OPEN ACCESS
ISSN: 2249-6645

International
Journal
Of Modern Engineering Research (IJMER)

IP Core Design of APB Bridge Using AMBA4.0

S.Rama Kishore Reddy[1] , K.Prabhakar Reddy[2] , B.Papachary [3]
**(ECE, CMR Engineering College, Hyderabad) Email: ramki430@gmauil.com*
*** (ECE, GIST, Nellore) Email:kpreddy1987@gmail.com*
**** (ECE, CMR Engineering College, Hyderabad) Email: biroju.chary@gmail.com*

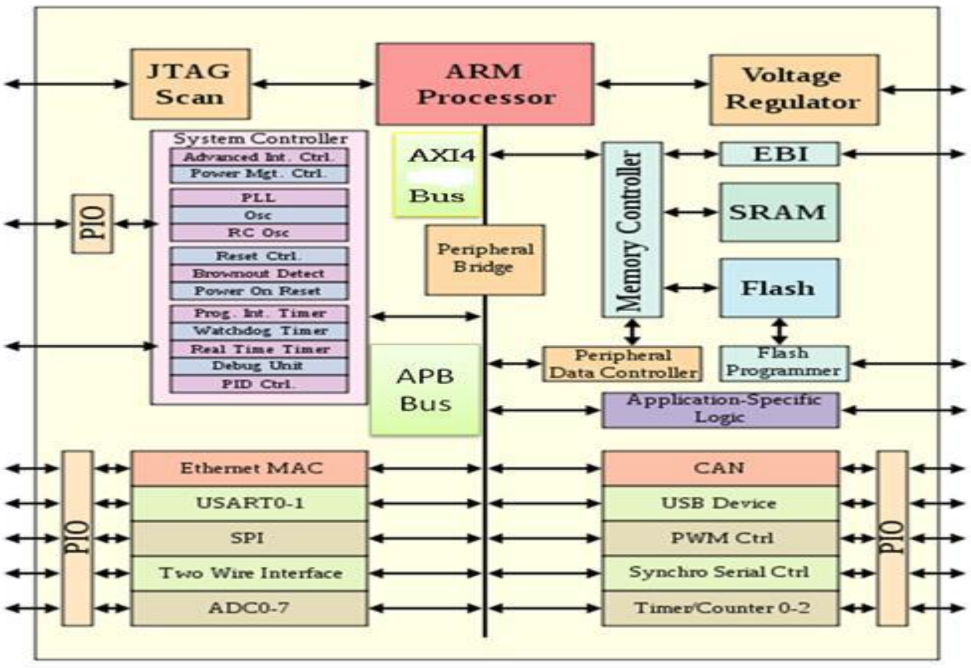
ABSTRACT: Integrated circuits have entered the era of system-on-a-chip (SoC), which refers to integrating all components of a computer or other electronic system into a single chip. It may contain digital, analog, mixed signal, and often radio-frequency functions- all on asingle chip substrate. With the increasing design size, IP is an inevitable choice for SoC design. And 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.ARM introduced the advanced microcontroller bus architecture (AMBA) 4.0specifications 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 AMBA4.0 bus, we design an intellectual property (IP) core of advanced peripheral bus (APB) bridge, which translates the AXI4.0 transactions in into APB 4.0 transactions. The bridge provides an interface between the high performance AXI bus and low power APB domain. With the help of the interface we can transfer address and data with out of order execution by burst transactions. Finally with this design the system will be more efficient and faster.

Keywords : *system-on-a-chip (SoC), on chip buses (OCB), advanced microcontroller bus architecture (AMBA), advanced extensible interface (AXI),advanced peripheral bus (APB) bridge ,*

I. INTRODUCTION

There are many companies that develop core IP for SoC products. The interfaces to these cores can differ from company to company and can sometimes be proprietary in nature. The SoC developer then must expend time, effort, and money to create “bridge” or “glue” logic that allows all of the cores inside the SoC to communicate properly with each other. SoC integrated circuits envisioned by this subcommittee span a wide breadth of applications, target system costs, and levels of performance and integration. 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. It may contain digital, analog, mixed-signal, and often radio-frequency functions – all on a single chip substrate. With the increasing design size, IP is an inevitable choice for SoC design. And 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. On March 8, 2010, ARM announced availability of the AMBA 4.0 specifications. As the de facto standard SoC bus, AMBA bus is widely used in the high-performance SoC designs. The AMBA specification defines an on-chip communication standard for designing high-performance embedded microcontrollers. 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, This design is an Intellectual Property (IP) core of AXI to APB Bridge, which translates the AXI4.0 transactions into APB 4.0 transactions. The bridge provides interfaces between the high-performance AXI bus and low-power APB domain.

II. STRUCTURE OF SOC



A typical SoC consists of:
1. A microcontroller, microprocessor or DSP core(s). Some SoCs— called multiprocessor system on chip (MPSoC)—include more than one processor core.
2. Memory blocks including a selection of ROM, RAM, EEPROM and flash memory.
3. Timing sources including oscillators and phase-locked loops.
4. Peripherals including counter-timers, real-time timers and power-on reset generators.
5. External interfaces including industry standards such as USB, FireWire, Ethernet, USART, SPI.
6. Analog interfaces including ADCs and DACs.
7. Voltage regulators and power management circuits.

These blocks are connected by either a proprietary or industry standard bus such
as the AMBA bus from ARM Holdings. DMA controllers route data directly between
external interfaces and memory, bypassing the processor core and thereby increasing the data throughput of the SoC.

HIGH PERFORMANCE BUSES
The AMBA family of buses was chosen for consideration because of their widespread acceptance in the industry and large amount of existing IP cores.
High-Performance Buses
a. ASB (Advanced System Bus)
b. AHB (Advanced High-performance Bus)
c. AXI (Advanced eXtensible Interface)
- **AXI4**
- AXI4-Lite
- AXI4-Stream

a. ASB (Advanced System Bus)
The Advanced System Bus (ASB) specification defines a high-performance bus
that can be used in the design of high performance 16 and 32-bit embedded microcontrollers. AMBA ASB supports the efficient connection of processors, on-chip
memories and off-chip external memory interfaces with low-power peripheral microcell functions. The bus also provides the test infrastructure for modular macro cell test and diagnostic access. The features of ASB are;
- High performance
- Pipelined operation
- Burst transfers

- Multiple bus masters

b. AHB (Advanced High-performance Bus)
AHB is the mid-performance bus in the AMBA family. The protocol defines a 32-bit address bus (HADDR), but this has been extended in some implementations. The read and write data buses (HRDATA and HWDATA) may be defined under the specification as 2n bits wide, from 8-bit to 1024-bit, but the most common implementation has been 32-bit. With 27 additional control signals, and assuming the most common address and data bus width of 32-bit, AHB can have up to 123 I/O for each AHB master. Up to 16 AHB masters may be connected to a central interconnect which
arbitrates between master requests, and multiplexes the winning request and corresponding transfer qualifiers to the slaves. Slave read data is multiplexed back to the masters. In addition to previous release, it has the following features
Single edge clock protocol.

- Split transactions.
- Several bus masters.
- Pipelined operations.
- Large bus-widths (64/128 bit).

A simple transaction on the AHB consists of an address phase and a subsequent data phase (without wait states: only two bus-cycles). Access to the target device is controlled through a MUX (non-tristate), thereby admitting bus-access to one bus-master at a time.

c. AXI (Advanced eXtensible Interface)
AXI is the high-performance bus in the AMBA family. The architecture defines three write channels and two read channels. The write channels are address, write data, and response. The read channels are address and read data. The address channels include
32-bit address buses, AWADDR and ARADDR, but this could be extended in some implementations. The write and read data buses (WDATA and RDATA) may be defined under the specification as any 2n number, from 8-bit to 1024-bit.AXI masters and slaves are connected together through a central interconnect, which routes master requests and write data to the proper slave, and returning read data to the requesting master. AXI uses a handshake between VALID and READY signals. VALID is driven by the source, and READY is driven by the destination. Transfer of information, either address and control or data, occurs when both VALID and READY are sampled high.
AXI, the third generation of AMBA interface defined in the AMBA 3 specification, is targeted at high performance, high clock frequency system designs and includes features which make it very suitable for high speed sub-micrometer interconnect**.**

- separate address/control and data phases
- support for unaligned data transfers using byte strobes
- burst based transactions with only start address issued
- issuing of multiple outstanding addresses
- Easy addition of register stages to provide timing closure

The AXI Revisions (Advanced eXtensible Interface)
a. AXI4
b. AXI4-Lite
c. AXI4-Stream

DIFFERENT TYPES OF BRIDGES
a. ASB to APB Bridge
b. AHB to APB Bridge
c. AXI4 to APB Bridge
Out of these AXI4 to APB Bridge is used for this project.
a. ASB to APB Bridge
The APB Bridge provides an interface between the ASB and the Advanced Peripheral Bus (APB). It continues the pipelining of the ASB by inserting wait cycles on the ASB only when they are needed. It inserts them for burst transfers or read transfers when the ASB must wait for the APB**.**
b. AHB to APB Bridge

The AHB2APB interfaces the AHB to the APB. It buffers address, control and data from the AHB, drives the APB peripherals and returns data and response signals to the AHB. It decodes the address using an internal address map to select the peripheral.
The AHB2APBis designed to operate when the APB and AHB clocks have the same frequency and phase.
In addition to supporting all basic requirements for AMBA compliance, the
AHB2APB provides enhancements to the APB protocol. With AHB2APB, an APB
peripheral can insert wait states by holding its pready signal low for the required number of cycles.
In the AMBA specification, APB accesses are word-wide (32 bits). The AHB2APB provides the signal pbyte_enable [3:0] to allow byte and half-word accesses. These can be used by the APB peripheral as necessary. The AHB2APB does not perform any alignment of the data, but transfers data from the AHB to the APB for write cycles and from the APB to the AHB for read cycles. The AHB2APB does not support burst transfers and converts any burst transfers into a series of APB accesses. The AHB slave interface supplied by the AHB2APB does not make use of the split response protocol.

c. AXI4 to APB Bridge
In this study, we focused mainly on the implementation aspect of an AXI4 to APB Bridge. The AXI4 to APB Bridge provides an interface between the high performance AXI domain and the low power APB domain. Read and write transfers on the AXI bus are converted into corresponding transfers on the APB

Features of bridge
The Xilinx AXI to APB Bridge is a soft IP core with these features:
a. AXI interface is based on the AXI4 specification.
b. Supports 1:1 (AXI:APB) synchronous clock ratio.
c. Connects as a 32-bit slave on 32-bit AXI4.
d. Connects as a 32-bit master on 32-bit APB3/APB4
e. Supports optional data phase time out

III. FUNCTIONAL DESCRIPTION AXI4 TO APB BRIDGE

The AXI to APB Bridge translates AXI4 transactions into APB transactions. The bridge functions as a slave on the AXI4 interface and as a master on the APB interface. The AXI to APB
Bridge main use model is to connect the APB slaves with AXI masters. Both AXI4 and APB transactions are happened during rising edge of the clock. The AXI to APB Bridge block diagram is shown in below figure and described in subsequent sections.

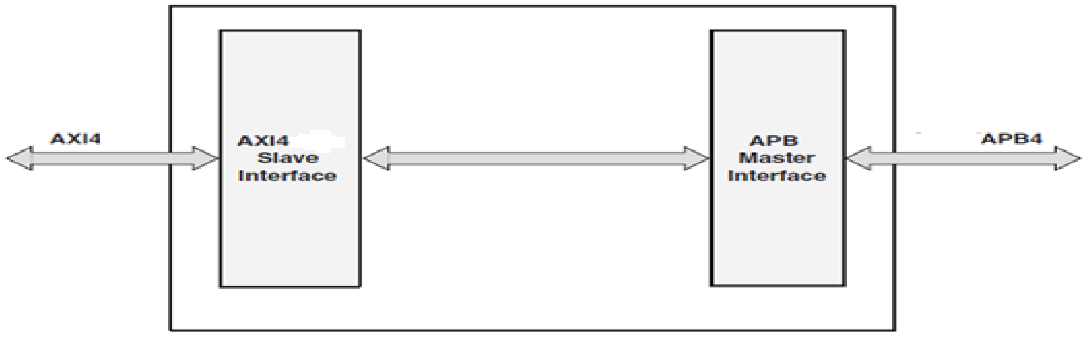


Fig AXI4 to APB4 Block Diagram

AXI4 Slave Interface:
The AXI4 Slave Interface module provides a bi-directional slave interface to the AXI. The AXI address and data bus widths are always fixed to 32-bits and 1024 bits. When both write and read transfers are simultaneously requested on AXI4-Lite, thread request is given more priority than the write request

APB Master Interface:
The APB Master module provides the APB master interface on the APB. This interface can be APB3 or APB4, which can be selected by setting the generic C_M_APB_PROTOCOL.When C_M_APB_PROTOCOL=apb4, the M_APB_PSTRB,
and M_APB_PPROT signals are driven at the APB Interface. The APB address and data bus widths are fixed to32-bits.

Burst size & Burst type tables shown below:
The maximum number of bytes to transfer in each data transfer, or beat, in a burst, is specified by:
ARSIZE [2:0], for read transfers.
AWSIZE [2:0], for write transfers.

AxSIZE[2:0]	Bytes in Transfer
0b000	1
0b001	2
0b010	4
0b011	8
0b100	16
0b101	32
0b110	64
0b111	128

Table: Burst size encoding

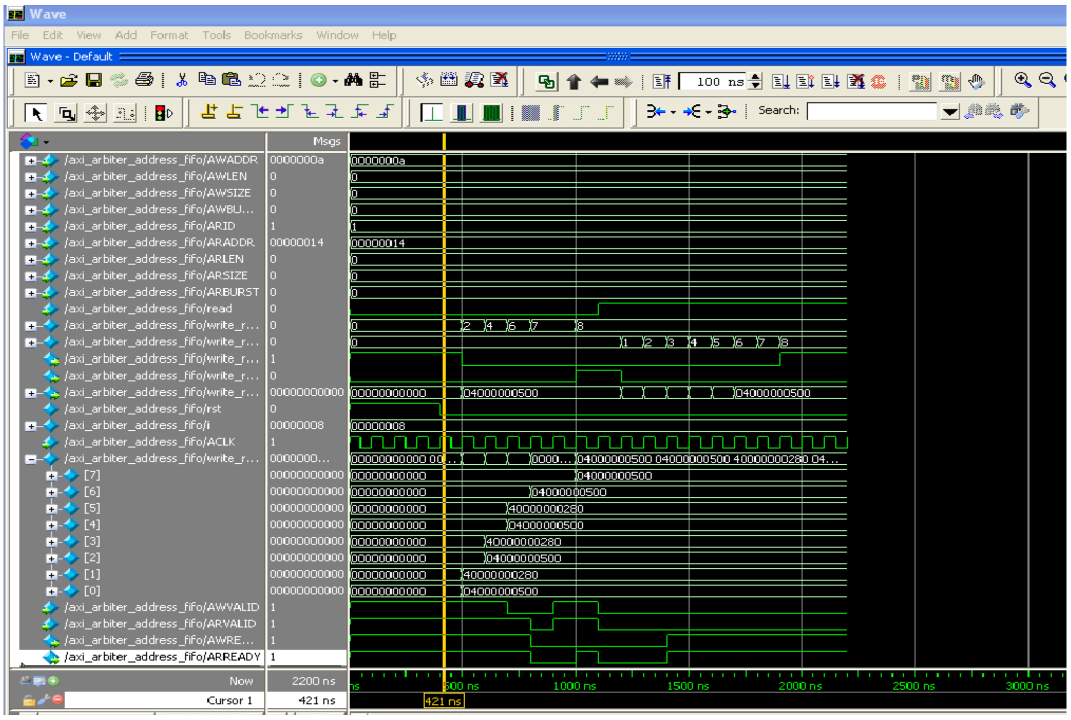
The burst type is specified by:
ARBURST [1:0], for read transfers.
AWBURST [1:0], for write transfer
In this specification, **AxBURST** indicates **ARBURST** or **AWBURST**

AxBURST[1:0]	Burst type
0b00	FIXED
0b01	INCR
0b10	WRAP
0b11	RESERVED

Table: Burst type encoding

IV. RESULTS

The below figure shows the simulation of address FIFO in which the transfer of address from AXI is shown under several conditions of FIFO full and empty. With the
help of this simulation we can see how the address is transferred in out of order execution with burst transactions



	Msgs
/axi_arbiter_address_fifo/AWADDR	0000000a
/axi_arbiter_address_fifo/AWLEN	0
/axi_arbiter_address_fifo/AWSIZE	0
/axi_arbiter_address_fifo/AWBU...	0
/axi_arbiter_address_fifo/ARID	1
/axi_arbiter_address_fifo/ARADDR	00000014
/axi_arbiter_address_fifo/ARLEN	0
/axi_arbiter_address_fifo/ARSIZE	0
/axi_arbiter_address_fifo/ARBURST	0
/axi_arbiter_address_fifo/read	0
/axi_arbiter_address_fifo/write_r...	0
/axi_arbiter_address_fifo/write_r...	0
/axi_arbiter_address_fifo/write_r...	1
/axi_arbiter_address_fifo/write_r...	0
/axi_arbiter_address_fifo/write_r...	00000000000
/axi_arbiter_address_fifo/rst	0
/axi_arbiter_address_fifo/i	00000008
/axi_arbiter_address_fifo/ACLK	1
/axi_arbiter_address_fifo/write_r...	0000000...
[7]	00000000000
[6]	00000000000
[5]	00000000000
[4]	00000000000
[3]	00000000000
[2]	00000000000
[1]	00000000000
[0]	00000000000
/axi_arbiter_address_fifo/AWVALID	1
/axi_arbiter_address_fifo/ARVALID	1
/axi_arbiter_address_fifo/AWRE...	1
/axi_arbiter_address_fifo/ARREADY	1
Now	2200 ns
Cursor 1	421 ns

Fig: Address FIFO

The below figure shows the simulation of data FIFO in which the transfer of data from AXI is shown under several conditions of FIFO full and empty. With the help of this simulation we can see how the data is transferred in out of order execution with burst transactions and is finally matched with respective address with the help of ID pin.

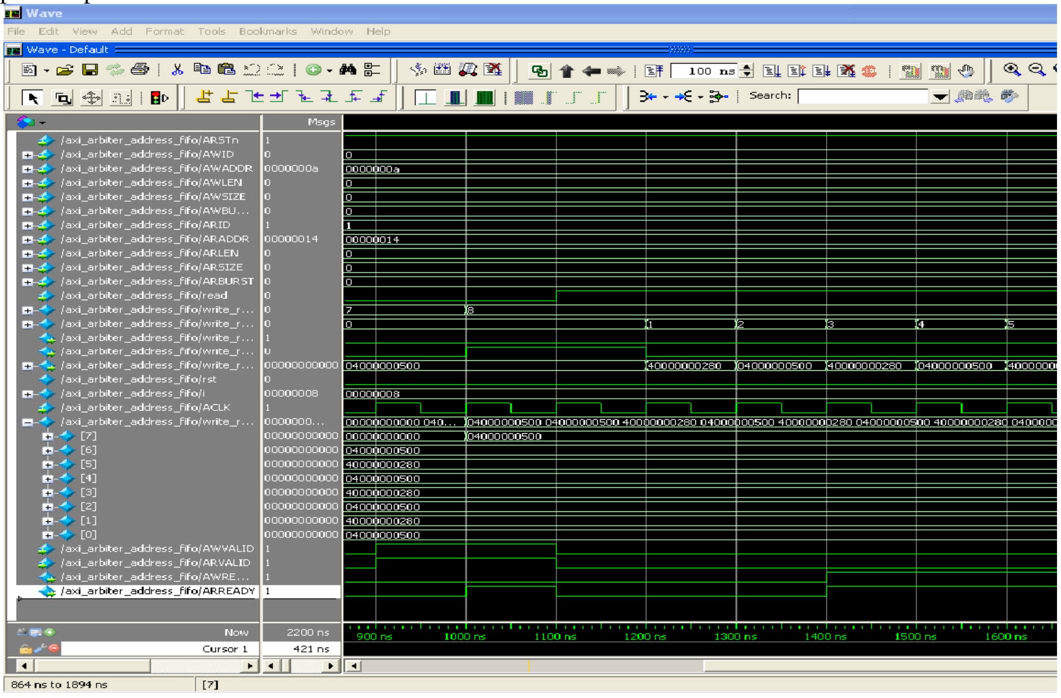

Wave
Wave - Default
Msgs
/axi_arbiter_address_fifo/ARSTn
/axi_arbiter_address_fifo/AWID
/axi_arbiter_address_fifo/AWADDR
/axi_arbiter_address_fifo/AWLEN
/axi_arbiter_address_fifo/AWSIZE
/axi_arbiter_address_fifo/ARID
/axi_arbiter_address_fifo/ARADDR
/axi_arbiter_address_fifo/ARLEN
/axi_arbiter_address_fifo/ARSIZE
/axi_arbiter_address_fifo/ARBURST
/axi_arbiter_address_fifo/read
/axi_arbiter_address_fifo/rst
/axi_arbiter_address_fifo/i
/axi_arbiter_address_fifo/ACLK
/axi_arbiter_address_fifo/AWVALID
/axi_arbiter_address_fifo/ARVALID
/axi_arbiter_address_fifo/ARREADY
Now 2200 ns
Cursor 1 421 ns
864 ns to 1894 ns

Fig:Data FIFO

The final outcome simulation result shows us how the complete address and data are transferred with out of order execution and burst transactions from AXI to APB

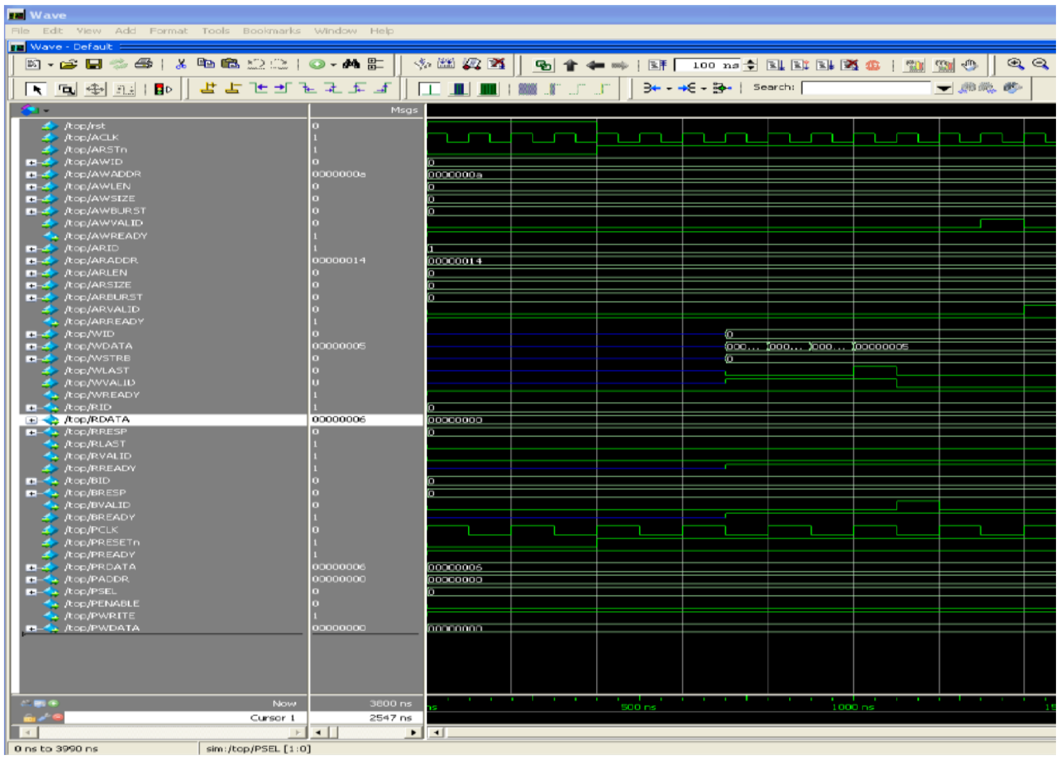


Fig: Total outcome of the design

V. CONCLUSION

We conclude by providing an implementation of AXI4 to APB Bridge using 32 bit AXI slave and APB master interfaces. Here AXI master can access up to 16 APB peripherals where the APB clock is independent of AXI clock. With this design we will be able to transfer the data efficiently making system performance better and faster.

REFERENCES

[1] *Design and Implementation of APB Bridge based on AMBA 4.0 (IEEE 2011),* ARM Limited.

[2] ARM, "AMBA Protocol Specification 4.0", *www.arm.com, 2010*

[3] Ying-Ze Liao, *"System Design and Implementation of AXI Bus"*, National Chiao Tung University, October 2007.

[4] Clifford E. Cummings, "Coding And Scripting Techniques For FSMDesigns With Synthesis-Optimized, Glitch-Free Outputs," *SNUG(Synopsys Users Group Boston, MA 2000)* Proceedings, September 2000.

[5] Clifford E. Cummings, "Synthesis and Scripting Techniques for Designing Multi-Asynchronous Clock Designs," *SNUG 2001*

[6] Lahir, K., Raghunathan A., Lakshminarayana G., "LOTTERYBUS: a new high performance communication architecture for system-on-chip deisgns," *in Proceedings of Design Automation Conference, 2001*

[7] Sanghun Lee, Chanho Lee, Hyuk-Jae Lee, "A new multi-channel onchip-bus architecture for system-on-chips," *in Proceedings of IEEE International SOC Conference, September 2004*

S.RAMA KISHORE REDDY having **9+** years of teaching experience and completed hi B.TECH degree in Electronics and Communication Engineering. He received his M.Tech in Embedded Systems. His interesting area is Embedded Systems & Signal Processing.

K.Prabhakar Reddy having **6** years of teaching experience and completed his B.TECH degree in Electronics and Communication Engineering. He received his M.Tech in VLSID. His interesting area is VLSI & Signal Processing.

Biroju Papachary received the B.Tech degree in Electronics and Communication Engineering from JNTU Hyderabad and M.Tech degree in Computers and Communication Engineering from JNTU Hyderabad. He was published 4 International journals. He has 9 years of teaching experience. His current research includes, computer networks, embedded systems and wireless sensor networks. Now he was working as Associate professor in CMR Engineering College, Hyderabad, Telangana state, India.