Verification of AHB Protocol using UVM

Tejaswini H N¹ Asst. Prof. Dept. of ECE, Sambhram Inst of Technology, Bangalore, India Revati Bothe² Team leader, SmartPlay Technologies Bangalore, India Ravishankar C V ³ Prof. Dept. of ECE Sambhram Inst of Technology, Bangalore, India

Abstract - This paper describes the verification of AHB Protocol using the methodology UVM(Universal Verification Methodology). AHB is an Advanced High performance system Bus that supports multiple masters and multiple slaves. It implements burst transfers, split transactions, single-cycle bus master handover, single-clock edge operation, wider data bus configuration (64/128bits). Verification IP is the one which provides a smart way to verify the AHB Components such as Master, Slave, Arbiter and Decoder. UVM is used for the verification of AHB Protocol which provides the best framework to achieve CDV (Coverage Driven Verification) which combines automatic test generation, self-checking test benches, and coverage metrics to significantly reduce the time spent on verifying a design. An UVM test bench is composed of reusable called (verification verification environments VCs Componenets). This paper examines the verification of VCs which are structured to work with Verilog, VHDL, System Verilog and System C.

Keywords : AHB, CDV, UVM, TLM, VC, Test Bench, Sequencer

I. INTRODUCTION

This section describes the survey that was done and the verification methodology is described and an outline of the rest of this dissertation provided.

AHB (Advanced High Performance Bus) is a new generation of AMBA (Advanced Microcontroller Bus Architecture) bus which is intended to address the requirements of high-performance synthesizable designs. AMBA AHB is a new level of bus which sits above the APB and implements the features required for high-performance, high clock frequency systems. UVM is a complete verification methodology that codifies the best practices for development of verification environments targeted at verifying large gatecount, IP-based SoCs (System on chip). Verification productivity stems from the ability to quickly develop individual verification components, encapsulate them into larger reusable verification components (VCs), and reuse them in different configurations and at different levels of abstraction. UVM uses a System Verilog implementation of standard Transaction Level Modeling (TLM) interfaces for modular communication between AHB components such as Master and Slave . The System Verilog UVM Class Library also provides various utilities to simplify the development and use of verification environments. These utilities support

debugging by providing a user- controllable messaging utility. The System Verilog UVM Class Library provides global messaging facilities that can be used for failure reporting and general reporting purposes.

A. Objective of the study

1) To study the specifications of AMBA AHB which include all the scenarios .

2) To generate the Test Plan comprises of Test Cases which meets the specified scenarios .

3) To understand the development of Verification Environment for Single Master -Single Slave which follows the UVM topology of Driver, Sequencer and a Monitor along with Test Plan.

4) Analysis on how the driver drives the sequences from the sequencer to the Score Board.

5) In the scoreboard, the actual output is compared with the expected one. If the obtained output matches with the expected result means verification is completed successfully

II. ARCHITECTURE OF UVM TEST BENCH

A. UVM Test Bench and Environment

An UVM test bench is composed of reusable verification environments called Verification Components (VCs). The VCs are applied to the device under test (DUT) to verify the implementation of the AHB protocol. Architecture of UVM Test Bench is shown in Figure.1

B. Building Blocks of Test Bench

The three main building blocks of a test bench in UVM based verification are

• Uvm_env:

It is the top level component of the verification components. uvm_env is extended from uvm_component. It is used to create and connect the uvm_components like drivers, monitors, sequencers. It can also used as sub environment in another environment.



Figure.1. Typical UVM Test Bench Architecture

- UVM test: The uvm_test class defines the test scenario for the test bench specified in the test. The test class enables configuration of the test bench and verification components.
- UVM Verification Components:

Sequencer (stimulus generator) : The sequencer generates stimulus data and passes it to a driver for execution. The uvm_sequencer is the base class of uvm class library contains all of the base functionality required to allow a sequence to communicate with a driver.

- 1. Driver : The driver drives data items to the bus following the interface protocol. The driver obtains the data items from the sequencer for execution. The UVM Class Library provides the uvm_driver base class.
- 2. Monitor: The monitor extract signal information from the bus and translates it into events, structs, and status information.
- 3. Agent : An agent has two basic operating modes
- i. Active mode : In this mode, the agent emulates a device in the system and drives DUT signals. This mode requires that the agent instantiate a driver and sequencer. A monitor also is instantiated for checking and coverage.
- ii. Passive mode : In this mode, the agent does not instantiate a driver or sequencer and operates passively. Only the monitor is instantiated and configured. This mode is used only when checking and coverage collection is desired.

III UVM CLASS HIERARCHY

The components in an UVM verification environment are derived either directly or indirectly from the uvm_component class as shown in Figure 2



Figure.2 UVM Class Hierarchy

- Build() : This phase is used to construct various child components/ports/exports and configures them.
- Connect() : This phase is used for connecting the ports/exports of the components.
- End of elaboration() : This phase is used for con gusing the components if required.
- Start of simulation() : This phase is used to print the banners and topology.
- Run() : In this phase, main body of the test is executed where all threads are forked off.

IV. RESULTS AND DISCUSSION

The results of verification components such as Master Agent and the Slave Agent of the UVM Environment are presented. According to Test Plan, the test cases are verified by developing the Verification IP for the AHB Protocol. The Test Cases are written in the form of sequences in the Sequencer using System Verilog. The sequencer drives the sequences to the driver and thereby to Score Board. In the scoreboard, the actual output is compared with the expected one. If the obtained output matches with the expected result then we conclude that the verification is completed successfully. By using the tool Questa, the Verification of AHB Components known as VC's such as Master Agent and Slave agent are done and the log files for the test cases are generated and simulated waveforms are achieved and shown in Figures 2 to 6

		Detaga						
00	/shb_b_bp/v6hck /shb_b_bp/v6hresetn /shb_b_bp/v6haddr /shb_b_bp/v6haddr	1 1 00000034 2	2	00000034	00000038	0000003c	00000030	30000
0.4	rahb th top/vit/hprot	2	2					21
-	/ahb_tb_sop/vit/membe	1			addre	ss wrapped by 4		
0.4	/ahb_tb_top/vi8htrans	2	3	2	2			2
0 1	/ahb_tb_top/vit/hburst	2	2					10.00
1	/ahb_tb_top/vi0/mordata.	ca796520	Za42495a	11465460	1764445e	\$603c1c5	[1681ce9e	\$395
D	/ahb_tb_top/vithresp	0	0					
	/ahb_th_top/vithready	1	Contraction of the	1000000	2000 - 100 -	1000000		10.00
-	ranb_tb_top/vionroata	07346906	8670640	947cdato	194000206	200804551	22424953	1145
	ranb_m_top/vonisaster	1. The second se	2000					
-	/ahb_tb_top/vibhspitx	C I					read transation at address (00000030)	
-	/ahb_tb_top/vt/hsets	×						
	/ahb_tb_top/vi0hbusreqk	0						
	/ahb_tb_top/vii/hitockx	×						
	/ahb_tb_top/vit/hgrants	1						
0.4	(abb th too/vi/hbusted.)	Abc (



* •	Maga						
/anb_tb_top/vi9hclk /ahb_tb_top/vi9hresetri /ahb_tb_top/vi9haddr - /ahb_tb_top/vi9haddr	1 1 00000034	00000034	> 20000000	00000034	200000038	C0000003c>	00000030
<pre>> /anb_tb_top/vi0/ts/ts/ > /anb_tb_top/vi0/tiprot > /anb_tb_top/vi0/twrne > /anb_tb_top/vi0/tirans > /anb_tb_top/vi0/tirans</pre>	2 1 2 2	3	address get with by 4	apped 2	3		
/ahb_tb_top/v/0hwslata	ca796b20	3812ca0e 0	35/700417	S47cdalb	34eecae5	08404551	2.6424958
 /anb_tb_top/vi0/hrdala /ahb_tb_top/vi0/hmaster /ahb_tb_top/vi0/hmastlock 	87346966 2000	d01926e1 xxxx	10399205	write tr	ansaction done	(Shr2ca0e)	\$1700417
- /anb_tb_top/vit/hsplitx	×			by wrap	4		

Figure 3: Write Transaction for wrapburst 4

	Mega		1.14						100	
/ahb_tb_top/vit/hclk	1									
/ahb_tip_lop/vil/hresetn	AT									
/ahb_tb_lop/vit/has_ch			-						-	_
/ahb_tb_tcp/vit/has_co										
	0000003c	00000028	00000026	00000030	000000034	00000038	10000003c	000000020	00000024	<10000 J
💽 🔶 /ahb_tis_top/vit/hsize	2	2								1
interpretation	3	1			3					1
/ahb_tb_top/vii/hwrite			1		_	1	1			1
🔹 🧇 /ehb_tb_top/vit/hbrans	3	3	1		2	13	1			1
Ahb_tb_topAvitAburst	4	4	1		4	1			1	
	66ee5146	d4d9b02f	Col?abee	db65b0f3	d2f3a2e1	/d261b2d6	[a0192ee1	le39b205c	d0973e72	G105
All to_top/vidthresp	0	0	1		1					
/ahb_tb_top/vit/hreadly		-	1	1	1				1	
Ahb_tb_kpAvi/Andata	68ee5146	3q7e4113	C		8e6e203		1	-		-
	×	× /						1		
/ahb_tb_top/vii/hmastlock	×		Enon ((Isinential)	- Community	and the later	-		1	-
	X		CIEVIL	Sectorences.	seque	Intial	-	1		
/ahb_tb_lop/vi/hselx	×	read trans	action				1	address	getting wrap	by8
/ahb_tb_top/vit/tb usre gx	0			-		write tra	ansaction			
/ahb_tb_tep/vit/hiockx	1×		_		_			_	_	

Figure 4: Read and Write Transaction for wrapburst 8

🐲 -	Maga																		
 /ahb_tb_top/vti/hcik /ahb_tb_top/vti/hreseth /ahb_tb_top/vti/hres_che 	S10 S11 1	1						ddress	getting	wrapp	ed by	16		1	1				
/ahb_tb_tbp/vi/has_cov	1 26 010	010	250	00	F 4	53	112	16	320	Z4	28	32	36	49	44	48	152	36 (50 2
/ahb_tb_top/vis/hprot	0000	0000				-						1					10010		
Ahb_tb_top/vit/hbrans Ahb_tb_top/vit/hburst	11 0110	0110	-								/						10/	111	
 Arb to top/vit/hwebla Arb to top/vit/hwebla Arb to top/vit/hwebla 	891cbi 86	00	5000	0000	ca7	c-3e	cbc	Ic45	Sec1	Tra53	Yada.	(de\$f	dêa	d73	GD1	koa	2091	1d98	lce71
Ahb_tb_top/vithroats	20000000	20005	(10000)	10001	010601	011100	101001	101 rea	d trans	action						/			
Pehb_tb_tbp/vit/hs pilbs	2		-						1	1	1	-	-	writ	e transa	action		-	

Figure 4: Read and Write Transaction for wrapburst 16

 /ahb_tb_top/vt0/hclk /ahb_tb_top/vt0/hreseth 	-1		1		1						1				-		-	-
<pre>/ahb_tb_top/vit/has_checks /ahb_tb_top/vit/has_coverage /ahb_tb_top/vit/haddr</pre>	-1 -1 52	56	60	264 <	152	156	060	164	52	56	160	264	152	56	160	164	52	-
/ahb_tb_top/vi0hsize /ahb_tb_top/vi0hprot /ahb_tb_top/vi0hprot	1	0			13		5		11		-	-	13	-	-	=	12	
/ahb_tb_top/vit/hitrans	2	3			2	13	1	1	2	3			12	13			2	13
/ahb_tb_top/vit/hwdata /ahb_tb_top/vit/hresp	-16356	-94	-197	158	1-183	196	1255	-139	1335	1074	11926	1633	1917	1942	1167	1-195	11196	+
 /ahb_tb_top/vit/hready /ahb_tb_top/vit/hrdata 	-1 de861	ca796	620	1		1	08a19f5	14	1		1		1076560	76		10.		
/ahb_tb_top/vit/hmaster /ahb_tb_top/vit/hmastlock	×	×	non	sequen	tial	1 6	write tr	ansacti	on	1	Te	ad tran	saction	1				-
/ahb_tb_top/vit/hsplitx	××				sequ	ential								T	-	-		
 /ahb_lb_lop/vit/hickstellx /ahb_lb_lop/vit/hickstellx 	×											-				1		
/ahb_tb_top/vit/hbusreqx_arb	×			-	-		-		-		addre	ess incr	emente	d by 4	1		-	-

Figure 5: Read and Write Transaction for incrementing burst 4

20	Maga										
/ahb_tb_top/vit/hclk	-1	JULL			ากกก		unn	hunn	hunn	hunn	LUUL
/ahb_tb_top/vit/hresetn	-1						Caddenses in	aromanting has 1	6]		
/ahb_tb_top/vif/has_checks	-1			1.8			auch ess m	cremencing by 1	0		
/ahb_tb_top/vif/has_covera	-1	101 22	1.52	1/2	the second second						Sector 2005
/ahb_tb_top/vit?haddr	52	188[92](9	6X X I	1<	52)56)60(64	68)72)76(80)	341881921960	((52)(56)6	0 64 68 72 76 80	84(88)92196	
/ahb_tb_top/vit/hsize	2	2									
/ahib_tb_top/vif/hprot	2	3			2				-		() ()
/ahb_tb_top/vii/hwrite	0			1.18		X			1		11 SI
/ahb_tb_top/vii/htrans	2	3		2	2 13		Z	12 13			D. 23
/ahb_tb_top/vit/hburst	0111	0111									U
/ahb_tb_top/vit/hwdata	011011		111	1	X X X	1 1 1 1 1 1	TIXX	1 1 1 1 1	1 1 1 1		
/ahb_tb_top/vit/hresp	00	00		-					1	-	
/ahb_tb_top/vil/hready	1								1		
/ahb_tb_top/vit/hrdata	01011	000000.	.)010111	1000	0000011011	111110001110		010101010010	0011/0010000101	q010101	-
<pre>/ahb_tb_top/vit/hmaster</pre>	XXXXX	20000		-		1	-	-	1		
/ahb_tb_top/vit/hmastlock	×			-		(read transac	uon	(wr	ite transaction)		
/ahb_lb_top/vit/hsplitx	×			1.2							11 192

Figure 6: Read and Write Transaction for incrementing burst 16

V. CONCLUSION

Verification of AHB Protocol for Single Master-Single Slave has been verified by developing the Verification IP using the UVM methodology. UVM used System Verilog implementation of standard TLM interfaces for modular communication between Verification Components. Coding of AHB components such as Master Agent and Slave Agent has been done in System Verilog using UVM. The Verification IP s developed using tool Questa and it is sure that developed VIP is also compatible with other tools such as Cadence, Synopsys etc. The developed Verification IP can be used in the verification of SoCs.

VI. ACKNOWLEDGEMENT

We the authors of this paper would like to great fully acknowledge with thanks to Head of ECE department Prof. C V Ravishankar and Gaurav Jalan Senior Manager, SmartPlay Technologies (I) Pvt.Ltd.,Bangalore. for constant encouragement and support for enabling us to submit this paper.

VII. REFERENCES

- [1] Jack Erickson, "TLM-Driven Design and Verification Time For a Methodology Shift", Cadence Design Systems, Inc.
- [2] Rath A.W, Esen.V and Ecker.W, "A transaction –oriented UVM-based library for verification of analog behavior "Publication Year: 2014 Page(s): 806 – 811
- [3] Stuart Sutherland, Don Mills, "Synthesizing System Verilog Busting the myth that System Verilog is only for Verification"
- [4] Soo-Yun Hwang and Kyoung-Sun Jhang , "An Improved Implementation Method Of AHB BusMatrix" SOC Conference 2005, Proceedings, IEEE International pp. 211-214
- [5] Mulani, "Level Verification Using SystemVerilog" Emerging Trends in Engineering and Technology (ICETET), 2009 2nd International Conference on 16-18 Dec.2009 pp.378-380
- [6] Pockrandt, M, Herber, P and Glesner, S, "Model checking a System C/TLM design of the AMBA AHB Protocol "Embedded Systems for Real-Time Multimedia (ESTIMedia), 2011 9th IEEE Symposium on 13-14 Oct.2011 pp.66 – 75
- [7] IEEE Draft Standard for System Verilog Unified Hardware Design, Specification and Verification Language, IEEE P1800/DS, February 2012 pp.1-1304
- [8] Young-Nam Yun, Jae-Beom Kim, Nam-Do Kim and Byeong Min, "Beyond UVM for practical SoC verification" SoC Design Conference (ISOCC), 2011 International pp. 158-162
- [9] Keaveney.Martin,McMahon.Antony, O'Keeffe.Niall, Keane.Kevin and O.Reilly James "The Development of advanced verification environments using System Verilog" Signals and Systems Conference,208.(ISSC 2008),IET Irish pp.325-330
- [10] Universal Verification Methodology 1.1 User's Guide May 18, 2011 http://www.accellera.org

AUTHOR'S PROFILE



 Tejaswini H N Currently working as Assistant Professor in ECE Dept of Sambhram Institute of Technology, Bangalore.
 Specialization : VLSI Design and Testing.



2. Revati Bothe Team Leader in SmartPlay Technologies (I) Pvt. Ltd , Bangalore



 Prof.Ravishankar C V Head of the Dept. Department of Electronics & Communication Engineering, Sambhram Institute of Technology, Bangalore.