Ollscoil Chathair
Bhaile Átha Cliath
Dublin City University

SEMESTER 1 EXAMINATIONS 2021/2022

MODULE: EE496 - Computer Architecture and HDL

PROGRAMME(S):

ECE	BEng Electronic & Computer Engineering
ECSAO	Study Abroad (Engineering & Computing)

YEAR OF STUDY: 4,O

EXAMINER(S):

Xiaojun Wang	(Internal)	(Ext:5808)
Dr. Josep R. Casas	(External)	External

TIME ALLOWED: 2 Hours

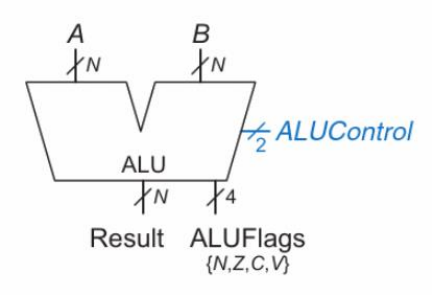
INSTRUCTIONS: **Answer 4 questions. All questions carry equal marks.**

PLEASE DO NOT TURN OVER THIS PAGE UNTIL YOU ARE INSTRUCTED TO DO SO.
The use of programmable or text storing calculators is expressly forbidden.
Please note that where a candidate answers more than the required number of questions, the examiner will mark all questions attempted and then select the highest scoring ones.

There are no additional requirements for this paper.

QUESTION 1 ***[TOTAL MARKS: 25]***

Figure Q1 symbolises an ALU with N-bit two's complement data inputs, `A[N-1:0]` and `B[N-1:0]`, and a 2-bit control input `ALUControl[1:0]`. The ALU has N-bit `Result[N-1:0]` and 4-bit `ALUFlags` outputs. Each bit of the `ALUFlags` represents whether the `Result` is **N**`egative`, **Z**`ero`, has a **C**`arry` or is o**V**`erflowed`, respectively. The `ALUControl[1:0]` controls the ALU operations, as shown in the table to the right of the ALU symbol.



ALU operations

$ALUControl_{1:0}$	Function
00	Add
01	Subtract
10	AND
11	OR

Figure Q1 ALU symbol with output flags

Q 1(a) **[18 Marks]**

Assume `N = 32`, write a VHDL description of the ALU that produces the hardware implementation shown in Figure Q1(a) below.

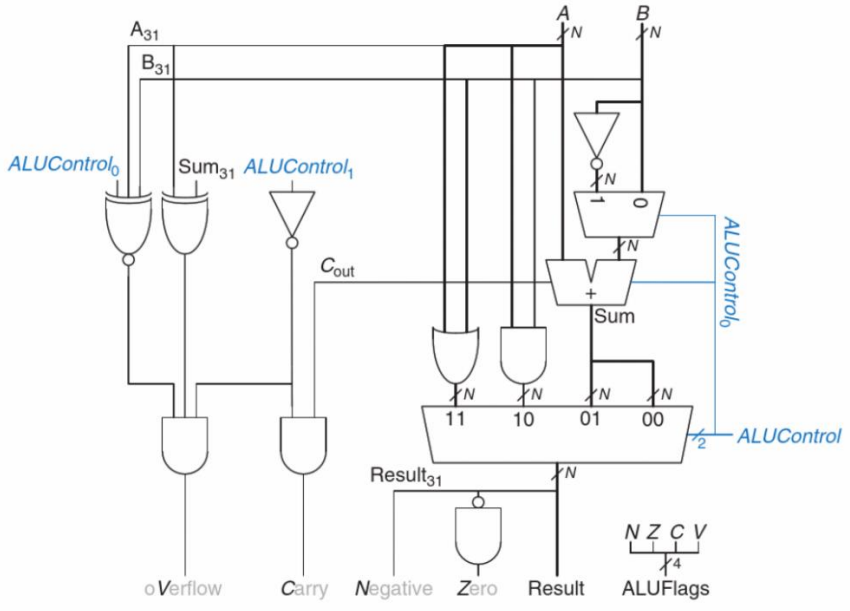


Figure Q1(a) Hardware implementation of the N-bit ALU with flags

Q 1(b) **[7 Marks]**

Explain the hardware implementation of each of the four ALU flags.

[End of Question 1]

QUESTION 2 ***[TOTAL MARKS: 25]***

Q 2(a) **[10 Marks]**

Write a VHDL model that infers a simple single port 16x32 RAM block with 16 32-bit words. The RAM write is synchronous to a clock rising edge, and when RAM write is enabled (we). The RAM read is asynchronous. The figure below shows the RAM input and output ports.

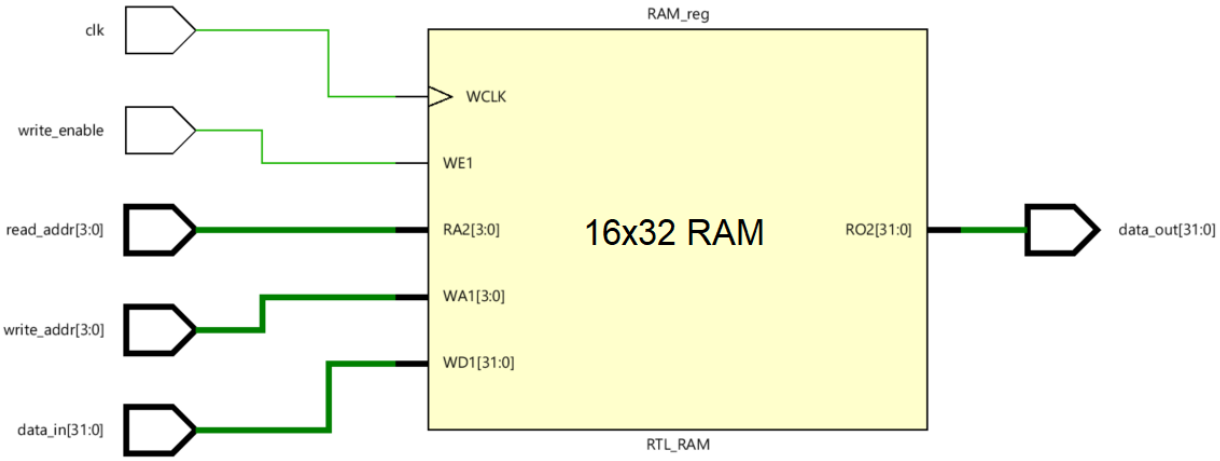


Figure Q 2(a)

Q 2(b) **[8 Marks]**

Figure Q2(b) shows a 16-bit bidirectional IO buffer. Data_out writes into the IO_bus when output is enabled (OE = '1'), and Data_in reads data from the IO_bus when output is disabled (OE = '0'). Write a VHDL model that infers this bidirectional IO buffer.

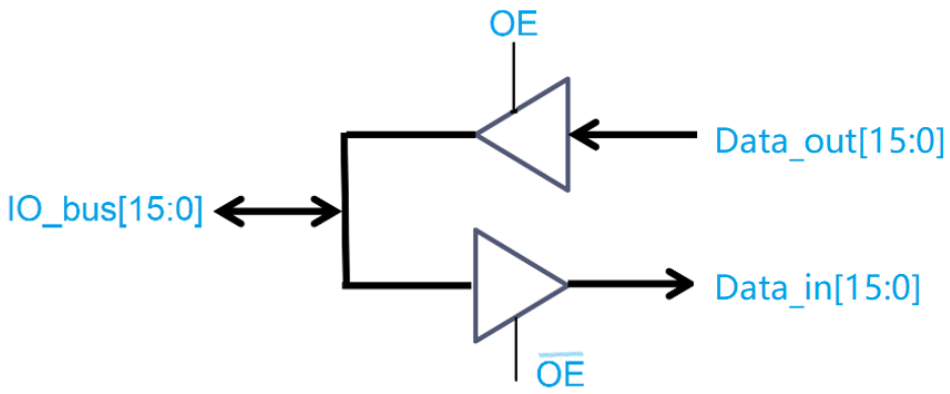


Figure Q 2(b) Bidirectional IO buffer

Q 2(c) **[9 Marks]**

The following is the truth table of a bus arbiter, in which **Brq0** to **Brq3** are the bus request signals from devices 0 to 3; **Bgt0** to **Bgt3** are the bus grant signals to devices 0 to 3. Do:

i) *write a synthesisable VHDL model* for the bus arbiter and

ii) *Draw the logic circuit* implied by the VHDL model.

Brq_3	Brq_2	Brq_1	Brq_0	Bgt_3	Bgt_2	Bgt_1	Bgt_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	**1**
0	0	1	X	0	0	**1**	0
0	1	X	X	0	**1**	0	0
1	X	X	X	**1**	0	0	0

[End of Question 2]

QUESTION 3 ***[TOTAL MARKS: 25]***

Q 3(a) **[15 Marks]**

Manually synthesise and draw the logic circuit implied by the following VHDL code. Then, explain the code that infers each of the logic components.

```
library ieee;
use ieee.std_logic_1164.all, ieee.numeric_std.all;
entity Q3a is
  port (d   : in  std_logic_vector(4 downto 0);
        clk : in  std_logic;
        sel : in  std_logic_vector(1 downto 0);
        q   : out std_logic_vector(4 downto 0));
end Q3a;
architecture rtl of Q3a is
begin
 p0: q(0) <= d(to_integer(unsigned(sel))) when clk = '1' else 'Z';

 p1: q(1) <= d(1) when clk ='1';

 p2: process(clk)
       variable a, b : std_logic;
     begin
        if rising_edge(clk) then
           q(2) <= b;
           b    := a;
           a    := d(2);
        end if;
     end process p2;

 p3: process
      variable a, b : std_logic;
    begin
      wait until falling_edge(clk);
        a    := d(3);
        b    := a;
        q(3) <= b;
    end process p3;

 p4: q(4) <= d(4) when rising_edge(clk);
end rtl;
```

Q 3(b) **[5 Marks]**

A 4-bit `std_logic_vector` type signal *s1* has the value "1101". When checking whether *s1* equals "11-1" using the standard comparison operator =, will the result be *true* or *false*? Briefly explain your answer.

Q 3(c) **[5 Marks]**

An 8-bit **signed** number N has hexadecimal value X"83". What is the equivalent decimal value of N? What is the hexadecimal value of N when it is extended to 16-bit while maintaining the same decimal value?

[End of Question 3]

QUESTION 4 ***[TOTAL MARKS: 25]***

Figure Q4a shows the timing constraints in sequential logic designs, in which t_{pcq} is the clock-to-Q propagation delay, t_{pd} is the delay of the combinational logic between two registers, and t_{setup} specifies the minimum time that the inputs to a register have to be stable before the rising edge of the clock for the register to sample the inputs correctly. The minimum clock T_c can be represented as $T_c \geq t_{pcq} + t_{pd} + t_{setup}$. Moreover, t_{pcq} and t_{setup} are determined by the technology that will implement the design, so the only variable under the control of a design engineer is t_{pd}, which has to satisfy the condition $t_{pd} \leq T_c - (t_{pcq} + t_{setup})$.

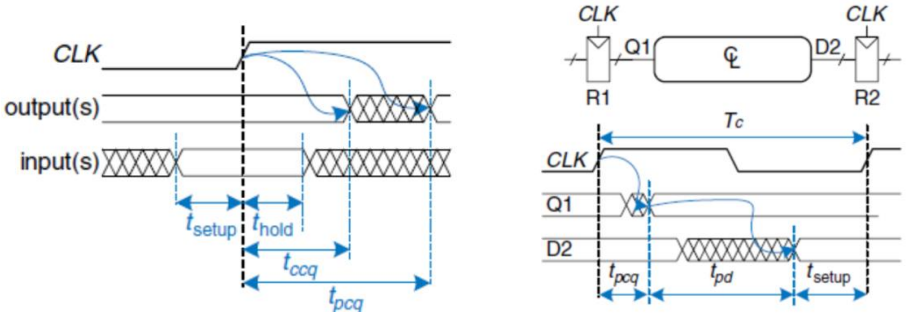


Figure Q4a Time constraints of sequential circuits

The following VHDL model adds four 256-bit signed numbers together.

```
library ieee; use ieee.std_logic_1164.all; use ieee.numeric_std.all;
entity addition is
generic (width : positive := 256);
port( clk : in std_logic;
      di0 : in std_logic_vector(width-1 downto 0);
      di1 : in std_logic_vector(width-1 downto 0);
      di2 : in std_logic_vector(width-1 downto 0);
      di3 : in std_logic_vector(width-1 downto 0);
       do : out std_logic_vector(width-1 downto 0));
end addition;

architecture syn of addition is
   type reg_file is array (3 downto 0) of signed(width-1 downto 0);
   signal REG : reg_file;
begin
p1: process(clk)  -- inputs are registered
begin
if (clk'event and clk = '1') then
       REG <= (signed(di3), signed(di2), signed(di1), signed(di0));
  end if;
end process p1;
p2: process(clk) -- serial addition and register outputs
begin
  if rising_edge(clk) then
     do <= std_logic_vector(REG(0)+REG(1)+REG(2)+REG(3));
  end if;
end process p2;
end syn;
```

Figure 4(b) shows the Vivado synthesised circuit from the above VHDL description. Assume that the clock-to-Q propagation delay t_{pcq} is 50 ps, the register setup time t_{setup} is 40 ps, and a 256-bit addition takes 8,000 ps.

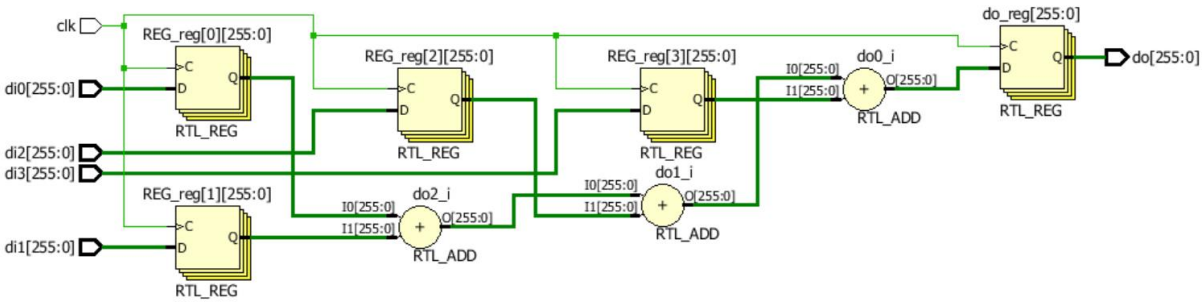


Figure Q4b Vivado synthesised circuit

Q 4(a) **[5 Marks]**

What is the minimum clock period T_c of the Vivado synthesised circuit in Figure Q4b?

Q 4(b) **[8 Marks]**

(i) Write an improved VHDL description so that when synthesised by Vivado, the generated circuit can run faster without using more hardware resources than the original design. (*Only need to write the modified code, no need to write the full description*).

[5 marks]

(ii) Draw the circuit corresponding to the improved VHDL model.

[2 marks]

(iii) What's the minimum clock period T_c of this improved design?

[1 marks]

Q 4(c) **[12 Marks]**

(i) Improve the VHDL description further so that Vivado will generate a pipelined implementation to increase the system throughput significantly. (*Only need to write the modified code, no need to write the full model*).

[6 marks]

(ii) Draw the circuit corresponding to the improved VHDL model.

[3 marks]

(iii) What's the maximum throughput of this pipelined design?

[3 marks]

[End of Question 4]

QUESTION 5 ***[TOTAL MARKS: 25]***

Q 5(a) **[5 Marks]**

Assuming a multi-cycle and pipelined computer datapath is broken down into five stages

1. Hardware to support an instruction *fetch*
2. Hardware to support an instruction *decode*
3. Hardware to support instruction *execution* (i.e. the ALU)
4. Hardware to support a *memory* read or write
5. Hardware to support the *write-back* of the ALU result to the register file

Assume that each of the above stages takes the amount of time specified in the table below.

Fetch Instruction from Memory	**Decode** & read operands from reg files	Instruction **Execution** ALU	**Memory** Read/Write	Results **write-back** to Reg files
250 ps	150 ps	200 ps	250 ps	100 ps

Given the times for the datapath stages listed above, what would the *clock period* be for the entire datapath?

Q 5(b) **[5 Marks]**

For the pipelined datapath in Q5(a), assuming no hazards or stalls, how long does it take to execute *one* instruction, or what is the instruction latency?

Q 5(c) **[5 Marks]**

What is the throughput of the pipelined datapath in Q5(a)?

5(d) **[10 Marks]**

For **a single-cycle processor** with the same data path time specification as in the table of Q5(a), What would be the clock period, the instruction latency and the throughput of the single-cycle processor?

[End of Question 5]

[END OF EXAM]