International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Design and Implementation of Efficient FSM for AHB Master and Arbiter

Muzammel Hoque[#1], Owais Shah[#2]

Abstract: ***Due to Moore's law more and more amount of logic is being placed onto a single silicon die and it is driving the development of highly integrated SoC designs. So this high computational power must be matched with interconnect fabric which can handle it. There are many interconnect buses that are widely used in the industry like AMBA, Wishbone, Core Connect, Avalon etc. AMBA is most proffered among all of them because it has a hierarchy of buses with AHB (Advance high performance bus) can be connected to high performance peripherals and APB (Advance Peripheral Bus) that can be connected to low performance peripherals. Nowadays in industry development of Silicon on Chip (SOC) devices with reusable IP cores are given higher priority, the major challenge faced here is to ensure proper lossless communication between these IP cores in SOC device, this can be ensured with the help of standard communication protocols such as AMBA from ARM Ltd. In this paper we design and synthesize efficient Finite State Machine (FSM) for master and arbiter interface in AMBA AHB.***

Keywords: FSM, AHB, Master and Arbiter, hamster

1. Introduction

Advanced Microcontroller Bus Architecture (AMBA) is a protocol that is used as an open standard, on chip interconnect specification for the connection and management of functional blocks in a system-on-chip (SoC) [1] AMBA assists the progress of right-first-time development of multiprocessor designs with large number of controllers &peripherals [2] The Advanced Microcontroller Bus Architecture (AMBA) has the ability to re-use designs and here it means it has the ability to re-use IP. IP re-use in today's technology is an important factor in reducing the development costs and timescales for System-on-chip (SoC) [3] AMBA is a standard interface specification that makes sure of the compatibility between IP components provided by different design teams or vendors.

The worldwide reception of AMBA specifications all over the semiconductor industry has driven a comprehensive market in third party IP products and tools to support the development of AMBA based systems. OVERVIEW OF AMBA BUSES there are three different buses defined within the AMBA 2.0 specification – Advanced High Performance Bus (AHB), Advanced System Bus (ASB), and Advance Peripheral Bus (APB). This paper mainly deals with AMBA AHB and particularly AHB master and arbiter. Advanced High-performance Bus (AHB) The AMBA AHB is for high-performance, high clock frequency system modules. The AHB acts as the high-performance system backbone bus. AHB supports the efficient connection of processors, on-chip memories and off-chip external memory interfaces with low-power peripheral macro cell functions. AHB is also specified to ensure ease of use in an efficient design flow using synthesis and automated test techniques. Advanced System Bus (ASB) The AMBA ASB is for high-performance system modules. AMBA ASB is an alternative system bus suitable for use where the high-performance features of AHB are not required. ASB also supports the efficient connection of processors, on-chip memories and off-chip external memory interfaces with low-power peripheral macro cell functions. Advanced Peripheral Bus (APB) The AMBA APB is for low-power peripherals. AMBA APB is optimized for minimal power consumption and reduced interface complexity to support peripheral functions. APB can be used in conjunction with either version of the system bus.

2. Objectives of the AMBA Specification

The AMBA specification has been derived to satisfy four key requirements:

- To facilitate the right-first-time development of embedded microcontroller products with one or more CPUs or signal processors
- To be technology-independent and ensure that highly reusable peripheral and system macro cells can be migrated across a diverse range of IC processes and be appropriate for full-custom, standard cell and gate array technologies
- To encourage modular system design to improve processor independence, providing a development road-map for advanced cached CPU cores and the development of peripheral libraries
- To minimize the silicon infrastructure required to support efficient on-chip and off-chip communication for both operation and manufacturing test.

3. A Typical AMBA-Based Microcontroller

An AMBA-based microcontroller typically consists of a high-performance system backbone bus (AMBA AHB or AMBA ASB), able to sustain the external memory bandwidth, on which the CPU, on-chip memory and other Direct Memory Access (DMA) devices reside. This bus provides a high-bandwidth interface between the elements that are involved in the majority of transfers. Also located on the high performance bus is a bridge to the lower bandwidth APB, where most of the peripheral devices in the system are located.

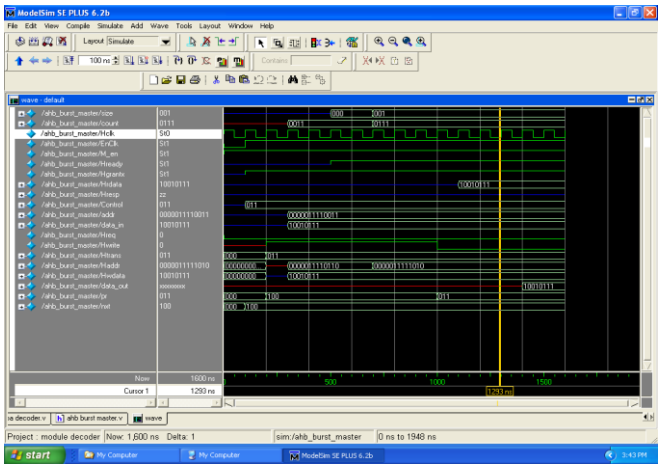
AMBA AHB Slave Interface

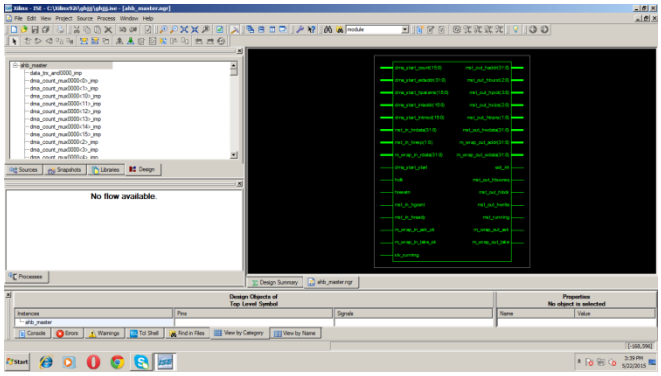
An AHB bus slave responds to transfers initiated by bus masters within the system. The slave uses a HSELx select signal from the decoder to determine when it should respond to a bus transfer, in this work initially for the design of slave module the hselx signal was given from the test bench, later while simulating the top module the hselx signal is given from the decoder itself according to the address requested by the master module. The other signals required for the transfer, such as address and control information is generated by the bus master, and is given to the slave module through arbiter module. The slave in this work does use the hsplitx signal i.e. the slaves in this paper is capable of SPLIT and RETRY functions. SPLIT transfers improve the overall utilization of the bus by separating (or splitting) the operation of the master providing the address to a slave from the operation of the slave responding with the appropriate data. When a transfer occurs the slave can decide to issue a SPLIT response if it believes the transfer will take a large number of cycles to perform. This signals to the arbiter that the master which is attempting the transfer should not be granted access to the bus until the slave indicates it is ready to complete the transfer. Therefore the arbiter is responsible for observing the response signals and internally masking any requests from masters which have been SPLIT. During the address phase of a transfer the arbiter generates a tag, or bus master number, on hmaster [1:0] which identifies the master that is performing the transfer. Any slave issuing a SPLIT response must be capable of indicating that it can complete the transfer, and it does this by making a note of the master number on the hmaster [1:0] signals. Later, when the slave can complete the transfer, it asserts the appropriate bit, according to the master number, on the hsplitx [1:0] [Refer Appendix A] signals from the slave to the arbiter. The Figure 3 shows the AHB slave interface synthesized in this work.

4. Result Analysis

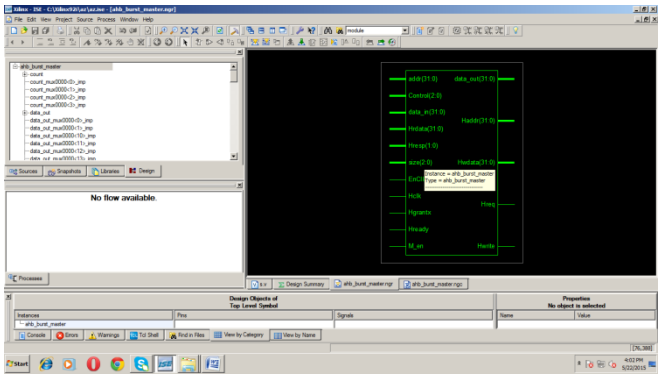
After designing the finite state machine of the AHB master and slave, Verilog hardware description language is used to implement it and check its functionality and correctness. In this paper the simulation is done with the help of Modelsim 10.03a and the synthesis is done with the help of Xilinx ISE design tool.

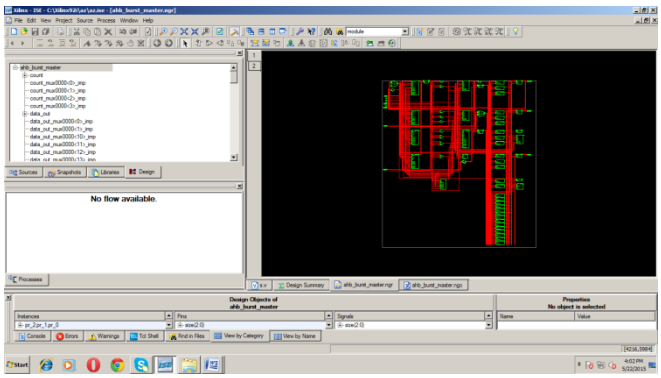
Simulation Result

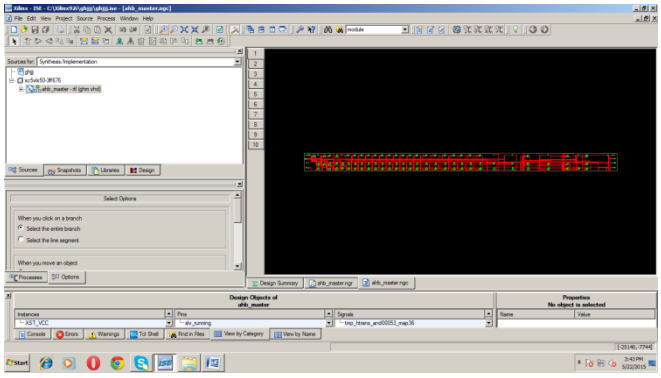


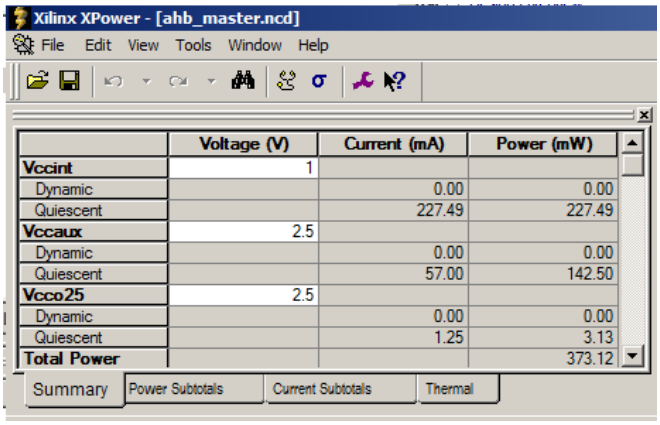


RTL Schematic



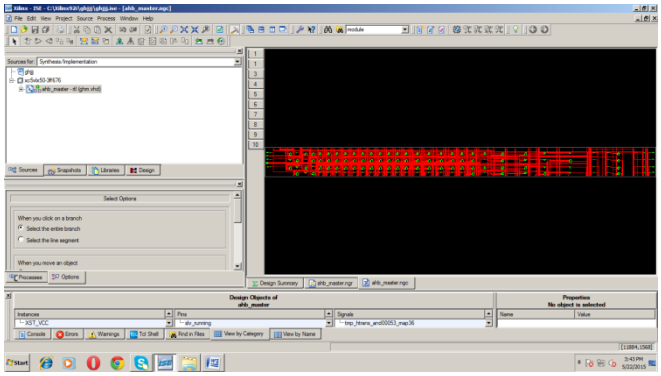


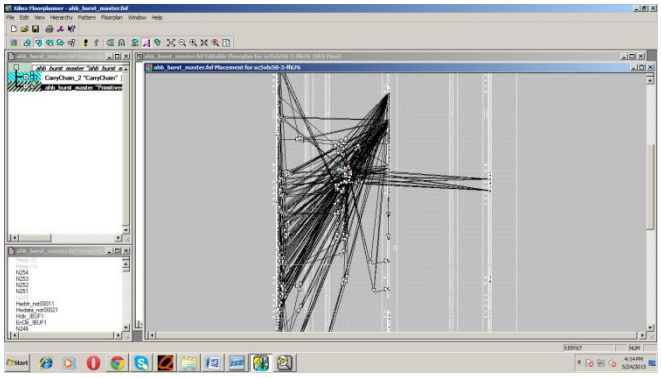


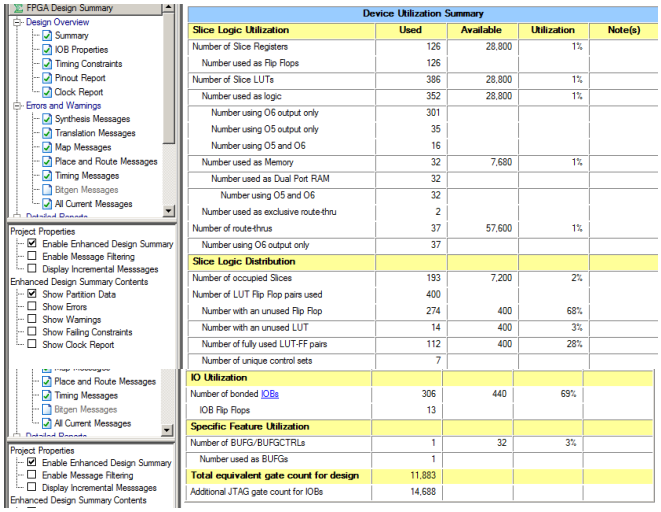


	Voltage (V)	Current (mA)	Power (mW)
Vccint	1		
Dynamic		0.00	0.00
Quiescent		227.49	227.49
Vccaux	2.5		
Dynamic		0.00	0.00
Quiescent		57.00	142.50
Vcco25	2.5		
Dynamic		0.00	0.00
Quiescent		1.25	3.13
Total Power			373.12

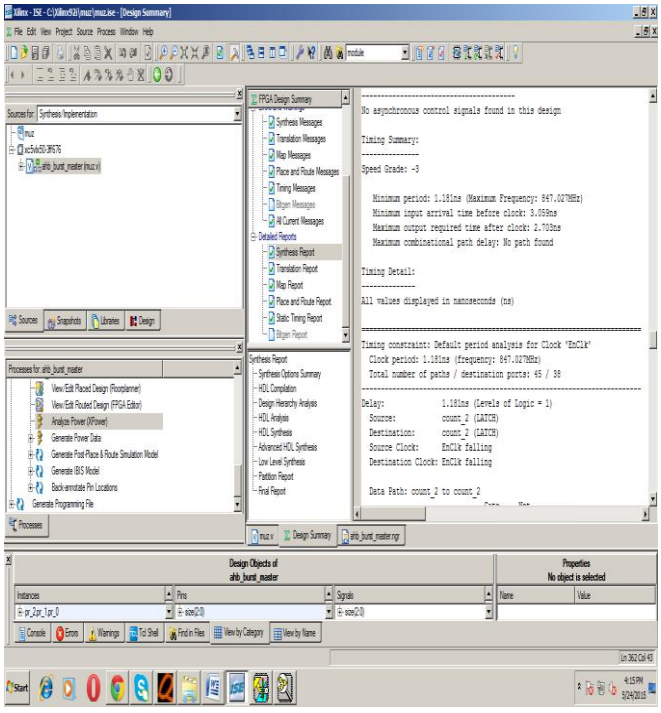
Technological View







Device Utilization Summary				
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	126	28,800	1%	
Number used as Flip Flops	126			
Number of Slice LUTs	386	28,800	1%	
Number used as logic	352	28,800	1%	
Number using O6 output only	301			
Number using O5 output only	35			
Number using O5 and O6	16			
Number used as Memory	32	7,680	1%	
Number used as Dual Port RAM	32			
Number using O5 and O6	32			
Number used as exclusive route-thru	2			
Number of route-thrus	37	57,600	1%	
Number using O6 output only	37			
Slice Logic Distribution				
Number of occupied Slices	193	7,200	2%	
Number of LUT Flip Flop pairs used	400			
Number with an unused Flip Flop	274	400	68%	
Number with an unused LUT	14	400	3%	
Number of fully used LUT-FF pairs	112	400	28%	
Number of unique control sets	7			
IO Utilization				
Number of bonded IOBs	306	440	69%	
IOB Flip Flops	13			
Specific Feature Utilization				
Number of BUFG/BUFGCTRLs	1	32	3%	
Number used as BUFGs	1			
Total equivalent gate count for design	11,883			
Additional JTAG gate count for IOBs	14,688			



Power Consumption