Projecto de Sistemas Digitais
Digital Systems Design
MEEC, 2023/24, 1st Exam

Table 1: Timing characteristics and resource utilization of logic-arithmetic components				
	Area	t_P	t_{SETUP}	t_{HOLD}
16-bit Register	10	2 ns	1 ns	1 ns
BRAM	100	2 ns	1 ns	1 ns
16-bit Adder / Subtractor	16	4 ns		
16-bit Multiplier (simple)	120	9 ns		
MUX 2:1 (16-bit data words)	16	3 ns		
MUX 4:1 (16-bit data words)	32	4 ns		
MUX 8:1 (16-bit data words)	64	5 ns		
16-bit logic operators	16	3 ns		

Note: all answers must be properly justified, without which they will not be classified.

1. **[4 val]** Consider the matrix-vector multiplication **C = A × B**, according to the following pseudocode, where **A** is a 100×4 matrix, **B** is a 4-element vector and **C** is a 100-element vector:

```
for (i=0 ; i < 100 ; i++)    {
  m1 = a(i,0) * b(0);
  m2 = a(i,1) * b(1);
  m3 = a(i,2) * b(2);
  m4 = a(i,3) * b(3);
  a1 = m1 + m2;
  a2 = m3 + m4;
  c(i) = a1 + a2;
  end_of_loop = (i >= 99);   }
```

Sketch the data flow graph (DFG) that corresponds **directly** to one iteration of the loop.
In the DFG, keep the sequence of operations exactly as indicated.

Define an ordered priority list using the critical path as metric.

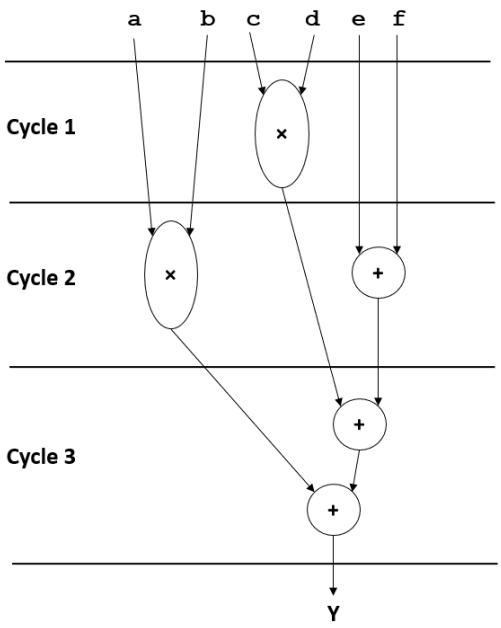
The elements of matrices **A** and **B** are 16-bit signed integers. The signal **i** is a 16-bit unsigned integer.
The output **`end_of_loop`** is a 1-bit signal.

Matrix **A** is stored in one independent 100×64-bit memory with 1-port for reading (4 row elements in each memory position).
Vector **B** is stored in 4 independent 16-bit registers.
Matrix **C** is stored in one independent 100×16-bit memory with 1-port for writing.

Consider that the arithmetic operators available are **1 multiplier (simple)** and **2 adders/subtracters**, that one multiplication requires one clock cycle, and that 1 addition can be performed in half clock cycle.
Obtain a list scheduling using the priority list defined.

Estimate the clock period to be used, considering that **all operators, buses and datapath registers are 16-bit wide** with the timing/area characteristics shown in table 1.

2. Consider the schedule defined in the figure, at the right.



The circuit datapath is to be implemented with the minimum number of adders and the minimum number of multipliers required to implement this schedule.

All input values are 16-bit signed integers, stored in independent **non-shareable** registers.

All operators, buses and datapath registers are 16-bit wide, with the timing/area characteristics shown in table 1. All circuit registers must be positive edge-triggered, fully synchronous and be synchronized with a unique global clock.

a) **[4 val]** Design the circuit datapath to achieve the best **performance** (given the constraints indicated).

Indicate the number of registers and multiplexers required.
Draw the block diagram of your datapath.
Identify the control signals required.

Estimate the area occupied, the minimum clock period, and the latency for your design.
Justify all the bindings and sharing of resources, and explain how the performance was maximized.

b) **[2 val]** Draw the state diagram of the control unit for the datapath designed in (a).

Indicate explicitly the values of the control output signals for each state.

The control must have synchronous reset and done signals. The computation should start when the reset signal becomes inactive. The done signal should become active when the result has been stored in the output register.

3. **[2 val]** Specify the VHDL architecture of an arithmetic component to perform the operation

```
p_sm = a_2c + b_2c + c_2c + d_2c
```

Inputs *a_2c*, *b_2c*, *c_2c*, *d_2c* are 8-bit signed integers represented in 2's complement format.
Output *p_sm* is a signed integer represented in sign-magnitude format.

The operations are performed using 2's complement adders. Dimension all adders and buses to preserve maximum accuracy in the calculations and avoid overflows.

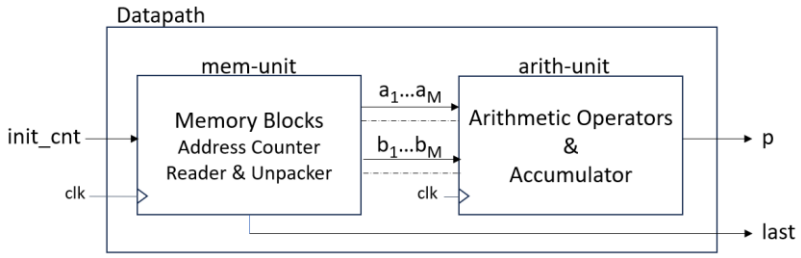
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity add4 is
 port ( a_2c, b_2c, c_2c, d_2c : in std_logic_vector (7 downto 0);
        p_sm : out std_logic_vector ( ??? ));
end addm;

architecture behavioral of addm is
  ???
begin
  ???
end behavioral;
```

4. Consider that you want to design a circuit, in a Xilinx Artix-7 device, to calculate the product of 2 vectors **A** and **B** of 1024 elements each, **<u>as fast as possible</u>**:

$$p = \sum_{i=0}^{1023} a_i \times b_i$$

The elements of vector **A** are real numbers represented in the fixed-point format Q4.12.
The elements of vector **B** are real numbers represented in the fixed-point format Q8.8.

Vector **A** is stored in one independent 256×64-bit memory with 1 port for reading.
Vector **B** is stored in one independent 1024×16-bit memory with 2 ports for reading.



The datapath is composed of 2 main components:

i. The memory unit includes the memory blocks, the address counter(s), and the reading/unpacking logic. It provides M a-elements and M b-elements, per clock cycle, to the arithmetic unit.
ii. The arithmetic unit computes the M multiply-accumulations.

All registers must be *positive edge-triggered* and be synchronized with a unique global clock.

a) **[3 val]** Estimate the best throughput achievable with the memory organization specified.

Indicate how to organize the arithmetic operations to achieve this throughput.
Sketch the block diagram of the arithmetic unit.

Estimate the clock period to be used, using the timing characteristics shown in table 1 (consider the timing characteristics of the most approximately sized operator).
Also estimate the total execution time of the vector product.

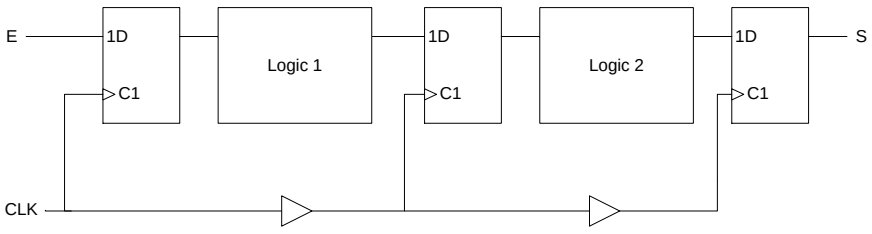
b) **[1 val]** Define the dimensions of all operands, buses, and registers in the arithmetic unit to avoid the existence of overflows and to avoid any loss of precision in the computations.
Indicate the fixed-point representation format of all operators and buses.

c) **[3 val]** Complete the pseudo-VHDL specification (in the following page) of the *mem_unit* component (don't care about minor syntax issues).

Define the address counter(s) and all the logic necessary to read (from the memories) and to provide the maximum number of vector elements per clock cycle (for the memory organization specified).
The *last* signal indicates the last accumulation cycle of the vector product.

5. **[1 val]** Consider the circuit below, whose components have the following timing characteristics:

t_{PFFmax}= t_{PFFmin}=3ns; t_{SETUP}=2ns; t_{HOLD}=1ns;
$t_{PLOG1max}$=8ns; $t_{PLOG1min}$=4ns; $t_{PLOG2max}$=16ns; $t_{PLOG2min}$=7ns;



What is the maximum value of the clock skew ($t_{Pbuffers}$) that can be tolerated by this circuit? Justify.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity mem_unit is
 port (
  clk, init_cnt: in std_logic;

  a1, ... : out std_logic_vector(15 downto 0); -- define all M a-elem ports
  b1, ... : out std_logic_vector(15 downto 0); -- define all M b-elem ports

  last: out std_logic );
end mem_unit;

architecture Behavioral of mem_unit is
 component memA
  port ( clka : in std_logic;
         wea : in std_logic_vector(0 downto 0);
         addra : in std_logic_vector(7 downto 0);
         dina : in std_logic_vector(63 downto 0);
         douta : out std_logic_vector(63 downto 0)   );
 end component;

 component memB
  port ( clka, clkb : in std_logic;
         wea, web : in std_logic_vector(0 downto 0);
         addra, addrb : in std_logic_vector(9 downto 0);
         dina, dinb : in std_logic_vector(15 downto 0);
         douta, doutb : out std_logic_vector(15 downto 0)   );
 end component;

 -- declare all internal signals
 ???


begin

 ima : memA port map ( clka => clk, wea => "0", dina => (others => '0'),
                       addra => ???, douta => ???   );  -- define connections

 imb : memB port map ( clka => clk, wea => "0", dina => (others => '0'),
                       clkb => clk, web => "0", dinb => (others => '0'),
                       addra => ???, douta => ???,      -- define connections
                       addrb => ???, doutb => ???   );

 -- counter
 process(clk)
 begin
   if(rising_edge(clk)) then

        ???

   end if;
 end process;

-- define internal logic and connections
 ???

-- define internal registers, if necessary
 ???

-- define output logic and connections
 ???

end Behavioral;
```