Build Testbenches for Verification in Shift Register ICs using SystemVerilog

by Widianto Widianto

Submission date: 19-May-2023 08:47AM (UTC+0700)
Submission ID: 2096689619
File name: 70-3488-Widianto-sk.pdf (511.92K)
Word count: 3223
Character count: 16487

INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2022, VOL. 68, NO. 3, PP. 619-623
Manuscript received September 17, 2021; revised August, 2022. DOI: 10.24425/ijet.2022.141281

Build Testbenches for Verification in Shift Register ICs using SystemVerilog

Widianto, M. Chasrun H., and Robert Lis

***Abstract*—A testbench is built to verify a functionality of a shift register IC (Integrated Circuit) from stuck-at-faults, stuck-at-1 as well as stuck-at-0. The testbench is supported by components, i.e., generator, interface, driver, monitor, scoreboard, environment, test, and testbench top. The IC consists of sequential logic circuits of D-type flip-flops. The faults may occur at interconnects between the circuits inside the IC. In order to examine the functionality from the faults, both the testbench and the IC are designed using SystemVerilog and simulated using Questasim simulator. Simulation results show the faults may be detected by the testbench. Moreover, the detected faults may be indicated by error statements in transcript results of the simulator.**

***Keywords*—testbench; verification; shift register IC; stuck-at-faults; SystemVerilog**

I. Introduction

SHIFT register Integrated Circuits (ICs) may be applicated to serial-to-parallel data conversion and remote control holding register[1]. The ICs may be implemented to displays and control units. They consist of sequential logic circuits, i.e. D-type flip-flops. Stuck-at-faults may occur at either inputs or output interconnects between the circuits inside the ICs[2].

The faults may be caused by shorted the interconnects to supply voltage (stuck-at-1) and ground (stuck-at-0)[3]–[5]. The inputs and outputs of the circuits may stick at high and low values caused by the stuck-at-1 and stuck-at-0, respectively. Since the faults may cause functions of the ICs become errors, they should be detected.

The faults may be detected by an Automatic Test Pattern Generation (ATPG)[6]–[9]. However, it needs to integrate the external tools and programs where integrating them is not easy. Furthermore, it requires a circuit partitioning and a bit parallel processing.

A Built-in Self Test (BIST) was proposed to detect the faults[10]–[12]. The BIST consisted by a signature analyzer[13]. The analyzer is used to compare signals generated by the faults and faults-free and further analyze them. However, it takes time consuming, since the signal faults are detected by checking the generated signals one-by-one. Furthermore, the BIST is still designed by a Verilog Hardware Description Languages (HDL) in which it isn't supported by Object Oriented Programming (OOP) features[14]–[16], thus the detected faults aren't indicated by pass/ error statements.

A SystemVerilog is an extension of the Verilog HDL[17]–[21] used to design and verify hardware[22][23], e.g., Field Programmable Gate Arrays (FPGAs) and Application-Specific Integrated Circuits (ASICs). The hardware design is created in a Register Transfer Level (RTL) model. Moreover, to verify functionality of the design working as expected, a testbench should be built[24]. The testbench drives different input stimulus to the design and is supported by the OOP features.

In this paper, a proposed shift register IC is designed using the SystemVerilog. It will be verified the functionality from both stuck-at-faults, stuck-at-1 and stuck-at-0 by a testbench. The designed IC and its testbench are simulated using Questasim simulator[25]. Simulation results denote that the faults inside the IC may be detected by the testbench. Furthermore, indicating pass/ error statements is included in the detected faults by it.

II. Research Method

A testbench is built to verify a functionality of a shift register IC from both stuck-at-faults, stuck-at-1 and stuck-at-0. The testbench architecture is shown in Figure 1. There are some supported components, namely, transaction object, generator, interface, driver, monitor, scoreboard, environment, test, and testbench top. The IC is as a Design Under Test (DUT).

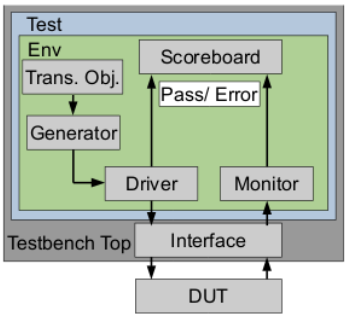


Fig.1. Testbench architecture

The DUT is a sample 8-bit shift register IC produced by Nexperia Co. Ltd. A logic diagram of the DUT is shown in Figure 2. As shown in Figure 2, the DUT consists of four buffer gates, two inverter gates, and eight stages, stage 0 to stage 7. Each of the stages is made of a buffer gate and two D flip-flop types in which the D flip-flop types are a shift register *sh* and a storage register *st*. Symbol descriptions in the DUT are shown in Table I.

This work was supported by University Muhammadiyah Malang under Grant PDK No.: E.2.a/132/BAA-UMM/IV/2020.

First Author and Second Author are with University of Muhammadiyah Malang, Department of Electrical Engineering, Indonesia (e-mail: widianto@umm.ac.id and chasrun@umm.ac.id)

Third Author is with Wroclaw University of Science and Technology, Poland (e-mail: Robert.lis@pwr.edu)

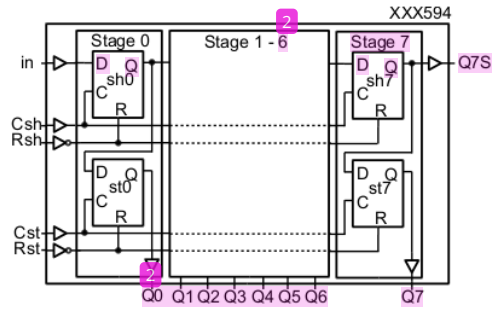


Fig.2. Logic diagram of the DUT

TABLE I
SYMBOL DESCRIPTION OF THE DUT

Symbol	Description
in	Serial data input
Csh	Shift register clock input
Rsh	Shift register reset input
Rst	Storage register reset input
Q0-to-Q7	Parallel data output
Q7S	Serial data output

Moreover, the DUT is designed using a SystemVerilog. The SystemVerilog code of the DUT is shown in Figure 3. The code is based on logic diagram as shown in Figure 2. There are eight stages, four inverters, and two inverters in the code.

```
module sh594b (in, Csh, Rsh, Cst, Rst, Q, Q7S, qsh);
input in, Csh, Rsh, Cst, Rst;
output [7:0] Q;
output Q7S;
output [7:0] qsh;

inva u3 (.g(rsh), .f(Rsh));
 …
stage u5 (.dsh(dsh), .csh(csh), .rsh(rsh), .cst(cst), .rst(rst),
.qsh(qsh[0]), .qst(Q0));
…
buff u13 (.d(Q7S), .c(qsh[7]));

assign Q = {Q7, Q6, Q5, Q4, Q3, Q2, Q1, Q0};

endmodule
```

Fig.3. SytemVerilog code of the DUT

A timing diagram of the DUT is shown in Figure 4. The DUT is reset by providing low level signals to the *Rsh* and the *Rst*. If the DUT is reset, all outputs of the Q7S and the *Q*0-to-*Q*7 are low level signals.

On the other hand, the DUT will run by providing high signals to the *Rsh* and the *Rst*. When a high pulse and clock signals are provided to the *in* and the *Csh*, respectively, the pulse signal will be shifted to each the stage and outputted serially to the Q7S since low-to-high transitions of the *Csh*. Moreover, clock signals are provided to the *Cst* behind one clock pulse of the *Csh*, the pulse signal will be shifted parallelly to the *Q*0-to-*Q*7 when low-to-high transitions of the *Cst*.

Stuck-at-faults may occur at interconnects between the stages inside the DUT. For example, Figure 5a shows a stuck-at-1 occurring between the stage 0 and the stage 1. The stuck-at-1 may cause an output of the *sh*0 in the stage 0 will stick at a high level signal regardless provided level signals of the *in*. On the other hand, occurring a stuck-at-0 is shown in Figure 5b. An output of the *sh*0 in the stage 0 will stick at a low level signal caused by the stuck-at-0.

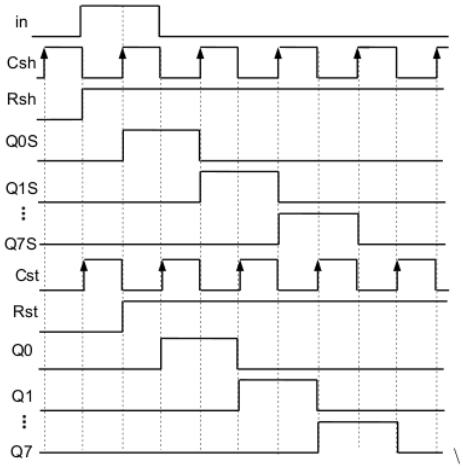
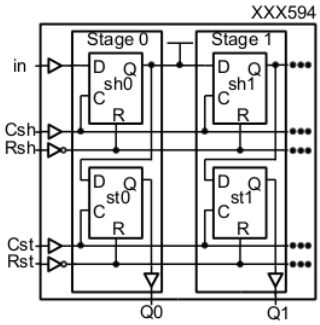
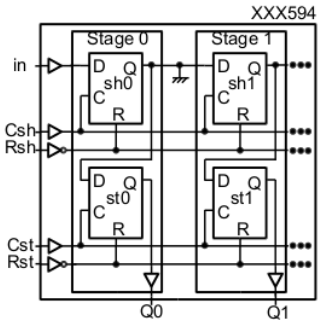


Fig.4. Timing Diagram of the DUT

Furthermore, all components in the testbench are coded by the SystemVerilog. There are classes and a module. The classes are the transaction object, the generator, the interface, the driver, the monitor, the scoreboard, the environment, and the test. However, the testbench top is the module.



(a)



(b)

Fig.5. Stuck-at-faults (a) Stuck-at-1 (b) Stuck-at-0

The transaction object is a base transaction object will be used to verify the DUT may work as the timing diagram shown in Figure 4. Moreover, the base transaction is used in the environment to initiate new transactions and capture transactions at the interface. The code of the transaction object is shown in Figure 6.

```
class Packet;
        bit in;
        bit Csh;
        bit Rsh;
        bit Cst;
        bit Rst;
        rand bit[7:0] Q;
        bit Q7S;
       rand bit[7:0] qsh;
…
endclass
```

Fig.6. Code of the transaction object

The generator is used to generate the random input stimulus signals to be sent to the driver. Figure 7 shows the generator code.

```
class generator;
        int    loop = 10;
        event drv_done;
       mailbox drv_mbx;
….
endclass
```

Fig.7. Code of the generator

Furthermore, the driver is to drive the stimulus signals to the interface and the scoreboard. The code of the driver is shown in Figure 8.

```
class driver;
        virtual switch_if m_switch_vif;
        virtual sh594b_if m_sh594b_vif;
        virtual clk_if  m_clk_vif;
        event drv_done;
       mailbox drv_mbx;
…
endclass
```

Fig.8. Code of the driver

Moreover, the interface contains the stimulus signals driven to the DUT and response signals derived from the DUT. All the signals are required by the DUT to be operated. Figure 9 shows the code of the interface.

```
interface sh594b_if();
        logic in;
        logic Csh;
        logic Rsh;
        logic Cst;
        logic Rst;
        logic[7:0] Q;
        logic Q7S;
        logic[7:0] qsh;
endinterface
```

Fig.9. Code of the driver

Then, the response signals will be monitored by the monitor and captured to the scoreboard. The code of the monitor is shown in Figure 10.

```
class monitor;
        virtual sh594b_if   m_sh594b_vif;
        virtual clk_if     m_clk_vif;
         mailbox scb_mbx;
…
endclass
```

Fig.10. Code of the monitor

Further, the scoreboard will check the response signals compared to expected signals. When the response signals are not match to the expected signals, the DUT is a faulty indicated by *error* statements. Otherwise, pass statements indicate the DUT is a fault-free. Therefore, the scoreboard may have a refence model behaving as the DUT. The reference model code of the scoreboard is shown in Figure 11.

```
class scoreboard;
…
if (!ref_item.Rsh) begin
        {ref_item.qsh} = 0;
    end else begin
        {ref_item.qsh} = {ref_item.qsh[6:0], ref_item.in};
      end
      if (!ref_item.Rst) begin
         {ref_item.Q} = 0;
    end else begin
       {ref_item.Q} = {ref_item.Q[6:0], ref_item.qsh[0]};
    end
…
endclass
```

Fig.11. Reference model of the scoreboard

The environment contains all the verification components, e.g., the generator, driver, interface, the monitor, and the scoreboard. Figure 12 shows the code of the environment.

```
class env;
        generator      g0;
        driver         d0;
        monitor          m0;
        scoreboard       0;
…
endclass
```

Fig.12. Code of the environment

The test contains the environment that can be tweaked with different configuration settings. Also, the test may instantiate any environment. Figure 13 is the code of the test.

```
class test;
        env e0;
       mailbox drv_mbx;
…
endclass
```

Fig.13. Code of the test

The testbench top contains the test. Moreover, the faults are inserted to it by *force* and *release* instructions. The code of the testbench top is shown in Figure 14.

```
module tb_sh594b3sa;
…
sh594b u0 (    .in(in),
               .Csh(Csh),
               .Rsh(Rsh),
               .Cst(Cst),
               .Rst (Rst),
```

```
                .Q(m_sh594b_if.Q),
                .Q7S(m_sh594b_if.Q7S),
               .qsh(qsh));
….
initial begin
test t0;
…
        #35 force qsh[0]=1;
       #4 release qsh[7];
….
endmodule
```

Fig.14. Code of the testbench top

III. RESULTS AND DISCUSSION

Building a testbench is proposed to verify a functionality of a shift register IC from stuck-at-faults. The testbench based on Figure 1 in which the register IC is as a DUT. The DUT is 8-bit shift register produced by Nexperia Co. Ltd in which it has a timing diagram as shown in Figures 4.

The testbench and the DUT are simulated using Questasim simulator. Simulation and transcript results of a fault-free DUT are shown in Figures 15 and 16, respectively. As shown in Figure 15, the result is an identic timing diagram of the DUT in Figure 4. Moreover, the scoreboard indicates *pass* statements verifying the DUT is a fault-free as shown in Figure 7.

In order to verify the DUT from the faults, stucks-at-1 are inserted each output *sh* of stage 0 to stage 7 when the output should be in low level signals. Figures 17 and 18 show simulation and transcript results of faulty DUT, respectively. As shown in Figure 18, error statements in the scoreboard indicate the DUT is a faulty.

```
…
# [11] Scoreboard Pass! Q match ref_item=0x2 item=0x2
# [11] Scoreboard Pass! Q7S match ref_item=0x0 item=0x0
…
# [31] Scoreboard Pass! Q match ref_item=0x40 item=0x40
# [31] Scoreboard Pass! Q7S match ref_item=0x0 item=0x0
…
# [51] Scoreboard Pass! Q match ref_item=0x0 item=0x0
# [51] Scoreboard Pass! Q7S match ref_item=0x0 item=0x0
…
# [71] Scoreboard Pass! Q match ref_item=0x0 item=0x0
# [71] Scoreboard Pass! Q7S match ref_item=0x0 item=0x0
….
# [91] Scoreboard Pass! Q match ref_item=0x0 item=0x0
# [91] Scoreboard Pass! Q7S match ref_item=0x0 item=0x0
```

Fig.16. Transcript result of fault-free

```
…
# [51] Scoreboard Error! Q mismatch ref_item=0x0 item=0x7
# [51] Scoreboard Pass! Q7S match ref_item=0x0 item=0x0
…
# [71] Scoreboard Error! Q mismatch ref_item=0x0 item=0xff
# [71] Scoreboard Pass! Q7S match ref_item=0x1 item=0x1
…
# [91] Scoreboard Error! Q mismatch ref_item=0x0 item=0xff
# [91] Scoreboard Pass! Q7S match ref_item=0x1 item=0x1
```

Fig.18. Transcript result of faulty DUT

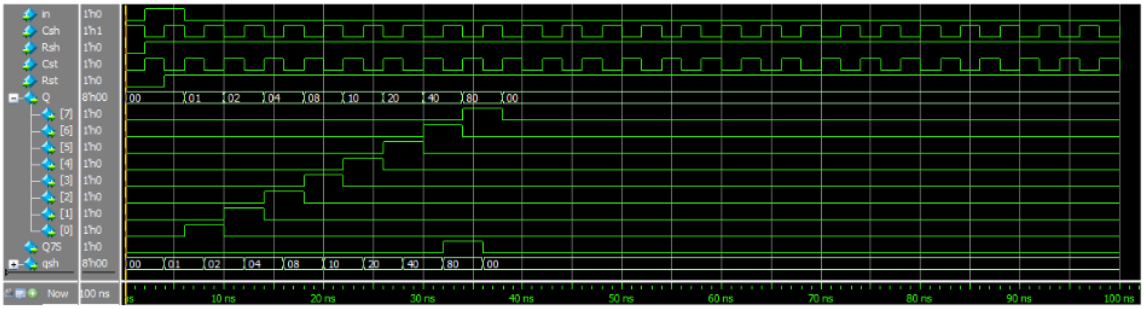


Fig.15. Simulation result of fault-free

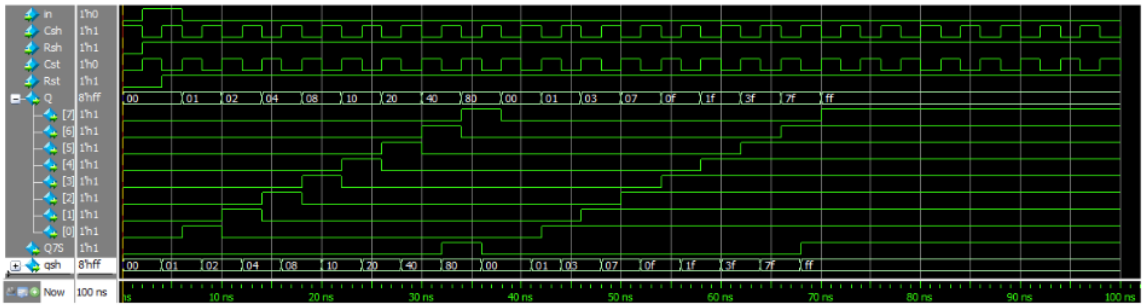


Fig.17. Simulation result of faulty DUT

CONCLUSION

A testbench is built to verify a functionality of a shift register IC by inserting stuck-at-faults inside it. The testbench and the IC are designed by SystemVerilog. Moreover, they are simulated by Questasim simulator. Simulation results show the faults may be detected by the testbench. Moreover, *error* statements will indicate the detected faults in transcript results of the simulator.

ACKNOWLEDGEMENTS

We would like to express gratitude to University of Muhammadiyah Malang for funding this work.

REFERENCES

[1] T. Ndjountche, Digital Electronics 2: Sequential and Arithmetic Logic Circuits. 2016.

[2] G. Nithya and M. Ramaswamy, "Very large scale integrated solution for stuck at faults in synchronous sequential circuits," J. Comput. Theor. Nanosci., vol. 16, no. 4, 2019, http://doi.org/10.1166/jctn.2019.8047

[3] A. A. Abou-Auf, M. M. Abdel-Aziz, M. A. Abdel-Aziz, and A. A. Ammar, "Fault Modeling and Worst Case Test Vector Generation for Flash-Based FPGAs Exposed to Total Dose," IEEE Trans. Nucl. Sci., vol. 64, no. 8, 2017, http://doi.org/10.1109/TNS.2017.2687982

[4] D. Addala, P. Teja, and S. Saxena, "Fault simulation algorithm for transistor single stuck short faults," in Intelligent Circuits and Systems, 2021.

[5] H. M. Gaur, A. K. Singh, and U. Ghanekar, "Design for Stuck-at Fault Testability in Toffoli–Fredkin Reversible Circuits," Natl. Acad. Sci. Lett., vol. 44, no. 3, 2021, http://doi.org/10.1007/s40009-020-00967-3

[6] P. Wang, A. M. Gharehbaghi, and M. Fujita, "An Automatic Test Pattern Generation Method for Multiple Stuck-At Faults by Incrementally Extending the Test Patterns," IEEE Trans. Comput. Des. Integr. Circuits Syst., vol. 39, no. 10, 2020, http://doi.org/10.1109/TCAD.2019.2957364

[7] P. Wang, A. M. Gharehbaghi, and M. Fujita, "An Incremental Automatic Test Pattern Generation Method for Multiple Stuck-at Faults," in Proceedings of the IEEE VLSI Test Symposium, 2019, vol. 2019-April, http://doi.org/10.1109/VTS.2019.8758668

[8] P. Wang, A. M. Gharehbaghi, and M. Fujita, "Automatic Test Pattern Generation for Double Stuck-at Faults Based on Test Patterns of Single Faults," in Proceedings - International Symposium on Quality Electronic Design, ISQED, 2019, vol. 2019-March,
http://doi.org/10.1109/ISQED.2019.8697831

[9] B. Alizadeh and S. R. Sharafinejad, "Incremental SAT-Based Accurate Auto-Correction of Sequential Circuits Through Automatic Test Pattern Generation," IEEE Trans. Comput. Des. Integr. Circuits Syst., vol. 38, no. 2, 2019, http://doi.org/10.1109/TCAD.2018.2812123

[10] Y. Ogasahara et al., "Implementation of pseudo-linear feedback shift register-based physical unclonable functions on silicon and sufficient Challenge–Response pair acquisition using Built-In Self-Test before shipping," Integration, vol. 71, 2020,
http://doi.org/10.1016/j.vlsi.2019.12.002

[11] V. Shivakumar, C. Senthilpari, and Z. Yusoff, "A Low-Power and Area-Efficient Design of a Weighted Pseudorandom Test-Pattern Generator for a Test-Per-Scan Built-in Self-Test Architecture," IEEE Access, vol. 9, 2021, http://doi.org/10.1109/ACCESS.2021.3059171

[12] M. Sharma and J. Dhanoa, "Smart Logic Built in Self-Test in SOC," 2020, http://doi.org/10.1109/ICRAIE51050.2020.9358296

[13] Widianto, "A SIGNATURE REGISTER OF A BIST TO DETECT STUCK-AT-FAULTS IN COMBINATIONAL LOGIC ICS," in SENTRA, 2020, pp. 39–44,
http://doi.org/https://doi.org/10.22219/sentra.v0i6.3811

[14] T. D. Prasad and B. R. Babu, "Design and Simulation of SPI Master / Slave Using Verilog HDL," Int. J. Sci. Res., vol. 3, no. 8, 2014.

[15] P. Flake, P. Moorby, S. Golson, A. Salz, and S. Davidmann, "Verilog HDL and its ancestors and descendants," Proc. ACM Program. Lang., vol. 4, no. HOPL, 2020, http://doi.org/10.1145/3386337

[16] M. Qiu, S. Yu, Y. Wen, J. Lü, J. He, and Z. Lin, "Design and FPGA Implementation of a Universal Chaotic Signal Generator Based on the Verilog HDL Fixed-Point Algorithm and State Machine Control," Int. J. Bifurc. Chaos, vol. 27, no. 3, 2017,
http://doi.org/10.1142/S0218127417500407

[17] M. W. Anwar, M. Rashid, F. Azam, and M. Kashif, "Model-based design verification for embedded systems through SVOCL: an OCL extension for SystemVerilog," Des. Autom. Embed. Syst., vol. 21, no. 1, 2017, http://doi.org/10.1007/s10617-017-9182-z

[18] K. K. Yadu and R. Bhakthavatchalu, "Block Level SoC Verification Using Systemverilog," 2019, http://doi.org/10.1109/ICECA.2019.8821909

[19] M. W. Anwar, M. Rashid, F. Azam, M. Kashif, and W. H. Butt, "A model-driven framework for design and verification of embedded systems through SystemVerilog," Des. Autom. Embed. Syst., vol. 23, no. 3–4, 2019, http://doi.org/10.1007/s10617-019-09229-y

[20] A. A. Vivekananda and E. Enoiu, "Automated test case generation for digital system designs: A mapping study on vhdl, verilog, and systemverilog description languages," Designs, vol. 4, no. 3, 2020, http://doi.org/10.3390/designs4030031

[21] "Design and Verification of UART using System Verilog," Int. J. Eng. Adv. Technol., vol. 9, no. 5, 2020,
http://doi.org/10.35940/ijeat.e1135.069520

[22] K. Benefits, "Industry's Highest Performance Simulation Solution," Synopsys, 2020.

[23] L. A. Kadlubowski and P. Kmon, "Test and Verification Environment and Methodology for Vernier Time-to-Digital Converter Pixel Array," 2021, http://doi.org/10.1109/DDECS52668.2021.9417054

[24] D. Ahlawat and N. Kr. Shukla, "Performance Analysis of Verilog Directed Testbench vs Constrained Random SystemVerilog Testbench," Int. J. Comput. Appl., vol. 118, no. 22, 2015, http://doi.org/10.5120/20874-3612

[25] B. Chinna Munaiah and S. M. Shamsheer Daula, "Design and verification of advanced high-performance bus lite protocol using questa sim," J. Adv. Res. Dyn. Control Syst., vol. 11, no. 9 Special Issue, 2019, http://doi.org/10.5373/JARDCS/V11/20192572

Build Testbenches for Verification in Shift Register ICs using SystemVerilog

ORIGINALITY REPORT

4%	4%	3%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	bibliotekanauki.pl Internet Source	2%
2	www.farnell.com Internet Source	1%
3	Submitted to Universitas Muhammadiyah Surakarta Student Paper	1%
4	Ali, F.M.. "Hardware-software co-synthesis of hard real-time systems with reconfigurable FPGAs", Computers and Electrical Engineering, 200410 Publication	1%

Exclude quotes	On	Exclude matches	< 1%
Exclude bibliography	On		

Digital Receipt

This receipt acknowledges that Turnitin received your paper. Below you will find the receipt information regarding your submission.

The first page of your submissions is displayed below.

Submission author:	Widianto Widianto
Assignment title:	Publication Articles April-Juni 2023
Submission title:	Build Testbenches for Verification in Shift Register ICs using ...
File name:	70-3488-Widianto-sk.pdf
File size:	511.92K
Page count:	5
Word count:	3,223
Character count:	16,487
Submission date:	19-May-2023 08:47AM (UTC+0700)
Submission ID:	2096689619

INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2022, VOL. 68, NO. 3, PP. 619-623
Manuscript received September 17, 2021; revised August, 2022. DOI: 10.24425/ijet.2022.141281

Build Testbenches for Verification in Shift Register ICs using SystemVerilog

Widianto, M. Chasrun H., and Robert Lis

***Abstract*—A testbench is built to verify a functionality of a shift register IC (Integrated Circuit) from stuck-at-faults, stuck-at-1 as well as stuck-at-0. The testbench is supported by components, i.e., generator, interface, driver, monitor, scoreboard, environment, test, and testbench top. The IC consists of sequential logic circuits of D-type flip-flops. The faults may occur at interconnects between the circuits inside the IC. In order to examine the functionality from the faults, both the testbench and the IC are designed using SystemVerilog and simulated using Questasim simulator. Simulation results show the faults may be detected by the testbench. Moreover, the detected faults may be indicated by error statements in transcript results of the simulator.**

***Keywords*—testbench; verification; shift register IC; stuck-at-faults; SystemVerilog**

I. Introduction

SHIFT register Integrated Circuits (ICs) may be applicated to serial-to-parallel data conversion and remote control holding register[1]. The ICs may be implemented to displays and control units. They consist of sequential logic circuits, i.e. D-type flip-flops. Stuck-at-faults may occur at either inputs or output interconnects between the circuits inside the IC[2].

The faults may be caused by shorted the interconnects to supply voltage (stuck-at-1) and ground (stuck-at-0)[3]–[5]. The inputs and outputs of the circuits may stick at high and low values caused by the stuck-at-1 and stuck-at-0, respectively. Since the faults may cause functions of the ICs become errors, they should be detected.

The faults may be detected by an Automatic Test Pattern Generation (ATPG)[6]–[9]. However, it needs to integrate the external tools and programs where integrating them is not easy. Furthermore, it requires a circuit partitioning and a bit parallel processing.

A Built-in Self Test (BIST) was proposed to detect the faults[10]–[12]. The BIST consisted by a signature analyzer[13]. The analyzer is used to compare signals generated by the faults and faults-free and further analyze them. However, it takes time consuming, since the signal faults are detected by checking the generated signals one-by-one. Furthermore, the BIST is still designed by a Verilog Hardware Description Languages (HDL) in which it isn't supported by Object Oriented Programming (OOP) features[14]–[16], thus the detected faults aren't indicated by pass/ error statements.

A SystemVerilog is an extension of the Verilog HDL[17]–[21] used to design and verify hardware[22][23], e.g., Field Programmable Gate Arrays (FPGAs) and Application-Specific Integrated Circuits (ASICs). The hardware design is created in a Register Transfer Level (RTL) model. Moreover, to verify functionality of the design working as expected, a testbench should be built[24]. The testbench drives different input stimulus to the design and is supported by the OOP features.

In this paper, a proposed shift register IC is designed using the SystemVerilog. It will be verified the functionality from both stuck-at-faults, stuck-at-1 and stuck-at-0 by a testbench. The designed IC and its testbench are simulated using Questasim simulator[25]. Simulation results denote that the faults inside the IC may be detected by the testbench. Furthermore, indicating pass/ error statements is included in the detected faults by it.

II. Research Method

A testbench is built to verify a functionality of a shift register IC from both stuck-at-faults, stuck-at-1 and stuck-at-0. The testbench architecture is shown in Figure 1. There are some supported components, namely, transaction object, generator, interface, driver, monitor, scoreboard, environment, test, and testbench top. The IC is as a Design Under Test (DUT).

Fig.1. Testbench architecture

The DUT is a sample 8-bit shift register IC produced by Nexperia Co. Ltd. A logic diagram of the DUT is shown in Figure 2. As shown in Figure 2, the DUT consists of four buffer gates, two inverter gates, and eight stages, stage 0 to stage 7. Each of the stages is made of a buffer gate and two D flip-flop types in which the D flip-flop types are a shift register *sh* and a storage register *st*. Symbol descriptions in the DUT are shown in Table I.

This work was supported by University Muhammadiyah Malang under Grant PDK No.: E.2.a/132/BAA-UMM/IV/2020.

First Author and Second Author are with University of Muhammadiyah Malang, Department of Electrical Engineering, Indonesia (e-mail: widianto@umm.ac.id and chasrun@umm.ac.id)

Third Author is with Wroclaw University of Science and Technology, Poland (e-mail: Robert.lis@pwr.edu)