Jurnal Penelitian Inovatif (JUPIN)
Vol. 5, No. 1, Februari 2025, Hal. 669-684
https://jurnal-id.com/index.php/jupin

DOI: https://doi.org/10.54082/jupin.994
p-ISSN: 2808-148X
e-ISSN: 2808-1366

Sequencer UVM dalam Mendeteksi Kerusakan Hubung Singkat didalam Rangkaian Terpadu Multivibrator

Widianto*[1], Baiq Dewi Eriyani[2], M. Chasrun H.[3]

[1]Teknologi Elektronika, Fakultas Vokasi, Universitas Muhammadiyah Malang, Indonesia
[2,3]Teknik Elektro, Fakultas Teknik, Universitas Muhammadiyah Malang, Indonesia
Email: [1]widianto@umm.ac.id, [2]baiqdewi21@umm.ac.id, [3]chasrun@umm.ac.id

Abstrak

Rangkaian terpadu multivibrator dapat diaplikasikan dalam mengatur waktu tunda di rangkaian kontrol. Output yang dihasilkan oleh rangkaian terpadu multivibrator berupa satu pulsa (monostable multivibrator) dan banyak pulsa (astable multivibrator). Lebar pulsa yang dihasilkan oleh rangkaian terpadu multivibrator ditentukan oleh komponen eksternal yaitu R (resistor) dan C (kapasitor) yang dipasang di rangkaian terpadu tersebut. Kerusakan hubung singkat ke suplai tegangan dan ke ground bisa saja terjadi pada input atau output rangkaian penyusun di dalam rangkaian terpadu multivibrator. Input suatu rangkaian penyusun di dalam rangkaian terpadu multivibrator akan tetap bernilai logika 1 (tinggi) jika kerusakan hubung singkat ke suplai tegangan terjadi pada input rangkaian penyusun tersebut. Sedangkan output suatu rangkaian penyusun di dalam multivibrator akan tetap bernilai logika 0 (rendah) tidak mempedulikan apapun logika nilai inputnya jika terjadi kerusakan hubung singkat ke ground pada input rangkaian penyusun tersebut. Oleh sebab itu, hubung singkat ke suplai tegangan dan ke ground yang terjadi di dalam rangkaian terpadu multivibrator harus dideteksi sebelum dikirimkan ke konsumen. Desain UVM testbench yang diajukan untuk memverifikasi rangkaian terpadu multivibrator dari kerusakan hubung singkat yang terjadi di dalamnya. Testbench tersusun dari beberapa komponen, yaitu: sequence, sequencer, interface, driver, monitor, scoreboard, agent, environment, test, dan testbench top. Sedangkan DUT (Design Under Test) merupakan desain yang akan diuji, dalam hal ini adalah rangkaian terpadu multivibrator. Kode UVM *testbench* dan DUT dalam Bahasa SystemVerilog kemudian disimulasikan menggunakan software Questasim 2021.1. Hasil simulasi menujukkan kesesuaian dengan diagram waktu DUT. Hasil *transcript* dari Questasim juga memberikan keterangan "UVM_ERROR : 0".

Kata Kunci: *Deteksi Kesalahan, Multivibrator, Rangkaian Terpadu, Sequencer, Systemverilog, UVM*

Abstract

Multivibrator integrated circuits can be applied in setting time delays in control circuits. The output produced by the multivibrator integrated circuit is in the form of one pulse (monostable multivibrator) and many pulses (astable multivibrator). The width of the pulse produced by the multivibrator integrated circuit is determined by the external components R (resistor) and C (capacitor) installed in the integrated circuit. Short circuit damage to the voltage supply and to ground may occur at the input or output of the building blocks in the multivibrator integrated circuit. The input of a circuit in a multivibrator integrated circuit will remain logic 1 (high) if a short circuit to the voltage supply occurs at the input of the circuit. Meanwhile, the output of a circuit in a multivibrator will remain logic 0 (low) regardless of the logic value of the input if a short circuit to ground occurs at the input of the circuit. Therefore, short circuits to the voltage supply and to ground that occur within the multivibrator integrated circuit must be detected before being shipped to consumers. The proposed UVM testbench is designed to verify the multivibrator integrated circuits for short circuit damage. The testbench is composed of several components, namely: sequence, sequencer, interface, driver, monitor, scoreboard, agent, environment, test, and testbench top. While the DUT (Design Under Test) is the design to be tested, in this case it is a multivibrator integrated circuit. The UVM testbench and DUT code in SystemVerilog language are then simulated using Questasim 2021.1 software. The simulation results show conformity with the DUT timing diagram. The transcript results from Questasim also provide information "UVM_ERROR : 0".

Keywords: *Error Detection, Multivibrator, Integrated Circuit, Sequencer, Systemverilog, UVM*

DOI: https://doi.org/10.54082/jupin.994
p-ISSN: 2808-148X
e-ISSN: 2808-1366

1. PENDAHULUAN

Kerusakan hubung singkat ke suplai tegangan dan ke ground bisa terjadi pada input atau output rangkaian penyusun di dalam rangkaian terpadu(Smirnov et al., 2020). 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.

Metode ATPG telah diajukan untuk mendeteksi kerusakan hubung singkat di dalam rangkaian terpadu(P. Wang et al., 2019; P. Wang et al., 2020; F. Wang & Gupta, 2020; Kung et al., 2020). Metode ini diujikan pada benchmark desain rangkaian terpadu ISCAS 1989 dan IWLS 2005. Namun metode ini memanfaatkan alat ATPG untuk menguji kerusakan hubung singkat, dimana alat ini harganya tidaklah murah.

Kerusakan hubung singkat di dalam rangkaian terpadu telah dideteksi menggunakan metode DNN (Deep Neural Network) (Ishii & Namba, 2022). Metode ini memerlukan pengaturan parameter ambang batas yang diukur berdasarkan nili deviasi dan median. Tetapi hasil pendeteksian kerusakan hubung singkat masih belum sempurna menggunakan metode ini.

Sinar X-Ray diajukan untuk mendeteksi kerusakan hubung singkat di dalam rangkaian terpadu(Tebina et al., 2022). X-Ray mampu menginduksi perpindahan oksidasi dalam komponen MOS (Metal Oxide Semiconductor). Tetapi, peralatan X-Ray ini memerlukan biaya yang tidaklah sedikit.

Metode fault tolerant telah diajukan untuk mendeteksi kerusakan hubung singkat di dalam rangkaian terpadu(Tebina et al., 2022). Metode ini memerlukan waktu yang lama dalam menentukan tipe kerusakan s-a-1 atau s-a-0 yang terjadi di dalam rangkaian terpadu, karena hanya satu tipe kerusakan yang diuji menggunakan metode ini dalam setiap pengujian.

Kerusakan hubung singkat di dalam rangkaian terpadu dapat dideteksi menggunakan Machine Learning(Higami et al., 2022). NN (Neural Network) yang akan mengolah data respon dari rangkaian terpadu dalam mendeteksi kerusakan. Tetapi NN memerlukan waktu lama dalam mendeteksi.

Augmented circuit diajukan untuk mendeteksi kerusakan hubung singkat di dalam rangkaian terpadu (Handique et al., 2021). Dalam mendeteksi kerusakan, rangkaian ini memerlukan kontruksi vector pengujian berurutan yang rumit.

Rangkaian penguji atau BIST yang terdiri dari rangkaian PG (pattern generator), SA (signature analyzer), multiplekser, dan demultiplekser telah diajukan untuk menguji rangkaian terpadu logika kombinasional dari kerusakan hubung singkat yang terjadi di dalam rangkaian terpadu tersebut(Widianto dan Robert Lis, 2019). Multiplekser dan demultiplekser digunakan oleh rangkaian penguji untuk memilih bekerja pada mode normal atau mode pengujian. Tetapi, 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(Widianto dan Robert Lis, 2020). Semua output dari rangkaian terpadu logika kombinasional akan dikoneksikan ke input rangkaian SR, sehingga hanya membutuhkan satu rangkaian SR. Tetapi respon dari rangkaian SR akan membutuhkan pencermatan ketika terjadi kerusakan hubung singkat, karena belum dilengkapi dengan fitur parameter-parameter pendeteksian kerusakan hubung singkat.

BIST dan rangkaian terpadu logika kombinasional (Widianto dan Robert Lis, 2019; Widianto dan Robert Lis, 2020) dirancang menggunakan Bahasa Verilog, dimana Bahasa Verilog memiliki keterbatasan fasilitas fitur parameter pengujian dalam pendeteksian kerusakan hubung singkat yang belum lengkap.

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(Widianto et al., 2022). Testbench tersusun dari beberapa komponen, yaitu: transaction object, generator, interface, driver, monitor, scoreboard, environment, test, dan testbench top. Testbench dan rangkaian terpadu komparator dirancang menggunakan SystemVerilog

DOI: https://doi.org/10.54082/jupin.994
p-ISSN: 2808-148X
e-ISSN: 2808-1366

dan disimulasikan menggunakan Questasim 10.7c. Tetapi komponen testbench yaitu driver, monitor, dan scoreboard akan berbeda ketika DUT juga berbeda.

Oleh sebab itu dalam penelitian ini, UVM (Universal Verification Methodology) diajukan untuk memverifikasi rangkaian terpadu multivibrator dari kerusakan hubung singkat ke suplai tegangan dan ke ground yang terjadi pada input atau output rangkaian penyusun di dalam rangkaian terpadu tersebut. UVM terdiri dari testbench dan DUT (Design Under Test).

Tesbench yang diajukan dapat mendeteksi 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 multivibrator dari kerusakan hubung singkat.

Testbench tersusun dari beberapa komponen, yaitu: sequence, sequencer, interface, driver, monitor, scoreboard, agent, environment, test, dan testbench top. Sedangkan DUT atau desain yang akan diuji yaitu rangkaian terpadu multivibrator.

Sequence merupakan sinyal kontrol input dan output yang akan digerakkan ke DUT. Sequencer berfungsi untuk menghasilkan berbagai sinyal stimulus input untuk digerakkan ke DUT. Sinyal stimulus yang dihasilkan oleh sequencer akan digerakkan ke DUT oleh driver. Interface berfungsi untuk menggerakkan sinyal dari sequencer 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. Agent berisi komponen sequencer, driver, dan monitor. Environment berisikan agent dan scoreboard. Test berisi environment yang dapat diatur konfigurasinya. Testbench top berisi test dan juga DUT.

Desain tesbench yang diajukan dan rangkaian terpadu multivibrator akan disusun menggunakan Bahasa SystemVerilog dan disimulasikan menggunkan Questasim 2021.1. 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 multivibrator tidak memiliki kerusakan hubung singkat di dalamnya. Ketika testbench dengan rangkaian terpadu multivibrator memiliki kerusakan hubung singkat di dalamnya, maka parameter error akan muncul di Questasim. Sehingga ke depan, desain testbench dan rangkaian terpadu multivibrator yang disusun menggunakan Bahasa SystemVerilog tidak hanya terbatas pada simulasi di Questasim, tetapi dapat diimpelementasikan ke dalam hardware, misalnya FPGA.

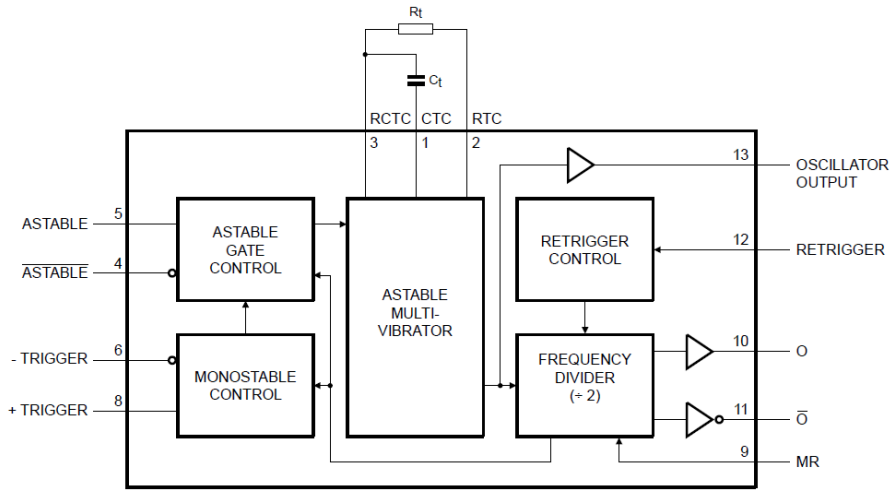
Oleh karena itu tujuan dari penelitian ini adalah dapat mengembangkan testbench berbasis Universal Verification Methodology (UVM) untuk mendeteksi kerusakan hubung singkat pada rangkaian terpadu multivibrator. Testbench ini dirancang menggunakan Bahasa SystemVerilog dan disimulasikan dengan Questasim 2021.1, yang memungkinkan verifikasi lebih akurat terhadap setiap input dan output rangkaian penyusun dalam rangkaian terpadu

2. TINJAUAN PUSTAKA

2.1. Rangkaian Terpadu Multivibrator

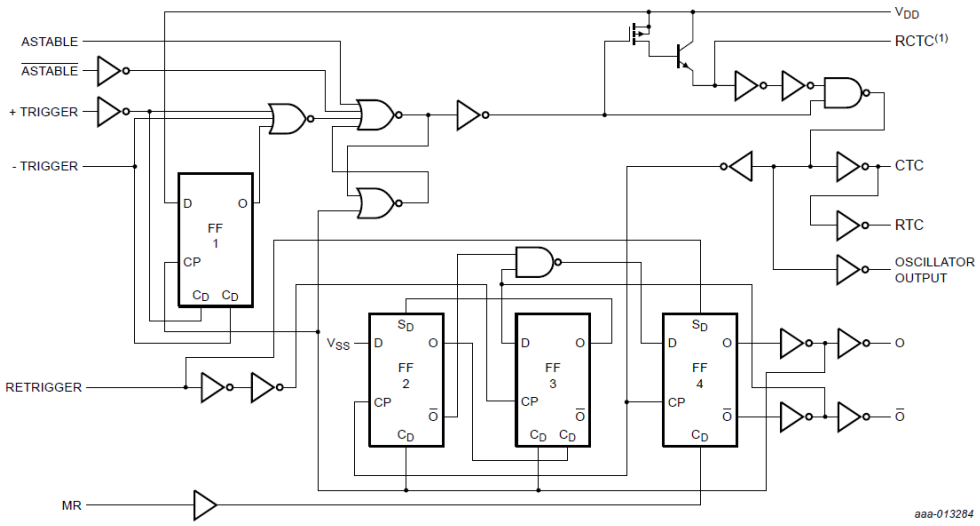
Rangkaian terpadu multivibrator dapat ditemukan dalam pengaturan waktu tunda di rangkaian kontrol(Christian Bakhau, 2020). Output yang dihasilkan oleh rangkaian terpadu multivibrator berupa satu pulsa (monostable multivibrator) dan banyak pulsa (astable multivibrator). Lebar pulsa yang dihasilkan oleh rangkaian terpadu multivibrator ditentukan oleh komponen eksternal yaitu R (resistor) dan C (kapasitor) yang dipasang di rangkaian terpadu tersebut. Diagram fungsi dari rangkaian terpadu multivibrator bisa dilihat dalam Gambar 1. Rangkaian terpadu multivibrator terdiri dari astable gate control, monostable control, astable multi-vibrator, retrigger control, dan frequency divider.

DOI: https://doi.org/10.54082/jupin.994
p-ISSN: 2808-148X
e-ISSN: 2808-1366



Gambar 1. Diagram fungsi rangkaian multivibrator

Rangkaian penyusun dari rangkaian terpadu multivibrator tediri gerbang logika yaitu INVERTER, NOR, D Flip-flop, Buffer, NAND, PMOS, dan NPN transistor. Diagram logika rangkaian multivibrator bisa dilihat dalam Gambar 2. Deskripsi pin rangkaian multivibrator ditunjukkan dalam Tabel 1.



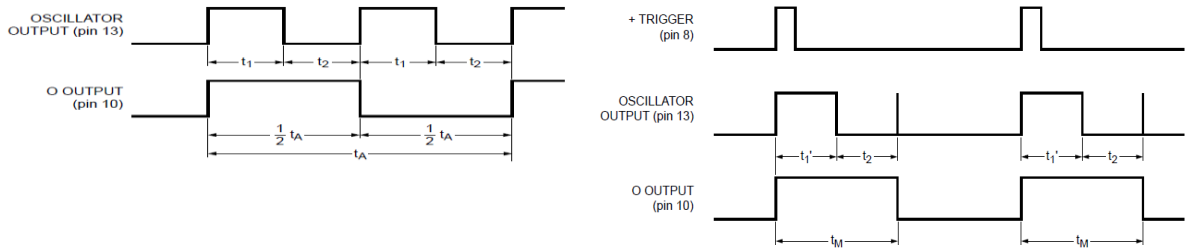
Gambar 2. Diagram logika rangkaian multivibrator

Tabel 1. Deskripsi Pin Rangkaian Multivibrator

Simbol	Deskripsi
CTC	Koneksi kapasitor eksternal
RTC	Koneksi resistor eksternal
RCTC	Koneksi resistor/ kapasitor eksternal
$\overline{ASTABLE}$	Input
ASTABLE	Input
-TRIGGER	Input
VSS	Ground
+TRIGGER	Input
MR	Master reset input
O	Output
$\bar{O}$	Ouput
RETRIGGER	Input
OSCILLATOR OUTPUT	Oscillator output
VDD	Suplai tegangan

DOI: https://doi.org/10.54082/jupin.994
p-ISSN: 2808-148X
e-ISSN: 2808-1366

Rangkaian multivibrator dapat menghasilkan pulsa yaitu satu pulsa (monostable multivibrator) dan banyak pulsa (astable multivibrator). Lebar pulsa yang dihasilkan oleh rangkaian terpadu multivibrator bergantung pada komponen eksternal yaitu R (resistor) dan C (kapasitor) yang dipasang di rangkaian terpadu tersebut. Bentuk gelombang rangkaian multivibrator bisa dilihat dalam Gambar 3.



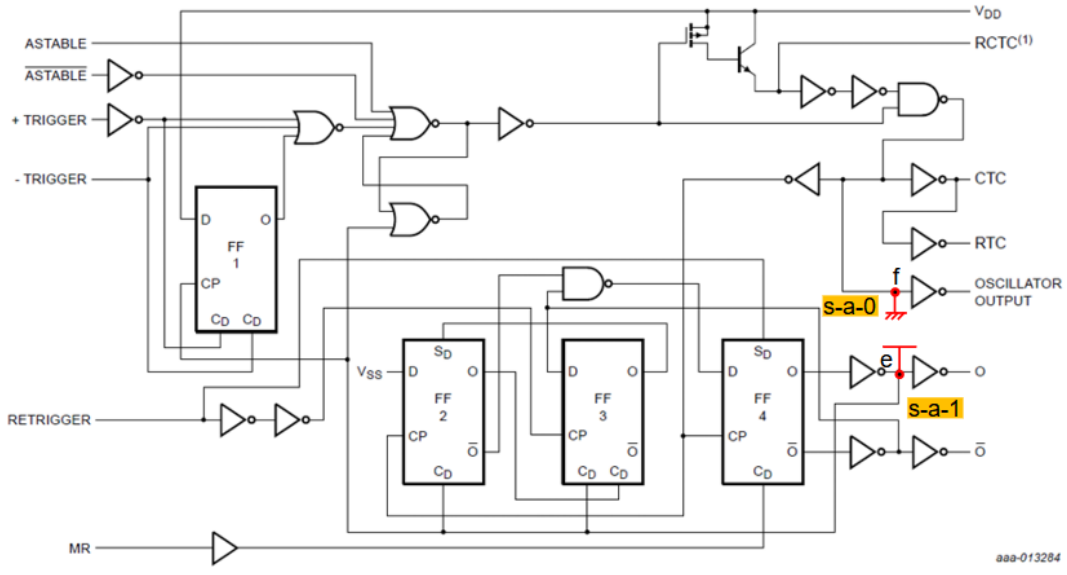
(a) Bentuk gelombang astable (b) Bentuk gelombang monostable

Gambar 3. Bentuk gelombang rangkaian multivibrator

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(Smirnov et al., 2020). Contoh hubung singkat s-a-1 yang terjadi pada output rangkaian penyusun rangkaian terpadu multivibrator yang ditunjukkan dalam Gambar 4 titik e. Sebagaimana ditunjukkan dalam Gambar 4, nilai logika O pada output logika INVERTER akan bernilai 1 apapun nilai logika dari titik 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(Smirnov et al., 2020). Contoh hubung singkat s-a-0 yang terjadi pada rangkaian penyusun rangkaian terpadu multivibrator bisa ditunjukkan dalam Gambar 4 titik f. Dalam Gambar 4, tanpa memperdulikan nilai logika dari inputan f, maka nilai logika pada output logika INVERTER OSCILLATOR OUTPUT akan bernilai 0.



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

2.3. 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*

DOI: https://doi.org/10.54082/jupin.994
p-ISSN: 2808-148X
e-ISSN: 2808-1366

Circuit)(Handique et al., 2021; Widianto dan Robert Lis, 2019; (Widianto dan Robert Lis, 2020). SystemVerilog juga dapat memverifikasi desain blok digital tersebut apakah dapat bekerja sesuai dengan yang diharapkan(Widianto et al., 2022; Kamina et al., 2020).

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*.

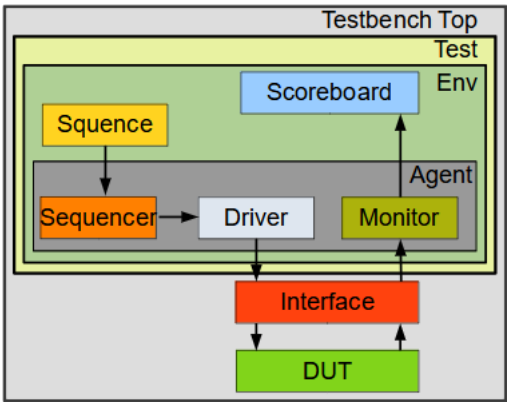
Fitur *constraint* memiliki konstruksi, yaitu: *inside*, *foreach*, *solve before*. Sedangkan fitur jenis-jenis konstruksi memiliki konstruksi, yaitu: *program*, *cast*, *package*. Untuk fitur cakupan fungsi memiliki konstruksi, yaitu: *mode*, *coverpoint*, *covergroup*, *bins*, *iff*, *start*, *stop*.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

2.4. Questasim 2021.1

Questasim 2021.1 merupakan *software* simulasi dan *debugging* yang dapat digunakan untuk memverifikasi desain rangkaian terpadu yang dirancang menggunakan Bahasa SystemVerilog(Tadros et al., 2020). *Software* ini memberikan fasilitas antar muka grafis yang menarik atau GUI (*graphical user interface*).

3. METODE PENELITIAN

Desain UVM testbench yang diajukan untuk memverifikasi rangkaian terpadu multivibrator dari kerusakan hubung singkat yang terjadi di dalamnya bisa dilihat dalam Gambar 5. Testbench tersusun dari beberapa komponen, yaitu: sequence, sequencer, interface, driver, monitor, scoreboard, agent, environment, test, dan testbench top. Sedangkan DUT (Design Under Test) merupakan desain yang akan diuji, dalam hal ini adalah rangkaian terpadu multivibrator.



Gambar 5. Desain testbench

DOI: https://doi.org/10.54082/jupin.994
p-ISSN: 2808-148X
e-ISSN: 2808-1366

Diagram logika DUT sebagaimana ditunjukkan dalam Gambar 6. Kode DUT yang dirancang menggunakan Bahasa SystemVerilog bisa ditunjukkan dalam Gambar 6.

```
module monostable1(
        input clk,
        input reset,
        input trigger,
        output reg pulse = 0
    );
        parameter PULSE_WIDTH = 20;
        reg [4:0] count;
        always @(posedge clk, posedge reset) begin
            if (reset) begin
                pulse <= 1'b0;
            end else if (trigger) begin
                pulse <= 1'b1;
            end else if (count == PULSE_WIDTH-1) begin
                pulse <= 1'b0;
            end
        end
          always @(posedge clk, posedge reset) begin
            if(reset) begin
                count <= 0;
            end else if (pulse) begin
                count <= count + 1'b1;
            end
        end
    endmodule
```

Gambar 6. Kode DUT

UVM package diperlukan dalam mendesain UVM testbench. Kode UVM package bisa dilihat dalam Gambar 7. Dalam Gambar 7, UVM package tersusun dari beberapa konstruksi, yaitu: import dan include.

```
`define LENGTH 1
import uvm_pkg::*;
`include "uvm_macros.svh"
```

Gambar 7. UVM package

Sequence merupakan sinyal kontrol input dan output yang dapat diacak dan akan digerakkan ke DUT. Gambar 8. merupakah kode sequence. Dalam Gambar 8, sequence tersusun dari beberapa konstruksi, yaitu: class, rand bit, dan bit.

```
class Item extends uvm_sequence_item;
  `uvm_object_utils(Item)
  rand bit  trigger;
  bit               pulse;
    virtual function string convert2str();
    return $sformatf("trigger=%0d, pulse=%0d", trigger, pulse);
  endfunction
    function new(string name = "Item");
    super.new(name);
```

DOI: https://doi.org/10.54082/jupin.994
p-ISSN: 2808-148X
e-ISSN: 2808-1366

```
  endfunction
    constraint c1 { trigger dist {0:/1, 1:/1}; }
endclass
```

Gambar 8. Kode Sequence

Sequencer berfungsi untuk menghasilkan berbagai sinyal stimulus input untuk digerakkan ke DUT. Kode sequencer bisa dilihat dalam Gambar 9. Sebagaimana ditunjukkan dalam Gambar 9, sequencer tersusun dari beberapa konstruksi, yaitu: class, function, rand, virtual, for.

```
class gen_item_seq extends uvm_sequence;
  `uvm_object_utils(gen_item_seq)
  function new(string name="gen_item_seq");
    super.new(name);
  endfunction
  rand int num;   // Config total number of items to be sent
  constraint c1 { soft num inside {[1:1]}; }
   virtual task body();
    for (int i = 0; i < num; i ++) begin
          Item m_item = Item::type_id::create("m_item");
          start_item(m_item);
          m_item.randomize();
      `uvm_info("SEQ", $sformatf("Generate new item: %s", m_item.convert2str()), UVM_HIGH)
          finish_item(m_item);
    end
    `uvm_info("SEQ", $sformatf("Done generation of %0d items", num), UVM_LOW)
  endtask
endclass
```

Gambar 9. Kode Sequencer

Sinyal stimulus yang dihasilkan oleh *sequencer* akan digerakkan ke DUT oleh *driver*. Kode *driver* bisa dilihat dalam Gambar 10. *Driver* tersusun dari beberapa konstruksi, yaitu: *class*, *virtual*, *event*, *mailbox*, *task*.

```
class driver extends uvm_driver #(Item);
  `uvm_component_utils(driver)
  function new(string name = "driver", uvm_component parent=null);
    super.new(name, parent);
  endfunction
  virtual des_if vif;
   virtual function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    if (!uvm_config_db#(virtual des_if)::get(this, "", "des_vif", vif))
      `uvm_fatal("DRV", "Could not get vif")
  endfunction
  virtual task run_phase(uvm_phase phase);
    super.run_phase(phase);
    forever begin
      Item m_item;
      `uvm_info("DRV", $sformatf("Wait for item from sequencer"), UVM_HIGH)
      seq_item_port.get_next_item(m_item);
      drive_item(m_item);
      seq_item_port.item_done();
```

DOI: https://doi.org/10.54082/jupin.994
p-ISSN: 2808-148X
e-ISSN: 2808-1366

```
  end
 endtask
 virtual task drive_item(Item m_item);
  @(vif.cb);
   vif.cb.trigger <= m_item.trigger;
 endtask
endclass
```

Gambar 10. 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 11. *Interface* tersusun dari beberapa konstruksi, yaitu: *interface*, *logic*.

```
interface des_if (input bit clk);
        logic reset;
        logic trigger;
        logic pulse;
        clocking cb @(posedge clk);
     default input #1step output #3ns;
                input pulse;
                output trigger;
        endclocking
endinterface
```

Gambar 11. Kode interface

Monitor akan mengamati sinyal port input atau output dari *interface* untuk ditangkap atau disimpan. Contoh *pseudocode monitor* bisa dilihat dalam Gambar 12. *Monitor* tersusun dari beberapa konstruksi, yaitu: *class*, *virtual*, *mailbox*, *task*.

```
class monitor extends uvm_monitor;
 `uvm_component_utils(monitor)
 function new(string name="monitor", uvm_component parent=null);
  super.new(name, parent);
 endfunction
  uvm_analysis_port  #(Item) mon_analysis_port;
 virtual des_if vif;
  virtual function void build_phase(uvm_phase phase);
  super.build_phase(phase);
  if (!uvm_config_db#(virtual des_if)::get(this, "", "des_vif", vif))
   `uvm_fatal("MON", "Could not get vif")
  mon_analysis_port = new ("mon_analysis_port", this);
 endfunction
  virtual task run_phase(uvm_phase phase);
  super.run_phase(phase);
  // This task monitors the interface for a complete
  // transaction and writes into analysis port when complete
  forever begin
   @ (vif.cb);
                        if (vif.reset) begin
                                Item item = Item::type_id::create("item");
                                item.trigger = vif.trigger;
                                item.pulse = vif.cb.pulse;
```

DOI: https://doi.org/10.54082/jupin.994
p-ISSN: 2808-148X
e-ISSN: 2808-1366

```
                                mon_analysis_port.write(item);
        `uvm_info("MON", $sformatf("Saw item %s", item.convert2str()), UVM_HIGH)
                            end
  end
 endtask
endclass
```

Gambar 12. Kode monitor

Scoreboard berfungsi untuk memeriksa apakah sinyal output yang dihasilkan oleh DUT yang tertangkap di *monitor* sudah sesuai dengan yang diharapkan. Kode *scoreboard* bisa dilihat dalam Gambar 13. *Scoreboard* tersusun dari beberapa konstruksi, yaitu: *class, mailbox, task, forever.*

```
class scoreboard extends uvm_scoreboard;
 `uvm_component_utils(scoreboard)
 function new(string name="scoreboard", uvm_component parent=null);
  super.new(name, parent);
 endfunction
        bit[`LENGTH-1:0]      ref_pattern;
        bit[`LENGTH-1:0]      act_pattern;
        bit                           exp_pulse;
 uvm_analysis_imp #(Item, scoreboard) m_analysis_imp;
  virtual function void build_phase(uvm_phase phase);
  super.build_phase(phase);
  m_analysis_imp = new("m_analysis_imp", this);
  if (!uvm_config_db#(bit[`LENGTH-1:0])::get(this, "*", "ref_pattern", ref_pattern))
                        `uvm_fatal("SCBD", "Did not get ref_pattern !")
 endfunction
  virtual function write(Item item);
  act_pattern = act_pattern << 1 | item.trigger;
    `uvm_info("SCBD", $sformatf("trigger=%0d pulse=%0d ref=0b%0b act=0b%0b",
                item.trigger, item.pulse, ref_pattern, act_pattern), UVM_LOW)
    if (item.pulse != exp_pulse) begin
   `uvm_error("SCBD", $sformatf("ERROR ! pulse=%0d exp=%0d",
                                                item.pulse, exp_pulse))
  end else begin
   `uvm_info("SCBD", $sformatf("PASS ! pulse=%0d exp=%0d",
                        item.pulse, exp_pulse), UVM_HIGH)
  end
    if (!(ref_pattern ^ act_pattern)) begin
   `uvm_info("SCBD", $sformatf("Pattern found to match, next out should be 1"), UVM_LOW)
                exp_pulse = 1;
  end else begin
   exp_pulse = 0;
  end
 endfunction
endclass
```

Gambar 13. Kode scoreboard

Environment berisi komponen *agent* dan *scoreboard*. Kode *environment* bisa dilihat dalam Gambar 14. *Environment* tersusun dari beberapa konstruksi, yaitu: *class, agent, scoreboard, virtual, function*.

DOI: https://doi.org/10.54082/jupin.994
p-ISSN: 2808-148X
e-ISSN: 2808-1366

```
class env extends uvm_env;
  `uvm_component_utils(env)
  function new(string name="env", uvm_component parent=null);
    super.new(name, parent);
  endfunction
   agent          a0;              // Agent handle
  scoreboard      sb0;             // Scoreboard handle
      virtual function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    a0 = agent::type_id::create("a0", this);
    sb0 = scoreboard::type_id::create("sb0", this);
  endfunction

  virtual function void connect_phase(uvm_phase phase);
    super.connect_phase(phase);
    a0.m0.mon_analysis_port.connect(sb0.m_analysis_imp);
  endfunction
endclass
```

Gambar 14. Kode *environment*

Test berisi *environment* yang dapat diatur konfigurasinya. Kode *test* bisa dilihat dalam Gambar 15. *Test* tersusun dari beberapa konstruksi, yaitu: *class*, *env*, *function*, *virtual*, dan *task*.

```
class base_test extends uvm_test;
  `uvm_component_utils(base_test)
  function new(string name = "base_test", uvm_component parent=null);
    super.new(name, parent);
  endfunction

  env                             e0;
  bit[`LENGTH-1:0]  pattern = 4'b1011;
  gen_item_seq            seq;
  virtual           des_if   vif;
  virtual function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    e0 = env::type_id::create("e0", this);
     if (!uvm_config_db#(virtual des_if)::get(this, "", "des_vif", vif))
       `uvm_fatal("TEST", "Did not get vif")
    uvm_config_db#(virtual des_if)::set(this, "e0.a0.*", "des_vif", vif);
      uvm_config_db#(bit[`LENGTH-1:0])::set(this, "*", "ref_pattern", pattern);
     seq = gen_item_seq::type_id::create("seq");
    seq.randomize();
  endfunction
  virtual task run_phase(uvm_phase phase);
    phase.raise_objection(this);
    apply_reset();
    seq.start(e0.a0.s0);
    #200;
    phase.drop_objection(this);
  endtask
 virtual task apply_reset();
    vif.reset <= 0;
```

DOI: https://doi.org/10.54082/jupin.994
p-ISSN: 2808-148X
e-ISSN: 2808-1366

```
    vif.trigger <= 0;
    repeat(1) @ (posedge vif.clk);
    vif.reset <=1;
    repeat(1) @ (posedge vif.clk);
    vif.reset <=0;
  endtask
endclass
class test_1011 extends base_test;
  `uvm_component_utils(test_1011)
  function new(string name="test_1011", uvm_component parent=null);
    super.new(name, parent);
  endfunction

  virtual function void build_phase(uvm_phase phase);
    pattern = 4'b1011;
    super.build_phase(phase);
    seq.randomize() with { num inside {[10:10]}; };
  endfunction
endclass
```

Gambar 15. 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 16. *Testbench top* tersusun dari beberapa konstruksi, yaitu: *module, bit, test.*

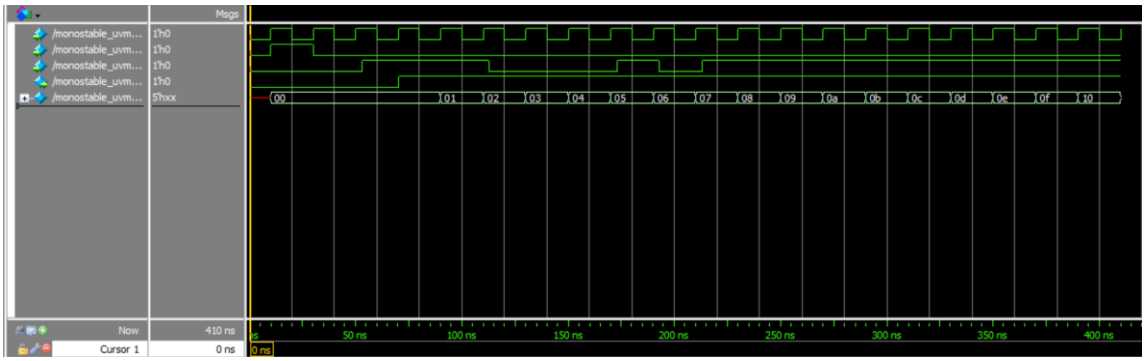
```
module monostable_uvm_tba;
  reg clk;
   always #10 clk =~ clk;
  des_if _if (clk);
      monostable1 u0       ( .clk(clk),
                  .reset(_if.reset),
                  .trigger(_if.trigger),
                  .pulse(_if.pulse));
    initial begin
    clk <= 0;
    uvm_config_db#(virtual des_if)::set(null, "uvm_test_top", "des_vif", _if);
    run_test("test_1011");
  end
endmodule
```

Gambar 16. Kode testbench top

4. HASIL DAN PEMBAHASAN

Kode UVM *testbench* dan DUT dalam Bahasa SystemVerilog kemudian disimulasikan menggunakan software Questasim 2021.1. Hasil simulasi DUT tanpa kerusakan hubung singkat bisa ditunjukkan dalam Gambar 17. Hasil simulasi menujukkan kesesuaian dengan diagram waktu DUT dalam Diagram logika rangkaian multivibrator. Hasil *transcript* dari Questasim juga memberikan keterangan “UVM_ERROR : 0” sebagaimana ditunjukkan dalam Gambar 18.

DOI: https://doi.org/10.54082/jupin.994
p-ISSN: 2808-148X
e-ISSN: 2808-1366



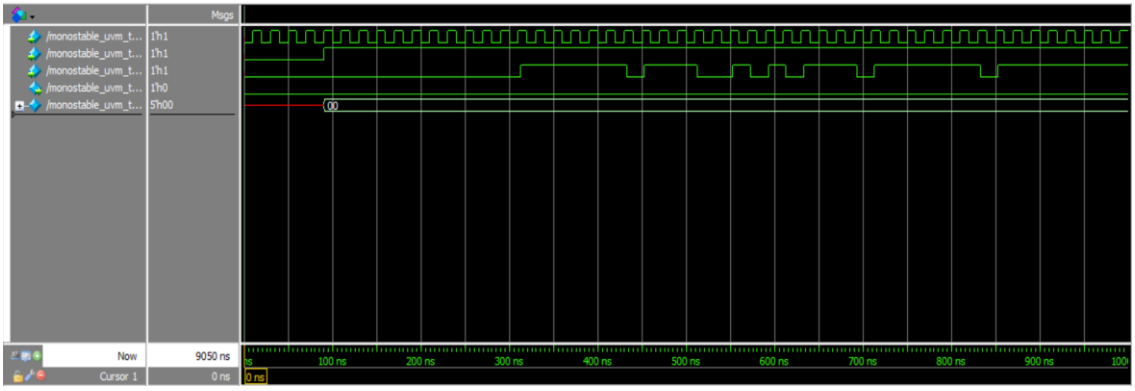
Gambar 17. Hasil simulasi DUT tanpa kerusakan hubung singkat

```
#      (Specify +UVM_NO_RELNOTES to turn off this notice)
#
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(277) @ 0: reporter [Questa UVM] QUESTA_UVM-1.2.3
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(278) @ 0: reporter [Questa UVM]  questa_uvm::init(+struct)
# UVM_INFO @ 0: reporter [RNTST] Running test test_1011...
#      UVM_INFO      C:\questasim64_2021.1\examples\monostable_uvm_tba.sv(149)      @      10: uvm_test_top.e0.sb0 [SCBD] trigger=0 pulse=0 ref=0b1 act=0b0
#      UVM_INFO      C:\questasim64_2021.1\examples\monostable_uvm_tba.sv(44)      @      210: uvm_test_top.e0.a0.s0@@seq [SEQ] Done generation of 10 items
#    UVM_INFO    verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267)    @    410:    reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :    6
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Questa UVM]     2
# [RNTST]     1
# [SCBD]     1
# [SEQ]     1
                                                    # [TEST_DONE]     1
```

Gambar 18. Hasil transcipt DUT tanpa kerusakan hubung singkat

Gambar 19 merupakan hasil simulasi DUT dengan kerusakan hubung singkat bisa ditunjukkan. Hasil simulasi menujukkan ketidakesesuaian dengan diagram waktu DUT dalam Diagram logika rangkaian multivibrator. Gambar 20 adalah hasil *transcript* dari Questasim juga memberikan keterangan "UVM_ERROR : 51".

DOI: https://doi.org/10.54082/jupin.994
p-ISSN: 2808-148X
e-ISSN: 2808-1366



Gambar 19. Hasil simulasi DUT dengan kerusakan hubung singkat

```
#       (Specify +UVM_NO_RELNOTES to turn off this notice)
#
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(277) @ 0: reporter [Questa UVM] QUESTA_UVM-1.2.3
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(278) @ 0: reporter [Questa UVM]  questa_uvm::init(+struct)
# UVM_INFO @ 0: reporter [RNTST] Running test test_1011...
#       UVM_INFO       C:/questasim64_2021.1/examples/monostbale_uvm_tb.sv(148)       @       90: uvm_test_top.e0.sb0 [SCBD] trigger=0 pulse=0 ref=0b1011 act=0b0
…
#       UVM_INFO       C:/questasim64_2021.1/examples/monostbale_uvm_tb.sv(148)       @       510: uvm_test_top.e0.sb0 [SCBD] trigger=1 pulse=0 ref=0b1011 act=0b111
…
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :  504
# UVM_WARNING :    0
# UVM_ERROR :   51
# UVM_FATAL :    0
# ** Report counts by id
# [Questa UVM]     2
# [RNTST]     1
# [SCBD]   550
# [SEQ]     1
# [TEST_DONE]     1
```

Gambar 20. Hasil transcipt DUT dengan kerusakan hubung singkat

Hasil penelitian ini menunjukkan bahwa testbench berbasis Universal Verification Methodology (UVM) yang dikembangkan mampu mendeteksi kerusakan hubung singkat pada rangkaian terpadu multivibrator dengan akurasi tinggi. Simulasi menggunakan Questasim 2021.1 memberikan hasil "UVM_ERROR : 0" ketika tidak ada kerusakan, dan hasil "UVM_ERROR : 51" saat terdapat kerusakan, menunjukkan efektivitas metode ini dalam mengidentifikasi cacat produksi sebelum rangkaian dikirim ke konsumen.

Dibandingkan dengan metode sebelumnya, seperti Automatic Test Pattern Generation (ATPG) yang diusulkan oleh Wang et al. (2019, 2020) dan Wang & Gupta (2020), testbench berbasis UVM lebih ekonomis karena tidak memerlukan alat ATPG yang mahal. Selain itu, dibandingkan dengan metode berbasis Machine Learning yang dikembangkan oleh Higami et al. (2022), metode UVM lebih cepat dalam proses deteksi, karena tidak memerlukan pelatihan model dan parameterisasi yang kompleks.

DOI: https://doi.org/10.54082/jupin.994
p-ISSN: 2808-148X
e-ISSN: 2808-1366

Metode berbasis X-Ray yang diajukan oleh Tebina et al. (2022) juga memiliki keunggulan dalam mendeteksi kerusakan internal, namun biayanya jauh lebih tinggi dibandingkan testbench berbasis UVM ini.

Dibandingkan dengan metode Built-in Self-Test (BIST) yang dikembangkan oleh Widianto dan Robert Lis (2019, 2020), testbench berbasis UVM menawarkan fleksibilitas lebih besar dalam pengujian berbagai rangkaian tanpa keterbatasan ruang dalam desain sirkuit terpadu. Selain itu, dibandingkan dengan testbench berbasis SystemVerilog yang diusulkan oleh Widianto et al. (2022), pendekatan UVM lebih modular dan dapat digunakan untuk berbagai desain rangkaian tanpa harus mengubah struktur dasar testbench secara signifikan.

5. KESIMPULAN

UVM diajukan untuk memverifikasi rangkaian terpadu multivibrator dari kerusakan hubung singkat ke suplai tegangan dan ke ground yang terjadi pada input atau output rangkaian penyusun di dalam rangkaian terpadu tersebut. UVM terdiri dari testbench dan DUT. Testbench tersusun dari beberapa komponen, yaitu: sequence, sequencer, interface, driver, monitor, scoreboard, agent, environment, test, dan testbench top. Sedangkan DUT atau desain yang akan diuji yaitu rangkaian terpadu multivibrator.

Hasil simulasi UVM testbench menggunakan Questasim menunjukkan bahwa sinyal yang dihasilkan oleh DUT sesuai dengan diagram waktu dari rangkaian multivibrator tanpa kerusakan hubung singkat. Hasil *transcript* dari Questasim juga memberikan keterangan "UVM_ERROR : 0".

DAFTAR PUSTAKA

Christian Bakhau, et al. (2020). *Logic Application Handbook, Product Feature and Application Insights, Design Engineer Guide*. Nexperia. https://www.nexperia.com/dam/jcr:851f7c27-b0e9-4627-84b9-13b132388708/Nexperia_LOGIC_Handbook.pdf

Handique, M., Deka, J. K., & Biswas, S. (2021). Detection of Stuck-at and Bridging Fault in Reversible Circuits using an Augmented Circuit. *Proceedings of the Asian Test Symposium*, *2021-November*. https://doi.org/10.1109/ATS52891.2021.00022

Higami, Y., Yamauchi, T., Inamoto, T., Wang, S., Takahashi, H., & Saluja, K. K. (2022). Machine Learning Based Fault Diagnosis for Stuck-at Faults and Bridging Faults. *ITC-CSCC 2022 - 37th International Technical Conference on Circuits/Systems, Computers and Communications*. https://doi.org/10.1109/ITC-CSCC55581.2022.9894966

Ishii, T., & Namba, K. (2022). Stuck-at Fault Tolerance in DNN Using Statistical data. *Proceedings of IEEE Pacific Rim International Symposium on Dependable Computing, PRDC, 2022-November*. https://doi.org/10.1109/PRDC55274.2022.00042

Kamina, Y., Iwai, K., Matsubara, T., & Kurokawa, T. (2020). A Translator from FDL to SystemVerilog for FPGA Implementation of Fuzzy Inference. *Proceedings - 2020 8th International Symposium on Computing and Networking Workshops, CANDARW 2020*. https://doi.org/10.1109/CANDARW51189.2020.00028

Kung, Y. C., Lee, K. J., & Reddy, S. M. (2020). Generating Single-and Double-Pattern Tests for Multiple CMOS Fault Models in One ATPG Run. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *39*(6). https://doi.org/10.1109/TCAD.2019.2921345

Smirnov, K., Nazarov, A., & Blinov, V. (2020). Methods of automated test solutions design for VLSI testing. *Proceedings - 2020 International Conference on Industrial Engineering, Applications and Manufacturing, ICIEAM 2020*. https://doi.org/10.1109/ICIEAM48468.2020.9111875

Tadros, R. N., Fayyazi, A., Pedram, M., & Beerel, P. A. (2020). SystemVerilog Modeling of SFQ and AQFP Circuits. *IEEE Transactions on Applied Superconductivity*, *30*(2). https://doi.org/10.1109/TASC.2019.2957196

Tebina, N. E. O., Zergainoh, N. E., & Maistri, P. (2022). X-Ray Fault Injection: Reviewing Defensive Approaches from a Security Perspective. *Proceedings - IEEE International Symposium on Defect*

DOI: https://doi.org/10.54082/jupin.994
p-ISSN: 2808-148X
e-ISSN: 2808-1366

and Fault Tolerance in VLSI and Nanotechnology Systems, DFT, *2022-October*. https://doi.org/10.1109/DFT56152.2022.9962362

Wang, F., & Gupta, S. K. (2020). An Effective and Efficient Automatic Test Pattern Generation (ATPG) Paradigm for Certifying Performance of RSFQ Circuits. *IEEE Transactions on Applied Superconductivity*, *30*(5). https://doi.org/10.1109/TASC.2020.2965933

Wang, P., Gharehbaghi, A. M., & Fujita, M. (2019). Automatic Test Pattern Generation for Double Stuck-at Faults Based on Test Patterns of Single Faults. *Proceedings - International Symposium on Quality Electronic Design, ISQED*, *2019-March*. https://doi.org/10.1109/ISQED.2019.8697831

Wang, P., Gharehbaghi, A. M., & Fujita, M. (2020). An Automatic Test Pattern Generation Method for Multiple Stuck-At Faults by Incrementally Extending the Test Patterns. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *39*(10). https://doi.org/10.1109/TCAD.2019.2957364

Widianto dan Robert Lis. (2019). Signature Analyzer of Built-in Self Test for Analyzing Stuck-at-Faults in Combinational Logic ICs. *Prosiding Seminar Nasional SENTRA*, 14–17. https://www.academia.edu/85873173/Signature_Analyzer_of_Built_in_Self_Test_for_Analyzing_Stuck_at_Faults_in_Combinational_Logic_Ics

Widianto dan Robert Lis. (2020). A Signature Register of A BIST to Detect Stuck-at-Faults in Combinational Logic ICs. *Prosiding Seminar Nasional SENTRA*, 39–43. https://eprints.umm.ac.id/id/eprint/354

Widianto, M., Chasrun, H., & Lis, R. (2022). Build Testbenches for Verification in Shift Register ICs using SystemVerilog. *International Journal of Electronics and Telecommunications*, *68*(3). https://doi.org/10.24425/ijet.2022.141281