MODUL 6 PROYEK PERANCANGAN SISTEM DIGITAL

Kevin Naoko (13218046)
Asisten: Muhammad Raihan Aziz
Tanggal Percobaan: 29/11/2019
EL2102-Praktikum Sistem Digital
Laboratorium Dasar Teknik Elektro - Sekolah Teknik Elektro dan Informatika ITB

Abstrak

Pada proyek ini telah dibuat sebuah sistem permainan Tic Tac Toe berbasis 2 pemain. Sistem ini memiliki masukkan berupa button_1 untuk memilih kotak penempatan warna dan button_2 untuk mengunci pilihan pemain. Keluaran dari sistem ini ialah VGA yang telah diberi tulisan Tic Tac Toe serta Gameboard dinamik yang dapat berubah sesuai masukan pengguna. Sistem ini memanfaatkan FSM yang telah dirancang sedemikian rupa sehingga dapat dimainkan oleh 2 pemain di FPGA.

Kata kunci: FPGA, FSM, VGA.

1. PENDAHULUAN

Percobaan ini dilakukan untuk mencapai beberapa tujuan. Tujuan yang ingin dicapai ialah, menspesifikasikan suatu sistem digital sederhana, membagi sistem menjadi satu atau lebih jalur data dan kendali, mendesain jalur data dan kendali untuk sistem, mengintegrasikan jalur data dan kendali ke dalam sistem, melakukan tes menyeluruh terhadap sistem, meng-implementasikan sistem digital menggunakan FPGA dan terakhir, menguji dan menganalisa sistem yang sudah dibangun. Untuk mencapai tujuan tujuan tersebut, praktikan diminta untuk membuat sebuah sistem yang menggunakan VHDL dengan persyaratan, menggunakan interface yang digunakan board DE1, mempunyai bagian FSM, dan sedikitnya terdiri dari tiga blok.

Sistem yang dibuat oleh praktikan adalah sebuah permainan retro yang bernama tic-tac-toe. Perbedaan dari tic-tac-toe pada umumnya adalah, digunakannya bidak warna sebagai pengganti bidak X dan O. pemain 1 akan menggunakan warna hijau dan pemain 2 akan menggunakan warna merah. Kondisi menang akan terjadi apabila warna yang sama terurut sepanjang tiga kotak horizontal atau vertical maupun diagonal.

2. STUDI PUSTAKA

2.1 FINITE-STATE MACHINE (FSM)

FSM ialah sistem digital yang menggunakan pendekatan logika sekuensial. Karena prinsip dasarnya menggunakan prinsip sekuensial, maka FSM memiliki kondisi-kondisi yang dijalankan dengan urutan tertentu [1]. Kondisi ini disebut sebagai state. Untuk mempermudah perancangan FSM, pertama tama perlu didesain sebuah state diagram yang menunjukkan alur FSM dan kondisi kondisi yang harus terpenuhi untuk lanjut ke state berikutnya. Pada perancangan sistem ini, kami menggunakan state diagram model Moore.

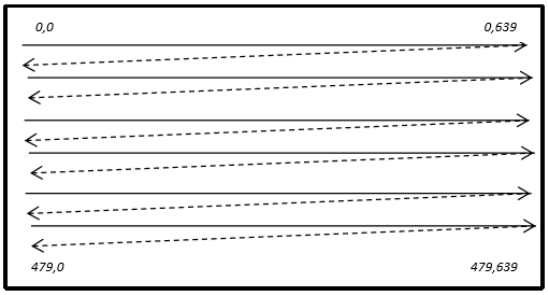
2.2 FIELD-PROGRAMMABLE GATE ARRAY (FPGA)

FPGA merupakan sirkuit terintegrasi yang terdiri atas blok blok logic yang dapat diatur oleh pengguna menggunakan bahasa VHDL. Pemanfaatan FPGA untuk pembuatan sistem sistem sederhana sangat umum digunakan oleh banyak orang. Pada proyek ini, kami akan memanfaatkan FPGA tipe DE1 untuk membuat game sederhana [2].

2.3 VIDEO GRAPHICS ARRAY (VGA)

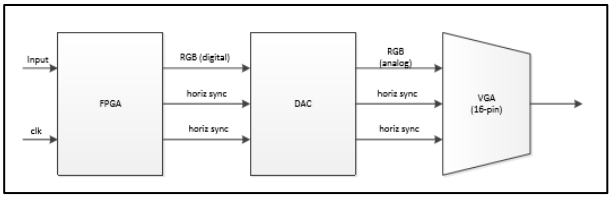
Pada mulanya, standar VGA dikembangkan oleh IBM dengan resolusi standar 640x480 pixel [3]. Dalam VHDL, *interface* ke VGA menggunakan 2 jenis sinyal yaitu sinyal warna (RGB) dan sinyal sinkron (HS dan VS). Sinyal sinkron tersebut menentukan pergantian baris (Horizontal Sync) dan pergantian layar (Vertical Sync). Sedangkan sinyal RGB menentukan warna yang ditampilkan oleh tiap pixel. Warna warna ini perlu dideklarasikan batasnya dengan fungsi if pada blok color rom.

Pada perancangan sistem ini, digunakan resolusi standar dan *refresh rate* 60Hz agar tidak terlihat *flickering* pada layar. Proses *scan* pixel berawal dari kiri atas ke kanan lalu ke kiri bawah dan kembali ke kiri atas ketika sudah mencapai pixel terakhir.

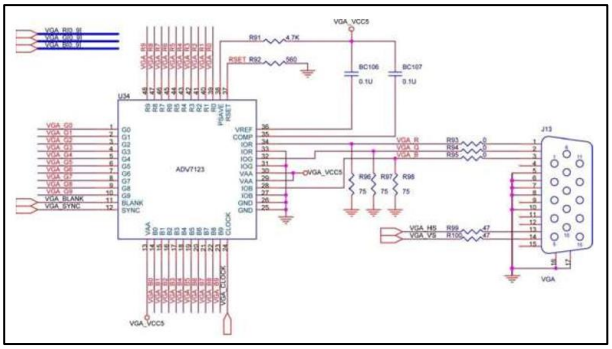


Gambar 2-1. Razor Scan Pada Layar LCD

Sinyal VGA merupakan sinyal analog. Maka dari itu diperlukan chip DAC untuk mengkonversi sinyal digital ke analog. Datapath dan skematik dari chip tersebut dapat dilihat dibawah ini [4].



Gambar 2-2. *Datapath* DAC [5]



Gambar 2-3. Skematik DAC [6]

3. METODOLOGI

3.1 KOMPONEN DAN ALAT

Komponen dan alat yang digunakan dalam praktikum ini adalah sebagai berikut.

1. Board FPGA tipe DE1
2. Catu daya + kabel dan konektor tambahan serta kabel downloader
3. Computer yang dilengkapi dengan software Quartus dan Modelsim
4. Layer LCD
5. Kabel VGA

3.2 SPESIFIKASI SISTEM

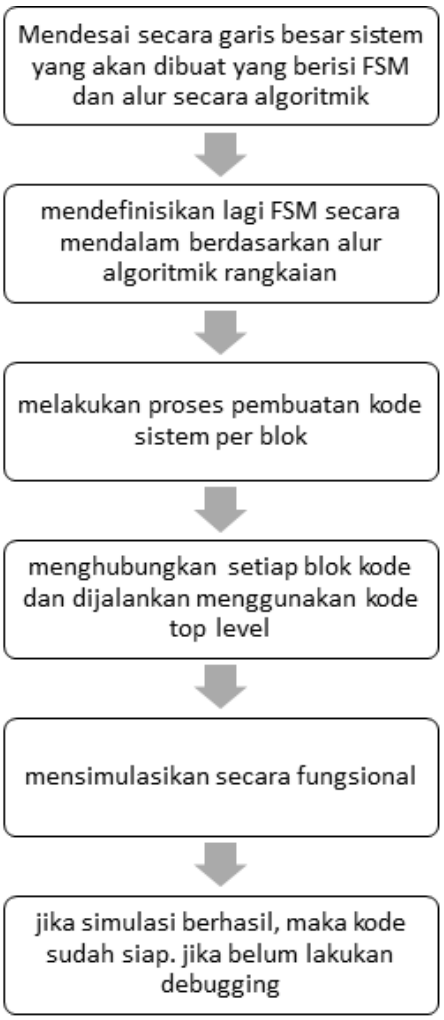
Sistem ini memiliki tiga input yang diantaranya, 2 button dan satu switch. Button yang pertama berfungsi untuk memilih kotak yang warna nya akan dimasukkan. Button kedua berfungsi untuk meng-assign warna pemain. Sebuah switch untuk me-reset permainan saat permainan sudah ditentukan pemenangnya.

3.3 FITUR SISTEM

Fitur dari sistem ini adalah pemain 1 akan menggunakan warna hijau selama keseluruhan permainan, pemain 2 akan menggunkan warna merah selama keseluruhan permainan. Fitur lain adalah, terdapat selector yang berguna untuk mengetahui posisi kotak yang akan kita pilih untuk dimasukkan warnanya dan saat permainan selesai akan ditunjukkan pemenang dengan warna yang ada dibawah kotak. Apabila pemain 1 menang, akan ditampilkan warna hijau dan apabila pemain 2 yang menang akan ditunjukkan warna merah. Saat terjadi hasil yang seri, akan ditunjukkan warna abu- abu. Ssebagai tambahan, ditampilkan tulisan yang menambah estetika yang bertuliskan "TIC", "TAC", "TOE" dan grid tipis berwarna putih sebagai bingkai dari permainan.

3.4 LANGKAH KERJA

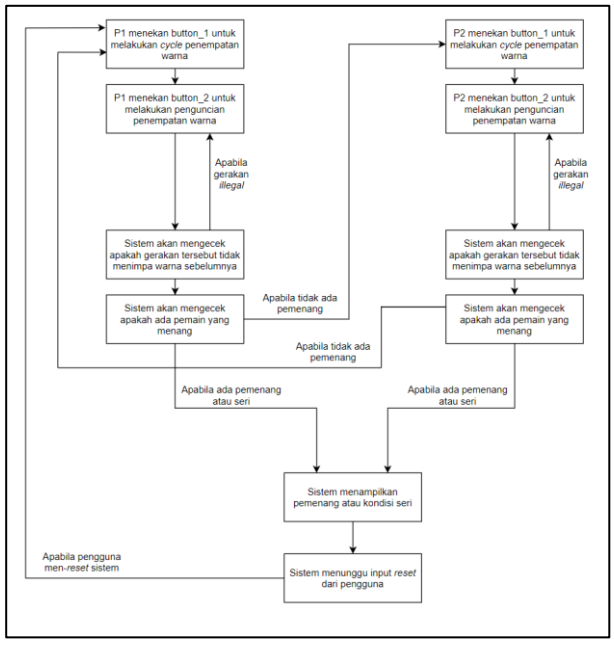
3.4.1 PERANCANGAN SISTEM TIC TAC TOE



3.4.2 PENGUJIAN SISTEM

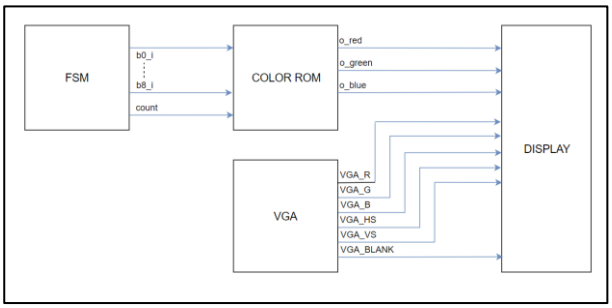
4. HASIL DAN ANALISIS

4.1 ALUR KERJA SISTEM

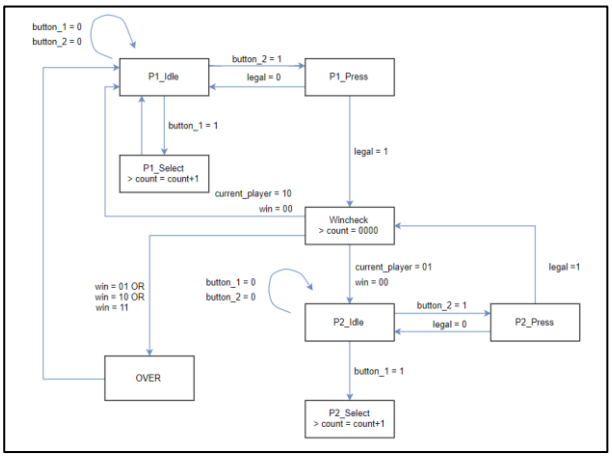


Gambar 4-1. Alur Kerja Sistem Secara Umum

4.2 DESAIN SISTEM PRA-PEMROGRAMAN



Gambar 4-2. *Datapath* Sistem

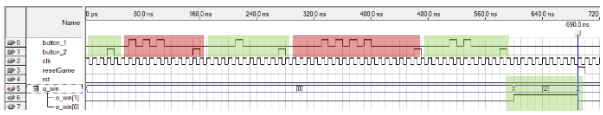


Gambar 4-3. FSM Sistem

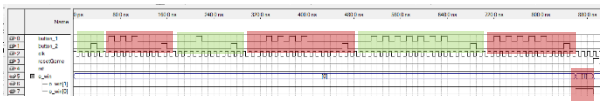
FSM pada gambar 4-3 merupakan pengembangan dari FSM yang diajukan pada proposal proyek ini, karena dengan adanya state-state tersebut, dapat mempermudah realisasi sistem ini.

4.3 HASIL PENGUJIAN

4.3.1 DOKUMENTASI SIMULASI DAN IMPLEMENTASI

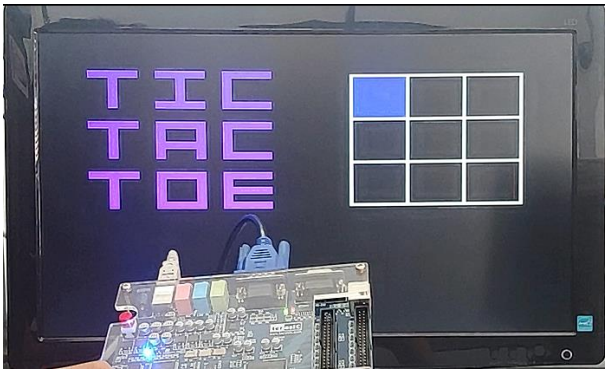


Gambar 4-4. Simulasi waveform saat P1 Menang



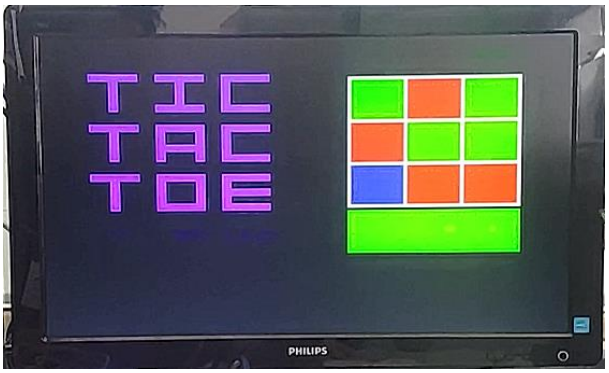
Gambar 4-5. Simulasi waveform saat P2 Menang

Gambar 4-6. Simulasi waveform saat Kondisi Seri

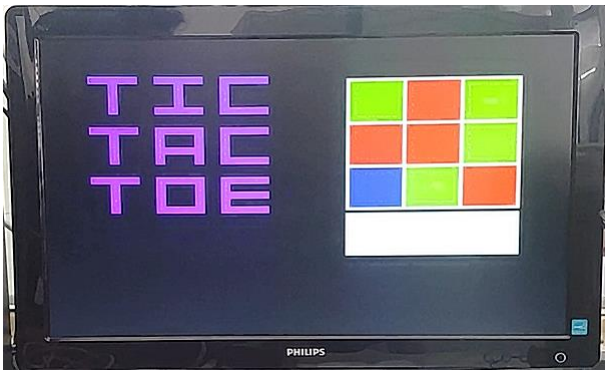


Gambar 4-7. Tampilan Pada Awal Permainan

Gambar 4-8. Tampilan LCD Saat P1 Menang



Gambar 4-9. Tampilan LCD Saat P2 Menang

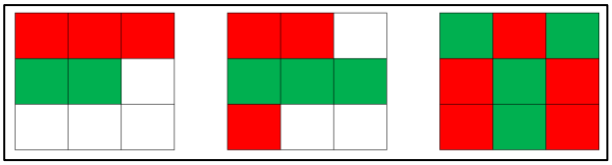


Gambar 4-10. Tampilan LCD Saat Seri

4.3.2 KEBERJALANAN PENGUJIAN

Setelah melakukan *debugging*, semua fitur yang direncanakan pada sistem ini dapat berjalan sesuai ekspektasi. Berikut fitur fitur yang kami uji

- Simulasi *waveform* pada FSM untuk kondisi P1 menang, P2 menang, dan kondisi seri, seperti yang dapat dilihat pada gambar 4-4 hingga 4-6. Untuk mempermudah pembacaan, kami menandakan gerakan yang dilakukan oleh P1 dengan warna hijau, dan P2 dengan warna merah. Dibawah ini juga akan ditampilkan visualisasi hasil simulasi ketiga kondisi tersebut



Gambar 4-11a (Kiri). Pengujian kondisi P2 menang
Gambar 4-11b (Tengah). Pengujian kondisi P1 menang
Gambar 4-11c (Kanan). Pengujian kondisi Seri

- Tampilan awal permainan, seperti yang dapat dilihat pada gambar 4-4.
- *Cycle* penempatan warna tiap pemain. Apabila dilakukan *cycling* hingga papan terakhir, maka *selector* akan kembali ke papan awal. Pengujian ini dapat dilihat lebih jelas pada video kami.
- Penguncian penempatan warna tiap pemain.
  Apabila dilakukan penguncian pada papan yang telah ditempati sebelumnya, maka pemain yang bersangkutan diharuskan untuk memilih kembali tempat peletakkan warnanya. Pengujian ini dapat dilihat lebih jelas pada video kami.
- Kondisi akhir permainan. Sesuai pada gambar 4-5 hingga 4-7, tampilan pemenang atau kondisi seri akan ditampilkan pada bagian bawah papan permainan.
- *Reset* papan permainan dapat dilakukan dengan menggunakan switch pada FPGA.

4.3.3 ANALISIS SISTEM

Blok FSM

Pada sistem ini digunakan FSM yang ditampilkan pada gambar 4-3. Sebelum masuk ke pembahasan ini, perlu diketahui bahwa sistem ini melakukan *assignment* bit unik berupa ”10” untuk P1 dan ”01” untuk P2.

Alur pada FSM ini dimulai oleh ’P1_Idle’ yang menunggu masukkan oleh P1 berupa button_1 atau button_2. Apabila button_1 ditekan, maka sistem akan melakukan *cycling* penempatan warna pada papan bermain. Setelah P1 yakin akan

pilihannya, maka P1 dapat menekan button_2 untuk mengunci pilihannya. Pilihan tersebut akan disimpan ke variabel tertentu (o_b0...0_b8) sesuai dengan pilihan pemain.

1 (o_b0)	2 (o_b1)	3 (o_b2)
4 (o_b3)	5 (o_b4)	6 (o_b5)
7 (o_b6)	8 (o_b7)	9 (o_b8)

Gambar 4-12. Visualisasi Assignment Variabel o_b0...o_b8 Pada Papan Permainan

Setelah itu, sistem akan menuju state 'P1_Press'. Pada state ini, dilakukan pengecekan apakah P1 mengisi papan yang kosong. Namun karena ini adalah giliran pertama, sudah pasti semua papan kosong. Pada state ini, nilai o_b(x) akan berubah sesuai dengan bit unik P1, pengosongan variabel 'count', dan *assignment* untuk variabel 'current_player' akan menjadi bit unik P2.

Setelah pengecekan dan *assignment* bit unik tersebut, state akan berpindah ke 'Wincheck', yang berisi pengecekan kondisi menang. State ini akan mengecek semua kemungkinan kondisi menang pada papan sesuai dengan bit unik yang dialokasikan pada tiap variabel (Kemungkinan yang diuji ialah nilai bit unik yang terdapat pada kotak [1,2,3], [4,5,6], [7,8,9], [1,4,7], [2,5,8], [3,6,9], [1,5,8], dan [3,5,7]). Apabila belum ada pemain yang menang, maka alur FSM akan ditentukan oleh bit unik yang tersimpan pada variabel 'current_player'. Bila variabel tersebut menyimpan bit P2, maka alur akan berpindah menuju state 'P2_Idle.

Semua fungsi pada state ini dapat dikatakan sama persis dengan state 'P1_idle'. Alur dari state ini juga sama seperti 'P1_idle'. Namun yang menjadi pembeda disini ialah *assignment* bit unik pada tiap proses yang telah dijabarkan sebelumnya akan menggunakan bit unik P2 (kecuali variabel 'current_player' karena variabel tersebut merupakan penanda giliran selanjutnya. Variabel ini akan menggunakan bit P1).

Apabila pengecekan pada state 'Wincheck' mendapatkan bahwa P1 menang, P2 menang, atau seri, maka bit unik P1 dan P2 akan disimpan kedalam variabel 'win'. Bit unik untuk kondisi seri adalah "11". Apabila isi dari variabel 'win' ini ialah salah satu dari ketiga bit unik tersebut, maka alur FSM akan menuju state 'OVER'

Pada state ini hanya terdapat 1 fungsi, yaitu menunggu input pengguna untuk men-*reset* papan permainan. Apabila switch diaktifkan, maka variabel 'ResetGame' akan menjadi 1, dan alur FSM kembali ke 'P1_idle'.

Blok Color ROM

Beberapa variabel dari FSM akan digunakan sebagai input dari Color ROM agar hasil implementasi dapat ditampilkan pada display LCD. Beberapa variabel tersebut antara lain

- o_b0...o_b8 yang dinyatakan sebagai b0_i...b8_i
- count yang dinyatakan sebagai count_i
- win yang dinyatakan sebagain win_i
- ResetGame yang dinyatakan sebagai ResetGame_i

Variabel diatas sangat membantu dalam proses penampilan warna warna untuk P1 dan P2 pada papan bermain, serta kondisi menang atau seri yang ditampilkan pada bagian bawah papan bermain.

Sedangkan pada bagian independen pada blok ini, terdapat sintesis tulisan statik "TIC TAC TOE" berwarna ungu pada bagian kiri layar, serta sintesis grid papan bermain.

Pada bagian deklarasi warna pada blok ini, akan dicek variabel 'o_b(x)'. Apabila variabel tersebut bernilai "00", maka kotak ke-x (penomoran dapat dilihat kembali pada gambar 4-12) pada papan akan berwarna hitam. Apabila variabel tersebut bernilai kode unik P1 atau P2, maka kotak yang bersangkutan akan berwarna sesuai dengan warna P1 atau P2.

Dan akan dicek pula variabel 'count'. Bila nilai integer dari variabel ini cocok dengan nilai x pada o_b(x), maka warna pada kotak tersebut akan menjadi biru, terlepas dari bit unik P1, P2, atau warna hitam. Warna biru ini merupakan *selector* yang didesain untuk mempermudah pemilihan tempat peletakkan warna bagi P1 atau P2.

5. KESIMPULAN

1. Perancangan Proyek *Tic Tac Toe* berhasil dilakukan dengan baik dan sesuai ekspektasi
2. Pemahaman praktikan mengenai konsep konsep yang digunakan pada proyek ini semakin baik, terkhususnya pada pembuatan FSM dan sintesis warna pada display LCD.

DAFTAR PUSTAKA

[1] https://www.allaboutcircuits.com/technical-articles/implementing-a-finite-state-machine-in-vhdl/, Diakses pada 3/12/2019, 03.14

[2] https://techterms.com/definition/fpga, Diakses pada 3/12/2019, 03.44

[3] https://techterms.com/definition/vga, Diakses pada 3/12/2019, 03.48

[4] Mervin T. Hutabarat, dkk. *Petunjuk Praktikum Sistem Digital,* ITB, Bandung, 2019.

[5] Mervin T. Hutabarat, dkk. *Petunjuk Praktikum Sistem Digital,* ITB, Bandung, 2019.

[6] Mervin T. Hutabarat, dkk. *Petunjuk Praktikum Sistem Digital,* ITB, Bandung, 2019.

LAMPIRAN

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity tictactoe is
  port (
    button_1 : in std_logic;
    button_2 : in std_logic;
    resetGame  : in std_logic;
    o_b0, o_b1, o_b2, o_b3, o_b4, o_b5, o_b6, o_b7, o_b8,
o_win  : out std_logic_vector (1 downto 0);
    o_count, o_countTurn                          : out
std_logic_vector (3 downto 0);
    rst, clk : in std_logic
  );
end entity;


architecture FSM of tictactoe is
--P1 and P2 denoted as 10 and 01 respectively

type states is (P1_idle, P1_select, P1_press, P2_idle,
P2_select, P2_press, winCheck, OVER);
signal cstate                  : states;
signal count, countTurn           : std_logic_vector (3
downto 0) := "0000";
signal b0, b1, b2, b3, b4, b5, b6, b7, b8  :
std_logic_vector (1 downto 0) := "00";
signal win                     : std_logic_vector (1 downto
0) := "00";
signal current_player             : std_logic_vector (1
downto 0) := "00";
--component CLOCKDIV is
--port(
--CLK        : IN std_logic;
--DIVOUT     : buffer std_logic
--);
--end component;

begin
--detik : CLOCKDIV port map (CLK => clk, DIVOUT => clk_i);

  process (rst, clk)
  begin
  if rising_edge(clk) then
  case cstate is
    when P1_idle =>       -- Menunggu input P1
      if button_1 = '1' then cstate <= P1_select;
      elsif button_2 = '1' then cstate <= P1_press;
      else cstate <= P1_idle;
      end if;
    when P1_select =>     -- Menambah count apabila
button_1 ditekan untuk cycle posisi penempatan bidak
      if (count < "1000") then
        count <= count + 1;
        cstate <= P1_idle;
      else count <= "0000"; cstate <= P1_idle;
      end if;

    when P1_press =>    -- Meletakkan bidak pada papan
apabila button_2 ditekan, lalu mengarahkan ke state
pengecekan kemenangan serta mengassign current_player
menjadi P2
      if (count = 0 and b0 = "00") then b0 <= "10";
cstate <= winCheck; current_player <= "01"; countTurn <=
countTurn + 1;
      elsif (count = 1 and b1 = "00") then b1 <= "10";
cstate <= winCheck; current_player <= "01"; countTurn <=
countTurn + 1;
      elsif (count = 2 and b2 = "00") then b2 <= "10";
cstate <= winCheck; current_player <= "01"; countTurn <=
countTurn + 1;
      elsif (count = 3 and b3 = "00") then b3 <= "10";
cstate <= winCheck; current_player <= "01"; countTurn <=
countTurn + 1;
      elsif (count = 4 and b4 = "00") then b4 <= "10";
cstate <= winCheck; current_player <= "01"; countTurn <=
countTurn + 1;
      elsif (count = 5 and b5 = "00") then b5 <= "10";
cstate <= winCheck; current_player <= "01"; countTurn <=
countTurn + 1;
      elsif (count = 6 and b6 = "00") then b6 <= "10";
cstate <= winCheck; current_player <= "01"; countTurn <=
countTurn + 1;
      elsif (count = 7 and b7 = "00") then b7 <= "10";
cstate <= winCheck; current_player <= "01"; countTurn <=
countTurn + 1;
      elsif (count = 8 and b8 = "00") then b8 <= "10";
cstate <= winCheck; current_player <= "01"; countTurn <=
countTurn + 1;
      else count <= "0000" ; cstate <= P1_select;
      end if;

    when P2_idle =>       -- Menunggu input P2
      if button_1 = '1' then cstate <= P2_select;
      elsif button_2 = '1' then cstate <= P2_press;
      else cstate <= P2_idle;
      end if;
    when P2_select =>     -- Menambah count apabila
button_1 ditekan untuk cycle posisi penempatan bidak
      if (count < "1000") then
        count <= count + 1;
        cstate <= P2_idle;
      else count <= "0000"; cstate <= P2_idle;
      end if;

    when P2_press =>    -- Meletakkan bidak pada papan
apabila button_2 ditekan, lalu mengarahkan ke state
pengecekan kemenangan serta mengassign current_player
menjadi P1
      if (count = 0 and b0 = "00") then b0 <= "01";
cstate <= winCheck; current_player <= "10"; countTurn <=
countTurn + 1;
      elsif (count = 1 and b1 = "00") then b1 <= "01";
cstate <= winCheck; current_player <= "10"; countTurn <=
countTurn + 1;
      elsif (count = 2 and b2 = "00") then b2 <= "01";
cstate <= winCheck; current_player <= "10"; countTurn <=
countTurn + 1;
      elsif (count = 3 and b3 = "00") then b3 <= "01";
cstate <= winCheck; current_player <= "10"; countTurn <=
countTurn + 1;
      elsif (count = 4 and b4 = "00") then b4 <= "01";
cstate <= winCheck; current_player <= "10"; countTurn <=
countTurn + 1;
      elsif (count = 5 and b5 = "00") then b5 <= "01";
cstate <= winCheck; current_player <= "10"; countTurn <=
countTurn + 1;
      elsif (count = 6 and b6 = "00") then b6 <= "01";
cstate <= winCheck; current_player <= "10"; countTurn <=
countTurn + 1;
      elsif (count = 7 and b7 = "00") then b7 <= "01";
cstate <= winCheck; current_player <= "10"; countTurn <=
countTurn + 1;
      elsif (count = 8 and b8 = "00") then b8 <= "01";
cstate <= winCheck; current_player <= "10"; countTurn <=
countTurn + 1;
      else count <= "0000" ; cstate <= P2_select;
      end if;

    when winCheck =>    -- Pengecekan kemenangan
      if     (b0 = "10" and b1 = "10" and b2 = "10") or
(b3 = "10" and b4 = "10" and b5 = "10") or
          (b6 = "10" and b7 = "10" and b8 = "10") or (b0
= "10" and b3 = "10" and b6 = "10") or
          (b1 = "10" and b4 = "10" and b7 = "10") or (b2
= "10" and b5 = "10" and b8 = "10") or
          (b0 = "10" and b4 = "10" and b8 = "10") or (b2
= "10" and b4 = "10" and b6 = "10") then
          win <= "10"; cstate <= OVER; --P1 WINS
      elsif  (b0 = "01" and b1 = "01" and b2 = "01") or
(b3 = "01" and b4 = "01" and b5 = "01") or
          (b6 = "01" and b7 = "01" and b8 = "01") or (b0
= "01" and b3 = "01" and b6 = "01") or
          (b1 = "01" and b4 = "01" and b7 = "01") or (b2
= "01" and b5 = "01" and b8 = "01") or
          (b0 = "01" and b4 = "01" and b8 = "01") or (b2
= "01" and b4 = "01" and b6 = "01") then
          win <= "01"; cstate <= OVER; --P2 WINS
      elsif   (countTurn > "1000") then
          win <= "11"; cstate <= OVER; --TIE
      else
        if current_player <= "01" then count <= "0000";
cstate <= P2_idle;
        elsif current_player <= "10" then count <=
"0000"; cstate <= P1_idle;
        end if;
      end if;

    when OVER =>     -- Menunggu input apakah ingin
bermain lagi atau tidak
      if (resetGame = '1') then
        win <= "00";
        countTurn <= "0000";
        count <= "0000";
        b0 <= "00";
        b1 <= "00";
        b2 <= "00";
        b3 <= "00";
        b4 <= "00";
        b5 <= "00";
        b6 <= "00";
        b7 <= "00";
        b8 <= "00";
        cstate <= P1_idle;
      end if;
  end case;
  end if;
  end process;

  o_b0 <= b0;
  o_b1 <= b1;
  o_b2 <= b2;
```

```
  o_b3 <= b3;
  o_b4 <= b4;
  o_b5 <= b5;
  o_b6 <= b6;
  o_b7 <= b7;
  o_b8 <= b8;
  o_win <= win;
  o_count <= count;
  o_countTurn <= countTurn;

end FSM;
```

Lampiran 1. Kode Blok FSM Permainan Tic Tac Toe

```
LIBRARY  IEEE;
USE  IEEE.STD_LOGIC_1164.ALL;
USE  IEEE.STD_LOGIC_ARITH.ALL;
USE  IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY color_rom_vhd  IS
  PORT(
    Button_1_i   : IN STD_LOGIC;
    Button_2_i   : IN STD_LOGIC;
    ResetGame_i    : IN STD_LOGIC;
    ResetAll_i   : IN STD_LOGIC;
    B0_i     : IN STD_LOGIC_VECTOR (1 downto 0);
    B1_i     : IN STD_LOGIC_VECTOR (1 downto 0);
    B2_i     : IN STD_LOGIC_VECTOR (1 downto 0);
    B3_i     : IN STD_LOGIC_VECTOR (1 downto 0);
    B4_i     : IN STD_LOGIC_VECTOR (1 downto 0);
    B5_i     : IN STD_LOGIC_VECTOR (1 downto 0);
    B6_i     : IN STD_LOGIC_VECTOR (1 downto 0);
    B7_i     : IN STD_LOGIC_VECTOR (1 downto 0);
    B8_i     : IN STD_LOGIC_VECTOR (1 downto 0);
    Win_i          : IN STD_LOGIC_VECTOR (1 downto 0);
    count_i      : IN STD_LOGIC_VECTOR (3 downto 0);
      i_pixel_column  : IN STD_LOGIC_VECTOR( 9 DOWNTO 0
);
      i_pixel_row     : IN STD_LOGIC_VECTOR( 9 DOWNTO 0
);
      o_red           : OUT STD_LOGIC_VECTOR( 7 DOWNTO 0
);
      o_green         : OUT STD_LOGIC_VECTOR( 7 DOWNTO 0
);
      o_blue          : OUT STD_LOGIC_VECTOR( 7 DOWNTO 0
);
      ledtest      : OUT STD_LOGIC);
END color_rom_vhd;

ARCHITECTURE behavioral OF color_rom_vhd  IS
--GAMEBOARD [DYNAMIC]
CONSTANT R0   : INTEGER := 65;
CONSTANT R1   : INTEGER := 68;
CONSTANT R2   : INTEGER := 134;
CONSTANT R3   : INTEGER := 137;
CONSTANT R4   : INTEGER := 203;
CONSTANT R5   : INTEGER := 206;
CONSTANT R6   : INTEGER := 272;
CONSTANT R7   : INTEGER := 275;

CONSTANT R7b  : INTEGER := 280;
CONSTANT R8    : INTEGER := 350;

CONSTANT C0   : INTEGER := 375;
CONSTANT C1   : INTEGER := 378;
CONSTANT C2   : INTEGER := 444;
CONSTANT C3   : INTEGER := 447;
CONSTANT C4   : INTEGER := 513;
CONSTANT C5   : INTEGER := 516;
CONSTANT C6   : INTEGER := 582;
CONSTANT C7   : INTEGER := 585;

SIGNAL b0_T                 :  STD_LOGIC;
SIGNAL b1_T                 :  STD_LOGIC;
SIGNAL b2_T                 :  STD_LOGIC;
SIGNAL b3_T                 :  STD_LOGIC;
SIGNAL b4_T                 :  STD_LOGIC;
SIGNAL b5_T                 :  STD_LOGIC;
SIGNAL b6_T                 :  STD_LOGIC;
SIGNAL b7_T                 :  STD_LOGIC;
SIGNAL b8_T                 :  STD_LOGIC;
signal win_T        :  STD_LOGIC;
SIGNAL grid                 :  STD_LOGIC;
SIGNAL selector_T    : STD_LOGIC;

--COSMETICS [STATIC]
-- "TIC
--  TAC
--  TOE"
CONSTANT C0a     : INTEGER := 60;
CONSTANT C1a     : INTEGER := 84;
CONSTANT C2a     : INTEGER := 96;
CONSTANT C3a     : INTEGER := 120;
CONSTANT C4a     : INTEGER := 140;
```

```
CONSTANT C5a     : INTEGER := 152;
CONSTANT C6a     : INTEGER := 164;
CONSTANT C7a     : INTEGER := 176;
CONSTANT C8a     : INTEGER := 188;
CONSTANT C9a    : INTEGER := 200;
CONSTANT C10a    : INTEGER := 220;
CONSTANT C11a  : INTEGER := 232;
CONSTANT C12a   : INTEGER := 280;

CONSTANT R0a     : INTEGER := 60;
CONSTANT R1a     : INTEGER := 72;
CONSTANT R2a     : INTEGER := 108;
CONSTANT R3a     : INTEGER := 120;
CONSTANT R4a     : INTEGER := 140;
CONSTANT R5a     : INTEGER := 152;
CONSTANT R6a     : INTEGER := 172;
CONSTANT R7a     : INTEGER := 184;
CONSTANT R8a     : INTEGER := 188;
CONSTANT R9a     : INTEGER := 200;
CONSTANT R10a    : INTEGER := 220;
CONSTANT R11a  : INTEGER := 232;
CONSTANT R12a   : INTEGER := 244;
CONSTANT R13a   : INTEGER := 256;
CONSTANT R14a   : INTEGER := 268;
CONSTANT R15a   : INTEGER := 280;

Signal titleColor1 : STD_LOGIC;
Signal titleColor2 : STD_LOGIC;
Signal titleColor3 : STD_LOGIC;

-- "WINS!"



BEGIN


PROCESS(i_pixel_row,i_pixel_column, B0_i, B1_i, B2_i,
B3_i, B4_i, B5_i, B6_i, B7_i, B8_i, b0_T,b1_T, b2_T, b3_T,
b4_T,b5_T, b6_T,b7_T, b8_T, win_i, Win_T, grid)
BEGIN

--GAMEBOARD
--COLOR PARAMETERS
  --BOXES
  IF ((i_pixel_row > R1)  AND (i_pixel_row < R2)) AND
((i_pixel_column > C1)  AND (i_pixel_column < C2)) THEN
b0_T <=  '1'; --kotak 0
  ELSE  b0_T <=  '0';
  END IF;

  IF ((i_pixel_row > R1)  AND (i_pixel_row < R2)) AND
((i_pixel_column > C3)  AND (i_pixel_column < C4)) THEN
b1_T <=  '1'; -- kotak 1
  ELSE b1_T <=  '0';
  END IF;

  IF ((i_pixel_row > R1)  AND (i_pixel_row < R2)) AND
((i_pixel_column > C5)  AND (i_pixel_column < C6)) THEN
b2_T <=  '1'; --kotak 2
  ELSE b2_T <=  '0';
  END IF;

  IF ((i_pixel_row > R3)  AND (i_pixel_row < R4)) AND
((i_pixel_column > C1)  AND (i_pixel_column < C2)) THEN
b3_T <=  '1'; --kotak 3
  ELSE b3_T <=  '0';
  END IF;

  IF ((i_pixel_row > R3)  AND (i_pixel_row < R4)) AND
((i_pixel_column > C3)  AND (i_pixel_column < C4)) THEN
b4_T <=  '1'; -- kotak 4
  ELSE b4_T <=  '0';
  END IF;

  IF ((i_pixel_row > R3)  AND (i_pixel_row < R4)) AND
((i_pixel_column > C5)  AND (i_pixel_column < C6)) THEN
b5_T <=  '1'; -- kotak 5
  ELSE b5_T <=  '0';
  END IF;

  IF ((i_pixel_row > R5)  AND (i_pixel_row < R6)) AND
((i_pixel_column > C1)  AND (i_pixel_column < C2)) THEN
b6_T <=  '1'; -- kotak 6
  ELSE b6_T <=  '0';
  END IF;

  IF ((i_pixel_row > R5)  AND (i_pixel_row < R6)) AND
((i_pixel_column > C3)  AND (i_pixel_column < C4)) THEN
b7_T <=  '1'; -- kotak 7
  ELSE b7_T <=  '0';
  END IF;

  IF ((i_pixel_row > R5)  AND (i_pixel_row < R6)) AND
((i_pixel_column > C5)  AND (i_pixel_column < C6)) THEN
b8_T <=  '1'; -- kotak 8
  ELSE b8_T <=  '0';
  END IF;
```

```
  IF ((i_pixel_row > R7b)  AND (i_pixel_row <= R8)) AND
((i_pixel_column >= C0)  AND (i_pixel_column <= C7)  )
THEN win_T <=  '1'; -- pemenang
  ELSE win_i <=  '0';
  END IF;


  --GRID
  IF  (((i_pixel_row >= R0) AND (i_pixel_row <= R1)) AND
((i_pixel_column >= C0) AND (i_pixel_column <= C7)))
  OR  (((i_pixel_row >= R2) AND (i_pixel_row <= R3)) AND
((i_pixel_column >= C0) AND (i_pixel_column <= C7)))
    OR  (((i_pixel_row >= R4) AND (i_pixel_row <= R5))
AND ((i_pixel_column >= C0) AND (i_pixel_column <= C7)))
  OR  (((i_pixel_row >= R6) AND (i_pixel_row <= R7)) AND
((i_pixel_column >= C0) AND (i_pixel_column <= C7)))--
horizontal
  OR  (((i_pixel_row >= R0) AND (i_pixel_row <= R7)) AND
((i_pixel_column >= C0) AND (i_pixel_column <= C1)))--
vertical
  OR  (((i_pixel_row >= R0) AND (i_pixel_row <= R7)) AND
((i_pixel_column >= C2) AND (i_pixel_column <= C3)))
    OR  (((i_pixel_row >= R0) AND (i_pixel_row <= R7))
AND ((i_pixel_column >= C4) AND (i_pixel_column <= C5)))
  OR  (((i_pixel_row >= R0) AND (i_pixel_row <= R7)) AND
((i_pixel_column >= C6) AND (i_pixel_column <= C7)))
  THEN grid <=  '1';
  ELSE grid <=  '0';
  END IF;

  --TITLE
  IF (((i_pixel_row >= R0a )  AND (i_pixel_row < R1a ))
AND ((i_pixel_column >= C0a )  AND (i_pixel_column < C3a
))) -- 60 up to 72
  OR (((i_pixel_row >= R0a )  AND (i_pixel_row < R1a ))
AND ((i_pixel_column >= C4a )  AND (i_pixel_column < C9a
)))
  OR (((i_pixel_row >= R0a )  AND (i_pixel_row < R1a ))
AND ((i_pixel_column >= C10a)  AND (i_pixel_column <
C12a)))

  OR (((i_pixel_row >= R1a )  AND (i_pixel_row < R2a ))
AND ((i_pixel_column >= C1a )  AND (i_pixel_column < C2a
))) -- 72 up to 108
  OR (((i_pixel_row >= R1a )  AND (i_pixel_row < R2a ))
AND ((i_pixel_column >= C6a )  AND (i_pixel_column < C7a
)))
  OR (((i_pixel_row >= R1a )  AND (i_pixel_row < R2a ))
AND ((i_pixel_column >= C10a)  AND (i_pixel_column <
C11a)))

  OR (((i_pixel_row >= R2a )  AND (i_pixel_row < R3a ))
AND ((i_pixel_column >= C1a )  AND (i_pixel_column < C2a
))) -- 108 - 120
  OR (((i_pixel_row >= R2a )  AND (i_pixel_row < R3a ))
AND ((i_pixel_column >= C4a )  AND (i_pixel_column < C9a
)))
  OR (((i_pixel_row >= R2a )  AND (i_pixel_row < R3a ))
AND ((i_pixel_column >= C10a)  AND (i_pixel_column <
C12a)))
  THEN titleColor1 <=  '1';
  ELSE titleColor1 <=  '0';
  END IF;

  IF (((i_pixel_row >= R4a )  AND (i_pixel_row < R5a ))
AND ((i_pixel_column >= C0a )  AND (i_pixel_column < C3a
))) -- 140 up to 152
  OR (((i_pixel_row >= R4a )  AND (i_pixel_row < R5a ))
AND ((i_pixel_column >= C4a )  AND (i_pixel_column < C9a
)))
  OR (((i_pixel_row >= R4a )  AND (i_pixel_row < R5a ))
AND ((i_pixel_column >= C10a)  AND (i_pixel_column <
C12a)))
  --
  OR (((i_pixel_row >= R5a )  AND (i_pixel_row < R6a ))
AND ((i_pixel_column >= C1a )  AND (i_pixel_column < C2a
))) -- 152 up to 172
  OR (((i_pixel_row >= R5a )  AND (i_pixel_row < R6a ))
AND ((i_pixel_column >= C4a )  AND (i_pixel_column < C5a
)))
  OR (((i_pixel_row >= R5a )  AND (i_pixel_row < R6a ))
AND ((i_pixel_column >= C8a )  AND (i_pixel_column < C9a
)))
  OR (((i_pixel_row >= R5a )  AND (i_pixel_row < R6a ))
AND ((i_pixel_column >= C10a)  AND (i_pixel_column <
C11a)))
  --
  OR (((i_pixel_row >= R6a )  AND (i_pixel_row < R7a ))
AND ((i_pixel_column >= C1a )  AND (i_pixel_column < C2a
))) -- 172 up to 184
  OR (((i_pixel_row >= R6a )  AND (i_pixel_row < R7a ))
AND ((i_pixel_column >= C4a )  AND (i_pixel_column < C9a
)))
  OR (((i_pixel_row >= R6a )  AND (i_pixel_row < R7a ))
AND ((i_pixel_column >= C10a)  AND (i_pixel_column <
C11a)))
  --
  OR (((i_pixel_row >= R7a )  AND (i_pixel_row < R8a ))
AND ((i_pixel_column >= C1a )  AND (i_pixel_column < C2a
))) -- 184 up to 188
  OR (((i_pixel_row >= R7a )  AND (i_pixel_row < R8a ))
AND ((i_pixel_column >= C4a )  AND (i_pixel_column < C5a
)))
  OR (((i_pixel_row >= R7a )  AND (i_pixel_row < R8a ))
AND ((i_pixel_column >= C8a )  AND (i_pixel_column < C9a
)))
  OR (((i_pixel_row >= R7a )  AND (i_pixel_row < R8a ))
AND ((i_pixel_column >= C10a)  AND (i_pixel_column <
C11a)))
  --
  OR (((i_pixel_row >= R8a )  AND (i_pixel_row < R9a ))
AND ((i_pixel_column >= C1a )  AND (i_pixel_column < C2a
))) -- 188 up to 200
  OR (((i_pixel_row >= R8a )  AND (i_pixel_row < R9a ))
AND ((i_pixel_column >= C4a )  AND (i_pixel_column < C5a
)))
  OR (((i_pixel_row >= R8a )  AND (i_pixel_row < R9a ))
AND ((i_pixel_column >= C8a )  AND (i_pixel_column < C9a
)))
  OR (((i_pixel_row >= R8a )  AND (i_pixel_row < R9a ))
AND ((i_pixel_column >= C10a)  AND (i_pixel_column <
C12a)))
  THEN titleColor2 <=  '1';
  ELSE titleColor2 <=  '0';
  END IF;

  IF (((i_pixel_row >= R10a)  AND (i_pixel_row < R11a))
AND ((i_pixel_column >= C0a )  AND (i_pixel_column < C3a
))) -- 220 up to 232
  OR (((i_pixel_row >= R10a)  AND (i_pixel_row < R11a))
AND ((i_pixel_column >= C4a )  AND (i_pixel_column < C9a
)))
  OR (((i_pixel_row >= R10a)  AND (i_pixel_row < R11a))
AND ((i_pixel_column >= C10a)  AND (i_pixel_column <
C12a)))
  --
  OR (((i_pixel_row >= R11a)  AND (i_pixel_row < R12a))
AND ((i_pixel_column >= C1a )  AND (i_pixel_column < C2a
))) -- 232 up to 244
  OR (((i_pixel_row >= R11a)  AND (i_pixel_row < R12a))
AND ((i_pixel_column >= C4a )  AND (i_pixel_column < C5a
)))
  OR (((i_pixel_row >= R11a)  AND (i_pixel_row < R12a))
AND ((i_pixel_column >= C8a )  AND (i_pixel_column < C9a
)))
  OR (((i_pixel_row >= R11a)  AND (i_pixel_row < R12a))
AND ((i_pixel_column >= C10a)  AND (i_pixel_column <
C11a)))
  --
  OR (((i_pixel_row >= R12a)  AND (i_pixel_row < R13a))
AND ((i_pixel_column >= C1a )  AND (i_pixel_column < C2a
))) -- 244 up to 256
  OR (((i_pixel_row >= R12a)  AND (i_pixel_row < R13a))
AND ((i_pixel_column >= C4a )  AND (i_pixel_column < C5a
)))
  OR (((i_pixel_row >= R12a)  AND (i_pixel_row < R13a))
AND ((i_pixel_column >= C8a )  AND (i_pixel_column < C9a
)))
  OR (((i_pixel_row >= R12a)  AND (i_pixel_row < R13a))
AND ((i_pixel_column >= C10a)  AND (i_pixel_column <
C12a)))
  --
  OR (((i_pixel_row >= R13a)  AND (i_pixel_row < R14a))
AND ((i_pixel_column >= C1a )  AND (i_pixel_column < C2a
))) -- 256 up to 268
  OR (((i_pixel_row >= R13a)  AND (i_pixel_row < R14a))
AND ((i_pixel_column >= C4a )  AND (i_pixel_column < C5a
)))
  OR (((i_pixel_row >= R13a)  AND (i_pixel_row < R14a))
AND ((i_pixel_column >= C8a )  AND (i_pixel_column < C9a
)))
  OR (((i_pixel_row >= R13a)  AND (i_pixel_row < R14a))
AND ((i_pixel_column >= C10a)  AND (i_pixel_column <
C11a)))
  --
  OR (((i_pixel_row >= R14a)  AND (i_pixel_row < R15a))
AND ((i_pixel_column >= C1a )  AND (i_pixel_column < C2a
))) -- 268 up to 280  4rtreg
  OR (((i_pixel_row >= R14a)  AND (i_pixel_row < R15a))
AND ((i_pixel_column >= C4a )  AND (i_pixel_column < C9a
)))
  OR (((i_pixel_row >= R14a)  AND (i_pixel_row < R15a))
AND ((i_pixel_column >= C10a)  AND (i_pixel_column <
C12a)))
  THEN titleColor3 <=  '1';
  ELSE titleColor3 <=  '0';
  END IF;

-- COLOR DECLARATION
  --BOXES AND SELECTOR
  IF (b0_T = '1') THEN
    if  (count_i = 0) then o_red <= X"00"; o_green <=
X"00"; o_blue <= X"FF";
    else
```

```
      if     (B0_i= "01") then o_red <= X"FF"; o_green <=
X"00"; o_blue <= X"00";
      elsif  (B0_i= "10") THEN o_red <= X"00"; o_green <=
X"FF"; o_blue <= X"00";
      elsif  (B0_i= "00") THEN o_red <= X"00"; o_green <=
X"00"; o_blue <= X"00";
      end if;
    END IF;

  ELSIF (b1_T = '1') THEN
    if   (count_i = 1) then o_red <= X"00"; o_green <=
X"00"; o_blue <= X"FF";
    else
      if     (B1_i= "01") then o_red <= X"FF"; o_green <=
X"00"; o_blue <= X"00";
      elsif  (B1_i= "10") THEN o_red <= X"00"; o_green <=
X"FF"; o_blue <= X"00";
      elsif  (B1_i= "00") THEN o_red <= X"00"; o_green <=
X"00"; o_blue <= X"00";
      END IF;
    end if;

  ELSIF (b2_T = '1') THEN
    if   (count_i = 2) then o_red <= X"00"; o_green <=
X"00"; o_blue <= X"FF";
    else
      if     (B2_i= "01") then o_red <= X"FF"; o_green <=
X"00"; o_blue <= X"00";
      elsif  (B2_i= "10") THEN o_red <= X"00"; o_green <=
X"FF"; o_blue <= X"00";
      elsif  (B2_i= "00") THEN o_red <= X"00"; o_green <=
X"00"; o_blue <= X"00";
      end if;
    END IF;

  ELSIF (b3_T = '1') THEN
    if   (count_i = 3) then o_red <= X"00"; o_green <=
X"00"; o_blue <= X"FF";
    else
      if     (B3_i= "01") then o_red <= X"FF"; o_green <=
X"00"; o_blue <= X"00";
      elsif  (B3_i= "10") THEN o_red <= X"00"; o_green <=
X"FF"; o_blue <= X"00";
      elsif  (B3_i= "00") THEN o_red <= X"00"; o_green <=
X"00"; o_blue <= X"00";
      END IF;
    end if;

  ELSIF (b4_T = '1') THEN
    if   (count_i = 4) then o_red <= X"00"; o_green <=
X"00"; o_blue <= X"FF";
    else
      if     (B4_i= "01") then o_red <= X"FF"; o_green <=
X"00"; o_blue <= X"00";
      elsif  (B4_i= "10") THEN o_red <= X"00"; o_green <=
X"FF"; o_blue <= X"00";
      elsif  (B4_i= "00") THEN o_red <= X"00"; o_green <=
X"00"; o_blue <= X"00";
      end if;
    END IF;

  ELSIF (b5_T = '1') THEN
    if   (count_i = 5) then o_red <= X"00"; o_green <=
X"00"; o_blue <= X"FF";
    else
      if     (B5_i= "01") then o_red <= X"FF"; o_green <=
X"00"; o_blue <= X"00";
      elsif  (B5_i= "10") THEN o_red <= X"00"; o_green <=
X"FF"; o_blue <= X"00";
      elsif  (B5_i= "00") THEN o_red <= X"00"; o_green <=
X"00"; o_blue <= X"00";
      end if;
    END IF;

  ELSIF (b6_T = '1') THEN
    if   (count_i = 6) then o_red <= X"00"; o_green <=
X"00"; o_blue <= X"FF";
    else
      if     (B6_i= "01") then o_red <= X"FF"; o_green <=
X"00"; o_blue <= X"00";
      elsif  (B6_i= "10") THEN o_red <= X"00"; o_green <=
X"FF"; o_blue <= X"00";
      elsif  (B6_i= "00") THEN o_red <= X"00"; o_green <=
X"00"; o_blue <= X"00";
      end if;
    END IF;

  ELSIF (b7_T = '1') THEN
    if   (count_i = 7) then o_red <= X"00"; o_green <=
X"00"; o_blue <= X"FF";
    else
      if     (B7_i= "01") then o_red <= X"FF"; o_green <=
X"00"; o_blue <= X"00";
      elsif  (B7_i= "10") THEN o_red <= X"00"; o_green <=
X"FF"; o_blue <= X"00";
      elsif  (B7_i= "00") THEN o_red <= X"00"; o_green <=
X"00"; o_blue <= X"00";
      end if;
```

```
    END IF;

  ELSIF (b8_T = '1') THEN
    if   (count_i = 8) then o_red <= X"00"; o_green <=
X"00"; o_blue <= X"FF";
    else
      if     (B8_i= "01") then o_red <= X"FF"; o_green <=
X"00"; o_blue <= X"00";
      elsif  (B8_i= "10") THEN o_red <= X"00"; o_green <=
X"FF"; o_blue <= X"00";
      elsif  (B8_i= "00") THEN o_red <= X"00"; o_green <=
X"00"; o_blue <= X"00";
      end if;
    END IF;

  --GRID
  ELSIF (grid = '1') THEN  o_red <= X"FF"; o_green <=
X"FF"; o_blue <= X"FF";

  ELSIF (titleColor1 ='1' ) THEN  o_red <= X"22"; o_green
<= X"00"; o_blue <= X"FF";
  ELSIF (titleColor2 ='1' ) THEN  o_red <= X"22"; o_green
<= X"00"; o_blue <= X"FF";
  ELSIF (titleColor3 ='1' ) THEN  o_red <= X"22"; o_green
<= X"00"; o_blue <= X"FF";
  ELSIF (b0_T = '0' and b1_T = '0'and b2_T = '0' and b3_T
= '0'and b4_T = '0'and b5_T = '0'and b6_T = '0'and b7_T =
'0'and b8_T = '0'and grid = '0' and titleColor1 ='0' and
titleColor2 ='0' and titleColor3 ='0' ) then
    o_red <= X"00"; o_green <= X"00"; o_blue <= X"00";
  END IF;

  --WIN
  IF (win_T = '1' AND Win_i = "10" ) THEN  o_red <=
X"00"; o_green <= X"FF"; o_blue <= X"00";
  ELSIF (win_T = '1' AND Win_i = "01" ) THEN  o_red <=
X"FF"; o_green <= X"00"; o_blue <= X"00";
  ELSIF (win_T = '1' AND Win_i = "11" ) THEN  o_red <=
X"77"; o_green <= X"77"; o_blue <= X"77";
  END IF;

  --TITLE

END PROCESS;

END behavioral;
```

Lampiran 2. Kode Blok Color ROM

```
LIBRARY  IEEE;
USE  IEEE.STD_LOGIC_1164.ALL;
USE  IEEE.STD_LOGIC_ARITH.ALL;
USE  IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY display_vhd  IS
  PORT(
      i_clk           : IN STD_LOGIC;
    Button_1_i_disp     : IN STD_LOGIC;
    Button_2_i_disp     : IN STD_LOGIC;
    ResetGame_i_disp  : IN STD_LOGIC;
    ResetAll_i_disp     : IN STD_LOGIC;
    B0_i_disp       : IN STD_LOGIC_VECTOR (1 downto 0);
    B1_i_disp       : IN STD_LOGIC_VECTOR (1 downto 0);
    B2_i_disp       : IN STD_LOGIC_VECTOR (1 downto 0);
    B3_i_disp       : IN STD_LOGIC_VECTOR (1 downto 0);
    B4_i_disp       : IN STD_LOGIC_VECTOR (1 downto 0);
    B5_i_disp       : IN STD_LOGIC_VECTOR (1 downto 0);
    B6_i_disp       : IN STD_LOGIC_VECTOR (1 downto 0);
    B7_i_disp       : IN STD_LOGIC_VECTOR (1 downto 0);
    B8_i_disp       : IN STD_LOGIC_VECTOR (1 downto 0);
    Win_i1_disp       : IN STD_LOGIC_VECTOR (1 downto 0);
    count_i_disp    : IN STD_LOGIC_VECTOR (3 downto 0);
      VGA_R           : OUT STD_LOGIC_VECTOR( 5 DOWNTO 0
);
      VGA_G           : OUT STD_LOGIC_VECTOR( 5 DOWNTO 0
);
      VGA_B           : OUT STD_LOGIC_VECTOR( 5 DOWNTO 0
);
      VGA_HS          : OUT STD_LOGIC;
      VGA_VS          : OUT STD_LOGIC;
      VGA_CLK         : OUT STD_LOGIC;
      VGA_BLANK       : OUT STD_LOGIC;
      ledtest1    : OUT STD_LOGIC);
END display_vhd;

ARCHITECTURE behavioral OF display_vhd  IS

SIGNAL  red                 :       STD_LOGIC_VECTOR (5
DOWNTO 0);
SIGNAL  green               :       STD_LOGIC_VECTOR (5
DOWNTO 0);
SIGNAL  blue                :       STD_LOGIC_VECTOR (5
DOWNTO 0);
SIGNAL  red_color           :       STD_LOGIC_VECTOR (7
```

```
DOWNTO 0);
SIGNAL  green_color          :        STD_LOGIC_VECTOR (7
DOWNTO 0);
SIGNAL  blue_color           :        STD_LOGIC_VECTOR (7
DOWNTO 0);
SIGNAL  pixel_row            :        STD_LOGIC_VECTOR (9
DOWNTO 0);
SIGNAL  pixel_column         :        STD_LOGIC_VECTOR (9
DOWNTO 0);
SIGNAL  red_on               :        STD_LOGIC;
SIGNAL  green_on             :        STD_LOGIC;
SIGNAL  blue_on              :        STD_LOGIC;



COMPONENT vga  IS
  PORT(
    i_clk              :  IN   STD_LOGIC;
    i_red              :  IN   STD_LOGIC;
    i_green            :  IN   STD_LOGIC;
    i_blue             :  IN   STD_LOGIC;
    o_red              :  OUT  STD_LOGIC;
    o_green            :  OUT  STD_LOGIC;
    o_blue             :  OUT  STD_LOGIC;
    o_horiz_sync       :  OUT  STD_LOGIC;
    o_vert_sync        :  OUT  STD_LOGIC;
    o_pixel_row        :  OUT  STD_LOGIC_VECTOR( 9 DOWNTO
0 );
    o_pixel_column     :  OUT  STD_LOGIC_VECTOR( 9 DOWNTO
0 ));
END COMPONENT;

COMPONENT color_rom_vhd  IS
  PORT(
    Button_1_i   : IN STD_LOGIC;
    Button_2_i   : IN STD_LOGIC;
    ResetGame_i    : IN STD_LOGIC;
    ResetAll_i   : IN STD_LOGIC;
    B0_i     : IN STD_LOGIC_VECTOR (1 downto 0);
    B1_i     : IN STD_LOGIC_VECTOR (1 downto 0);
    B2_i     : IN STD_LOGIC_VECTOR (1 downto 0);
    B3_i     : IN STD_LOGIC_VECTOR (1 downto 0);
    B4_i     : IN STD_LOGIC_VECTOR (1 downto 0);
    B5_i     : IN STD_LOGIC_VECTOR (1 downto 0);
    B6_i     : IN STD_LOGIC_VECTOR (1 downto 0);
    B7_i     : IN STD_LOGIC_VECTOR (1 downto 0);
    B8_i     : IN STD_LOGIC_VECTOR (1 downto 0);
    Win_i1         : IN STD_LOGIC_VECTOR (1 downto 0);
    count_i      : IN STD_LOGIC_VECTOR (3 downto 0);
      i_pixel_column  : IN STD_LOGIC_VECTOR( 9 DOWNTO 0
);
      i_pixel_row     : IN STD_LOGIC_VECTOR( 9 DOWNTO 0
);
      o_red           : OUT STD_LOGIC_VECTOR( 7 DOWNTO 0
);
      o_green         : OUT STD_LOGIC_VECTOR( 7 DOWNTO 0
);
      o_blue          : OUT STD_LOGIC_VECTOR( 7 DOWNTO 0
);
      ledtest      : OUT STD_LOGIC);
END COMPONENT;

BEGIN

vga_driver0 : vga
   PORT MAP (
   i_clk             => i_clk,
   i_red             => '1',
   i_green           => '1',
   i_blue            => '1',
   o_red             => red_on,
   o_green           => green_on,
   o_blue            => blue_on,
   o_horiz_sync      => VGA_HS,
   o_vert_sync       => VGA_VS,
   o_pixel_row       => pixel_row,
   o_pixel_column    => pixel_column);

color_rom0 : color_rom_vhd
   PORT MAP (
  Button_1_i     => Button_1_i_disp,
  Button_2_i   => Button_1_i_disp,
  ResetGame_i    => ResetGame_i_disp,
  ResetAll_i   => ResetAll_i_disp,
  B0_i     => B0_i_disp,
  B1_i     => B1_i_disp,
  B2_i     => B2_i_disp,
  B3_i     => B3_i_disp,
  B4_i     => B4_i_disp,
  B5_i     => B5_i_disp,
  B6_i     => B6_i_disp,
  B7_i     => B7_i_disp,
  B8_i     => B8_i_disp,
  Win_i1         => Win_i1_disp,
  count_i      => count_i_disp,
    i_pixel_column    => pixel_column,
    i_pixel_row       => pixel_row,
    o_red             => red_color,
    o_green           => green_color,
    o_blue            => blue_color,
    ledtest       => ledtest1
```

```
 );

red   <= red_color  (7 DOWNTO 2) ;
green <= green_color(7 DOWNTO 2) ;
blue  <= blue_color (7 DOWNTO 2) ;


PROCESS(red_on,green_on,blue_on,red,green,blue)
BEGIN

  IF (red_on = '1'  ) THEN VGA_R <=  red;
  ELSE  VGA_R <=  "000000";
  END IF;

  IF (green_on = '1'  ) THEN VGA_G <=  green;
  ELSE VGA_G <=  "000000";
  END IF;

  IF (blue_on = '1'  ) THEN VGA_B <=  blue;
  ELSE VGA_B <=  "000000";
  END IF;


END PROCESS;



END behavioral;
```

Lampiran 3. Kode Blok Display

```
LIBRARY  IEEE;
USE  IEEE.STD_LOGIC_1164.ALL;
USE  IEEE.STD_LOGIC_ARITH.ALL;
USE  IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY vga  IS
  PORT(
    i_clk              :  IN   STD_LOGIC;
    i_red              :  IN   STD_LOGIC;
    i_green            :  IN   STD_LOGIC;
    i_blue             :  IN   STD_LOGIC;
    o_red              :  OUT  STD_LOGIC;
    o_green            :  OUT  STD_LOGIC;
    o_blue             :  OUT  STD_LOGIC;
    o_horiz_sync       :  OUT  STD_LOGIC;
    o_vert_sync        :  OUT  STD_LOGIC;
    o_pixel_row        :  OUT  STD_LOGIC_VECTOR( 9 DOWNTO
0 );
    o_pixel_column     :  OUT  STD_LOGIC_VECTOR( 9 DOWNTO
0 ));
END vga;

ARCHITECTURE behavioral OF vga  IS

CONSTANT TH     : INTEGER := 800;
CONSTANT THB1   : INTEGER := 660;
CONSTANT THB2   : INTEGER := 756;
CONSTANT THD    : INTEGER := 640;

CONSTANT TV     : INTEGER := 525;
CONSTANT TVB1   : INTEGER := 494;
CONSTANT TVB2   : INTEGER := 495;
CONSTANT TVD    : INTEGER := 480;

SIGNAL clock_25MHz  :  STD_LOGIC;
SIGNAL horiz_sync   :  STD_LOGIC;
SIGNAL vert_sync    :  STD_LOGIC;
SIGNAL video_on     :  STD_LOGIC;
SIGNAL video_on_v   :  STD_LOGIC;
SIGNAL video_on_h   :  STD_LOGIC;
SIGNAL h_count      :  STD_LOGIC_VECTOR( 9 DOWNTO 0 );
SIGNAL v_count      :  STD_LOGIC_VECTOR( 9 DOWNTO 0 );

BEGIN

video_on    <= video_on_h  AND video_on_v;

o_red       <= i_red AND video_on;
o_green     <= i_green AND video_on;
o_blue      <= i_blue AND video_on;

o_horiz_sync  <= horiz_sync;
o_vert_sync   <= vert_sync;

PROCESS (i_clk)
  BEGIN
    IF i_clk'EVENT AND i_clk='1' THEN
      IF (clock_25MHz = '0') THEN
        clock_25MHz <= '1';
      ELSE
        clock_25MHz <= '0';
      END IF;
    END IF;
  END PROCESS;

PROCESS
  BEGIN
  WAIT UNTIL(  clock_25MHz'EVENT  ) AND ( clock_25MHz  =
'1'  );
```

```
  END IF;

  IF ( v_count  <= TVD-1 ) THEN
      video_on_v  <= '1';
      o_pixel_row  <= v_count;
  ELSE
      video_on_v  <= '0';
  END  IF;

END  PROCESS;
END behavioral;
```

Lampiran 4. Kode Blok VGA

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

entity CLOCKDIV is port(
  CLK: IN std_logic;
  DIVOUT: buffer BIT);
end CLOCKDIV;

architecture behavioural of CLOCKDIV is
  begin
    PROCESS(CLK)
      variable count: integer:=0;
      constant div: integer:=2500000;
    begin
        if CLK'event and CLK='1' then

          if(count<div) then
            count:=count+1;
            if(DIVOUT='0') then
              DIVOUT<='0';
            elsif(DIVOUT='1') then
              DIVOUT<='1';
            end if;
          else
            if(DIVOUT='0') then
              DIVOUT<='1';
            elsif(DIVOUT='1') then
              DIVOUT<='0';
            end if;
          count:=0;
          end if;

        end if;
    end process;
end behavioural;
```

Lampiran 5. Kode Blok Clockdiv

```
LIBRARY  IEEE;
USE  IEEE.STD_LOGIC_1164.ALL;
USE  IEEE.STD_LOGIC_ARITH.ALL;
USE  IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY top_level_vhd  IS
  PORT(
      CLOCK_50    : IN STD_LOGIC;
      SW          : IN STD_LOGIC_VECTOR( 9 DOWNTO 0 );
      VGA_R       : OUT STD_LOGIC_VECTOR( 5 DOWNTO 0 );
      VGA_G       : OUT STD_LOGIC_VECTOR( 5 DOWNTO 0 );
      VGA_B       : OUT STD_LOGIC_VECTOR( 5 DOWNTO 0 );
      VGA_HS      : OUT STD_LOGIC;
      VGA_VS      : OUT STD_LOGIC;
      VGA_CLK     : OUT STD_LOGIC;
      VGA_BLANK   : OUT STD_LOGIC;
      GPIO_0      : OUT STD_LOGIC_VECTOR( 35 DOWNTO 0 );
      LEDR        : BUFFER STD_LOGIC_VECTOR( 9 DOWNTO 0
));
```

```
END top_level_vhd;

ARCHITECTURE behavioral OF top_level_vhd  IS

SIGNAL  Button_1_Top  : STD_LOGIC;
SIGNAL  Button_2_Top  : STD_LOGIC;
SIGNAL  ResetGame_Top   : STD_LOGIC;
SIGNAL  ResetAll_Top  : STD_LOGIC;
SIGNAL  ledtest2    : STD_LOGIC;

SIGNAL  clk_o       : STD_LOGIC;

SIGNAL  B0temp      : STD_LOGIC_VECTOR(1 DOWNTO 0);
SIGNAL  B1temp      : STD_LOGIC_VECTOR(1 DOWNTO 0);
SIGNAL  B2temp      : STD_LOGIC_VECTOR(1 DOWNTO 0);
SIGNAL  B3temp      : STD_LOGIC_VECTOR(1 DOWNTO 0);
SIGNAL  B4temp      : STD_LOGIC_VECTOR(1 DOWNTO 0);
SIGNAL  B5temp      : STD_LOGIC_VECTOR(1 DOWNTO 0);
SIGNAL  B6temp      : STD_LOGIC_VECTOR(1 DOWNTO 0);
SIGNAL  B7temp      : STD_LOGIC_VECTOR(1 DOWNTO 0);
SIGNAL  B8temp      : STD_LOGIC_VECTOR(1 DOWNTO 0);
SIGNALWin_i1temp    : STD_LOGIC_VECTOR (1 downto 0);
SIGNALcount_itemp     :  STD_LOGIC_VECTOR (3 downto 0);

COMPONENT display_vhd  IS
  PORT(
      i_clk           : IN STD_LOGIC;
    Button_1_i_disp     : IN STD_LOGIC;
    Button_2_i_disp     : IN STD_LOGIC;
    ResetGame_i_disp  : IN STD_LOGIC;
    ResetAll_i_disp     : IN STD_LOGIC;
    B0_i_disp       : IN STD_LOGIC_VECTOR (1 downto 0);
    B1_i_disp       : IN STD_LOGIC_VECTOR (1 downto 0);
    B2_i_disp       : IN STD_LOGIC_VECTOR (1 downto 0);
    B3_i_disp       : IN STD_LOGIC_VECTOR (1 downto 0);
    B4_i_disp       : IN STD_LOGIC_VECTOR (1 downto 0);
    B5_i_disp       : IN STD_LOGIC_VECTOR (1 downto 0);
    B6_i_disp       : IN STD_LOGIC_VECTOR (1 downto 0);
    B7_i_disp       : IN STD_LOGIC_VECTOR (1 downto 0);
    B8_i_disp       : IN STD_LOGIC_VECTOR (1 downto 0);
    Win_i1_disp       : IN STD_LOGIC_VECTOR (1 downto 0);
    count_i_disp    : IN STD_LOGIC_VECTOR (3 downto 0);
      VGA_R           : OUT STD_LOGIC_VECTOR( 5 DOWNTO 0
);
      VGA_G           : OUT STD_LOGIC_VECTOR( 5 DOWNTO 0
);
      VGA_B           : OUT STD_LOGIC_VECTOR( 5 DOWNTO 0
);
      VGA_HS          : OUT STD_LOGIC;
      VGA_VS          : OUT STD_LOGIC;
      VGA_CLK         : OUT STD_LOGIC;
      VGA_BLANK       : OUT STD_LOGIC;
      ledtest1    : OUT STD_LOGIC);
END COMPONENT;

component CLOCKDIV is
  port(
  CLK       : IN std_logic;
  DIVOUT    : buffer std_logic
  );
end component;

component tictactoe is
  port (
    button_1 : in std_logic;
    button_2 : in std_logic;
    resetGame  : in std_logic;
    o_b0, o_b1, o_b2, o_b3, o_b4, o_b5, o_b6, o_b7, o_b8,
o_win  : out std_logic_vector (1 downto 0);
    o_count, o_countTurn                    : out
std_logic_vector (3 downto 0);
    rst, clk : in std_logic
  );
end component;

BEGIN

module_vga : display_vhd
   PORT MAP (
  i_clk           => CLOCK_50,
  Button_1_i_disp     => Button_1_Top,
  Button_2_i_disp     => Button_2_Top,
  ResetGame_i_disp  => ResetGame_Top,
  ResetAll_i_disp     => ResetAll_Top,
  B0_i_disp       => B0temp,
  B1_i_disp       => B1temp,
  B2_i_disp       => B2temp,
  B3_i_disp       => B3temp,
  B4_i_disp       => B4temp,
  B5_i_disp       => B5temp,
  B6_i_disp       => B6temp,
  B7_i_disp       => B7temp,
  B8_i_disp       => B8temp,
  Win_i1_disp       => Win_i1temp,
  count_i_disp    => count_itemp,
    VGA_R                 => VGA_R
```

```
    VGA_R                => VGA_R,
    VGA_G                => VGA_G,
    VGA_B                => VGA_B,
    VGA_HS               => VGA_HS,
    VGA_VS               => VGA_VS,
    VGA_CLK              => VGA_CLK,
    VGA_BLANK            => VGA_BLANK,
    ledtest1        => ledtest2
);

tictactoe0 : tictactoe
  port map(
  button_1   => Button_1_Top,
  button_2   => Button_2_Top,
  resetGame  => ResetGame_Top,
  o_b0   => B0temp,
  o_b1   => B1temp,
  o_b2   => B2temp,
  o_b3   => B3temp,
  o_b4   => B4temp,
  o_b5   => B5temp,
  o_b6   => B6temp,
  o_b7   => B7temp,
  o_b8   => B8temp,
  o_win    => Win_i1temp,
  o_count    => count_itemp,
  rst      => ResetAll_Top,
  clk      => clk_o
  );

detik : CLOCKDIV port map (CLK => CLOCK_50, DIVOUT =>
clk_o);

Button_1_Top   <= NOT SW(0);
Button_2_Top   <= NOT SW(1);
ResetGame_Top  <= SW(2);
ResetAll_Top <= SW(3);
ledtest2   <= LEDR(1);

END behavioral;
```

Lampiran 6. Kode Blok Top Level