00075

THE OPEN UNIVERSITY OF SRI LANKA
BACHELOR OF TECHNOLOGY - LEVEL 6
ECX6236 – PROCESSOR DESIGN
FINAL EXAMINATION 2009

DURATION: THREE HOURS

Date : 25th March 2010 **Time :** 1400 - 1700 Hrs

Answer **three** questions including the question in Section A and selecting two from Section B.

Section A

The following question is compulsory. It carries 70 marks.

1. Several types of elections are being held frequently in Sri Lanka such as Presidential, Parliamentary and Provincial elections. The procedure of handling these elections is very much similar to each other. Voters will go to polling stations and cast their votes in polling booths. After closure of the polling, ballot boxes will be transported to counting centres. Counting of ballot papers is carried out by counting officers at these centres. Finally, results will be announced by the election commissioner. The most time consuming and difficult task of this procedure is counting the ballot papers where a lot of human resources are needed.

   One way of overcoming these difficulties is to setup specially designed electronic voting machines at polling booths. In place of marking the vote on a paper, voters can cast their votes by just pressing a button on these machines. This system is called Direct-Recording Electronic (DRE) voting. For this system, instead of using a general purpose processor it is proposed to manufacture a special purpose microcontroller.

   Your task is to design this microcontroller – Vote Counting Processor (VCP), which can be fabricated later. Consequently the VCP can be used for building DRE voting systems. You must design the VCP in such a way that DRE voting systems can be used for different types of elections after reprogramming them.

   As the VCP is used only for the mentioned purpose, your design may differ from general purpose processors. The Instruction Set Architecture (ISA) of the VCP should be capable enough to develop programs for various elections, i.e. handling a number of candidates, preferential voting if any, and so on.

   You are free to decide on how to keep the counts for each candidate – in the processor itself or in an external memory of the system. However, finally there should be a provision to access these counts at the counting centre. The main idea of using VCP is to avoid the user (authorised election officer at the Department of Elections) writing complex programs and to handle appropriate election-oriented functions by its hardware.

   Your design of the processor must be simple. You may include special functional units/ components along with a description. You may assume that there are only Presidential and Parliamentary elections. Clearly state any other assumptions you made (if any).

   a) Draw a diagram to show how to deploy the VCP (once it is fabricated) for developing DRE voting systems for polling booths. (*You do not need to design the DRE voting systems but you have to identify the inputs and outputs of the VCP according to possible components of such systems.*)
   b) Write a short description on the working procedure of a DRE voting system indicating the internal functionality of your processor.

c) Accordingly, identify the necessary instructions needed for this processor and design the ISA.
d) Using your instruction set write a simple program for a Presidential election contesting 4 candidates.
e) Draw a block diagram for the processor. Clearly state all functions of each block inside the processor and show the data path. Indicate all input and output signals of the processor.
f) Identify entities for which you need to write VHDL codes to synthesise the processor.
g) Write the behavioural/ structural VHDL codes for each entity except for the Control Unit of the processor. You may define the Control Unit as a component. (Refer the Annexure for syntax of VHDL instructions).

Section B

*Answer **two** questions from this section. Each question carries 15 marks.*

2.

a) Briefly describe the modelling methods available in VHDL.
b) Integrate the VHDL codes for different entities in *Question (1.g)* of *Section A* to obtain a complete VHDL code for the VCP.

3.

a) What are the steps that you follow to design a hardwired control unit of a processor?
b) Construct the state diagram of the Control Unit of the VCP in *Question 1*.

4.

a) Briefly describe the tasks of a Computer Designer.
b) Estimate the performance of the VCP you designed in *Question 1*.
c) When you implement the VCP on an FPGA you will not be able to check all internal signals of the processor other than input/output signals. Propose a technique that you can use in order to check all signals inside the FPGA. You are free to add new components into your design.

5.

a) Write a Behavioural VHDL code for the *8-bit Full Adder* as given in the Fig 5.1.

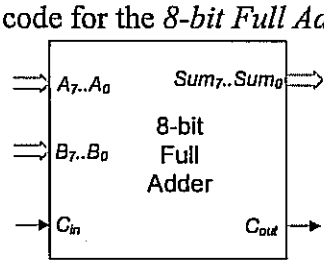

Fig 5.1

b) Construct a unit (draw a schematic diagram) to implement the following function using the *8-bit Full Adder*.

$$y = \sum_{i=1}^{n} i$$

Calculation must be stopped when it reaches the maximum possible value that can be handled by the Adder.
c) Write a Structural VHDL code for the *unit* you designed according to your schematic diagram drawn for the question 5.b.

Annexure

Syntax of selected instructions of the VHDL

```
ARCHITECTURE architecture_name OF entity_name IS
  [declaration part]
BEGIN
  Concurrent statements part
END architecture_name
```

```
CASE expression IS
  WHEN value=> statements;
  WHEN value=> statements;
  WHEN OTHERS statements;
END CASE;
```

```
COMPONENT component_name
  PORT (port1_name : port1_type;
        port2_name : port2_type;
        …);
END COMPONENT [component_name];
```

```
ENTITY entity_name IS
  PORT (port1 : port1_type;
        port2 : port2_type;
        …);
END entity_name;
```

```
IF condition THEN
  Sequence of statements
  {ELSIF condition THEN
     Sequence of statements}
[ELSE
  Sequence of statements]
END IF;
```

```
LIBRARY library_name;
```

```
Instance_label: component_name PORT MAP (first_port, second_port,
                                         third_port, …);
Instance_label: component_name PORT MAP (formal1=> actual1,
                                         formal1=> actual1,
                                         formal1=> actual1, …);
```

```
[process_label:] PROCESS (signal1, signal2, …)
                   [declaration part]
                 BEGIN
                   Sequential statements part
                 END PROCESS;
```

```
SIGNAL signal_name : signal_type;
```

```
TYPE type_name;
```

```
USE library_name.type_expression.inclussion;
```

```
WAIT FOR time_expression;
```

WAIT ON **signal1, signal2, …;**

WAIT UNTIL ***condition;***

```
WHILE condition LOOP
  Sequential statements
END LOOP;
```