Testbench untuk Mendeteksi Kerusakan Hubung Singkat di dalam Rangkaian Terpadu Register Geser yang Didesain Menggunakan SystemVerilog

by Widianto Widianto

Submission date: 05-Jul-2023 10:23AM (UTC+0700)
Submission ID: 2126645257
File name: WIDIANTO_ELKTRO_UMM_2022.docx (290K)
Word count: 3774
Character count: 23226

Testbench untuk Mendeteksi Kerusakan Hubung Singkat di dalam Rangkaian Terpadu Register Geser yang Didesain Menggunakan SystemVerilog

PENGUSUL

Widianto, ST., MT

TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS MUHAMMADIYAH MALANG

2022

Kata Pengantar

Alhamdulillah, peneliti mengucapkan syukur kehadhirat Alloh Subhanahu Wa Ta'ala atas limpahan nikmat sehat dhahir dan bathin karena telah menyelesaikan penelitian dengan judul "*testbench* untuk mendeteksi kerusakan hubung singkat di dalam rangkaian terpadu register geser yang didesain menggunakan SystemVerilog".

Kerusakan hubung singkat ke suplai tegangan dan ke *ground* bisa terjadi pada input atau output rangkaian penyusun di dalam rangkaian rangkaian terpadu register geser. Maka kerusakan hubung hubung singkat yang terjadi di dalam rangkaian terpadu register geser harus dideteksi sebelum rangkaian terpadu tersebut dikirimkan ke para konsumen.

Desain *testbench* diajukan untuk mendeteksi kerusakan hubung singkat ke suplai tegangan dan ke ground di dalam rangkaian terpadu register geser. Desain *tesbench* dan rangkaian terpadu register geser disusun menggunakan Bahasa SystemVerilog dan disimulasikan menggunkan Questasim 10.7c. 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 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. Dengan penelitian ini diharapkan ketika terjadi kerusakan hubung singkat di dalam rangkaian terpadu register geser segera dengan cepat dapat dideteksi menggunkan desain *testbench* yang diajukan. Oleh sebab itu, peneliti berharap penelitian ini dapat diakui sebagai hak cipta sebagai bagian dari Hak Kekayaan Intelektual (HKI).

Malang, Februari 2022
Peneliti

Widianto, ST., MT

Daftar Isi

Kata Pengantar 2
Daftar Isi 3
Daftar Gambar 4
Daftar Tabel 5
Desain *Testbench* 6
DUT 6
Kerusakan Hubung Singkat di DUT 7
SystemVerilog 8
Kode SystemVerilog 9
Transaction Object 10
Generator 11
Driver 12
Interface 13
Monitor 14
Scoreboard 15
Environment 15
Test 17
Testbench Top 17
Kode SystemVerilog dengan kerusakan hubung singkat di DUT 19
Questasim 10.7c 23
Hasil Simulasi dan Pembahasan 23
Kesimpulan 27
Daftar Pustka 28

Daftar Gambar

Gambar 1 Desain *testbench* 6

Gambar 2 Rangkaian register geser 8-bit.....7

Gambar 3 Diagram waktu rangkaian register geser 8-bit 7

Gambar 4 Contoh hubung singkat s-a-1 dan s-a-0.....8

Gambar 5 Kode DUT.....9

Gambar 6 Kode *Transaction Object* 10

Gambar 7 Kode *generator* 11

Gambar 8 Kode *driver* 12

Gambar 9 Kode *interface* 13

Gambar 10 Kode *monitor* 14

Gambar 11 Kode *scoreboard* 15

Gambar 12 Kode *environment* 16

Gambar 13 Kode *test* 17

Gambar 14 Kode *testbench top* 19

Gambar 15 Kode *scoreboard* ketika DUT dengan kerusakan hubung singkat 20

Gambar 16 Kode *testbench top* ketika DUT dengan kerusakan hubung singkat 22

Gambar 17 Hasil simulasi DUT tanpa kerusakan hubung singkat 23

Gambar 18 Hasil *transcipt* DUT tanpa kerusakan hubung singkat 25

Gambar 19 Hasil simulasi DUT tanpa kerusakan hubung singkat 25

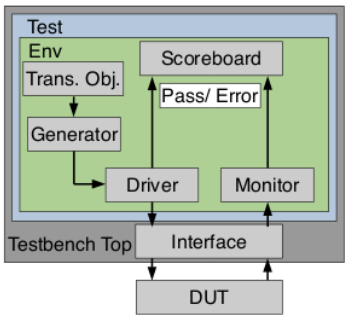
Gambar 20 Hasil *transcipt* DUT dengan kerusakan hubung singkat 26

Daftar Tabel

Tabel 1 Deskripsi Simbol dalam DUT.....5

Desain *Testbench*

Desain *testbench* bisa dilihat dalam Gambar 1. *Testbench* tersusun dari beberapa komponen, yaitu: *transaction object*, *generator, interface*, *driver*, *monitor*, *scoreboard*, *environment*, *test*, dan *testbench top*. Sedangkan DUT (*Design Under Test*) merupakan desain yang akan diuji



Gambar 1 Desain *testbench*

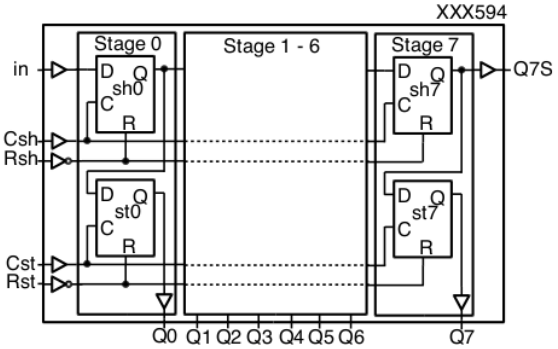
DUT

Rangkaian terpadu register geser atau *shift register* dapat ditemukan dalam unit pengaturan yaitu aplikasi pengaturan LED (*light emitting diode*) dan ekspansi/ antarmuka ke I/O dengan tujuan umum atau GPIO (*general purpose input/output*) [1]. Register geser dapat digunakan untuk menyimpan dan menggeser data. Dalam menggeser data, register geser memiliki beberapa cara, yaitu: input serial dan output serial atau SISO (serial-in serial-out), input serial dan output paralel atau SIPO (serial-in parallel-out), input paralel dan output serial atau PISO (paralel-in paralel-out), dan input paralel dan output paralel atau PIPO (paralel-in paralel-out) [2].

Contoh rangkaian register geser 8-bit SISO dan SIPO yang tersusun dari gerbang logika yaitu buffer dan inverter, dan rangkaian sekuensial yaitu flip-flop tipe D yang tersusun secara seri dan paralel bisa dilihat dalam Gambar 2. Sebagaimana ditunjukkan dalam Gambar 1, rangkaian register geser terdiri dari 8 *stage*, dimana di setiap *stage* tersusun dari dua flip-flop tipe D, yaitu satu flip-flop tipe-D sebagai penggeser *sh* dan yang satunya lagi sebagai penyimpan *st*, dan satu buffer. Jadi dalam hal ini, DUT adalah rangkaian terpadu register geser 8-bit. Deskripsi simbol dalam DUT bisa dijelaskan dalam Tabel 1.

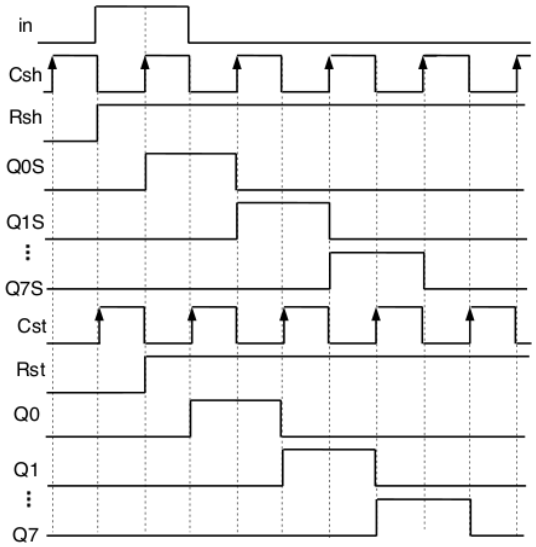
Tabel 1. Deskripsi simbol dalam DUT

Simbol	Deskripsi
in	Input data serial
Csh	Input klok register geser
Rsh	Input reset register geser
Rst	Input reset register penyimpan
Q0-to-Q7	Output data paralel
Q7S	Output data serial



Gambar 2 Rangkaian register geser 8-bit

Diagram waktu rangkaian register bisa dilihat dalam Gambar 3[3]. Ketika nilai logika *Rsh* dan *Rst* adalah rendah, maka semua flip-flop tipe D di dalam setiap stage akan *reset*, sehingga Q0 sampai Q7, dan Q7S akan bernilai logika rendah. Jika data di *in* merupakan sebuah pulsa logika tinggi dan pulsa klok di *Csh* berubah dari rendah ke tinggi, maka penggeser *sh* akan menggeser data tersebut secara serial dari *stage* 0 *Q0S* sampai ke *stage* 7 *Q7S*. Selanjutnya, ketika pulsa klok *Csh* berubah dari rendah ke tinggi, maka data di *Q0S* sampai *Q7S* tersebut akan dikeluarkan secara paralel di *Q*0 sampai *Q*7 oleh penyimpan *st*.

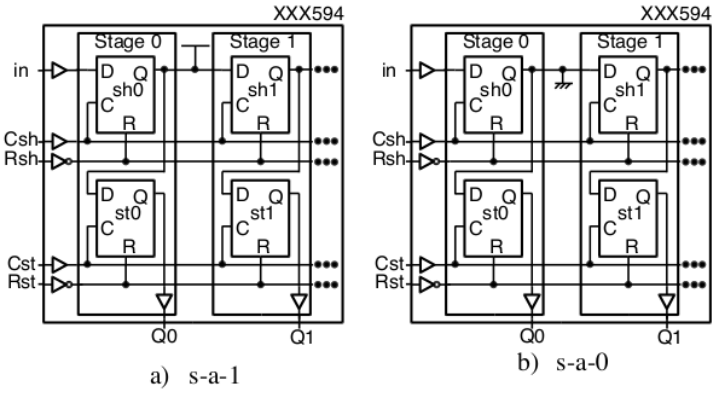


Gambar 3 Diagram waktu rangkaian register geser 8-bit

Kerusakan Hubung Singkat di DUT

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[4]. Contoh hubung singkat s-a-1 yang terjadi pada output rangkaian penyusun rangkaian terpadu register ditunjukkan dalam Gambar 4a. Sebagaimana ditunjukkan dalam Gambar a, nilai logika *Q0S* pada flip-flop D akan bernilai 1 apapun nilai logika dari inputan *in*.

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[5]. Contoh hubung singkat s-a-0 yang terjadi pada rangkaian penyusun rangkaian terpadu register geser bisa ditunjukkan dalam Gambar 4b. Dalam Gambar 4b, tanpa memperdulikan nilai logika dari *in*, maka nilai logika *Q0S* akan bernilai 0.



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

Gambar 4 Contoh hubung singkat s-a-1 dan s-a-0

SystemVerilog

SystemVerilog merupakan Bahasa deskripsi perangkat keras atau *Hardware Description Languange* (HDL) yang dapat digunakan untuk memprogram desain blok digital yang terdiri dari rangkaian kombinasional dan sekuensial yang bisa disimulasikan dan bisa disintesis ke dalam *hardware*, misalnya adalah FPGA (*Field Programmable Gate Array*) dan ASIC (*Application Specific Integrated Circuit*)[6]-[8]. SystemVerilog juga dapat memverifikasi desain blok digital tersebut apakah dapat bekerja sesuai dengan yang diharapkan[9],[10].

Untuk mendesain *testbench* dan rangkaian terpadu register geser menggunakan Bahasa SystemVerilog, ada beberapa fitur yang perlu diperhatikan yaitu tipe data, arah pengaturan, proses, komunikasi, antarmuka, kelas, *constraint*, jenis-jenis konstruksi, cakupan fungsi, *assertion*.

SystemVerilog memiliki tujuh tipe data, yaitu: *logic*, *bit*, *byte*, *shortint*, *int*, *longint*, *shortreal*. Tipe data *logic* memiliki empat keadaan data yaitu 0, 1, x (tidak diketahui), dan z (impedansi tinggi). Tipe data *logic* dapat digunakan dalam perintah blok procedural yaitu *always* and *initial* dan juga perintah *assign*.

Tipe data *bit* memiliki dua keadaan data yaitu 0 dan 1. Perintah tipe data *bit* banyak digunakan di *testbench* untuk merepresantikan bit tunggal dan untuk menyimpan beberapa bit dari MSB (*most significant bit*) ke LSB (*least significant bit*).

Fitur arah pengaturan memiliki beberapa konstruksi, yaitu: *forever*, *repeat*, *while*, *for*, *do while*, *for each, break*, *continue*, *if-else-if*, *case*, *events*, *function*, *tasks*, *blocking* dan *nonblocking*. Sedangkan fitur proses memiliki konstruksi, yaitu *fork join*, *fork join any*, *fork join-none*.

Fitur komunikasi memiliki konstruksi, yaitu: *event*, *semaphores*, dan *mailbox*. Untuk fitur antarmuka memiliki konstruksi, yaitu: *interface*, *clocking block* dan *modport*. Fitur kelas memiliki konstruksi, yaitu: *class, this*, *super*, *typedef*, *extern*, *rand, randomize, virtual*.

Fitur *constraint* memiliki konstruksi, yaitu: *inside*, *foreach*, *solve before*. Sedangkan fitur jenis-jenis konstruksi memiliki konstruksi, yaitu: *program*, *cast*, *package*. Untuk fitur cakupan fungsi memiliki konstruksi, yaitu: *mode*, *coverpoint*, *covergroup*, *bins*, *iff*, *start*, *stop*. Fitur *assertion* memiliki konstruksi, yaitu: *assert*, *assume*, *cover*, *restrict*, *sequence*, *property*.

Kode SystemVerilog

Rangakaian di dalam DUT sebagiamana ditunjukkan dalam Gambar 2 bisa dituliskan dalam bentuk kode SystemVerilog. Gambar 5 merupakan hasil penulisan kode DUT dalam Bahasa SystemVerilog.

```
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;

buff u0 (.d(dsh), .c(in));
buff u1 (.d(csh), .c(Csh));
buff u2 (.d(cst), .c(Cst));
inva u3 (.g(rsh), .f(Rsh));
inva u4 (.g(rst), .f(Rst));
stage u5 (.dsh(dsh), .csh(csh), .rsh(rsh), .cst(cst), .rst(rst), .qsh(qsh[0]), .qst(Q0));
stage u6 (.dsh(qsh[0]), .csh(csh), .rsh(rsh), .cst(cst), .rst(rst), .qsh(qsh[1]), .qst(Q1));
stage u7 (.dsh(qsh[1]), .csh(csh), .rsh(rsh), .cst(cst), .rst(rst), .qsh(qsh[2]), .qst(Q2));
stage u8 (.dsh(qsh[2]), .csh(csh), .rsh(rsh), .cst(cst), .rst(rst), .qsh(qsh[3]), .qst(Q3));
stage u9 (.dsh(qsh[3]), .csh(csh), .rsh(rsh), .cst(cst), .rst(rst), .qsh(qsh[4]), .qst(Q4));
stage u10 (.dsh(qsh[4]), .csh(csh), .rsh(rsh), .cst(cst), .rst(rst), .qsh(qsh[5]), .qst(Q5));
stage u11 (.dsh(qsh[5]), .csh(csh), .rsh(rsh), .cst(cst), .rst(rst), .qsh(qsh[6]), .qst(Q6));
stage u12 (.dsh(qsh[6]), .csh(csh), .rsh(rsh), .cst(cst), .rst(rst), .qsh(qsh[7]), .qst(Q7));
buff u13 (.d(Q7S), .c(qsh[7]));
assign Q = {Q7, Q6, Q5, Q4, Q3, Q2, Q1, Q0};

endmodule
```

Gambar 5 Kode DUT

Transaction Object

Transaction object berfungsi untuk memverifikasi apakah DUT dapat bekerja sesuai dengan diagram waktu sebagaimana ditunjukkan dalam Gambar 3. Kode *transaction object* dalam Bahasa SystemVerilog bisa ditunjukkan dalam Gambar 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;

  function void print(string tag="");
  $display ("T=%0t %s in=0x%0h Csh=0x%0h Rsh=0x%0h Cst=0x%0h Rst=0x%0h Q=0x%0h Q7S=0x%0h qsh=0x%0h", $time, tag, in, Csh, Rsh, Cst, Rst, Q, Q7S, qsh);
  endfunction

  function void copy(Packet tmp);

  this.in = tmp.in;
  this.Csh = tmp.Csh;
  this.Rsh = tmp.Rsh;
  this.Cst = tmp.Cst;
  this.Rst = tmp.Rst;
  this.Q = tmp.Q;
  this.Q7S = tmp.Q7S;
  this.qsh = tmp.qsh;

  endfunction
endclass
```

Gambar 6 Kode *Transaction Object*

Generator

Generator berfungsi untuk menghasilkan berbagai sinyal stimulus input untuk digerakkan ke DUT. Kode *generator* bisa dilihat dalam Gambar 7. Sebagaimana ditunjukkan dalam Gambar 7, *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
      Packet item = new;
      item.randomize();
      $display ("T=%0t [Generator] Loop:%0d/%0d create next item", $time, i+1, loop);
      drv_mbx.put(item);
      $display ("T=%0t [Generator] Wait for driver to be done", $time);
      @(drv_done);
    end
  endtask
endclass
```

Gambar 7 Kode *generator*

Driver

Sinyal stimulus yang dihasilkan oleh *generator* akan digerakkan ke DUT oleh *driver*. Kode *driver* bisa dilihat dalam Gambar 8. *Driver* tersusun dari beberapa konstruksi, yaitu: *class*, *virtual*, *event*, *mailbox*, *task*.

```
class driver;
  virtual sh594b_if m_sh594b_vif;
  virtual clk_if  m_clk_vif;
  event drv_done;
  mailbox drv_mbx;

  task run();
    $display ("T=%0t [Driver] starting ...", $time);

    forever begin
      Packet item;

      $display ("T=%0t [Driver] waiting for item ...", $time);
      drv_mbx.get(item);
      @ (posedge m_clk_vif.tb_clk);
            item.print("Driver");

          m_sh594b_vif.in <= item.in;
          m_sh594b_vif.Csh <= item.Csh;
          m_sh594b_vif.Rsh <= item.Rsh;
          m_sh594b_vif.Cst <= item.Cst;
          m_sh594b_vif.Rst <= item.Rst; ->drv_done;

    end
  endtask
endclass
```

Gambar 8 Kode *driver*

Interface

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 9. *Interface* tersusun dari beberapa konstruksi, yaitu: *interface*, *logic*.

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

interface clk_if();
  logic tb_clk;

  initial tb_clk <= 0;

  always #10 tb_clk = ~tb_clk;
endinterface
```

Gambar 9 Kode *interface*

Monitor

Monitor akan mengamati sinyal port input atau output dari *interface* untuk ditangkap atau disimpan. Contoh *pseudocode monitor* bisa dilihat dalam Gambar 10. *Monitor* tersusun dari beberapa konstruksi, yaitu: *class, virtual, mailbox, task*.

```
class monitor;
  virtual sh594b_if      m_sh594b_vif;
  virtual clk_if m_clk_vif;

  mailbox scb_mbx;               // Mailbox connected to scoreboard

  task run();
    $display ("T=%0t [Monitor] starting ...", $time);

      forever begin
            Packet m_pkt = new();
      @(posedge m_clk_vif.tb_clk);
      #1;
          m_pkt.in        = m_sh594b_vif.in;
        m_pkt.Csh         = m_sh594b_vif.Csh;
          m_pkt.Rsh       = m_sh594b_vif.Rsh;
          m_pkt.Cst       = m_sh594b_vif.Cst;
        m_pkt.Rst         = m_sh594b_vif.Rst;
          m_pkt.Q         = m_sh594b_vif.Q;
          m_pkt.Q7S       = m_sh594b_vif.Q7S;
        m_pkt.qsh         = m_sh594b_vif.qsh;
        m_pkt.print("Monitor");
      scb_mbx.put(m_pkt);
    end
  endtask
endclass
```

Gambar 10 Kode *monitor*

Scoreboard

Scoreboard berfungsi untuk memeriksa apakah sinyal output yang dihasilkan oleh DUT yang tertangkap di *monitor* sudah sesuai dengan yang diharapkan. Contoh *pseudocode scoreboard* bisa dilihat dalam Gambar 11. *Scoreboard* tersusun dari beberapa konstruksi, yaitu: *class, mailbox, task, forever*

```
class scoreboard;
  mailbox scb_mbx;

   task run();
    forever begin
      Packet item, ref_item;
      scb_mbx.get(item);
      item.print("Scoreboard");

          ref_item = new();
      ref_item.copy(item);

      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);


      if (ref_item.Q7S!= item.Q7S)
        $display("[%0t] Scoreboard Error! Q7S mismatch ref_item=0x%0h item=0x%0h",
$time, ref_item.Q7S, item.Q7S);
      else
        $display("[%0t] Scoreboard Pass! Q7S match ref_item=0x%0h item=0x%0h", $time,
ref_item.Q7S, item.Q7S);

    end
  endtask
endclass
```

Gambar 11 Kode *scoreboard*

Environment

Environment berisi komponen *generator*, *driver*, *monitor*, dan *scoreboard*. Contoh *pseudocode environment* bisa dilihat dalam Gambar 12. *Environment* tersusun dari beberapa konstruksi, yaitu: *class, generator, driver, monitor, scoreboard, mailbox, virtual, function*

```
class env;
  generator          g0;                    // Generate transactions
  driver                  d0;                    // Driver to design
  monitor                 m0;                    // Monitor from design
  scoreboard         s0;                    // Scoreboard connected to monitor
  mailbox                 scb_mbx;               // Top level mailbox for SCB <-> MON

  virtual sh594b_if     m_sh594b_vif;
  virtual clk_if m_clk_vif;              // TB clk

  event drv_done;
  mailbox drv_mbx;

  function new();
    d0 = new;
    m0 = new;
    s0 = new;
    scb_mbx = new();
    g0 = new;
    drv_mbx = new;
  endfunction

  virtual task run();

    d0.m_sh594b_vif = m_sh594b_vif;
    m0.m_sh594b_vif = m_sh594b_vif;
    d0.m_clk_vif = m_clk_vif;
    m0.m_clk_vif = m_clk_vif;

    d0.drv_mbx = drv_mbx;
    g0.drv_mbx = drv_mbx;

    m0.scb_mbx = scb_mbx;
    s0.scb_mbx = scb_mbx;

    d0.drv_done = drv_done;
    g0.drv_done = drv_done;

    fork
          s0.run();
                d0.run();
          m0.run();
          g0.run();
    join_any
  endtask
endclass
```

Gambar 12 Kode *environment*

Test

Test berisi *environment* yang dapat diatur konfigurasinya. Kode *test* bisa dilihat dalam Gambar 13. *Test* tersusun dari beberapa konstruksi, yaitu: *class, env, mailbox, function, virtual,task,*

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

  function new();
    drv_mbx = new();
    e0 = new();
  endfunction

  virtual task run();
    e0.d0.drv_mbx = drv_mbx;
    e0.run();
  endtask
endclass
```

Gambar 13 Kode *test*

Testbench top

Testbench top digunakan untuk menguji DUT. Kode *testbench* bisa dilihat dalam Gambar 14. *Testbench top* tersusun dari beberapa konstruksi, yaitu: *module, bit, test*.

```
module tb_sh594b3;
  bit tb_clk;

  bit in, Csh, Rsh, Cst, Rst;
  wire [7:0] qsh;

  always #2 Csh =~ Csh;
  always #2 Cst =~ Cst;

  clk_if        m_clk_if      ();
  sh594b_if     m_sh594b_if   ();
```

```
 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;

   Csh <= 1;
   Rsh <= 0;
   Rst <= 0;
   Cst <= 0;
   in <= 0;

   // Apply reset and start stimulus
   #2 Rsh <= 1;
      in <= 1;
   #2 Rst <= 1;
   #2   in <= 0;

   t0 = new;
   t0.e0.m_sh594b_vif = m_sh594b_if;
   t0.e0.m_clk_vif = m_clk_if;
   t0.run();

   #50 $finish;
 end

 initial begin
   $dumpvars;
   $dumpfile ("dump.vcd");
```

```
 end
endmodule
```

Gambar 14 Kode *testbench top*

Kode SystemVerilog dengan kerusakan hubung singkat di DUT

Untuk menguji DUT dari kerusakan hubung singkat, dua komponen *testbench* perlu dirubah kodenya, yaitu: *scoreboard* dan *testbench top*. Kode *scoreboard* yang digunakan untuk menguji DUT dengan kerusakan hubung singkat bisa dilihat dalam Gambar 15.

```
class scoreboard;
  mailbox scb_mbx;

   task run();
    forever begin
      Packet item, ref_item;
      scb_mbx.get(item);
      item.print("Scoreboard");
      ref_item = new();
      ref_item.copy(item);

           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

       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);


    if (ref_item.Q7S!= item.Q7S)
     $display("[%0t] Scoreboard Error! Q7S mismatch ref_item=0x%0h item=0x%0h", $time, ref_item.Q7S, item.Q7S);
    else
     $display("[%0t] Scoreboard Pass! Q7S match ref_item=0x%0h item=0x%0h", $time, ref_item.Q7S, item.Q7S);


   end
  endtask
endclass
```

Gambar 15 Kode *scoreboard* ketika DUT dengan kerusakan hubung singkat

. Kode *testbench top* yang digunakan untuk menguji DUT dengan kerusakan hubung singkat ke suplai tegangan pada interkoneksi antara *stage* 0 ke *stage* 1 sebagaimana diilustrasikan dalam Gambar 4a bisa dilihat dalam Gambar 16. Instruksi *force* dan *release* dimasukkan dalam kode *testbench top*.

```
module tb_sh594b3;
  bit tb_clk;

  bit in, Csh, Rsh, Cst, Rst;
  wire [7:0] qsh;

  always #2 Csh =~ Csh;
  always #2 Cst =~ Cst;

  clk_if          m_clk_if        ();
  sh594b_if       m_sh594b_if     ();
  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;

   Csh <= 1;
   Rsh <= 0;
   Rst <= 0;
   Cst <= 0;
   in <= 0;

   // Apply reset and start stimulus
   #2 Rsh <= 1;
      in <= 1;
   #2 Rst <= 1;
   #2   in <= 0;
   #35 force qsh[0]=1;
```

```
    #4 release qsh[7];

    t0 = new;
    t0.e0.m_sh594b_vif = m_sh594b_if;
    t0.e0.m_clk_vif = m_clk_if;
    t0.run();

    #50 $finish;
  end

  initial begin
    $dumpvars;
    $dumpfile ("dump.vcd");
  end
endmodule
```

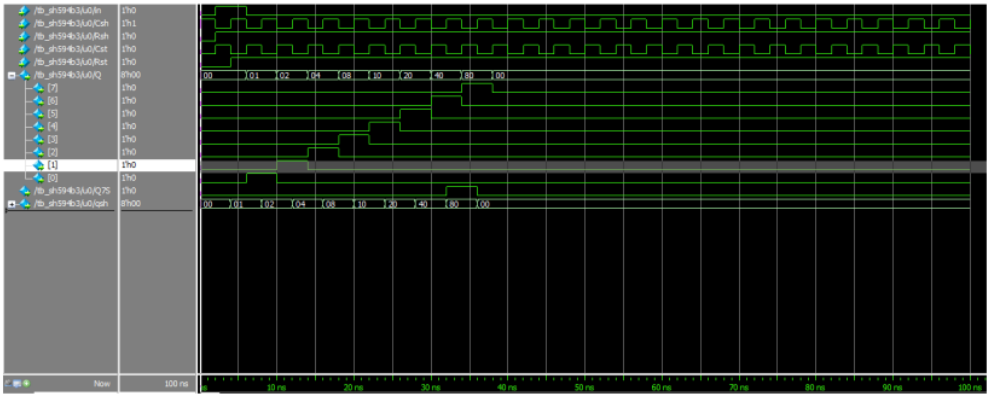
Gambar 16 Kode *testbench top* ketika DUT dengan kerusakan hubung singkat

Questasim 10.7c

Kode *testbench* dan DUT dalam Bahasa SystemVerilog kemudian disimulasikan menggunakan software Questasim 10.7c. Questasim 10.7c merupakan *software* simulasi dan *debugging* yang dapat digunakan untuk memverifikasi desain rangkaian terpadu yang dirancang menggunakan Bahasa SystemVerilog[15]. *Software* ini memberikan fasilitas antar muka grafis yang menarik atau GUI (*graphical user interface*).

Hasil Simulasi dan Pembahasan

Hasil simulasi DUT tanpa kerusakan hubung singkat bisa ditunjukkan dalam Gambar 17. Hasil simulasi menujukkan kesesuaian dengan diagram waktu DUT dalam Gambar 3. Hasil *transcript* dari Questasim juga memberikan keterangan *Scoreboard Pass* sebagaimana ditunjukkan dalam Gambar 18.



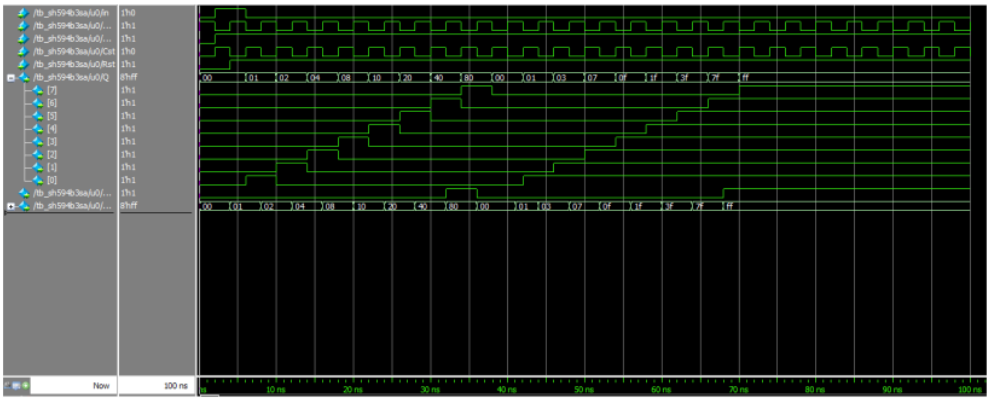
Gambar 17 Hasil simulasi DUT tanpa kerusakan hubung singkat

```
# T=6 [Driver] starting ...
# T=6 [Driver] waiting for item ...
# T=6 [Monitor] starting ...
# T=6 [Generator] Loop:1/10 create next item
# T=6 [Generator] Wait for driver to be done
# T=10 Driver in=0x0 Csh=0x0 Rsh=0x0 Cst=0x0 Rst=0x0 Q=0xc9 Q7S=0x0 qsh=0x8a
# T=10 [Driver] waiting for item ...
# T=10 [Generator] Loop:2/10 create next item
# T=10 [Generator] Wait for driver to be done
# T=11 Monitor in=0x0 Csh=0x0 Rsh=0x0 Cst=0x0 Rst=0x0 Q=0x2 Q7S=0x0 qsh=0x0
# T=11 Scoreboard in=0x0 Csh=0x0 Rsh=0x0 Cst=0x0 Rst=0x0 Q=0x2 Q7S=0x0 qsh=0x0
# [11] Scoreboard Pass! Q match ref_item=0x2 item=0x2
# [11] Scoreboard Pass! Q7S match ref_item=0x0 item=0x0
# T=30 Driver in=0x0 Csh=0x0 Rsh=0x0 Cst=0x0 Rst=0x0 Q=0xd0 Q7S=0x0 qsh=0xa2
# T=30 [Driver] waiting for item ...
# T=30 [Generator] Loop:3/10 create next item
# T=30 [Generator] Wait for driver to be done
# T=31 Monitor in=0x0 Csh=0x0 Rsh=0x0 Cst=0x0 Rst=0x0 Q=0x40 Q7S=0x0 qsh=0x0
# T=31 Scoreboard in=0x0 Csh=0x0 Rsh=0x0 Cst=0x0 Rst=0x0 Q=0x40 Q7S=0x0 qsh=0x0
# [31] Scoreboard Pass! Q match ref_item=0x40 item=0x40
# [31] Scoreboard Pass! Q7S match ref_item=0x0 item=0x0
# T=50 Driver in=0x0 Csh=0x0 Rsh=0x0 Cst=0x0 Rst=0x0 Q=0x49 Q7S=0x0 qsh=0x9e
# T=50 [Driver] waiting for item ...
# T=50 [Generator] Loop:4/10 create next item
# T=50 [Generator] Wait for driver to be done
# T=51 Monitor in=0x0 Csh=0x0 Rsh=0x0 Cst=0x0 Rst=0x0 Q=0x0 Q7S=0x0 qsh=0x0
# T=51 Scoreboard in=0x0 Csh=0x0 Rsh=0x0 Cst=0x0 Rst=0x0 Q=0x0 Q7S=0x0 qsh=0x0
# [51] Scoreboard Pass! Q match ref_item=0x0 item=0x0
# [51] Scoreboard Pass! Q7S match ref_item=0x0 item=0x0
# T=70 Driver in=0x0 Csh=0x0 Rsh=0x0 Cst=0x0 Rst=0x0 Q=0x27 Q7S=0x0 qsh=0xd0
# T=70 [Driver] waiting for item ...
```

```
# T=70 [Generator] Loop:5/10 create next item
# T=70 [Generator] Wait for driver to be done
# T=71 Monitor in=0x0 Csh=0x0 Rsh=0x0 Cst=0x0 Rst=0x0 Q=0x0 Q7S=0x0 qsh=0x0
# T=71 Scoreboard in=0x0 Csh=0x0 Rsh=0x0 Cst=0x0 Rst=0x0 Q=0x0 Q7S=0x0 qsh=0x0
# [71] Scoreboard Pass! Q match ref_item=0x0 item=0x0
# [71] Scoreboard Pass! Q7S match ref_item=0x0 item=0x0
# T=90 Driver in=0x0 Csh=0x0 Rsh=0x0 Cst=0x0 Rst=0x0 Q=0xb7 Q7S=0x0 qsh=0xcb
# T=90 [Driver] waiting for item ...
# T=90 [Generator] Loop:6/10 create next item
# T=90 [Generator] Wait for driver to be done
# T=91 Monitor in=0x0 Csh=0x0 Rsh=0x0 Cst=0x0 Rst=0x0 Q=0x0 Q7S=0x0 qsh=0x0
# T=91 Scoreboard in=0x0 Csh=0x0 Rsh=0x0 Cst=0x0 Rst=0x0 Q=0x0 Q7S=0x0 qsh=0x0
# [91] Scoreboard Pass! Q match ref_item=0x0 item=0x0
# [91] Scoreboard Pass! Q7S match ref_item=0x0 item=0x0
```

Gambar 18 Hasil *transcipt* DUT tanpa kerusakan hubung singkat

Gambar 19 merupakan hasil simulasi DUT dengan kerusakan hubung singkat ke *ground*. Hasil simulasi menujukkan ketidaksesuaian dengan diagram waktu DUT dalam Gambar 3. Keterangan *Scoreboard Error* juga muncul dalam hasil *transcript* dari Questasim sebagaimana ditunjukkan dalam Gambar 20.



Gambar 18 Hasil simulasi DUT tanpa kerusakan hubung singkat

```
# T=45 [Driver] starting ...
# T=45 [Driver] waiting for item ...
# T=45 [Monitor] starting ...
# T=45 [Generator] Loop:1/10 create next item
# T=45 [Generator] Wait for driver to be done
# T=50 Driver in=0x0 Csh=0x0 Rsh=0x0 Cst=0x0 Rst=0x0 Q=0x37 Q7S=0x0 qsh=0xde
# T=50 [Driver] waiting for item ...
# T=50 [Generator] Loop:2/10 create next item
# T=50 [Generator] Wait for driver to be done
# T=51 Monitor in=0x0 Csh=0x0 Rsh=0x0 Cst=0x0 Rst=0x0 Q=0x7 Q7S=0x0 qsh=0x0
# T=51 Scoreboard in=0x0 Csh=0x0 Rsh=0x0 Cst=0x0 Rst=0x0 Q=0x7 Q7S=0x0 qsh=0x0
# [51] Scoreboard Error! Q mismatch ref_item=0x0 item=0x7
# [51] Scoreboard Pass! Q7S match ref_item=0x0 item=0x0
# T=70 Driver in=0x0 Csh=0x0 Rsh=0x0 Cst=0x0 Rst=0x0 Q=0xf0 Q7S=0x0 qsh=0xf1
# T=70 [Driver] waiting for item ...
# T=70 [Generator] Loop:3/10 create next item
# T=70 [Generator] Wait for driver to be done
# T=71 Monitor in=0x0 Csh=0x0 Rsh=0x0 Cst=0x0 Rst=0x0 Q=0xff Q7S=0x1 qsh=0x0
# T=71 Scoreboard in=0x0 Csh=0x0 Rsh=0x0 Cst=0x0 Rst=0x0 Q=0xff Q7S=0x1 qsh=0x0
# [71] Scoreboard Error! Q mismatch ref_item=0x0 item=0xff
# [71] Scoreboard Pass! Q7S match ref_item=0x1 item=0x1
# T=90 Driver in=0x0 Csh=0x0 Rsh=0x0 Cst=0x0 Rst=0x0 Q=0xe6 Q7S=0x0 qsh=0x6c
# T=90 [Driver] waiting for item ...
# T=90 [Generator] Loop:4/10 create next item
# T=90 [Generator] Wait for driver to be done
# T=91 Monitor in=0x0 Csh=0x0 Rsh=0x0 Cst=0x0 Rst=0x0 Q=0xff Q7S=0x1 qsh=0x0
# T=91 Scoreboard in=0x0 Csh=0x0 Rsh=0x0 Cst=0x0 Rst=0x0 Q=0xff Q7S=0x1 qsh=0x0
# [91] Scoreboard Error! Q mismatch ref_item=0x0 item=0xff
# [91] Scoreboard Pass! Q7S match ref_item=0x1 item=0x1
```

Gambar 19 Hasil *transcipt* DUT dengan kerusakan hubung singkat

Kesimpulan

Desain *testbench* yang dirancang menggunakan Bahasa SystemVerilog diajukan untuk mendeteksi kerusakan hubung singkat yang terjadi pada interkoneksi antar rangkaian penyusun di dalam rangkaian terpadu register geser 8-bit. *Testbench* tersusun dari komponen-komponen yaitu: *transaction object*, *generator, interface*, *driver*, *monitor*, *scoreboard*, *environment*, *test,* dan *testbench top*. Selanjutnya, *testbench* dan rangkaian terpadu register geser disimulasikan menggunakan *software* Questasim. Hasil simulasi menunjukkan bahwa kerusakan hubung singkat yang terjadi dalam rangkaian terpadu register geser dapat dapat dideteksi menggunkan *testbench*. Hasil *transcript* juga memberikan keterangan *scoreboard error*.

Daftar Pustaka

1. Thomas L. Flyod. Digital Fundamental. Pearson Education Limited: 2015
2. Thomas L. Flyod. Digital Fundamental A System Approach. Pearson Education Limited: 2014
3. Selahattin Sayil. Contactless VLSI Measurement and Testing Technique. Springer. 2018.
4. C. Wu and K. Lee, "An Efficient Diagnosis Pattern Generation Procedure to Distinguish Stuck-at Faults and Bridging Faults," 2014 IEEE 23rd Asian Test Symposium, Hangzhou, China, 2014, pp. 306-311, doi: 10.1109/ATS.2014.56.
5. A. P. Singh, S. S. Sodhi, P. Chandra and C. S. Rai, "Metrics for Weight Stuck-at-Zero Fault in Sigmoidal FFANNs," 2013 European Modelling Symposium, Manchester, UK, 2013, pp. 61-66, doi: 10.1109/EMS.2013.11.
6. Tigranuhi G, Heghineh M, Gevorg M, Valery V. Fault Collapsing for Digital Circuits Based on Relations between Stuck-at-Faults. Prosiding Seminar Internasional CSIT. 2015: 15 – 18.
7. X. Lin, S. M. Reddy and J. Rajski, "Using Boolean Tests to Improve Detection of Transistor Stuck-Open Faults in CMOS Digital Logic Circuits," 2015 28th International Conference on VLSI Design, Bangalore, India, 2015, pp. 399-404, doi: 10.1109/VLSID.2015.73.
8. Mark Zwolinski. Digital System Design with SystemVerilog. Prentice Hall. 2018
9. Stuart Stuherland. RTL Modeling with SystemVerilog for Simulation and Synthesis using SystemVerilog for ASIC and FPGA design. Stuherland Inc. 2017
10. Spear. SystemVerilog for Verification. Springer. 2012.
11. Vaibbhav Taraate. SystemVerilog for Hardware Description: RTL Design and Verification. Springer. 2020

Testbench untuk Mendeteksi Kerusakan Hubung Singkat di dalam Rangkaian Terpadu Register Geser yang Didesain Menggunakan SystemVerilog

ORIGINALITY REPORT

3%	1%	2%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Vaibbhav Taraate. "SystemVerilog for Hardware Description", Springer Science and Business Media LLC, 2020 Publication	2%
2	simppm.umm.ac.id Internet Source	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 Juli-September 2023
Submission title:	Testbench untuk Mendeteksi Kerusakan Hubung Singkat di ...
File name:	WIDIANTO_ELKTRO_UMM_2022.docx
File size:	290K
Page count:	28
Word count:	3,774
Character count:	23,226
Submission date:	05-Jul-2023 10:23AM (UTC+0700)
Submission ID:	2126645257

Testbench **untuk Mendeteksi Kerusakan Hubung Singkat di dalam Rangkaian Terpadu Register Geser yang Didesain Menggunakan SystemVerilog**

PENGUSUL

Widianto, ST., MT

TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS MUHAMMADIYAH MALANG

2022

Copyright 2023 Turnitin. All rights reserved.