A SPI Interface Module Verification Method Based on UVM

Yong Guo, Yubo Wang, Xiaoke Tang, Yi Hu, Jie Gan, Wennan Feng, Yi Hao
Beijing Smart-chip Microelectronics Technology Co., Ltd., Beijing, China
823408232@qq.com

***Abstract*—The serial peripheral interface (SPI) is an important module for realizing communication between the APB bus in the SOC chip and peripheral SPI devices. Therefore, efficient and sufficient verification of the function of the SPI module is very important for the design and manufacture of the SOC chip. In this article, a verification environment for SPI module is built based on universal verification methodology (UVM). By introducing the register abstraction layer (RAL), the efficiency of register attribute checking and configuration is greatly improved. In addition, the use of constrained random excitation and automatic result comparison function has realized the full verification of the SPI function, with a coverage rate of 100%. Finally, the verification environment of the SPI module has been successfully migrated to the SOC verification environment.**

Keywords—SPI; UVM; RAL; module verification

I. INTRODUCTION

With the development of the integrated circuit industry, global companies have launched fierce competition in the system on chip (SOC) field. SOC usually integrates microprocessor, IP core, memory, and other components on a single chip to work together [1]. The interfaces provide paths for data transmission between components. According to the data transmission mode between integrated circuits and peripheral devices, the interface is divided into serial interface and parallel interface [2]. Serial peripheral interface (SPI) is a synchronous serial interface bus, which can transmit data without interruption. SPI has a higher transmission rate and the working characteristics of synchronous full-duplex. SPI is easy to use and it can reduce component space, power dissipation and manufacturing cost [3]. SPI is usually used for data transmission between microcontrollers and small peripherals (such as shift registers, sensors, and SD cards), and is widely used in industrial SOC chips. Compared with ordinary consumer electronic chips, the working environment of industrial SOC chips is more stringent and requires higher communication reliability. Therefore, in order to ensure the functional correctness of the SPI module, accurate and efficient verification is essential.

As the SOC structure becomes more and more complex, new verification methodologies are constantly being proposed to ensure the success rate of chip tape-out. Universal verification methodology (UVM) is a set of efficient and open source verification environment development library based on SystemVerilog language. It has the characteristics of object-oriented programming language. It can encapsulate methods and functions into different types of libraries. UVM can easily create a testbench with a standardized hierarchical structure, as shown in Fig.1, it is a typical tree structure of UVM [4,5].

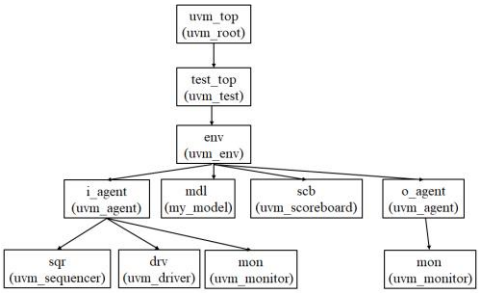

Fig. 1. UVM tree

In this paper, a reusable SPI interface testbench based on UVM-based verification methodology is built for the verification of a self-developed SPI interface module. The verification efficiency is improved by introducing the register abstraction layer RAL. The restricted random excitation and automated result comparison functions is introduced to achieve the full verification to the function of SPI interface, and 100% coverage has been achieved.

II. FEATURES OF SPI PROTOCOL

A. The SPI Working Mode

The SPI bus consists of four signals: serial clock (SCK), master output and slave input (MOSI), master input and slave output (MISO), and slave selection (SS) [6-8]. Based on the data transmission and reception mechanism of the rising or falling edge of the serial clock, SPI is divided into four working modes: 0/1/2/3. The four working modes are respectively controlled by the clock polarity (CPOL) register and clock phase (CPHA) register. The CPOL register determines the potential level of the synchronous clock signal when the bus is at the idle time. When CPOL is 0, the synchronous clock signal is at low potential. When CPOL is 1, the synchronous clock signal is at high potential. The CPHA register determines the time of data sampling. When CPHA is 0, sampling is performed on the first edge of the clock signal. When CPHA is 1, sampling is performed on the second edge of the clock signal. The four modes are shown in Fig.2.

978-1-7281-5224-0/20/$31.00 ©2020 IEEE

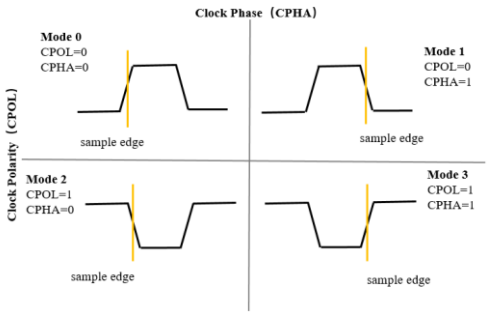

Fig. 2. SPI working model

Mode0: CPOL and CPHA are both 0, then the clock signal is at low potential when the bus is idle, and the data is sampled on the first edge (the rising edge) of the clock signal after the transmission starts.

Mode1: CPOL is 0 and CPHA is 1, then the clock signal is at low potential when the bus is idle, and the data is sampled on the second edge of the clock signal after transmission starts, and data is sent on the first edge.

Mode2: CPOL is 1 and CPHA is 0, then the clock signal is at high potential when the bus is idle, and the data is sampled at the first edge of the clock signal after the transmission starts.

Mode3: CPOL and CPHA are both 1, then the clock signal is at high potential when the bus is idle, and the data is sampled on the second edge of the clock signal after the transmission starts [8,9].

B. The SPI Communication Method

As shown in Fig.3, SPI is a single-master communication protocol, that is a central master initiates all communication with slaves, which can ensure that the master controls the entire communication process. Therefore, this requires that the master and slave must use the same group parameter (SCK/CPOL/CPHA) when SPI is communicating. If there are multiple slave devices with different configurations in the bus, the master device must reconfigure its own parameters when it communicates with the slave device every time.

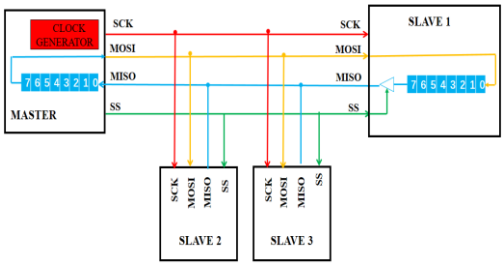

Fig. 3. The forms of large-scale network survivability associations.

In addition, data output and data input are independent of each other during SPI communication, but data output and data input can be completed at the same time, to achieve full-duplex communication. For example, when the SPI master wants to send data or request information to the slave, it selects a slave by pulling down the corresponding SS line and activates the clock signal at the clock frequency which is used by the master and the slave. The master generates information on the MOSI line while it samples the MISO line [10].

C. Introduction to Self-Developed SPI Architecture

The design under test (DUT) is a self-developed SPI module that complies with the SPI interface protocol standard, which allows full-duplex synchronous serial communication between the mcu and peripheral devices. The internal structure of the spi module is shown in Fig.4.

Spi_reg provides abp conversion interface, txfifo is 8×8bit sending fifo, rxfifo is 8×8bit receiving fifo, mst_ctrl is the unit that controls the work in the master mode, which controls the master transmission and reception in the four modes of simplex and duplex; slv_ctrl is the unit that controls the work in the slave mode, which controls the simplex, duplex, and functions in the four modes in the slave mode.

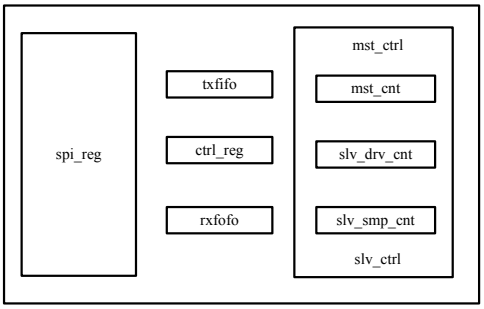

Fig. 4. Bilateral survivability chain decomposed into unilateral survivability associations.

Spi_reg as the abp bus interface conversion module, is used to configure registers/report interrupts to exchange data with the CPU.

Mst_ctrl is the unit that controls the work in the master mode. It controls the sending and receiving functions in the four modes of simplex and duplex in the master mode. The data sent is subject to the counter master_cnt. The master_cnt is the counter in the master mode. The master mode is a byte The sending of data requires at least 10 master_cnt cycles to complete. The sending and receiving units of master mode will read txfifo and rxfifo respectively in the last cycle of master_cnt, and give the data in the next cycle and indicate the status.

Slv_ctrl is a unit that controls the work in the slave mode. It is used to receive and send and receive data in the slave mode. It has two counters slv_drv_cnt and slv_smp_cnt. The former is used to control the data transmission in the slave mode, and the latter controls the data reception in the slave mode. When slv_drv_cnt is 7, slv_ctrl will read txfifo and prepare to send next time. Similarly, when slv_smp_cnt is 7, slv_ctrl will write rxfifo. It controls simplex, duplex and functions in the four modes in the slave mode. In addition, working in

slave mode does not consider the full or empty state of fifo. Therefore, when the SPI speed is very high, the speed of reading rxfifo and writing txfifo needs to be considered from the system perspective. It is recommended to set the water level buffer, the water level setting depends on the system.

Rxfifo is an 8×8bit asynchronous receiving FIFO, used for data interaction between the system clock and the SPI working clock. The cpu can be read and cleared, and the hardware can be written. The rxfifo clock is mst_clk in the master mode and always exists. In the slave mode, only During the transmission, if the rxfifo is full, the transmission will be stopped in the master mode, and the SPI will still accept it in the slave mode and will still send out the write fifo action, but the data cannot be written. In the same way, if the txfifo is empty, the transmission will be stopped in the master mode, and the SPI will still try to read the txfifo in the slave mode, but the read data will not change. The next transmission data will still be sent, but the data content cannot Guarantee. The use of FIFO can realize data buffering and reduce the communication pressure of SPI.

III. SPI INTERFACE VERIFICATION

For the characteristics of the SPI interface, a verification environment for the SPI interface module is built, which is based on the verification methodology of UVM. Each verification component inherits the base class of UVM, so they have a clear hierarchical structure, which is convenient for inheritance and expansion.

A. Verification Idea

The simulation first uses VIP module-level verification, and then transplants the module-level verification environment to the SOC verification environment for system-level verification. The SPI interface module is connected to the APB bus, and the VIP is generated by DesignWare. The VIP verification method can reduce the time to build the verification platform, improve the verification efficiency, and has high reliability. At the same time, it can construct rich bus operations and improve the reusability of the verification environment. The SOC verification can check the communication and data transmission capabilities between the SPI interface module and the DMA module in a more macroscopic way, and judge whether the integrated functions of the system are normal, so as to realize the comprehensive verification of the SPI module function.

B. Verification Environment Construction

The design under test (DUT) in this work is the SPI interface module to be verified. The APB bus end uses VIP to perform register access and data read/write operations on the DUT. The SPI interface communication model is used as a DUT slave to give the DUT interface master-slave communication operations. By constructing various timings, traversing various communication rates and data, the DUT is fully verified. Since the SPI interface supports full-duplex communication, a virtual sequencer is used in the verification environment to schedule and control the sequencer of the APB end and the communication end. At the same time, a scoreboard is constructed to determine whether the data transmission between the two data terminals is correct.

The structure of the verification platform based on UVM is shown in Fig. 5. DUT is the SPI interface module to be verified, including two sets of communication interfaces, APB and SPI. Top is the top level of the verification environment, which implements APB VIP, SPI and its interface instantiation, and constructs the clock. The component responsible for the APB side communication is the APB Master Agent built using APB VIP, and the component responsible for the SPI side communication is a self-developed SPI Agent. The ENV layer is responsible for encapsulating and instantiating environmental components.

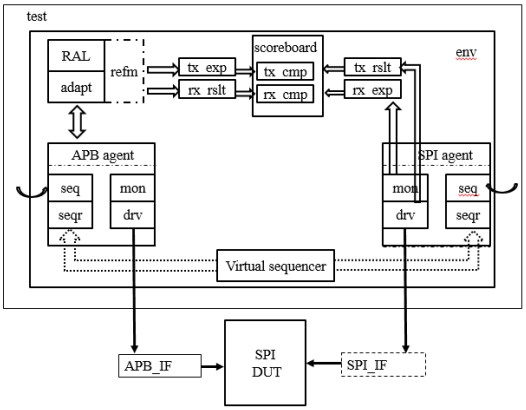

Fig. 5. Bilateral survivability chain decomposed into unilateral survivability associations.

In the build phase, ENV instantiates APB/SPI agent, scoreboard, virtual sequencer and other components, and configures the number of masters and slaves in the environment, data address width, bus transmission control signal constraints, etc.; In the connect phase, ENV completes the connection of the TLM communication pipeline and the connection of the virtual sequencer. According to the UVM structure, the two communication terminals are composed of three sub-components: sequencer, driver, and monitor. When realizing the generation and sending of incentives, the sequence uses uvm_do_with to generate the constrained transaction and send it to the sequencer. The sequencer and the driver communicate through TLM to pass the transaction to the driver. Finally, the driver converts the transaction into a signal that can be recognized by the DUT according to the SPI interface protocol and related timing, and drives it to the DUT. Since the SPI interface supports full-duplex communication, the virtual sequencer is used in the component to schedule and coordinate the data packet transmission sequence of the APB and SPI interface communication, as shown in

Fig.5. In addition, the scoreboard judges the correctness of the results by comparing the data collected from the monitor of the APB agent and the SPI agent.

C. Register Abstract Layer

In the field of engineering applications, convenient and efficient verification methods are essential. The UVM register model enables verifiers to better organize and configure registers, which can simplify the verification process and improve work efficiency. Therefore, a set of register model (register abstract layer, RAL) is built in the verification environment of this design. RAL encapsulates the verification operation of the register in a model, and completes the verification of the register through the mapping. In this way, all maintenance and modification of the registers can be completed only by modifying the model.

In order to further improve work efficiency and facilitate subsequent maintenance work, the RAL register model is generated using the Ralgen tool that comes with VCS. The generated register model is mainly divided into three structural levels: the topmost uvm_reg_block is an instance containing all registers (uvm_reg); the uvm_reg is a virtual class; the domain uvm_reg_filed of the register derived from uvm_reg is a variable with stored value and can be used directly.

In addition, the implementation of RAL model integration requires uvm_reg_map and adapter classes. uvm_reg_map is the address corresponding to the storage register after being added to the RAL model. The adapter is used to implement the type conversion of the transaction between the bus and the RAL model.

IV. Simulation Results

The verification platform uses constrained random technology to configure the verification environment and DUT, and uses a combination of direct test cases and random test cases to ensure that the DUT achieves all expected functional requirements.

There are a total of 25 verification cases for this SPI module, including 18 direct use cases and 7 random use cases. The final coverage result of this module verification is shown in Fig.6.

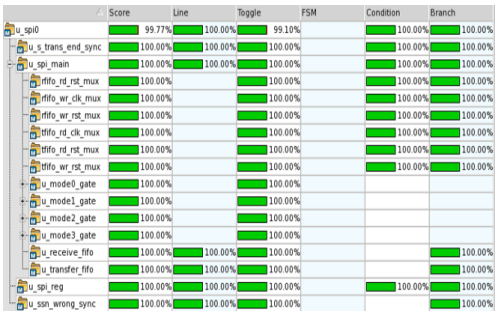

Fig. 6. SPI interface module coverage statistics

By using the verification method described in this article, the verification coverage of the line, condition, toggle, and branch of the target u_spi0 and its submodules are all close to 100%. Therefore, the full verification of the SWP interface module is completed, and the verification requirements of the interface module are achieved.

In addition, during the construction of the SOC verification environment, the APB IF in the UVM verification environment designed in this paper was mounted to the APB bus of the SOC, and the SPI AGENT was encapsulated as an SPI model for data comparison. The register model RAL is extended to verify the register value of each module. After testing, the design environment was successfully integrated into the SOC verification environment, which not only verified the data connectivity between the SPI module and the CPU, DMA and other modules, but also greatly improved the construction efficiency of the system-level verification environment. Therefore, it is fully proved that the verification environment has excellent reusability.

V. Conclusions

In this work, a UVM-based reusable verification platform is developed for the SPI interface module in the SOC chip. The verification result shows that the RAL model generated using DesignWare's AMBA VIP and VCS ralgen tools enables verifiers to better organize and configure registers, to simplify the verification process, and to improve work efficiency. Finally, the full verification of the SPI interface is implemented, and the coverage rate reaches 100%, which meets the design requirements. In addition, the verification environment is smoothly transplanted to the SOC verification environment, which greatly improves the verification efficiency of the SOC chip.

References

[1] A.K. Swain, and K. K. Mahapatra, “Design and verification of WISHBONE bus interface for System-on-Chip integration.” ieee india conference, 2010, pp. 1-4.

[2] H. Geng, Microcomputer principle and interface, China Water & Power Press, 2005.

[3] A.K. Oudjida, M.L. Berrandjia, et al. “Design and test of general purpose SPI master/slave IPs on OPB bus”. 7th International MultiConference on Systems, Signals and Devices, 2010, pp. 1-6.

[4] K. Salah, “A uvm-based smart functional verification platform: Concepts, pros, cons, and opportunities,” in 2014 9th International Design and Test Symposium (IDT), 2014, pp. 94-99.

[5] J. Bergeron, Writing testbenches using systemVerilog, New York, Springer-Verlag, 2006

[6] Z. Zhou, Z. Xie, X. Wang, et al., “Development of verification envioronment for SPI master interface using SystemVerilog”, International Conference on Signal Processing, 2012, pp. 2188-2192.

[7] D. Roopesh, K. Siddesha, “RTL design and verification of SPI master-slave using UVM”, International Journal of Advanced Research in Electronics and Communication Engineering, vol. 4, pp. 2159-2162, 2015.

[8] P.R. Reddy, P. Sreekanth, K.A. Kumar, “Serial peripheral interface-master universal verification component using UVM”,

International Journal of Advanced Technologies in Engineering and Management Sciences, vol. 3, pp. 27-30, 2017.

[9] C. Zhang, B. Li, X. Zeng, et al., "A high-speed, high-reliability SPI interface design", Information & Communication, vol. 5, pp. 61-63, 2019.

[10] S. Moon, "Design of an SPI interface for multimedia cards in ARM embedded systems", The Journal of the Korean Institute of Information and Communication Engineering, vol. 16, pp. 273-278, 2012.