Original scientific paper

https://doi.org/10.33180/InfMIDEM2018.402

Informacije MIDEM
Journal of Microelectronics,
Electronic Components and Materials
Vol. 48, No. 4(2018), 205 – 211

Implementation of VIP for bus interface logic of 32-bit processor using System Verilog

D. David Neels Ponkumar[1], P. Jagatheeswari[1], T.S.Arun Samuel[2]

[1,2]Dept. of Electronics and Communication Engineering, Dr. Sivanthi Aditanar College of Engineering, Tiruchendur, Tamil Nadu, India.

[2]Dept. of Electronics and Communication Engineering, National Engineering College, Kovilpatti, India

Abstract: A verification environment to verify an ARM-based SoC is proposed in this work. This work introduces the design of a Verification Intellectual Property (VIP) of Advanced Microcontroller Bus Architecture (AMBA). AMBA protocols are today the best standards for 32-bit processor because they are well documented and can be used without royalties. The VIP provides Coverage Driven Verification (CDV) which significantly reduces the design verification time. The code coverage verification of the AHB bus master, Icache controller, Dcache controller and APB peripherals such as APB bridge, timer, UART, and ACE is done in this work. The test cases done for the APB peripherals are ACE with the mil_std_protocol, Timers for generation of interrupt and watchdog reset, UART for transmitting and receive messages, and interrupt registers for Reading and Write. The functional verification of AMBA is carried out using the Mentor Graphics Questasim tool with the system Verilog language.

Keywords: Verification Intellectual Property; AMBA; Coverage Driven Verification; timer; ACE; UART; system Verilog

Uporaba VIP za vmesnik logičnega vodila 32-bitnega procesorja s uporabo System Verilog

Izvleček: V članku je predlagano okolje preverjanja SoC na osnovi ARMa. Struktura uvaja verifikacijo intelektualne lastnine (VIP) na napredni arhitekturi vodila mikrokontrolerja (AMBA). AMB protokoli so danes najbolj standardni protokoli na 32- bitni procesorjih, saj so dobro dokumentirani in brez avtorskih zaščit. VIP uporablja CVD, ki močno zmanjša čas verifikacije. V tem delu je predstavljena verifikacija AHB master vodila, Icache in Dcache kontrolerjev ter APB perifernih enot, kot so časovnik, UART in ACE. Testni primeri za APB periferne enote so ACE z mil_std_protocol, časovniki za generacijo prekinjanj in resetiranja kontrolne enote (watchdog), UART za pošiljanje in sprejemanje sporočil in prekinitveni registri za zapis in branje. Funkcionalno preverjanje je izvedeno s pomočjo orodja Mentor Graphics Questasim v jeziku Verilog.

Ključne besede: verifikacija intelektualne lastnine; AMBA; verifikacija; časovnik; ACE; UART; Verilog

** Corresponding Author's e-mail: david26571@gmail.com*

1 Introduction

With the continued progression of chip geometries to ever smaller sizes, designers are finding themselves with a wealth of available gates in which to create their latest designs [1-4]. With design from scratch entirely out of the question, designers now build these systems with off-the-shelf IP blocks that are pre-designed and verified, helping them meet their goals of differentiation, cost control and time to market.VIP blocks are well-tested simulation models of industry-standard buses and protocols that generate and respond to stimulus and check protocol rule adherence. VIP reduces system verification time and improves quality.

VIP design of AMBA AXI bus is done in the previous works [5-9][11], But the VIP for Dcache controller, Icache controller and APB peripherals such as APB bridge, timer, UART, and ACE is done for the first time. The implemented VIP finds application in the realization of onboard computers for navigation, guidance, and control processing in-flight applications as well as for general purpose processing applications.

The verification environment is managed with Questa Sim Simulator ver.10.0, test bench and SVA in System Verilog HDL and DUT in VHDL. Separate assertion files in system Verilog are bound with the corresponding test benches to validate design specifications.

2 Proposed system design

2.1 AMBA Bus

The Advanced Microcontroller Bus Architecture specification defines an on-chip communications standard regarding bus protocols for communication between various system devices and peripherals. AMBA is a registered trademark of ARM Limited and is an open standard, on-chip interconnect specification for the connection and management of functional blocks in a System-on-Chip (SoC) [2]. This work provides Coverage Driven Verification (CDV) for the implementation of Verification Intellectual Property (VIP) for the AMBA bus. In this paper, the verification of the APB peripherals such as APB Bridge, timer, UART, and ACE is done.

2.2 AMBA Architecture

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. A typical AMBA Architecture is shown in figure 1[2].

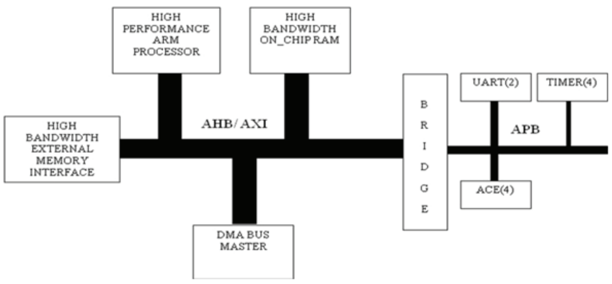

HIGH PERFORMANCE ARM PROCESSOR
HIGH BANDWIDTH ON_CHIP RAM
HIGH BANDWIDTH EXTERNAL MEMORY INTERFACE
AHB/AXI
DMA BUS MASTER
B R I D G E
UART(2)
TIMER(4)
APB
ACE(4)

Figure 1: AMBA Architecture

2.3 AHB Master

AHB_master generates chip select signals, the external memory address for instruction and data memory. The FSM of AHB_master is shown in figure 2.

The FSM works on clk_x2_pos. Wait states are added for external instruction and data memory access. They are selected from the Memory Configuration Register depending on the memory bank accessed.

Instr_state: It is selected when instruction access is requested by fetch stage of the pipeline when instruction cache is disabled. Chip select signals and address for external memory access are generated. The instruction read from external memory is sent to the fetch stage of the pipeline. FSM waits in the instr_state till the specified wait states are over and transitions back to idle state.

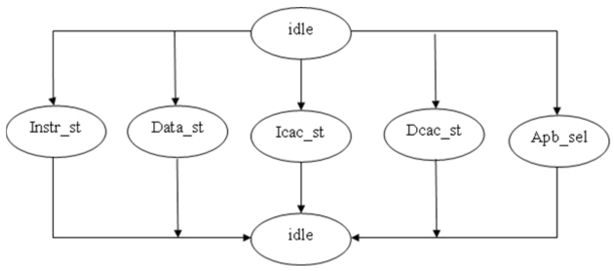


Figure 2: FSM of AHB_master

Data_state: It is selected when data memory access is requested by the memory stage of the pipeline when data cache is disabled. Chip select signals for load/store and address for external memory access are generated. For load, the data read from external memory is sent to the memory stage of the pipeline. For the store, the data from the memory stage of the pipeline is sent to the external memory. FSM waits in the data_state till the specified wait states are over and transitions back to idle state.

ICache_state: This state is selected if Icache is enabled and a cache miss occurs. FSM2 handling miss in icache_controller.vhd is active here. The chip select signals are generated when fsm2 in icache_controller.vhd is in wait_for_mfc (wait_for_mfc_icache = 1) state. FSM transitions to idle when icache access is complete (icache_access_complete=1, from icache_controller.vhd)

DCache_state: This state is selected if Dcache is enabled and a cache miss occurs. FSM2 handling miss in dcache_controller.vhd is active here. The chip select signals are generated when fsm2 in dcache_controller.vhd is in wait_for_mfc (wait_for_mfc_dcache = 1) state. FSM transitions to idle when dcache access is complete (dcache_access_complete=1, from dcache_controller.vhd).

APB_sel: If a memory mapped peripheral is selected apb_sel state is encountered. FSM transitions to idle when hready_apb = '1' from apb_bridge.vhd, the external memory address for instruction access, is generated in instr_state or icache_state. The external memory address for data access is generated in data_state or dcache_state.

2.4 Icache controller

The instruction cache controller is used to cache copies of frequently accessed instructions, thus eliminating the program memory access bottleneck. The cache controller receives instruction read request an address from the Fetch stage. Depending on hit/miss the cache controller supplies instruction from the on-chip cache or reads from external program memory via AHB and provides instruction to the pipeline. Till external memory access is complete, the pipeline is stalled.

- 32KB size.Can store 8K instructions
- Two way set associative
- Uses Least Recently Used (LRU) algorithm for block replacement
- Block size: 4 words

LRU Replacement Algorithm
The LRU (Least Recently Used) algorithm is implemented for the two-way associative cache conFigure uration. This algorithm selects a block for replacement based on its usage, thus benefiting from the temporal locality principle. A single bit is added as part of the tag entry in the tag ram. Whenever a tag match is found in a block, the LRU bit of that block is cleared and the LRU bit of the second block in the set is made '1'.When a block is to be evicted, the tag entry in the set which has its LRU bit set to 1 is selected.

2.5 Dcache controller

The data cache controller caches frequently used data items. The data cache implements copy back with write allocate on a write miss. The dirty blocks are written back to external memory only when they need to be evicted.

- Size = 32KB
- Cache line size = 4words
- Uses an LRU replacement algorithm
- Copy back policy
- Write allocate on a write miss
- Two way set associative

Hardware Organization: Data cache is identical to the instruction cache, except that each tag ram location has an additional bit called the dirty bit, which indicates whether the cache block had been modified during its cache residency. Thus each tag ram array has 23 bits * 1024 locations.

Address Decoding: This is identical to two-way associative instruction cache implementation. The 10-bit address is used to index the tag ram arrays. The two-word locator bits are appended to the 10-bit set address to address the cache ram arrays.

Copy Back Architecture: When there is a store request from the memory stage of the pipeline, the corresponding cache array entry is updated in the cache if it already exists in the cache. If it is a cache miss, the block which includes the address requested by the store operation is brought into the cache from the external memory (write allocate on a write miss) and the required location is updated. This policy is especially beneficial when frequent writes to a memory location (store instruction) occur since there is no need to access external memory once the word is in the cache. This policy uses the memory bandwidth more efficiently compared to the write through policy wherein each store location writes to the external memory.

Data Cache Parity Error detection of cache tags and data is implemented using two parity bits per tag and 4-byte data sub-block. The tag parity is generated from the tag value, LRU, dirty and the valid bits. The data parity is derived from the sub-block data. The parity bits are written simultaneously with the associated tag or sub-block and checked on each access. The two parity bits corresponding to the parity of odd and even data (tag) bits.

2.6 APR Bridge

APB bridge acts as the master for the APB slaves – four ACE, two UART, interrupt registers and four timers. The APB bridge converts the AHB signals from the bus master to corresponding signals in APB. Memory configuration register specifies the no. of wait states for different memory banks and internal RAM.

Select signal for selecting a memory mapped register is generated by decoding the address (haddr). Hwrite=1 indicates a register write operation. Hwrite=0 indicates a register read operation. The ACE, UART, timer, interrupt, and processor configuration registers are read or written in a single clock cycle, and hready_apb is asserted. For ace access hready_apb is asserted when already signal is asserted. The Processor Configuration Register is shown in table 1.

Table.1: Processor Configuration Register

Bit	Description
31:13	Unused
12	Interrupt Enable
11	Watchdog enable
10	SECDED enable for Internal RAM

9	SECDED enable for external memory bank 6
8	SECDED enable for external memory bank 5
7	SECDED enable for external memory bank 4
6	SECDED enable for external memory bank 3
5	SECDED enable for external memory bank 2
4	SECDED enable for external memory bank 1
3	SECDED enable for external memory bank 0
2	SECDED enable for Internal Registers
1	Instruction Cache Enable
0	Data Cache Enable

2.6 AMBA Advanced Peripheral Bus

The APB Bridge is the only bus master on the AMBA APB. Also, the APB Bridge is also a slave on the higher-level system bus (for example AHB). The bridge unit converts system bus transfers into APB transfers and performs the following functions:

- Latches the address and holds it valid throughout the transfer.
- Decodes the address and generates a peripheral select, PSELx. Only one select signal can be active during a transfer.
- Drives the data onto the APB for a write transfer.

2.7 Code Coverage Analysis

Code coverage is a verification technology is used to recognize what code has been executed (figure 3). It has to be checked only after the simulation part. If the design may look like a good design, but the problem is that it can contain unknown bugs. It is hardly possible to know the verification is functionally correct, with cent percent certainty and all of the test benches simulate successfully. The main objective of the code coverage is to find out which code has to forget to exercise in the design.

The term *"test bench"* specifies the stimulus used to initiate a predestined input sequence for the design and to examine its response. The test bench describes the stimulus for the DUT along with its responsibility for the outputs. Here, the test bench is written in SystemVerilog with a preset input sequence, and they may be included with external data files. The main task of the test bench is that to verify what input patterns to provide to the design and what is the anticipated throughput of a properly working design. If the test bench was not exactly executed, it should be returned in the design. So, code coverage technology is used for the 100% certainty. It can be classified into four categories. They are Statement coverage, Path coverage (Branch and Toggle coverage), Expression coverage, FSM coverage.

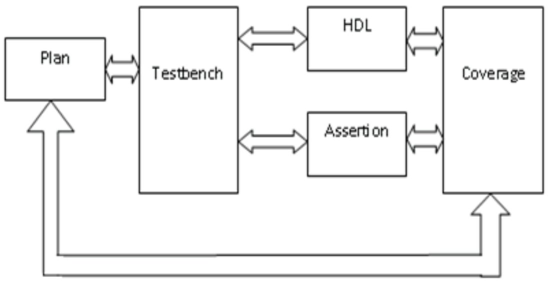
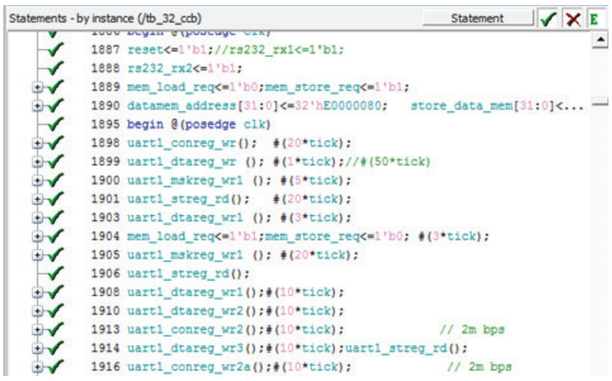


Figure 3: Verification plan

Statement coverage: It is also known as Block coverage, where the block is series of statements. If a single statement is executed, all of the statements in the block will be executed. By the verification suite, it measures how much of the total line of code was executed. Figure 4 shows the analysis window for the statement coverage verification. It will be generated after the simulation part. The tick (√) mark indicates that statement code which includes in the DUT are functionally correct. If shows (x) mark, indicate that design is functionally incorrect. It will quickly identify, and we can browse which statements that were not executed.



```
reset<=1'b1;//rs232_rx1<=1'b1;
rs232_rx2<=1'b1;
mem_load_req<=1'b0;mem_store_req<=1'b1;
datamem_address[31:0]<=32'hE0000080;   store_data_mem[31:0]<...
begin @(posedge clk)
uart1_conreg_wr();  #(20*tick);
uart1_dtareg_wr (); #(1*tick);//#(50*tick)
uart1_mskreg_wr1 (); #(5*tick);
uart1_streg_rd();   #(20*tick);
uart1_dtareg_wr1 (); #(3*tick);
mem_load_req<=1'b1;mem_store_req<=1'b0; #(3*tick);
uart1_mskreg_wr1 (); #(20*tick);
uart1_streg_rd();
uart1_dtareg_wr1();#(10*tick);
uart1_dtareg_wr2();#(10*tick);
uart1_conreg_wr2();#(10*tick);             // 2m bps
uart1_dtareg_wr3();#(10*tick);uart1_streg_rd();
uart1_conreg_wr2a();#(10*tick);             // 2m bps
```

Figure 4: Statement coverage

FSM: It is usually, coded using a choice in a case statement, the unvisited state identified with uncovered statements. During the verification time, it clearly or correctly identifies the state transitions. Figure 5 shows the bubble diagram for FSM. It indicates that state transitions of decoder sections.

Branch and Toggle coverage: A signal is considered to have fully toggled when it has experienced at least one rising edge and at least one falling edge during the simulation. It has been found from the simulated results that the coverage windows, which indicate all the branch and toggles present in the design of AMBA was functionally, correct.

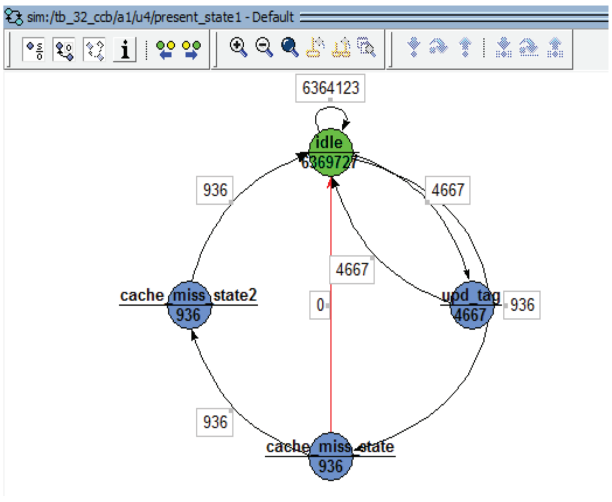


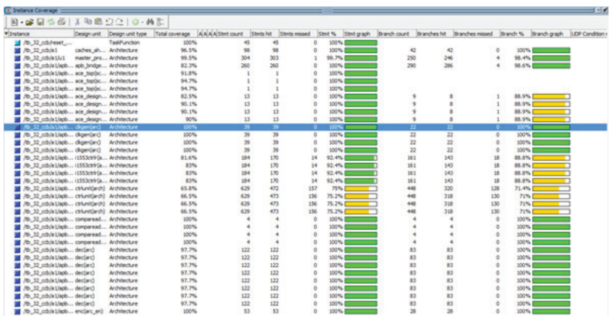
Figure 5: FSM coverage

Transition Coverage: Transition coverage measures the presence or occurrence of sequences of values. Transition coverage can involve more than two consecutive values of the same coverage point. However, the number of possible bins grows factorially with the number of transition states. Mechanically, transition coverage is identical to coverage points. Specific values are sampled at specific locations at specific points in time with specific bins. Table 2 and 3 show the Assertions for the Timer module and the UART module.

3 Results and discussion

The functional verification of AMBA is carried out using the Mentor Graphics Questasim tool in the code coverage mode with the SystemVerilog language. The SystemVerilog simulation is performed to verify the AMBA design by using the VIP. Functional integrity of DUT is checked by using Assertions and cover groups along with necessary test inputs.

Figure 6 shows the instance coverage analysis of the AMBA peripherals. The instance analysis is done a state of the peripherals during each instance. It provides the coverage of the individual modules in the AMBA peripherals. This window analyzes coverage statistics for each instance in a flat and non-hierarchical view. The window contains the same code coverage statistics columns as in the Files and Structure windows.


Instance Coverage

Instance	Design unit	Design unit type	Total coverage	A	A	A	A	Stmt count	Stmts hit	Stmts missed	Stmt %	Stmt graph	Branch count	Branches hit	Branches missed	Branch %	Branch graph	UDP Condition r
/tb_32_ccb/reset_...		TaskFunction	100%					45	45	0	100%	🟩						
/tb_32_ccb/a1	caches_ah...	Architecture	96.5%					98	98	0	100%	🟩	42	42	0	100%	🟩	
/tb_32_ccb/a1/u1	master_pro...	Architecture	99.5%					304	303	1	99.7%	🟩	250	246	4	98.4%	🟩	
/tb_32_ccb/a1/apb...	apb_bridge...	Architecture	82.3%					260	260	0	100%	🟩	290	286	4	98.6%	🟩	
/tb_32_ccb/a1/apb...	ace_top(sc...	Architecture	91.8%					1	1	0	100%	🟩						
/tb_32_ccb/a1/apb...	ace_top(sc...	Architecture	94.7%					1	1	0	100%	🟩						
/tb_32_ccb/a1/apb...	ace_top(sc...	Architecture	94.7%					1	1	0	100%	🟩						
/tb_32_ccb/a1/apb...	ace_design...	Architecture	82.5%					13	13	0	100%	🟩	9	8	1	88.9%	🟨	
/tb_32_ccb/a1/apb...	ace_design...	Architecture	90.1%					13	13	0	100%	🟩	9	8	1	88.9%	🟨	
/tb_32_ccb/a1/apb...	ace_design...	Architecture	90.1%					13	13	0	100%	🟩	9	8	1	88.9%	🟨	
/tb_32_ccb/a1/apb...	ace_design...	Architecture	90%					13	13	0	100%	🟩	9	8	1	88.9%	🟨	
/tb_32_ccb/a1/apb...	clkgen(arc)	Architecture	100%					39	39	0	100%	🟩	22	22	0	100%	🟩	
/tb_32_ccb/a1/apb...	clkgen(arc)	Architecture	100%					39	39	0	100%	🟩	22	22	0	100%	🟩	
/tb_32_ccb/a1/apb...	clkgen(arc)	Architecture	100%					39	39	0	100%	🟩	22	22	0	100%	🟩	
/tb_32_ccb/a1/apb...	clkgen(arc)	Architecture	100%					39	39	0	100%	🟩	22	22	0	100%	🟩	
/tb_32_ccb/a1/apb...	i1553ctrlr(a...	Architecture	81.6%					184	170	14	92.4%	🟩	161	143	18	88.8%	🟨	
/tb_32_ccb/a1/apb...	i1553ctrlr(a...	Architecture	83%					184	170	14	92.4%	🟩	161	143	18	88.8%	🟨	
/tb_32_ccb/a1/apb...	i1553ctrlr(a...	Architecture	83%					184	170	14	92.4%	🟩	161	143	18	88.8%	🟨	
/tb_32_ccb/a1/apb...	i1553ctrlr(a...	Architecture	83%					184	170	14	92.4%	🟩	161	143	18	88.8%	🟨	
/tb_32_ccb/a1/apb...	ctrlunit(arch)	Architecture	65.8%					629	472	157	75%	🟨	448	320	128	71.4%	🟨	
/tb_32_ccb/a1/apb...	ctrlunit(arch)	Architecture	66.5%					629	473	156	75.2%	🟨	448	318	130	71%	🟨	
/tb_32_ccb/a1/apb...	ctrlunit(arch)	Architecture	66.5%					629	473	156	75.2%	🟨	448	318	130	71%	🟨	
/tb_32_ccb/a1/apb...	ctrlunit(arch)	Architecture	66.5%					629	473	156	75.2%	🟨	448	318	130	71%	🟨	
/tb_32_ccb/a1/apb...	comparead...	Architecture	100%					4	4	0	100%	🟩	4	4	0	100%	🟩	
/tb_32_ccb/a1/apb...	comparead...	Architecture	100%					4	4	0	100%	🟩	4	4	0	100%	🟩	
/tb_32_ccb/a1/apb...	comparead...	Architecture	100%					4	4	0	100%	🟩	4	4	0	100%	🟩	
/tb_32_ccb/a1/apb...	comparead...	Architecture	100%					4	4	0	100%	🟩	4	4	0	100%	🟩	
/tb_32_ccb/a1/apb...	dec(arc)	Architecture	97.7%					122	122	0	100%	🟩	83	83	0	100%	🟩	
/tb_32_ccb/a1/apb...	dec(arc)	Architecture	97.7%					122	122	0	100%	🟩	83	83	0	100%	🟩	
/tb_32_ccb/a1/apb...	dec(arc)	Architecture	97.7%					122	122	0	100%	🟩	83	83	0	100%	🟩	
/tb_32_ccb/a1/apb...	dec(arc)	Architecture	97.7%					122	122	0	100%	🟩	83	83	0	100%	🟩	
/tb_32_ccb/a1/apb...	dec(arc)	Architecture	97.7%					122	122	0	100%	🟩	83	83	0	100%	🟩	
/tb_32_ccb/a1/apb...	dec(arc)	Architecture	97.7%					122	122	0	100%	🟩	83	83	0	100%	🟩	
/tb_32_ccb/a1/apb...	dec(arc)	Architecture	97.7%					122	122	0	100%	🟩	83	83	0	100%	🟩	
/tb_32_ccb/a1/apb...	dec(arc)	Architecture	97.7%					122	122	0	100%	🟩	83	83	0	100%	🟩	
/tb_32_ccb/a1/apb...	enc(arc_en)	Architecture	100%					53	53	0	100%	🟩	28	28	0	100%	🟩	

Figure 6: Instance coverage

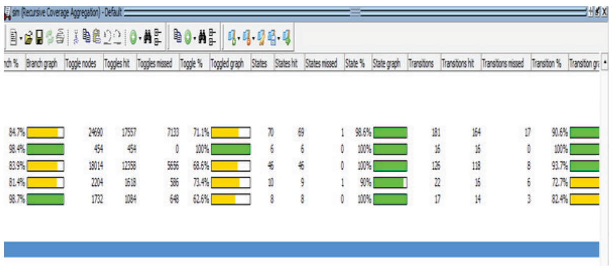
Table.2: Assertions for the Timer Module

Assertion	Description
Timer_hreset_prdata	prdata_timer=32’h00000000 when hreset is asserted
Timer_hreset_intr	Intr_timer1, Intr_timer2, Intr_timer3, Intr_timer4=0 when hreset is asserted
Timer1 underflow	When prdata_timer<=32’h00000000 and haddr <= 32’hE0000048 then intr_timer1=0
Timer2 underflow	When prdata_timer<=32’h00000000 and haddr <= 32’hE0000054 then intr_timer2=0
Timer3 underflow	When prdata_timer<=32’h00000000 and haddr <= 32’hE0000060 then intr_timer3=0
Timer4 underflow	When prdata_timer<=32’h00000000 and haddr <= 32’hE000006c then intr_timer4=0
Timer1_disable	When prdata_timer<=32’h00000000 and haddr <= 32’hE0000040 then intr_timer1=1
Timer2_disable	When prdata_timer<=32’h00000000 and haddr <= 32’hE000004c then intr_timer2=1
Timer3_disable	When prdata_timer<=32’h00000000 and haddr <= 32’hE0000058 then intr_timer3=1
Timer4_disable	When prdata_timer<=32’h00000000 and haddr <= 32’hE0000064 then intr_timer4=1

Table.3: Assertions for the UART Module

Assertion	Description
Uart1_transmit enable	When hreset is asserted and haddr==32’hE0000020 && hwdata==24’h0000A5 then rs232_tx1==1’b1
Uart1_recieve enable	When hreset is asserted and haddr==32’hE0000024 && rs232_rx1==1’b1 then hrdata_apb==32’h000000A5
Uart2_transmit enable	When hreset is asserted and haddr==32’hE0000080 && hwdata==24’h0000A5 then rs232_tx2==1’b1
Uart2_recieve enable	When hreset is asserted and haddr==32’hE0000084 && then rs232_rx2==1’b1 then hrdata_apb==32’h000000A5

Figure 7 shows the coverage aggregation analysis of the AMBA peripherals. The coverage aggregation analysis shows, the state of the toggle graph, state graph, and the transition graph during the coverage analysis.

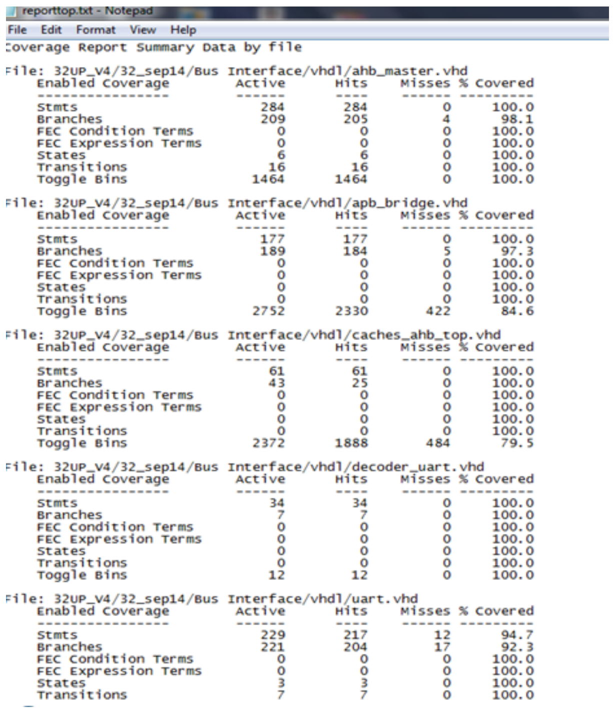


nch %	Branch graph	Toggle nodes	Toggles hit	Toggles missed	Toggle %	Toggled graph	States	States hit	States missed	State %	State graph	Transitions	Transitions hit	Transitions missed	Transition %	Transition gr
84.7%	🟨	24690	17557	7133	71.1%	🟨	70	69	1	98.6%	🟩	181	164	17	90.6%	🟩
98.4%	🟩	454	454	0	100%	🟩	6	6	0	100%	🟩	16	16	0	100%	🟩
83.9%	🟨	18014	12358	5656	68.6%	🟨	46	46	0	100%	🟩	126	118	8	93.7%	🟩
81.4%	🟨	2204	1618	586	73.4%	🟨	10	9	1	90%	🟩	22	16	6	72.7%	🟨
98.7%	🟩	1732	1084	648	62.6%	🟨	8	8	0	100%	🟩	17	14	3	82.4%	🟨

Figure 7: Coverage aggregation

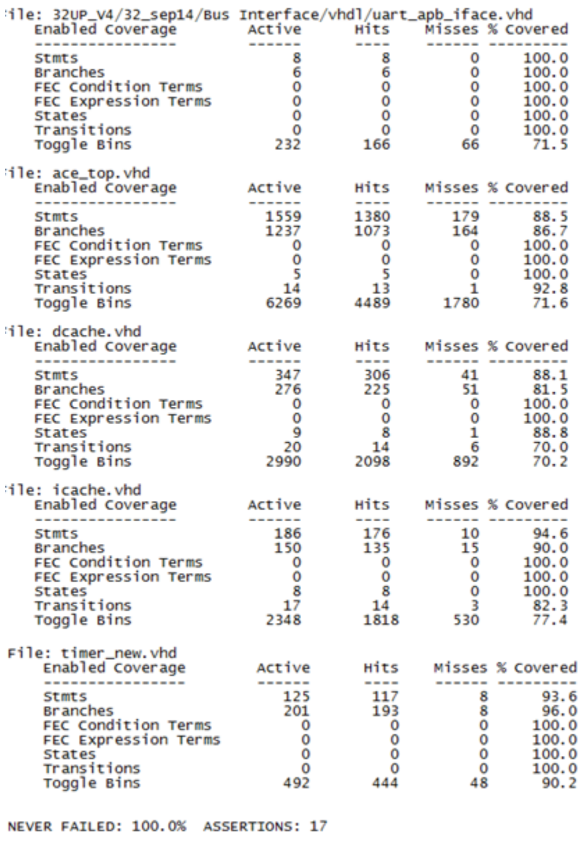
The Assertion Coverage of the AMBA peripherals is also analyzed using a simulation tool. Assertion/properties provide a clear indication to the VIP module. The assertions can be set as true or false throughout the VIP module. It has been found from the simulated results that, the total of 17 assertions are used for the analysis and each of them has been satisfied throughout the analysis. Hence, the assertion coverage of the proposed VIP module is 100 %.

Figure 8 and 9 shows the coverage analysis report of the AMBA peripherals. The coverage analysis depicts the coverage obtained by the proposed VIP module for each of the AMBA peripherals. For the coverage analysis, the VIP considered seven aspects for each module, and they are the statement, Branches, FEC condition trees, FEC expression trees, States, transition, and toggle bin. In total, proposed VIP has achieved coverage value of 91.4%, for each AMBA peripherals. Besides, the total 17 assertions provided for the analysis has never failed throughout the process,



```
Coverage Report Summary Data by file

File: 32UP_V4/32_sep14/Bus Interface/vhdl/ahb_master.vhd
    Enabled Coverage          Active      Hits    Misses % Covered
    ----------------          ------      ----    ------ ---------
    Stmts                        284       284         0     100.0
    Branches                     209       205         4      98.1
    FEC Condition Terms            0         0         0     100.0
    FEC Expression Terms           0         0         0     100.0
    States                         6         6         0     100.0
    Transitions                   16        16         0     100.0
    Toggle Bins                 1464      1464         0     100.0

File: 32UP_V4/32_sep14/Bus Interface/vhdl/apb_bridge.vhd
    Enabled Coverage          Active      Hits    Misses % Covered
    ----------------          ------      ----    ------ ---------
    Stmts                        177       177         0     100.0
    Branches                     189       184         5      97.3
    FEC Condition Terms            0         0         0     100.0
    FEC Expression Terms           0         0         0     100.0
    States                         0         0         0     100.0
    Transitions                    0         0         0     100.0
    Toggle Bins                 2752      2330       422      84.6

File: 32UP_V4/32_sep14/Bus Interface/vhdl/caches_ahb_top.vhd
    Enabled Coverage          Active      Hits    Misses % Covered
    ----------------          ------      ----    ------ ---------
    Stmts                         61        61         0     100.0
    Branches                      43        25         0     100.0
    FEC Condition Terms            0         0         0     100.0
    FEC Expression Terms           0         0         0     100.0
    States                         0         0         0     100.0
    Transitions                    0         0         0     100.0
    Toggle Bins                 2372      1888       484      79.5

File: 32UP_V4/32_sep14/Bus Interface/vhdl/decoder_uart.vhd
    Enabled Coverage          Active      Hits    Misses % Covered
    ----------------          ------      ----    ------ ---------
    Stmts                         34        34         0     100.0
    Branches                       7         7         0     100.0
    FEC Condition Terms            0         0         0     100.0
    FEC Expression Terms           0         0         0     100.0
    States                         0         0         0     100.0
    Transitions                    0         0         0     100.0
    Toggle Bins                   12        12         0     100.0

File: 32UP_V4/32_sep14/Bus Interface/vhdl/uart.vhd
    Enabled Coverage          Active      Hits    Misses % Covered
    ----------------          ------      ----    ------ ---------
    Stmts                        229       217        12      94.7
    Branches                     221       204        17      92.3
    FEC Condition Terms            0         0         0     100.0
    FEC Expression Terms           0         0         0     100.0
    States                         3         3         0     100.0
    Transitions                    7         7         0     100.0
```

Figure 8: Coverage report



```
File: 32UP_V4/32_sep14/Bus Interface/vhdl/uart_apb_iface.vhd
    Enabled Coverage          Active      Hits    Misses % Covered
    ----------------          ------      ----    ------ ---------
    Stmts                          8         8         0     100.0
    Branches                       6         6         0     100.0
    FEC Condition Terms            0         0         0     100.0
    FEC Expression Terms           0         0         0     100.0
    States                         0         0         0     100.0
    Transitions                    0         0         0     100.0
    Toggle Bins                  232       166        66      71.5

File: ace_top.vhd
    Enabled Coverage          Active      Hits    Misses % Covered
    ----------------          ------      ----    ------ ---------
    Stmts                       1559      1380       179      88.5
    Branches                    1237      1073       164      86.7
    FEC Condition Terms            0         0         0     100.0
    FEC Expression Terms           0         0         0     100.0
    States                         5         5         0     100.0
    Transitions                   14        13         1      92.8
    Toggle Bins                 6269      4489      1780      71.6

File: dcache.vhd
    Enabled Coverage          Active      Hits    Misses % Covered
    ----------------          ------      ----    ------ ---------
    Stmts                        347       306        41      88.1
    Branches                     276       225        51      81.5
    FEC Condition Terms            0         0         0     100.0
    FEC Expression Terms           0         0         0     100.0
    States                         9         8         1      88.8
    Transitions                   20        14         6      70.0
    Toggle Bins                 2990      2098       892      70.2

File: icache.vhd
    Enabled Coverage          Active      Hits    Misses % Covered
    ----------------          ------      ----    ------ ---------
    Stmts                        186       176        10      94.6
    Branches                     150       135        15      90.0
    FEC Condition Terms            0         0         0     100.0
    FEC Expression Terms           0         0         0     100.0
    States                         8         8         0     100.0
    Transitions                   17        14         3      82.3
    Toggle Bins                 2348      1818       530      77.4

File: timer_new.vhd
    Enabled Coverage          Active      Hits    Misses % Covered
    ----------------          ------      ----    ------ ---------
    Stmts                        125       117         8      93.6
    Branches                     201       193         8      96.0
    FEC Condition Terms            0         0         0     100.0
    FEC Expression Terms           0         0         0     100.0
    States                         0         0         0     100.0
    Transitions                    0         0         0     100.0
    Toggle Bins                  492       444        48      90.2

NEVER FAILED: 100.0%  ASSERTIONS: 17
```

Figure 9: Coverage report (Continued)

4 Conclusion

In this work, verification environment for Caches AHB_Top, AHB Master, Instruction cache, Data cache the free running counters/timers of APB, ACE Controller, UART and APB Bridge is done. The verification scenario includes Read and Write transfer phases of the APB which are verified with the values of the count and reset. The test cases done for the APB peripherals are ACE with the mil_std_protocol, Timers for generation of interrupt and watchdog reset, UART for transmitting and receive messages, and interrupt registers for Reading and Write. From the simulation results, the state of operation of the AMBA is obtained. The overall coverage obtained for the AMBA peripherals is 91.4%, and the Assertion Coverage is 100%. This work can be extended by creating the verification environment for AMBA with the UVM library. Questasim collects all coverage data, code coverage, assertions, formal, and functional coverage into a single highly efficient Unified Coverage Database (UCDB) and makes them available in real-time within the test bench.

5 Reference

1. "AMBA AHB"- specification by ARM limited.
2. AMBA timer data sheet, http://www.arm.com/.
3. ARM, "AMBA specification overview," http://www.arm.com/
4. Chris spear, SystemVerilog for Verification A Guide To Learning the Test Bench Language Features, 2nd edition.
5. Golla Mahesh, Sakthivel S M "Verification IP for an AMBA-AXI protocol using system Verilog", International Journal of Applied Engineering Research, Vol.12, No.17, pp. 6534-6541,November 2017.
6. Golla Mahesh and Sakthivel.S.M, "Verification IP for an AMBA-AXI Protocol using System Verilog" International Journal of Engineering Research and General Science, vol.3, no.1, pp.792-799, February, 2015.
7. Han Ke, Deng Zhongliang, Shu Qiong "Verification of AMBA bus model using SystemVerilog" The Eighth International Conference on Electronic Measurement and Instruments ICEMI' 2007. https://doi.org/10.1109/ICEMI.2007.4350567
8. Heli Shah P, Chinmay modi P, Bhargav Tarpara P "Design & Implementation of Advance Peripheral Bus Protocol," International journal of scientific engineering and applied science (IJSEAS) vol.1, no.3, June 2015.
9. Manu B, Prabhavathi P, "Design and Implementation of AMBA ASB APB bridge" Proceedings of 2013 International conference on fuzzy theory and its application national Taiwan University of science and technology, pp. 6-8, December 2013. https://doi.org/10.1109/iFuzzy.2013.6825442
10. Questa sim user's manual, by Mentor Graphics.
11. Richa Sinha, Akhilesh Kumar, and Archanakumari Sinha "Verification analysis of AHB-LITE protocol with coverage," International Journal of advances in Engineering & technology, Vol.2, No.1, pp.121-128, January 2012.

Arrived: 25. 04. 2018
Accepted: 03. 10. 2018