ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 9, Issue 4, October 2019

An Effective and Area Efficient VHDL Implementation of advanced high-performance bus Slave

Hitanshu Saluja, Naresh Grover
Manav Rachna International Institute of Research and Studies, Faridabad, India

Abstract –Most of the FPGA features communicate with each other through bus means. Each block is designed for a specific bus. The buses related to this work is the AMBA bus and the Wishbone bus. It works with the master / slave architecture. The AMBA bus is extensively utilized in the system on chip solution for interaction with different peripheral. This work elaborates the AMBA bus interface bridge between memory controller and other supporting peripheral. The work claims the implantation of AHB bus slave. Despite the shortcomings of the work performed study and development that followed has led the development of a memory controller on AMBA-AHB bus at a very advanced stage and next to prototyping. VHDL code is utilized to develop the design and it is synthesized in Xilinx Spartan 3 device (3s100evq100-5). The design claims a minor area overhead with improvement in speed 171.416 MHz.

Keywords **–AMBA, AHB, SoC, VHDL, Xilinx.**

I. INTRODUCTION

The model of advanced high-performance bus (AHB) was built incrementally [1, 2]. Following each increment, we proceeded to the verification of finiteness, completeness and coherence of the model, while solving (by means of abstractions) the problems of combinatorial explosion. Several other properties specifically defined for the AHB bus were then checked on the final model.

These specific properties were formulated by a well-known company in the field of computer assisted design. We received various properties covering the AHB bus specification. Since our case study only concerns the AHB bus specification, only a subset of these properties has been used. Nevertheless, our study considers a greater number of properties than most other published works on formal AHB bus verification.

The methodology we have proposed is mainly intended for the verification of shipboard systems. It therefore considers both a hardware part and a software part. Although she does not interested in the software part as such, our case study remains particularly interesting since few modelling experiments of material systems have been carried out using the coloured Petri nets [3]. On the other hand, the examples of modelling of software systems using this formalism are much more numerous [4].

Advanced High-Performance Bus (AHB)

The Advanced Microcontroller Bus Architecture (AMBA) specification is a hardware bus specification made by ARM. In fact, this specification contains the definition of three bus versions: Advanced Peripheral Bus (APB), Advanced System Bus (ASB), and Advanced High Performance Bus (AHB). The APB bus is a communication bus with slow devices. It is optimized to consume little energy, but it transmits the information hands quickly than the other two buses. To implement the main bus of a system, an ASB or AHB bus is recommended. The AHB bus is a more complicated bus, but more efficient than the others. You can bridge to the APB bus from the ASB or AHB bus [5].

The AHB bus presents a two-stage pipeline architecture dealing respectively with the address and data of the transfer. In addition, it offers several types of transfer including a burst [6].

As mentioned at the beginning of this paper, the AHB bus specification is a simplified version of the AHB bus specification (Figure 1) and this is the one considered for the case study [7].

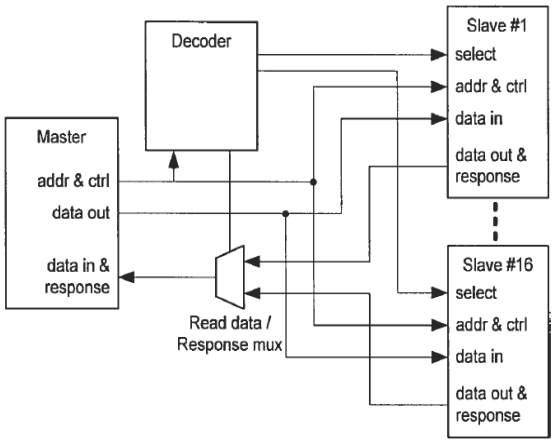

Fig 1: Schematic representation of an AHB bus [8]

This simplified version has the advantage of simplifying the design of the interface of the modules with this bus. Moreover, we can add a standard wrapper to ensure the compatibility of this simplified interface with the complete specification if it is necessary. Note that the uncomplicated version of the bus presents to the hands a case of deadlock, with the type transfers SPLIT, which is not present in the simplified version.

The AHB bus has three main components: a master, slaves and a decoder. The masters and slaves are actually external modules represented by interfaces interacting with the bus. The master type interface makes transfer requests. The slave interface receives and responds to transfer requests. If a module must initiate and receive transfer requests, it must implement both types of interface. From here, we call a master, the master interface of a module, and a slave, the slave interface of a

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 9, Issue 4, October 2019

module. The signals used by the bus are presented in Table 1.

Table 1: AHB bus signals

Signal Name	Definition	Source
HCLK	Clock of the bus	Source of the clock
HRESETn	Reset	Reset Controller
HADDR [3 1: 0]	Master	Addressing bus
HTRANS [1: 0]	Transfer	Type Master
HWRITE	Transfer Direction	Master
HSIZE [2: 0]	Transfer Type	Master
HBURST [2:0]	Burst Type	Master
HPROT [3: 0]	Master	Protection Control
HWDATA [31: 0]	Master	Write Bus
HSELx	Decoder	Slave Selector
HRDATA[31:0]	Slave	Slave read bus
HREADY	End of Transfer	Slave
HRESP [1:0]	Slave	Transfer response

Figure 2 shows the slave bus interface module.

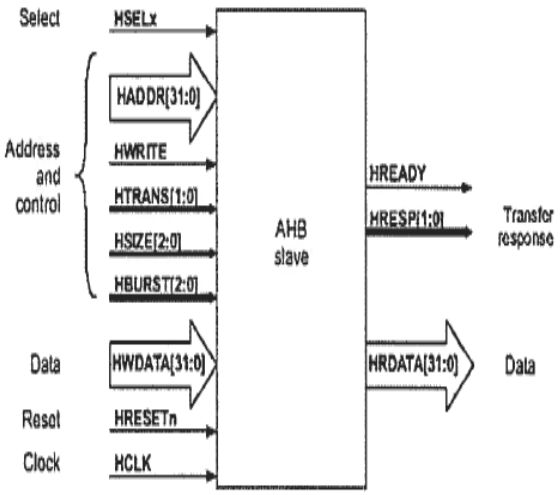

Fig 2: Slave Bus Interface AHB [9]

This simpler bus imposes several constraints on the AHB bus. It allows only one master. Slaves can only give a positive response (OKAY) OR an error response (ERROR) to a request from the master. Done, they cannot ask the master to retry the request (RETRY) or divide the request into two parts (SPLIT) as in the complete bus. Slaves can still give an OKAY response with the WAIT signal to have more time before responding to the request. As there is only one master, there is no referee who manages the access to the bus by the masters [10].

Proposed model is deployed on a top page called Top (Figure 3). It contains a Master page and a Slave page. The Top Page describes transitions made by the clock and decoder. The transitions made by the master and the slaves have been specified in the most restrictive way possible by means of sufficiently strong preconditions. This is important if one wants to avoid triggering these transitions in contexts at the specification would not command it explicitly [11]. This is necessary to allow the detection of the missing requirements. If the preconditions are not strong enough, the corresponding transitions may be triggered although the specification does not necessarily allow it, the missing requirements, generally detected by an absence of possible behaviour (a blocking) associated with a given state of the system, will then be masked by the execution of these transitions more lax that the specification does not imply it. Therefore, to have a good model, it is important to define preconditions that correspond precisely (and only) to the cases covered by the specification [12].

In this section, a simplified model of the AHB case study is presented. This model contains a master and a slave and allows only SINGLE type read transfers, without allowing the master to send IDLE, BUSY signals and without allowing the slave to send WAIT signals. In Figure 2, the slave regains the signal HSELx which comes from the decoder.

```
Block = 64;
BOOL = bool;
E = with e;
 ADDR = int with 0 ..(block-1);
 DATAT = with data l xxx;
 HOTONE16T = int with -1 ..15;
TRANST          with IDLE I BUSY I NONSEQ I SEQ;
BURSTT          with SINGLE | INCR | WRAP4 |
WRAP8 | WRAP16 | INCR4 | INCR8 | incrl6;
 PROTT with DATAF;
 SIZET  with  s8,sl6,s32,s64,sl28,s256,s512,sl024;
RESPT  with OKAY I ERROR;
CTRL   product BOOL*SIZET*PROTT;
TRAN   product
INT*ADDR*BOOL*SIZET*PROTT*BURSTT;
```

Here is the declaration of all the necessary variables in this example:

```
slav,n: INT;
dat, dat2: DATAT;
burst: BURSTT;
prot: PROTT;
addr, addr2: ADDR;
sizel, size2: SIZET;
write: BOOL;
```

Then we create the structure and interface of the modules. The decoder module has been included directly on the top page because it contains only one transition. The slave module contains only one transition and has its own page, because other transitions are added in the version complete model. Finally, we add the behaviour in the pages of the modules.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 9, Issue 4, October 2019

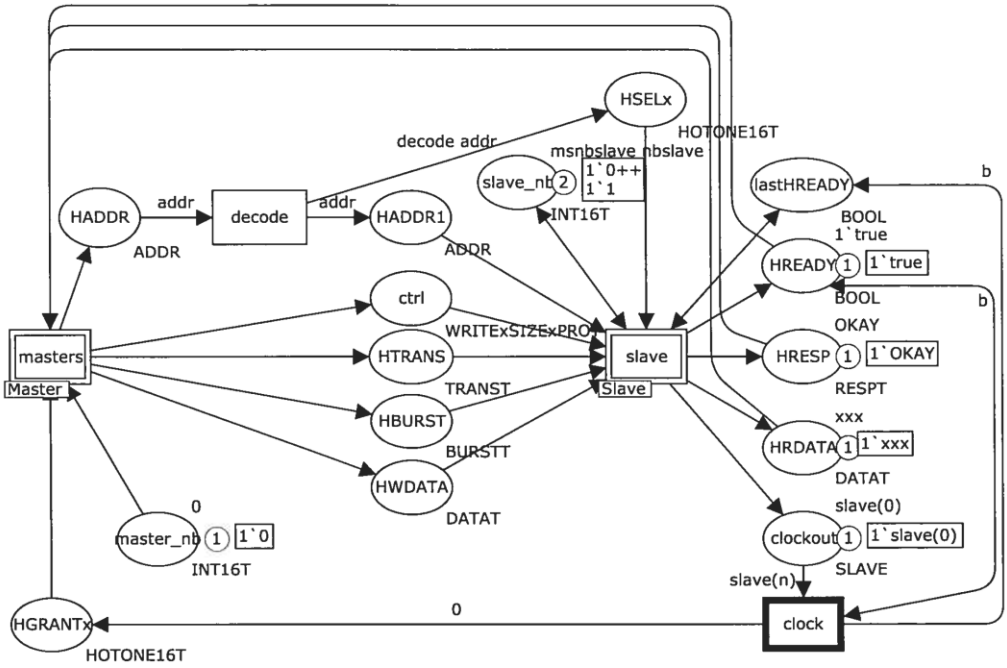

Fig 3: Root page of the case study [2]

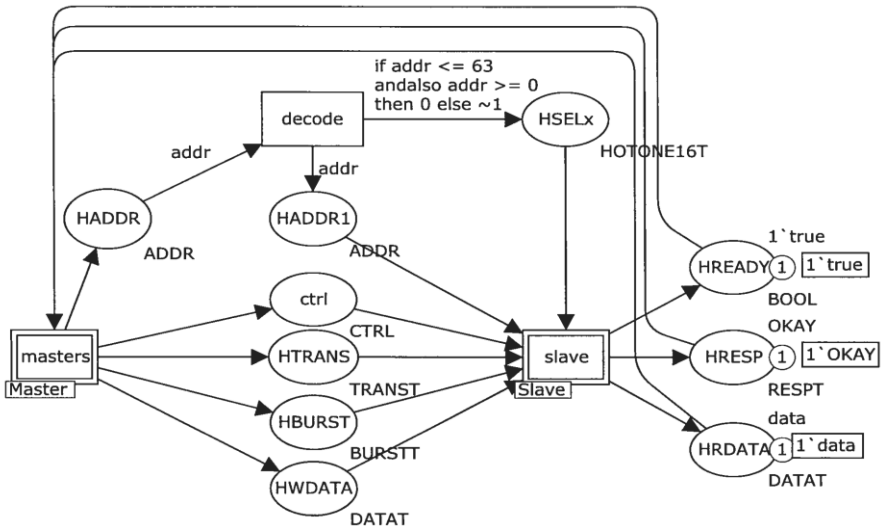

Fig 4: Demonstration of the case study: top page [2]

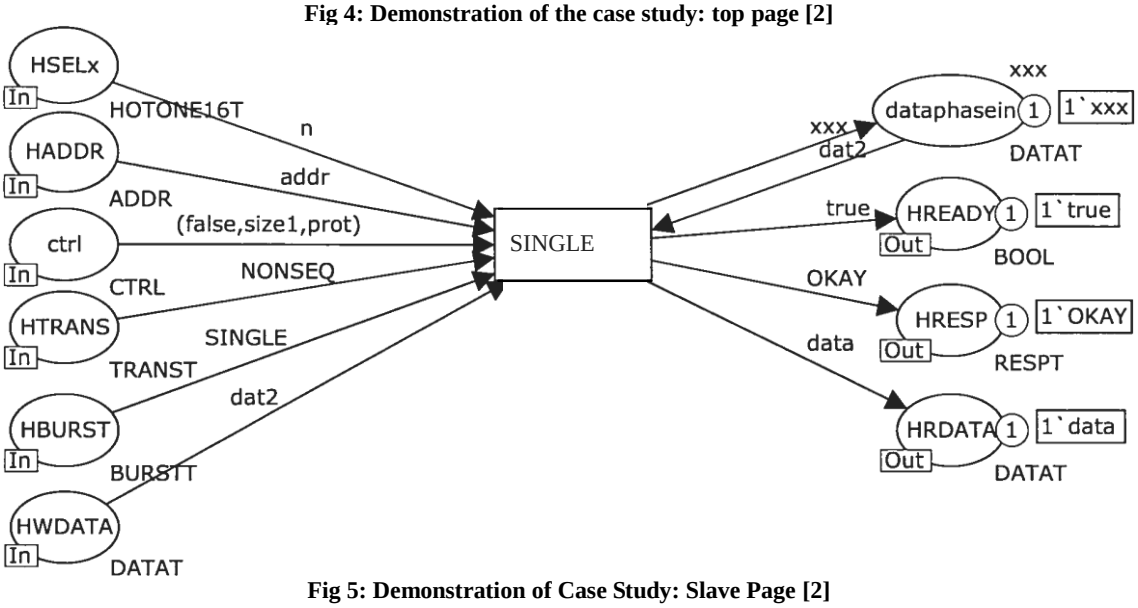

Fig 5: Demonstration of Case Study: Slave Page [2]

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 9, Issue 4, October 2019

II. AHB SLAVE BUS

An AHB bus slave shown in Figure 6 responds to transfers initiated by bushmasters within the system. The slave uses a HSELx select signal from the decoder to determine when it should respond to a bus transfer. All other signals required for the transfer, such as the address and control information, will be generated by the bus master [13].

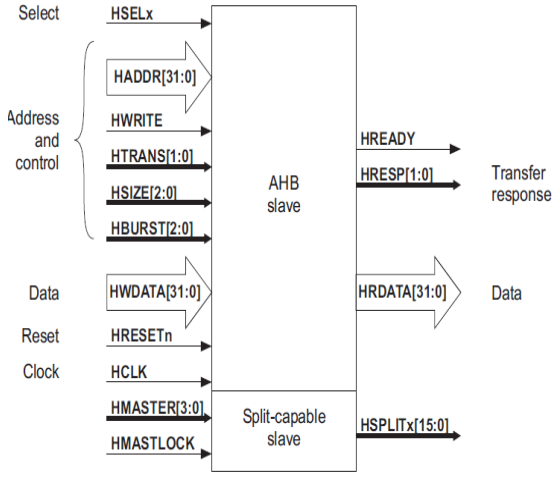

Fig 6: AHB bus slave interface [2]

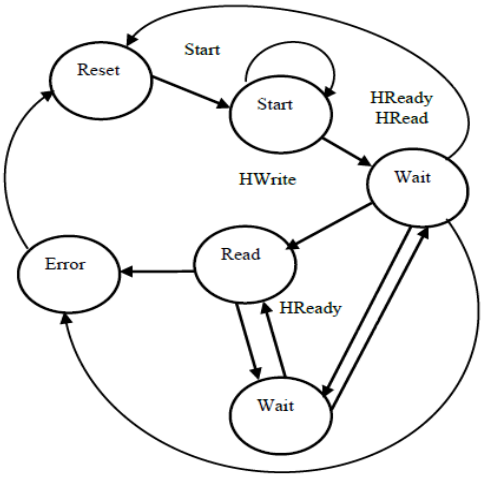

Fig 7: Slave Interface [2]

Figure 7 shows the state diagram of slave interface. It is a finite state machine implementation initial condition is reset state which is an idle state when no operation is there. When start signals arrived then this finite state machine (FSM) triggers, depends upon the hready and hwrite signal it decides in which further state it has to move. If hraedy is low then it will be start state only, if hready is one then depends upon hwrite it move to read or write state, if Hwrite is one then state moves to write otherwise it moves to read state. If hready will become low between these states then it moves to wait state, and it will remains in these sate until hready will become one .if any error occurs which is indicated by Hresp then state moves to error state.

III. SIMULATION RESULTS

This block takes hresp 2 bit as input based on that it start its operation if hresp is 00 then it is normal operation. hclk is the system clk. hready is the signal which is high then only normal operation can be performed otherwise it will be in wait state. hresset are the initialization signal. hwrite is the control signal which is high then it is write operation if it is low then read operation.cmd is the output which is control signal and further interact with memory controller, RAM and ROM is the control signal which tell which is ram or rom operation.

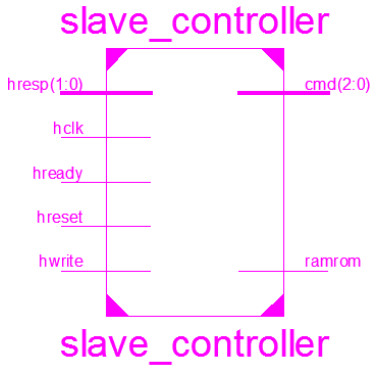

Fig 8: Pin diagram of slave controller

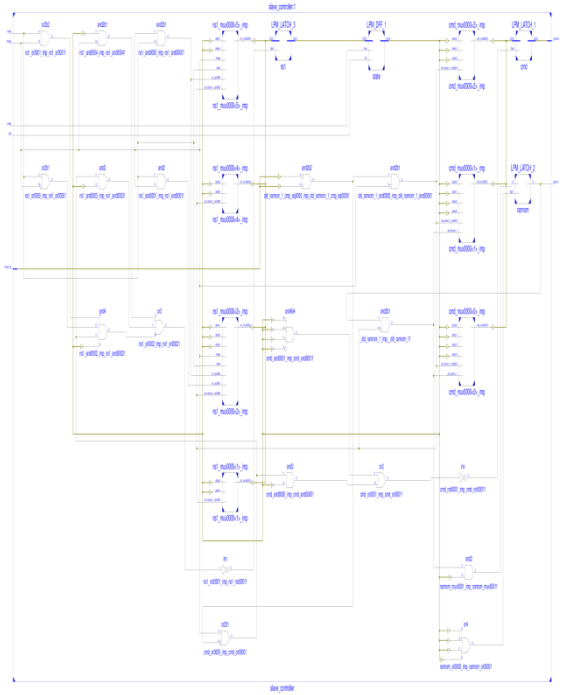

Fig 9: RTL view of slave

The above figure shows that register transfer level view of slave controller interface, it shows that in detail how flip flops and gates are interconnected each other of slave controller interface. Figure 8 shows that pin diagram of slave controller interface but here only difference is it will show register transfer level.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 9, Issue 4, October 2019

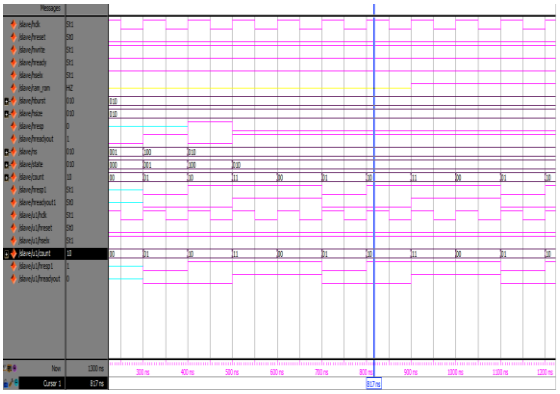

Fig 10. Simulation result of AHB slave interface

The above result shows the handshaking between slave and slave interface where slave is directly interacting with master and its response is given by slave, Further the read write operation is forwarded to memory controller by slave interface.

Device Utilization Summary:

Selected Device: 3s1000fg320-5
Number of Slices: 402 out of 7680 5%
Number of Slice Flip Flops:457 out of 15360 2%
Number of 4 input LUTs:738 out of 15360 4%
Number of IOs: 88
Number of bonded IOBs: 88 out of 221 39%
IOB Flip Flops: 71
Number of GCLKs: 7 out of 8 87%
Timing Summary:
Speed Grade: -5
Minimum period: 5.834ns (Maximum Frequency: 171.416MHz)
Minimum input arrival time before clock: 10.076ns
Maximum output required time after clock: 7.660ns

IV. CONCLUSION

In this paper, AHB based slave interface is designed. The design has been implemented using VHDL for SOC solution. The design has taken care of balance between area and speed. Every implementation is in structural approach hence it's become a well-documented frame. There is separate implementation of slave and slave interface with inter faces the memory controller further. To avoid the handshaking complexity we have used FIFO for slave interface. Even it increases the latency but at the same time it makes design simple and bottleneck problem free. The design has been synthesized on XILINX 13.1 Spartan 3, and simulated in MODELSIM.

REFERENCES

[1] A. Limited, "AHB-Lite Overview," Copyright: ARM Limited. All rights reserved 2001.

[2] A. Limited, "AMBA Specification (Rev 2.0)," Copyright: ARM Limited. All rights reserved 1999.

[3] Shilpa Rao and Arati S. Phadke, "Implementation of AMBA compliant Memory Controller on a FPGA", IJETEE, 2013.

[4] Archana C. Sharma, Prof.Zoonubiya Ali, "Construct High-Speed SDRAM Memory Controller Using Multiple FIFO's for AHB Memory Slave Interface", IJETAE, 2013.

[5] KeesGoossens, Om Prakash Gangwal, Jens R¨over, and A. P. Niranjan. Interconnect and Memory Organization in SOCs for advanced Set-Top Boxes and TV-Evolution, Analysis, and Trends. In JariNurmi, HannuTenhunen, JouniIsoaho, and Axel Jantsch, editors, Interconnect-Centric Design for Advanced SoC and NoC, chapter 15, pages 399–423. Kluwer, April 2004.

[6] Hu Yueli; Yang Ben, "Building an AMBA AHB Compliant Memory Controller", IEEE, 2011.

[7] Jayapraveen. D and T. GeethaPriya, "Design of memory controller based on AMBA AHB protocol", Elixir International Journal, 2012.

[8] Arun G, Vijaykumar T, "Improving Memory Access time by Building an AMBA AHB compliant Memory Controller", IJARCET, 2012.

[9] Ch.Vijayalakshmi, Mr B.Raghavaiah, "Implementation of AMBA AHB Compliant Memory Controller with Peripherals", ICITEC, 2012.

[10] VarshaVishwarkama, Abhishek choubey, Arvind Sahu, "Implementation of AMBA AHB protocol for high capacity memory management using VHDL", IJCSE, 2011.

[11] Shashisekhar Ramagundam, Sunil R. Das, Scott Morton, Satyendra N. Biswas, VoicuGroza, Mansour H. Assaf, and Emil M. Petriu, "Design and Implementation of High-Performance Master/Slave Memory Controller with Microcontroller Bus Architecture", IEEE International Conference on Instrumentation and Measurement Technology (I2MTC) Proceedings, pp. 10 – 15, May 2014.