Combining Internal Scan Chains and Boundary Scan Register: A Case Study

Conference Paper · May 2009

DOI: 10.1109/EURCON.2009.5167932 · Source: IEEE Xplore

CITATIONS
5

READS
2,003

3 authors, including:

Zoran Stamenkovic
IHP, Frankfurt (Oder)
125 PUBLICATIONS **456** CITATIONS

SEE PROFILE

COMBINING INTERNAL SCAN CHAINS AND BOUNDARY SCAN REGISTER: A CASE STUDY

Zoran Stamenković, *Senior Member*, *IEEE*, Mary Giles, and Francisco Russi

Abstract: **The paper presents a Design-For-Testability (DFT) approach for System-on-Chips (SOC) that combines internal scan chains and boundary scan register (BSR) into a single scan register known as Scan-Through-TAP (STT) methodology. We are using the IEEE Standard 1149.1 Instruction as a user defined instruction (UDI) to control the internal scan chains operation via TDI and TDO saving the scan pins, and allowing accessibility of the internal scan chains at the board level through the TAP controller.**

Index Terms: **Netlist, scan chain, boundary scan register, IEEE standard 1149.1, scan-through-TAP, test pattern generation**

I. INTRODUCTION

Today's emerging sub-micron silicon technologies bring enormous gains in respect of chip complexity, but they also dramatically increase design and verification complexity. A System-on-Chip (SOC) integrates processor cores, memory blocks, and custom logic into a closed system that must be assembled, verified and tested within a short timeframe [1]. With functional testing, the tester applies a sequence of system input data and detects the resulting sequence of system output data. The output sequence is compared against the expected behavior of the system. Functional testing exercises the system as it would actually be used in the target application. However, this type of testing has only a limited ability to test the integrity of the system's internal nodes.

Therefore, System-on-Chip (SOC) design methodology assumes incorporation of the additional advanced Design-For-Testability (DFT) features like chains of scanable flip-flops, boundary scan shift register, built-in-self-test logic, built-in current monitors, etc. [2]-[5]. It should provide both designer and user with a leverage for the complete (all the system components against every known defect/fault/failure type), meaningful (in respect of the results), and proper (in respect of the necessary resources) testing of a SOC.

In this paper, we concentrate on scan testing, where the sequential elements of the system are connected into chains and used as primary inputs and primary outputs for testing purposes. Using Automatic Test Pattern Generation (ATPG) techniques, it is possible to test a much larger number of internal faults than with functional testing alone. The goal of ATPG is to set all nodes of the circuit to both 0 and 1, and to propagate any faults to nodes where they can be detected by the test equipment.

In the next section (Section 2), we briefly review the traditional DFT techniques (scan chain insertion and boundary scan design) and ATPG. Also, we describe a DFT technique that relies on concatenating internal scan chains and boundary scan register into a single chain between test input and test output ports. In Section 3, we explain in detail how to implement the Scan-Through-TAP (STT) capability and generate the corresponding scan test patterns for a typical SOC comprising of the processor core, caches, system-bus (AMBA), memory controller, and several peripherals. Final results are discussed in Section 4. Summary is presented in Section 5.

II. DFT IN SOC DESIGN FLOW

SOC design flow starts with the creation of a HDL file describing the system and its modules as black boxes. Then configurable modules (IP cores) and custom modules are automatically synthesized and written out as logic netlists. After reading the HDL source code of the system (top-level) module, the system is linked to the logic netlists of synthesized modules. Now the system is ready for scan chain and boundary scan logic insertion.

II.a Scan Chain Design

In a scan chain, sequential elements (flip-flops) of a design are daisy-chained to provide stimulus and observe points internal to the design. Scan chains are implemented using multiplexers to create a shift register out of the existing sequential elements. Scan chains can be inserted after the design is logically synthesized in the form of a gate-level netlist.

Fig.1 below shows all the main steps of scan chain insertion and test pattern generation for scan testing. Before inserting the scan structure, we specify the scan architecture: scan ports, scan methodology, scan style, length of scan chains, number of scan chains, way of handling multiple clocks, three-state nets and bidirectional ports, etc. Next step (a preview of the scan architecture) reports the name, scan-in port, scan-out port and length of each scan chain, and displays the effects of the current specifications, which can be modified as many times as needed until getting the satisfactory results.

978-1-4244-3861-7/09/$25.00 ©2009 IEEE

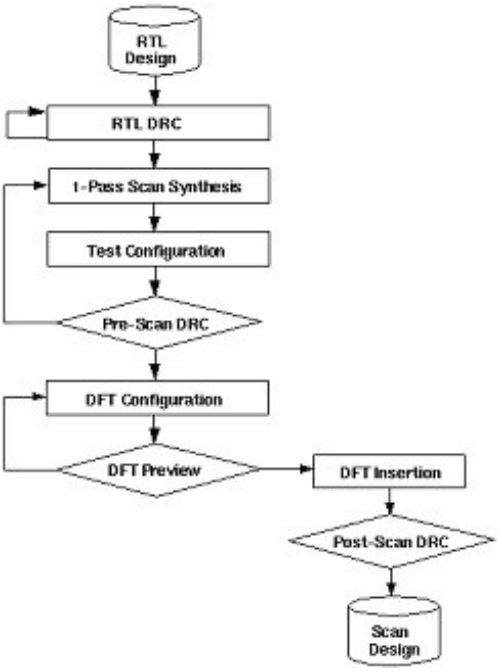

Fig.1 – Overview of the scan insertion flow

To insert and route the internal scan chain, the scan logic is synthesized from the target (technology) library. Scan equivalents of the non-scan library cells are initially selected based on library function identifiers. If no scan equivalent for a storage element is found, constraint-based sequential mapping techniques are used. These techniques consider the logic function implemented by each flip-flop to be replaced along with immediately surrounding logic. This consideration generates a set of possible alternatives, from which the best replacement is selected based on scan style, design rule considerations (for example, maximum fan-out), and design area. By default, as many scan chains as there are clocks and edges are constructed. By setting the clock mixing and chain count options, the number of scan chains can be controlled.

II.b Boundary Scan Design

Fig.2 shows a typical boundary scan design flow including boundary scan specifications, preview of boundary scan logic, insertion and optimization of boundary scan logic, the IEEE Standard 1149.1 JTAG compliance check, and generation of a Boundary Scan Design Language (BSDL) description and boundary scan test patterns.

To insert the IEEE Standard 1149.1 JTAG compliant boundary scan logic using macro cells [6], the four mandatory test access ports (TAP) must be specified. The pads on the top-level module ports must be present. The synthesized logic is mapped and the boundary scan cells are uniquified. By default, the Instruction Register (IR) is synthesized with the

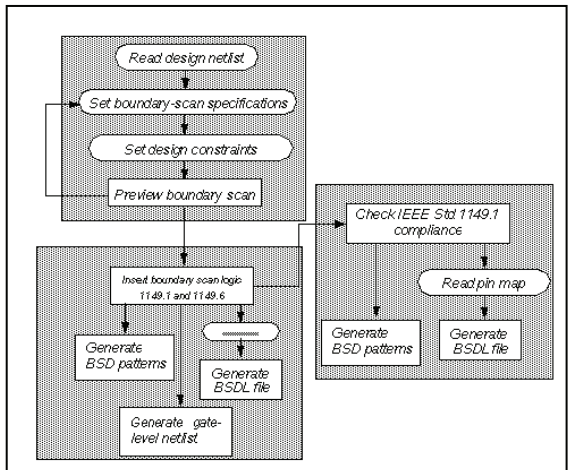

Fig.2 – Overview of the boundary scan design flow

minimum number of bits to accommodate all the selected instructions.

II.c Scan-Through-TAP

Scan-Through-TAP flow starts with definition of a STT register (STT_REG) and the specification of a STT instruction and other user instructions. It assumes existence of already inserted internal scan chains and corresponding test pins. The flow continues with implementation of a boundary scan shift register and its concatenation to the existing scan chains. This implementation step creates a single scan shift register (Fig.3) with boundary scan cells and internal scan cells concatenated together. After checking the IEEE Standard 1149.1 compliance, a STT test protocol file is generated by Boundary Scan Design tool. Optionally, it is possible to write an ATPG test protocol file with TAP in pass-through mode using DFT tool.

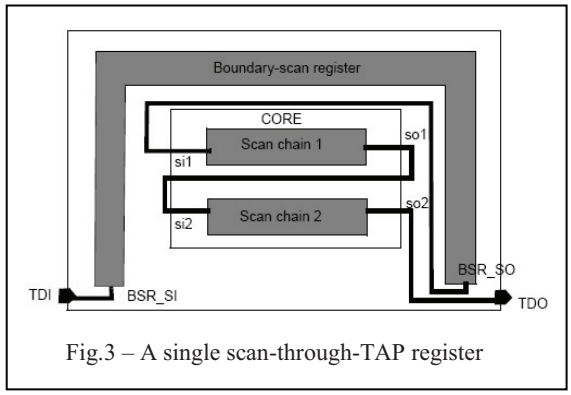

Fig.3 – A single scan-through-TAP register

III. IMPLEMENTATION OF STT

We have selected a general purpose processor based SOC to implement and test the features of described STT technique. It consists of a 32-bit processor core, caches, a combined PROM/SRAM memory controller, an AMBA bus (AHB, APB, AHB controller and AHB/APB bridge), and a standard set

of peripheral cores (timers, UARTs, I/O port, interrupt controller and debug interfaces).

Fig.4 below shows the EDA tools used for the complete implementation of the scan-through-TAP flow.

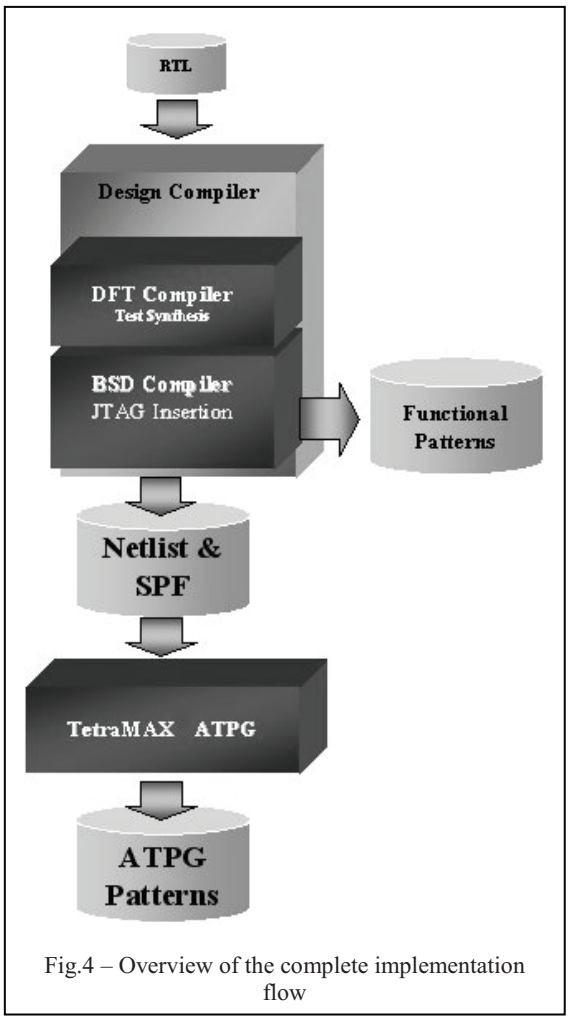

Fig.4 – Overview of the complete implementation flow

After the configured processor system (including SRAM models) had been verified by simulations, we have performed the logic synthesis. The system with directly instantiated SRAM blocks and pads has been synthesized for a target frequency of 125 MHz using Synopsys Design Compiler [7].

The tools used to insert internal scan (Synopsys DFT Compiler [8]) and boundary scan (Synopsys BSD Compiler [9]) are both run within the Synopsys Design Compiler environment. Synopsys TetraMAX Automatic Test Pattern Generator [10] is used afterwards.

III.a Internal Scan Chain Insertion

Synopsys DFT Compiler [8] is used to insert the internal scan chain. In this example, there is only one internal scan chain. It is also possible to specify multiple parallel scan chains, which would be concatenated into one single chain during BSD insertion. Commands description follows:

System Scan Insertion Script

Read in synthesized design

```
read_file -f verilog system.v
current_design system
link
```

Define clock constraints

```
create_clock -name clk -period 10 -waveform {0 5}
{clk}
set_clock_latency -rise 0.4 clk
set_clock_latency -fall 0.4 clk
create_clock -name tck -period 50 -waveform {0 25}
{tck}
set_clock_latency -rise 0.4 tck
set_clock_latency -fall 0.4 tck
set_false_path -from clk -to tck
set_false_path -from tck -to clk
set_fix_multiple_port_nets -all -buffer_constants
```

Define scan chain

```
set_scan_configuration -clock_mixing mix_clocks \
      -identify_shift_registers true -chain_count 1
set_dft_signal -view existing_dft -type ScanClock -
timing \ {45 55} -port clk
set_dft_signal -view spec -type ScanEnable -port
test_se
set_dft_signal -view spec -type ScanDataIn -port
test_si
set_dft_signal -view spec -type ScanDataOut -port
test_so
create_test_protocol
dft_drc
```

Insert scan chains

```
preview_dft -show all
insert_dft
```

Write out test protocol and netlist for TetraMAX

```
change_names -rules verilog
write_test_protocol -out system_scan.spf
write -f verilog -hier -out system_scan.v
```

III.b Boundary Scan Insertion

Synopsys BSD Compiler [9], [11] is used to implement the basic test data registers (TDR) including the Instruction Register (IR), BYPASS, BOUNDARY, optional Device Identification Register (DIR), and any TAP controlling logic for a User Test Data Register (UTDR) such as the STT_REG register. In this design, the boundary scan register (BSR) is combined with the already implemented scan chain to create the STT_REG register. (Alternatively, it is possible to exclude the BSR chain and create TAP controlling logic for the scan chain alone). We used the following sequence of commands to insert the boundary scan logic:

Boundary Scan Insertion Script

Read in scan-inserted design with IO pads

```
read_file -f verilog system_scan.v
read_file -f verilog system_io.v
current_design system_ top
```

Define system and test clock constraints

```
create_clock -name clk -period 10 -waveform {0 5}
{clk}
set_clock_latency -rise 0.4 clk
set_clock_latency -fall 0.4 clk
create_clock -name tck -period 50 -waveform {0 25}
{tck}
set_clock_latency -rise 0.4 tck
set_clock_latency -fall 0.4 tck
set_false_path -from clk -to tck
set_false_path -from tck -to clk
set_fix_multiple_port_nets -all -buffer_constants
```

Define JTAG configuration

```
set_dft_configuration -bsd enable -scan disable
set_bsd_configuration -asynchronous_reset false
set_dft_signal -view spec -type tdi -port test_tdi
set_dft_signal -view spec -type tdo -port test_tdo
set_dft_signal -view spec -type tms -port test_tms
set_dft_signal -view spec -type tck -port test_tck
```

STT register implementation

```
set_dft_signal -type tdi -hookup_pin I0/test_si
set_dft_signal -type tdo -hookup_pin I0/test_so
set_dft_signal -type bsd_shift_en -hookup_pin
I0/test_se
set_dft_signal -type capture_clk -hookup_pin I0/clk
set_scan_path STT_REG -class bsd -exact_length 4 \
   -hookup {BSR_SI BSR_SO \
            I0/test_si I0/test_so \
            I0/test_se I0/clk}
set_bsd_instruction STT -reg STT_REG
```

Insert JTAG logic

```
preview_dft -bsd all
insert_dft
```

Check for IEEE-1149.1 compliance

```
check_bsd
```

Create and write out BSD test patterns

```
create_bsd_patterns
write_test -f stil_testbench -out system_io_bsd
write_test -f wgl_serial -out system_io_bsd
```

Write out protocol file and netlist for TetraMAX

```
change_names -rules verilog
write_test_protocol -instruction STT \
            -o system_io_scan_bsd.spf
write -f verilog -hier -o system_io_scan_bsd.v
```

III.c Fault-Grading and Test Pattern Generation

The basic ATPG flow applies to designs that are developed using test design rules and for which a test protocol file is generated by DFT tool. It starts with reading in the design netlist and the library cell models. Then, the ATPG design model is built and the test protocol file is read in. Next step is the test design rule check. If violations encountered, it might be necessary to make corrections. Before running the test pattern generator, the fault list is set up and buses are analyzed for contention. If the fault coverage is high enough (typically, 95% or higher), the test patterns are compressed and saved.

Synopsys TetraMAX ATPG [10] is used to fault grade the patterns and generate the test patterns in the form of WGL files. The STIL Protocol File (SPF) generated from BSD Compiler contains all the information required by TetraMAX to control the tap and access the internal scan chain. Verilog STIL testbenches have also been prepared for simulation of all scan data. The sequence of TetraMAX ATPG commands is as follows:

ATPG Script

Read in library cell models and design

```
read netlist ../synopsys/zxall.v
read netlist ../../layout/netlist/system_io_scan_bsd.v
```

Build ATPG model

```
set build -black_box embedded_mem_1
set build -black_box embedded_mem_2
set build -black_box embedded_mem_3
run build_model system_top
```

Check design rules

```
run DRC ../../syn/system_io_scan_bsd.spf
```

Add faults, and generate and compress test patterns

```
add faults -all
run atpg –auto
```

Write out test patterns

```
write patterns system_io_scan_bsd.wgl -f wgl
write patterns system_io_scan_bsd_testbench.v \
          -f verilog_single_file -serial
```

III.d Simulation of Test Patterns

Both the BSD functional test patterns and the scan test patterns are simulated using the Synopsys VCS Simulation tool [12]. The simulation environment included the back-annotated timing information and all the analog and memory behavioral models. Waveforms of the BSD test pattern simulations are presented in Fig.5. Waveforms of the scan test pattern simulations with TAP in pass-through mode are presented in Fig.6. The pattern sequence shows the capture cycle and the corresponding signals (BSD signals and TAP state signals).

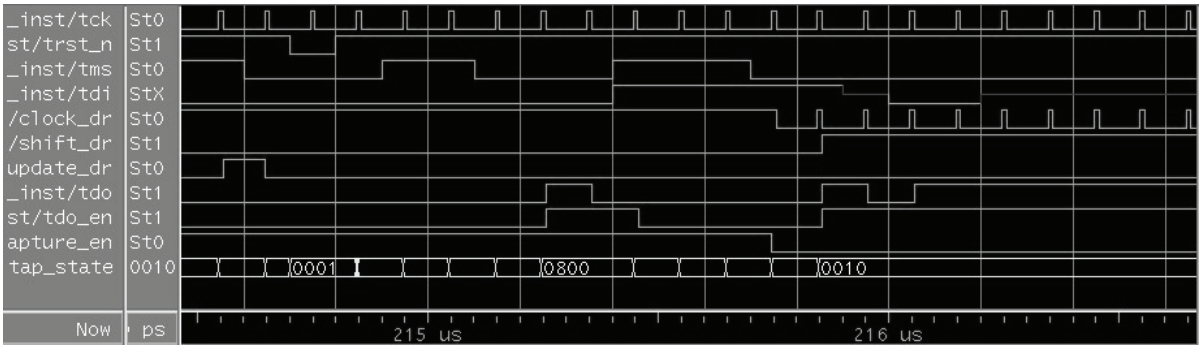

Fig.5 – Simulation of the BSD test patterns generated by BSD Compiler

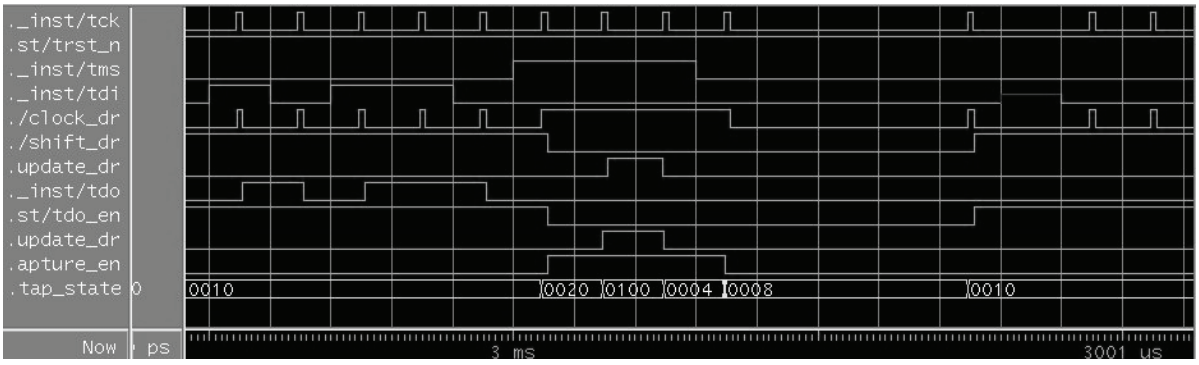

Fig.6 – Simulation of the STT test patterns generated by TetraMAX ATPG

IV. RESULTS

For the implemented single scan register (made of around 15000 scanable flip-flops and the boundary scan register of 151 cells), we have generated more than 32000 BSD functional test patterns and 1151 Synopsys TetraMAX ATPG test patterns. The chip area overhead caused by insertion of scanable flip-flops and boundary scan logic has been estimated below 7%. The combined fault coverage is slightly higher than 94% and met our goal.

V. CONCLUSION

We described the DFT Scan-Through-TAP methodology that provides the infrastructure for both testing of system's internal nodes and testing of system's boundary circuitry. Implementation of a single scan register and generation of scan test patterns are illustrated on the example of a middle-size processor-based SOC. The results show small chip area overhead, acceptable number of test patterns, and high fault coverage, achieved in a short development time.

REFERENCES

[1]. W. Wolf, *Modern VLSI Design: System-on-Chip Design (3rd Edition)*, Englewood Cliffs, New Jersey: Prentice Hall, 2002.

[2]. B. Koenemann, *Design for Test*, in *Electronic Design Automation for Integrated Circuits Handbook* (edited by L. Scheffer, L. Lavagno, and G. Martin), Taylor & Francis, 2006.

[3]. H. Bhatnagar, *Design for Test*, in *Advanced ASIC Chip Synthesis: Using Synopsys Design Compiler, Physical Compiler and PrimeTime*, Second Edition, Kluwer Academic Publishers, 2002.

[4]. Benoit Nadeau-Dostie, *Design for At-Speed Test, Diagnosis and Measurement*, Springer US, 2002.

[5]. Y. Zorian, S. Dey, and M.J. Rodgers, "Test of Future System-on-Chips," *Proc. IEEE/ACM International Conference on Computer Aided Design*, San Jose, California (USA), 2000, pp. 392-398.

[6]. DW-TAP Design Specification Doc., Synopsys Inc.

[7]. Design Compiler, Synopsys Inc.

[8]. DFT Compiler, Synopsys Inc.

[9]. BSD Compiler, Synopsys Inc.

[10]. TetraMAX ATPG, Synopsys Inc.

[11]. Application Notes: Design-For-Testability with 1149.1 JTAG Components, Test Access Port, and Boundary Scan IP, Synopsys Inc.

[12]. VCS Simulator, Synopsys Inc.

Dr. Zoran Stamenković is with IHP GmbH, Frankfurt (Oder), Germany. He received his Ph.D. degree in electronic engineering from the University of Niš, Serbia in 1995. His research interests include wireless SOC design, HDL modeling, logic synthesis, chip layout, and IC yield modeling and prediction. Currently, he is leading the EU funded project on a wireless MIMO system (MIMAX) at

IHP GmbH. He has published a book on IC yield, two chapters on IC yield and testing in the Computer Engineering Handbook (Outstanding Academic Title for 2002 by Choice Magazine, USA), and more than 40 scientific journal and conference papers.

Mary Giles is with Synopsys Inc., Mountain View, CA, USA. She received her BS degree in electrical engineering from Northeastern University, Boston, MA in 1986. She worked as a system/board-level design and test engineer for the first 10 years at MIT Lincoln Laboratory, Loral and Lockheed, and the last 13 years as a test specialist in the semiconductor industry at National Semiconductor and Synopsys. Her areas of expertise include Scan synthesis, automated test pattern generation, IDDQ, At-speed testing, Scan Compression, JTAG and design synthesis.

Francisco Russi is with Synopsys Inc., Mountain View, CA, USA. He received his BSEE from University of San Antonio TX, 1994, his AAEE from Triton College, IL, 1984, CQE from ASQ, 1985, and the CSSBB from Motorola Technical University, 1987. He has work expertise in the semiconductor industry for more than 25 years in the area of Design for Test (DFT) applications and synthesis including Scan Test, Memory BIST, IC Self tst, JTAG design for ASIC/SoC as well as IC functional verification, and Design for Manufacturing (DFM) in Quality Assurance and Process Control. He was instrumental in helping Motorola to win the Malcolm Baldrige Quality Award in manufacturing in 1988, and implementing the Six Sigma Quality program across the organization.