E&CE 327 Midterm

2011t1 (Winter)

Instructions and General Information

- 100 marks total.   Time limit: 1 hour, 20 minutes
- No books, no notes, no computers.   Calculators are allowed
- If you need extra paper, request some from a proctor.
- Write neatly. To earn partial credit, you must show the formulas you are using and all of your work.
- **The proctors and instructors will not answer questions, except in cases where an error on the exam is suspected. If you are confused about a question, write down your assumptions or interpretation.**
- **Justifications of answers will be marked according to correctness, clarity, and conciseness.**

			Total Marks	**Approx. Time**	**Page**
Q0	!!Almost Free!!		2	2	2
Q1	VHDL Semantics		24	15	3
Q2	The Return of the Three		25	20	6
Q3	Dataflow Diagrams		26	20	8
Q4	Performance		25	20	10
Totals			100	77	

Name:______________________________

UWuserid:______________________________

Q0 (2 Marks) !!Almost Free!!

(*estimated time: 2 minutes*)

Q0a (1 Mark) Best part

What is the best part of the course?

Q0b (1 Mark) Most improve

What one thing could be done to most improve the course for the remainder of the term?

Potentially useful information

$$\log_x y = \frac{\log y}{\log x}$$

Q1 (24 Marks) VHDL Semantics

(*estimated time: 15 minutes*)

This question analyzes the VHDL semantics of the two processes below. Q1a deals with simulation steps, Q1b deals with simulation cycles, and Q1c deals with register-transfer-level simulation.

NOTES:

1. The marks in this question are dependent mostly upon the quality of the justifications, which will be marked according to *correctness, clarity, and conciseness.*
2. You do *not* need to draw delta-cycle or register-transfer-level simulation charts, but you *may* do so if it is helpful.

```
proc_a : process ( clk ) begin
  if rising_edge( clk ) then
    if ce = '1' then
      q <= d;
    end if;
  end if;
end process;
```

```
proc_b : process ( clk, ce ) begin
  if rising_edge( clk ) and ce = '1' then
    q <= d;
  end if;
end process;
```

Q1a (8 Marks) Differences in delta-cycle simulation chart

If you were to fill out two ECE-327 style delta-cycle simulation charts, one for `proc_a` and one for `proc_b`, is there a scenario where the input signals (`clk`, `ce`, and `d`) have the same behaviour, but the chart for `proc_a` would be different from the chart of `proc_b`? **For full marks, you must justify your answer.**

NOTES:

1. Ignore the difference that one chart has "`proc_a`" and one has "`proc_b`".
2. If there would be a difference, describe the behaviour on `clk`, `ce`, and `d` that would lead to the difference and describe the difference in the two charts.
3. This question includes differences at the granularity of *simulation steps*: an individual column in the delta-cycle simulation chart.

Q1b (8 Marks) Simulation-cycle differences

Is there a scenario where the signals `clk`, `ce`, and `d` have the same behaviour for both `proc_a` and `proc_b`, but where at the end of a *simulation cycle*, the signal `q` in `proc_a` has a different value than the signal `q` in `proc_b`? **For full marks, you must justify your answer.**

NOTES:

1. If there would be a difference, describe the behaviour on `clk`, `ce`, and `d` that would lead to the difference and describe the difference in the two `q` signals.

Q1c (8 Marks) Register-transfer-level differences

Is there a scenario where the signals `clk`, `ce`, and `d` have the same behaviour for both `proc_a` and `proc_b`, but where, when doing *register-transfer-level* simulation, the signal `q` in `proc_a` has a different behaviour than the signal `q` in `proc_b`? **For full marks, you must justify your answer.**

NOTES:

1. If there would be a difference, describe the behaviour on `clk`, `ce`, and `d` that would lead to the difference and describe the difference in the two `q` signals.

Q2 (25 Marks) The Return of the Three

(*estimated time: 20 minutes*)

NOTES:

1. For each of the code fragments Q2a–Q2e, answer whether the code is legal VHDL.
2. If the code is legal VHDL, answer whether it is synthesizable.
3. If the code is synthesizable, answer whether it adheres to good coding practices, according to the guidelines for E&CE 327.
4. If the the code is not legal, or is not synthesizable, or does not follow good coding practices, explain why.
5. Each of the code fragments is the body of an architecture.
6. The signals are declared as follows:

```
clk, a, b, c, d, d0, d1, q, x, y  :  std_logic
m, n                              :  unsigned(3 downto 0)
```

Q2a

```
process (clk) begin
  if rising_edge(clk) then
    q <= d0;
  else
    q <= d1;
  end if;
end process;
```

	Yes	No
Legal	☐	☐
Synthesizable	☐	☐
Good practice	☐	☐

Explanation:

Q2b

```
process (a, b, c) begin
  if a = '1' then
    x <= b and c;
    y <= b or  c;
  else
    y <= not b;
  end if;
end process;
```

	Yes	No
Legal	☐	☐
Synthesizable	☐	☐
Good practice	☐	☐

Explanation:

Q2c

```
m(0) <= n(0);
m_for : for i in 1 to 3 generate
  process ( m, n ) begin
    m(i) <= m(i-1) xor n(i);
  end process;
end generate;
```

	Yes	No
Legal	☐	☐
Synthesizable	☐	☐
Good practice	☐	☐

Explanation:

Q2d

```
process begin
  clk <= '0';
  wait for 10 ns;
  clk <= '1';
  wait for 10 ns;
end process;
```

	Yes	No
Legal	☐	☐
Synthesizable	☐	☐
Good practice	☐	☐

Explanation:

Q2e

```
process ( a, n ) begin
  if a = '1' then
    m <= to_unsigned( 0, 4 );
  else
    m <= n + 1;
  end if;
end process;
process ( a, m ) begin
  if a = '1' then
    n <= to_unsigned( 0, 4 );
  else
    n <= m + 2;
  end if;
end process;
```

	Yes	No
Legal	☐	☐
Synthesizable	☐	☐
Good practice	☐	☐

Explanation:

Q3 (26 Marks) Dataflow Diagrams

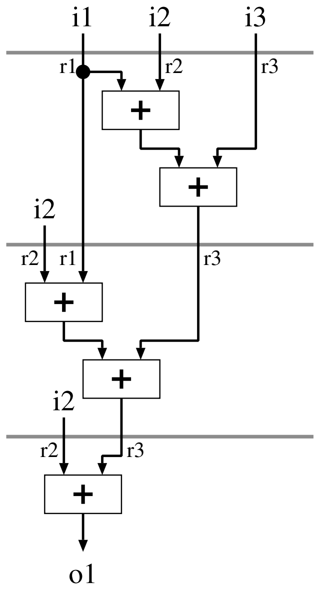
(*estimated time: 20 minutes*)

For each of the code fragments A, B, C, and D on the next page, answer whether the code fragment has the same behaviour as the dataflow diagram below.

If a VHDL code fragment does *not* have the same behaviour as the dataflow diagram, give the first signal (`r1`, `r2`, `r3`, or `o1`) and clock cycle where the behaviour differs, where clock cycles are counted from the first data values being on the inputs at cycle 0.

NOTES:

1. "Same behaviour" means that the registers and the output have the same values as in the dataflow diagram at the end of each clock cycle.
2. Each of the VHDL code fragments is legal, synthesizable VHDL.
3. The signal `clk` is declared as `std_logic` and each of the other signals is declared as `unsigned(7 downto 0)`.
4. Each of the code fragments is the body of an architecture.



Code fragment A

```
process begin
  wait until rising_edge(clk);
  r1   <= i1;
  r2   <= i2;
  r3   <= i3;
  add1 <= r1 + r2;
  add2 <= add1 + r3;
  wait until rising_edge(clk);
  r2   <= i2;
  r3   <= add2;
  add1 <= r1 + r2;
  add2 <= add1 + r3;
  wait until rising_edge(clk);
  r2   <= i2;
  r3   <= add2;
  add1 <= r2 + r3;
  o1   <= add1;
end process;
```

Same behaviour		First difference	
Yes	No	Signal	Cycle
☐	☐	________	________

Code fragment B

```
process begin
  wait until rising_edge(clk);
  r1 <= i1;
  r2 <= i2;
  r3 <= i3;
  wait until rising_edge(clk);
  r2 <= i2;
  r3 <= r1 + r2 + r3;
  wait until rising_edge(clk);
  r2 <= i2;
  r3 <= r1 + r2 + r3;
end process;
o1 <= r2 + r3;
```

Same behaviour		First difference	
Yes	No	Signal	Cycle
☐	☐	________	________

Code fragment C

```
process begin
  wait until rising_edge(clk);
  r1   <= i1;
  wait until rising_edge(clk);
  wait until rising_edge(clk);
end process;
process begin
  wait until rising_edge(clk);
  r2   <= i2;
  wait until rising_edge(clk);
  r2   <= i2;
  wait until rising_edge(clk);
  r2   <= i2;
end process;
process begin
  wait until rising_edge(clk);
  r3   <= i3;
  wait until rising_edge(clk);
  r3   <= r1 + r2 + r3;
  wait until rising_edge(clk);
  r3   <= r1 + r2 + r3;
end process;
o1 <= r2 + r3;
```

Same behaviour		First difference	
Yes	No	Signal	Cycle
☐	☐	________	________

Code fragment D

```
process begin
  wait until rising_edge(clk);
  r1   <= i1;
end process;
process begin
  wait until rising_edge(clk);
  r2   <= i2;
end process;
process begin
  wait until rising_edge(clk);
  r3   <= i3;
  wait until rising_edge(clk);
  r3   <= add2;
  wait until rising_edge(clk);
  r3   <= add2;
end process;
add1 <= r1   + r2;
add2 <= add1 + r3;
o1   <= r2 + r3;
```

Same behaviour		First difference	
Yes	No	Signal	Cycle
☐	☐	________	________

Q4 (25 Marks) Performance

(*estimated time: 20 minutes*)

You are developing a new design of the Waterluvian filter that will be implemented on an FPGA chip (the definition of a Waterluvian filter is irrelevant to this question). Your project leader just returned from a conference where she learned about a new FPGA chip. She has asked you to analyze the tradeoffs between remaining with your current FPGA chip, or delaying the project and switching to the new chip.

- The current prediction is that with the current FPGA chip, the average Waterluvian filter at the time you complete the project will have the 5% more area than your filter and be 25% faster than your filter.
- The average *performance* of Waterluvian filters doubles every 24 months.
- The average *area* of the circuits that implement Waterluvian filters has remained constant over time.
- The new FPGA chip claims that it will provide a 20% improvement in clock speed with just 85% of the area of the FPGA chip you are using now.
- The *optimality* of Waterluvian filters is measured by the ratio of performance to area.

If you keep your design the same and switch to the new chip, how long can you delay your project and have an *optimality* that is at least 7% more than the average *optimality* at the time you complete the project?

If you cannot achieve an optimality that is 7% more than the average, then calculate the optimality of your filter compared to the average if you use the new FPGA chip and delay the project by four months.

If you do not know how to answer the question for optimality, you may earn part marks by writing a $\surd$ in the box at the end of this question and answering the question for *performance* rather than optimality.

I am answering for *performance*, not optimality ☐