Projecto de Sistemas Digitais
Digital Systems Design
MEEC, 2021/22, 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	5 ns		
16-bit Multiplier (simple)	120	14 ns		
16-bit Multiplier (2-stage pipeline: 1 internal register-level)	130	8 ns		
16-bit Divider	500	85 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 algorithm, with 10 inputs and 2 outputs, defined by the following pseudo-code:

$$y_1 = (x_1 \times x_2) \times ((x_1 \times (x_3 + x_4)) + (x_5 + x_6))$$
$$y_2 = (((x_6 + x_7) + x_8) + x_9) + x_{10})$$

Draw the data flow graph that corresponds **directly** to the algorithm expressions (without reordering).

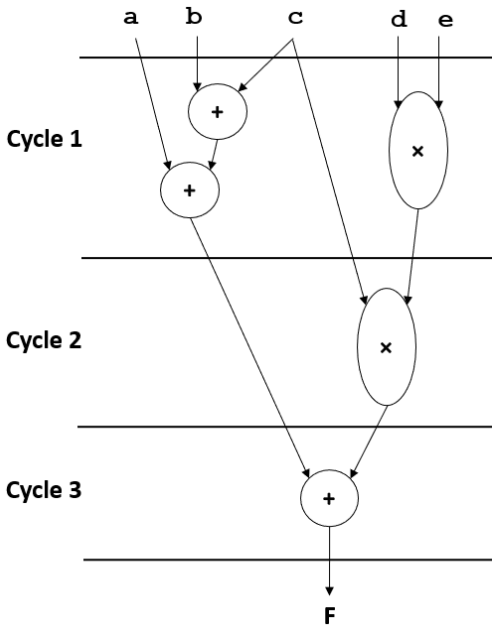
Define a priority list using the critical path as metric.

Consider that the inputs are stored in independent registers, that the arithmetic operators available are 1 multiplier (simple) and 2 adders, 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 input values are 16-bit integers in 2's complement representation and all operators, buses and datapath registers are 16-bit wide with the timing/area characteristics shown above.

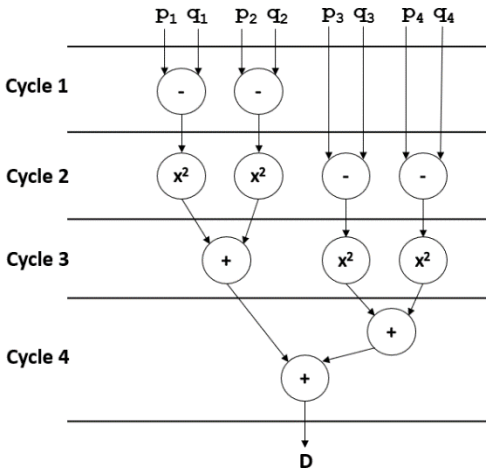
2. Consider the schedule defined in the figure, at the right, for a circuit whose datapath is to be implemented with 2 adders and one multiplier (simple).
   All input values are 16-bit integers in 2's complement representation.
   All operators, buses and datapath registers are 16-bit wide, with the timing/area characteristics shown on the 1st page.
   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 possible **performance** (given the constraints indicated).
   Indicate the number of registers and multiplexers required.
   Draw the block diagram of your datapath.
   Estimate the area occupied, the minimum clock period, and the latency for your design.
   Justify all the bindings and sharing of resources, and explain why the performance estimated is the best possible.

b) **[1 val]** Draw the state diagram of the control unit of the circuit for the datapath designed in (a).
   Define an output-based state encoding to eliminate the output logic of the finite state machine. Justify.

3. **[3 val]** Consider **the schedule at the right** to calculate the (square of the) Euclidean distance, *D(p,q)*, between two points *p* and *q* in 4 dimensions.



Consider also that you want to perform a sequence of distance calculations from a given ***p*** point to a set of 100 ***q*** values, which are stored in one independent 100×32-bit memory with 1-port for reading.
Sketch the draft block diagram of the datapath of a pipelined architecture, that executes the whole calculations as fast as possible, **not changing the schedule**.
What is the maximum throughput that you can achieve, and what is the minimum number of multipliers, adder/subtracters and intermediate pipeline registers required to achieve it? Justify.

4. Design an arithmetic component, with a registered output, to perform the operation

$$D = (p_1 - q_1)^2 + (p_2 - q_2)^2$$

where the inputs $\boldsymbol{p_i}$ and $\boldsymbol{q_i}$ are real numbers represented in unsigned fixed-point format Q12.4.
The ***p*** and ***q*** data are stored in 2 independent block memories BRAM1 and BRAM2, both configured as 512×32-bit memories, organized such that each memory position contains
$\boldsymbol{p} = <\boldsymbol{p_2}, \boldsymbol{p_1}>$ (BRAM1) or $\boldsymbol{q} = <\boldsymbol{q_2}, \boldsymbol{q_1}>$ (BRAM2).
The component includes a counter to generate the memories address to access the 512 memory positions in sequence (0 to 511).

a) **[2 val]** Define the fixed-point representation format required for the internal signals:

```
s_i = p_i - q_i ;    -- output of the subtracter
m = s^2 ;            -- output of the multiplier
```

and for the output distance D, to avoid the existence of overflows and to avoid any loss of precision in the computations. Justify.

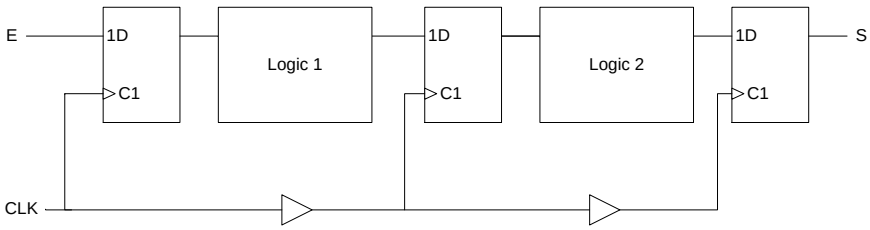
b) **[3 val]** Complete the pseudo-VHDL specification (in the following page) of the architecture of the component to perform the required operations (don't care about minor syntax issues).
Define all internal signals and dimension all signals (including the output), operators, and buses to preserve maximum precision in the calculations and avoid overflows.
The positive edge-triggered output register and counter share the same synchronous reset and the same enable.

c) **[1 val]** Consider that the first two 32-bit positions of BRAM1 have the following values (hexadecimal):

	Port-A	
Address	0	1
Value	0018 FFF0	0082 0204

Indicate what are the real values stored in position 0 for ***p1*** and ***p2***.

5. Consider the following circuit.



The circuit components have the following timing characteristics:

t_{PFFmax}= t_{PFFmin}=4ns; t_{SETUP}=2ns; t_{HOLD}=1ns;
$t_{PLOG1max}$=9ns; $t_{PLOG1min}$=3ns; $t_{PLOG2max}$=10ns; $t_{PLOG2min}$=5ns;

a) **[1 val]** Estimate the minimum clock period for the circuit to work correctly, considering the *buffers* in the clock network "ideal" ($t_{Pbuffers}$=0ns). Justify.

b) **[0.5 val]** If there exists a positive clock skew ($t_{Pbuffers}$=3ns) can the circuit operate with a faster clock? Justify and quantify the minimum clock period required.

c) **[0.5 val]** Will the circuit still work correctly with a larger positive clock skew ($t_{Pbuffers}$=7ns)? Justify.

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

entity compd is
  port ( p, q : in  std_logic_vector ( 31 downto 0 );
         rst, en, clk : in  std_logic;
         addr : out  std_logic_vector ( 8 downto 0 );
         d : out  std_logic_vector ( ??? ));
end compd;

architecture behavioral of compd is
  signal p1, p2, q1, q2 : unsigned(15 downto 0);
  -- internal signals
  ???

begin
 p2 <= unsigned(p(31 downto 16));
 p1 <= unsigned(p(15 downto 0));
 q2 <= unsigned(q(31 downto 16));
 q1 <= unsigned(q(15 downto 0));

 -- arithmetic operations
 ???

 -- registered output
 process(clk)
  begin
   if (clk = '1' and clk'event) then

    ???

   end if;
 end process;

 -- address counter
  process (clk)
  begin
    if (clk'event and clk = '1') then

      ???

    end if;
  end process;

end behavioral;
```