

Design and implementation of High performance Bus Architecture using FPGA

Muhammed Asharaf.T.P¹, Manu Ramesh², Sankarnarayana Bhat.M³

¹ Manipal University, Manipal Institute of Technology,
Manipal, udupi, 576104.
muhammedashru@gmail.com

² Manipal University, Manipal Institute of Technology,
Manipal, udupi, 576104.
ramesh.manu0@gmail.com

³ Manipal University, Manipal Institute of Technology,
Manipal, udupi, 576104.
msnbhat299@gmail.com

Abstract: In any system on chip (SoC) a reliable and optimized communication protocol is inevitable. In SoC communication high performance, low power consumption and testability are essential. AMBA bus architecture is an SoC communication protocol that aims at high performance and low power consumption by partitioning based on the bandwidth with which the devices operate, within the system. In this paper we discuss about the design of two such buses AXI (Advanced Extensible Interface) and APB (Advanced Peripheral Bus) in a four master four slave system. The whole design is simulated and implemented in FPGA.

Keywords: Advanced Microcontroller Bus Architecture (AMBA), System On Chip (SoC), Advanced Extensible Interface (AXI), Advanced Peripheral Bus (APB).

1. Introduction

In the modern era of VLSI technology the computers depend more on system on-chip communication (SoC) protocols for data exchange [1]. In any SoC the communication standard should be reliable. With the aim to deal with many off chip communication issues Advanced RISC Machine (ARM) introduced Advanced Microcontroller Bus Architecture (AMBA) in 1996. AMBA defines both a bus specification and a technology-independent methodology for designing, implementing [2], and testing customized embedded controllers. Introduction of AMBA buses by ARM made a remarkable change in the existing communication scenarios. It was a solution for many existing bottle necks like adhoc design approach, infrastructure portability, centralized state machine etc [3]. ARM has defined various bus specifications from 1996. Broadly, such specifications address two types of busses: system bus and peripheral bus [4]. In this paper we discuss two main bus specification of AMBA bus (AXI 4 and APB 4) and its communication between each other. Advanced Extensible Interface (AXI) is main system bus for high speed operation and Advanced peripheral Bus (APB) is the bus used for connecting low bandwidth peripherals where high speed performance is not required. In the design we have used four masters which communicate with four slaves.

The masters operate on the AXI side. Slaves are APB Bridge which forms the interface between the high speed AXI bus and low bandwidth APB bus [5]. APB protocol is implemented in the form of a controller which is connected to the bridge [6]. By partitioning high- and low-bandwidth devices within the system, AMBA ensures energy-efficient designs, suitable for low-power CPU cores.

The whole design is coded in Verilog and implemented in FPGA.

2. AXI (Advanced Extensible interface)

ARM included AXI in 2010, which is a replacement for AHB bus protocol. It's also designed in a way to perform as submicron interconnects. AXI having five independent channels which work based on handshaking mechanism [10]. All these five channels support duplex operation.

2.1 AXI handshaking Mechanism

In AXI 4.0 specification, there are five independent channels with corresponding handshake signals for every transaction. The source needs to assert VALID when control information or data is available and READY should be asserted by the destination when it accepts the control information or data. Transfer occurs only if both READY and VALID signals are asserted. A VALID signals should be asserted before READY signal gets asserted. In Figure 1 we can see all possibilities of VALID/READY handshaking. The arrows in the Figure 1

show the occurrence of respective transactions. In AXI, the source needs to sample the READY signal and is to be declared as a register. Similarly, the destination needs to sample the VALID signal. In case the READY and VALID signals are not declared as register then the source and destination will take one additional cycle for the next transaction and leads to error signals [5] [10].

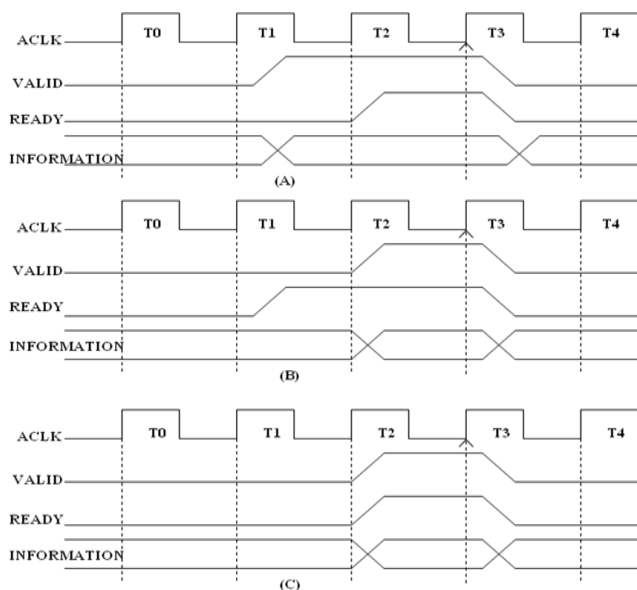


Figure 1: AXI handshaking

3. Advanced Peripheral Bus (APB)

APB bus deals with low power peripheral devices where high performance is not required [6]. All the signal transitions are occurring during the positive edge of the clock. APB is designed for low power consumption. APB is a non-pipelined protocol.

3.1 State Diagram

APB operation takes place through three states, namely Idle, Setup and Enable as shown in Figure 2.

IDLE : The default state of the APB bus.

SETUP: The bus moves into the SETUP state when a transfer is required. Here the appropriate select signal, PSEL, is asserted. The SETUP state lasts for one clock cycle and will always move to the ENABLE state on the next rising edge of the clock.

ENABLE: Enable signal. PENABLE is asserted in this state. During the transition from the SETUP to ENABLE state address, write and select signals should remain stable. This state also lasts for only one clock cycle but the transition from this state to idle or setup depends on the requirement of a transaction. If a transfer is required then the bus moves to setup state otherwise goes to idle state [6].

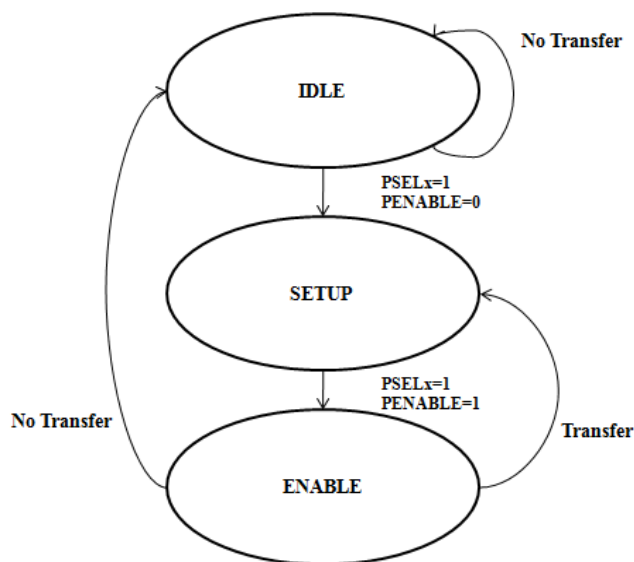


Figure 2: APB state machine [5]

3.2 APB Transactions

Figure 3 shows the timing diagram of write transfer of APB.

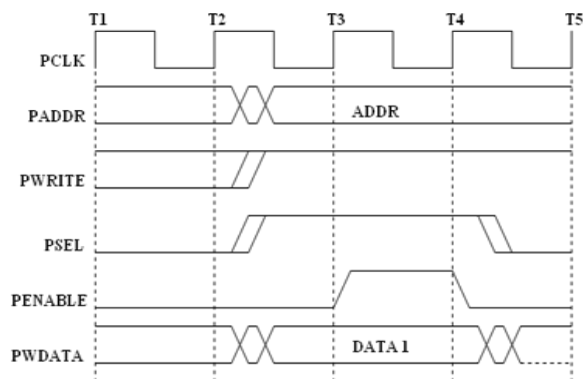


Figure 3: Write Operation

3.2.1 Write Operation

The state machine of the protocol can be understood from the timing diagram of write transfer shown in Figure 3 above. The first cycle T1 indicates the idle state where no transaction takes place and no control signals are asserted. The write operation starts in the cycle T2. Here all the signals except the enable signal changes. This indicates the setup phase. During T3 the enable control signal PENABLE is asserted which shows the access phase. Actual transfer of the data takes place in this cycle. All the signals are valid during the enable cycle. The write transfer completes at the end of this cycle. The write control signal PWRITE is asserted high during the write transfer, which indicates that the operation is a write transfer.

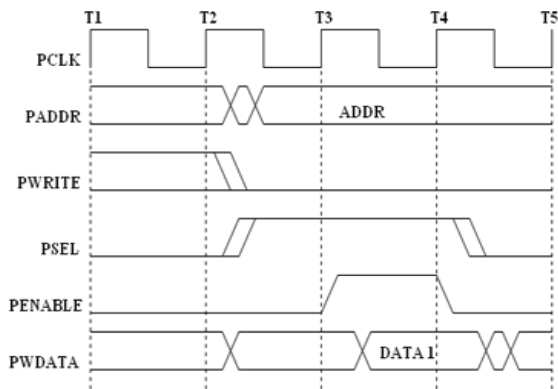


Figure 4: Read Operation

3.2.2 Read Operation

Read operation on APB is very much similar to write operation. The timing diagram is shown in Figure 4. The three states of the state machine can be understood as in the case of write operation. In this bus architecture, there is no read control signal for peripherals [4]. The write control signal pwrite is used for both write and read transfer. When it is a read transfer pwrite is made low. Read transfer take place only after making pwrite active low.

4. Design overview

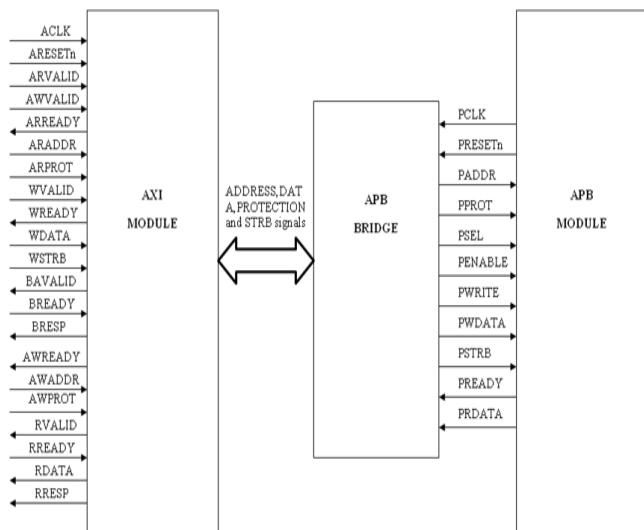


Figure 5: Main block diagram

The overall design overview is shown in the Figure 5. The design has mainly three sections the AXI part, the Bridge and the APB portion. AXI is the system bus which supports the high speed operation. In this paper the communication of AXI is with the low bandwidth peripheral devices. This is done through APB. For enabling this transaction we require a bridge. AXI to APB Bridge make the signals which are used in AXI side suitable to use in the APB side [6]. The signals shown in Figure 5, are the AXI and APB signals required for the transactions. Individual blocks are explained in the following section.

4.1 AXI Module

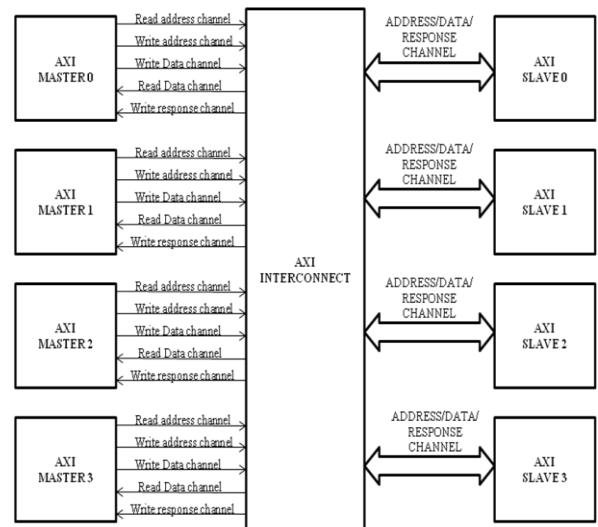


Figure 6: AXI Master to slave connection

The Figure 6 shows the signal transaction between AXI master and slave. Here we have designed 4 master and 4 slaves. Slaves are bridges which will convert the AXI signals to corresponding APB signals. The APB bridge acts as master in APB module and all the transactions initiated by the AXI masters. Whenever AXI master tries to access the slave, it requires completing the handshaking process with the corresponding slave. If two masters trying to access the same slave, Master 0 will have the higher priority and Master 3 has the least priority. If a master requests write and read operation simultaneously then read will occur first then write transaction will take place.

4.2 AXI Interconnect

In AXI, the interconnect is user defined. We designed the interconnect using decoder and multiplexers. All the masters are connected to all multiplexers. These multiplexers are connected to corresponding slaves. The selection of the multiplexers is decided by the decoder shown in Figure 7. The decoder decides the multiplexer by decoding the read or write address given by the ARADDR bus. Once any of the multiplexer is enabled the data transfer takes place. Activated multiplexer will send the data to the corresponding bridge.

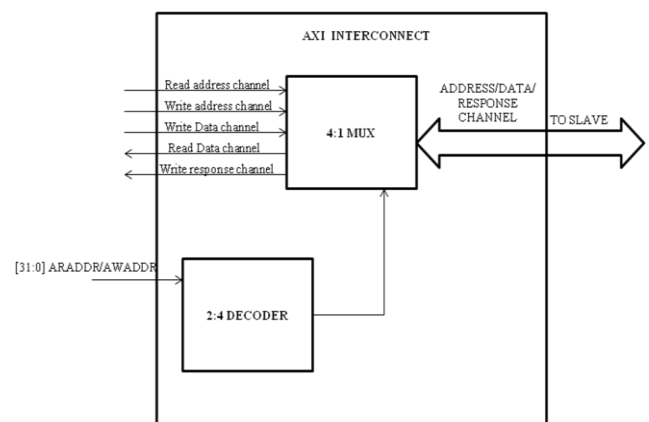


Figure 7: AXI Interconnect

The bridge being the master of APB, sends or receives the data to or from the peripheral devices.

4.3 Low Bandwidth Peripheral Controller:

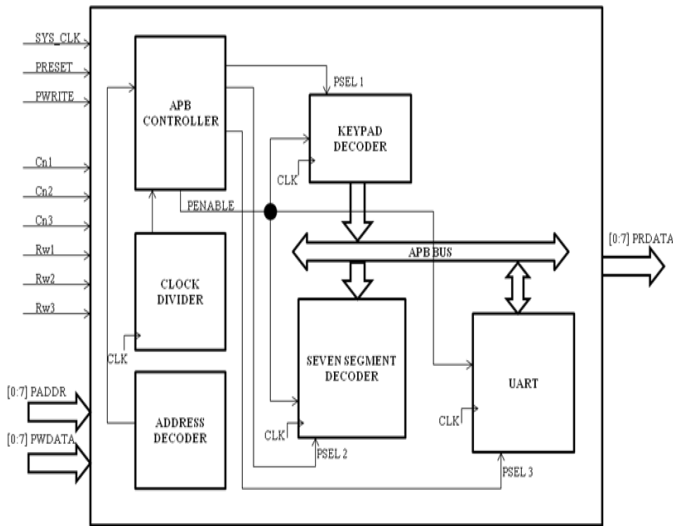


Figure8: APB controller

In the design we have used AXI slave 0 as an APB bridge and all the slave devices are connected to this bridge through a controller and APB Bridge acts as the master. The overall design module of AXI slave 0 is shown in the Figure 8. It consists of an APB Controller, address decoder and the low bandwidth peripheral devices. The slave devices include UART, seven segment decoder and a keypad decoder. Controller uses the APB protocol for the communication with the slave devices. Whenever a write operation is required, the controller will assert pwrite signal. For a read operation pwrite is made low. There is no read control signal in the APB.

4.4 I/O Peripherals

In this work, three I/O devices were made use of, namely, keypad decoder, Seven segment decoder and UART.

4.4.1 Address Decoder

The APB controller uses the address decoder for the selection of slave devices. Based on the address received from the bridge, over the address bus of APB (PADDR), address decoder selects the appropriate slave. This is done by asserting the corresponding PSEL signal. Once any of the PSEL signal is asserted the controller grants the access to the master on the AXI side which has placed request to the bridge for that particular slave. When the master gets the access requested slave, the required transaction can follow.

4.4.2 Keypad Decoder

It is yet another peripheral device used in the design. It is selected by asserting psel1. It is an input device. A 3 X 3 keypad is used. Cn1, Cn2, Cn3 are the column lines and Rw1, Rw2, Rw3 are the row lines. When a key is pressed the algorithm will generate the value by according to the row and column selected.

4.4.3 Seven Segment Decoder

The output device used is a seven segment display. It is selected by asserting psel3. It gives the seven segment code for the value entered through key pad.

4.4.4 UART module

Another peripheral device used is UART (Universal asynchronous receiver transmitter). In UART, the receiver and transmitter operate asynchronously. UART design consists of three sub modules, namely, transmitter receiver and a baud rate generator. The transmitter and the receiver operate with the clock generated by the baud rate generator using a system clock.

5.Results

The design is coded using verilog and simulated using XILINX ISIM simulator. Figure 9 shows the simulation result when master0 requires to read a character from the keypad.

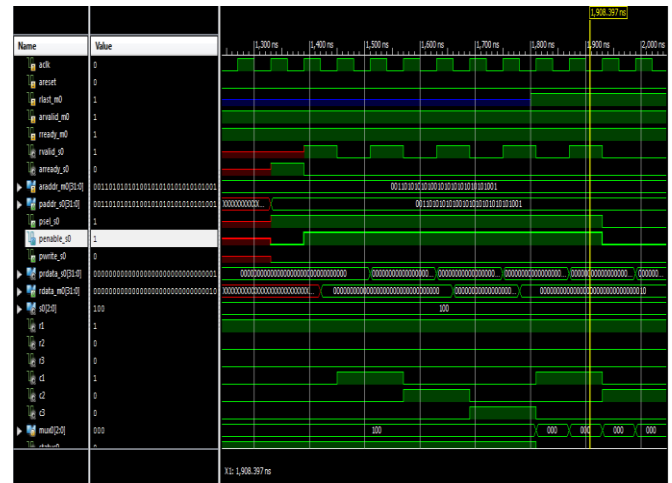


Figure 9: Master0 reading the data from keypad

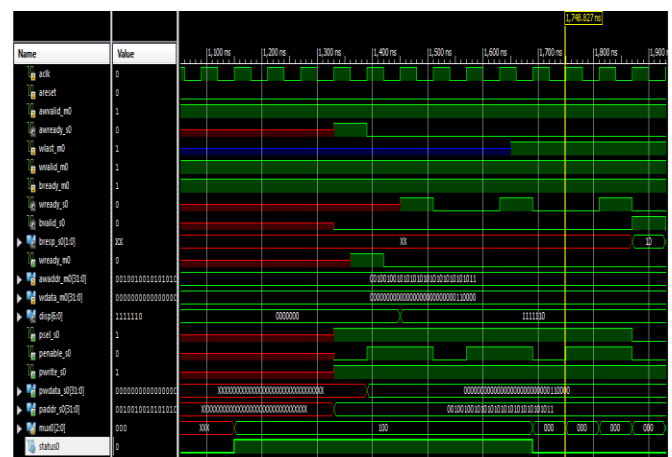


Figure 10: Master0 writing to seven segment display

The selection of the keypad decoder is done by address decoder and required transfer is done. Following figures show the simulation result for the seven segment decoder and UART. Upon synthesis of the design we get the device utilization summaries, which are also mentioned below.

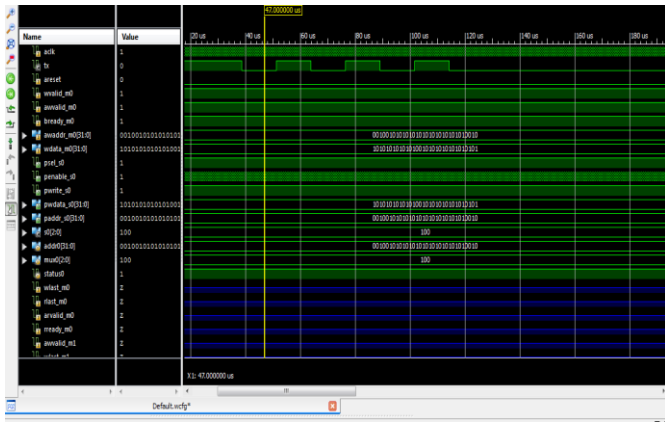


Figure 11: Master0 Transmitting data to UART

5.1 Device Utilization Summary

Table 1 shows the device utilization after the place and route.

Table 1: Slice Logic Utilization

Device resource	Utilized	Total number
Slice register	325	948,480
Slice LUTs	514	474,240
Used as logic	505	474,240
Used as O6	465	
Used as O5 & O6	40	
Exclusive route-thrus	9	
Bonded IOBs	994	1200

5.2 Timing Constraints

Table 2 shows the timing constraints of the design. All the timing constraints were met as indicated by the positive slack.

Table 2: Timing constraints

Met	Constraint	Check	Worst Case Slack	Best Case Achievable	Timing Errors
Yes	Auto time spec constraint for clock net aclk_BUFGP	SETUP HOLD	0.108ns	4.226ns	0

Table 3: clock period

Timing constraint	Default period analysis for Clock 'aclk'
Clock period	2.799ns (frequency: 357.315MHz)
Total number of	3886 / 497

paths / destination ports	
Delay	2.799ns (Levels of Logic = 5)

6. Conclusion:

Designed and implemented SoC bus architecture AMBA (AXI and APB). AXI4 the main system bus and APB4 the peripheral bus was implemented on a four master four slave system. Read and write transactions are done by different masters with different slaves. Whenever a master needs to communicate with a particular slave, it will assert the necessary handshaking signals and gain the access to that slave. The intended communication will then follow. The whole design is simulated and implemented on FPGA.

References

- [1] Milica Mitic and Mile Stojcev, An Overview of On-Chip Buses SER.: ELEC. ENERG. vol. 19, no. 3, December 2006, 405-428.
- [2] Clifford E. Cummings, "Coding And Scripting Techniques for FSM Designs With Synthesis-Optimized, Glitch-Free Outputs," SNUG (Synopsys Users Group Boston, MA 2000) Proceedings, September 2000.
- [3] Flynn, D. Adv. RISC Machines Ltd., Cambridge, "AMBA: enabling reusable on-chip designs", IEEE Micro, Publication Date: Jul/Aug1997.
- [4] AMBA specification (Rev 2.0)
- [5] ARM, "AMBA AXI protocol specifications (Version 2), March 2010", [Online] Available: <http://www.arm.com>.
- [6] AMBA APB Protocol v1.0.
- [7] Chenghai Ma, Zhijun Liu, Xiaoyue Ma, "Design and Implementation of APB Bridge based on AMBA 4.0", IEEE transaction ,2011.
- [8] Peter M Nyasulu, J Knight, Introduction to Verilog, Carleton University Press, Ottawa, 2003.
- [9] Samir Palnitkar, Verilog HDL: A Guide to Digital Design and synthesis, 2nd ed, Prentice Hall PTR Pub, 2003.
- [10] Clifford E. Cummings, "Non blocking assignments in verilog synthesis, coding styles that kill," SNUG (Synopsys Users Group Boston, MA 2000) Proceedings, September 2000.
- [11] Xilinx ISE synthesis and verification design guide.

Author Profile



Muhammed Ashraf T P received B.Tech. Degree from Viswajyothi College of engineering, Muvattupuzha, Kerala, India in 2011. Currently he is pursuing M. Tech in Microelectronics in the

Department of Electronics & Communication Engg, MIT, Manipal.
His area of interest is ASIC Design & Verification.



Manu Ramesh received B.E degree from Vidyaa Vikas College of Engineering and Technology, Thiruchengode, Tamilnadu, India in 2012. Currently he is pursuing M. Tech in Microelectronics in the Department of Electronics & Communication Engg, MIT, Manipal. His area of interest is ASIC design, VLSI design & testing, Verilog coding and Embedded design.



Shankarnarayana Bhat M obtained his M.Tech from IISc, Bangalore and is currently working as Associate professor-Senior in the department of Electronics and communication Engineering, M.I.T Manipal. He is also heading the Education Technology Cell of M.I.T. Manipal. In addition to contributing as resource person for technical workshops and conferences, he is also a recognized faculty trainer and successfully conducted faculty training in various Engineering colleges in India. He has presented technical papers, chaired many national and International conferences and reviewed technical papers for conferences and journals. His interests include Low Power VLSI Design and Processor architecture in addition to Engineering Education and Soft skills.