FUNDAMENTAL OF DIGITAL SYSTEM FINAL PROJECT REPORT

DEPARTMENT OF ELECTRICAL ENGINEERING

UNIVERSITAS INDONESIA

SHUTTLEGUARD PRO: A Program to Help the Table Tennis Umpire

GROUP AP09

Christopher Satya Fredella Balakosa	**2206059755**
Kania Aidilla Firka	**2206062983**
Fabsesya Muhammad Putra Ibradi	**2206829433**
Stefanus Simon Rilando	**2206830422**

PREFACE

Puji syukur kami panjatkan kehadirat Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga kami dapat menyelesaikan laporan proyek akhir praktikum Perancangan Sistem Digital dengan judul “SHUTTLEGUARD PRO: A Program to Help the Table Tennis Umpire” ini.

Kami ingin mengucapkan terima kasih kepada dosen pengampu mata kuliah Perancangan Sistem Digital, Bapak Muhammad Firdaus Syawaludin Lubis, S.T., M.T., Ph.D., Bapak Dr. Ruki Harwahyu, ST. MT. MSc., dan Bapak Yan Maraden, S.T., M.T., M.Sc.. Selain itu, kami juga ingin menyampaikan apresiasi kepada Bang Juan Jonathan selaku asisten laboratorium yang telah memberikan arahan, bimbingan, serta masukan yang berharga. Tanpa bantuan beliau, penulisan laporan ini tidak akan mencapai tingkat yang diharapkan.

Kami menyadari bahwa penyusunan makalah ini tidak lepas dari keterbatasan pengetahuan dan kemampuan kami. Oleh karena itu, saran dan kritik yang membangun sangat diharapkan guna perbaikan di masa mendatang. Semoga laporan ini dapat memberikan manfaat dan kontribusi positif bagi pembaca. Akhir kata, kami mohon maaf atas segala kekurangan dan kesalahan yang mungkin ada dalam laporan ini.

Depok, December 24, 2023

Group AP09

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Dalam dunia olahraga, teknologi semakin banyak digunakan untuk meningkatkan akurasi, keadilan, dan pengalaman pertandingan. Demikian juga olahraga tenis meja yang dapat mengambil manfaat dari integrasi teknologi, seperti dengan program VHDL (Very High-Speed Integrated Circuit Hardware Description Language) yang kami kembangkan, di mana program ini dirancang untuk dapat membantu umpire tenis meja dalam menentukan posisi pemain, pemain mana yang berhak melakukan servis, dan memonitor poin pertandingan.

Program ini merupakan simulasi rangkaian yang dikembangkan tanpa implementasi langsung pada perangkat keras, yaitu hanya dengan menggunakan konsep Truth Table, Testbench, dan Seven Segment Display. Dengan memanfaatkan VHDL, kami merangkai sebuah algoritma yang dapat secara akurat mengidentifikasi posisi pemain di lapangan, memastikan bahwa aturan servis diikuti dengan benar, dan secara *real-time* mencatat poin pertandingan. Selain itu, kelebihan program ini adalah kemampuannya untuk memberikan visualisasi hasil pertandingan melalui Seven Segment Display, memberikan pengalaman yang lebih interaktif dan informatif bagi pemain dan penonton.

Dengan program yang kami rancang ini, diharapkan pertandingan tenis meja dapat berlangsung dengan lebih efisien, mengurangi potensi kesalahan umpire, dan memberikan pengalaman yang lebih menarik bagi semua pihak yang terlibat. Program VHDL ini juga memberikan fleksibilitas untuk disesuaikan dengan perubahan aturan atau kebijakan dalam olahraga tenis meja, sehingga menjadikannya solusi yang dapat diterapkan dalam berbagai tingkatan kompetisi. Sebagai hasilnya, kami berharap program ini dapat menjadi kontribusi positif terhadap perkembangan teknologi dalam dunia olahraga, khususnya dalam memajukan cabang olahraga tenis meja secara global.

1.2 PROJECT DESCRIPTION

Proyek ini merupakan pengembangan sebuah program berbasis VHDL yang dirancang untuk meningkatkan efisiensi dan akurasi dalam mengelola pertandingan tenis meja. Dengan fokus pada simulasi rangkaian dan tanpa implementasi langsung pada perangkat keras, program ini menggunakan pendekatan berbasis Truth Table, Testbench, dan Seven Segment Display.

Salah satu fitur utama dari proyek ini adalah kemampuannya untuk menentukan posisi pemain di lapangan dengan mengenali letak pemain secara akurat, memberikan informasi yang jelas tentang posisi mereka pada saat tertentu selama pertandingan. Hal ini tidak hanya memudahkan kerja umpire, tetapi juga dapat memberikan informasi tambahan kepada penonton.

Selain itu, proyek ini mengimplementasikan aturan servis tenis meja dan memastikan bahwa pemain yang melakukan servis sesuai dengan ketentuan yang berlaku. Dengan menggabungkan logika dan kontrol yang tepat, program dapat mengenali pemain yang berhak melakukan servis pada setiap putaran pertandingan sehingga menghindari potensi kesalahan manusia dalam penilaian.

Poin pertandingan juga secara otomatis dicatat dan ditampilkan melalui Seven Segment Display. Dengan hal ini, hasil pertandingan dapat dengan cepat dan jelas disampaikan kepada pemain, penonton, dan pihak terkait lainnya. Visualisasi poin melalui display ini memberikan pengalaman yang lebih interaktif dan transparan.

1.3 OBJECTIVES

The objectives of this project are as follows:

1. Memastikan bahwa pemain tenis meja berada pada posisi yang sesuai dengan peraturan yang berlaku.
2. Menetapkan pemain tenis meja yang bertanggung jawab untuk melakukan servis.
3. Menyimpan dan menampilkan poin dalam format seven segment yang terhubung dengan sistem secara menyeluruh.

4. Membantu wasit dalam pengelolaan pertandingan tenis meja.

1.4 ROLES AND RESPONSIBILITIES

The roles and responsibilities assigned to the group members are as follows:

Roles	Responsibilities	Person
Membuat Top Level	Menyatukan component-component yang ada	Christoper Satya Fredella Balakosa
Membuat FSM	Menentukan kondisi servis dan posisi melalui FSM tipe Moore Machine	Seluruh anggota
Membuat Counter	Menyimpan poin dalam bentuk seven segment 2 bit	Fabsesya Muhammad Putra Ibradi

Membuat Testbench	Memeriksa input/output berdasarkan pertandingan sebenarnya	Kania Aidilla Firka
Membuat Laporan	Membuktikan pertanggung jawaban dari pelaksanaan proyek akhir	Seluruh anggota
Membuat PowerPoint	Membuat PowerPoint berdasarkan laporan yang telah dibuat	Stefanus Simon Rilando

Table 1. Roles and Responsibilities

CHAPTER 2

IMPLEMENTATION

2.1 EQUIPMENT

The tools that are going to be used in this project are as follows:

- Visual Studio Code
- ModelSim
- GoogleDocs
- Draw.io

2.2 IMPLEMENTATION

Hal pertama yang kami lakukan ialah merancang Finite State Machine untuk menentukan siapa yang akan melakukan servis. Karena dalam penggunaannya output yang dihasilkan hanya dipengaruhi oleh state pada saat itu aja tanpa dipengaruhi input, makas di sini kami menggunakan Moore machine. Selanjutnya, kami memahami poin-poin yang diperoleh team A dan team B, siapa yang melakukan servis, dan juga posisi dari team A maupun team B. Berikut cara kerja dari finite state machine melalui state diagram berikut:

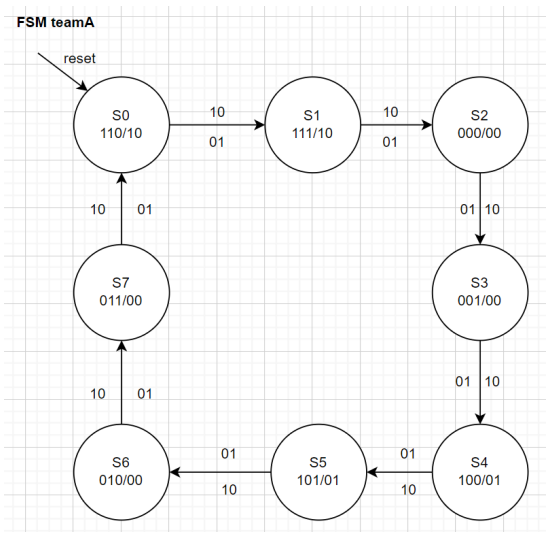


Fig 1. State Diagram for team A

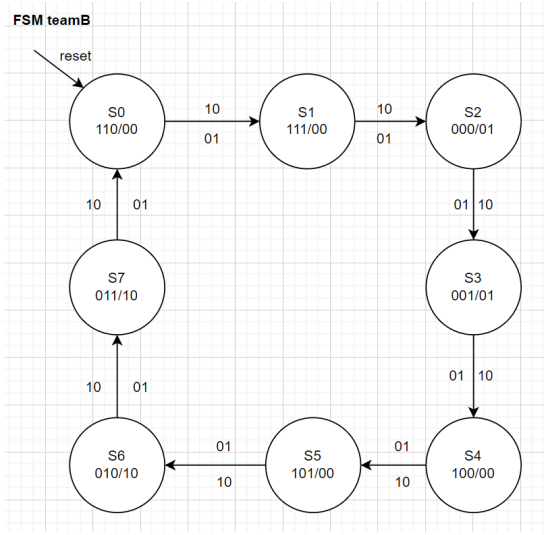


Fig 2. State Diagram for team B

State diagram tersebut disusun berdasarkan referensi peraturan tenis meja resmi dari ITTF (International Table Tennis Federation). State diagram tersebut menghasilkan output berupa servisA atau servis B, yaitu informasi mengenai pemain mana yang berhak melakukan servis.

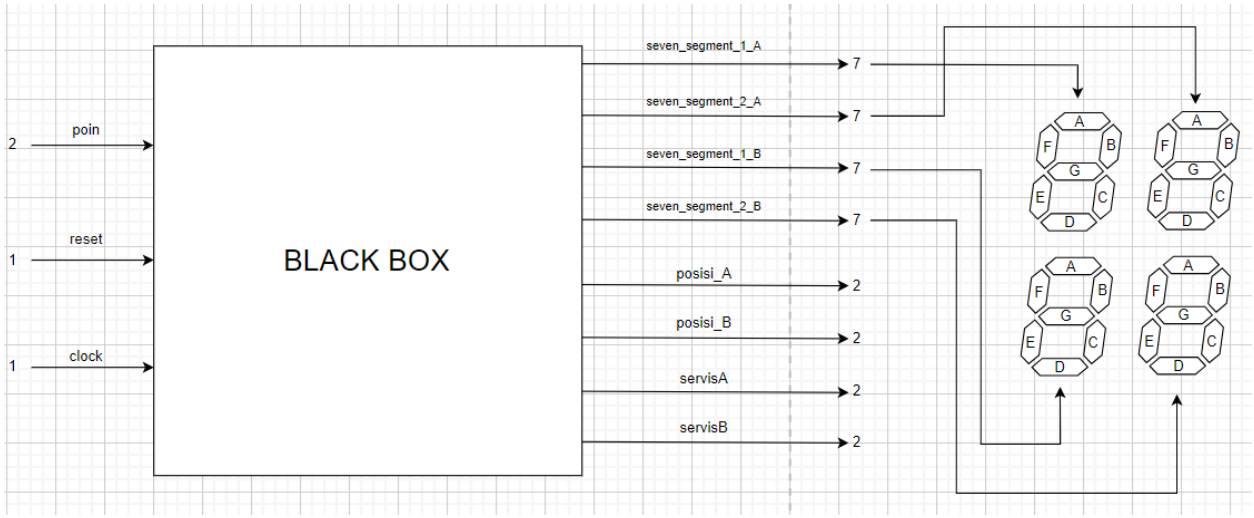


Fig 3. Schematic Black Box

CHAPTER 3

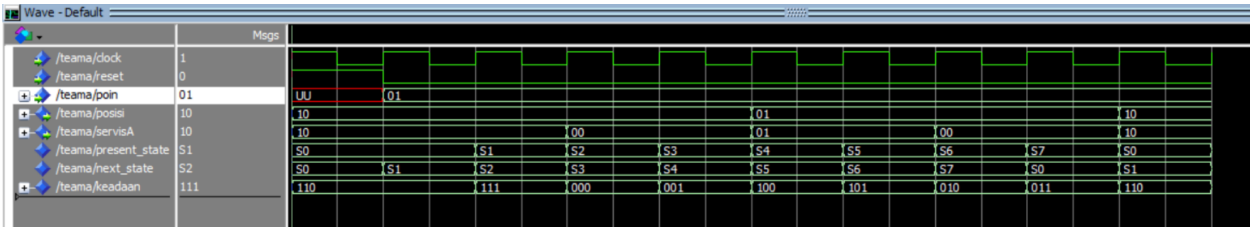
TESTING AND ANALYSIS

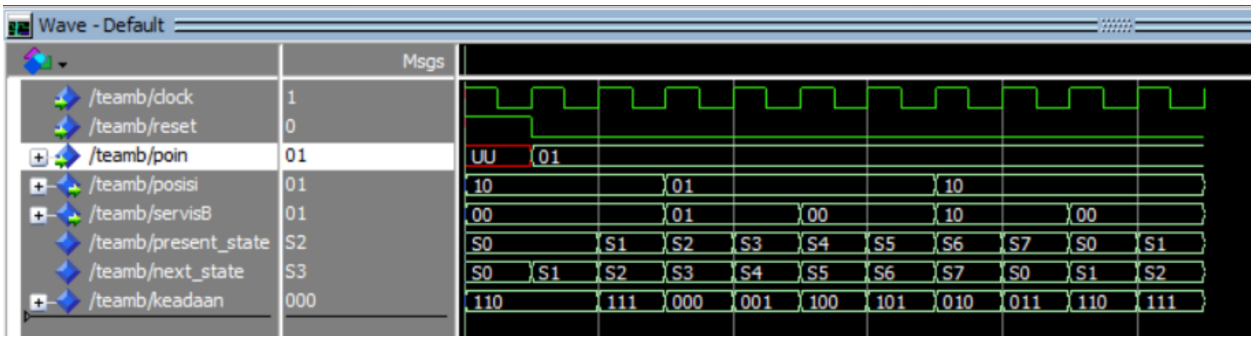
3.1 TESTING

Pada pengujian rangkaian digital SHUTTLEGUARD PRO ini, kami menggunakan testbench yang didasarkan pada pertandingan semi final ITTF worlds 2021 cabang ganda putri antara QIAN Tianyi dan CHEN Meng melawan HAYATA Hina dan ITO Mima. Testbench dilakukan hingga skor mencapai 10-9 dengan keunggulan pasangan HAYATA Hina dan ITO Mima. Referensi pertandingan berupa video dapat ditemukan di bagian referensi dalam laporan ini.

3.2 RESULT

SHUTTLEGUARD PRO yang telah dibuat diuji menggunakan model simulasi dengan parameter yang sesuai dengan truth table di atas. Dengan demikian, hasil simulasi tersebut adalah sebagai berikut:





3.3 ANALYSIS

Berdasarkan simulasi yang dilakukan, keadaan dan output akan berubah apabila poin bernilai 01 atau 10 serta clock pulse naik (rising edge). Namun, perubahan pada output tersebut membutuhkan delay 1 clock pulse.

Pengujian dilakukan dengan input poin sesuai dengan pertandingan referensi, dan diselingi dengan angka 00 pada setiap perubahan poin sebagai hold state. Hold state ini diperlukan untuk dapat melihat perubahan (next state) dari suatu input point yang diberikan. Selain itu juga, pada saat sedang berlangsungnya permainan di kondisi sebenarnya (permainan dalam kondisi hidup) input poin sebelumnya akan di hold hingga perubahan poin selanjutnya.

Untuk output pada seven segment decoder, MSB merupakan bit pada pin g dan LSB merupakan bit pada pin a. Seven segment decoder menggunaka prinsip counter up 2 bit. Pada testbench jika seven segment decoder telah bernilai 9, maka seven segment decoder tersebut akan kembali ke 0 sedangkan seven segment decoder lainnya akan melakukan increment. Mengacu kepada peraturan servis permainan ganda oleh ITTF, nilai servisA akan berubah jika team terkait telah melakukan dua kali service. Hal ini juga berlaku untuk nilai servisB.

CHAPTER 4

CONCLUSION

Berdasarkan proyek akhir yang telah kami buat, dapat disimpulkan hal-hal berikut:

- Rangkaian ini menghasilkan tiga output, yakni poin dari kedua tim, posisi pemain pada kedua tim sebelum servis, dan pemain yang memiliki hak untuk melakukan servis.
- Penghitung poin diimplementasikan menggunakan konsep counter, sementara penentu kondisi servis menggunakan konsep Finite State Machine (FSM) tipe Moore.
- Untuk menampilkan poin perolehan dari kedua tim, dibutuhkan dua seven segment display, sehingga secara total diperlukan empat buah.
- Penggunaan hold state membantu memperjelas perubahan next state dari suatu input dalam sistem ini.

REFERENCES

[1] "Full match | Qian Tianyi / Chen Meng vs Hayata Hina / ito Mima | WD SF | #ittfworlds2021," YouTube, https://www.youtube.com/watch?v=I-LRfiwZdww&t=277s&ab_channel=WorldTableTennis (accessed Dec. 24, 2023).

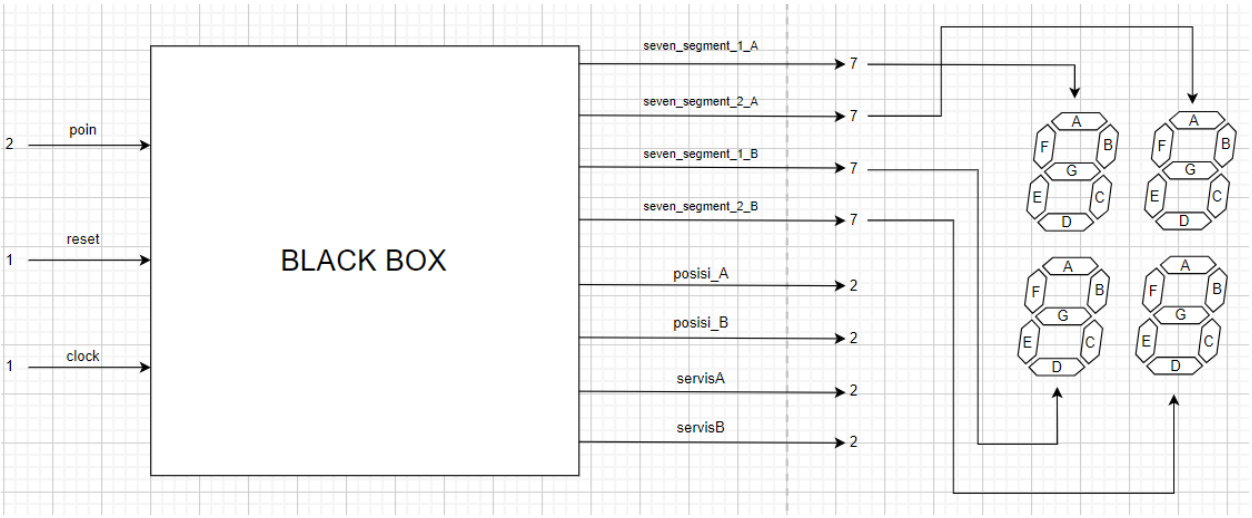
[2] Cornilleau, "ITTF - Official International Table Tennis Federation Rules," Cornilleau, https://cornilleau-tabletennis.com.au/official-ittf-table-tennis-rules (accessed Dec. 24, 2023).

[3] Jonas Julian Jensen. "DUAL 7-SEGMENT DISPLAY FPGA CONTROLLER" https://vhdlwhiz.com/dual-7-segment-display (accessed Dec. 24, 2023).

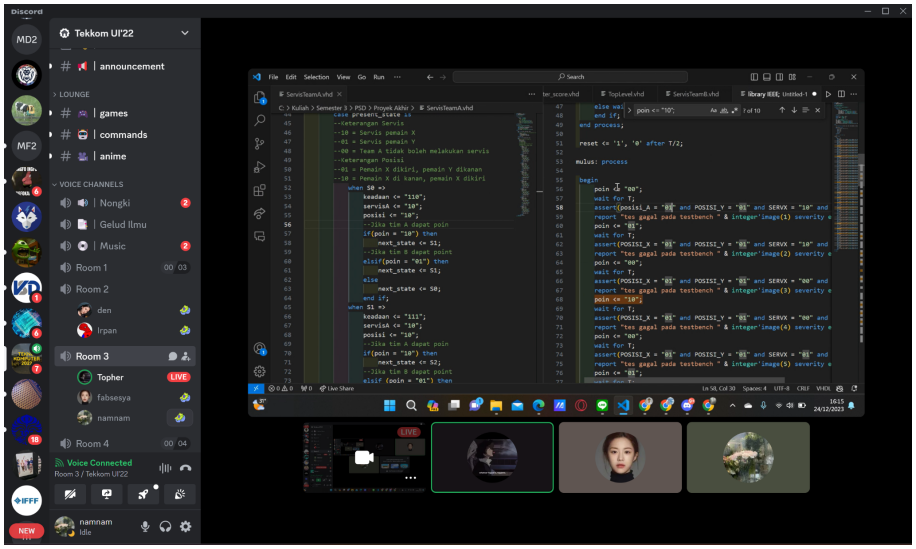
APPENDICES

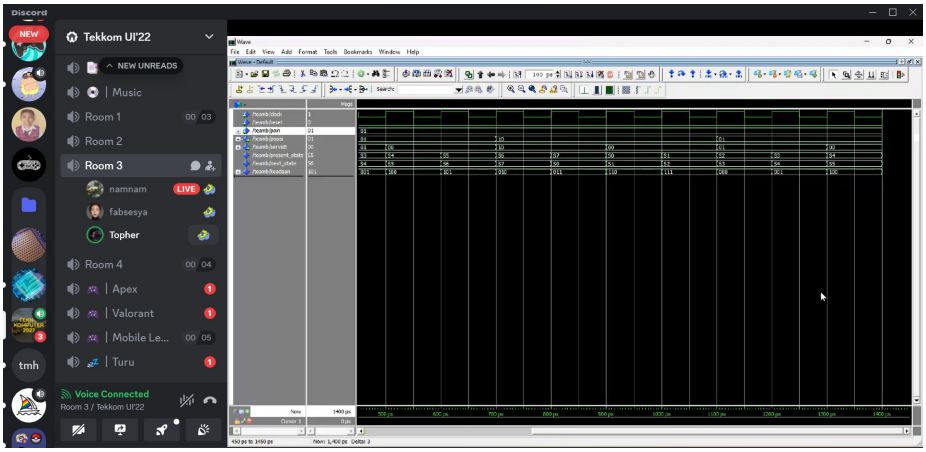
Appendix A: Project Schematic

Final schematic:



Appendix B: Documentation





teamA :

```
library ieee;

use ieee.std_logic_1164.all ;

use IEEE.Numeric_Std.all;


entity teamA is


    port(

        clock : in STD_LOGIC;

        reset : in STD_LOGIC;

        -- 10 : Untuk team A

        -- 01 : Untuk team B

        poin : in std_logic_vector(1 downto 0);

        --Sinyal untuk menentukan posisi pemain

        posisi : out std_logic_vector(1 downto 0);
```

```
        --Pemain mana yang berhak melakukan servis
        servisA : out std_logic_vector(1 downto 0)
    );
end teamA;

architecture arch_servA of teamA is
    --State dari FSM
    type state_types is (S0,S1,S2,S3,S4,S5,S6,S7);
    signal present_state : state_types;
    signal next_state : state_types;
    --keadaan dari FSM dalam biner
    signal keadaan : std_logic_vector (2 downto 0);
begin

    proc : process(clock, next_state, reset)
    begin
        --jika reset nilainya 1 maka akan mengembalikan ke S0
        if(reset = '1') then
            present_state <= S0;
        elsif(rising_edge(clock)) then
            present_state <= next_state;
        end if;
```

```
    end process proc;


    mix_proc : process(present_state, poin)
    begin
    --Output di set ke 00 untuk mencegah eror
        servisA <= "00";
        posisi <= "00";
        case present_state is
        --Keterangan Servis
        --10 = Servis pemain X
        --01 = Servis pemain Y
        --00 = Team A tidak boleh melakukan servis
        --Keterangan Posisi
        --01 = Pemain X dikiri, pemain Y dikanan
        --10 = Pemain X di kanan, pemain X dikiri
            when S0 =>
                keadaan <= "110";
                servisA <= "10";
                posisi <= "10";
                --Jika tim A dapat poin
                if(poin = "10") then
                    next_state <= S1;
```

```
            --Jika tim B dapat point
            elsif(poin = "01") then
                next_state <= S1;
            else
                next_state <= S0;
            end if;
        when S1 =>
            keadaan <= "111";
            servisA <= "10";
            posisi <= "10";
            --Jika tim A dapat poin
            if(poin = "10") then
                next_state <= S2;
            --Jika tim B dapat point
            elsif (poin = "01") then
                next_state <= S2;
            else
                next_state <= S1;
            end if;
        when S2 =>
            keadaan <="000";
            servisA  <= "00";
```

```
                posisi <= "10";
                --Jika tim A dapat poin
                if(poin = "10") then
                    next_state <= S3;
                --Jika tim B dapat point
                elsif (poin = "01") then
                    next_state <= S3;
                else
                    next_state <= S2;
                end if;
            when S3 =>
                keadaan <="001";
                servisA <= "00";
                posisi <= "10";
                --Jika timA dapat poin
                if(poin = "10") then
                    next_state <= S4;

                --Jika tim B dapat point
                elsif(poin ="01") then
                    next_state <= S4;
                else
```

```
                next_state <= S3;
            end if;
        when S4 =>
            keadaan <="100";
            servisA <="01";
            posisi <= "01";
            --Jika tim A dapat point
            if(poin = "10") then
                next_state <= S5;
            --Jika tim B dapat point
            elsif(poin = "01") then
                next_state <= S5;
            else
                next_state <= S4;
            end if;
        when S5 =>
            keadaan <="101";
            servisA <= "01";
            posisi <="01";
            --Jika tim A dapat point
            if (poin = "10") then
                next_state <= S6;
```

```
            --Jika tim B dapat point
            elsif (poin = "01") then
                next_state <= S6;
            else
                next_state <= S5;
            end if;
        when S6 =>
            keadaan <="010";
            servisA <= "00";
            posisi <= "01";
            --Jika tim A dapat point
            if (poin = "10") then
                next_state <= S7;

            --Jika tim B dapat point
            elsif (poin = "01") then
                next_state <= S7;
            else
                next_state <= S6;
            end if;
        when S7 =>
            keadaan <="011";
```

```
                    servisA <= "00";
                    posisi <= "01";
                    --Jika tim A dapat point
                    if (poin = "10") then
                        next_state <= S0;
                    --Jika tim B dapat point
                    elsif (poin = "01") then
                        next_state <= S0;
                    else
                        next_state <= S7;
                    end if;
                end case;
        end process mix_proc;
    end arch_servA;
```

teamB:

```
    library ieee;

    use ieee.std_logic_1164.all ;

    use IEEE.Numeric_Std.all;
```

```
entity teamB is



    port(

        clock : in STD_LOGIC;

        reset : in STD_LOGIC;

        -- 10 : Untuk team A

        -- 01 : Untuk team B

        poin : in std_logic_vector(1 downto 0);

        --Sinyal untuk menentukan posisi pemain

        posisi : out std_logic_vector(1 downto 0);

        --Pemain mana yang berhak melakukan servis

        servisB : out std_logic_vector(1 downto 0)

    );

end teamB;



architecture arch_servB of teamB is
```

```
        --State dari FSM

        type state_types is (S0,S1,S2,S3,S4,S5,S6,S7);

        signal present_state : state_types;

        signal next_state : state_types;

        --keadaan dari FSM dalam biner

        signal keadaan : std_logic_vector (2 downto 0);

    begin



        proc : process(clock, next_state, reset)

        begin

            --jika reset nilainya 1 maka akan mengembalikan ke S0

            if(reset = '1') then

                present_state <= S0;

            elsif(rising_edge(clock)) then

                present_state <= next_state;

            end if;
```

```
        end process proc;



        mix_proc : process(present_state, poin)

        begin

        --Output di set ke 00 untuk mencegah eror

            servisB <= "00";

            posisi <= "00";

            case present_state is

            --Keterangan Servis

            --10 = Servis pemain X

            --01 = Servis pemain Y

            --00 = Team B tidak boleh melakukan servis

            --Keterangan Posisi

            --01 = Pemain X dikiri, pemain Y dikanan

            --10 = Pemain X di kanan, pemain Y dikiri

                when S0 =>
```

```
                keadaan <= "110";

                servisB <= "00";

                posisi <= "10";

                --Jika tim A dapat poin

                if(poin = "10") then

                    next_state <= S1;

                --Jika tim B dapat point

                elsif(poin = "01") then

                    next_state <= S1;

                else

                    next_state <= S0;

                end if;

                when S1 =>

                    keadaan <= "111";

                    servisB <= "00";

                    posisi <= "10";
```

```
                --Jika tim A dapat poin

                if(poin = "10") then

                    next_state <= S2;

                --Jika tim B dapat point

                elsif (poin = "01") then

                    next_state <= S2;

                else

                    next_state <= S1;

                end if;

            when S2 =>

                keadaan <="000";

                servisB  <= "01";

                posisi <= "01";

                --Jika tim A dapat poin

                if(poin = "10") then

                    next_state <= S3;
```

```
            --Jika tim B dapat point

            elsif (poin = "01") then

                next_state <= S3;

            else

                next_state <= S2;

            end if;

        when S3 =>

            keadaan <="001";

            servisB <= "01";

            posisi <= "01";

            --Jika timA dapat poin

            if(poin = "10") then

                next_state <= S4;



            --Jika tim B dapat point

            elsif(poin ="01") then
```

```
                    next_state <= S4;

                else

                    next_state <= S3;

                end if;

            when S4 =>

                keadaan <="100";

                servisB <="00";

                posisi <= "01";

                --Jika tim A dapat point

                if(poin = "10") then

                    next_state <= S5;

                --Jika tim B dapat point

                elsif(poin = "01") then

                    next_state <= S5;

                else

                    next_state <= S4;
```

```
                end if;

            when S5 =>

                keadaan <="101";

                servisB <= "00";

                posisi <="01";

                --Jika tim A dapat point

                if (poin = "10") then

                    next_state <= S6;

                --Jika tim B dapat point

                elsif (poin = "01") then

                    next_state <= S6;

                else

                    next_state <= S5;

                end if;

            when S6 =>

                keadaan <="010";
```

```
                servisB <= "10";

                posisi <= "10";

                --Jika tim A dapat point

                if (poin = "10") then

                    next_state <= S7;



                --Jika tim B dapat point

                elsif (poin = "01") then

                    next_state <= S7;

                else

                    next_state <= S6;

                end if;

            when S7 =>

                keadaan <="011";

                servisB <= "10";

                posisi <= "10";
```

```
                    --Jika tim A dapat point

                    if (poin = "10") then

                        next_state <= S0;

                    --Jika tim B dapat point

                    elsif (poin = "01") then

                        next_state <= S0;

                    else

                        next_state <= S7;

                    end if;

                end case;

        end process mix_proc;

    end arch_servB;
```

Counter_score:

```
    library ieee;
```

```
    use ieee.std_logic_1164.all;



    entity counter_score is

        port (

            clock           : in std_logic;

            reset           : in std_logic;

            count           : in std_logic;

                    seven_segment_1  :  out  std_logic_vector  (6  downto
0):="0111111";

                    seven_segment_2  :  out  std_logic_vector  (6  downto
0):="0111111"

        );

    end counter_score;



    architecture arch_score of counter_score is

        signal digit : integer range 0 to 20;

    begin
```

```
        process (clock, reset, count, digit)

        begin

            if rising_edge(clock) then

                if reset = '1' then

                    seven_segment_1 <= "0111111";

                    seven_segment_2 <= "0111111";

                    digit <= 0;

                else

                    if count = '1' then

                        digit <= digit + 1;

                        case digit is

                            when 0 => seven_segment_2 <= "0000110";

                            when 1 => seven_segment_2 <= "1011011";

                            when 2 => seven_segment_2 <= "1001111";

                            when 3 => seven_segment_2 <= "1100110";

                            when 4 => seven_segment_2 <= "1101101";
```

```
                        when 5 => seven_segment_2 <= "1111101";

                        when 6 => seven_segment_2 <= "0000111";

                        when 7 => seven_segment_2 <= "1111111";

                        when 8 => seven_segment_2 <= "1101111";

                        when 9 => seven_segment_2 <= "0111111";

                        when 10 => seven_segment_2 <= "0000110";

                        when 11 => seven_segment_2 <= "1011011";

                        when others =>

                            seven_segment_1 <= "0111111";

                            seven_segment_2 <= "0111111";

                    end case;

                end if;

            end if;

        end if;

    end process;

end arch_score;
```

TopLevel:

```
library ieee;

use ieee.std_logic_1164.all;



entity TopLevel_entity is

    port (

        POIN : IN STD_LOGIC_VECTOR (1 downto 0);

        CLK, RST : IN STD_LOGIC;

        seven_segment_1_X   : out std_logic_vector (6 downto 0);

        seven_segment_2_X   : out std_logic_vector (6 downto 0);

        seven_segment_1_Y   : out std_logic_vector (6 downto 0);

        seven_segment_2_Y   : out std_logic_vector (6 downto 0);

          POSISI_X, POSISI_Y, SERVX, SERVY : OUT STD_LOGIC_VECTOR (1
downto 0)
```

```
    );

end TopLevel_entity;



architecture main_arch of TopLevel_entity is



    component counter_1 is

        port (

            clock           : in std_logic;

            reset           : in std_logic;

            count           : in std_logic;

            seven_segment_1 : out std_logic_vector (6 downto 0);

            seven_segment_2 : out std_logic_vector (6 downto 0)

        );

    end component;



    component servisX is
```

```
            port(

            --signal untuk clock dan reset

            CLK, RST : IN STD_LOGIC;



            -- signal untuk menentukan poin

            -- 10 : Poin untuk team X

            -- 01 : Pon untuk team Y

            POIN : IN STD_LOGIC_VECTOR (1 downto 0);



               --signal untuk menentukan posisi pemain di lapangan dan
pemain mana yang berhak melakukan servisX

            POSISI, SERVX : OUT STD_LOGIC_VECTOR (1 downto 0)

        );

        end component;



        component servisY is
```

```
            port(

            --signal untuk clock dan reset

            CLK, RST : IN STD_LOGIC;



            -- signal untuk menentukan poin

            -- 10 : Poin untuk team X

            -- 01 : Pon untuk team Y

            POIN : IN STD_LOGIC_VECTOR (1 downto 0);



               --signal untuk menentukan posisi pemain di lapangan dan
pemain mana yang berhak melakukan servis

            POSISI, SERVY : OUT STD_LOGIC_VECTOR (1 downto 0)

        );

        end component;



        signal poin_x   : std_logic;
```

```
        signal poin_y   : std_logic;



    begin

        poin_x <= poin(1);

        poin_y <= poin(0);

        FSM_X   : servisX port map (CLK, RST, POIN, POSISI_X, SERVX);

        FSM_Y   : servisY port map (CLK, RST, POIN, POSISI_Y, SERVY);

              score_X   :   counter_1   port   map   (CLK,   RST,   poin_x,
seven_segment_1_X, seven_segment_2_X);

              score_Y   :   counter_1   port   map   (CLK,   RST,   poin_y,
seven_segment_1_Y, seven_segment_2_Y);



    end main_arch;
```

testbench:

```
    library IEEE;

    use IEEE.Std_logic_1164.all;
```

```
use IEEE.Numeric_Std.all;



entity TopLevel_Testbench is

end TopLevel_Testbench;



architecture bench of TopLevel_Testbench is



    component TopLevel

        port (

            poin : IN STD_LOGIC_VECTOR (1 downto 0);

            clock, reset : IN STD_LOGIC;

            seven_segment_1_A   : out std_logic_vector (6 downto 0);

            seven_segment_2_A   : out std_logic_vector (6 downto 0);

            seven_segment_1_B   : out std_logic_vector (6 downto 0);

            seven_segment_2_B   : out std_logic_vector (6 downto 0);
```

```
                          posisi_A,  posisi_B,  servisA,  servisB  :  OUT
STD_LOGIC_VECTOR (1 downto 0)

            );

        end component;



        constant T       : time    := 20 ns;

        constant max_clk : integer := 43;

        constant count   : integer := 0;

        signal i         : integer := 0;



        signal poin : STD_LOGIC_VECTOR (1 downto 0);

        signal clock, reset : STD_LOGIC;

        signal seven_segment_1_A    : std_logic_vector (6 downto 0);

        signal seven_segment_2_A    : std_logic_vector (6 downto 0);

        signal seven_segment_1_B    : std_logic_vector (6 downto 0);

        signal seven_segment_2_B    : std_logic_vector (6 downto 0);
```

```
          signal posisi_A, posisi_B, servisA, servisB : STD_LOGIC_VECTOR
(1 downto 0);




     begin



          uut: TopLevel port map (poin, clock, reset, seven_segment_1_A,
seven_segment_2_A,    seven_segment_1_B,    seven_segment_2_B,    posisi_A,
posisi_B, servisA, servisB);



         clock_process: process

         begin

             if (i < max_clk) then i <= i + 1;

             clock <= '1';

             wait for T/2;

             clock <= '0';

             wait for T/2;

             i <= i + 1;

             else wait;
```

```
            end if;

        end process;



        reset <= '1', '0' after T/2;



    stimulus: process



        begin

            poin <= "00";

            wait for T;

              assert(posisi_A = "01" and posisi_B = "10" and servisA =
"10" and servisB = "00")

               report "tes gagal pada testbench " & integer'image(1)
severity error;

            poin <= "01";

            wait for T;

              assert(posisi_A = "10" and posisi_B = "10" and servisA =
"01" and servisB = "00")
```

```
                report "tes gagal pada testbench " & integer'image(2)
severity error;

            poin <= "00";

            wait for T;

              assert(posisi_A = "10" and posisi_B = "10" and servisA =
"01" and servisB = "00")

                report "tes gagal pada testbench " & integer'image(3)
severity error;

            poin <= "10";

            wait for T;

            wait;

     end process;



     end bench;
```