Department of
Computer Science and Engineering
Chalmers and Gothenburg University

VT08
TDA956
08-05-30

EXAMINATION in
Hardware Description and Verification

DAY : 30/05 -08 TIME : 14:00-18:00 ROOM : Maskin-huset

Responsible for course	:	Mary Sheeran, tel 772 1013
Result	:	To be available at the latest 13/06-08
Permitted in the exam	:	Paper and pen. No books, PDAs etc.
Minimum points for each grade	:	CTH: 3 : 24, 4 : 36, 5 : 48 out of 60
	:	GU: G : 24, VG : 48 out of 60

PLEASE NOTE THE FOLLOWING

- Start each new question on a new page.
- Write your personal number (or date of birth) and name on each page.
- Write on only one side of each page.
- Attempt all questions.
- Read through all the questions first. Questions that look long may need short answers!

Question 1 (14 points)

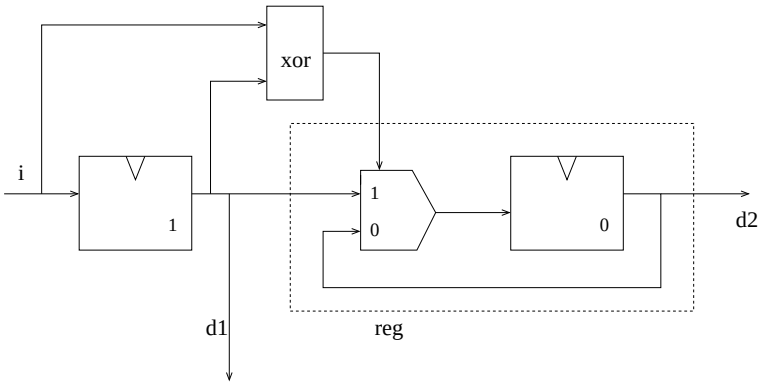
1. Briefly explain Gaisler's two process approach to design in VHDL. What are the advantages of this approach to design in VHDL, compared to an unstructured approach? Do you think that the method also eases the verification problem? Explain your answer. (4 points)

2. Based on your experience *as a user*, give the main advantages and disadvantages of a PSL-based approach to verification. (4 points)

3. Consider the problem of finite state machine (FSM) description and verification in a VHDL and PSL based design flow. Explain how FSMs are described in VHDL and how their behaviour is specified in PSL. Give small examples to make your explanation concrete. (Remember that you verified the FSM in your stopwatch in Lab 1.) (6 points)

Questions 2-5

Consider the following circuit:



This small circuit has one input **i**, and two outputs **d1** and **d2**. The dotted part corresponds to a one-bit register. A value is read into the register when the current input, and the input in the previous clock cycle are different from each other, and in that case it is the negation of the input that is read into the register. Here is a structural implementation in VHDL:

```
entity d_flip_flop is
  port(clk, i : in bit;
```

```
        o      : out bit);
end entity d_flip_flop;


entity mux is
  port(selector, i0, i1 : in bit;
        o               : out bit);
end entity mux;

architecture behavioral of mux is
begin
  o <= i0 when selector='0' else
        i1;
end architecture behavioral;


entity reg is
  port (clk, wrEn, i : in bit;
        o            : out bit);
end entity reg;

architecture structural of reg is
  signal do, di : bit;
begin
  o <= do;
  mux : entity work.mux(behavioural) port map (wrEn, do, i, di);
  dff : entity work.d_flip_flop(behavioural) port map (clk, di, do);
end architecture structural;


entity exor is
  port (i1, i2 : in bit;
        o      : out bit);
end entity exor;

architecture behavioural of exor is
begin
  o <= i1 xor i2;
end architecture behavioural;


entity exam_circuit is
  port (clk, i  : in bit;
        d1, d2: out bit);
```

```
end entity exam_circuit;

architecture structural of exam_circuit is
  signal diff, d1o, d2o : bit;
begin
  d1 <= d1o;
  d2 <= d2o;

  dff : entity work.d_flip_flop(behavioural) port map (clk, i, d1o);
  exor : entity work.exor(behavioural) port map (i, d1o, diff);

  reg : entity work.reg(structural) port map (clk, diff, d1o, d2o);
end architecture structural;
```

Question 2 (6 points)

1. Write a behavioural architecture for the **d_flip_flop** entity. The D flip flop outputs the same value throughout the entire clock cycle. At the rising edge of **clk**, it changes the output value to whatever value the input has at that particular moment. The **d_flip_flop** architecture does not need to have an explicitly defined initial value. (2 points)

2. Write a behavioural architecture for **examcircuit**. (4 points)
   Hint: There are many ways of doing this. However, using Gaisler's two-process method is one of the simpler ways.

Question 3 (5 points)

1. A property that holds of **examcircuit** is that if the two outputs are different from each other at one clock cycle, then they are also different from each other on the next clock cycle. Write the PSL property that expresses this. (2 points)

2. Express in PSL the property stated earlier: A value is read into the register when the current input, and the input in the previous clock cycle are different from each other, and in that case it is the negation of the input that is read into the register. (3 points)

Question 4 (9 points)

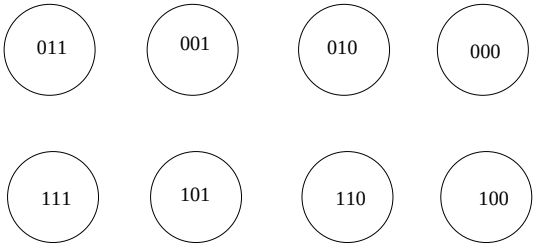
1. Write down the state transition system of **examcircuit**, in the following form:

$$(i, d1, d2) \rightarrow (i', d1', d2')$$
where
$$\vdots$$

In this description, i corresponds to the input, and $d1$ and $d2$ to the outputs of the two flip-flops. You should replace the dots with equations that define the correct values for i', $d1'$, and $d2'$ that are enforced by the circuit. Read $(i, d1, d2)$ as the *current* state, and $(i', d1', d2')$ as the *next* state. (2 points)

2. Draw the state transition diagram of this system. Indicate the initial states of the system, based on the information about the intitial values of the flip-flops given in the circuit diagram. (4 points)

Hint: A good way to organise the diagram is shown below:



3. Calculate the set of states for which $AG(\neg(d1 \leftrightarrow d2))$ holds using fixed point iteration (where $\leftrightarrow$ means logical equivalence).
Show *all* the steps of your calculation carefully. (3 points)

Hints: The following formulas are from Seger's paper:

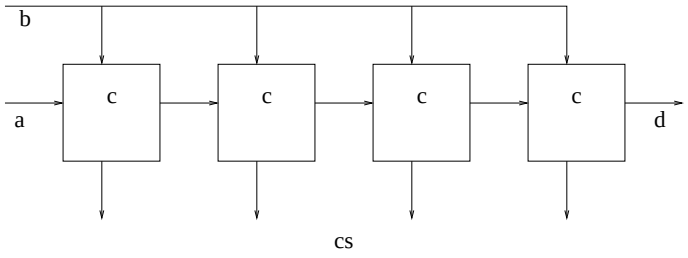
- $H(\neg f) = S - H(f)$
- $H(AGf) = Gfp\ \ U.\ \ H(f) \cap \{s | \forall t\ sRt \Rightarrow t \in U\}$

Question 5 (8 points)

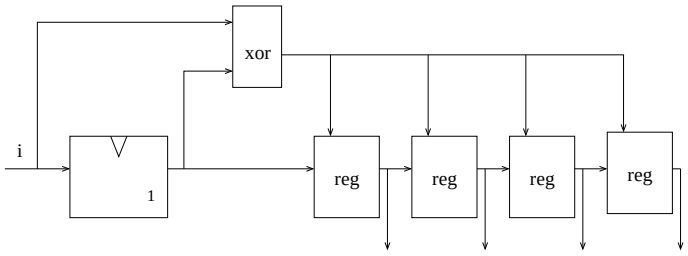
1. In Lava, define the circuit `examcircuit` considered in Questions 2-4, and pictured earlier. It has a single input `i` and two outputs `d1` and `d2`. Use the delay element (`delay`) for the D flip-flops, and the built in `mux` and `xor2` gates. Note that the initial values of the two D flip-flops are different. (2 points)

2. In Lava, write the code for an *observer* circuit `diff` that checks that its two inputs (which form a pair of bits) are different from each other. Show how you would use this observer (and Lava and SMV) to formally verify that the two outputs of `examcircuit` are always different from each other. (2 points)

3. In Lava, write a *connection pattern* called `rcon` that composes `n` identical sub-circuits, as shown below. The resulting circuit should have two inputs, `a` and `b`, where `b` goes to each of the cells. (**Hint:** It is ok to use the `row` connection pattern without giving its definition.) (2 points)



Consider a circuit that is similar to `examcircuit` but has a series of `n` registers, connected so that a value can be shifted along them. All registers receive the same `read` signal. The outputs of the circuit are the contents of each of the registers, that is a list of bits of length `n`. Call this circuit `examGen`.



4. Now, use `rcon` to define the `examGen` circuit.

```
examGen n init1 init2 i = ds
  where
    ???
```

The function `examGen` should take a parameter `n`, giving the number of one-bit registers composed, and two further parameters (`init1` and `init2`) that give the initial values of the left-hand flip-flop and of the registers (which all have the same initial value). (2 points)

Question 6 (9 points)

1. Construct the Ordered Binary Decision Diagram for the formula

   $(a \oplus b) \vee c$

   where $\oplus$ is *exclusive or* and $\vee$ is *or*. Start with the decision tree, and *show each step on the way to the BDD*. Choose the variable ordering a, then b, then c. (3 points)

2. Give two *advantages* of BDDs as a data structure? (2 points)

3. How are BDDs used in CTL model checking to represent the transition relation? (2 points)

4. It is possible to use a propositional (or Boolean) formula to represent a set of states. Assume that each state comprises three bits (a,b,c). Give a formula that represents the state set {(0,1,0),(0,0,1)}. (2 points)

Question 7 (9 points)

1. Briefly explain how regular expressions are used in PSL, giving example assertions that benefit from their use. (3 points)

2. Express in CTL the assertion "If the request line *Req* goes high, then eventually the acknowledge line *Ack* will go high.". (2 points)

3. Can the above assertion be expressed as an observer circuit in the Lava style? If so, how? If not, why not? (2 points)

4. Express in LTL the *mutual exclusion* assertion "It is impossible to get to a state in which both *critical1* and *critical2* hold." Is this a safety or a liveness property? (2 points)