A Novel Arbitration Technique of AMBA AHB

Pallavi Kumari Gautam  
Naveen Upadhyay

Abstract – Resolution is a big issue in SOC (System on Chip) while dealing with number of master trying to sense a single data bus. The effectiveness of a system to resolve this priority resides in its ability to logical assignment of the chance to transmit data with of the data, response to the interrupts etc. The purpose of this paper is to propose the scheme to implement such a system using the specification of AMBA bus protocol. The scheme involves the typical AMBA features of ‘single clock edge transition’, ‘split transaction’, ‘several bus masters’ and ‘burst transfer’. The bus arbiter ensures that only one bus master at a time is allowed to initiate data transfers. Even though the arbitration protocol is fixed, any arbitration algorithm, such as highest priority or fair access can be implemented depending on the application requirements. The design architecture is written using VHDL (Very High Speed Integrated Circuits Hardware Description Language) code using Xilinx ISE Tools. The architecture is modeled and synthesized using RTL (Register Transfer Level) abstraction and Implemented on Virtex-2 series.

Keywords – AMBA, SOC, RTL, VHDL.

I. INTRODUCTION

Systems-on-Chip (SoC) and in particular embedded real-time systems typically consist of several computational elements. These elements fulfill different tasks for processing an overall solution. Let’s take a set-top box for TV sets as an example [1]. A set-topmost generate a TV-signal for a particular TV channel from a digital satellite signal. This process takes different tasks. One task is to split the incoming digital signal into data streams, such as video and audio. Another task is to convert the video stream into an actual TV-signal. One more conversion has to be made to turn the audio stream into an audio signal for the TV set. Meanwhile, another task handles the user input such as changing the channel when the remote control is pressed. All these tasks have to be done in parallel and are bound by real-time deadlines. The cost of missing these deadlines is visible as black boxes on the screen or audible as noise. This is unacceptable and therefore it is necessary to always deliver this data within hard real-time deadlines. These computational elements are either general-purpose processors or digital signal processors. Nowadays, multiple of them are integrated into a System-on-Chip solution. A processor needs to interact with other processors, memories or I/O devices to complete a task. Currently busses are used to interconnect these IP blocks. The current research in the field suggests using Networks-on-Chip (NoC) to interconnect IP blocks, because NoCs allow more flexibility than busses [1].

With the need of application, chip with a single processor can’t meet the need of more and more complex computational task. We are able to integrate multiple processors on a chip thanks to the development of integrated circuit manufacturing technology. Now as there are multiprocessing units and processors is getting faster, so compatibility with slow communication architectures a bit difficult furthermore this slow and conventional communication architecture limits the throughput.

To improve the performance we have to develop such efficient on chip Architecture which will be much faster system on chip solution which removes the limitation of communication architecture one of the solution is “AHB bus” but it can’t give perfect parallelism as it can allow only one master to communicate at one slave only. While in our design there are five independent transfer channels which make multiple masters access multiple slaves at the same time and gain a perfect parallelism performance in MPSOC design. The bus arbiter ensures that only one bus master at a time is allowed to initiate data transfers. Even though the arbitration protocol is fixed, any arbitration algorithm, such as highest priority or fair access can be implemented depending on the application requirements.

AHB is a new generation of AMBA bus which is intended to address the requirements of high-performance synthesizable designs. It is a high performance system bus that supports multiple bus masters and provides high bandwidth operation.

Figure 1: AMBA AHB Block diagram
AMBA AHB implements the features required for high performance, high clock frequency systems including:

- Burst transfers
- Split transactions
- Single-cycle bus master handover
- Single-clock edge operation
- Non-tristate implementation
- Wider data bus configurations (64/128 bits)

In this paper, an industry-standard ARBITER is implemented for AHB (AMBA). The specification describes the bus attributes, the protocol definition and types of transactions, bus management, and the programming interface required to design.

II. PROPOSED METHODOLOGY

Arbitration to choose the next bus master uses a round robin arbitration algorithm. This ensures that no master gets starved. When a master has locked the bus, the round robin arbitration is overridden and the master with the lock retains highest priority to the bus.

Figure 2: Pin Diagram for AHB Arbiter
The sixteen AMBA Bus masters are Master 0 through Master 15. Slave 0 through Slave 15 are the sixteen AMBA Bus slaves. The AMBA AHB Bus Arbiter/Decoder contains a default master-Master0, and a default slave-Slave 0.

AMBA AHB Bus Arbiter features are summarized:
- AMBA AHB Bus arbiter function
- Round robin arbitration
- Default master- Master 0
- Default slave- Slave 0

The Arbiter block monitors the AMBA Bus for requests and chooses the master with highest priority request as the next AMBA bus transaction master. If there are no
requests, the Default Master is chosen as the master to drive the next AMBA Bus transaction.

**Testing the IP**

The method of testing the IP was kept simple since it was more important to concentrate on the functionality of the IP.

**The Subsystem Specifications**

The AHB ARBITER IP can be broken into two subsystems. The two major components of the system under design are the controller and data path.

**Controller**

The controller is essentially a Mealy state machine. It keeps track of the different sections of an ARBITER transaction. The first state of the controller is the start state. The next state to check is the grant if grant is there then it will make the necessary signal high which will further control the counter, and counter interface block.

**Data path**

The data path are further divided into several sub-system blocks, few of them are controlled by controller. Followings are different data path with different functionality.

1. Priority logical block
2. Priority storage block
3. Mux arrangement for grant and bus request.
4. Or gate
5. Counter

For details please refer Figure 3.

**Overall Description of Design**

The followings are the major steps involves in the Arbiter designs form architectural or functional point of view:

- This Grant signal goes to priority storage block, encoder block and as out_put port to interact with Master. After getting the Grant signal Master will send Address, Burst to indicate the type of transfer, and Slave will also send Hready, Hresp, Hsplit.
- At the same time when master samples the signals to the Arbiter Grant signals which are the output of the Arbiter pass through the mux, inside the Arbiter, for this mux bus_master no is select line, which indicates that which master is accessing the bus.
- The out_put mux then passes to the controller block which will generate the necessary signals for counter, i.e., the controller will control the operation of counter.
- The grant output from the priority logic block is Or and then sent to the priority storage block which will store the priority and pass the enable signal to the next priority depending upon the grant value. And the whole operation is repeated depends upon the transaction mode.

**III. SIMULATION RESULTS**

![Figure 4: Pin diagram of top arbiter](image)

![Figure 5: RTL Schematic of arbiter](image)
Device utilization summary for arbiter:

- Selected Device: 2v80cs144-6
- Number of Slices: 1566
- Number of Slice Flip Flops: 533
- Number of 4 input LUTs: 2752
- Number of bonded IOBs: 64
- IOB Flip Flops: 4
- Number of GCLKs: 2

Figure 6: Pin diagram of Counter Interface

Figure 7: RTL Schematic of Counter interface

Figure 8: Pin diagram of priority storage

IV. CONCLUSION

The design is simulated on Modelsim 6.5 & verified through effective test bench. The RTL is implemented on Vertex-2 (XC2V80, CS144 package). The advantage of this design is that we have taken care of latch formation, as it is a FPGA implementation hence with less latch & maximum flip-flop have enhanced our area efficiency. Cyclic FSM (grey encoding) has been done for controller design & we controlled power Consumption the design can further be optimized for ASIC design.

REFERENCE


