The first part is a set of five questions (2 points each), the second and third parts are two small problems (5 points each).

1 Questions

1.1. You are in charge of designing a digital hardware component that is synchronous of one single clock \(\text{clk}\). One of the primary inputs of the component is a one bit data line \(\text{data}\). \(\text{data}\) is not synchronous to \(\text{clk}\): it can change any time relatively to \(\text{clk}\). What shall you do with this primary input to guarantee the correct behaviour of your design?

1.2. After logic synthesis it sometimes happens that warnings are found in the synthesis reports about «inferring latches». Why does the synthesizer consider that inferred latches deserve warnings? If these warnings really indicate that something is wrong, where does the problem come from? In which circumstances can these warnings be safely ignored?

1.3. List the different constraints a combinational VHDL process must fulfil for synthesis?

1.4. In your opinion, how many bits of register will be inferred when synthesizing the VHDL design of listing 1? Draw a schematic of the synthesis result.

```vhdl
entity e is
  port (clk : in  bit;
       x : in  bit_vector(8 downto 5);
       s : out bit_vector(7 downto 0));
end entity e;

architecture a of e is
begin
  process (clk)
  variable v : bit_vector(7 downto 0);
  begin
    if clk = '1' and clk'event then
      s <= v;
      v := v(3 downto 0) & x;
    end if;
  end process p;
end architecture a;
```

Listing 1: Synthesizable model
1.5. In your opinion, what will happen when simulating the VHDL model shown on listing 2?

```vhdl
entity foo is
end entity foo;
architecture arc of foo is
signal a, b, c, d, e: bit;
begin
  process(a, b, c)
  begin
    if a = '0' then
      d <= b xor c;
    elsif b = '1' then
      d <= not c;
    else
      d <= c;
    end if;
  end process;
  process(a, d, e)
  begin
    if a = '0' then
      c <= not (d or e);
    elsif d = '0' then
      c <= not e;
    else
      c <= d;
    end if;
  end process;
end architecture arc;
```
Listing 2: A VHDL model

2 Design and VHDL coding of a matrix transposer

In digital signal processing it is sometimes useful to transpose matrices. This problem consists in designing, in synthesizable VHDL, a simple matrix transposition unit. The interface is shown on listing 3.

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity mt is
  port(clk: in std_logic; — Clock
ce: in std_logic; — Clock Enable
arstn: in std_logic; — Reset
dsi: in std_logic; — Data Strobe In
dir: in std_logic; — Shift direction
di: in std_logic_vector(31 downto 0); — Data In
do: out std_logic_vector(31 downto 0)); — Data Out
end entity mt;
```
Listing 3: Matrix transposer interface

The matrix transposer contains 1024 one-bit internal registers organized in a $32 \times 32$ square matrix. We denote $M[i, j]$ the matrix element in row $i$ and column $j$, where $0 \leq i < 32$ and $0 \leq j < 32$. The top row is indexed $i = 0$ and the bottom row is indexed $i = 31$. The left column is indexed $j = 0$ and the right column is indexed $j = 31$. The role of the input/output ports is detailed in table 1.

When shifting from left to right each row of the matrix is shifted to the right by
<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>1</td>
<td>Input</td>
<td>CLocK. The unit is synchronous on the rising edge of CLK.</td>
</tr>
<tr>
<td>ARSTN</td>
<td>1</td>
<td>Input</td>
<td>Asynchronous, active low ReSeT. If ARSTN='0' the content of all internal registers is forced to '0'.</td>
</tr>
<tr>
<td>CE</td>
<td>1</td>
<td>Input</td>
<td>Clock Enable. If ARSTN='1' and CE='0' on a rising edge of CLK, all internal registers are frozen and keep their current value.</td>
</tr>
<tr>
<td>DSI</td>
<td>1</td>
<td>Input</td>
<td>Data Strobe Input. If ARSTN='1' and CE='1' and DSI='1' on a rising edge of CLK, the value carried by the DI input is shifted in the internal matrix register.</td>
</tr>
<tr>
<td>DIR</td>
<td>1</td>
<td>Input</td>
<td>DIREction indicator. Specifies in which direction the internal matrix register shifts when ARSTN='1' and CE='1' and DSI='1' on a rising edge of CLK. If DIR='0' the matrix is shifted from left to right. If DIR='1' the matrix is shifted from top to bottom.</td>
</tr>
<tr>
<td>DI</td>
<td>32</td>
<td>Input</td>
<td>Data Input. The 32-bits value that is shifted in the internal matrix register when ARSTN='1' and CE='1' and DSI='1' on a rising edge of CLK.</td>
</tr>
<tr>
<td>DO</td>
<td>32</td>
<td>Output</td>
<td>Data Output. The 32-bits output value.</td>
</tr>
</tbody>
</table>

Table 1: Interface specification
one position: \( 0 \leq i < 32 \), \( M[i,0] \) takes the value of \( DI(i) \), the previous value of \( M[i,31] \) is lost. When shifting from top to bottom each column of the matrix is shifted to the bottom by one position: \( 0 \leq j < 32 \), \( M[0,j] \) takes the value of \( DI(j) \), the previous value of \( M[31,j] \) is lost.

\( DO \) combinatorially depends on the \( DIR \) input and on the internal matrix register: if \( DIR='0' \), \( DO(i) \) takes the value of \( M[i,31] \) while, if \( DIR='1' \), \( DO(j) \) takes the value of \( M[31,j] \).

**TODO (2 points):** Carefully study the specification. Decide what VHDL types you will use, how many VHDL processes you will use, which are synchronous and which are combinatorial. Clearly explain all this.

**TODO (3 points):** Design in synthesizable VHDL the architecture of the matrix transposition unit.

### 3 Design and VHDL coding of a watchdog

A watchdog is a special kind of timer that designers use to prevent a computer system from crashing. An internal register \( R \) decrements automatically. If \( R \) decrements from 1 to 0, it means that the computer system is frozen, \( R \) stops decrementing and the watchdog raises an interrupt. The interrupt forces the microprocessor to leave the deadlock situation and to execute an Interrupt Service Routine (ISR). The ISR discovers what is wrong, takes the appropriate actions to restore the normal behaviour and reloads \( R \) before giving control back to the Operating System (OS).

In order to prove that it is still responsive, the OS must regularly reload \( R \) with a strictly positive value before it reaches value 0.

In this small problem you will design a simple watchdog that could be used in a computer system. Its interface is shown on listing 4 and detailed in table 2. The generic parameter \( N \) is the number of bits of register \( R \).

#### Listing 4: Watchdog interface

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity watchdog is
  generic(n : positive);
  port(clk : in std_logic; -- Clock
       ce : in std_logic; -- Clock Enable
       srs : in std_logic; -- Reset
       load : in std_logic; -- Load register
       di : in unsigned(n-1 downto 0); -- Data In
       irq : out std_logic); -- Interrupt Reqest
end entity watchdog;
```

**TODO (1 point):** The clock frequency of the watchdog is 1 GHz. We consider that the system is frozen if one second elapses since the last time the watchdog register has been reloaded. What is the minimum value of generic parameter \( N \)? What is the value that will be used to reload the watchdog register?

**TODO (1 point):** Carefully study the specification and draw a block diagram of your watchdog. Decide how many VHDL processes you will use to code your watchdog, represent the boundaries of your processes on the block diagram and make clear which are synchronous and which are combinatorial.
<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>1</td>
<td>Input</td>
<td>CLocK. The watchdog is synchronous on the rising edge of CLK.</td>
</tr>
<tr>
<td>CE</td>
<td>1</td>
<td>Input</td>
<td>Clock Enable. If CE='0' on a rising edge of CLK, R is frozen and keeps its current value.</td>
</tr>
<tr>
<td>SRST</td>
<td>1</td>
<td>Input</td>
<td>Synchronous, active high ReSeT. If CE='1' and SRST='1' on a rising edge of CLK, the content of R is forced to 0.</td>
</tr>
<tr>
<td>LOAD</td>
<td>1</td>
<td>Input</td>
<td>LOAD register. If CE='1' and SRST='0' and LOAD='1' on a rising edge of CLK, R is reloaded with DI.</td>
</tr>
<tr>
<td>DI</td>
<td>N</td>
<td>Input</td>
<td>Data Input. The value loaded in R when CE='1' and SRST='0' and LOAD='1' on a rising edge of CLK.</td>
</tr>
<tr>
<td>IRQ</td>
<td>1</td>
<td>Output</td>
<td>Interrupt ReQuest. If, on a rising edge of CLK, R decrements from 1 to 0, IRQ takes value '1' for one clock cycle only.</td>
</tr>
</tbody>
</table>

Table 2: Interface specification

**TODO (3 points):** Design, in plain synthesizable VHDL the architecture of your watchdog.