00052

The Open University of Sri Lanka
Faculty of Engineering Technology
Department of Electrical and Computer Engineering

Study Programme	: Bachelor of Technology Honours in Engineering
Name of the Examination	: Final Examination
Course Code and Title	**: EEX6536/ECX6236 Processor Design**
Academic Year	: 2019/2020
Date	: 09th October 2020
Time	: 1400-1700 hrs
Duration	**: 3 hours**

General Instructions

1. Read all instructions carefully before answering the questions.
2. This question paper contains one (1) question in SECTION A and four (4) questions in SECTION B on four (4) pages.
3. Answer the question in SECTION A. [70 Marks] and answer any TWO questions from SECTION B. [30 Marks]
4. The answer to each question should commence from a new page.
5. Refer to the Annexure of the VHDL syntax given on page four (4) to write VHDL code.
6. This is a Closed Book Test (CBT).
7. Answers should be in clear handwriting.
8. Do not use Red colour pen and clearly state your assumptions if any

00052

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

Refer the Annexure for the syntax of VHDL instructions

Section A

The following question is compulsory. It carries 70 marks.

1. In computing Pattern matching (PM) is a very important area. It is used in different real-world applications such as text editors, information retrieval, pattern recognition, virus scanning, data mining, machine learning, DNA sequence analysis, and network security.

   In PM a given sequence of tokens/template is checked for existence in an input data. These tokens/templates can be words, values, data, pixels, and so on, which are represented in binary form. Usually, the matching of the tokens/templates must be exact in PM. However, some applications require this matching in a given range of values without fixing to a particular value.

   Considering the above requirements, your task is to design a special-purpose processor for PM (PMP), which can be used for improving the performance. The main function of the PMP is to identify any matching token/template in a given data stream. These tokens/templates can be in different data formats and represent in one dimensional (1D) or two dimensional (2D) arrays. Similarly, input data streams are also in 1D or 2D arrays with the same data format as the tokens/templates. You have to clearly show how these tokens/templates and the input data streams are stored and fetched during the matching. The output may be YES/NO or a percentage of mapping as the case may be. However, you may also propose any other acceptable output according to your design.

   The ISA should be in a standard instruction format similar to an instruction set in general-purpose processors (in the way of Opcode and Operands in an instruction format). Moreover, it should be capable enough to develop programs for various applications with different data streams by using user-defined tokens/templates.

   As this is a special-purpose processor, your design may differ from general-purpose processors. You may include special functional units/ components along with a description. Clearly state any other assumptions you made (if any).

   a) Write a short description of the working procedure of the PMP, indicating the internal functionality of your processor. You have to show clearly how the pattern matching is done and give the output for different data formats, representations (1D or 2D) and different kind of applications.

   b) Accordingly, identify the necessary instructions and design an ISA for this processor.

   c) Using your ISA write a program which can match a pattern for a given data stream. You can use your example for this.

   d) Draw a block diagram for the processor indicating all input and output signals. Clearly state all functions of each block inside the processor and show the data path.

   e) Identify entities for which you need to write VHDL codes to synthesise the processor.

   f) 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.

Section B

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

2.
   a) Integrate the VHDL codes for different entities in *Question (1.f)* of *Section A* to obtain a complete VHDL code for the PMP.
   b) Name the modelling methods available in VHDL. Which of the modelling methods is suitable for PMP processor? Justify your answer.

3.
   a) Construct the state diagram of the Control Unit of the processor you designed in *Question 1.*
   b) Briefly explain the steps that you have to follow to implement PMP on an FPGA.

4.
   a) Briefly describe how you are estimating the performance of a processor. Estimate the performance of the PMP you designed in *Question 1.*
   b) Which factors do you need to consider estimating the cost of the PMP?

5.
   a) Draw a block diagram for *D Flip-flop* with *1-bit* input signals *D, Clock, SET,* and *CLR* and write a Behavioural VHDL code for the same.
   b) Draw a schematic diagram for a *4-bit serial-in serial-out shift register* (Fig 1) using the *D Flip-flop* in question 5.a. The shift register shifts its stored data by one bit at each tick of the clock. The serial input, S_{in}, receives a bit to be shifted into the register at each clock tick and the bit will be sent out at the serial output, S_{out}, after four (4) clock ticks. When *CLR* is 1 (High), shift register sets all its bits to 0 at a clock tick, otherwise, it will function as described earlier.

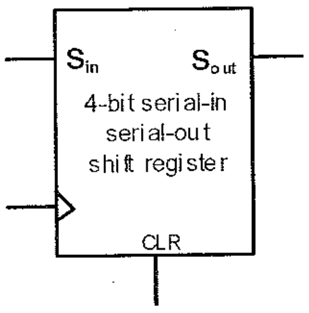

Fig 1

   c) Write a Structural VHDL code for the *4-bit serial-in serial-out shift register* 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;
```