IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
ISSN: 2278-2834, ISBN: 2278-8735. Volume 4, Issue 2 (Nov. - Dec. 2012), PP 01-06
www.iosrjournals.org

Design and Implementation Based On Amba4.0 Of Apb Bridge

G.Madhuri[1], P. Sharmila Rani[2], Jeevan Reddy K[3], Vasuja Devi Midasala[4]
[2,3] Associate Professor, 4Assistant Professor,
[1,2,3,4] Dept. of ECE, TeegalaKrishnaReddy Engineering College, Andhra Pradesh, India

Abstract: *ARM introduced the Advanced Microcontroller Bus Architecture (AMBA) 4.0 specifications which includes Advanced eXtensiable 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 AMBA4.0 bus. Based on AMBA 4.0 bus, we designed an Intellectual Property (IP) core of Advanced Peripheral Bus (APB) Bridge, which translates the AXI4.0-lite transactions into APB 4.0 transactions. The bridge provides an interfaces between the high-performance AXI bus and low-power APB domain.*
Keywords: *SoC, AMBA, AXI, APB*

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. Incompatible interfaces are thus barriers to both IP developers and SoC developers.

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..

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(Advanced extensible Interface) Lite to APB(Advanced Peripheral Bus) Bridge, which translates 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**.**

II. Literature Review

Advanced Microcontroller Bus Architecture (AMBA) specification defines an on chip communications standard for designing high-performance embedded microcontrollers. AMBA has 4 versions as follows

1. VER1.0 (ASB & APB)
2. VER 2.0 (AHB)
3. VER 3.0 (AXI, ATB)
4. VER 4.0 (AXI 4,AXI LITE,AXI STREAM (AXI))

AMBA 4.0 specification buses/interfaces

1. Advanced eXtensible Interface (AXI)
2. Advanced High-performance Bus (AHB)
3. Advanced System Bus (ASB)
4. Advanced Peripheral Bus (APB)
5. Advanced Trace Bus (ATB)

In this project these are to be used Advanced eXtensible Interface (AXI4-Lite) & Advanced Peripheral Bus (APB) because these are high bandwidth data transfer between high performance devices like processor, DMA, RAM etc..,. and Peripheral devices.

III. Top View

3.1. Block Diagram

The AXI4-Lite to APB Bridge provides an interface between the high-performance AXI domain and the low power APB domain. It appears as a slave on AXI bus but as a master on APB that can access up to sixteen slave peripherals. Read and write transfers on the AXI bus are converted into corresponding transfers on the APB. The AXI4-Lite to APB bridge Block diagram is shown in Figure1

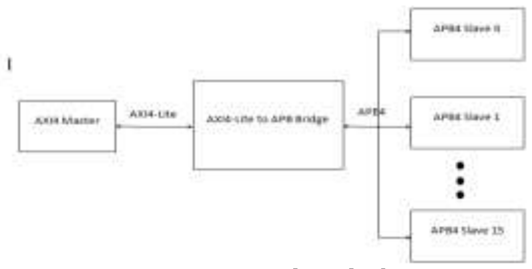

Figure1. AXI to APB Bridge Block Diagram

Features of bridge

The Xilinx AXI to APB Bridge is a soft IP core with these features:

1. AXI interface is based on the AXI4-Lite specification
2. APB interface is based on the APB3 specification, supports optional APB4 selection
3. Supports 1:1 (AXI:APB) synchronous clock ratio
4. Connects as a 32-bit slave on 32-bit AXI4-Lite
5. Connects as a 32-bit master on 32-bit APB3/APB4
6. Supports optional data phase time out.

3.1.1 AXI4-Lite Slave Interface

The AXI4-Lite 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 1024bits.

When both write and read transfers are simultaneously requested on AXI4-Lite, the read request is given more priority than the write request. This module also contains the data phase time out logic for generating OK response on AXI interface when APB slave does not respond.

3.1.2 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 to 32-bits.

3.2 Signal Connections

Figure2 shows the component signal connections. The bridge uses:

- AMBA AXI-Lite and APB signals as described in the AMBA AXI-Lite 4.0 protocol specification.

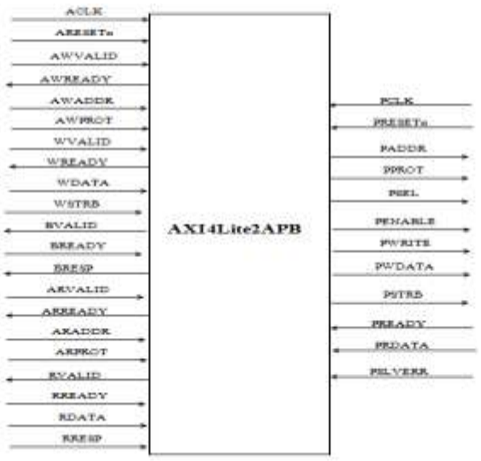

Figure 2. Signal Connection

3.3. Handshake Mechanism of AXI & APB

In AXI 4.0 specification, each channel has VALID and READY signals for handshaking. The source asserts VALID when the control information or data is available. The destination asserts READY when it can accept the control information or data. Transfer occurs only when both the VALID and READY are asserted. Figure3 Shows all possible cases of VALID/READY handshaking. Note that when source asserts VALID, the corresponding control information or data must also be available at the same time. The arrows in Figure. Indicate when the transfer occurs. A transfer takes place at the positive edge of clock. Therefore, the source needs a register input to sample the READY signal. In the same way, the destination needs a register input to sample the VALID signal. Considering the situation of last Figure, we assume the source and destination use output registers instead of combination circuit, they need one cycle to pull low VALID/READY and sample the VALID/READY again at T4 cycle. When they sample the VALID/READY again at T4, there should be another transfer which is an error. Therefore source and destination should use combinational circuit as output. In short, AXI protocol is suitable register input and combinational output circuit.

The APB Bridge buffers address, control and data from AXI4-Lite, drives the APB peripherals and returns data and response signal to the AXI4-Lite. It decodes the address using an internal address map to select the peripheral. The bridge is designed to operate when the APB and AXI4-Lite have independent clock frequency and phase. For every AXI channel, invalid commands are not forwarded and an error response generated. That is once a peripheral accessed does not exist, the APB Bridge will generate DE CERR as response through the response channel (read or write). And if the target peripheral exists, but asserts PSLVERR, it will give a SLVERR response.

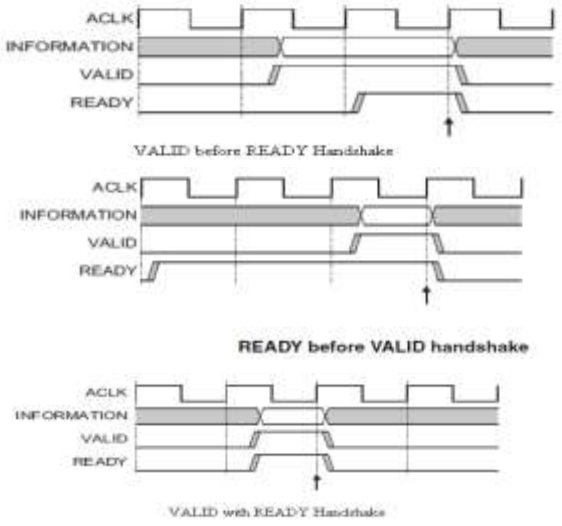
ACLK
INFORMATION
VALID
READY
VALID before READY Handshake
ACLK
INFORMATION
VALID
READY
READY before VALID handshake
ACLK
INFORMATION
VALID
READY
VALID with READY Handshake

Figure3. Waveforms for handshaking mechanism

3.4 Finite State Machine

A finite-state machine (FSM) or finite-state automaton (plural: automata), or simply a state machine, is a mathematical model used to design computer programs and digital logic circuits. It is conceived as an abstract machine that can be in one of a finite number of states. The machine is in only one state at a time; the state it is in at any given time is called the current state. It can change from one state to another when initiated by a triggering event or condition, this is called a transition. A particular FSM is defined by a list of the possible transition states from each current state, and the triggering condition for each transition.

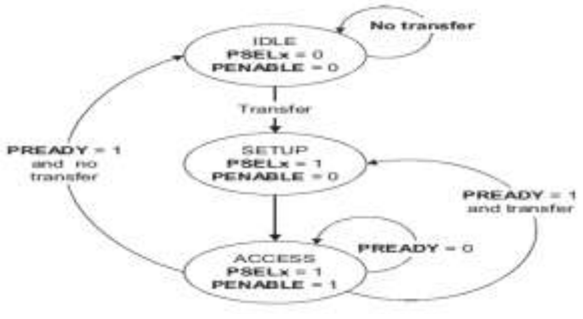

Figure4. Operational activity of the APB.

The state machine operates through the following states:
IDLE: This is the default state of the APB.
SETUP: 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 always moves to the ACCESS state on the next rising edge of the clock.
ACCESS: The enable signal, **PENABLE**, is asserted in the ACCESS state. The address, writes, select, and write data signals must remain stable during the transition from the SETUP to ACCESS state.
Exit from the ACCESS state is controlled by the **PREADY** signal from the slave:
• If **PREADY** is held LOW by the slave then the peripheral bus remains in
the ACCESS state.
• If **PREADY** is driven HIGH by the slave then the ACCESS state is exited and the bus returns to the IDLE state if no more transfers are required.

3.5 Simulation

The Timing Diagram shows the AXI4Lite to APB bridge operation for various read and write transfers. It shows that when both read and write requests are active, read is given more priority than write
.

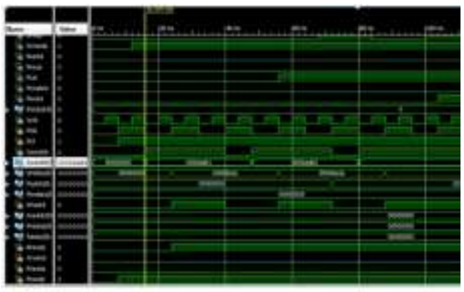

Figure5.Write side

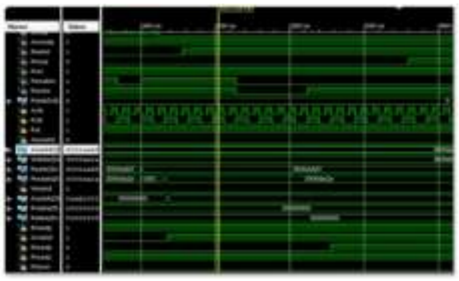

Figure6.Read side

3.6 Synthesis Result

```
---- Source Parameters
Input File Name                    : "apb_bridge_cha.prj"
Input Format                       : mixed
Ignore Synthesis Constraint File   : NO
---- Target Parameters
Output File Name                   : "apb_bridge_cha"
Output Format                      : NGC
Target Device                      : xc3s500e-4-fg320
---- Source Options
Top Module Name                    : apb_bridge_cha
Automatic FSM Extraction           : YES
FSM Encoding Algorithm             : Auto
Safe Implementation                : No
FSM Style                          : lut
RAM Extraction                     : Yes
RAM Style                          : Auto
ROM Extraction                     : Yes
Mux Style                          : Auto
Decoder Extraction                 : YES
Priority Encoder Extraction        : YES
Shift Register Extraction          : YES
```

```
Logical Shifter Extraction        : YES
XOR Collapsing                    : YES
ROM Style                         : Auto
Mux Extraction                    : YES
Resource Sharing                  : YES
Asynchronous To Synchronous       : NO
Multiplier Style                  : auto
Automatic Register Balancing      : No
---- Target Options
Add IO Buffers                    : YES
Global Maximum Fanout             : 500
Add Generic Clock Buffer(BUFG)    : 24
Register Duplication              : YES
Slice Packing                     : YES
Optimize Instantiated Primitives  : NO
Use Clock Enable                  : Yes
Use Synchronous Set               : Yes
Use Synchronous Reset             : Yes
Pack IO Registers into IOBs       : auto
Equivalent register Removal       : YES
=========================================================================
HDL Synthesis Report
Macro Statistics
# RAMs                                : 3
 256x32-bit dual-port RAM             : 3
# Counters                            : 6
 8-bit up counter                     : 6
# Registers                           : 69
 1-bit register                       : 49
 32-bit register                      : 10
 4-bit register                       : 4
 8-bit register                       : 6
# Latches                             : 3
 1-bit latch                          : 3
# Comparators                         : 3
 8-bit comparator equal               : 3
# Multiplexers                        : 1
 32-bit 4-to-1 multiplexer            : 1
# Xors                                : 12
 1-bit xor2                           : 6
 7-bit xor2                           : 6
=========================================================================
Advanced HDL Synthesis Report
Macro Statistics
# RAMs                                : 3
 256x32-bit dual-port distributed RAM : 3
# Counters                            : 6
 8-bit up counter                     : 6
# Registers                           : 433
 Flip-Flops                           : 433
# Latches                             : 3
 1-bit latch                          : 3
# Comparators                         : 3
 8-bit comparator equal               : 3
# Multiplexers                        : 1
 32-bit 4-to-1 multiplexer            : 1
# Xors                                : 12
 1-bit xor2                           : 6
 7-bit xor2                           : 6
```

```
=========================================================================
*                    Final Report                    *
=========================================================================
TIMING REPORT
Timing Summary:
Speed Grade: -4
   Minimum period: 5.254ns (Maximum Frequency: 190.331MHz)
   Minimum input arrival time before clock: 5.198ns
   Maximum output required time after clock: 4.394ns
   Maximum combinational path delay: No path found
CPU : 21.40 / 21.99 s | Elapsed : 21.00 / 22.00 s
Total memory usage is 222952 kilobytes
Number of errors   :    0 (   0 filtered)
Number of warnings :   24 (   0 filtered)
Number of infos    :   14 (   0 filtered)
```

IV. Conclusion

The Implementation APB Bridge is designed in this project. Verilog has been used for implementing the bridge. The bridge has the low cost interface optimized for minimal power consumption and reduced interface complexity.

An implementation of AXI4-Lite to APB bridge which has the following features:

- 32-bit AXI slave and APB master interfaces.
- PCLK clock domain completely independent of ACLK clock domain.
- Support up to 16 APB peripherals.
- Support the PREADY signal which translate to wait states on AXI.
- An error on any transfer results in SLVERR Bus the AXI read/write response.
-

V. Acknowledgement

I G. Madhuri would like to thank Associate Prof P. Sharmila Rani, who had been guiding through out to complete the work successfully, and would also like to thank the K. Jeevan Reddy HOD, ECE Department and other Professors for extending their help & support in giving technical ideas about the paper and motivating to complete the work effectively & successfully.

References

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

[2] http://en.wikipedia.org/wiki/System_on_a_chip#Structure

[3] Power.org Embedded Bus Architecture Report Presented by the Bus Architecture TSC Version 1.0 – 11 April 2008

[4] http://www.arm.com/products/system-ip/amba/amba-open-specifications.php

[5] ARM, "AMBA Protocol Specification 4.0", www.arm.com, 2010 ARM,AMBA Specification (Rev 2.0).AMBA® 4 AXI4™, AXI4-Lite™, and AXI4-Stream™ Protocol Assertions Revision: r0p0 User Guide.

[6] AMBA® APB Protocol Version: 2.0 Specifications.

[7] ASB Example AMBA System Technical Reference Manual Copyright © 1998-1999 ARM Limited.

[8] AHB to APB Bridge (AHB2APB) Technical Data Sheet Part Number: T-CS-PR-0005-100 Document Number: I-IPA01-0106-USR Rev 05 March 2007.

[9] LogiCORE IP AXI to APB Bridge (v1.00a) DS788 June 22, 2011 Product Specification.

[10] Simulation and Synthesis Techniques for Asynchronous FIFO Design Clifford E.Cummings, Sunburst Design, Inc. SNUG San Jose 2002 Rev 1.2., FIFO Architecture, Functions, and Applications SCAA042A November 1999.

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

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

[13] 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.

[14] Clifford E. Cummings, “Synthesis and Scripting Techniques for Designing Multi-Asynchronous Clock Designs,” SNUG 2001

[15] Simulation and Synthesis Techniques for Asynchronous FIFO Design Clifford E.Cummings, Sunburst Design, Inc. SNUG San Jose 2002 Rev 1.2.,

[16] 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.

[17] 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.