Exam 1, VHDL/PSL
due 18:00 April 24th, 2012

Emil Axelsson, Mary Sheeran and Thomas Hallgren

April 17, 2012

1 Important information

This is not a normal assignment, but part of the course examination. You are to do all of the work yourself. You are *not* allowed to cooperate or copy solutions (whole or partial). You should not even discuss your solution with anyone else. Act as if you were sitting in an exam hall with someone watching you. Cheating is unprofessional, and morally wrong. We will take a very dim view of any cheating that we discover, but feel hopeful that there will not be any. If you have questions about how to interpret the assignments, or have problems with the tools, do not hesitate to contact us. Also, if you get completely stuck, contact us (and not your friends). In that case, any assistance that we give will be noted and may influence your grade. More often, though, we just issue further information or explanation to all students. So do contact us! Remember also that exam solutions need not be perfect to pass.

2 The FIFO

In this exam you are going to implement a FIFO (First-In-First-Out) queue in VHDL, and verify properties about it. A FIFO is a memory into which you can store words one by one, and from which words can be read one by one. When a word is stored, it is placed last in the queue. When the FIFO is read from, it always gives out its first element, which is then removed from the queue so that the second element now becomes the first.

Your FIFO should be implemented according to Figure 1. It is a memory which takes two addresses: A_r for reading, and A_w for writing. When the signal *WE* (write enable) is high, the word at the input port D_{in} is stored in memory cell $M(A_w)$, and the pointer A_w then moves one step to the right. When the signal *RE* (read enable) is high, the word in memory cell $M(A_r)$ is fed to the output port D_{out}, and the pointer A_r then moves one step to the right. When a pointer reaches the end of the memory, it should wrap over to the first position.

The FIFO should have the interface shown in Figure 2. The generic parameters `width` and `length` give the number of bits per word and the number of words in the FIFO respectively.

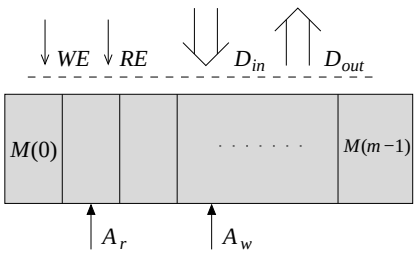

Figure 1: FIFO implementation

```
type fifo_status is record
                      level                  : natural;
                      empty, half_full, full : std_logic;
                    end record;

entity fifo is
  generic(width  : positive;
          length : positive);

  port( clk          : in  std_logic;
        reset        : in  std_logic;

        write_enable : in  std_logic;
        write_data   : in  unsigned(width-1 downto 0);
        write_error  : out std_logic;

        read_enable  : in  std_logic;
        read_data    : out unsigned(width-1 downto 0);
        read_error   : out std_logic;

        status       : out fifo_status );
end entity fifo;
```

Figure 2: FIFO interface

If the FIFO is full and we try to write a word to it, the signal `write_error` should be set to high. Likewise, if the queue is empty and we try to read from it, `read_error` should be set.

The `status` output combines some signals that should indicate the current FIFO status:

`level`	should give the number of elements currently stored in the queue
`empty`	should be high when and only when `level` $= 0$
`half_full`	should be high when and only when `level` $\geq$ `length`$/2$
`full`	should be high when and only when `level` $=$ `length`

These will come in handy when we want to specify certain properties about the FIFO.

The memory is defined as a vector of bit vectors:

```
type MemoryType is array (0 to length-1) of
     unsigned(width-1 downto 0);
```

We *require* you to use Gaisler's method and implement the whole FIFO in one entity with just two processes. For your guidance, a partial solution is provided in the file `fifoRemoved.vhdl` on the Assignments page. We strongly encourage you to make use of it.

3 Verification

The main task is to convince yourself and us that your design is correct. **Follow the guidelines on the homepage**. You should use formal verification for this.

It is not possible to completely specify the behaviour of the FIFO in PSL, so this is instead done by verifying smaller general properties. A good solution needs to formally verify all aspects of the FIFO; that is, your properties should cover all input and output signals somehow.

Since the PSL properties are incomplete, you probably want to complement the verification with a testbench in order to motivate correctness.

4 Extra assignment, gives extra credit up to max 20%

Design or find an interesting circuit in VHDL and formally verify some properties of it. Make clear how much of the design is your own work. Designs from other courses are acceptable here. We are mainly interested in the verification.

5 Administrative stuff

Electronic submission:

Submit

- All relevant files
- A report on the verification process

You will be reviewed for the following:

- The implementation of the FIFO. Since we have given you a partial solution, your implementation must be very close to correct. (10p)
- The verification of the FIFO, and how well it is documented and explained (20p).

Aim for clarity in your report. We like short clear reports. Try to demonstrate what you have learned!

Finally, remember to hand in whatever you have got at the deadline, even if you are unsatisfied with it. After all, that is what you have to do in a normal exam. Good luck!