The first part is a set of five questions (2 points each) and the second part is a small problem (10 points).

1 Questions

1.1. Assume you are asked to design a digital circuit for which the most important criteria is the power consumption. What will you care about and what techniques will you use to reduce the power consumption as much as possible?

1.2. After logic synthesis and when investigating the various synthesis reports, why is it important to pay attention to the inferred latches, if any?

1.3. In the DHT11 controller project we used a VHDL resolved type to model the data line between the controller and the sensor. What is a resolved type and why is it different from a non-resolved type? Why did we need this for this modelling?

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(7 downto 0);
         s: out bit_vector(7 downto 0));
end entity e;

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

Listing 1: Synthesizable model

1.5. Reminder: the u_unsigned VHDL type is an unresolved type similar to std_ulogic_vector for which all arithmetic operators have been defined.

Reminder: when called with two 32 bits u_unsigned operands, the multiplication operator (*) returns a 64 bits u_unsigned value.
Reminder: when called with two 64 bits `unsigned` operands, the addition operator (+) returns a 64 bits `unsigned` value.

In your opinion, how many 32 bits multipliers will be synthesized when synthesizing the VHDL design of listing 2? How many 64 bits adders? How many bits of register?

```
architecture bar of foo is
  type vt is array(natural range <>) of unsigned(31 downto 0);
  signal ai, xi: vt(0 to 15);
  signal s: unsigned(63 downto 0);
begin
  process(clk)
  variable tmp: unsigned(63 downto 0);
  begin
    if rising_edge(clk) then
      if resetn = '0' then
        s <= (others => '0');
      else
        tmp := (others => '0');
        for i in 0 to 15 loop
          tmp := tmp + ai(i) * xi(i);
        end loop;
        s <= tmp;
      end if;
    end if;
  end process;
end architecture bar;
```

Listing 2: Synthesizable model

2 Design and VHDL coding of a timer

A hardware timer is very simple in principle but its implementation in a dedicated piece of hardware poses some interesting challenges.

In this small problem you will design a simple timer that could be used in a computer system. Its interface is shown on listing 3 and detailed in table 1:

| library ieee;              | — Clock  |
| use ieee.std_logic_1164.all; | — Clock Enable |
| use ieee.numeric_std.all; | — Reset |
| entity timer is | — Data Strobe In |
| port(clk: in std_logic; on; ce: in std_logic; | — Mode indicator |
| rst: in std_logic; | — Data In |
| dsi: in std_logic; | — Data Out |
| mode: in std_logic; | — Interrupt ReQuest |
| di: in unsigned(31 downto 0); | — Data In |
| do: out unsigned(31 downto 0); | — Data Out |
| irq: out std_logic; | — Interrupt ReQuest |

Listing 3: Timer interface

The timer contains a 32-bits internal register $T$ used as a counter. A value is first loaded in $T$; then, each clock cycle, $T$ is decremented, until it reaches zero. When $T$ is zero, an interrupt request output is raised for one clock cycle. Depending on a mode indicator, the timer is then restarted from the loaded value or stopped until a new value
Table 1: Interface specification of the TIMER module

<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 timer is synchronous on the rising edge of CLK.</td>
</tr>
<tr>
<td>CE</td>
<td>1</td>
<td>Input</td>
<td>Clock enable. When low, all internal registers are frozen and hold their current value.</td>
</tr>
<tr>
<td>RST</td>
<td>1</td>
<td>Input</td>
<td>Synchronous, active high reset. When high on a rising edge of CLK for which CE is also high, force the content of all internal registers to a predefined value. T is initialized to zero.</td>
</tr>
<tr>
<td>DSI</td>
<td>1</td>
<td>Input</td>
<td>Data Strobe Input. When high on a rising edge of CLK for which CE is also high and RST is low, stores the value carried by the DI input in the internal register T.</td>
</tr>
<tr>
<td>MODE</td>
<td>1</td>
<td>Input</td>
<td>Mode indicator. When low, the timer stops at zero. When high, the timer restarts at the initial value after it reached zero.</td>
</tr>
<tr>
<td>DI</td>
<td>32</td>
<td>Input</td>
<td>Data Input. A 32-bits unsigned value used as initial value of the timer when DSI is high on a rising edge of CLK for which CE is also high and RST is low.</td>
</tr>
<tr>
<td>DO</td>
<td>32</td>
<td>Output</td>
<td>Data Output. The 32-bits unsigned current value of the timer.</td>
</tr>
<tr>
<td>IRQ</td>
<td>1</td>
<td>Output</td>
<td>Interrupt ReQuest. When high on a rising edge of CLK for which CE is also high and RST is low, indicates that the timer reached zero. IRQ stays high for one clock cycle only.</td>
</tr>
</tbody>
</table>
is loaded. Of course, in order to restart from the same value, another 32-bits register is required to hold the loaded value while $T$ is decremented.

**TODO (5 point):** Carefully study the specification and draw a block diagram of your timer architecture. Clearly identify and name the internal registers. Clearly identify and name the computing elements. If possible, put a kind of pseudo-code in your computing elements. Name all internal signals and specify their bit-widths. Finally, decide how many VHDL processes you will use to code your timer, which are synchronous and which are combinatorial and allocate the registers and the computing elements to one of your processes. Make all this specification work very clear and easy to understand.

**TODO (5 points):** Design, in plain synthesizable VHDL the architecture of your timer.