UNIVERSITI MALAYSIA PAHANG

BORANG PENGESAHAN STATUS TESIS♦

JUDUL: **CHESS DIGITAL CLOCK**

SESI PENGAJIAN: **2008/2009**

Saya **ROSMIRA BINTI ROSLAN (860925-14-5122)**
(HURUF BESAR)

mengaku membenarkan tesis (Sarjana Muda/~~Sarjana~~ /~~Doktor Falsafah~~)* ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Tesis adalah hakmilik Universiti Malaysia Pahang (UMP).
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan tesis ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. **Sila tandakan (√)

☐	**SULIT**	(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)
☐	**TERHAD**	(Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)
√	**TIDAK TERHAD**	

Disahkan oleh:

_____ (TANDATANGAN PENULIS)

_____ (TANDATANGAN PENYELIA)

Alamat Tetap:

NO. 1, JLN. INDAH 5, SG.JELOK,
43000 KAJANG,
SELANGOR.

NURUL HAZLINA BINTI NOORDIN
(Nama Penyelia)

Tarikh: **12 NOVEMBER 2008**

Tarikh: : **12 NOVEMBER 2008**

CATATAN:
- * Potong yang tidak berkenaan.
- ** Jika tesis ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali tempoh tesis ini perlu dikelaskan sebagai atau TERHAD.
- ♦ Tesis dimaksudkan sebagai tesis bagi Ijazah doktor Falsafah dan Sarjana secara Penyelidikan, atau disertasi bagi pengajian secara kerja kursus dan penyelidikan, atau Laporan Projek Sarjana Muda (PSM).

CHESS DIGITAL CLOCK

ROSMIRA BINTI ROSLAN

UNIVERSITI MALAYSIA PAHANG

"I hereby acknowledge that the scope and quality of this thesis is qualified for the award of the Bachelor Degree of Electrical Engineering (Electronics)"

Signature : _____

Name : NURUL HAZLINA BINTI NOORDIN

Date : 12 NOVEMBER 2008

CHESS DIGITAL CLOCK

ROSMIRA BINTI ROSLAN

This thesis is submitted as partial fulfillment
of the requirements for the award of the Bachelor of
Electrical Engineering (Hons.) (Electronics)

Faculty of Electrical & Electronics Engineering
Universiti Malaysia Pahang

NOVEMBER 2008

Source code for Menu

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity first is
port (  clk, reset : in std_logic;
        swA : in std_logic;
         swB : in std_logic;
         swC : in std_logic;
         outA : out std_logic;
        outB : out std_logic;
        outC : out std_logic      );
end first;
architecture Behavioral of first is
begin
process (swA,swB,swC,reset,clk) begin

if reset = '1' then outA <= '0'; outB <= '0' ; outC <= '0';
else if (clk'event and clk='1') then
          if swA = '1' then outA <= '1'; outB <= '0' ; outC <= '0';
          else if swB = '1' then outB <= '1'; outA <= '0' ; outC <= '0';
          else if swC = '1' then outC <= '1'; outA <= '0' ; outB <= '0';
end if;
end if;
end if;
end if;
end if;
end process;
end Behavioral;
```

Source code for Blitz

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity b is
Port(  clk, reset          : in std_logic;
       count      : in std_logic;
       s_2, m_2   : buffer integer range 0 to 60;
        j_2        : buffer integer range 0 to 10          );
end b;
architecture Behavioral of b is
signal temp_s2, temp_m2 : integer range 0 to 60;
signal temp_j2              : integer range 0 to 10;
begin
process ( clk,reset) begin
       if reset = '1' then temp_s2 <= 0;
       else if (clk'event and clk='1') then
              if count = '1' then temp_s2<= temp_s2-1;
              if temp_s2 = 0 then temp_s2 <=59; temp_m2<= temp_m2-1;
              if temp_m2 =0 then temp_m2 <=59; temp_j2<= temp_j2-1;
end if;
end if;
end if;
end if;
end if;
end process;
       s_2 <= temp_s2 ;
       m_2 <= temp_m2 ;
       j_2 <= temp_j2 ;
end Behavioral;
```

Source code for Standard (1)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity se is
Port(  clk, reset            : in std_logic;
          count              : in std_logic;
          infopponent        : in std_logic;
          s_3, m_3           : buffer integer range 0 to 60 ;
           j_3               : buffer integer range 0 to 10            );
end se;
architecture Behavioral of se is
signal temp_s3, temp_m3  : integer range 0 to 60  ;
signal temp_j3 : integer range 0 to 10;
begin
process ( clk,reset) begin
          if reset = '1' then
          temp_s3 <= 0;
          temp_m3 <= 30;
          temp_j3 <= 1;
          else if (clk'event and clk='1') then
          if count = '1' then temp_s3<= temp_s3-1;
          if infopponent = '1' then temp_s3 <= temp_s3+30;
          else if temp_s3 = 0 then temp_s3 <= 59; temp_m3 <= temp_m3-1;
          else if temp_m3 = 0 then temp_m3 <= 59; temp_j3 <= temp_j3-1;
          else if temp_j3 = 0 then temp_j3 <= 9;
          else if temp_s3 > 59 then temp_s3 <= temp_s3-60; temp_m3 <= temp_m3
+1;
          else if temp_m3 > 59 then temp_m3 <= temp_m3-60; temp_j3 <= temp_j3
+1;
end if;
end if;
```

```
end if;
end if;
end if;
end if;
end if;
end if;
end if;
end process;
        s_3 <= temp_s3 ;
         m_3 <= temp_m3 ;
        j_3 <= temp_j3 ;
end Behavioral;
```

Source code for Standard (2)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity sg is
Port(  clk, reset            : in std_logic;
       count                 : in std_logic;
        infopponentg  : in std_logic;
       trigger               : buffer integer range 0 to 41;
       s_4, m_4          : buffer integer range 0 to 60 ;
        j_4                  : buffer integer range 0 to 10 );
end sg;
architecture Behavioral of sg is
signal temp_s4, temp_m4  : integer range 0 to 60  ;
signal temp_j4 : integer range 0 to 10;
signal temp_trigger : integer range 0 to 41;
begin
process ( clk,reset) begin
       if reset = '1' then
       temp_s4 <= 0;
       temp_m4 <= 30;
       temp_j4 <= 1;
       else if (clk'event and clk='1') then
              if count = '1' then temp_s4<= temp_s4-1;
              if infopponentg = '1' then temp_s4 <= temp_s4+30;

temp_trigger <= temp_trigger + 1;
       else if temp_trigger = 40 then temp_m4 <= temp_m4 + 40;
       else if temp_s4 = 0 then temp_s4 <= 59; temp_m4 <= temp_m4-1;
       else if temp_m4 = 0 then temp_m4 <= 59; temp_j4 <= temp_j4-1;
       else if temp_j4 = 0 then temp_j4 <= 9;
```

```
        else if temp_s4 > 59 then temp_s4 <= temp_s4-60; temp_m4 <= temp_m4
+1;
                        else if temp_m4 > 59 then temp_m4 <= temp_m4-60; temp_j4
<= temp_j4 +1;
end if;
end if;
end if;
end if;
end if;
end if;
end if;
end if;
end if;
end if;
end process;
        s_4 <= temp_s4 ;
         m_4 <= temp_m4 ;
        j_4 <= temp_j4 ;
        trigger <= temp_trigger ;
end Behavioral;
```

Source code for port map

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity standard is
port ( clk_T, reset_T  : in std_logic;
          swA_T          : in std_logic;
          swB_T          : in std_logic;
          swC_T          : in std_logic;
          info_T         : in std_logic;
          infog_T        : in std_logic;
          trigger_T      : buffer integer range 0 to 41;
          s_2T, m_2T     : buffer integer range 0 to 60;
           j_2T          : buffer integer range 0 to 10;
           s_3T, m_3T    : buffer integer range 0 to 60;
           j_3T          : buffer integer range 0 to 10;
          s_4T, m_4T     : buffer integer range 0 to 60;
           j_4T          : buffer integer range 0 to 10);
end standard;
architecture structural of standard is
component first
port (   clk, reset : in std_logic;
          swA : in std_logic;
         swB : in std_logic;
         swC : in std_logic;
         outA : out std_logic;
        outB : out std_logic;
        outC : out std_logic              );
end component;
component b
port ( clk, reset : in std_logic;
          count      : in std_logic;
```

```
        s_2, m_2       : buffer integer range 0 to 60;
         j_2          : buffer integer range 0 to 10);
end component;
component se
port ( clk, reset : in std_logic;
        count      : in std_logic;
        infopponent : in std_logic;
         s_3, m_3       : buffer integer range 0 to 60;
         j_3         : buffer integer range 0 to 10);
end component;
component sg
port ( clk, reset : in std_logic;
        count      : in std_logic;
        infopponentg: in std_logic;
        trigger    : buffer integer range 0 to 41;
        s_4, m_4       : buffer integer range 0 to 60;
         j_4         : buffer integer range 0 to 10);
end component;
signal bus_1: std_logic;
signal bus_2 :std_logic;
signal bus_3 :std_logic;


begin
U1 : first port map ( clk=>clk_T,
                      reset=>reset_T,
                      swA=>swA_T,
                      swB=>swB_T,
                       swC=>swC_T,
                      outA=>bus_1,
                      outB => bus_2,
                      outC => bus_3);
U2 : b port map ( clk=>clk_T,
                  reset=>reset_T,
```

```
                count=>bus_1,
                s_2=>s_2T,
                m_2=>m_2T,
                j_2=>j_2T );
U3 : se port map ( clk=>clk_T,
                reset=>reset_T,
                count=>bus_2,
                infopponent=>info_T,
                s_3=>s_3T,
                 m_3=>m_3T,
                 j_3=>j_3T);
U4 : sg port map ( clk=>clk_T,
                reset=>reset_T,
                count=>bus_3,
                infopponentg=>infog_T,
                 trigger=>trigger_T,
                s_4=>s_4T,
                m_4=>m_4T,
                j_4=>j_4T );
end structural;
```

Signature : _____

Author : ROSMIRA BINTI ROSLAN

Date : 12 NOVEMBER 2008

To my beloved father and mother

Mr. Haji Roslan Bin Haji Mohd Noor

Mrs. Hajjah Nor Hayati Binti Haji Ramlan

ACKNOWLEDGEMENT

First, I want to express my grateful to ALLAH s.w.t for blessing and giving me enough courage in achieving the objectives of my final year project (PSM).

I would like to express my sincere thanks to my supervisor, Mrs. Nurul Hazlina binti Noordin for her sincere support and supervising this project. Not to forget, the special thank to all UMP lecturers and technicians who had helped directly or indirectly.

My appreciates to Mr. Reza Ezuan bin Samin, Final Year Project Coordinator who acted promptly to the student's problem.

I also like to acknowledge the contributions of my colleagues, especially final year students of Electric and Electronics Engineering 2008, which has openhandedly and kindly assisted and supported me to make this project successful.

Last, but not least, I am always indebted to my dearest family, for their love and prey on me throughout this project. I always appreciate and treasured the great cooperation, kindheartedness and readiness to share worth experiences that have been shown by them.

ABSTRACT

The purpose of my project is to design and implement of the Chess Digital Clock. The project employs ISE software (ISE Design Suite 10.1) and implementation on Field-Programmable Gate Arrays (FPGAs) Xilinx board. It is a new technique for testing the interconnects of an arbitrary design mapped into an FPGA. Field-Programmable Gate Arrays (FPGAs) have become one of the key digital circuit implementation media over the last decade. A crucial part of their creation lies in their architecture, which governs the nature of their programmable logic functionality and their programmable interconnect. The experimental result on various benchmarks using the ISE software is on its simulation. The software is designed using VHDL code. Digital circuit modeling with hardware description languages (HDLs) is the key to modern design of integrated circuits (ICs). The state-of-the-art technique of designing complex digital systems and integrated circuits is to apply an HDL-based CAD approach, in which a high-level, text-based, abstract description of the circuit is created, then synthesized to a hardware implementation in a selected technology, and finally verified for its functionality and timing.

ABSTRAK

Tujuan projek saya ini adalah untuk menghasilkan dan melaksanakan jam catur digital. Projek ini mengunakan perisian ISE (ISE Design Suite 10.1) dan pelaksanaan diatas panel *Field-Programmable Gate array* (FPGAs) Xilinx. Ini merupakan teknik baru untuk menguji plan pnyambungan dalaman yang secara rawak yang dipetakan ke dalam FPGA. *Field-Programmable Gate Array* (FPGAs) telah menjadi salah satu kunci pelaksanaan litar digital media selama sepuluh tahun. Satu bahagian penting dari penciptaan alat ini terletak pada plannya, yang di mana sifat dan fungsi penyambungan dalaman litar. Percubaan pada berbagai hasil yang menggunakan perisian ini adalah pada simulasi. Perisian yang dirangka ini adalah dengan menggunakan kod VHDL. Model litar digital dengan *hardware description languages* (HDLs) merupakan kunci untuk rancangan litar terpadu (ICs) yang baru. Teknik untuk merancang sistem digital yang sukar dan litar terpadu adalah untuk menyerapkan ke atas HDL berdasarkan pendekatan CAD, yang tinggi, berdasarkan teks, abstrak keterangan mengenai litar yang dibuat, dan kemudian sintesiskan ke atas model yang dipilih dalam pelaksanaan teknologi, dan akhirnya disahkan mengikut fungsi dan waktu.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ASSP – Application-Specific Standard Product

CLB – Configurable Logic Block

FPGA – Field-Programmable Gate Array

HDL – Hardware Description Language

LUT – Look-Up Table

MXE – ModelSim Xilinx Edition

VHDL – Very High-speed integrated circuit Hardware Description Language

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

This chapter focuses on the methodologies for the development and implementation of the Chess Digital Clock project. The project includes the ISE software coding (ISE Design Suite 10.1) and FPGA Xilinx board.

In very general terms, coding can be said that the communication between human and technologies. One example of an application is a Chess Digital Clock. This application (software coding) executes on FPGA board (technology) that can support that application.

This chapter also explains the overview of Chess Digital Clock project, the objectives of the project, project scopes and thesis outline.

1.2 Overview of Chess Digital Clock

Sport is an activity that is governed by a set of rules or customs and often engaged in competitively. Sports commonly refer to activities where the physical capabilities of the competitor are the sole or primary determiner of the outcome (winning or losing), but the term is also used to include activities such as mind sports (a common name for some card games and board games with little to no element of chance) and motor sports where mental acuity or equipment quality are major factors. [6]

Besides casual games without exact timing, chess is also played with a time control, mostly by club and professional players. If a player's time runs out before the game is completed, the game is automatically lost. The timing ranges from long games played up to seven hours to shorter rapid chess games lasting usually 30 minutes or one hour per game. Even shorter is blitz chess with a time control of three to fifteen minutes for each player and bullet chess (under three minutes). [7]

The development of this Chess Digital Clock consists of two parts. Project part one which is concentrate on software coding. The software that is used in this project is ISE software in VHDL code. The software coding started with ISE 6.0 and its simulation done with MXE (ModelSim Xilinx Edition). After the several months the using software of ISE 6.0 changes to the latest version which is ISE 10.0. This latest version of ISE software is much easier in the simulation. ISE Design Suite 10.1 includes the Integrated Software Environment (ISE), ChipScope Pro, Xilinx Embedded Development Kit (EDK), DSP Tools (including AccelDSP and System Generator), and Plan Ahead/PlanAhead Lite. It also describes how to use Xilinx online documentation.

The second part of the project is concentrate in implementing the software to the FPGA board. FPGA board that used in the project is FPGA Xilinx board. FPGA requires user hardware programming to perform the desired operation. Xilinx Spartan FPGAs are ideal for low-cost, high-volume applications and are targeted as replacements for fixed-logic gate arrays and ASSP products such as bus interface chip sets.

Figure 1.1 shows the methodology of the Chess Digital Clock. The first step is to design the digital concept. The designer should familiar with the module which has to be design. This is the part of starting the coding. The design entry where is the design is created and entered into the computer in the form of an HDL source code, using a design entry tool. After all modules have been completely designed, the final design is portmapping. Port-map is the place where is the combination of all modules. It is a method for associating signals with their respective ports.

After a design is generated, the resulting VHDL code may be simulated for the behavior of the designed circuit using a VHDL (very high-speed integrated circuit hardware description language) simulation tool. The VHDL code generated from design entry tool is passed to the synthesis module, converting the code to a logic netlist file.

The netlist obtained from the synthesis tool may be verified for design correctness using a functional simulation tool. The netlist file is converted to a physical design in the target implementation technology. Where each logic function is mapped (implemented) to the logic elements available in the target chip.

The physical layout obtained from implementation process can be simulated to verify the design, but this time, with timing information.

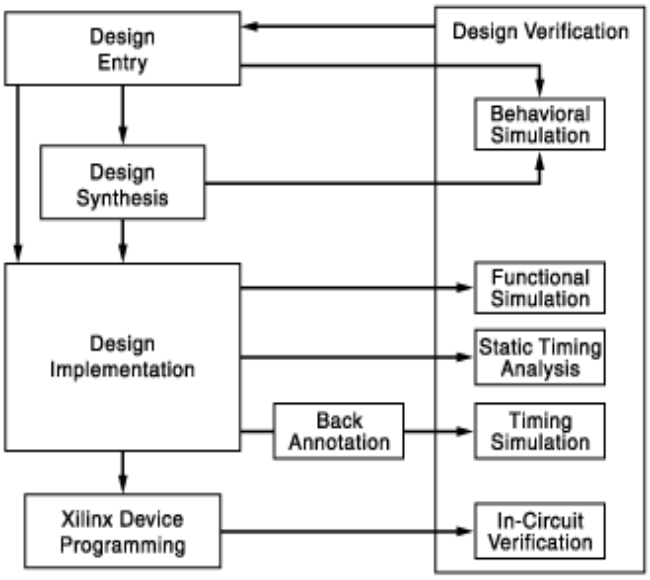

Figure 1.1 : General Design Flow

1.3 Project Objectives

The overall target of the whole project is to allow people playing chess interestingly. However the objectives of the project are display the three time controls in numbers and implement the design onto FPGA board.

1.4 Project Scopes

This project is concentrates on the time of playing chess according to the type of playing through the display Chess Digital Clock. The ISE 10.0 software is used to construct the project that contain three different setting time of playing chess. The successful in simulation of the design is one of the scopes of the project. To achieve the objective of the project, the three different setting times of playing chess need to

implement in the FPGA Xilinx board. The product of the project help the user of the digital clock go easier while playing chess.

1.5 Thesis Outline

Chapter 1 focuses on the methodologies for the development and implementations of the user of the Chess Digital Clock. It gives a brief the steps and the purpose of the Chess Digital Clock.

Chapter 2 explains the background of the ISE software and FPGA Xilinx board and the relationship of each part in develop Chess Digital Clock. The concept of the software and the FPGA board are the two essentials concept as a guide to the construction of Chess Digital Clock. In this chapter also explain the three rules of playing chess for designing the system.

Chapters 3 explain and discuss the process of using and control the chess clock. It discusses the brief review of how the Chess Digital Clock works and the algorithm of the user.

Chapter 4 discusses all the result obtained and the limitation of the project. All discussions are concentrating on the result and the overall performance of the project

Chapter 5 discusses the conclusion of development of whole the system. This chapter also discusses the problem and the recommendation for this project and overall of the system for the future development or modification. Besides that, this chapter also explains about the costing and commercialization.

CHAPTER 2

BACKGROUND

2.1 Background

This chapter explains the background of ISE Software and FPGA Xilinx board and the relationship of each part in develop Chess Digital Clock. These are the main tools as a guide to the development of the Chess Digital Clock. Else in this chapter, also explain about the three of the chess time control game. These rules of the chess game must be considered to design the concept of the Chess Digital Clock.

2.2 ISE software

WebPack is a shell script for automatically packing Web sites by shrinking them without affecting their functionality or appearance. It is also useful for lossless shrinking image collections and locating corrupt files. It works by stripping unnecessary information and optimizing the compression of images, and by removing comments/whitespace from HTML, using readily-available tools. [1]

A Webpack is a packaged service to make top quality web sites accessible to small businesses at minimal cost. A Webpack sites contain everything that a small business requires to project a professional image online. Individual Webpack ISE

modules give you the ability to tailor the design environment to your chosen PLDs as the preferred design flow. In general, the design flow for FPGAs and CPLDs is identical. You can choose wether to enter the design in schematic form or in HDL, such as VHDL, Verilog or ABEL. The design can also comprise of a mixture of schematic diagrams and embedded HDL symbols. There is also a facility to create state machines in a diagrammatic form and let the software tools generate optimized code from a state diagram. WebPACK ISE software incorporates a Xilinx version of the ModelSim simulator from Model Technology (a Mentor Graphics company), referred to as MXE (ModelSim Xilinx Edition). This powerful simulator is capable of simulating functional VHDL before synthesis, or simulating after the implementation process of timing verification. WebPACK ISE software offers an easy-to-use GUI to visually create a test pattern. A testbench is then generated and compiled into MXE, along with the design under test. The flow diagram below shows the similarities and differences between CPLD and FPGA software flows. [2]

Xilinx programmable logic solutions help minimize risks for electronic equipment manufacturers by shortening the time required to develop products. Webpack ISE design software offers a complete design suite based on the Xilinx ISE series software.

2.2.1 ISE Design Suite 10.0

For this project, the design is enter in HDL (Hardware Description Language), which is VHDL code.

VHDL is an industry standard language for modeling digital circuits. It is intended for design documentation and simulation. The basics of VHDL are including VHDL Design Unit, VHDL Data Object and Types, and VHDL operators.

A typical VHDL module consists of library declarations, an entity, and architecture. The library declarations are needed to tell the compiler which packages are required.

Webpack ISE software incorporates a Xilinx version of the ModelSim simulator from Model Technology (a Mentor Graphics company), referred to as MXE (ModelSim Xilinx Edition). This powerful simulator is capable of simulating functional VHDL before synthesis, or simulating after the implementation process for timing verification.

Individual can design and verify the unique circuits in Xilinx programmable devices much faster than by choosing traditional methods such as mask-programmed, fixed logic gate arrays.

2.3 Field Programmable Gate Arrays (FPGAs)

Short for **F**ield-**P**rogrammable **G**ate **A**rray, a type of logic chip that can be programmed. An FPGA is similar to a PLD, but whereas PLDs are generally limited to hundreds of gates, FPGAs support thousands of gates. They are especially popular for prototyping integrated circuit designs. Once the design is set, hardwired chips are produced for faster performance. [3]

A field-programmable gate array is a semiconductor device containing programmable logic components called logic blocks, and programmable interconnects. Logic blocks can be programmed to perform the function of basic logic gates such as AND, and XOR, or more complex combinational functions such as decoders or mathematical functions. In most FPGAs, the logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory.

Field-Programmable Gate Arrays (FPGAs) have become one of the key digital circuit implementation media over the last decade. A crucial part of their creation lies in their architecture, which governs the nature of their programmable logic functionality and their programmable interconnect. FPGA architecture has a dramatic effect on the quality of the final device's speed performance, area efficiency, and power consumption.

2.3.1 FPGA Xilinx board

The ISE™ design flow comprises the following steps: design entry, design synthesis, design implementation, and Xilinx® device programming. Design verification, which includes both functional verification and timing verification, takes places at different points during the design flow. [5]

In this project, using FPGA Xilinx board is the best choose. This is the latest upgrade product of Xilinx. Xilinx is also actively developing breakthrough technology that will enable the hardware in Xilinx-based systems to be upgraded remotely over any kind of network – including the Internet – even after the equipment has been shipped to a customer.

Xilinx Spartan FPGAs are ideal for low-cost, high-volume applications and are targeted as replacements for fixed-logic gate arrays and ASSP (Application-Specific Standard product) products such as bus interface chip sets.

2.4 Time Control

Chess clock are really two connected clocks. While player A is thinking, his clock is running and the player B clock is stopped. Once the player A makes a move and he hit the clock, which stop his clock and starts the player B clock. There is only one clock running at a time because, each player gets their own separate amount of time. This is to allow for the fact that some moves take only a few seconds to play, while others might take several minutes, depending on the complexity of the position.

The term time control refers to the amount of time each player has to make some or all moves during a game. The three different kinds of time control are Blitz, Standard (1) and Standard (2). The different names distinguish the different maximum duration of a game.

2.4.1 Blitz

In Blitz chess, each player gets a fixed amount of time for the entire game. For example of Blitz kind of time control is five minutes per side game. Each player gets five minutes on their clock, so the time might be set to 4:55 on each side.

As the player with White is thinking about the first move, his clock is running. After a few seconds, he makes the move and hits his clock. This starts his

opponent's clock. He can take as much time as he wants to for each move. Then he hits his clock. This goes on, back and forth.

In Blitz, when the five minutes is up, the person’s clock is the first one out of time, he lose, regardless of the position on the board unless his opponent has insufficient material to mate. Blitz chess is very exciting, and lots of fun for social games and one-day tournaments.

2.4.2 Standard (1)

Most serious international tournaments, and many amateur tournaments, use a Standard (1) (quota system) for time controls. As in Blitz, each player gets their own time, and need to finish their game in the time that allocated. The difference is that the player will be given more time to continue playing. That is mean some extra time will be added to their time when the game is running.

Each player also gets a fixed amount of time for the entire game. But when the players hit his clock after make their move, then their time will get 30 seconds extra time to they continue the game. Every time when they hit their clock, the will be added 30 seconds at their clock. This make the game held longer than Blitz. The rest of the game runs the same with the Blitz method.

2.4.3 Standard (2)

The newest wrinkle in time controls is standard (2). This fact holds a patent on a mechanical chess clock which provides an increment to an adopted a digital increment clock as the standard clock for the game.

This is an advanced method of Standard (1) of time control. Players also get to carry forward any extra time. In Standard (1), after a player makes a move and they hit the clock, his time will be added 30 seconds extra time. This rule is same in Standard (2) method.

The different is when the player had make 40 moves and that is mean they hit the clock for 40 times, this time the extra time that will be added is 40 minutes. So, the players get longer time to complete their game. After that, the clock will continue the same condition which is 30 seconds once their hit the clock. The players will realize that they both have more time that they started with.

CHAPTER 3

DESIGN

3.1 Introduction

Before looking at the detail of the software implementation in this chapter, it is best to begin with brief review of how the Chess Digital Clock works with the type of time playing chess game.

There are three different types of time playing chess game which are Blitz, Standard (1) and Standard (2). There are extra designs for this project instead of for the three time control.

3.2 Flowchart and coding design

The flowchart in Figure 3.1 explains for the Menu design. If the reset is ‘1’, then the output is to be ‘0’ for outA, outB and outC. When if clock is ‘1’ (positive edge), and switch A (swA) is ‘1’ then the outA is ‘1’. OutA is assigned for Blitz type of playing. Same goes for switch B (swB) and switch C (swC), where if outB is ‘1’, then the game follow the Standard (1) and outC follow the Standard (2) type of playing chess. If there is no condition, the processes go for the previous step.

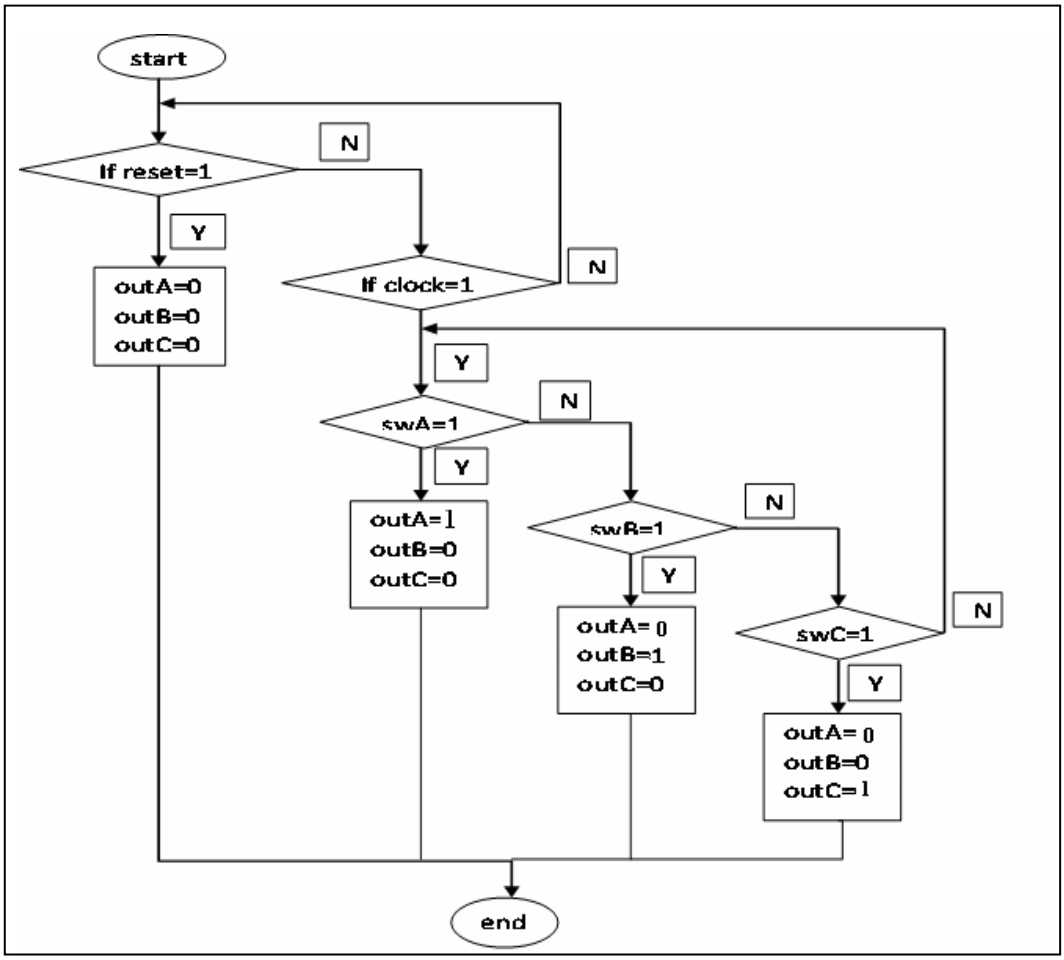

Figure 3.1: Menu Flowchart

Figure 3.2 shows the important part in doing the source code for the Menu design. The process that took part is the three inputs of switches, reset and internal clock. In the programming, it is states the three conditions excluding reset.

```
begin
process (swA,swB,swC,reset,clk) begin
if reset = '1' then outA <= '0'; outB <= '0' ; outC <= '0';
else if (clk'event and clk='1') then
if swA = '1' then outA <= '1'; outB <= '0' ; outC <= '0';
else if swB = '1' then outB <= '1'; outA <= '0' ; outC <= '0';
else if swC = '1' then outC <= '1'; outA <= '0' ; outB <= '0';
```

Figure 3.2: Important Part in Menu Design

Blitz time playing known as rapid game. The flow of the system is shown in Figure 3.3. In Blitz game, the players are allocated with a time of playing the game. They need to finish their game in the period of time. When the seconds (temp_s2) start counting downward until it reach zero, the minutes (temp_m2) will minus one to give the seconds be 59 seconds. The process is continuing similar to the previous process. There goes to be same for the minutes when it reach zero. The hour (temp_j2) will minus by one and the minutes get extra 59 minutes. The game is end when the time is zero which is meant no time left to play the game.

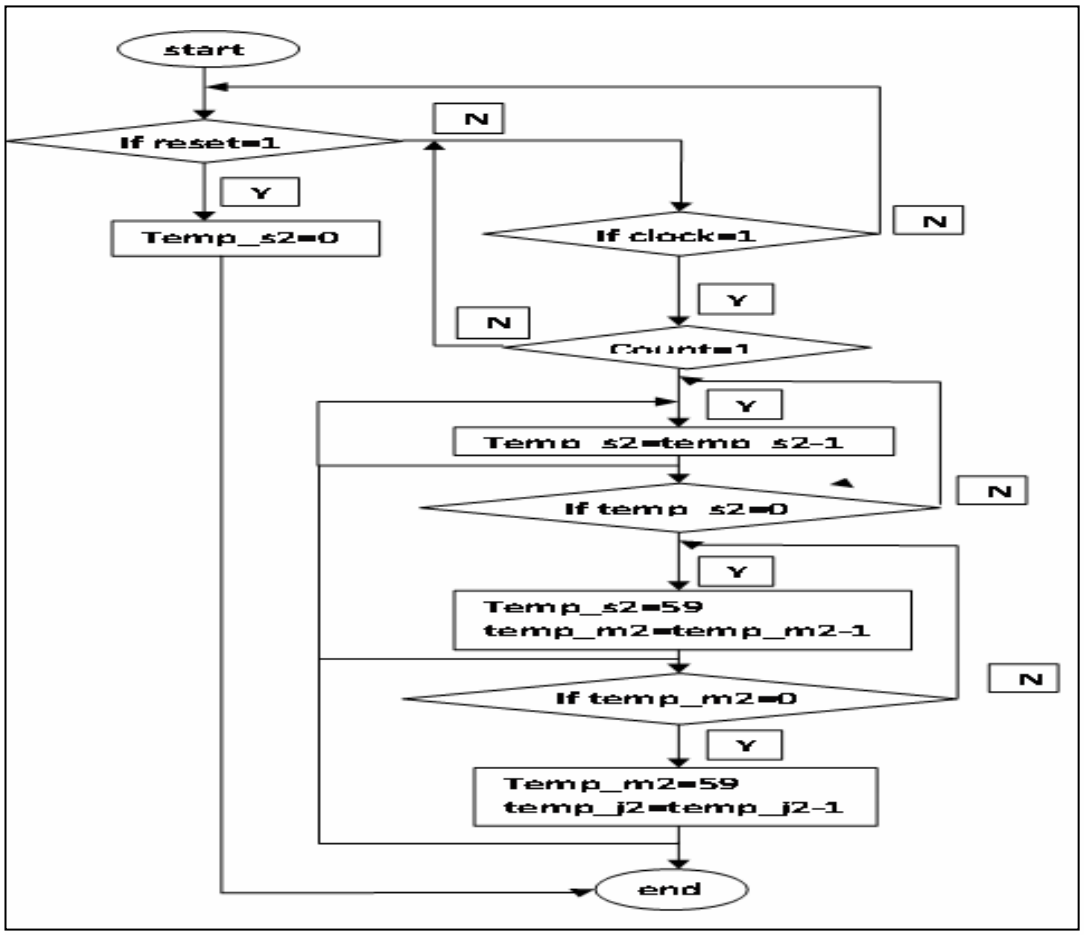

Figure 3.3: Blitz Flowchart

Blitz is one of the outputs of the three conditions which have explained in the previous design. In Figure 3.4, figure out that after the clock give the positive edge and count is declared ‘1’, the time will counting downward. All the conditions at seconds, minutes and hours are change by the instructions in the program.

```
begin
process ( clk,reset) begin
if reset = '1' then temp_s <= 0;
else if (clk'event and clk='1') then
if count = '1' then temp_s<= temp_s-1;
if temp_s = 0 then temp_s <=59; temp_m<= temp_m-1;
if temp_m =0 then temp_m <=59; temp_j<= temp_j-1;
```

Figure 3.4: Important Part in Blitz Design

This standard of playing chess game is similar to Blitz game. The different is when a player presses his button to allow his opponent to make a move. Once the player hit his button, his time will be added 30 seconds extra time. The rest of the game is flowing like usual. The flow of these conditions can see in Figure 3.5.

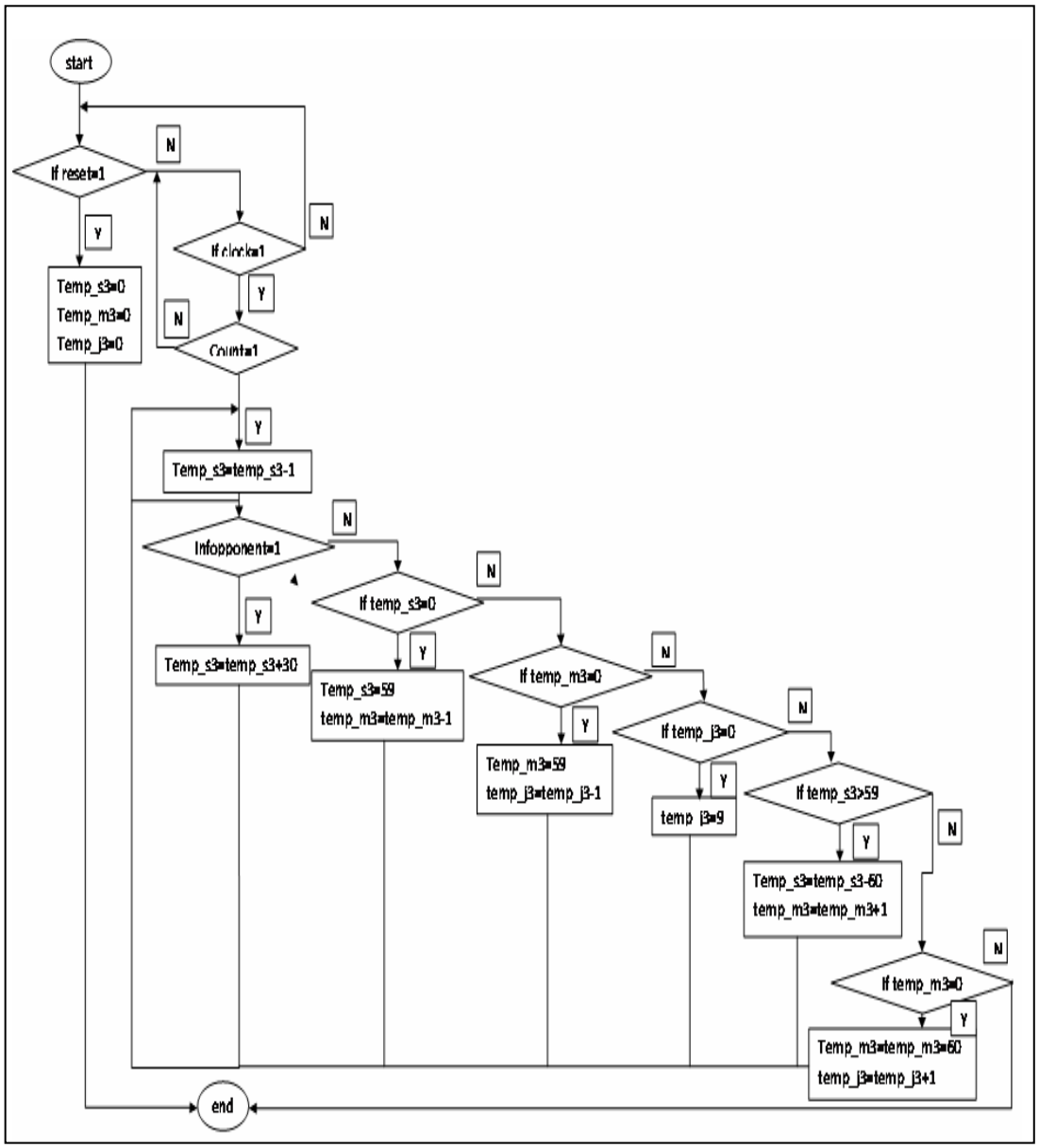

Figure 3.5: Standard (1) Flowchart

For the example in Figure 3.6, shows that the time for the player is 1 hour and 30 minutes. After the clock and count have declared, the time is starting count down. In the system, temp_s3 is assigned for the seconds, starting the operation. If the infopponent in the program is ‘1’, then the seconds will be added by 30. The

program divided into two conditions for the time place. There are if it reaches 0 and if it exceeds 59.

```
begin
process ( clk,reset) begin
if reset = '1' then
temp_s3 <= 0;
temp_m3 <= 30;
temp_j3 <= 1;
else if (clk'event and clk='1') then
if count = '1' then temp_s3<= temp_s3-1;
if infopponent = '1' then temp_s3 <= temp_s3+30;
else if temp_s3 = 0 then temp_s3 <= 59; temp_m3 <= temp_m3-1;
else if temp_m3 = 0 then temp_m3 <= 59; temp_j3 <= temp_j3-1;
else if temp_j3 = 0 then temp_j3 <= 9;
else if temp_s3 > 59 then temp_s3 <= temp_s3-60; temp_m3 <= temp_m3 +1;
else if temp_m3 > 59 then temp_m3 <= temp_m3-60; temp_j3 <= temp_j3 +1;
```

Figure3.6: Important Part in Standard (1) Design

Standard (2) time playing is an advanced from the Standard (1) behavior of time playing. When the player had hit his button for 40 times, he will get 40 minutes extra time. The flowchart in Figure 3.7 is the design flow for the Standard (2) time control game.

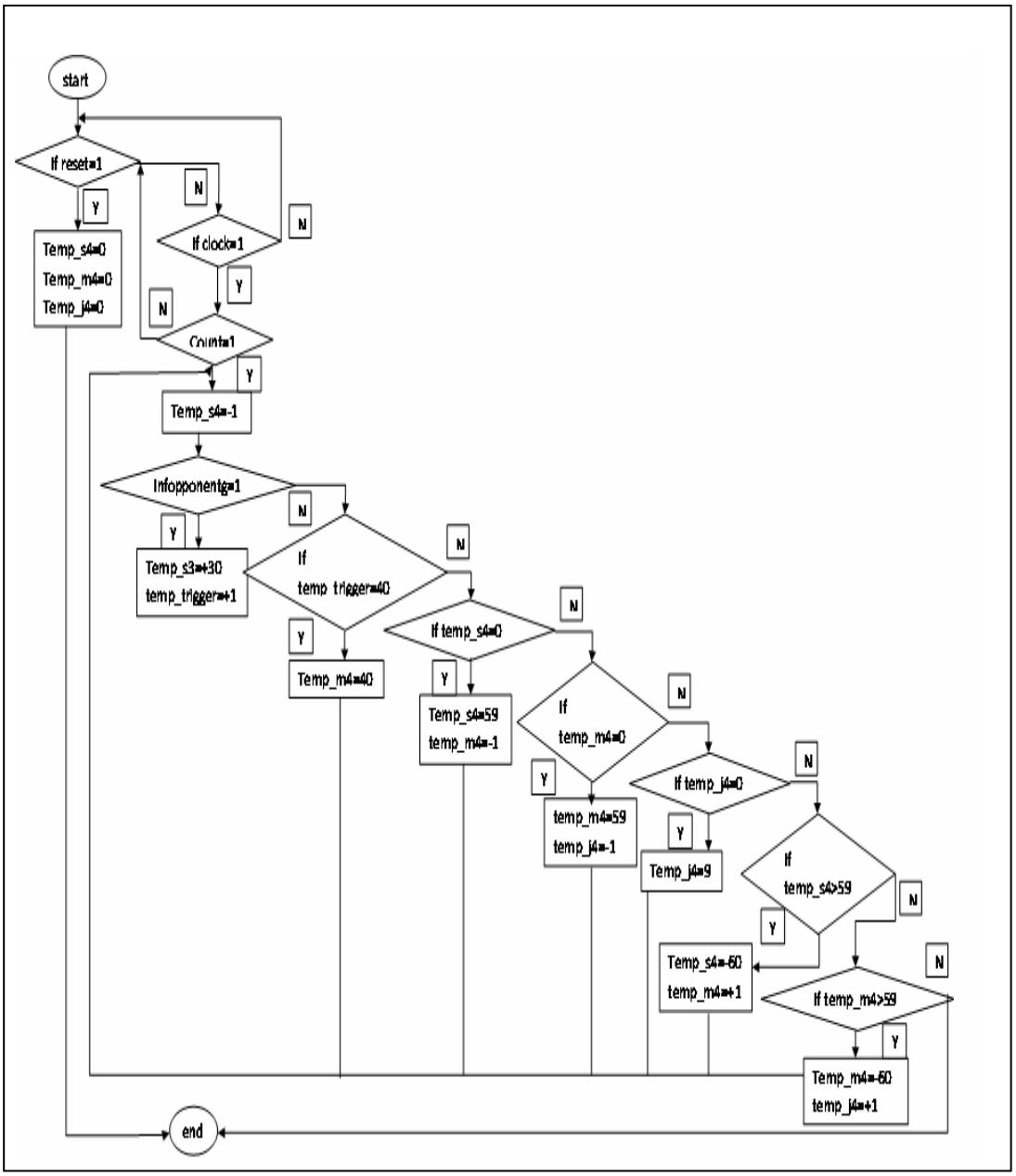

Figure 3.7: Standard (2) Flowchart

As indicate in Figure 3.8, the time that allocated to the player also in one hour and a half. Seconds starting count down when the count is ‘1’. The condition of infopponentg same as the condition of infoopponent in the previous program. Temp_trigger in the programming, will be added by one, in there is any input at infopponentg.

```
begin
process ( clk,reset) begin
if reset = '1' then
temp_s4 <= 0;
temp_m4 <= 30;
temp_j4 <= 1;
else if (clk'event and clk='1') then
if count = '1' then temp_s4<= temp_s4-1;
if infopponentg = '1' then temp_s4 <= temp_s4+30;
temp_trigger <= temp_trigger + 1;
else if temp_trigger = 40 then temp_m4 <= temp_m4 + 40;
else if temp_s4 = 0 then temp_s4 <= 59; temp_m4 <= temp_m4-1;
else if temp_m4 = 0 then temp_m4 <= 59; temp_j4 <= temp_j4-1;
else if temp_j4 = 0 then temp_j4 <= 9;
else if temp_s4 > 59 then temp_s4 <= temp_s4-60; temp_m4 <= temp_m4 +1;
else if temp_m4 > 59 then temp_m4 <= temp_m4-60; temp_j4 <= temp_j4 +1;
```

Figure 3.8: Important Part in Standard (2) Design

According to the Figure 3.9, port-map is the part where all the combinations of modules are meeting. It is also called top level module. This level is used to describe the synthesis system that takes this level as input. It is connecting up the two components to each other and to the primary ports of the multiplexer.

The ports in a component declaration must usually match the ports in the entity declaration one-for-one. The component declaration defines the names, order, mode and types of the ports to be used when the component is instanced in the architecture body. Instancing a component implies making a local copy of the corresponding design entity. A component is declared once within any architecture, but may be instanced any number of times.

```
begin
U1 : first port map ( clk=>clk_T,
                      reset=>reset_T,
                      swA=>swA_T,
swB=>swB_T,
                        swC=>swC_T,
                        outA=>bus_1,
                        outB => bus_2,
                        outC => bus_3);
U2 : b port map ( clk=>clk_T,
                     reset=>reset_T,
                    count=>bus_1,
                    s_2=>s_2T,
                    m_2=>m_2T,
                    j_2=>j_2T );
U3 : se port map ( clk=>clk_T,
                      reset=>reset_T,
                      count=>bus_2,
                      infopponent=>info_T,
                      s_3=>s_3T,
                      m_3=>m_3T,
                      j_3=>j_3T);
U4 : sg port map ( clk=>clk_T,
                      reset=>reset_T,
                     count=>bus_3,
                     infopponentg=>infog_T,
                     trigger=>trigger_T,
                     s_4=>s_4T,
                    m_4=>m_4T,
                    j_4=>j_4T );
```

Figure 3.9: Important part in Port Map Design

CHAPTER 4

RESULT AND DISCUSSION

4.1 Introduction

This chapter discusses all the results obtained and the limitation of the project. All discussions concentrate on the result and performance of the overall project.

4.2 RTL Schematics

After complete the certain coding program, need to run the synthesize to check any syntax error in the command. If there has no error occur, the RTL schematic can be view. The schematic can be view in the top view and bottom view. For the bottom view, it shows all the logic gates that are use for the system while the top view is the image of the integrate circuit (ICs). By dragging the cursor at any component at the bottom view, the behavior of the component will appear. As well as if click at any line, the line will highlight and viewer wills knows where the line does connected to. Follows are models that gets from the synthesize of those programs.

Figure 4.1 shows the top and bottom view model for Menu. In the model, there are five inputs and three outputs for Menu. Menu’s design is using only several components to get a complete schematic.

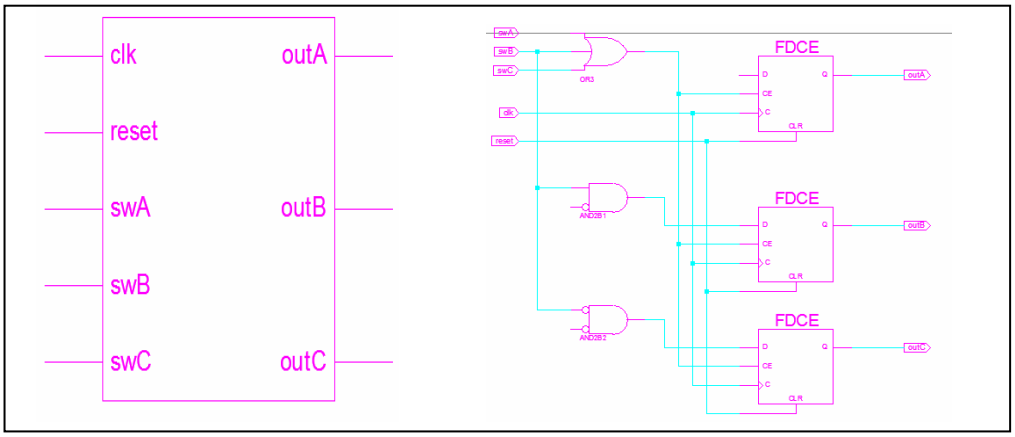

Figure 4.1: Menu's Top and Bottom View Model

Blitz top and bottom view model that are shown in figure 4.2 has three inputs and outputs. Blitz's schematic design looks more complex than Menu's.

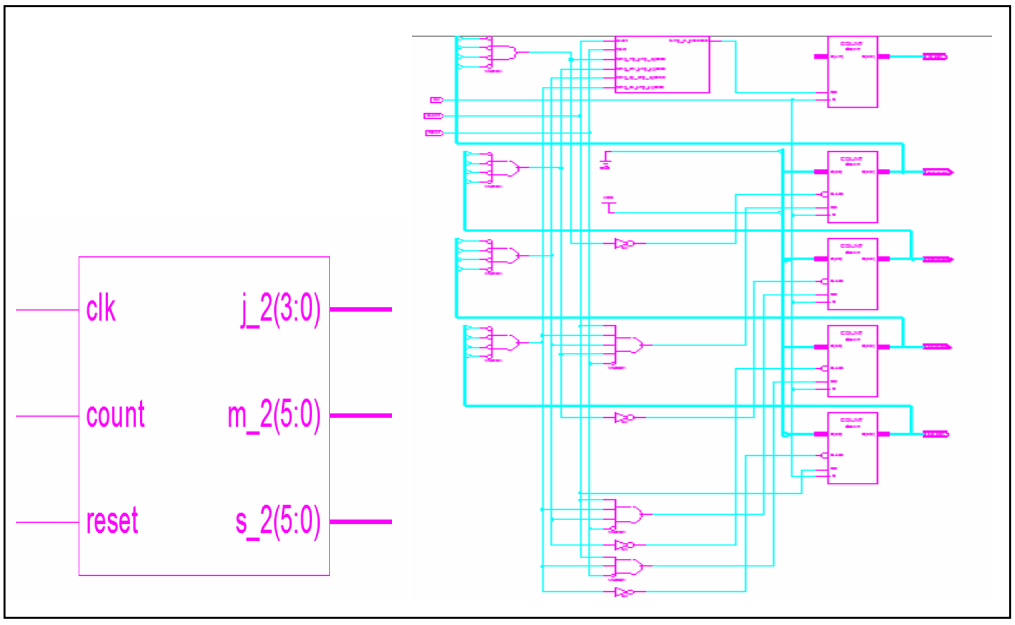

Figure 4.2: Blitz's Top and Bottom View Model

Figure 4.3 shows the top and bottom view model for Standard (1). Standard (1) is using four inputs and three outputs. The schematic is more complex. This is because the influences from the programming design.

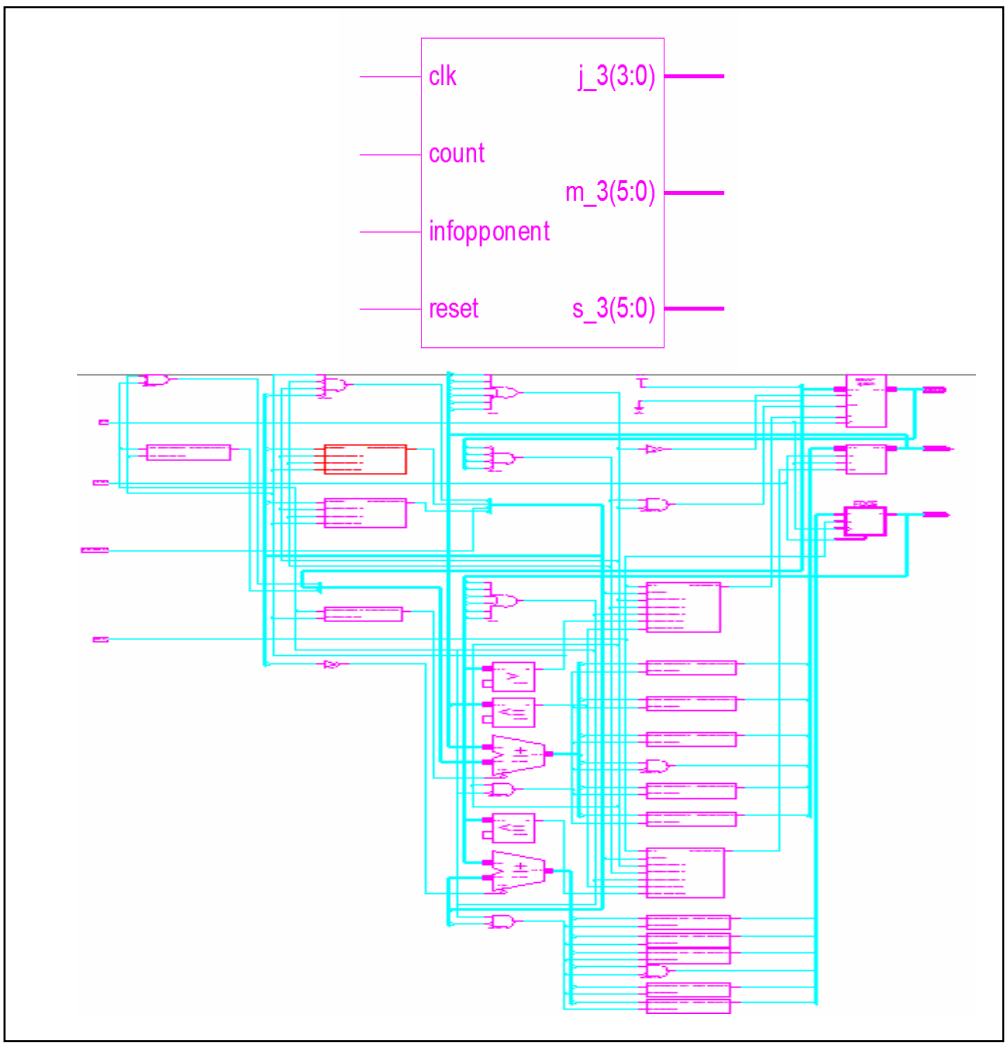

Figure 4.3: Standard (1)'s Top and Bottom View Model

Standard (2) schematic top and bottom view model can see in Figure 4.4. From comparison between Standard (1) and Standard (2), Standard (2) produces more complex schematic than Standard (1). This can be approved by according to their programming design. The schematics design should follow the programming design.

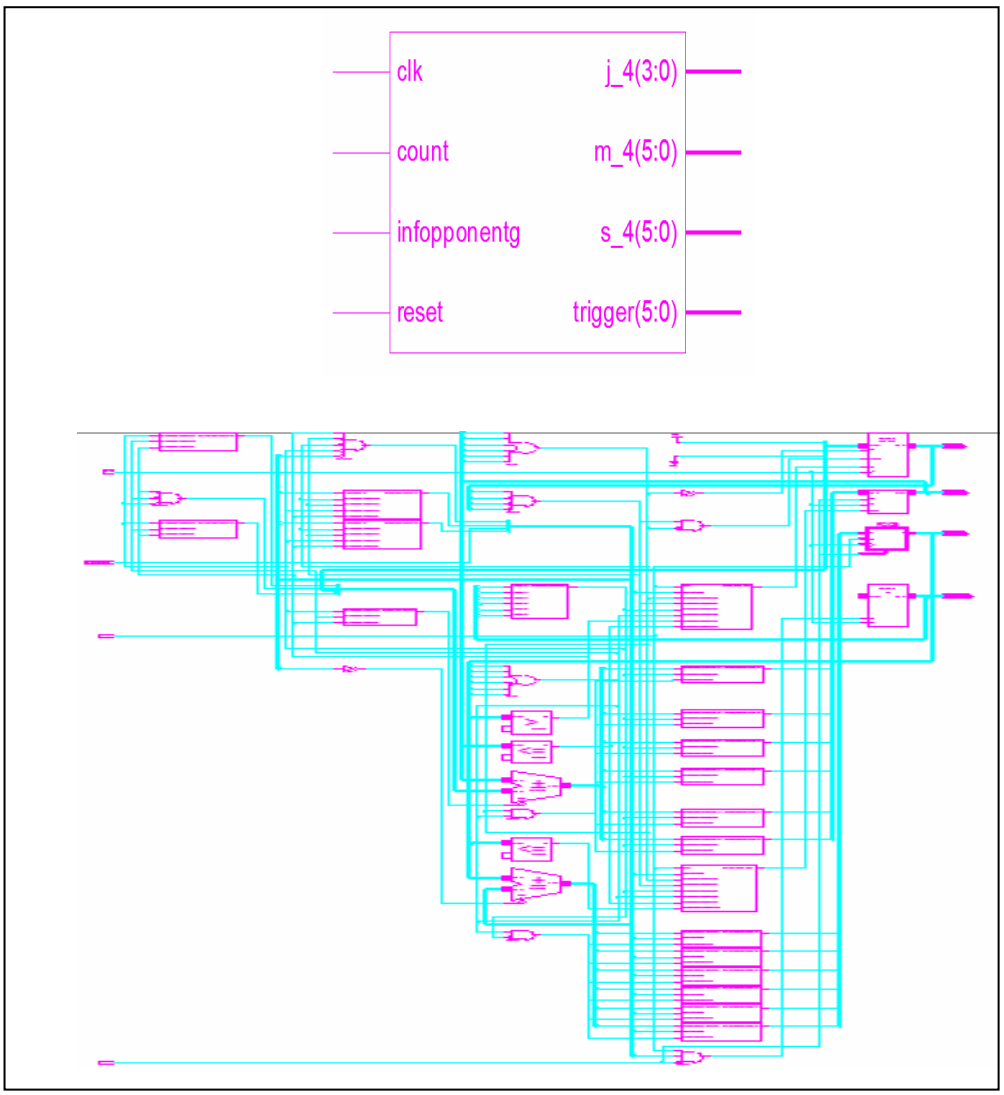

Figure 4.4: Standard (2)'s Top and Bottom View Model

Figure 4.5 is the top and bottom view model of Top Level. This model is the combination of all design. The more complex schematics can be view if all the combination design is in bottom view model. In the figure, can see that just integrated circuits for each part.

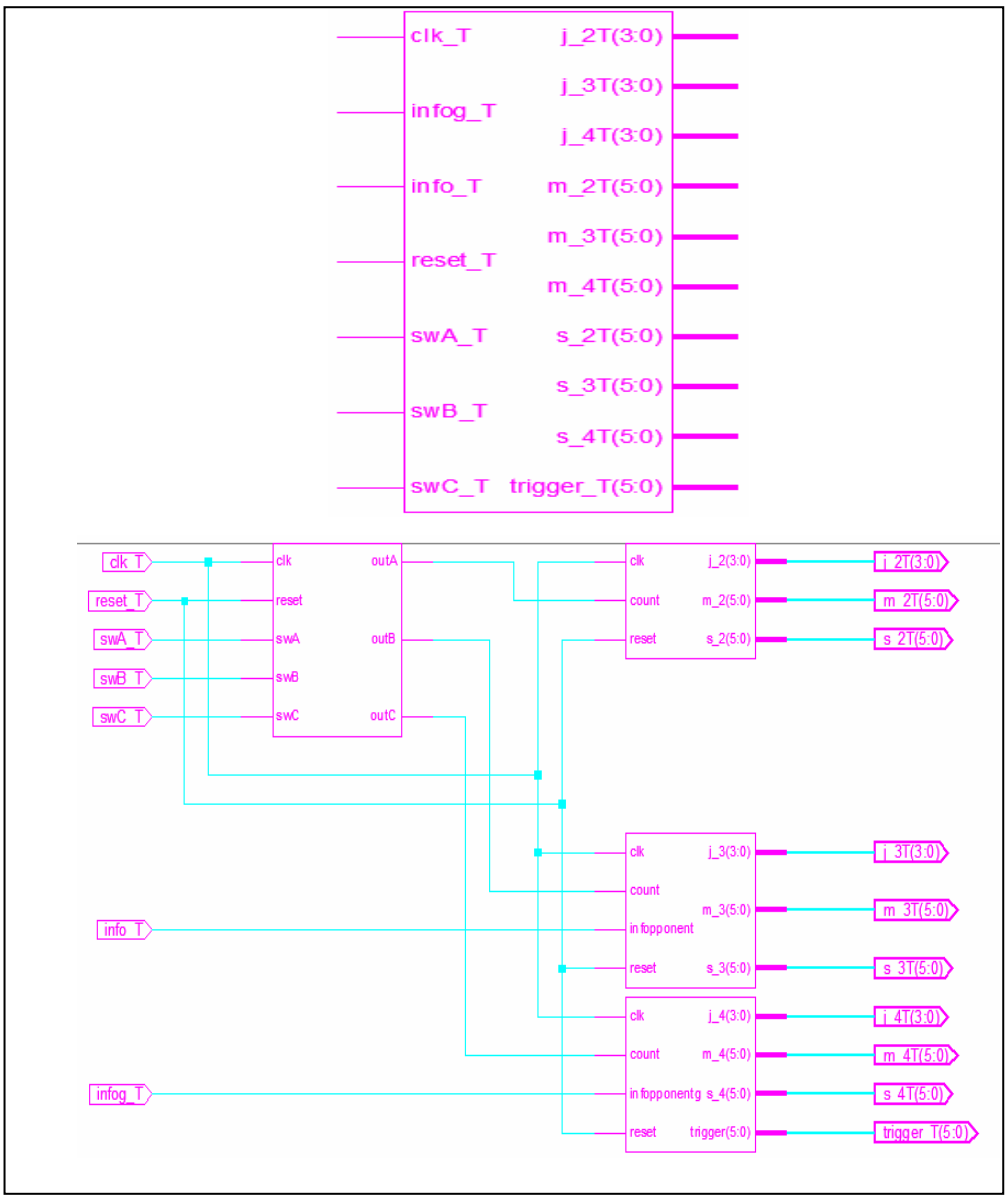

Figure 4.5: Top Level's Top and Bottom View Model

4.3 Behavioral Simulation

To simulate a VHDL file, must create a testbench first. The testbench is going to simulate the module. The simulator that is use in this project is Xilinx ISE Simulator. The simulation is running at another window called wave window.

The first simulation in Figure 4.6 is for menu simulation. The simulation shows when the player got selected the switch A (swa) and it allows the player to play Blitz game where it had assign the game as outa.

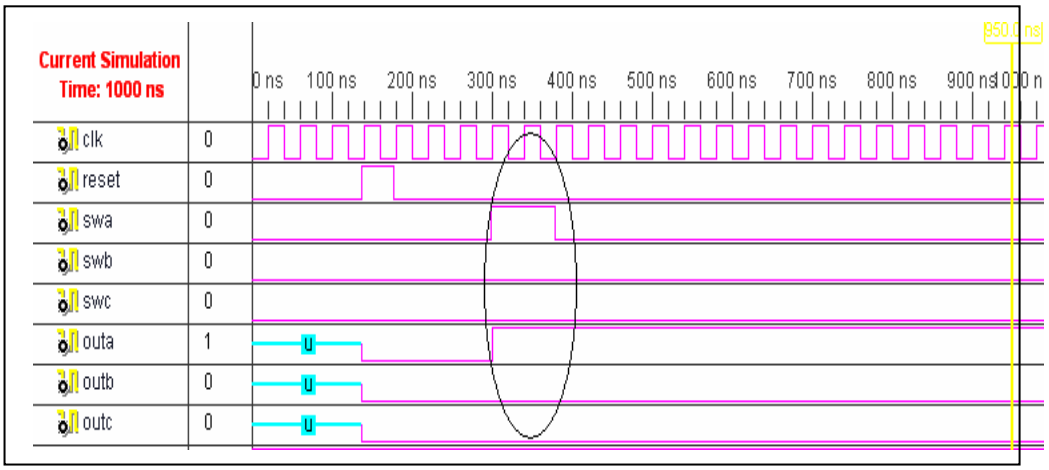

Figure 4.6: Menu Simulation

To get the explanation clear, Table 4.1 conclude the Menu design. If the input is swa, the output is outa where is the Blitz time control active. The other two conditions are swb and swc where give the output for Standard (1) and Standard (2). All these conditions active when clock and reset are '1'.

Table 4.1: Menu Relationship

clock	↑
reset	1
swa	outa (Blitz)
swb	outb (Standard (1))
swc	outc (Standard (2))

The simulation shows in Figure 4.7, when the count is '1' then the counter will start counting downward. The example shows that the game has one hour duration. When the moment that indicates by the circle, it shows that the player has 59 minutes and 58 seconds time left to finish his game.

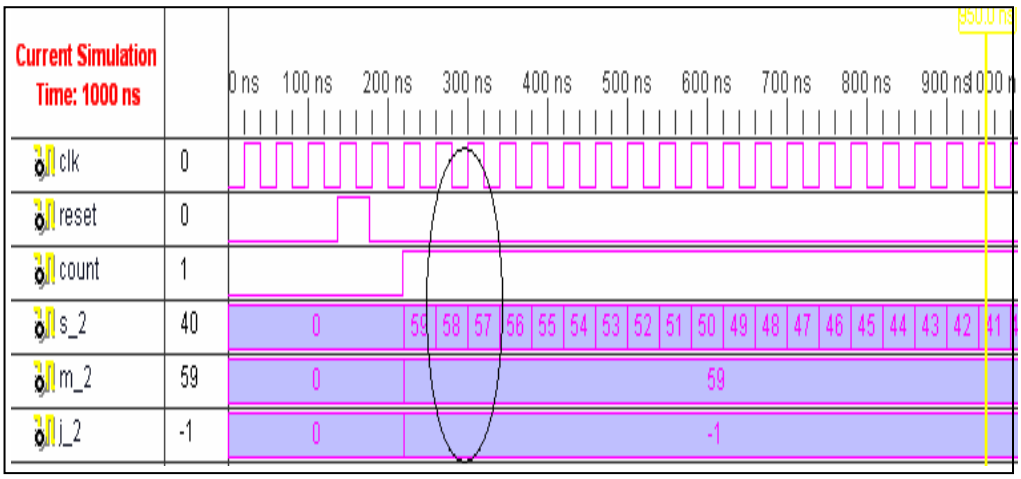

Figure 4.7: Blitz Simulation

Blitz time control is the simplest game to understand. The time is counting downward. The period of the time counting, the player can make his move until there is no time to count. The example process of Blitz time control can be seen in Table 4.2.

Table 4.2: Blitz Relationship

clock	reset	count	j	m	s
↑	1	0	0	0	0
↑	0	1	1	0	0
↑	0	1	-1	59	59
↑	0	1	-1	59	58
↑	0	1	-1	59	57

The simulation shows in the Figure 4.8, when the count is ‘1’ the counter will start counting downward. The time that allocated for the player is 1 hour and 30 minutes at the first starting game. Then, if there has an interrupt from a player (infopponent), 30 seconds will be added at the time makes the seconds part change

from 49 to 79 and his time is 1 hour 29 minutes and 79 seconds. But in reality, there has no 79 seconds so the time calculated is 1 minute and 19 seconds. Now, the player would have 1 hour 30 minutes and 19 seconds times to continue his game.

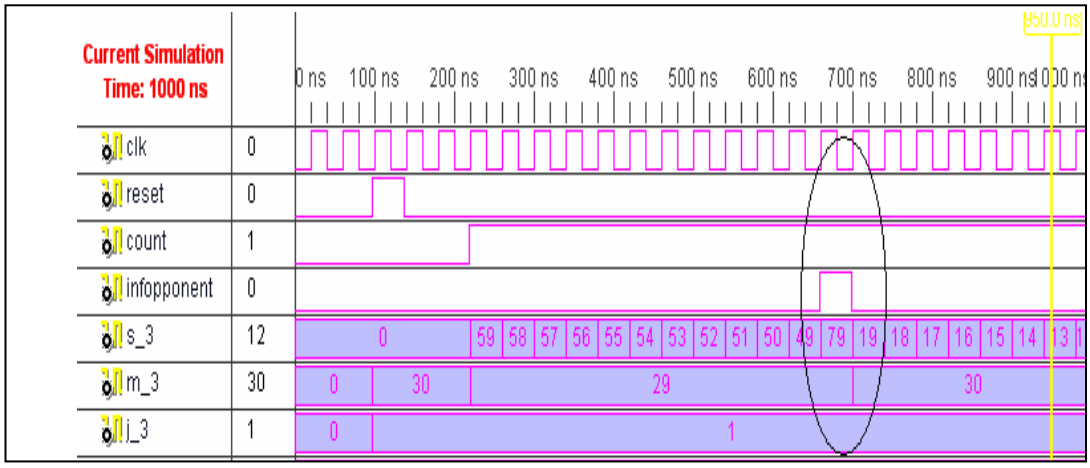

Figure 4.8: Standard (1) Simulation

It is complex to understand the system of Standard (1) time control. Table 4.3 is the example of the process. The value at seconds and minutes cannot exceed 59. This is because there are 59 seconds and minutes. If when the extra 30 seconds added to the time and the value of the seconds exceed by 59, the minutes will get extra one and the seconds is minus by 60. This condition fulfilled where one minute is equal to 60 seconds.

Table 4.3: Standard (1) Relationship

clock	reset	count	info	j	m	s
↑	0	0	0	0	0	0
↑	1	0	0	1	30	0
↑	0	1	0	1	29	59
↑	0	1	0	1	29	58
↑	0	1	1	1	30	28
↑	0	1	0	1	30	27

In Figure 4.9, the simulation shows the counter is counting downward when the count is '1' starting from 1 hour and 30 minutes. The flow is same as the previous module but it is advanced from the standard1, when the interrupt from player is reach for 40 times, 40 minutes will be added to the time left. The player would have longer period to play the chess.

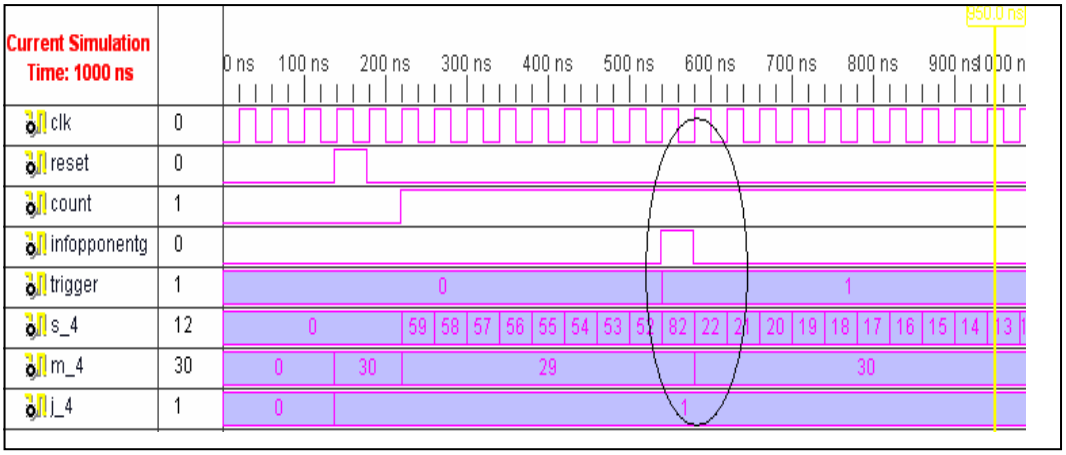

Figure 4.9: Standard (2) Simulation

By guide from Table 4.4, reader can easily understand the process of Standard (2) time control. The process seem like Standard (1) process flow. The value of trigger influenced by the mode of infopponent. When the trigger reaches 40, the player get 40 minutes extra to complete the game.

Table 4.4: Standard (2) Relationship

clock	reset	count	info	trigger	j	m	s
↑	1	0	0	0	1	30	0
↑	0	1	0	0	1	29	59
↑	0	1	1	1	1	30	29
↑	0	1	0	1	1	30	38
↑	0	1	1	2	1	30	58

4.4 Implementing FPGA's

After successfully simulated the design, the synthesize stage converts the text-based HDL design into an NGC netlist file. The netlist is a non-readable file that describes the actual circuit to be implemented at a very low level. The implementation phase uses the netlist and a constraints file to recreate the design using the available resources within FPGA. Constraints may be physical or timing and are commonly used for setting the required frequency of the design or declaring the required pin-out.

The map stage distributes the design to the resources available in the FPGA. The mapping will be incomplete if the design is too big for the specified device. The map stage also uses the UCF file to understand timing and may sometimes decide to add further logic (replication) to meet the given timing requirements. Map has the ability to shuffle the design around LUTs (Look-Up Table) to create the best possible implementation for the design. The whole process is automatic and requires little user input.

The place and route stage works with the allocated CLBs (Configurable Logic Block) and chooses the best location for each block. For a fast logic path, it

makes sense to place relevant CLBs next to each other simply to minimize the path length. The routing resources are then allocated to each connection, again using a careful selection of the best possible routing types. The place and route tool would use a longline to span the chip with minimal delay or skew if many areas of the design needed.

At this point, it is good practice to re-simulate. As all of the logic delays are added by the LUTs and flip-flops are known (as well as the routing delays), MXE can use this information for timing simulation.

Finally, a program called bitgen takes the output of place and route and creates a programming bitstream.

When developing a design, it may not be necessary to create a bit file on every implementation, just need to ensure that a particular portion of the design passes timing verification.

The steps of implementation must be carried out in this order:

1. Synthesize
2. Fit
3. Timing Simulate
4. Program

WebPACK ISE software will automatically perform the steps required if a particular step is selected. Double click on the target device, xc3s500e-5fg320, in the Source window and enter the characteristics that are required. The project, originally targeted on whatever device, is now targeting a Xilinx Spartan-3 FPGA.

The green ticks in the process window should have disappeared and bees replaced by orange question marks, indicating that the design must be re-synthesized and re-implemented.

4.4.1 Synthesis

The XST synthesis tool will only attempt to synthesize the file highlighted in the Source Window. The synthesis tool recognizes all the lower level blocks used in the top-level code and synthesizes them together to create a single netlist.

The design can check by double-clicking on Check Syntax in the Process window, expand the Synthesis subsection by clicking on the “+” next to Synthesize. Ensure that any errors in the code are corrected before continue. The design should be okay because both of the Bencher and MXE have already checked for syntax errors. It is useful, when writing code, to periodically check the design for any mistakes using this feature.

For FPGAs, state machines are usually one hot encoded. This is because of the abundance of flip-flops in FPGA architectures. A one hot encoded state machine will use one flip-flop per state. Although this may seem wasteful, the next state logic is reduced, and the design is likely to run much faster.

The synthesis tool will never alter the function of the design, but it has a huge influence on how the design will perform in the targeted device.

4.4.2 The Constraints File

To get the ultimate performance from the device, the most required is telling the implementation tools what and where performance is required.

This design is particularly slow and timing constraints are unnecessary. Constraints can also be physical; pin locking is a physical constraint.

At the PACE tool, assign all I/O pins in the Design Object List, save and exit the PACE session. At this part, the designer can enter the period of his program.

The Implement Design in the Process window is to implement the design by double-clicking on Implement Design. When there is a green tick next to Translate, Map, and Place and Route, the design has completed the implementation stage.

A green tick means that the design ran through without any warnings. A yellow exclamation point may mean that there is a warning in one of the reports. To avoid errors or warnings, the design procedure outlined in the manual have to be followed.

4.4.3 FPGA Reports

Each stage has its own report. Clicking on the "+" next to each stage lists the reports available:

1. The Translate Report shows any errors in the design or the UCF.

2. The Map Report confirms the resources used within the device and describes trimmed and merged logic. It will also describe exactly where each portion of the design is located in the actual device. A detailed Map Report can be chosen in Properties for map.

3. The Post-Map Static Timing Report shows the logic delays only (no routing) covered by the timing constraints. If the logic-only delays do not meet timing constraints, the additional delay added by routing will only add to the problem.

4. The Place and Route Report give a step-by-step progress report. The place and route tool must be aware of timing requirements. It will list the given constraints and report how comfortably the design fell within - or how much it failed - the constraints.

5. The Asynchronous Delay Report is concerned with the worst path delays in the design – both logic and routing.

6. The Pad Report displays the final pin-out of the design, with information regarding the drive strength and signaling standard.

7. The Guide Report shows how well a guide file has been met (if one was specified).

The Post Place and Route Static Timing Report add the routing delays. Notice that the max frequency of the clock has dropped.

WebPACK ISE software has additional tools for complex timing analysis and floor planning, which are beyond the scope of this introductory book.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

Chess clocks are a great way to improve the game. The mind is more focused on that executing move that will proffer an advantage. Digital clocks have spurred a wave of experimentation with more varied and complex time controls than the traditional standards.

The project is based on the design of the Digital Chess Clock. It employs ISE software (ISE Design Suite 10.1) and implementation on Field-Programmable Gate Arrays (FPGAs) Xilinx board. These are the two major parts of the structural design. This project is considers as half successful and half of the objective of the project have successfully fulfilled. The initial theoretical researches support the development of this project.

The increasing size and speed of modern FPGAs allow complex computations, on the order of an average sized program, to be performed in a small collection of processing elements. Software design is a process of problem-solving and planning for a software solution. After the purpose and specifications of software are determined, software developers will design or a plan for a solution. It includes low-level component and algorithm implementation issues as well as the architectural view.

5.2 Design Challenge

One of the biggest challenges of the project was implementing the software code onto the FPGA Xilinx board. The steps of implementation must be carried out in several orders. The several orders are including synthesize, fit, timing simulate and program. All these tools are listed in the Source window when enter the project navigator.

It is necessary to ensure that a particular portion of the design passes timing verification when developing a design. A lot of time was taken to understand and familiar with the VHDL, their syntax and structure. The designer must know the basic from the viewpoint using an HDL to write a text-based description of a digital circuit for design entry, pre-synthesis simulation, and logic synthesis.

5.3 Recommendation

There are some improvement can be done to the system design. Some of the possible enhancements are discussed in this topic. The improvise system can give more reliability to the user and can add some extra credits to the system’s market value.

Some of the recommendation is on the extension of the FPGA board. This is because, the project need 7-segment instead of LCD screen.

5.4 Costing and Commercialization

This project takes costing into consideration. The system was intended to be built at low cost. For the hardware, the project using the FPGA Xilinx board which is borrowed from the lab. Besides that, the output of the system is 7-segment is also taken from the lab. The product of this project actually needs lower cost. A low cost microcontroller coupled with a low cost CPLD from Xilinx can deliver the same performance at approximately half the cost.

The demand of chess clock is likely to involve the improvement of the digital chess clock. Today, chess is one of the world's most popular games, played by millions of people worldwide. Digital timers are much more sensitive to hard knock. Any of these fragile soldering on the motherboard can go, and while cost has come down a lot. It is very waste of value to misuse expensive and complicated digital timers. In digital clocks, clock stopper, buttons, balance in digital timers there are a whole lot of modes that have to be incorporated. Digital timer is easy to operate and reprogram. It is also offer a retainment function.

Thus, it is essential to further develop this project as it has a market value and could be commercialized. Having found the right approach to the market and established volume products, the Chess Digital clock will be a starting point of good business opportunity.

REFERENCES

1. Ezza. (2008). *WebPack-Default branch.* SourceForge.
2. White, A. (2007). *WebPack – website templates for small and start – up businesses.* Cardiff Website Design Company.
3. (2006). *FPGA - What is FPGA.* Jupiter Online Media
4. ISE Design Flow Overview, internet sources URL http://toolbox.xilinx.com/docsan/xilinx8/help/iseguide/html/ise_fpga_design_flow_overview.htm
5. Hughson, J. (2006). Common Culture, Commodity Fetishism and The Cultural Contradictions Sport. *International Journal of Cultural Studies.* Vol. 9, No. 1, 83-104.
6. Bodlaender, H. (1996). *The Rules of Chess.* Netherlands: Houten.
7. Anderson, G. (2008). *How to Write A Paper Journal.* Lewiston: Department of Biology, Bates College.
8. Calvin, J. D. (2005). *Answers Your Questions: A Guide for Fans and tournament Players*. California: International Directory of Chess Teachers.
9. Hani, M. K. (2007). *Starter's Guide to Digital Systems VHDL & Verilog Design*. Selangor, Malaysia: Prentice Hall.
10. Parnell, K. and Mehta, N. (2004). *Introduction to Programmable Logic.* USA: Xilinx.

APPENDIX A

Source code for Menu

APPENDIX B

Source code for Blitz

APPENDIX C

Source code for Standard (1)

APPENDIX D

Source code for Standard (2)

APPENDIX E

Source code for port map