ISSN NO. 2320-5407

Journal homepage: http://www.journalijar.com

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH

RESEARCH ARTICLE

An efficient approach for Boundary Scan Verification for a System-on-Chip

Kiran K P [1*], Namita P[2], Siva Sankar A[3]

1. 4th Sem, M. Tech.VLSI Design and Embedded Systems, Dept. of E and C R. V. College of Engineering Bangalore, India.

2. Asst. Professor Dept. Of E and C R. V. College of Engineering, Bangalore, India.

3. Senior Component Design Engineer Intel Technology Pvt. Ltd. Bangalore, India.

Manuscript Info

Manuscript History:

Received: 14 May 2015
Final Accepted: 22 June 2015
Published Online: July 2015

Key words:

Test Access Port (TAP), SystemVerilog, Open Verification Methodology (OVM), Bus Functional Model (BFM), Boundary-scan, Test Clock (TCK), Test Data Input (TDI), Test Data Output (TDO), and Alternate Current (AC).

***Corresponding Author**

Kiran K P

Abstract

System-on-Chip is an Integrated Circuit(IC) that integrates different modules on a single chip. It consists of different Intellectual Properties (IP's) from different vendors. Verifying such a complex IC is challenging as it consists of analog, digital and glue logic on a single chip. As a part of pre-silicon verification, Boundary-scan involves checking interconnection between different components. Due to integration of different IP's with forming heterogeneous pin structure at SOC level, boundary-scan cells incorporated per pin will also vary according to the pin structure. Some pins are single ended and others may be differential, some are slow speed pins and some pins allow high-speed data transaction. Boundary-scan verification infrastructure needs to be updated for different projects, as it involves updating the pins present at SOC level. In this project, an approach is proposed to verify boundary-scan logic for an SOC. Automated script is developed to automatically update the changes required. This helps in reducing time spent in updating the changes, which would require a manual intervention of couple of weeks.

I. Introduction

As SOC integrates different Intellectual Properties (IP's), with each IP designed for particular operation based on the specification obtained from the customer, integrating them on a single chip requires proper interface as there may be changes in electrical properties for different pins. To access remote parts of SOC and interconnection between the components, test logic defined by the standard 1149.1, TAP and boundary-scan architecture will be inserted. The TAP network [1] architecture defined by the IEEE standard 1149.1 provides access to different register contents situated across different IP's, while boundary-scan architecture concentrates on the pin level data transaction. Boundary-scan test logic consists of boundary-scan cell integrated along the pins of SOC with few exceptions such as, power rails and clock signals which will not include boundary-scan cells. Different IP's integrated will contain different set of I/O's. Depending on the type of the I/O, single ended, differential, number of boundary-scan cells per pin will vary. After integrating all the boundary-scan cells, boundary-scan should be verified at SOC level.

Pre-silicon verification involves data transaction between the registers of same module and with other modules and checking a set of data after every transaction. With the help of specification document and architecture of models

and that of SOC, data can be presumed for each transaction. TAP network provides access to registers and transaction data can be checked as and when required. TAP controller provides ability to observe and control the TAP network states, signals and registers which can be accessed by OVM[3] environment agents and monitors[4] for checking data flow at different intervals of time and trackers can be used to inform the type of transaction which have been executed at different intervals of time with specific data.

II. Boundary-scan Overview

As boundary-scan is used to obtain ability to observe and control of pins, a shift register stage is incorporated adjacent to each pin of every module of on-chip system logic. Boundary-scan cell forms test logic incorporated into the design, whose implementation is illustrated in figure 1. If a pin is used as input, boundary-scan cell appears after the pin and connects to system logic. In case a pin tends to be output, system logic is connected to boundary-scan cell and thus to the output pin via boundary-scan cell. A bi-directional pin is associated with a pair of boundary scan cells, as the data driven from or into the pin via single buffer.

Boundary-scan technique supports a number of instructions based on which different set of test can be implemented. With the help of TAP controllers from SOC level, boundary-scan can be enabled for whole SOC as well as for a particular module without affecting the functionality of the adjacent modules. A Centralized TAP Controller (CLTAPC) will have control over other modules via Embedded TAP Controller (EMTAPC)[5]. Test cases are written based on methods derived out of Boundary Scan Description Language (BSDL) and Procedural Description Language (PDL)[6]. To select a particular register whose final data has to be checked after a transaction which would be residing in a particular IP, where direct access to remote registers is not recommended and impossible. EMTAP of the IP is selected via CLTAP and then the register is accessed by placing CLTAP and EMTAP in bypass mode.

If on-chip system logic is under scan mode, all the boundary-scan cells are connected in a manner to form a daisy chain, starting from CLTAPC to all boundary-scan cells which are to be tested and back to CLTAPC. By knowing the scan length, data can be shifted into scan-chain until it reaches the last boundary-scan cell just before CLTAPC.

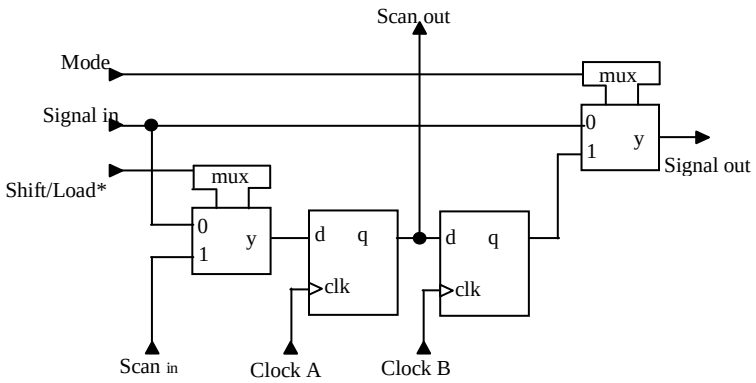

Figure 1: A Boundary-Scan Cell

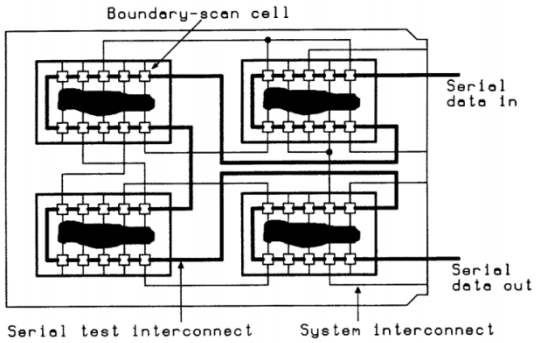

Figure 2: Implementation of boundary-scan at SOC level

Under normal operation, on-chip system logic behaves as if there are no boundary-scan cells where the external signals are directly connected to system logic pins without the intervention of intermediate flops. During scan mode, Clock A shifts-in data into boundary-scan cell as well as shifts the data along shift register stage. Clock B updates all the system-logic inputs with test data in the normal mode. Output of the system-logic is computed up-front for the corresponding test data, so that after updating system-logic works in normal mode. The result can be obtained by shifting out the data present in boundary-scan cells, by keeping the system in scan mode. Thus shifted-out data is compared with pre-computed result. In case if there is any mismatch, a bug will be reported which has to be root caused and fixed by the designer.

When on-chip system logic is in scan mode, all the boundary-scan registers are interconnected to form a shift register-stage. If a module is to be by-passed, which can be done by placing the corresponding EMTAP in bypass mode where it inserts a single flop delay by placing a bypass register at the intermediate stage. Figure 2 illustrates the implementation of boundary-scan technique at SOC level.

III. Boundary-scan instructions

To validate boundary-scan logic, boundary-scan instructions are loaded into the boundary-scan register and then test data are applied to boundary-scan cells. Following section gives the boundary-scan instructions executed in the project.

1. BYPASS instruction

BYPASS instruction will place a Bypass register which is one-bit register, between TDI and TDO. Entire System-logic can be bypassed by loading BYPASS instruction into the instruction register.

2. SAMPLE/PRELOAD instruction

SAMPLE instruction is used to take snapshot of signal value on the pins into the boundary-scan cells. SAMPLE instruction is used to ensure the data present at the pins, which will flow into the system-logic. PRELOAD instruction will keep all the boundary-scan cells serially to form a shift register stage between TDI and TDO.

3. EXTEST Instruction

EXTEST instruction is used to test off-chip circuitry. External components can be tested by applying stimulus through boundary-scan cells, while data coming from the external components into the boundary-scan cells can be used to capture data.

4. EXTEST_PULSE Instruction

TAP controller will transit to Test-Run/Idle state and signal values on pins will undergo a pulse transition.

5. EXTEST_TRAIN Instruction

When EXTEST_TRAIN instruction is loaded, it causes pins to toggle continuously.

Both EXTEST_PULSE and EXTEST_TRAIN instructions are used to check stuck-at, short-circuit, open-circuit faults, which can also be used to check AC-coupled pins.

6. HIGHZ Instruction

HIGHZ instruction will force a high-z value on the pins irrespective of previous value.

IV. Automation

Boundary-scan technique is adopted as a standard in all integrated circuit designs. Validating boundary-scan at SOC level involves execution of boundary-scan instructions, in which pins are forced for executing SAMPLE instruction. SAMPLE instruction will take a snapshot of signal value present at the pins where as PRELOAD instruction is used to shift-in the bit pattern through TDI, into boundary-scan chain. Therefore boundary-scan verification environment

contains pin details, which often needs to be updated when reusing the verification collaterals across different projects. Update process requires a couple of weeks which needs manual intervention. Manual intervention again may lead to errors and pin-mismatches while updating the details. Therefore update process is automated in this project by using Perl scripting. The Perl script which has developed will take specification document as input and will update the pin changes required for boundary-scan verification environment, used in different projects. The automation process reduces update effort and verification time consumed can be reduced by couple of weeks. In addition, efficient task/function can be developed by using Perl scripting. Figure 3 shows the flow chart of Perl script execution. It takes specification document as input. The specification document will contain all the information related to boundary-scan such as, pin names, I/O family order while stitching boundary-scan cells, number of boundary-scan cells per pin, type of pins and pin order within the I/O family.

The script will generate "Bscan_base_seq.sv" file, which is a class based SystemVerilog file from which boundary-scan test will be derived. In addition, it contains all the necessary SystemVerilog methods (task/function). These methods are developed according to the test requirement. SAMPLE instruction requires forcing all the pins and then these forced values are sampled into the boundary-scan cell by loading SAMPLE instruction. EXTEST_PULSE, EXTEST_TRIAN and HIGHZ instructions will check the value on the output/bi-directional pins. Individual tests cannot force and check all the pins at SOC level, so hierarchical structure of OOP concept helps in placing all the necessary methods in a base class and test cases are derived from this base class. As the base class contains pin names and all the methods with respect to each family, automated script will overwrite the previous code and update it whenever the script is executed. As the pin list is directly taken from the specification document, both the script and boundary-scan verification environment can be reused, with small or no modification in test cases.

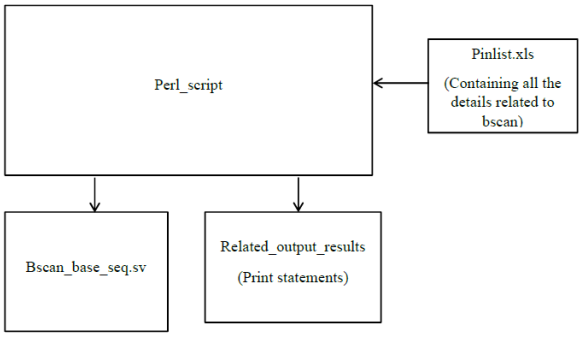

Figure 3: Flow chart Perl script execution.

V. OVM environment

SystemVerilog provides wide range of capabilities and OVM is one of the methodologies developed to ease verification. Test cases are developed based on object-oriented programming concept, typical OVM environment is illustrated in figure 4. A test invokes a sequence in which all the required flow for a particular test is programmed. Before starting actual sequence for a test, initially system is to be brought out of reset and all the necessary clocks should be up and running. OVM provides a wide range of pre-defined functions/macros, reporting errors, displaying messages, defining parameters, which remain constant throughout the execution of a test and defining a particular set of variables tightly related to design and can be modified according to the changes in the design to provide flexibility and re-usability, along with other virtual components to aid the verification process. Reusable IP's are available from different vendors, which can be used in SOC design to minimize the design effort and reusable IP's are already verified by the vendor, so adopting such design will minimize verification effort.

To execute boundary-scan tests, boundary-scan instruction is loaded into the instruction register, which selects a set of test data registers without affecting remaining test data register. Selected test data registers will interact with system-logic without affecting its normal functionality. Access to system-logic is provided via test-logic, when the system is kept under test, with the help of interfaces from design to test-bench environment, to gain control over remote areas of on-chip system logic.

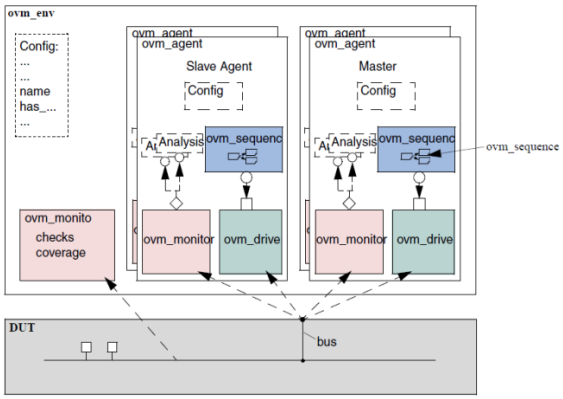

Figure 4: A typical OVM environment

VI. Result

Boundary-scan tests are executed at SOC level, by loading boundary-scan instruction into the instruction register. A simple boundary-scan test may check for a chain length connectivity in which by knowing the chain length data is shifted-in by keeping design under scan mode. For simple chain connectivity, the data scanned-in has to be obtained at the output, as there is no intervention of system logic. If all the pins are properly designed, shift-in and shift-out data should be similar with a delay equal to chain length. Figure 5 shows the result obtained from a boundary-scan chain connectivity test.

BYPASS instruction incurs a single bit delay between TDI and TDO. Figure 6 shows simulated waveform after running bypass test.

To execute SAMPLE/PRELOAD instruction, pins are forced to a particular value (0 or 1). When SAMPLE/PRELOAD instruction is loaded, SAMPLE instruction will take a snapshot of values on the pins, which can be shifted-out to check for correctness of pins. Figure 7 shows the snapshot of simulated waveform for SAMPLE/PRELOAD instruction.

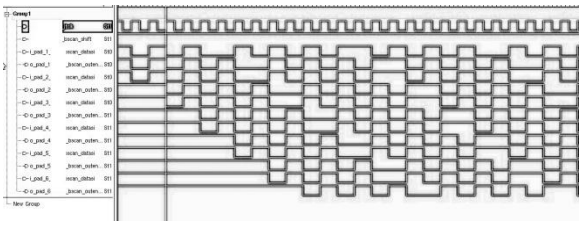

Figure 5: Snapshot simulated waveform for chain connectivity test

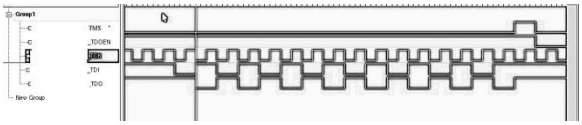

Figure 6: Snapshot simulated of waveform for bypass test

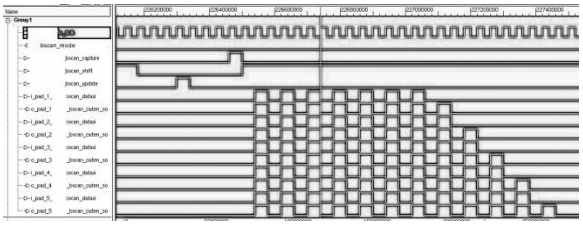

Figure 7: Snapshot of simulated waveform for sample_preload test

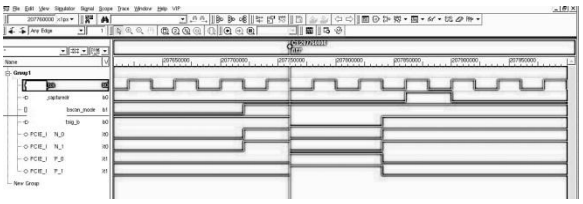

Figure 8: Snapshot of simulated waveform for extest_pulse test

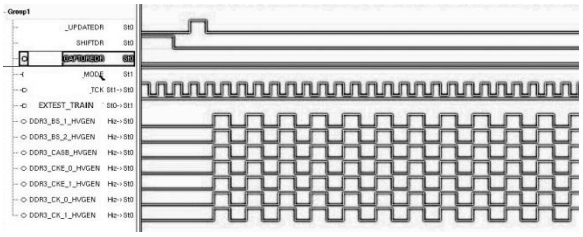

Figure 9: Snapshot of simulated waveform for extest_pulse test

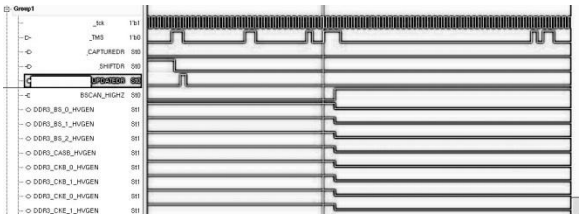

Figure 10 – Snapshot of simulated waveform for high-z test

EXTEST_PULSE instruction will make pins to undergo a pulse transition. Before executing EXTEST_PULSE or EXTEST_TRAIN instructions; a known pattern is shifted into the boundary-scan data cells. EXTEST_PULSE will make pins to toggle with frequency twice that of TCK. Figure 8 and 9 shows simulated waveform after running EXTEST_PULSE and EXTEST_TRAIN instructions respectively.

HIGHZ instruction will force high-z value on the data pins. Figure 10 shows simulated wave form result for HIGHZ instruction.

VII. Conclusion

By implementing boundary-scan technique, intermediate pins can be observable and controllable. Most of the issues will be solved by executing all the boundary-scan instructions, which will result in a good quality product. Insertion of boundary-scan cells adds delay and increases the chip area, but it will be helpful for on-chip debug purposes. As the Perl script developed automatically updates boundary-scan verification related changes, code re-usability is increased and verification time is reduced by couple of weeks. As a future work, entire boundary-scan related verification collaterals can be automated to execute boundary-scan validation process in couple of days.

ACKNOWLEDGEMENT

The authors would like to thank Intel Corporation for the valuable support. The guidance provided by Shabbir H Topiwala, Manjunatha B Prabhushankar, Siva Sankar at Intel Corporation and Namita P in-spite of their busy schedule, were instrumental in progress of the project.

REFERNCES

[1] IEEE Std 1149.1-1990, "IEEE Standard Test Access Port and Boundary-Scan Architecture", IEEE Standards Board, 345 East 47th Street, New York, NY 10017-2394, June 17, 1993.

[2] "SYSTEMVERILOG FOR VERIFICATION: A Guide to Learning the Testbench Language Features" by Chris Spear, Synopsys, Inc.

[3] OVM Class Reference Version 2.1.2 June 2011.

[4] OVM User Guide Version 2.1.2 June 2011.

[5] IEEE Std 1149.7 -2009, "IEEE Standard for Reduced-Pin and Enhanced-Functionality Test Access Port and Boundary-Scan Architecture", IEEE-SA Standards Board, February 2010.

[6] IEEE Std 1149.1 -2013, "IEEE Standard for Test Access Port and Boundary-Scan Architecture", IEEE Standards Board, 6 February 2013.