On Compliance Test of On-Chip Bus for SOC

Hue-Min Lin, Chia-Chih Yen, Che-Hua Shih and Jing-Yang Jou

Department of Electronics Engineering,
National Chiao Tung University
Hsinchu 300, Taiwan, R.O.C.
Tel : 886-3-5726111
Fax : 886-3-5710580
e-mail : {hmlin, Jackr, matar, jyjou}@eda.ee.nctu.edu.tw

Abstract - In this paper, we employ a monitor-based approach for on-chip bus (OCB) compliance test. To describe the OCB protocols, we propose a FSM model, which can help to extract the necessary properties systematically and verify the data part of a bus transfer efficiently. To demonstrate our methodology, we illustrate two OCB protocols, WISHBONE and AMBA AHB, as the study cases. The experimental results show that we can verify the OCB protocols efficiently and detect the design errors when tests fail.

I Introduction

The OCB architecture is a popular method for rapid building of System-on-chip (SOC) devices by integrating a number of different IP components onto it. By providing a standard interface, the OCB specification enables the communication among a set of IP components. However, components of a SOC device are usually designed in different places by different designers. Due to the increasing complexity of the overall designs, it is very important for each designer to guarantee his IP components can function correctly after being integrated into a SOC design. Hence, the compliance tests for OCB protocol verification have become an essential part of the SOC design flow.

Up to now, several compliance test schemes have been proposed for verifying the OCB protocol. Typically, they can be classified into two types. The first one is based on the formal verification methodology. In this approach, the Design-Under-Verification (DUV) is modeled as a finite state transition system and the properties of specification are specified in terms of a temporal logic formula such as Computation Tree Logic (CTL) [1]. Then, the DUV can be verified by using some existing verification tools such as SMV [2], VIS [3] and FormalCheck [4]. The other type of OCB compliance test is performed using a simulation environment in which the DUV is exercised. The properties of OCB specification are represented as a fixed set of rules. Then, the cycle by cycle checks are used to ensure that the DUV obeys all rules. If the DUV does not violate any rule in the simulation, it can be claimed compliance to the OCB specification.

Both approaches discussed above are commonly used in industry and can achieve a pre-defined quality. However, there are some inconveniences in these two methods. The obvious one is about the property extraction. There is still no systematic method for extracting the required properties from an informal OCB specification. Frequently, we may obtain some redundant properties or miss some necessary properties. It will make the compliance test based on these extracted properties ineffective. Furthermore, it is difficult to generate the rules which consider the data part of a transfer because most of the rule-based verification is only suitable for the control behavior of OCB protocols.

In this paper, we propose a monitor-based approach for the OCB compliance test. This approach uses a bus monitor to observe the interface behavior and then check if it conforms to the OCB specification in a simulation environment. To propose a more systematic approach of modeling an OCB protocol, we use a FSM model. Based on this model, we can extract the necessary properties from specification by following a pre-defined architecture. Furthermore, this FSM model can also indicate exactly when a data move occurs in the transfer. Therefore, we can also check the data part of a transfer during the simulation.

The rest of this paper is organized as follows. In the next section, a FSM model which represents the properties of OCB specification is proposed. In section 3, a monitor-based methodology and the detail of our approach are presented. In section 4, we demonstrate our methodology with two OCB protocols, WISHBONE and AMBA AHB, and show the experimental results. Finally, the conclusions are given in section 5.

II. A Finite State Machine (FSM) Model

A. Definition of FSM

A Finite State Machine (FSM) M is defined as a 5-tuple, $\{Q, I, S, q0, \delta\}$, where:

- Q is the finite set of states,
- I is a set of bus signals that need to be considered,
- S is a finite set of statuses describing the values of bus signals for each state,
- $q0 \in Q$ is the initial state,
- $\delta : Q \times 2^I \rightarrow Q$ is the set of state transition function.

The proposed FSM model consists of three major elements, state, status and event which is depicted by the

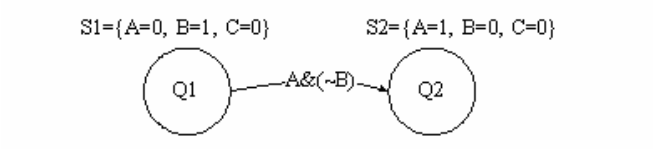


Fig. 1. The basic elements of FSM

combination of bus signals causing the state transition. Fig.1 shows a simple FSM module with Q = {Q1, Q2}, I = {A, B}, and S = {S1, S2}. We use a node and an edge to indicate a state and a transition respectively. The status which describes the correct values of bus signals is shown beside its corresponding state. For example, the status S1 consists of "A=0","B=1", and "C=0". The event enabling a given transition is shown as a Boolean function beside the transition. For example, the event "A&(~B)" in Fig.1 indicates that a state transition from Q1 to Q2 is enabled when the signal A changes to 1 and the signal B changes to 0.

There are two notations in this FSM model. First, the state transition is enabled only when its own event is true; that is, if no event is true, the FSM will still stay in the original state. Second, any two events in each state must not take place simultaneously. There is at most one event that may occur in a state for each cycle.

B. FSM Model Creation for an OCB Specification

Because of the informal specifications of OCB protocols, it is hard to analyze them automatically. To solve this problem, some research recommends re-writing the specifications in a formal form. They extract the properties from a specification and describe them as a set of rules [5]. However, there is not a clear guideline on how many properties being sufficient for describing a specification completely. As a result, we propose a more systematic method on recording the specification by using a FSM model.

The flow in Fig.2 shows the detail about translating a specification into a FSM. The first step is to determine the required bus signals. Since our verification focuses on the

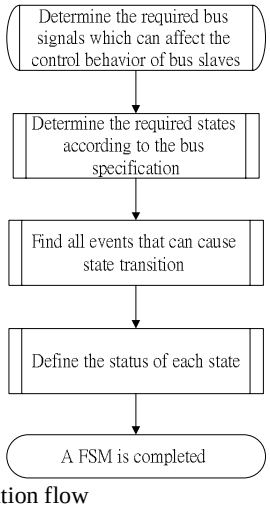


Fig. 2. The FSM creation flow

Fig. 3. A routine of bus slave

interaction between a bus master and a bus slave, the required bus signals must contain two parts: (Assume the DUV is a bus slave.)

Input signals: the signals from the bus master and the system into the bus slave,

Output signals: the signals from the bus slave to the bus master.

The second step is to determine the required states. In order to avoid creating redundant states, we may define a reasonable set of states to achieve the requirements of OCB specification. By observing the control behavior of bus slaves, we can find that there is a routine for all bus slaves. Fig.3 shows the routine of bus slaves. First, when the bus master does not assert any transfer, the bus slave stays in the origin state. It means that the bus slave is idle and is waiting for the bus master to assert a transfer. Next, the bus slave is selected and asserted a transfer. The bus slave enters the start state and a set of signals including the address, the transfer type, and some control signals are received. Finally, the bus slave enters the response state to return the corresponding response.

Besides, two extra situations, reset and wait, may take place. As a result, we can classify the states of a FSM into five types, which are defined as follows:

Origin

The Origin state is the initial state, in which the bus slave is idle.

Reset

The Reset state is the state in which the signals of bus slave are reset.

Start

The Start state is the state in which a bus master begin to assert a transfer

Response

There are many kinds of Response states depending on the responses that the bus slave supports. The common responses are Normal, Error, and Retry. Normal means that the transfer has completed successfully. Error means that an error has occurred in the transfer. Retry means that the transfer has not yet completed, so the master should retry the transfer until it completes.

Wait

The Wait state is the state that the bus master and the bus slave can use it to insert additional cycles. So, there are two kinds of Wait states, which are "Wait for master" and "Wait for slave".

We need to extract the properties from an OCB protocol specification and build them into the FSM. In this FSM model, the combination of a state and a transition can indicate one or more properties. Besides, the transition between two states can also indicate the relationship of these two states. The status of each state describes the correct values of bus signals, which can also be used to represent the properties. An example of FSM model is shown in Fig.4, where Origin, Reset, Start, Normal, Error and Wait are states, and S1~S6 are statuses.

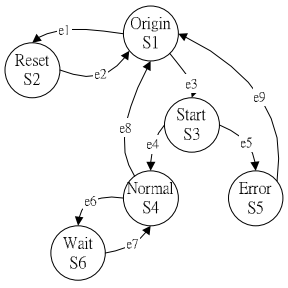


Fig. 4. An example of FSM model

C. *Verifying the correctness and Completeness of the FSM*

Since the FSM model plays a major role in our approach, it is very important to check the correctness and the completeness of the FSM. To verify these characteristics, we propose two approaches：

- **Using the existing properties of specification**
  We can use the existing properties of specification to verify our FSM. These properties can come from other compliance tests or the protocol specification. If the FSM conforms to all the properties, then we will have more confidence that the FSM is correct and complete.
- **Using the IP that has passed other compliant tests**
  We take the compliant IP as a DUV in our test bench to check if it can pass our verification. It may help us to check the correctness and completeness of the FSM in an experimental method.

III. Proposed Approach

A. *A Monitor-based approach*

For dealing with the interactive behavior of OCB protocols, we introduce a monitor-based approach for the bus protocol verification [6]. Fig.5 shows the block diagram of the monitor-based approach. The bus system in this diagram has four agents and the inputs of each agent are the outputs of other agents. The interactive behavior among these agents has to obey the bus protocol specification. A bus monitor which checks the agents' compliance to the protocol is added to the bus system. It is a machine with the bus signals as its inputs, and the $correct_i$ signals as the outputs. Besides, it also involves the properties of the bus specification. At each execution step, the bus monitor observes the bus signals and immediately checks if they conform to the bus protocol specification. Once an agent breaks the bus protocol, the monitor will signal out the error agent by making the corresponding $correct_i$ signal false.

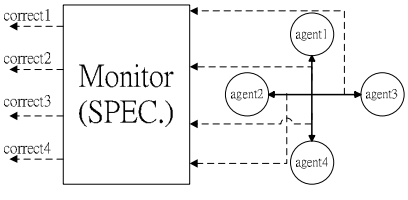


Fig. 5. The block diagram of monitor-based approach

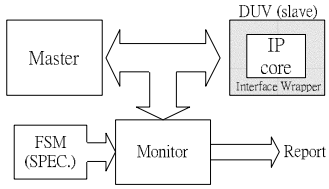


Fig. 6. The block diagram of our approach

B. *Our Approach*

In this section, we propose a practical approach for OCB compliance test, which is developed by modifying the monitor-based methodology. Since most IP components are designed as a "slave" style, we will focus on the verification of bus slave interface. Fig.6 shows the block diagram of our approach. It consists of four parts: a DUV, a bus master, a FSM model, and a bus monitor. The DUV is the bus slave which we want to verify. The bus master is used to initiate the transfers to the DUV. The FSM model is used to describe the specification of an OCB protocol, which has been discussed in section 2. The bus monitor is a checker that catches the values of the bus signals between the bus master and the DUV, and then compares them with the FSM model to determine whether the DUV passes this test or not.

Our approach is performed in a simulation environment in which the IP component is exercised. By controlling the bus master, we can use it to deliver the appropriate bus transfers to the DUV. As a result, we can conveniently verify the DUV's compliance to the specification of the OCB protocol. Since the aim of our approach is to verify the bus interface, we assume that the IP core in the DUV operates correctly during the simulation; that is, the original IP without connecting the bus interface must be error-free.

Fig.7 shows the flow of our approach. First, we model the specification of OCB protocol as a FSM, which specifies the interactive behavior among the bus system, especially for the bus master and the bus slave. Then, we simulate the whole system, including the bus master, the bus monitor, and the DUV. According to the observation of the monitor, we obtain the actual interactive behavior. Next, we compare the interaction with the FSM. If any conflict occurs, the

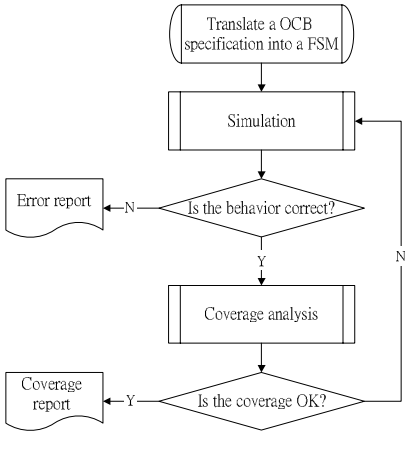
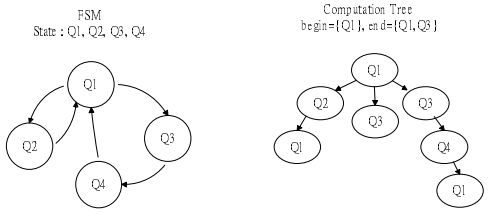


Fig. 7. The flow of our approach



(a) (b)

Fig. 8. An example for finding all possible M-paths

violations will be recorded in the error report. Otherwise, a coverage analysis of this simulation will be executed and then the coverage percentage will be shown in the coverage report.

In our approach, only the information about Primary Inputs and Outputs (PIOs) and supporting responses of DUV are required for the simulation environment. Therefore, we can apply it to any slave-typed IP component without knowing the detail of it.

C. *Coverage Analysis*

In order to perform the quantitative analysis of simulation completeness, some coverage metrics are needed. In our approach, since we use the FSM model to specify the OCB protocol, we can employ the FSM coverage metric with respect to some meaningful paths (M-paths) for coverage analysis.

According to the observation of the OCB protocols, we find a scenario: any two neighbor transfers would not affect each other. Therefore, we can use a complete bus transfer in the FSM as the basic element for coverage analysis. The definition of M-path is shown as follows.

The M-path is a path which can form a complete bus transfer in the FSM model. It is composed of some states and some edges.

In order to find all possible M-paths from a FSM, we can unfold the FSM in the form of "computation tree". The computation tree has the capability of showing all of the possible executions starting from the initial state. However, this tree would stretch infinitely and furthermore the bus transfer does not always start with the initial state. Therefore, we have to modify it by adding a constraint that indicates the beginning state and the ending state. Figure 3-4 shows an example of translating the FSM into the form of computation tree. The FSM which consists of four states, Q1, Q2, Q3, and Q4, is shown in Fig.8(a). Each transfer in this FSM must begin with the Q1 state, and end with the Q1 or Q3 state. Fig.8(b) shows the corresponding computation tree, in which there are three branches, "Q1→Q2→Q1", "Q1→Q3", and "Q1→Q3→Q4→Q1". As a result, we can find three M-paths in this FSM.

Finally, for calculating the percentage of FSM coverage, we define a coverage function, which is shown as follows.

$$\text{Coverage} = \frac{\text{P\_verified}}{\text{P\_total}} \times 100\%$$

where P_total is # of total M-paths in the FSM;
P_verified is # of the verified M-paths in the FSM.

PI_1	PI_2	PI_3	…	PI_n	PO
1	0	11	…	32	-
2	1	0	…	2	50
…					

Fig. 9. The data information file format

D. *Data Checking*

The data part of a bus transfer is functionally irrelevant to the bus protocol specification. It is very difficult to specify a rule for verifying the correctness of the data transfer. As a result, most of the rule-based approaches for OCB compliance test usually neglect to verify it and assume it is correct during a "compliance" run.

In our approach, since the proposed FSM model can exactly indicate when a data move happens, we can verify the correctness of the data by comparing with a Data Information File (DIF). The DIF is used to describe the information about the PIOs of the DUV. Only the bus signals relative to the data transfer are needed. Therefore, the required PIs include the address signal, the control signals, and the data input signals; the required POs include the data output signals. Fig.9 shows the format of the DIF. The first line of it indicates the signal names of the required PIOs. In the following, the values of the PIOs for a bus transfer are presented in each line.

The DIF supports sufficient information to the bus monitor and the bus master. According to the description in the DIF, the bus master can assert the corresponding bus transfers to the DUV. Then, the bus monitor catches the value of the data signal when the FSM enters the state that the data move occurs. By comparing with the expected value in the DIF, the monitor can determine if the data is correctly transmitted. The block diagram for verifying the correctness of the data is shown in Fig.10.

IV. Experiments

In order to demonstrate our approach, we choose two kinds of OCB protocols, WISHBONE [7] and AMBA AHB [8], as examples.

A. *WISHBONE*

The WISHBONE specification uses a handshaking protocol between the MASTER and SLAVE interface. When the

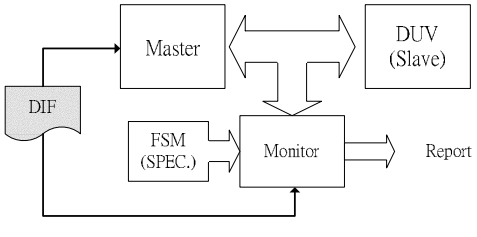


Fig. 10. The block diagram for verifying the data

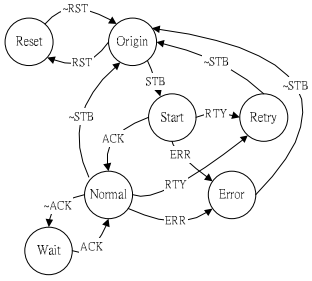


Fig. 11. The FSM for WISHBONE slave specification

MASTER is ready to transfer data, it asserts a strobe signal. Then, the SLAVE asserts one of terminating signals, ACK, ERR, and RTY, to indicate the transfer response. For creating a FSM model, first we need to choose the required bus signals that affect the slave behavior. In the WISHBONNE specification, only the signals, RST, STB, CYC, ACK, ERR and RTY, are required. By following the FSM creation flow, we can create a FSM that conforms to WIDHBONE specification. This FSM is shown in Fig.11.

B. AMBA AHB

The AMBA AHB specification is a high-performance system backbone bus. For achieving high clock frequency, the AMBA AHB adopts pipeline architecture to implement the bus system. Every transfer of AMBA AHB can be divided into two phases, address phase and data phase.

The key control signals for creating the FSM model are HSEL**,** HRESETn, HTRANS [1:0], HREADY and HRESP [1:0]. According to the FSM creation flow, we can build the FSM, which is shown in Fig.12. Due to the pipeline architecture of AMBA AHB, we can find that there are two normal states in the FSM model. The normal state with BUSY/IDLE indicates no data move in the following transfer, and the normal state with NONSEQ/SEQ indicates that there is a data move in the following transfer.

C. Experimental Results

The experiments are conducted over six real designs written in Verilog HDL. The information about these designs is shown in Table I. The design AC97 controller is a simple

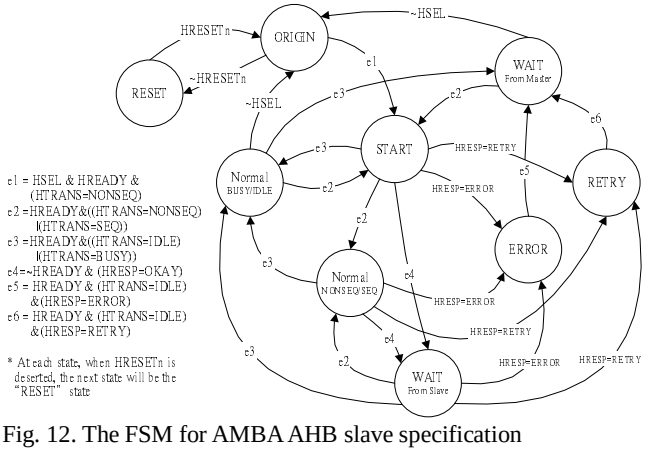


Fig. 12. The FSM for AMBA AHB slave specification

TABLE I
Design information

Design	Protocol	# PI	# PO	Supporting responses
AC97 controller	WISHBONE	11	9	Normal (wait)
SPI	WISHBONE	9	7	Normal, Error
PWM/Timer/Counter	WISHBONE	10	6	Normal, Error
RGB2YCrCb	AMBA AHB	8	3	Normal
Convolution	AMBA AHB	8	3	Normal (wait)
MAC	AMBA AHB	8	3	Normal, Error

AC97 controller IP core that supports one AC97 codec with 6 output and 3 input channels. The design SPI is the serial peripheral interface IP core. The design PWM/Timer/ Counter is a multi-function IP core, which provides the user-programmable PWM, timer, and counter controller. The design RGB2YCrCb is a RGB-to-YCrCb translator. The design Convolution is a convolution calculator for discrete wavelet transfer. The design MAC is a multiplier accumulator.

The proposed OCB compliance test is implemented in Veiling HDL on an Ultra Sparc II workstation. The verification is run in a simulation environment which consists of a bus master and a bus monitor. We verify the designs by giving the stimulus composed of all types of transfers, such as "single read/write" and "block read/write". Table II shows the experimental results.

The experimental results demonstrate that all DUVs can be verified effectively and achieve 100% in a reasonable execution time. Besides, we also detect an error of the design SPI, which is claimed compliance to WISHBONE specification by other compliance test. After being modified, this design is correct and can achieve 100% coverage in 19 clock cycles.

Furthermore, we also try to inject some errors into the correct designs and then conduct some experiments for these erroneous designs. The results are shown in Table III. We inject some errors into three correct designs, PWM/Timer /Counter, MAC, and Convolution. The experimental results for the erroneous designs demonstrate that our approach can indeed detect the errors and indicate the states in which the errors occur. This can help designers to shorten the debugging process.

V. Conclusions

In this paper, we propose a monitor-based approach for OCB compliant test, in which a bus monitor is used to observe the bus signals among the bus system and to check if they conform to the OCB protocol specification. In order to systematically extract the properties from an informal OCB specification, we also propose a FSM model. Based on this model, we can extract the necessary properties and avoid obtaining the redundant properties. Besides, we can also check the data part of a transfer during the simulation. Finally, we propose an appropriate coverage metric to perform a quantitative analysis of simulation completeness on our OCB compliance test. The experimental results show that we can indeed verify the OCB protocols and detect the design errors when tests fail.

TABLE II
Experimental results for compliance test

Design		# states	# total M_paths	Time (cycle counts)	# violations	Coverage (%)
AC97 controller		5	3	36	0	100
SPI	original	5	4	-	1	-
	modified			19	0	100
PWM/Timer/Counter		5	4	14	0	100
RGB2YCrCb		6	16	31	0	100
Convolution		7	24	60	0	100
MAC		7	20	42	0	100

TABLE III
Experimental results for the erroneous designs

Design	Inserted error	Results
PWM/Timer/Counter (WISHBONE)	ACK and ERR are asserted simultaneously	Test fails in the Normal state: ERR!=0
MAC (AMBA AHB)	The ERROR response is given with a single cycle	Test fails in the ERROR state: ERROR response needs two cycles
Convolution (AMBA AHB)	Provide a wait state to IDLE transfer	Test fails in the "Normal with IDLE/BUSY" state: HREADY!=1 Test fails in the "WAIT from Master" state: READY!=1

Acknowledgements

This work was supported in part by ROC National Science Council under Grant NSC91-2215-E-009-074

References

[1] K. L. McMillan, "Symbolic Model Checking", Kluwer Academic Publishers, 1993.
[2] The SMV Model Checker, http://www-cad.eecs.berkeley.edu/~kenmcmil/smv/
[3] The VIS Group, "VIS: A system for Verification and Synthesis ", Proceedings of the 8th International Conference on Computer Aided Verification, LNCS 1102, pp. 428-432, 1996.
[4] Cadence Formalcheck Tool, http://www.cadence.com/datasheets/formalcheck.html
[5] Andy Nightingale and John Goodenough, "Testing for AMBA™ Compliance", ASIC/SOC Conference, 2001. Proceedings.14th Annual IEEE International, pp. 301-305, 2001.
[6] Kanna Shimizu, David L. Dill and Alan J. Hu, "Monitor-Based Formal Specification of PCI", Proceedings of the Third International Conference of Formal Methods in Computer-Aided Design, Nov. 2000.
[7] WISHBONE, Revision B.3 Specification, http://www.opencores.org/wishbone/
[8] ARM, "AMBA Specification (Rev 2.0)" ARM Document No. ARM IHI0011A, http://www.arm.com
[9] Prakash Rashinker, Peter Paterson, and Leena Singh, "System-on-a-chip Verification Methodology and Technique", Kluwer Academic Publishers, 2001.