EDC: exam

Renaud PACALET

24 June 2011

You can use any document you need. On each page of your paper please indicate your name. Put your answers in any order but indicate first the corresponding question number. Example:

```
Question 2.1: I think this is a very good question and...
```

Advice: read every word very carefully and answer first the easy parts. You can write your answers in English or in French, as you like. Do not forget to attach your labs paperwork if you did not send it to me already or send them to me today, sharp deadline.

The five questions of the first part are worth 2 points each. If you consider a question as ambiguous explain why, then make the assumptions needed to solve the ambiguities and answer the modified question. If you consider a question as absurd explain why. If you find an error in a question explain why you think it is a error, propose a correction if it is reasonable and answer the modified question.

The second part is a small problem about a timer. It is very simple in principle but its implementation in a dedicated piece of hardware poses some interesting challenges. This part is worth 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. Why is it important not to use VHDL resolved types when unresolved types can be used instead?

1.3. When designing a finite state machine in VHDL is it preferable to hard-code the different states with bit-strings or to use an enumerated type? Why?

1.4. List the different constraints a combinatorial VHDL process must fulfil for synthesis?

1.5. Translate the following property from natural language to CTL: *If the START input is raised, then the DONE output is raised 3 clock cycles later, unless START is raised again before that*

2 Design of a timer

In this small problem you will design a simple timer that could be used in a computer system, like the NiosII system we saw during the labs. Its interface is the following and detailed in table 1:

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

entity timer is
  port(clk:  in  std_ulogic;             -- Clock
       ce:   in  std_ulogic;             -- Clock Enable
       rst:  in  std_ulogic;             -- Reset
       dsi:  in  std_ulogic;             -- Data Strobe In
       mode: in  std_ulogic;             -- Mode indicator
       di:   in  unsigned(31 downto 0); -- Data In
       do:   out unsigned(31 downto 0); -- Data Out
       irq:  out std_ulogic);            -- Interrup ReQuest
end entity timer;
```

Name	Size	Direction	Description
CLK	1	Input	Clock. The timer is synchronous on the rising edge of CLK.
CE	1	Input	Clock enable. When low, all internal registers are frozen and hold their current value.
RST	1	Input	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.
DSI	1	Input	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.
MODE	1	Input	Mode indicator. When low, the timer stops at zero. When high, the timer restarts at the initial value after it reached zero.
DI	32	Input	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.
DO	32	Output	Data Output. The 32-bits unsigned current value of the timer.
IRQ	1	Output	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.

Table 1: Interface specification of the TIMER module

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 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.