The Design Of UVM Verification Platform Based On Data Comparison

ShengYuan Zhou †
VLSI & System Lab
Beijing University of Technology
Beijing China
1216129148@qq.com

ShuQin Geng
VLSI & System Lab
Beijing University of Technology
Beijing China
gengshuqin@bjut.edu.cn

XiaoHong Peng
VLSI & System Lab
Beijing University of Technology
Beijing China
pengxiaohong@bjut.edu.cn

MingMing Zhang
VLSI & System Lab
Beijing University of Technology
Beijing China
mingming.htte@qq.com

MengHao Chu
VLSI & System Lab
Beijing University of Technology
Beijing China
18810926279@163.com

PengKun Li
VLSI & System Lab
Beijing University of Technology
Beijing China
873193604@qq.com

Hang Lu
VLSI & System Lab
Beijing University of Technology
Beijing China
nirvana_201@163.com

RongHao Zhu
VLSI & System Lab
Beijing University of Technology
Beijing China
ronghaozai666@163.com

ABSTRACT

This article is based on UVM (Universal Verification Methodology) to build a verification platform that can quickly verify the DUT (Design Under Test). The test platform skillfully uses the UVM components and flexible configuration scheme, which fully shows the strong advantages of the Universal Verification Methodology. The uniqueness of this verification platform is that it does not use UVM's reference model verification component. This verification platform is suitable for modules without algorithm models, such as high-speed interface to low-speed interface, Interrupt handling module, etc. A hierarchical register model is applied in the verification platform, which can configure registers quickly and flexibly. At the end of this article, the IP-level verification work of the Interrupt handling module and the coverage collection work are realized, which proves that the verification platform built in this article has higher flexibility and reusability. The verified Interrupt handling module is based on the AXI high-speed bus transmission protocol. The result of the verification is that the Interrupt handling module realizes the full and overflow of ring FIFO, and the bus meets the protocol requirements. At the same time, the coverage rate meets the verification requirements. Chip verification occupies an increasingly important position in the process of chip research and development, which is a key part of ensuring the smooth tapeout of chips.

EITCE 2020, November 6–8, 2020, Xiamen, China
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-8781-1/20/11…$15.00
https://doi.org/10.1145/3443467.3443908

CCS CONCEPTS

• Hardware~Hardware validation~Functional verification~Transaction-level verification

KEYWORDS

UVM; Verification Platform; Register Model; AXI; Coverage;

1 Introduction

With the rapid development of the integrated circuit industry, the complexity of chips has gradually increased, and the development of miniaturization and complexity of chips has become a trend. As foundries such as TSMC have upgraded the chip process to 3 nanometers, the number of transistors per square millimeter has increased to hundreds of millions. While the performance and complexity of the chip have increased significantly, the verification of the chip has become more and more complicated. The verification of the chip is inseparable from the verification platform. The construction of the verification platform has become the first task that verification engineers must solve. Different modules in the chip require different verification platforms, and verification components can be flexibly selected to build a more efficient verification platform.

Interrupt handler determines whether a chip can run stably, which directly affects the running speed of the chip. From the whole process of chip research and development, it can improve the speed of SoC（System on Chip）to complete the research and development of interrupt handler in time. Therefore, verifying the correctness of interrupt handler function plays an important role in the whole chip development. Chip verification is divided into IP level and SoC level. IP level verifies the function of modules, while SoC level verifies the interconnection between modules[1]. A chip cannot be successfully streamed without SoC level verification, and cannot do without IP level verification. In order to improve the efficiency of IP level verification, it is more and more popular to use VIP to quickly build a special verification environment for IP[2]. The DUT (Design under test) to be verified in this paper is a message based interrupt handler, which can reduce interrupt delay and is easy to implement at the top level. The verification in this paper belongs to the functional verification of modules. Traditional verification methods require verification engineers to manually write many test sets to test the input excitation of DUT interface[3], and also write many test sets to test the registers in DUT, so as to check the accuracy of DUT function through the behavior reflected by DUT[4]. With the increasing of the chip scale, the complexity of the chip is also increasing, and the number of registers needed will also increase. At this time, if each register is tested by manually writing a test set, it will waste a lot of research and development time of the chip. Therefore, the verification platform built in this article uses a register model to configure registers flexibly.

In this paper, UVM is used to verify the function of interrupt handler based message. UVM is an effective verification framework based on the system Verilog class library. With the support of Synopsys, Mentor and Cadence, the three major EDA manufacturers in the world, many chip design and manufacturing companies have adopted UVM methodology to build their own chip verification environment[5]. Config_db mechanism in UVM can realize multiple settings across levels. Phase mechanism can control the running order of the whole environment and make the verification environment run orderly. Sequence mechanism in UVM enables the test sequence to test DUT efficiently and orderly. TLM communication can realize the communication between different components through the analysis port. The register model in UVM realizes the unified management of registers, and the built-in power on detection sequence can realize the read and write check of all registers, reducing the misoperation caused by the verification engineer manually writing the test set detection. Using these mechanisms in UVM to build the verification environment can improve the reusability of the environment and make the verification environment more standardized[6].

The UVM verification platform proposed in this article verifies the Interrupt handling module. The proposed verification platform flexibly uses universal UVM verification components and test cases to achieve more efficient verification. In 2 the verification platform is proposed, and the validity of the verification platform is verified by verifying the Interrupt handling module. The conclusion is presented in 3.

2 Construction And Verification Of UVM Verification Platform

2.1 A Typical Verification Platform For UVM With Register Model

UVM based on System Verilog language uses class uvm_object and class uvm_component to build the most basic skeleton of the verification platform, and class uvm_sequence_item is used as the blood flow in the skeleton. uvm_object, which inherits from class uvm_void, is the most basic class in UVM, and almost all classes are derived from it.

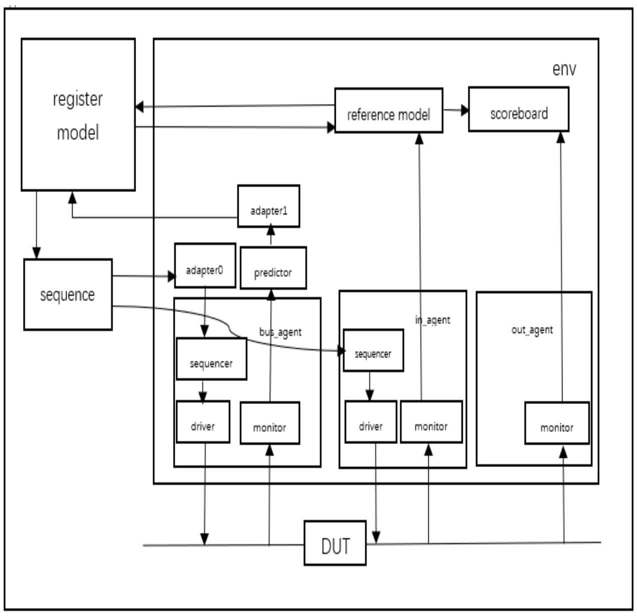


Figure 1 A Typical Verification Platform For UVM With Register Model

Env, reference model, scoreboard, agent, sequencer, driver, monitor and predictor in the verification platform are inherited from class uvm_component, and will always exist in the whole simulation process. Register model related classes (except predictor), sequence, and adapter are inherited from class uvm_object. Transaction, which has a short life cycle and is encapsulated by users, is derived from class uvm_sequence_item. The structure of a typical UVM verification platform with register model is shown in Figure 1. Reference model, scoreboard, agent, adapter and predictor are instantiated in env. Sequencer, driver and monitor are instantiated in agent. UVM provides the agent with two working modes: active mode and passive mode. When in active mode, all the above three modes will be instantiated in the agent. If it is passive mode, only monitor will be instantiated in the agent.

2.1.1 Working Process Of Typical UVM Verification Platform

The entry function run_test starts base_test or env in tb_top. Reference model simulates the behavior of DUT and configures the registers in the register model. The register model uses sequence to convert the register transaction into the transaction recognized by the bus through adapter0. adapter0 passes the transaction to the sequencer. The sequencer communicates with the driver through TLM, and the sequencer transmits the transaction to the driver after waiting for the driver's request information. After receiving a transaction, the driver will give a response to the sequence. There is a response mechanism in UVM to handle this situation. It is worth noting that this mechanism maintains a queue within the sequencer. By default, an error will be reported when the number of responses exceeds 8. Finally, the driver transmits the transaction information to DUT through the interface. When the DUT produces some behaviors, the monitor receives the data through the interface and converts it into a sequence_item in transaction level. Then the monitor transfers the converted transaction to the predictor or scoreboard through the analysis port. The predictor converts the received bus transaction into a form that can be recognized by the register model through adapter1, and the register model transmits the information to the reference model to finish the complete configuration of registers. The reference model communicates with the scoreboard through the analysis port. The scoreboard will compare the expected value collected by the reference model with the actual value received by the monitor, and judge whether the DUT works normally according to the comparison results. For other scenarios that do not need to configure registers, the test process is similar to the above. The difference is the data collected by monitor directly transferred to reference model.

2.2 Reusable Verification Platform Based On Data Comparison

2.2.1 Structure Of A Reusable Verification Platform The verification platform based on the standard Universal Verification Methodology, using a hierarchical register model and TLM communication mechanism to achieve functional verification of the Interrupt handling module. The structure diagram of the verification platform of the Interrupt handling module is shown in Figure 2. In the simulation of the platform, random tests are used to cover different parameter combinations, and directional tests are used to cover boundary conditions.

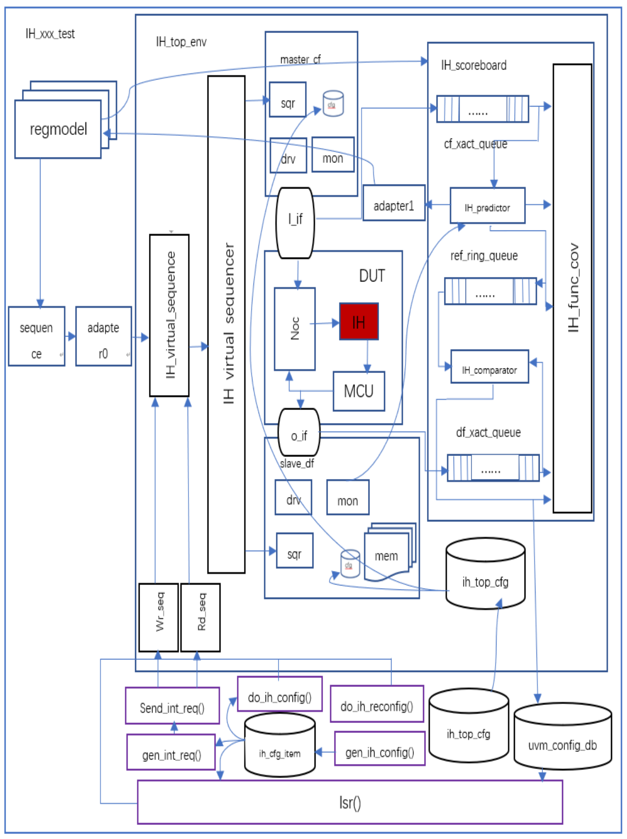


Figure 2 IH Verification Platform Structure

The name and function description of the verification component are shown in Table I:

Table I Verification Components

Category	Component	Description
IH_top_env	ih_top_cfg	IH top level environment configuration.
	IH_predictor	Generate predicted result and exception status as golden reference. Update the regmodel mirror value based on observed bus transactions .
	IH_scoreboard	For result checking.
	IH_virtual_sequencer	Coordinate multiple incentives.
	IH_virtual_sequence	When multiple components in the verification platform need to generate incentives in parallel, in order to control the synchronization of component incentives.
	IH_func_cov	Function coverage collector.
	master_cf	one AXI master agent.

	slave_df	one AXI slave agent.

2.2.2 The Work Process Of The Reusable Verification Platform On The Interrupt Handling Module The difference between the verification platform and the typical verification platform is that the data is directly transmitted to the scoreboard through the interface for comparison. The verification platform ensures the accuracy and real-time performance of the input and output data. The difference is that the reference model in typical verification platform is omitted, and ISR System is used to simulate hardware write interrupt and software read interrupt. This verification platform does not use the reference model, which saves a lot of analysis ports used in TLM communication, and simplifies the structure of the verification platform. Different from the register model in the typical verification platform, this verification platform uses the hierarchical register model with multiple maps, and instantiates the explicit predictor in IH_predictor to avoid the situation that the transaction information collected by the monitor is sent to the register model when the bus has multiple masters. In the future study, multiple maps are used to improve the access efficiency when multiple masters from multiple cores access the register model at the same time.

First of all, the top-level configuration must be performed at the beginning of the verification platform. By placing the configuration information in ih_cfg_item, and then configuring the cfg in the two agents through ih_top_cfg. ih_top_cfg can instantiate svt_axi_system_configuration to configure the number of AXI_UVC master/slave, interface type and other parameters such as axi address width, axi data width, etc. At the same time, UVM components are configured to enable or disable sub-components such as: scoreboard enable, predictor enable, etc.

Secondly, after configuring the above information, add base_test to the entry function run_test and start the entry function in tb_top. IH_virtual_sequencer will be called in base_test to coordinate multiple incentives and start different sequencers to achieve the corresponding functions. IH_virtual_sequence does not send a transaction by itself, but can controls other sequences and plays the role of unified scheduling. IH_virtual_sequencer does not specify a specific transaction type, because IH_virtual_sequencer will perform multiple types of transactions. IH_virtual_sequencer and IH_virtual_sequence are one-to-one correspondence. IH_virtual_sequence will coordinate the order of execution among the three sequences Sequence, Wr_seq and Rd_seq. IH_virtual_sequencer will send the sequence to the sequencer in master_cf and slave_df according to the sequence specified by IH_virtual_sequence. IH_scoreboard configures the register through a hierarchical register model. In the master_cf, the driver requests the transaction information from the sequencer and passes the transaction to the DUT through the I_if interface to realize the driving of the DUT. At the same time, the data of the I_if interface is transferred to the IH_scoreboard. The slave_df will collect the data output by the DUT through the o_if interface and send the output data to the IH_scoreboard for comparison. In this verification platform, TLM communication is used between sequencer and driver, between monitor and scoreboard to realize transaction transfer between verification components, which ensures sufficient information and accuracy in transaction. The sequence in this paper includes the sequence of DUT verification and the sequence of register model scanning. In this verification platform, the built-in sequence in regmodel is adapted. For register, the specified register field is scanned, and for memory, the specified memory space is scanned. This improvement accelerates the scanning of registers or memories. The isr System at the bottom of the verification platform is used to simulate the process of software and hardware dealing with interrupt. The system consists of six functions, namely gen_ih_config(), gen_int_req(), send_int_req(),do_ih_config(), do_ih_reconfig() and Isr() functions. The workflow of the system is shown in Figure2, where ih_cfg_item contains configuration information for IH.

Finally, the main functions of the verification platform are further introduced from two aspects: IH_scoreboard and IH_predictor.

IH_scoreboard defines three interfaces, uvm_analysis_imp_cf, uvm_analysis_imp_df, and uvm_analysis_imp_mon where uvm_analysis_imp_cf connects I_If is used to receive CF （Configuration Fabric） master interface transactions and uvm_analysis_imp_df connect o_If is used to receive DF slave interface transactions and uvm_analysis_imp_mon connect slave_df.mon is used to receive monitor interface transactions. In the scoreboard, uvm_analysis_imp_cf stores all the received data to cf_xact_queue, which is composed of interrupt request, blocked interrupt request and drop interrupt request due to ring FIFO overflow. uvm_analysis_imp_df connection df_xact_queue is used to store all the received data in the df_xact_queue, which is the interrupt request for normal processing of ring FIFO output. uvm_analysis_imp_mon passes the bus transaction collected by monitor to reg_predictor in IH_predictor. Then through adapter1, bus transaction is transformed into register transaction, and finally connected to register model.

The function of IH_predictor is to filter the blocked interrupt requests and drop interrupt requests due to ring FIFO overflow, and store the processed data in ref_ring_queue. Finally, compare whether the data in df_xact_queue and ref_ring_queue is consistent in IH_comparator. IH_predictor is instantiated inside IH_scoreboard. It mimics the behavior of IH design by monitoring real-time configuration from CF slave interface. Meanwhile, it snoops the interrupt request from CF slave interface and predict the outputs as a combination of descriptor in ring buffer plus corresponding INTX / error message / exception status indicator. Use monitor in slave_df to pass transaction to IH_predictor, and update regmodel mirror value according to observed bus transaction.

The main function of IH_scoreboard is to obtain the reference results from IH_predictor and compare the results from DUT output. Whenever received an DUT output on DF（Data Fabric） slave interface, it will compare it with reference result and report error if there is any mismatch or violate on any pre-defined rules. More specifically, the function of IH_scoreboard includes seven parts: check ring buffer entries send out from IH is correct in address, order, format and content, check interrupt request send out from IH is correct in address, format and content, check interrupt request send out from IH is aligned with interrupt mask settings, check if there is any un-expected interrupt request missing or repeat， check ring queue write pointer update in correct order against queue entry update, check interrupt/message is sent out to cover all entries in ring queue and update the regmodel mirror value based on observed bus transactions.

2.2.3 IH Feature List And Verification Flow In the verification plan, different function points are verified by writing testcase, and the sequencer is started in testcase to send transaction to driver. Transaction inherits from Class uvm_sequence_item and is passed through sequence. The function points verified in this paper are shown in Table II, which implements the verification of FIFO full and overflow, and verifies that the DUT can receive and send interrupts normally.

Table II Feature List

Feature	Feature point	Testcase
interface	cf master data width is 128bit, aligned 32bit	ih_intx_nomask_smallring_wrap_ringonly_test
	cf master support 16 outstanding	ih_outstanding_test
internal FIFO	Ring fifo full	ih_intx_fifo_full_test
	Ring fifo overflow with no intx report, polling mode	ih_polling_fifo_ovfl_test
RINGGEN	RINGEN ==0, interrupt request to update ring are dropped in slience	ih_intx_ring_disable_test
	RINGEN flips from high to low, uncommited int requests are all dropped in slience.	
	RINGEN flips from high to low, WPTR 、WR_WRAP and RPTR 、 RD_WRAP are reset	

2.2.4 Simulation Results The function points verified are IH (ring FIFO) full and overflow states. Because the DUT verified in this paper is a message based interrupt handler, the congestion problem caused by wiring is reduced. Meanwhile, the interrupt request information will be stored in the interrupt handling module, which is easy to converge. The commands of compilation, simulation and coverage collection are written in the configuration file. The simulation waveform and coverage are observed by Verdi software. The ring FIFO verified in this paper has a depth of 256 and a width of 64, using a 128 bit AXI interface. The simulation timing of full ring FIFO of high frequency clock is shown in Figure 3, and the simulation timing of ring FIFO overflow is shown in Figure 4.

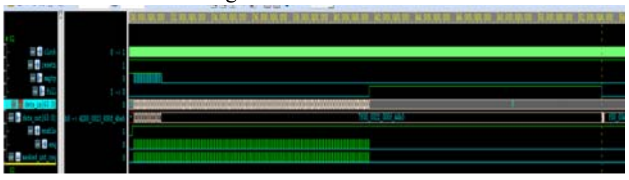


Figure 3 Simulation Timing Of Full Ring FIFO

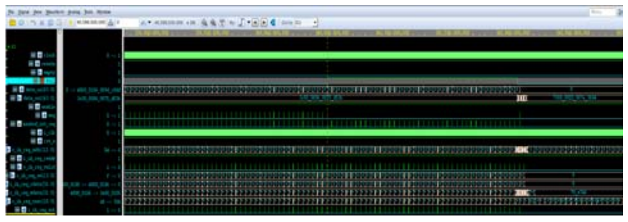
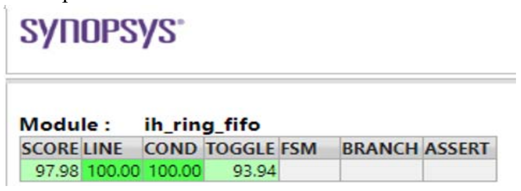


Figure 4 Simulation Timing Of Ring FIFO Overflow

Coverage is an important index to verify whether the design meets the design requirements. The function of code coverage is to check whether each line of code in the DUT has been verified, and the function coverage is to check whether the function of DUT is completed[9]. In this paper, IH configuration is generated randomly by plus arguments from test cases, and coverage is collected through regression testing. The code coverage rate of IH internal ring FIFO reaches 100%, as shown in Figure 5, so the verification platform has verified each line of code in DUT.



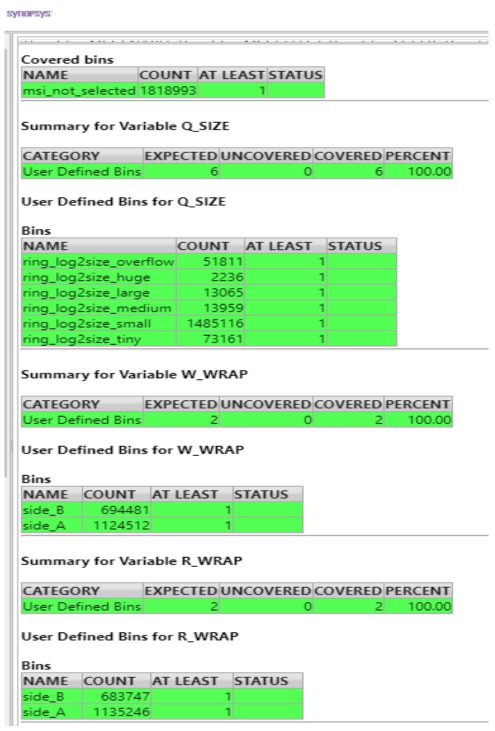
SYNOPSYS

Module : ih_ring_fifo

SCORE	LINE	COND	TOGGLE	FSM	BRANCH	ASSERT
97.98	100.00	100.00	93.94			

Figure 5 Code Coverage Of Ring FIFO

The coverage of software read pointer and hardware write pointer function reaches 100%, as shown in Figure 6, so the DUT can receive and send interrupts normally.


synopsys

Covered bins

NAME	COUNT	AT LEAST	STATUS
msi_not_selected	1818993	1	

Summary for Variable Q_SIZE

CATEGORY	EXPECTED	UNCOVERED	COVERED	PERCENT
User Defined Bins	6	0	6	100.00

User Defined Bins for Q_SIZE

Bins

NAME	COUNT	AT LEAST	STATUS
ring_log2size_overflow	51811	1	
ring_log2size_huge	2236	1	
ring_log2size_large	13065	1	
ring_log2size_medium	13959	1	
ring_log2size_small	1485116	1	
ring_log2size_tiny	73161	1	

Summary for Variable W_WRAP

CATEGORY	EXPECTED	UNCOVERED	COVERED	PERCENT
User Defined Bins	2	0	2	100.00

User Defined Bins for W_WRAP

Bins

NAME	COUNT	AT LEAST	STATUS
side_B	694481	1	
side_A	1124512	1	

Summary for Variable R_WRAP

CATEGORY	EXPECTED	UNCOVERED	COVERED	PERCENT
User Defined Bins	2	0	2	100.00

User Defined Bins for R_WRAP

Bins

NAME	COUNT	AT LEAST	STATUS
side_B	683747	1	
side_A	1135246	1	

Figure 6 Function Coverage Of Pointers

2.3 Design of interrupt handling module

In this paper, the interrupt handler based message is designed to receive and send interrupts. This module is divided into seven parts: interrupt source, CF, NOC, IH, IH_Buffer, MCU and operating system are shown in Figure 7. The IH to be verified in this paper consists of a ring FIFO, which is made of dual port SRAM and can work at 1200 MHz clock frequency. Because frame dropping exists in common FIFO, the current data cannot be guaranteed. Using ring FIFO can avoid frame dropping [7]. In order to ensure the data transmission speed of IH module under high-frequency clock, the AXI protocol applied in this paper adopts multi-channel transmission, and supports the transmission mode of outstanding, out of order and interleaving, which meets the requirements of high-speed transmission [8].

IH module is designed to be a CF slave which accepts interrupt requests from multiple interrupt masters. In order to achieve the convergence of system interrupt request, the interrupt received by IH is transmitted by message mode, which changes the traditional transmission mode through signal line. Such processing can also reduce the problem of routing congestion. There will be multiple interrupt sources in a chip. After the interrupt requests are sent by these interrupt sources, they are passed to NOC through cf. After the interrupt requests are further processed by NOC, they are sent to IH in the form of burst. IH can cache these interrupts, and then store the cached data into IH_buffer through NOC, it is also connected to MCU through intx signal line to send interrupt data to MCU for processing. IH_buffer indicates a period of memory located in either interrupt ring buffer .

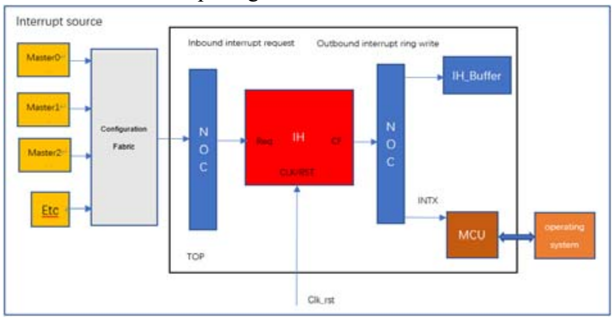


Figure 7 Interrupt Handler Structure Based Message

3 Conclusion

Compared with the traditional verification methods, the verification platform based on UVM has higher flexibility and reusability, which can be used to verify DUT quickly and efficiently. The improved test sequence of the register or memory scan accelerates the scan of the register or memory and improves the simulation efficiency of the verification platform. The verification platform validates the interrupt handling module , and realizes the verification of ring FIFO full and overflow in IH. The code coverage of the interrupt handling module reaches 100%, and the AXI bus meets the protocol requirements. The functional components of the verification platform have high reusability and portability for the verification of other interrupt systems.

REFERENCES

[1] K. Salah and H. Mostafa, "Constructing Effective UVM Testbench for DRAM Memory Controllers," 2018 New Generation of CAS (NGCAS), Valletta, 2018, pp. 178-181.

[2] Kotha S., Ravimony R., Mohankumar N. (2020) Automated UVM Based Verification of Device Life Cycle Management IP. In: Pandian A., Ntalianis K., Palanisamy R. (eds) Intelligent Computing, Information and Control Systems. ICICCS 2019. Advances in Intelligent Systems and Computing, vol 1039. Springer, Cham

[3] L. Shiva and L. Saiteja, " UVM based reusable verification IP for wishbone compliant SPI master core, " International Journal of VLSI Design and Communication Systems,pp. 21-29, October 2018.

[4] L. Kappaganthu, A. Yadlapati and M. Durga Prakash, "High level verification of I2C protocol using system verilog and UVM, " Smart Innovation, Systems and Technologies pp, 1-8, October 2017.

[5] X. Peng et al., "Function Verification of SRAM Controller Based on UVM," 2019 IEEE 13th International Conference on Anti-counterfeiting, Security, and Identification (ASID), Xiamen, China, 2019, pp. 1-5.

[6] IEEE Standard for Universal Verification Methodology Language Reference Manual," in IEEE Std 1800.2-2017 , vol., no., pp.1-472, 26 May 2017.

[7] S. Windmann and J. Jasperneite, "An FPGA based FIFO with efficient memory management," 2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA), Luxembourg, 2015, pp. 1-4.

[8] Malhotra S., Prakash N.R. (2019) UVM-Based Verification IP of AMBA AXI Protocol Showing Multiple Transactions and Data Interconnect. In: Bera R., Sarkar S., Singh O., Saikia H. (eds) Advances in Communication, Devices and Networking. Lecture Notes in Electrical Engineering, vol 537. Springer, Singapore.

[9] C. Elakkiya, N. S. Murty, C. Babu and G. Jalan, "Functional Coverage - Driven UVM Based JTAG Verification," 2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), Coimbatore, 2017, pp. 1-7.Conference Name:ACM Woodstock conference.