Vol.11 No.3 S1, pISSN:2303-0577 eISSN:2830-7062 http://dx.doi.org/10.23960/jitet.v11i3%20s1.3527

CAKUPAN VERIFIKASI TESTBENCH DALAM MENDETEKSI KERUSAKAN HUBUNG SINGKAT DI RANGKAIAN TERPADU KOMPARATOR

Widianto[1*], M. Chasrun H.[2]

[1]Universitas Muhammadiyah Malang, Teknologi Elektronika; Jl. Raya Tlogomas 246 Malang, +62341 464318
[2]Universitas Muhammadiyah Malang, Teknik Elektro; Jl. Raya Tlogomas 246 Malang, +62341 464318

Riwayat artikel:
Received: 18 Agustus 2023
Accepted: 1 September 2023
Published: 11 September 2023

Keywords:
testbench;
functional coverage;
stuck-at-faults;
comparator IC.

Corespondent Email:
widianto@umm.ac.id

Abstrak. Dalam penelitian ini, *testbench* diajukan untuk mendeteksi kerusakan hubung singkat ke suplai tegangan dan ke *ground* yang terjadi di dalam rangkaian terpadu komparator dengan cakupan pendeteksian/ *coverage* di setiap titik rangkaian penyusun dalam rangkaian terpadu tersebut. *Testbench* tersusun dari beberapa komponen, yaitu: *transaction object*, *generator*, *interface*, *driver*, *monitor*, *scoreboard*, *environment*, *test*, dan *testbench top*. DUT (*Design Under Test*) atau desain yang akan diuji yaitu rangkaian terpadu komparator. Desain *testbench* dan DUT dirancang menggunakan Bahasa SystemVerilog dan diverifikasi menggunkan simulator Questasim 2021.1. Hasil verifikasi menunjukkan bahwa kerusakan hubung singkat yang terjadi di dalam DUT dapat dideteksi dengan keterangan *error* dan disertai dengan *coverage* 94.44%

Abstract. *In this paper, a testbench is proposed to detect stuck-at-faults in a comparator IC (Integrated Circuit). Moreover, the testbench utilized by a functional coverage in each input and output of the IC. The testbench composed by some components, i.e., transaction object, generator, interface, driver, monitor, scoreboard, environment, test, and testbench top. The IC is as a DUT (Design Under Test). Furthermore, the testbench and the DUT are designed by a SystemVerilog language and verified by a Questasim 2021.1 simulator. Verification results show the stuck-at-faults occurring in the DUT may be detected. The detected faults are indicated by error statements. Further, the coverage of the faults is 94.44%*

1. PENDAHULUAN

Kerusakan hubung singkat ke suplai tegangan dan ke *ground* bisa terjadi pada input atau output rangkaian penyusun di dalam rangkaian rangkaian terpadu[1]. Kerusakan hubung singkat ini bisa terjadi karena adanya cacat ketika proses produksi rangkaian terpadu. Kerusakan hubung singkat di dalam rangkaian terpadu bisa menyebabkan fungsi dari rangkaian terpadu tersebut tidak sesuai dengan spesifikasinya. Oleh sebab itu, kerusakan hubung hubung singkat yang terjadi di dalam rangkaian terpadu harus dideteksi sebelum rangkaian terpadu tersebut dikirimkan ke para konsumen[2].

Analisis kerusakan hubung singkat yang terjadi dalam rangkaian terpadu logika kombinasional bisa dideteksi menggunakan rangkaian penguji yang terbangun di dalam rangkaian terpadu logika kombinasional

tersebut atau BIST (*built-in self test*). Rangkaian penguji terdiri LFSR (*liner feedback shift register*) dan multiplekeser [3]. Multiplekser digunakan oleh rangkaian penguji untuk memilih bekerja pada mode normal atau mode pengujian. Pada mode normal, sinyal yang akan diolah oleh rangkaian kombinasional berdasarkan sinyal yang masuk ke inputan rangkaian kombinasional tersebut. Sedangkan pada mode pengujian, maka LFSR berfungsi untuk memberikan stimulus sinyal ke input rangkaian logika kombinasional tersebut. Pengujian ini memerlukan waktu yang lama karena respon dari rangkaian logika kombinsional akan dianalisis satu persatu dari kerusakan hubung singkat.

BIST yang dilengkapi dengan rangkaian SA (*signature analyzer*) yang dirancang menggunakan Bahasa Verilog telah diajukan untuk mendeteksi kerusakan hubung singkat di dalam rangkaian terpadu logika kombinasional[4]. Setiap output dari rangkaian logika kombinasional memerlukan satu rangkaian SA. Tentunya ini akan memerlukan tempat yang lebih luas di dalam rangkaian terpadu untuk mendesain BIST yang dilengkapi dengan SA.

BIST yang dilengkapi dengan rangkaian SR (*signature register*) diajukan untuk mengatasi masalah tempat yang dihabiskan di dalam rangkaian terpadu logika kombinasional[5]. Semua output dari rangkaian logika kombinasional hanya membutuhkan satu rangkaian SR. Respon dari rangkaian SR akan membutuhkan pencermatan ketika terjadi kerusakan hubung singkat, karena belum dilengkapi dengan parameter-parameter pendeteksian kerusakan hubung singkat.

Desain *testbench* telah diajukan untuk mendeteksi kerusakan hubung singkat ke suplai tegangan dan ke *ground* di dalam rangkaian terpadu register geser[6]. Desain *tesbench* dan rangkaian terpadu register geser disusun menggunakan Bahasa SystemVerilog [7] dan disimulasikan menggunkan Questasim 10.7c. Bahasa SystemVerilog memberikan fitur parameter-parameter yang dapat memudahkan dalam mendeteksi kerusakan hubung singkat, yaitu parameter *pass* dan *error*. Parameter *pass* akan muncul di Questasim ketika tidak ada kerusakan hubung singkat di dalam rangkaian terpadu register geser. Ketika terjadi kerusakan hubung singkat di dalam rangkaian terpadu register, maka parameter *error* akan muncul di Questasim. Tetapi desain *testbench* belum dilengkapi dengan cakupan pendeteksian kerusakan hubung singkat pada setiap input dan output rangkaian penyusun dalam rangkaian terpadu tersebut.

Oleh sebab itu dalam penelitian ini, *testbench* diajukan untuk memverifikasi rangkaian terpadu komparator dari kerusakan hubung singkat ke suplai tegangan dan ke *ground* yang terjadi pada input atau output rangkaian penyusun di dalam rangkaian terpadu tersebut. *Tesbench* yang diajukan dapat mendeteksi kerusakan hubung singkat dengan cakupan pendeteksian kerusakan hubung singkat yang terjadi pada setiap input dan output dari rangkaian penyusun. Sehingga *testbench* yang diajukan dapat dimanfaatkan sebagai kontrol kualitas terhadap rangkaian terpadu komparator dari kerusakan hubung singkat.

Testbench tersusun dari beberapa komponen, yaitu: *transaction object*, *generator*, *interface*, *driver*, *monitor*, *scoreboard*, *environment*, *test*, dan *testbench top*. Sedangkan DUT (*Design Under Test*) atau desain yang akan diuji yaitu rangkaian terpadu komparator.

Transaction object merupakan sinyal kontrol input dan output yang akan digerakkan ke DUT. *Generator* berfungsi untuk menghasilkan berbagai sinyal stimulus input untuk digerakkan ke DUT. Sinyal stimulus yang dihasilkan oleh *generator* akan digerakkan ke DUT oleh *driver*. *Interface* berfungsi untuk menggerakkan sinyal dari generator yang masuk ke DUT atau untuk mengamati sinyal yang keluar dari DUT. *Monitor* akan mengamati sinyal *port* input atau output dari *interface* untuk ditangkap atau disimpan. *Scoreboard* berfungsi untuk memeriksa apakah sinyal output yang dihasilkan oleh DUT yang tertangkap di *monitor* sudah sesuai dengan yang diharapkan. *Environment* berisi komponen *generator*, *driver*, *monitor*, dan *scoreboard*. *Test* berisi environment yang dapat diatur konfigurasinya. *Testbench top* berisi test dan juga DUT.

Desain *tesbench* yang diajukan dan rangkaian terpadu komparator akan disusun menggunakan Bahasa SystemVerilog dan disimulasikan menggunkan Questasim 2021.1[8]. Bahasa SystemVerilog memberikan fitur parameter-parameter yang dapat memudahkan dalam mendeteksi kerusakan

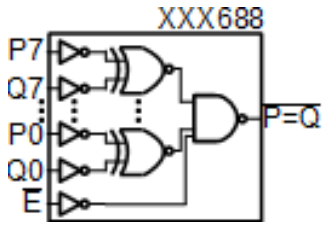
hubung singkat, misalkan yaitu parameter *pass* dan *error*. Parameter *pass* akan muncul di Questasim ketika *testbench* dengan rangkaian terpadu komparator tidak memiliki kerusakan hubung singkat di dalamnya. Ketika *testbench* dengan rangkaian terpadu komparator memiliki kerusakan hubung singkat di dalamnya, maka parameter *error* akan muncul di Questasim. Sehingga ke depan, desain testbench dan rangkaian terpadu komparator yang disusun menggunakan Bahasa SystemVerilog tidak hanya terbatas pada simulasi di Questasim, tetapi dapat diimplementassikan ke dalam *hardware*, misalnya FPGA[9][10].

Dengan penelitian ini diharapkan desain *testbench* yang diajukan dapat mendeteksi dengan cepat kerusakan hubung singkat yang terjadi di dalam rangkaian terpadu komparator dengan cakupan pendeteksian kerusakan hubung singkat pada setiap input dan output rangkaian penyusun dalam rangkaian terpadu tersebut

2. TINJAUAN PUSTAKA

2.1. Rangkaian Terpadu Komparator

Rangkaian terpadu komparator dapat ditemukan dalam aplikasi di kontrol motor-servo dan kontroler proses[11]. Contoh rangkaian komparator yang tersusun dari gerbang logika INVERTER, XNOR, dan NAND bisa dilihat dalam Gambar 1. Deskripsi pin dari rangkaian komparator bisa dilihat dalam Tabel 1.



Gambar 1 Rangkaian komparator 8 bit
Tabel 1 Pin Rangakain Komparator

Simbol	Deskripsi
E	enable
P0 – P7	Input P
Q0 – Q7	Input Q
P=Q	Output

Rangkaian komparator dapat digunakan dalam membandingkan data. Untuk membandingkan data, rangakaian komparator bekerja berdasarkan deskripsi fungsi yang ditunjukkan dalam Tabel 2.

Tabel 2 Fungsi rangkaian komparator

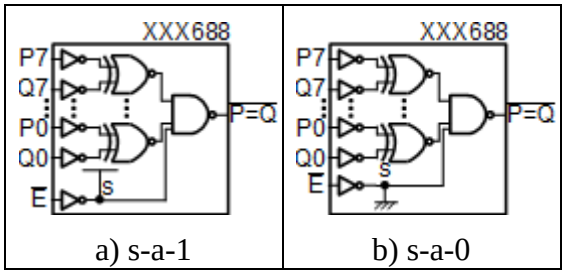
Input		Output
Data Pn dan Qn	**E**	**P=Q**
P=Q	0	0
P>Q	0	1
P<Q	0	1
X	1	1

Keterangan:
X = *don't care* atau tidak dipedulikan nilainya

2.2. Kerusakan Hubung Singkat

Kerusakan hubung singkat ke suplai tegangan atau stuck-at-1 (s-a-1) menyebabkan nilai logika pada input atau output rangkaian bernilai tinggi atau 1[12]. Contoh hubung singkat s-a-1 yang terjadi pada output rangkaian penyusun rangkaian terpadu komparator ditunjukkan dalam Gambar 2a. Sebagaimana ditunjukkan dalam Gambar 2a, nilai logika pada output logika INVERTER *s* akan bernilai 1 apapun nilai logika dari inputan *E*.

Sedangkan nilai logika input atau output rangkaian akan bernilai rendah atau 0 ketika terjadi hubung singkat ke ground atau stuck-at-0 (s-a-0) pada input atau output rangkaian[12]. Contoh hubung singkat s-a-0 yang terjadi pada rangkaian penyusun rangkaian terpadu komparator bisa ditunjukkan dalam Gambar 2b. Dalam Gambar 2b, tanpa memperdulikan nilai logika dari inputan *E*, maka nilai logika pada output logika INVERTER s akan bernilai 0.



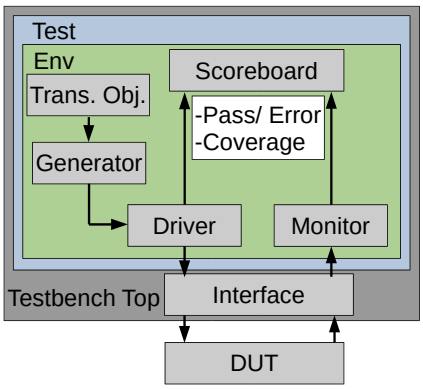
a) s-a-1 b) s-a-0

Gambar 2 Contoh hubung singkat s-a-1 dan s-a-0

3. METODE PENELITIAN

Desain *testbench* yang diajukan untuk menguji kerusakan hubung singkat yang terjadi di dalam rangkaian terpadu komparator bisa dilihat dalam Gambar 3. *Testbench* tersusun dari beberapa komponen, yaitu: *transaction object*, *generator*, *interface*, *driver*, *monitor*, *scoreboard*, *environment*, *test, and testbench top*. Sedangkan DUT (*Design Under Test*) merupakan desain yang akan diuji, dalam hal ini adalah rangkaian terpadu komparator.

Testbench dan DUT didesain menggunakan Bahasa SystemVerilog. Ada dua kontruksi yang digunakan, yaitu *module* dan *class*. Module digunakan pada komponen *transaction object*, *generator*, *interface*, *driver*, *monitor*, *scoreboard*, *environment*, dan *test*. Sedangkan komponen *testbench top* dan DUT menggunakan *module.*



Gambar 3 Desain *testbench*

Diagram logika DUT sebagaimana ditunjukkan dalam Gambar 1. Kode DUT dalam Bahasa SystemVerilog bisa ditunjukkan dalam Gambar 4.

```
module mc688 (E, Q0, Q1, Q2, Q3, Q4, Q5,
Q6, Q7, P0, P1, P2, P3, P4, P5, P6, P7, Q);
//74HC688 8-bit magnitude comparator
input
E,Q0,Q1,Q2,Q3,Q4,Q5,Q6,Q7,P0,P1,P2,P3,
P4,P5,P6,P7;
    logic x[7:0];
    …
    output Q;
    inva u1 (.g(z), .f(E));
    …
    inva u17 (.g(y[7]), .f(P7));
    XNOR u18 (.c(c[0]), .a(y[0]), .b(x[0]));
    …
    XNOR u25 (.c(c[7]), .a(y[7]), .b(x[7]));
    NAND9I u26
(.j(Q),.a(c[7]),.b(c[6]),.c(c[5]),.d(c[4]),.e(c[3
]),.f(c[2]),.g(c[1]),.h(c[0]),.i(z));
    endmodule
```

Gambar 4 Kode DUT

Transaction object merupakan objek transaksi dasar yang digunakan di komponen *Environment* untuk memulai transaksi baru dan menangkap transaksi dari DUT melalui komponen *Interface*. Kode *transaction object* bisa ditunjukkan dalam Gambar 5.

```
class Packet;
  bit E;
  …
  bit Q;
   ….
endclass
```

Gambar 5 Kode *Transaction Object*

Generator berfungsi untuk menghasilkan berbagai sinyal stimulus input untuk digerakkan ke DUT. Kode *generator* bisa dilihat dalam Gambar 6. *Generator* tersusun dari beberapa konstruksi, yaitu: *class*, *int*, *event*, *mailbox*, *task*, *for*.

```
class generator;
int      loop = 10;
event drv_done;
mailbox drv_mbx;
task run();
for (int i = 0; i < loop; i++) begin
…
end
endtask
```

```
endclass
```

Gambar 6 Kode *generator*

Sinyal stimulus yang dihasilkan oleh *generator* akan digerakkan ke DUT oleh *driver*. Kode *driver* bisa dilihat dalam Gambar 7. *Driver* tersusun dari beberapa konstruksi, yaitu: *class, virtual, event, mailbox, task*.

```
class driver;
  virtual mc688_if m_mc688_vif;
  virtual clk_if  m_clk_vif;
  event drv_done;
  mailbox drv_mbx;
    task run();
    ….
  endtask
endclass
```

Gambar 7 Kode *driver*

Interface berfungsi untuk menggerakkan sinyal dari *generator* yang masuk ke DUT atau untuk mengamati sinyal yang keluar dari DUT. Kode *interface* bisa dilihat dalam Gambar 8. *Interface* tersusun dari beberapa konstruksi, yaitu: *interface*, *logic*.

```
interface mc688_if();
  logic E;
  …
  logic Q;
endinterface
```

Gambar 8 Kode *interface*

Monitor akan mengamati sinyal port input atau output dari *interface* untuk ditangkap atau disimpan. Contoh *pseudocode monitor* bisa dilihat dalam Gambar 9. *Monitor* tersusun dari beberapa konstruksi, yaitu: *class, virtual, mailbox, task*.

```
class monitor;
  virtual mc688_if          m_mc688_vif;
  virtual clk_if    m_clk_vif;
  mailbox scb_mbx;

  task run();
   ….
  endtask
endclass
```

Gambar 9 Kode *interface*

Scoreboard berfungsi untuk memeriksa apakah sinyal output yang dihasilkan oleh DUT yang tertangkap di *monitor* sudah sesuai dengan fungsi rangkaian komparator sebagaimana ditunjukkan dalam Tabel 2. *Pseudocode scoreboard* bisa dilihat dalam Gambar 10. *Scoreboard* tersusun dari beberapa konstruksi, yaitu: *class, mailbox, task, forever*.

```
class scoreboard;
mailbox scb_mbx;
  task run();
  forever begin
  ….
     if (ref_item.E) begin
          {ref_item.Q} = 1;
        end else begin
      {ref_item.Q} = 0;
        end
    if (ref_item.Q!= item.Q)
   $display("[%0t] Scoreboard Error! Q
mismatch ref_item=0x%0h item=0x%0h",
$time, ref_item.Q, item.Q);
     else
     $display("[%0t] Scoreboard Pass! Q
match ref_item=0x%0h item=0x%0h",
$time, ref_item.Q, item.Q);

   end
  endtask
endclass
```

Gambar 10 Kode *interface*

Environment berisi komponen *generator*, *driver*, *monitor*, dan *scoreboard*. Contoh *pseudocode environment* bisa dilihat dalam Gambar 11. *Environment* tersusun dari beberapa konstruksi, yaitu: *class, generator, driver, monitor, scoreboard, mailbox, virtual, function.*

```
class env;
  generator              g0;
  driver                 d0;
  monitor                m0;
  scoreboard             s0;
  mailbox                scb_mbx;
  virtual mc688_if       m_mc688_vif;
  virtual clk_if   m_clk_vif;
  event drv_done;
  mailbox drv_mbx;
  function new();
```

```
  …
 endfunction
 virtual task run();
  …
 endtask
endclass
```

Gambar 10 Kode *environment*

Test berisi *environment* yang dapat diatur konfigurasinya. Kode *test* bisa dilihat dalam Gambar 11. *Test* tersusun dari beberapa konstruksi, yaitu: *class, env, mailbox, function, virtual, task*.

```
class test;
 env e0;
 mailbox drv_mbx;
 function new();
  …
 endfunction
 virtual task run();
  ….
 endtask
endclass
```

Gambar 11 Kode *test*

Testbench top digunakan untuk menguji DUT dengan cakupan pendeteksian yang terjadi pada setiap input dan output dari rangkaian penyusun. Kode *testbench top* bisa dilihat dalam Gambar 12. *Testbench top* tersusun dari beberapa konstruksi, yaitu: *module, bit, test, covergroup,* dan *coverpoint.*

```
module tb_mc688bsa;
…
mc688
u0(.E(E),.Q0(Q0),.Q1(Q1),.Q2(Q2),.Q3(Q3)
,.Q4(Q4),.Q5(Q5),
.Q6(Q6),.Q7(Q7),.P0(P0),.P1(P1),.P2(P2),.P
3(P3),.P4(P4),
.P5(P5),.P6(P6),.P7(P7),.Q(m_mc688_if.Q))
;
..
 covergroup cg @(posedge tb_clk);
  p_E :coverpoint E;
  cp_Q0 :coverpoint Q0;
  …
  cp_Q7 :coverpoint Q7;
  cp_P0 :coverpoint P0;
  …
  cp_P7 :coverpoint P7;
  cp_Q :coverpoint Q;
  endgroup
  ….
   #35 force  Q0=1;
   #15 release Q0;
   …
   #50 $display ("Coverage = %0.2f %%",
cg_inst.get_inst_coverage());
  end
endmodule
```

Gambar 12 Kode *testbench top*

4. HASIL DAN PEMBAHASAN

Kode *testbench* dan DUT dalam Bahasa SystemVerilog kemudian disimulasikan menggunakan software Questasim 2021.1. Hasil simulasi DUT tanpa kerusakan hubung singkat bisa ditunjukkan dalam Gambar 13. Hasil simulasi menunjukkan kesesuaian dengan fungsi rangkaian komparator sebagaimana ditunjukkan dalam Tabel 2. Hasil *transcript* dari Questasim juga memberikan keterangan *Scoreboard Pass* dan *Coverage* sebagaimana ditunjukkan dalam Gambar 14.

```
…
# Coverage = 94.44 %
…
# T=52 E=0 Q0=0 Q1=0 Q2=0 Q3=0 Q4=0
Q5=0 Q6=0 Q7=0 P0=0 P1=0 P2=0 P3=0
P4=0 P5=0 P6=0 P7=0 Q=0
# [52] Scoreboard Pass! Q match
ref_item=0x0 item=0x0
…
```

Gambar 14 Hasil *transcript* DUT tanpa kerusakan hubung singkat

Sedangkan hasil simulasi DUT dengan kerusakan hubung singkat ditunjukkan dalam Gambar 15. Hasil simulasi menujukkan ketidaksesuaian dengan fungsi rangkaian komparator dalam Tabel 2. Keterangan *Scoreboard Error* dan *Coverage* juga muncul dalam hasil *transcript* dari Questasim sebagaimana ditunjukkan dalam Gambar 16.

```
# Coverage = 94.44 %
…
```

```
# T=52 E=0 Q0=0 Q1=0 Q2=0 Q3=0 Q4=0
Q5=0 Q6=0 Q7=0 P0=0 P1=0 P2=0 P3=0
P4=0 P5=0 P6=0 P7=0 Q=1
# [52] Scoreboard Error! Q mismatch
ref_item=0x0 item=0x1
…
```

Gambar 16 Hasil *transcript* DUT dengankerusakan hubung singkat

Prosentase *Coverage* yang muncul dalam *transcript* dari Questasim bisa didapatkan dari *sim* sebagaimana ditunjukkan dalam Gambar 17. Prosentase *coverage* juga bisa dihitung menggunakan Persamaan 1.

$$Coverage\ (\%) = \frac{\sum_{i=1}^{n} xi}{n} \qquad (1)$$

Keterangan

n = banyaknya *Coverpoint*

xi = Prosentase *Coverpoint* ke i

5. KESIMPULAN

a. *Testbench* diajukan untuk memverifikasi rangkaian terpadu komparator/ DUT (*Design Under Test*) dari kerusakan hubung singkat ke suplai tegangan dan ke *ground* yang terjadi pada input atau output rangkaian penyusun di dalam rangkaian terpadu tersebut. *Testbench* yang diajukan juga dilengkapi cakupan pendeteksian/ *coverage* kerusakan hubung singkat pada setiap input dan output rangkaian penyusun dalam rangkaian terpadu tersebut

b. *Testbench* dan DUT didesain menggunakan SystemVerilog dan diverifikasi menggunakan simulator Questasim.

Hasil verifikasi menunjukkan bahwa kerusakan hubung singkat yang terjadi di dalam DUT dapat dideteksi dengan keterangan *error* dan disertai dengan *coverage* 94.44%

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Universitas Muhammadiyah Malang yang telah mendanai penelitian ini dengan skema Penelitian Dasar Keilmuan (PDK) nomor: E.2.a/334/BAA-UMM/IV/2022

DAFTAR PUSTAKA

[1] F. L. Aguirre, S. M. Pazos, F. Palumbo, A. Morell, J. Suñé, and E. Miranda, "Assessment and improvement of the pattern recognition performance of memdiode-based cross-point arrays with randomly distributed stuck-at-faults," *Electron.*, vol. 10, no. 19, 2021, doi: 10.3390/electronics10192427.

[2] V. I. Enns, S. V. Gavrilov, and R. Z. Chochaev, "Automatic FPGA Placement Configuration for Customer Designs," *Russ. Microelectron.*, vol. 51, no. 7, 2022, doi: 10.1134/S1063739722070046.

[3] A. Menbari and H. Jahanirad, "A Tunable Concurrent BIST Design Based on Reconfigurable LFSR," *J. Electron. Test. Theory Appl.*, 2023, doi: 10.1007/s10836-023-06055-w.

[4] R. Rotar and S. L. Jurj, "Configurable Built-In Self-Test Architecture for Automated Testing of a Dual-Axis Solar Tracker," in *21st IEEE International Conference on Environment and Electrical Engineering and 2021 5th IEEE Industrial and Commercial Power System Europe, EEEIC / I and CPS Europe 2021 - Proceedings*, 2021. doi: 10.1109/EEEIC/ICPSEurope51590.2021.9584768.

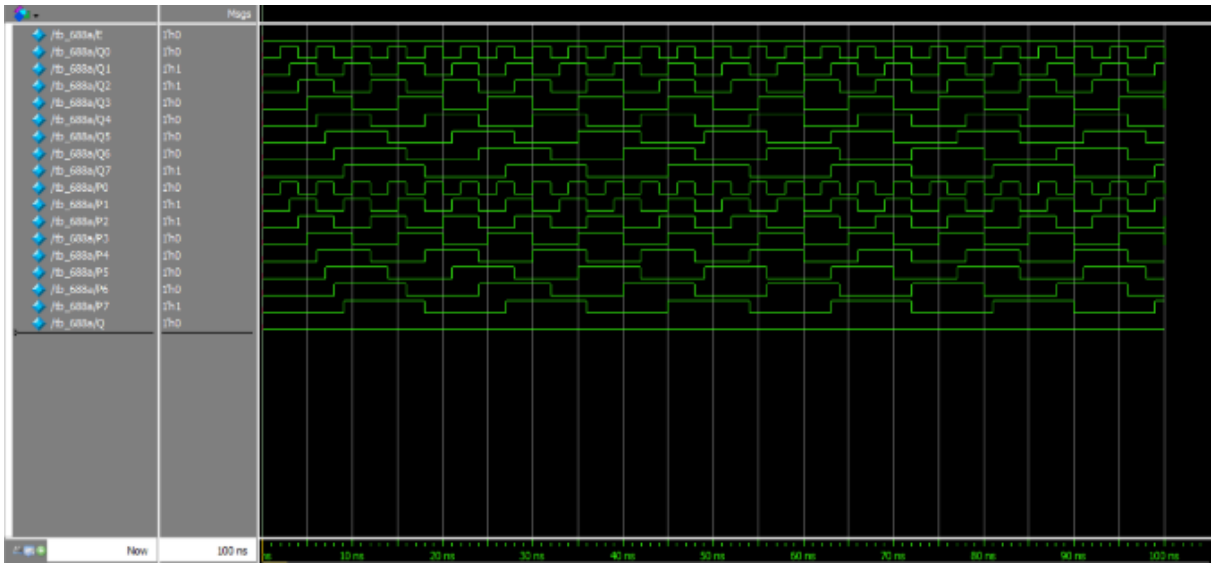
[5] S. Demidenko, M. T. Chew, B. T. Nguyen, M. P. L. Ooi, and Y. C. Kuang, "Logic built-in self-test instrumentation system for engineering test technology education," in *I2MTC 2020 - International Instrumentation and Measurement Technology Conference, Proceedings*, 2020. doi: 10.1109/I2MTC43012.2020.9128856.

[6] M. Widianto, H. Chasrun, and R. Lis, "Build Testbenches for Verification in Shift Register ICs using SystemVerilog," *Int. J. Electron. Telecommun.*, vol. 68, no. 3, 2022, doi: 10.24425/ijet.2022.141281.

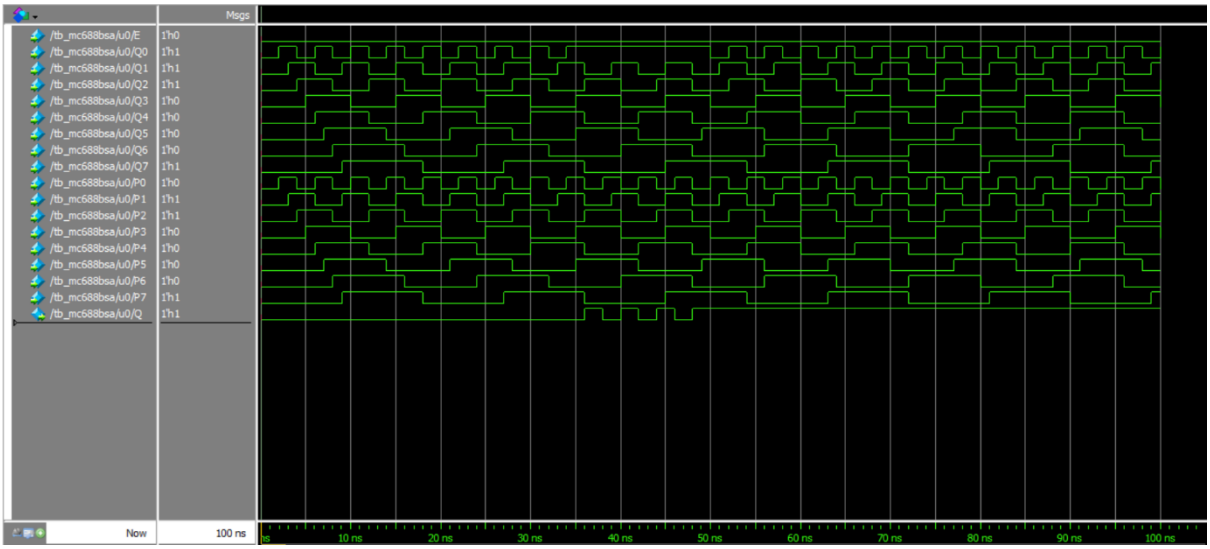
[7] M. Tetteh, D. M. Dias, and C. Ryan, "Grammatical Evolution of Complex Digital Circuits in SystemVerilog," *SN Comput. Sci.*, vol. 3, no. 3, 2022, doi: 10.1007/s42979-022-01045-9.

[8] S. Niharika and S. Chandrahas, "Boosting chip verification efficiency: UVM-based adder verification with QuestaSim," *i-manager's J. Digit. Signal Process.*, vol. 11, no. 1, 2023, doi: 10.26634/jdp.11.1.19768.

[9] J. Deschamps, C. Kieser, P. Hoess, T. Deguchi, and J. Ries, "MicroFPGA: An affordable FPGA platform for microscope control," *HardwareX*, vol. 13, 2023, doi: 10.1016/j.ohx.2023.e00407.
[10] S. Herbst, G. Rutsch, W. Ecker, and M. Horowitz, "An Open-Source Framework for FPGA Emulation of Analog/Mixed-Signal Integrated Circuit Designs," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 41, no. 7, 2022, doi: 10.1109/TCAD.2021.3102516.
[11] C. B. B. Laue *et al.*, "Logic Application Handbook Product Features and Application Insights," *Nexperia*, 2020.
[12] H. Shin, M. Kang, and L. S. Kim, "Fault-Free: A Framework for Analysis and Mitigation of Stuck-at-Fault on Realistic ReRAM-Based DNN Accelerators," *IEEE Trans. Comput.*, vol. 72, no. 7, 2023, doi: 10.1109/TC.2022.3227871.



Gambar 13 Hasil *simulasi* DUT tanpa kerusakan hubung singkat



Gambar 14 Hasil *simulasi* DUT dengan kerusakan hubung singkat

Instance	Design unit	Design unit type	Top Category	Visibility	Total coverage	Covergroup %
tb_mc688bsa	tb_mc688bsa(fast)	Module	DU Instance	+acc=...	94.44%	94.44%
cg	tb_mc688bsa(fast)	SVCovergroup	-	+acc=...	94.44%	94.44%
p_E	tb_mc688bsa(fast)	SVCoverpoint	-	+acc=...	50.00%	50.00%
cp_Q0	tb_mc688bsa(fast)	SVCoverpoint	-	+acc=...	100.00%	100.00%
cp_Q1	tb_mc688bsa(fast)	SVCoverpoint	-	+acc=...	100.00%	100.00%
cp_Q2	tb_mc688bsa(fast)	SVCoverpoint	-	+acc=...	100.00%	100.00%
cp_Q3	tb_mc688bsa(fast)	SVCoverpoint	-	+acc=...	100.00%	100.00%
cp_Q4	tb_mc688bsa(fast)	SVCoverpoint	-	+acc=...	100.00%	100.00%
cp_Q5	tb_mc688bsa(fast)	SVCoverpoint	-	+acc=...	100.00%	100.00%
cp_Q6	tb_mc688bsa(fast)	SVCoverpoint	-	+acc=...	100.00%	100.00%
cp_Q7	tb_mc688bsa(fast)	SVCoverpoint	-	+acc=...	100.00%	100.00%
cp_P0	tb_mc688bsa(fast)	SVCoverpoint	-	+acc=...	100.00%	100.00%
cp_P1	tb_mc688bsa(fast)	SVCoverpoint	-	+acc=...	100.00%	100.00%
cp_P2	tb_mc688bsa(fast)	SVCoverpoint	-	+acc=...	100.00%	100.00%
cp_P3	tb_mc688bsa(fast)	SVCoverpoint	-	+acc=...	100.00%	100.00%
cp_P4	tb_mc688bsa(fast)	SVCoverpoint	-	+acc=...	100.00%	100.00%
cp_P5	tb_mc688bsa(fast)	SVCoverpoint	-	+acc=...	100.00%	100.00%
cp_P6	tb_mc688bsa(fast)	SVCoverpoint	-	+acc=...	100.00%	100.00%
cp_P7	tb_mc688bsa(fast)	SVCoverpoint	-	+acc=...	100.00%	100.00%
cp_Q	tb_mc688bsa(fast)	SVCoverpoint	-	+acc=...	50.00%	50.00%

Gambar 17 Prosentase *Coverage* di s*im*