0030

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

Study Programme	:	Bachelor of Science Honours in Engineering
Name of the Examination	:	Final Examination
Course Code and Title	:	**EEX7436 – Processor Design**
Academic Year	:	2024/25
Date	:	7th February 2026
Time	:	1330-1630h
Duration	:	3 hours

General instructions

1. Read all instructions carefully before answering the questions.
2. This question paper consists of **One (1)** question in **Five (5)** pages.
3. Answer **all** questions.
4. Answer for each question should commence from a new page.
5. This is a Closed Book Test (**CBT**).

Question 01

Answer all questions. All questions are based on the Scenario given below.
Refer to the Annexure for the syntax of VHDL instructions.

Scenario

Modern scientific and engineering applications such as **numerical simulations, matrix computations, particle transport models, and differential equation solvers** involve repetitive arithmetic operations applied to large data sets. Executing these workloads on a conventional scalar processor leads to high instruction overhead and inefficient utilization of arithmetic units. To address this limitation, the processor is enhanced as **Vector Arithmetic Unit (VAU)** with a **Vector Arithmetic Logic Unit (Vector ALU)** that exploits **data-level parallelism. Vector ALU** can perform addition (+), subtraction (−), multiplication (* - dot product), and division (/ - element wise).

Let vector registers:

$$VR1 = [2, 4, 6, 8]$$

$$VR2 = [1, 3, 5, 7]$$

Vector Addition as follows.

Operation

$$VR3[i] = VR1[i] + VR2[i]$$

Result

$$VR3 = [3, 7, 11, 15]$$

Similarly, subtraction, multiplication and division are performed by VAU.

The primary objective of integrating a **Vector Arithmetic Unit** into the processor is to enable the execution of a **single instruction over multiple data elements simultaneously**, thereby improving computational throughput, reducing instruction count, and increasing energy efficiency. The design targets workloads dominated by floating-point operations and array-based computations, which are common in scientific simulations.

Figure Q1 shows the Vector Arithmetic Unit (VAU) architecture diagram.

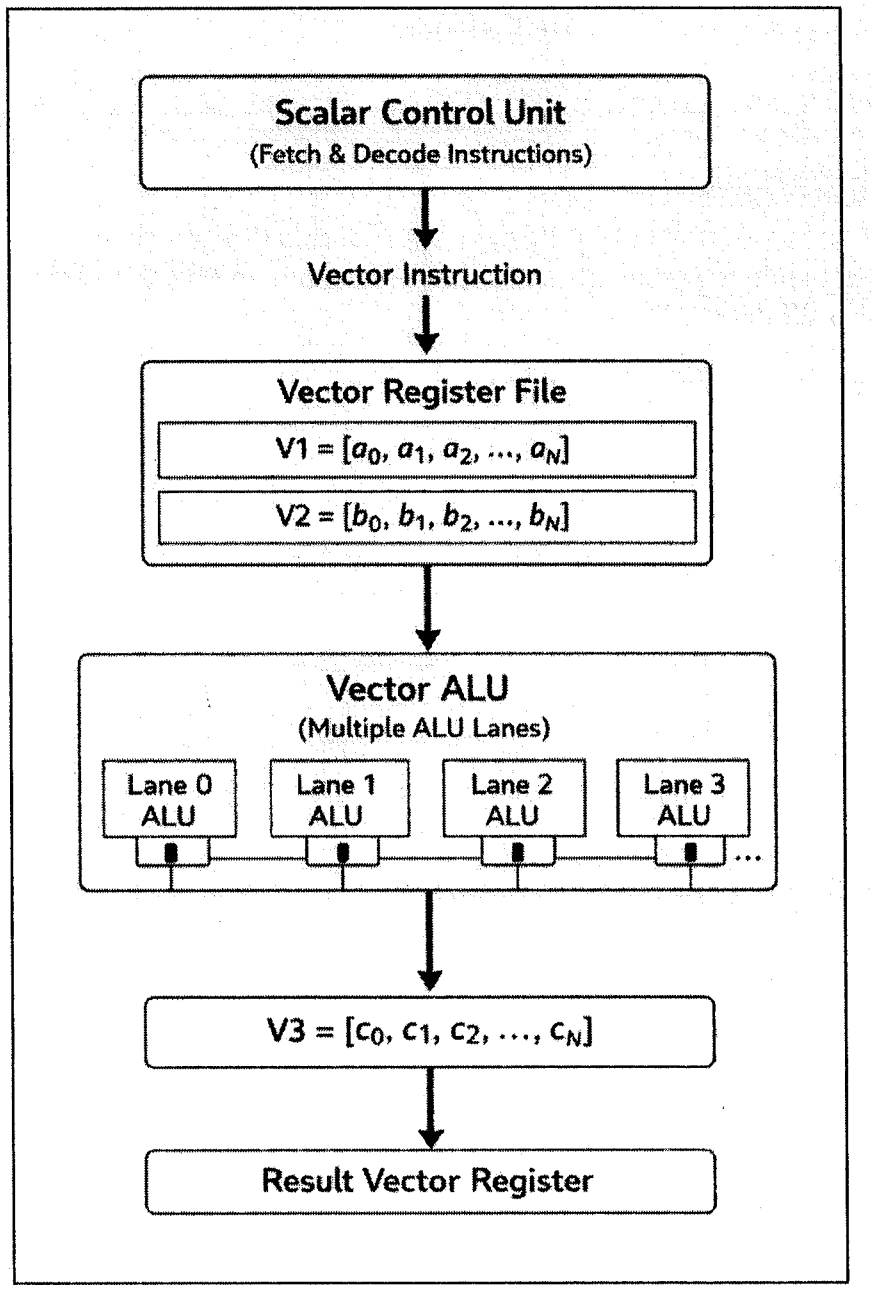


Figure Q1

Explanation of Diagram Components

1. **Scalar Control Unit**
   The scalar control unit is responsible for instruction fetch, decode, and control flow. It issues vector instructions to Vector ALU but does not perform data-parallel computations itself.

2. **Vector Register File**
   The vector register file stores operands and results in vector form. Each vector register contains multiple data elements, allowing bulk data to be processed with a single instruction.

3. **Vector ALU**
   The Vector ALU consists of multiple identical **ALU lanes**. Each lane performs arithmetic or logical operations on a single element of the vector. All lanes execute the same operation concurrently, enabling data-level parallelism.

4. **ALU Lanes**
   Each lane operates independently on corresponding elements from the source vector registers. For example, Lane0 processes element 0, Lane1 processes element 1, and so on.

5. **Result Vector Register**
   After execution, the computed results from all lanes are written back into a destination vector register, completing the vector operation.

The processor follows a **vector-scalar architecture**, where a **scalar control unit** is responsible for instruction fetch, decode, and control flow, while the Vector ALU performs data-parallel arithmetic operations. Operands are stored in **vector registers**, each containing multiple data elements. The Vector ALU itself is composed of several identical **ALU lanes**, with each lane operating on one element of the vector.

During execution, a vector instruction is issued by the scalar unit, and all ALU lanes execute the same operation concurrently on corresponding elements of the input vectors. The results are written back to a destination vector register.

Answer the following questions:

You are free to include any special functional units. However, you must describe any specific unit or component included in your design. When answering the questions, clearly state any other assumptions you made, if any.

a. Write a short description of the working procedure of the VAU, indicating the internal functionality with parallel Vector ALU operation of your processor. Also give limitations of the VAU. (12 marks)

b. Design ISA for vector operations and supporting instructions for VAU operations. Briefly describe them. (12marks)

c. Define instruction formats for all instructions listed above. (10 marks)

d. Write a sample program for execution of for loop using your ISA. (10 marks)

```
for (i = 0; I <= 4; i++)
    C[i] = (A[i] + B[i])
```

e. Draw a diagram for Vector ALU (VALU) that process arithmetic operations for 4 lanes and give output. (8 marks)

f. Draw a block diagram för the Vector Arithmetic Unit (VAU). Clearly, give all functions of each block inside the processor and show the data path. Indicate all input and output signals of the processor. (15 marks)

g. Identify entities for which you need to write VHDL codes to synthesise the processor. (5 marks)

h. Write the behavioural/ structural VHDL codes for each entity except for the Control Unit and obtain a complete VHDL code för the VAU. You may define the Control Unit as a component. (15 marks)

i. Briefly describe how you are estimating the cost and the performance of the VAU. (8 marks)

j. Estimate the performance of the VAU in CPI. (5 marks)

All Rights Reserved

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, formal2 => actual2, ...};

- [*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.inclusion*;

- **WAIT FOR** *time_expression*;
  WAIT ON *signal1, signal2, ...*;
  WAIT UNTIL *condition*;

- **WHILE** *condition* **LOOP**
  Sequential statements
  END LOOP;