THE OPEN UNIVERSITY OF SRI LANKA
BACHELOR OF TECHNOLOGY - LEVEL 6
ECX6236 – PROCESSOR DESIGN
FINAL EXAMINATION 2007

DURATION: THREE HOURS

Date : **10th May 2008** **Time :** **0930 - 1230 Hrs**

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

Section A

The question in this section is compulsory (70 marks). Answer all parts of the question.

1. Matrix operations are extensively used in Computer Graphics applications. Making these operations faster will improve the execution time of these applications. Your task is to design a special unit for matrix operations to support the microprocessor/ graphic processor.

   There should be special instructions to handle all matrix operations (however, in your implementation at the exam you need only to implement addition, subtraction and multiplication). Moreover you must provide other supporting instructions for matrix operations such as defining size of a matrix, moving elements of a matrix from memory to registers and from registers to memory and so on.

   You are advised to do all calculations through registers in order to obtain maximum performance. You may assume the number of registers is enough to hold three matrices at a given time.

   You are free to include any special functional units. However you must give a description of any special/ specific unit or component included in your design. When you are answering the questions state any other assumptions you made (if any) clearly.

   a) List all instructions and supporting instructions for matrix operations. Briefly describe them.
   b) Define instruction formats for all instructions listed above.
   c) Write a sample program for adding and multiplying two matrices using your ISA.
   d) Draw a block diagram for the processor. Clearly, give all functions of each block inside the processor and show the data path. Indicate all input and output signals of the processor.
   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. (Refer the Annexure for syntax of VHDL instructions).
   g) Propose a method to handle matrix operations if the number of registers is not enough to hold all elements of the matrices in use for a particular operation at the same time.

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,f)* of *Section A* to obtain a complete VHDL code for the special unit for matrix operations.

3.

a) What are the advantages in designing Microprogrammed Control Units in different processors using VHDL?
b) Construct the state diagram of the Control Unit of the special unit for matrix operations in *Question 1*.

4.

a) What are the major advantages of designing systolic architectures when considering the manufacturing of processors?
b) Estimate the performance of the special unit you designed in *Question 1*.
c) How do you estimate the cost of a processor?

5.

a) Assume that *8-bit Full Adder* (Fig. 1) has been designed. Using this *Full Adder* design a unit to perform addition and subtraction operations for 8 bit data. You must use one *8-bit Full Adder*, one multiplexer, and one inverter only.

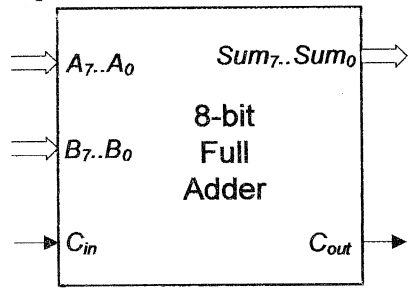

Fig 1

b) Write a VHDL code for your design given in 5.a).

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;
```