International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 9, September - 2013

APB Bridge Based on AMBA 4.0

G Prathibha
Department of ECE
Sree Buddha College of Engineering, Pattoor
Pandalam, Kerala

Ambika Sekhar
Department of ECE
Sree Buddha College of Engineering, Pattoor
Pandalam, Kerala

Abstract

Integrated circuits have entered the era of System-on-a-Chip (SoC), which refers to integrating all components of a computer or other electronic system into a single chip. With the increasing design size, IP has become an inevitable choice for SoC design. The widespread use of all kinds of IPs has changed the nature of the design flow, making On-Chip Buses (OCB) essential to the design. Of all OCBs existing in the market, the AMBA bus system is widely used as the de facto standard SoC bus. The latest specification introduced by ARM is AMBA 4.0 specifications. A bus bridge is used to convert transactions from one bus protocol to another. Based on AMBA 4.0 bus, the Intellectual Property (IP) core of Advanced Peripheral Bus (APB) bridge has been designed, which translates the AXI4.0-lite transactions (AXI Master) into APB 4.0 transactions (APB Master). As most of the peripherals don't use the advanced features of AXI4 bus, the APB bus has been implemented to interact with the processor to reduce complexity. The bridge provides an interface between the high-performance AXI bus and low-power APB domain. It has a Slave interface which receives the AXI4 master transactions and converts them to APB master transactions and initiates them on the APB bus. As the APB protocol is significantly simpler than AXI4, the AXI4 transactions are properly downgraded to APB transactions.

1. Introduction

The Advanced Microcontroller Bus Architecture (AMBA) is used as the on-chip bus in system-on-a-chip (SoC) designs. Since its inception, the scope of AMBA has gone far beyond microcontroller devices and is now widely used on a range of ASIC and SoC parts including applications processors used in modern portable mobile devices. AMBA protocol is an open standard, on-chip interconnect specification for the connection and management of functional blocks in a System-on-Chip (SoC). It facilitates right-first-time development of multi-processor designs with large numbers of controllers and peripherals.

AMBA was introduced by ARM Ltd in 1996. The first AMBA buses were Advanced System Bus (ASB) and Advanced Peripheral Bus (APB). In its 2nd version, AMBA 2, ARM added AMBA High-performance Bus (AHB) that is a single clock-edge protocol. In 2003, ARM introduced the 3rd generation, AMBA 3, including AXI to reach even higher performance inter-connects and the Advanced Trace Bus (ATB) as part of the CoreSight on-chip debug and trace solution. These protocols are today the de facto standard for 32-bit embedded processors because they are well documented and can be used without royalties.

The important aspect of a SoC is not only which components or blocks it houses, but also how they are interconnected. AMBA is a solution for the blocks to interface with each other. The objective of the AMBA specification is to:

- 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 macrocells 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.

ARM introduced the Advanced Microcontroller Bus Architecture (AMBA) 4.0 specifications in March 2010, which includes Advanced eXtensible Interface (AXI) 4.0. AMBA bus protocol has become the de facto standard SoC bus. That means more and more existing IPs must be able to communicate with AMBA 4.0 bus. Based on AMBA 4.0 bus, an Intellectual Property (IP) core of AXI (Advanced extensible Interface) Lite to APB (Advanced Peripheral Bus) Bridge can be designed, which would translate the AXI4.0-lite transactions into APB 4.0 transactions. The bridge provides interfaces between the high- performance AXI bus and low-power APB domain.

2. Block Diagram

The design specification mainly deals with the implementation and verification of the APB bridge which converts AXI4 protocol transactions to APB protocol transactions. AXI4 and APB are protocols of the AMBA bus family from ARM for mobile SOCs. The AXI4 forms the main processor communication bus and APB is the primary peripheral bus. As most of the peripherals don't use the advanced features of AXI4 bus they implement the APB bus to interact with the processor to reduce complexity. An RTL which converts AXI4 transactions to APB bus will be helpful in integrating peripherals which use

the simpler APB bus, into an SOC with standard ARM processor/s.

A bus bridge is used to convert transactions from one bus protocol to another. Here the transactions coming from a processor (AXI4 Master) are converted to APB Master transactions. Hence the bus bridge has a Slave interface which receives the AXI4 master transactions and converts them to APB master transactions and initiates them on the APB bus. As the APB protocol is significantly simpler than AXI4, the AXI4 transactions will have to be downgraded to APB transactions. The block diagram of the bus bridge system can be given as shown below.

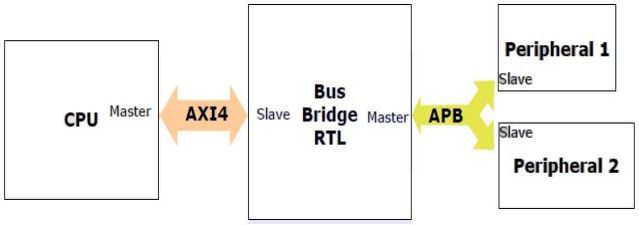

Figure 1: APB Bridge block diagram

3. RTL Block Diagram

A bus bridge is used to convert transactions from one bus protocol to another. Here the transactions coming from a processor (AXI4 Master) has been converted to APB Master transactions. Hence the bus bridge has a Slave interface which receives the AXI4 master transactions and converts them to APB master transactions and initiates them on the APB bus. As the APB protocol is significantly simpler than AXI4, the AXI4 transactions have been properly downgraded to APB transactions.

As the first step, we have the RTL block diagram as show in figure 2. The various signal channels of AXI and the APB signals were studied thoroughly. From this study, a

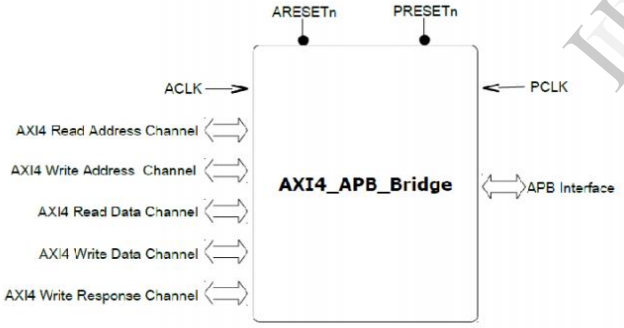

Figure 2: RTL Block Diagram

transaction table was generated, which gives the relation between various AXI and APB signals. It is from this table that we get an idea of how to generate the signals necessary and how they can be used for read and write operations.

4. Waveforms Expected

Depending on the transaction table generated, there read and write operations are expected to be as shown n the figures given below.

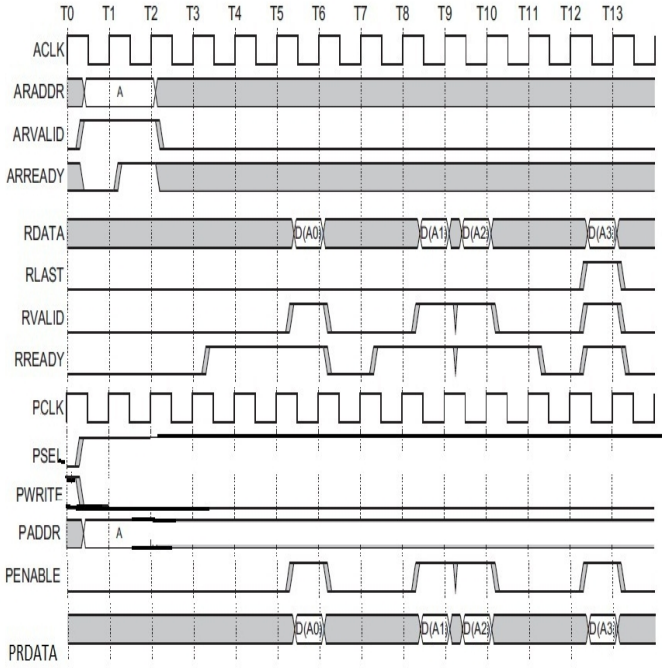

Figure 3: Expected read access waveform

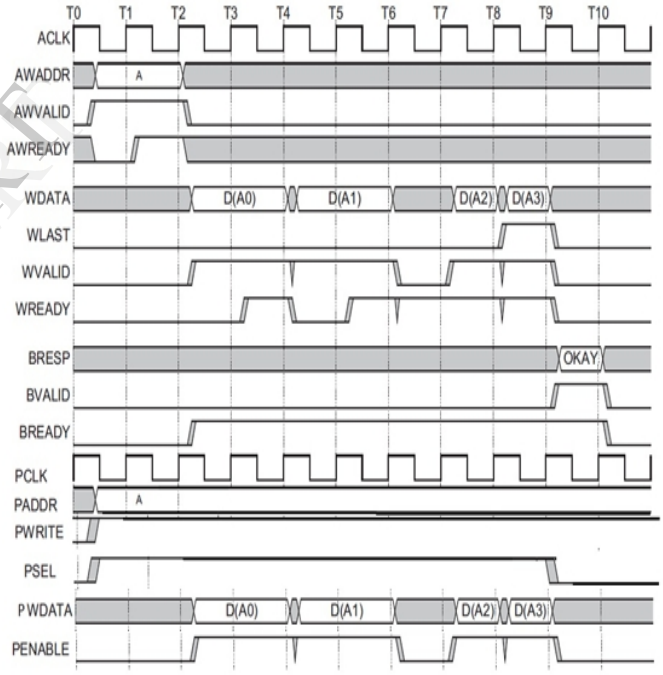

Figure 4: Expected write access waveform

When read/write address is aligned to the read/write size, for 'FIXED' burst AXI4 type, address for all APB reads/writes will remain the same. For 'INCREMENTING' burst AXI4 type, address for APB reads/writes will be increased according to the data bus width on the APB size. For 'WRAPPING' burst AXI4 type, address for APB reads/writes will be increased according to the data bus width on the APB size. It will also have to be wrapped according to the wrap size calculated as given in AXI4

specifications.

When read/write address is unaligned to the read/write size, for 'FIXED' burst AXI4 type, address for all APB reads will remain the same. The size of transfer should not exceed data bus width. If system address is higher than that of master, then higher order bits are filled with zeroes, else they are left unconnected if narrower. For 'INCREMENTING' burst AXI4 type, address for APB reads will be increased according to the data bus width on the APB size. The size of transfer should not exceed data bus width. If system address is higher than that of master, then higher order bits are filled with zeroes, else they are left unconnected if narrower. For 'WRAPPING' burst AXI4 type, address for APB reads will be increased according to the data bus width on the APB size. It will also have to be wrapped according to the wrap size calculated as given in AXI4 specifications. The size of transfer should not exceed data bus width. If system address is higher than that of master, then higher order bits are filled with zeroes, else they are left unconnected if narrower.

For AXI read/write data transactions, as APB is single data exchange bus we will need to wait for each data to come before we can initiate a data beat on the AXI read bus. More complications come in if data bus size is different in AXI4 and APB.

The write response channel signals is for letting the AXI peripherals know about the conditions of transfer. Errors and such related conditions will be made known**.**

5. Simulation and Implementation

5.1 Clock and Reset Block

As in any SoC designs, the main unit is the clock and reset block. The purpose of this block is to generate the necessary clock and reset signals. In the APB bridge, we need two clock signals and two reset signals. ACLK and ARESETn for the AXI part and the PCLK and PRESETn for the APB part. The figure below shows the simulation result for clock and reset block.

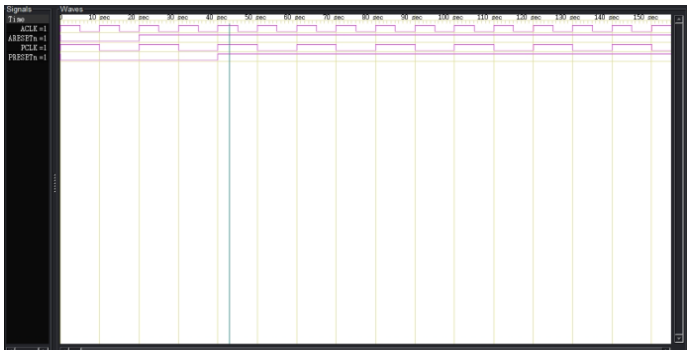

Figure 5: Clock and reset block output

5.2 Transaction Generator Block

The purpose of this block is to generate random values for all the signals related to AXI. For each positive edge of the ACLK, the signals have different values. This is generated by using the keyword $random in the verilog language. Random values are obtained for the signals when this keyword is used. An intermediate signal is generated for all the signals involved in the transactions in this block.

The signal values depend on the reset value. As the reset is an active low signal, when its value s 0, all the values for the signals will be 0, else when the reset value is 1, random value is generated for each clock cycle.

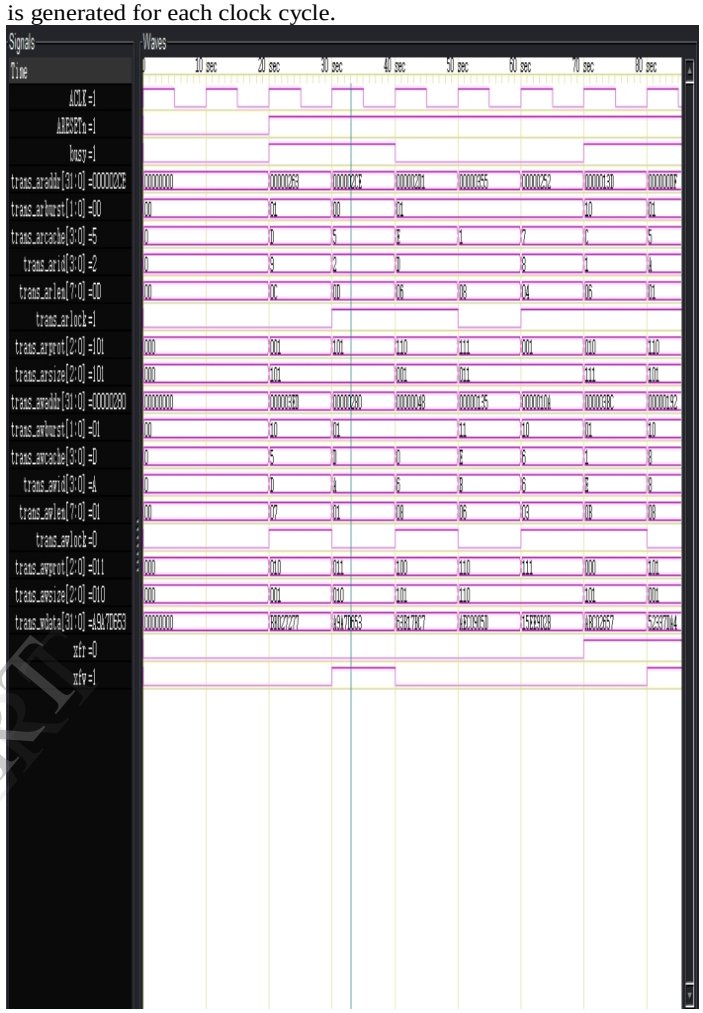

Figure 6: Transaction generator block output

5.3 APB Slave Block

The APB slave block contains all the APB signals. The function of this block is to check whether the function that is to be performed by the slave is a read one or a write one. This depends on the value of the PWRITE signal. If the value on that signal is HGH, it's a write process or else it's a read process, provided PSEL, PREADY and PENABLE are HIGH. When it's a read operation, the value in the memory location is read by the PRDATA signal. When it's a write operation, the value on the PWDATA is written on to the memory location.

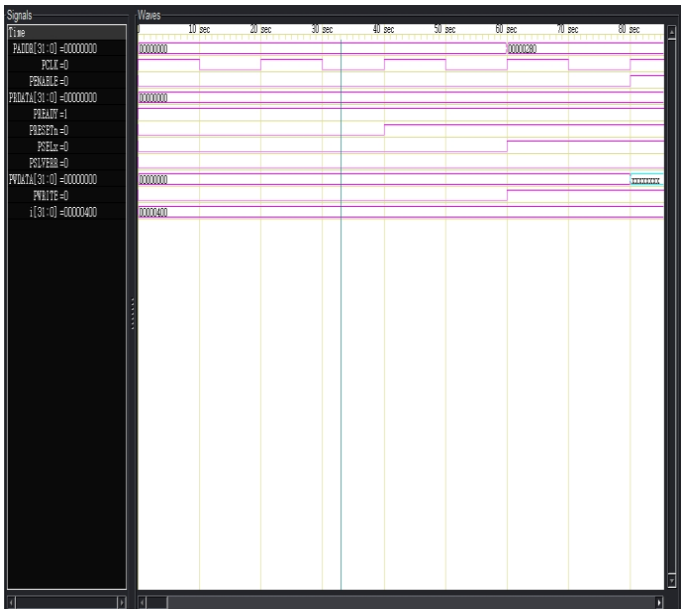

Figure 7: APB Slave block output

5.4 AXI Master Block

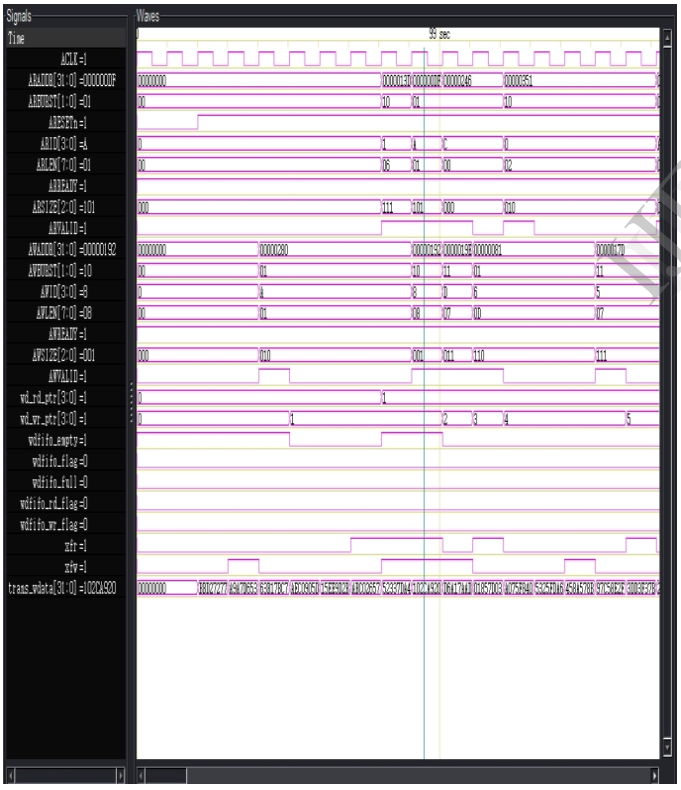

Figure 8: AXI master block output

The master block performs the operations for the AXI4 Master Read Address section, AXI4 Master Write Address section and AXI4 Master Write Data section. In the read address section, the read address is generated. The states present in the read/write address section are IDLE and VALID, whereas for the read/write sections are IDLE and READY.

In the AXI4 Master Read/Write Address section, depending on the condition of ARESETn, ARREADY/AWREADY and signal xfr/xfw from the transaction generator, the states shift between IDLE and VALID. The necessary signals values depends on ARESETn condition and assumes values from transaction generator.

The AXI4 Master Write Data section uses FIFO to store and read values. There are flags to indicate if FIFO is full or empty and to know whether it's read or write. The address and length are stored in the FIFO. The write pointer is incremented after the write address. In case of read operation, address is incremented after one batch of data is completely accessed. The read pointer is reverted back to 0 after reaching maximum count and WLAST is asserted at the end of data transfer. The output of this section is shown in the figure 8.

5.5 Bridge Block

The bridge block consists of the AXI and the APB signals. The APB part works on the basis of the state diagram shown in figure 9.

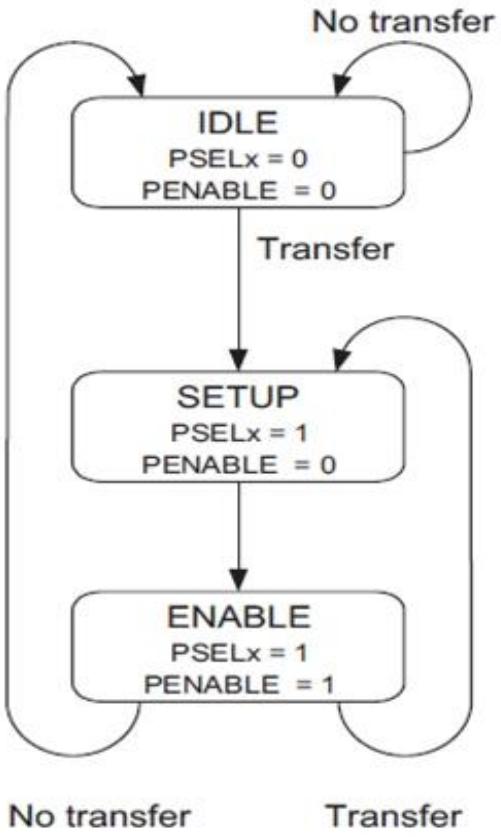

Figure 9: APB State diagram

The IDLE state is the default state for the peripheral bus. When a transfer is required the bus moves into the SETUP state, where the appropriate select signal, **PSELx**, is asserted. The bus

only remains in the SETUP state for one clock cycle and will always move to the ENABLE state on the next rising edge of the clock. In the ENABLE state the enable signal, **PENABLE** is asserted. The address, write and select signals all remain stable during the transition from the SETUP to ENABLE state. The ENABLE state also only lasts for a single clock cycle and after this state the bus will return to the IDLE state if no further transfers are required. Alternatively, if another transfer is to follow then the bus will move directly to the SETUP state. It is acceptable for the address, write and select signals to glitch during a transition from the ENABLE to SETUP states.

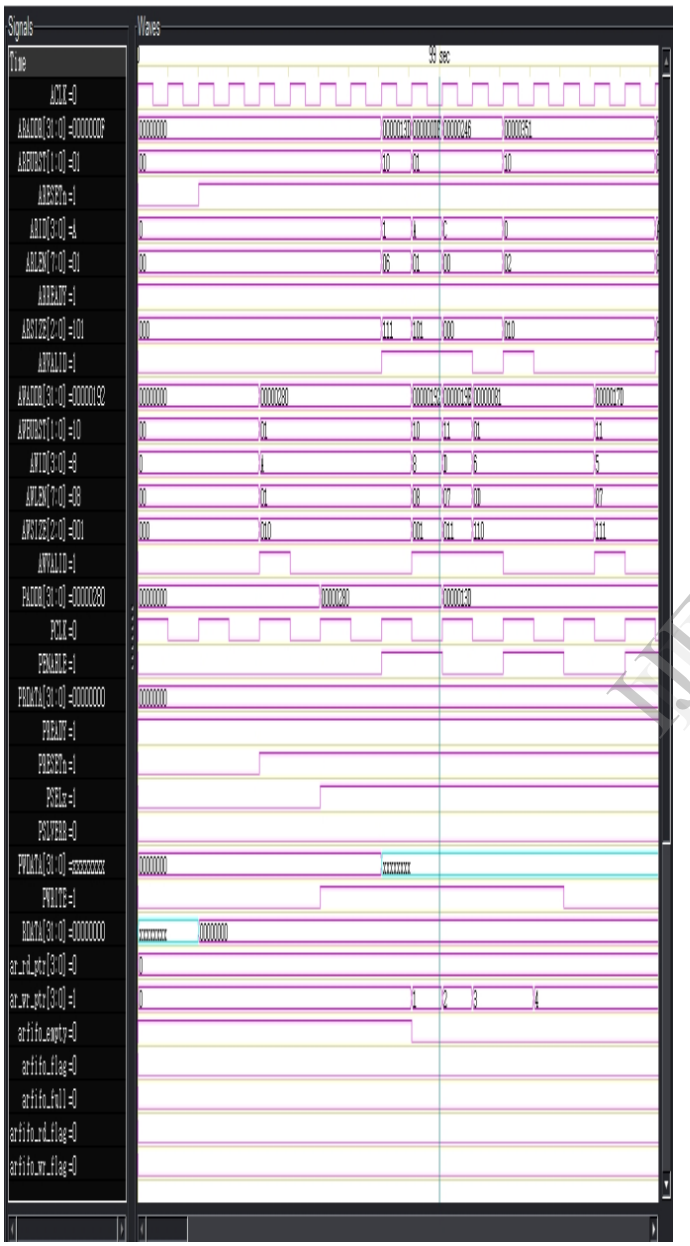

Figure 10(a): Bridge block output

For the AXI, we have the storing read/write address section and read/write data section. In the AXI read/write address storing sections, the FIFO flags and pointers are initialized. Upon reset, all the values are zero. Otherwise, depending on the APB signals, the pointer and flag conditions are altered. The read/write address storing section is for the AXI master unit.

In the AXI4 Slave Read Data storing section, bridge takes data from APB slave and initiates read transaction on AXI4 slave interface. The write and read data signals of AXI work on the condition of the flags and pointers.

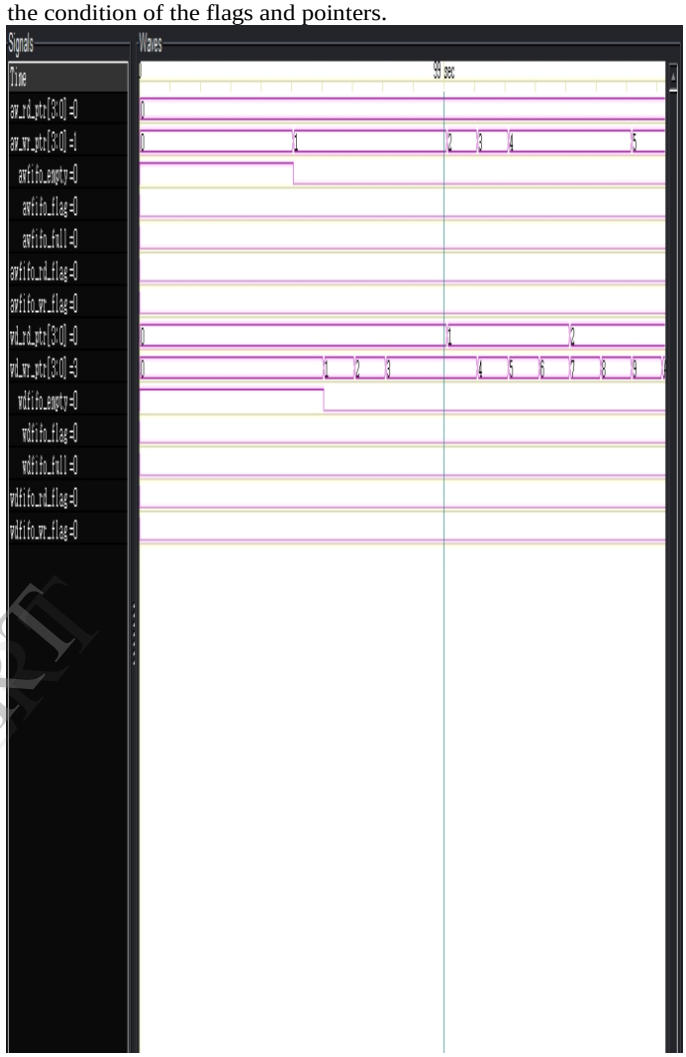

Figure 10(b): Bridge block output

The entire code for the clock and reset block, and for the block diagram part was done in Verilog. The simulation software used is the Icarus Verilog. For the complete implementation on the FPGA board, Xilinx ISE Design suite 13.2 has been used. The board used is the Digilent Basys board, which has been intefaced to the PC using the Adept applicaton of Digilent.

5.6 Complete Simulation

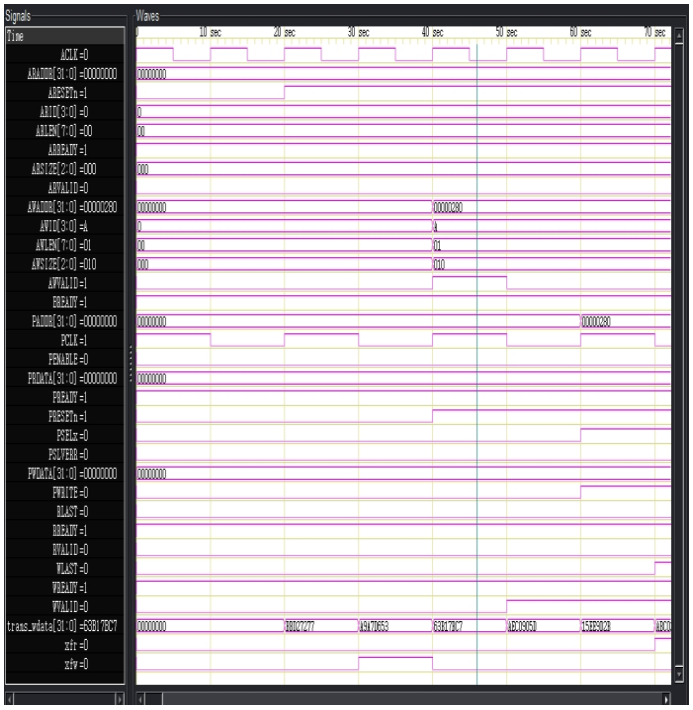

Figure 11: Complete output

The device utilization summary is given in the table below.

Table 1: Device Utilization Summary

Logic Utilization	Used	Available	Utilization
Number of Slices	948	960	98%
Number of Slice Flip Flops	928	1920	48%
Number of 4 input LUTs	1469	1920	76%
Number of IOs	18		
Number of bonded IOBs	18	83	21%
Number of GCLKs	3	24	12%

6. Conclusion

The APB bridge that has been designed is used for reducing the processor complexity as most of the peripherals don't use the advanced features of AXI 4 bus. AXI 4 being the ARM's latest protocol has lots of new features which are not used by most of the peripherals. Thus, the transactions from AXI master has to be converted to APB master transactions for the further processing.

The implemented bridge generates the necessary AXI signals an all the APB signals. The clock and reset modules, AXI master module and the APB slave module have together been incorporated along with the bridge module to for the complete implementation. Conditions for checking errors have also been implemented in the bridge. The language used for coding is VERILOG. The simulation software is Icarus Verilog.

7. References

[1] ARM, "AMBA Specification Rev 2.0", www.arm.com

[2] ARM, "AMBA APB Protocol Specification Version 2.0", www.arm.com

[3] Guoliang Ma, et. al., Inst. of Microelectron., Tsinghua Univ., Beijing, China, "Design and Implementation of an Advanced DMA Controller on AMBA-Based SoC", IEEE International Conference on ASIC, 2009, pp 419 - 422.

[4] Hyun-min Kyung,et. al., Processor Architecture Lab, SOC R&D Center, System LSI Division, Semiconductor Business, Samsung Electronics, Yongin-City, Kyeong-gi Do, Republic of Korea, "Design and implementation of Performance Analysis Unit (PAU) for AXI-based multi-core System on Chip (SOC)", Journal in Microprocessors and Microsystems, 2010,pp 102–116

[5] Shaila S Math, Manjula R. B, S.S. Manvi , Dept. of Electronics and Communication Engineering, REVA Institute of Technology and Management, Bangalore, India "Data Transactions on System-on-Chip Bus Using AXI4 Protocol", International Conference on Recent Advancements in Electrical, Electronics and Control Engineering, 2011, pp 423-427.

[6] Sangik Choi and Shinwook Kang Mobile Convergence Team, Digital Media R&D Center, Samsung Electronics Co., Ltd., Suwon-city, Kyunggi-do, Republic of Korea, "Implementation of an On-Chip Bus Bridge between Heterogeneous Buses with Different Clock Frequencies", Proceedings in Fifth International Workshop on System-on-Chip for Real-Time Applications, 2005, pp 530-534.

[7] Sergio Saponara, Tommaso Cecchini, Francesco Sechi, Luca Fanucci, Department of Information Engineering, University of Pisa, Pisa, Italy, "Pin-limited Frequency Converter IP Bridge for Efficient Communication of Automotive IC Sensors with Off-chip ECUs", IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2009,167-171.

[8] Darshana Dongre, Anil Kumar Sahu, Dept. of ECE, Shri. Shankaracharya college of Engineering & Technology, Bhilai, India, "Implementation of AXI Design Core with DDR3 Memory Controller for SoC", IJCST, Vol. 2, Issue 4, Oct.- Dec. 2011,pp 452-455.

[9] Khan, A.M., et. al., Aoste Project, INRIA Sophia-Antipolis Mediterranee, Sophia-Antipolis, France, "IP-XACT components with Abstract Time Characterization", IEEE Conference on Specification & Design Languages, 2009, pp 1-6.

[10] Shrivastav, A. , et. al., Dept. of Electron. & Comm., Priyatam Inst. of Technol. & Manage., Indore, India, "Performance Comparison of AMBA Bus-Based System-On-Chip Communication Protocol", International Conference on Communication Systems and Network Technologies, 2011,pp 449-454.

[11] Muhlbach,S., et. al., Hamburg Univ. Of Technol., Hamburg, "Secure And Authenticated Communication In Chip-Level Microcomputer Bus Systems With Tree Parity Machines", International Conference On Embedded Computer Systems: Architectures, Modeling And Simulation, 2007, Pp 201-208.

[12] Vijay D'silva, S. Ramesh, Indian Institute of Technology Bombay, "Synchronous Protocol Automata: A Framework for Modelling and Verification of SoC Communication Architectures", Proceedings in Design, Automation and Test in Europe Conference and Exhibition, 2004. , pp 390-395.

[13] Chenghai Ma et.al., , University of Bologna, Forlì, Italy, "Design and Implementation of APB Bridge based on AMBA 4.0", in Satellite Navigation Technologies and European Workshop on GNSS Signals and Signal Processing (NAVITEC), 2010 5th
ESA Workshop, pp 1-7.

[14] GaoMing Du, DuoLi Zhang, et. al. , VLSI Research institute,China, "FPGA Prototype Design of Network on Chips", 2nd International Conference on Anti-counterfeiting, Security and Identification, 2008 ,pp 348-351.

[15] Mohammed H. Alser, Maher M. Assaad, Electrical and Electronics Engineering,
Universiti Teknologi PETRONAS, Perak Darul Ridzuan, Malaysia, "Design and Modeling of Low-Power Clockless Serial Link for Data Communication Systems", National Postgraduate Conference (NPC), 2011,pp 1-5.

[16] Jovanovic, S., Lab. d'Instrum. et d'Electron. de Nancy, Univ. Henri Poincare,Nancy, "A new high-performance scalable dynamic interconnection for FPGA based reconfigurable systems", International Conference on Application-Specific Systems, Architectures and Processors, 2008,pp 61 - 66 .

[17] M. H. Neishaburi, Zeljko Zilic , McGill University, Department of Electrical Engg., Montreal, Quebec Canada , "Debug Aware AXI-based Network Interface", IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, 2011.

[18] Fei Hong, Aviral Shrivastava, and Jongeun Lee, "Return Data Interleaving for Multi-Channel Embedded CMPs Systems", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 20, No. 7, July 2012, pp 1351-1354.