.D from Guru Gobing Singh Indraprastha University. She joined IGDTUW in 2002 and is currently working as assistant professor in department of Electronics and Communication in IGDTUW. She has published more than 23 research papers in international and national journals. Her areas of interest are VLSI Design, Digital Circuits and Systems, Advanced Computer Architecture.