E&CE 327 Midterm Solution

2010t1 (Winter)

All requests for re-marks must be submitted in writing to Mark Aagaard before noon on Wed March 3.

A random collection of midterms were photocopied. Exams that are submitted for re-marking will be verified against this set.

Q1 (8 Marks) Short Answer

(estimated time: 5 minutes)

What are the two goals of zero-delay simulation?

Answer:

1. Make events appear to propagate instantaneously through combinational logic.
2. Make gates appear to operate in parallel.

How does VHDL delta-cycle simulation achieve these goals?

Answer:

1. A delta cycle is an infinitesimally small amount of time.
2. The effect of a provisional assignment is not visible until the next simulation cycle.

Marking:

2 marks *per answer*

Q2 (23 Marks) VHDL Semantics

(estimated time: 18 minutes)

In this question, you will describe the behaviour of the code below based on the delta-cycle simulation semantics of VHDL.

NOTES:

1. The clock period is 33 ns.
2. The code is legal VHDL.

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

entity sim_cyc is
  port (
    clk, a, b, c : in std_logic;
    z            : out std_logic
  );
end entity;

architecture main of sim_cyc is
  signal d, e, f, g, h, i, j, k : std_logic;
begin

  proc_1 : process begin
             wait until rising_edge(clk);
             d <= a;
             e <= b;
             f <= c;
             z <= k;
           end process;

  proc_2 : process ( d, k ) begin
             g <= not d;
           end process;

  proc_3 : process (d, e, g, j) begin
             h <= d xor e;
             k <= g nand j;
           end process;

  proc_4 : process (h, f, i) begin
             i <= h and f;
             j <= not i;
           end process;

end architecture;
```

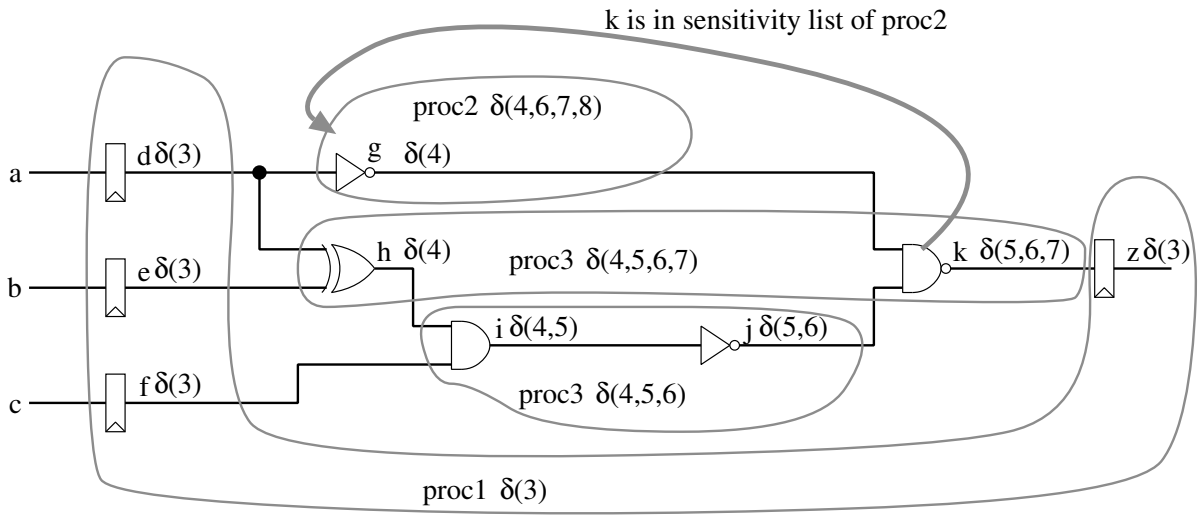
For the code on the previous page, what is the maximum number of simulation cycles that can occur in a single simulation round? Answer the question by giving a **brief** description of what happens in each simulation cycle of the longest simulation round. Also, for each simulation cycle, answer whether the simulation cycle is a delta cycle.

NOTES:

1. Your descriptions should be brief: do *not* include all of the information that is in a waveform diagram. The goal of this question is to evaluate your understanding of delta-cycle simulation semantics without the tedious details of working through an actual delta-cycle simulation.

Answer:

1. *Delta: Yes Time advances*
2. *Delta: Yes an external process that drives the clock is run, proc_1 becomes postponed.*
3. *Delta: Yes proc1 runs. d,e,f,z change value*
4. *Delta: Yes proc2, proc3, and proc4 run. g,h,i change value.*
5. *Delta: Yes proc3 and proc4 run. k, i, j change value.*
6. *Delta: Yes proc2, proc3, and proc4 run. j, k change value.*
7. *Delta: Yes proc3 and proc3 run. k changes value*
8. *Delta: NO proc2 runs.*



Marking:

For answers with a good sequence of sim cycles:

3 marks no *sim cycle for inputs*

3 marks *time advances in first sim cycle*

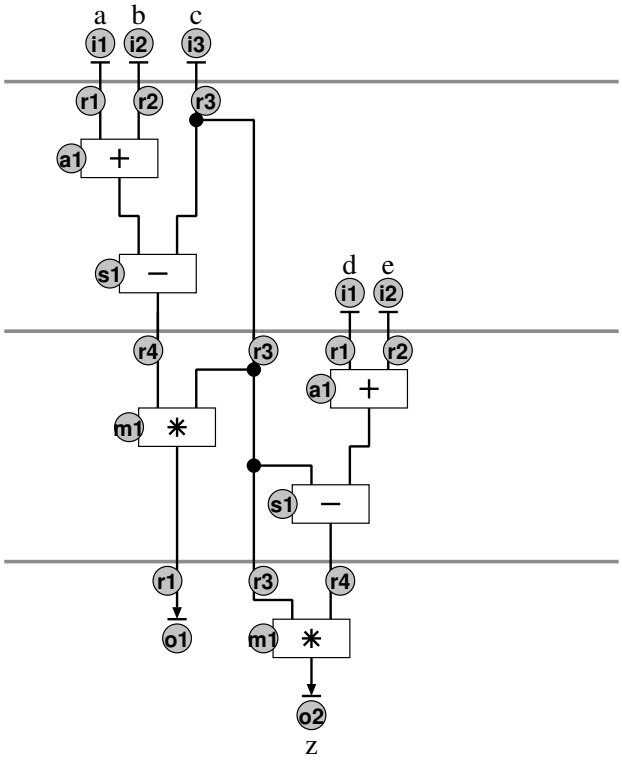
3 marks *proc_1 runs*

3 marks *multiple processes run in same sim cycle*
3 marks *seq of: value change to process running based on sens list*
3 marks *proc_2 runs at end*
3 marks no *sim cycle for z at the end*
2 marks *all but last sim cycles are delta cycles*

Q3 (23 Marks) Dataflow Diagram

(estimated time: 18 minutes)

Before leaving for his annual ski trip to St. Moritz, your manager, Ilov H. Del, assigned you the task of designing the control signals for a dataflow diagram that he was working on.



Q3a (12 Marks) Control Signals

List the control signals (multiplexer select, chip-enable, *etc*) needed for each register and datapath component in the dataflow diagram. If a register or datapath component does not need any control signals, write "NONE". For each signal, give a *brief* description of its purpose.

NOTES:

1. The primary purpose of the description is to enable the marker to understand Q3b, where you will draw waveforms for the signals. You will *not* be marked upon the completeness of the descriptions, so do not waste time with detailed descriptions.
2. You may perform optimizations that simplify the control signals, so long as they do *not* affect: allocation, scheduling, or latency.
3. The state encoding scheme has not been chosen yet.

Answer:

r1_sel	*2:1 mux sel for r1 between i1 and m1*
r2	*NONE*
r3_ce	*chip enable for r3*
r4	*NONE*
a1	*NONE*
s1_src1	*2:1 mux sel for s1 src1 between a1 and r3*
s1_src2	*2:1 mux sel for s1 src2 between r3 and a1*

Marking

1 marks	chip enable for some registers
1 marks	mux sel for some registers
1 marks	src1 mux sel for some datapath components
1 marks	src2 mux sel for some datapath components
−1 marks	each extra category of control signal (e.g. reset, chip enable for datapath component)
2 marks	correct inputs to mux select signals
2 marks	constant ce means no ce needed (r1,r2, r3, r4)
2 marks	no mux for a1
−2 marks	used commutivity to remove muxes for s1
2 marks	no mux for m1

Q3b (11 Marks) Signal Behaviour

Draw a waveform diagram showing the behaviour of the control signals for 5 clock cycles beginning when a new set of data is available on the input pins.

NOTES:

1. Put a $\surd$ at each clock cycle where a new set of data can enter the circuit on the input pins.
2. Do *not* select values for don't cares; write "DC" as the value for "don't care".
3. For the values of multiplexer select signals, do *not* use "0" and "1", instead use the name of the component, register, or input that is selected; or "DC".
4. If you are unsure of what notation to use, choose one that is simple, and then describe it.

	Time (clock cycles)				
Signal name	0	1	2	3	4
New data at inputs	$\surd$			$\surd$	
r1_sel	i1	i1	m1	i1	i1
r3_ce	1	0	0	1	0
s1_src1	DC	a1	r3	DC	a1
s1_src2	DC	r3	a1	DC	r3

Marking

2 marks	alignment of register signals to clock cycles
2 marks	alignment of dp signals to clock cycles
2 marks	repeat input pattern for 2nd set of data
2 marks	time when load new data
3 marks	values of signals 3 = 0 mistakes, 2=1-2 mistakes, 1=3+ mistakes

Q4 (23 Marks) Code Review

(estimated time: 20 minutes)

You are supervising an intern from some UTher school and are responsible for reviewing the intern's VHDL code. The intern has compiled the `count_gt` process on the next page, but has not simulated or synthesized it yet. Your task is to write comments that describe the five most important changes that should be made to the code.

NOTES:

1. The purpose of the code is to receive a sequence of 256 data values and count the number of data values that are greater than 63.
2. The input data are unsigned 8-bit numbers.
3. For each data value, `i_valid` is asserted ('1') for exactly one clock cycle, followed by 0 or more clock cycles of bubbles.
4. The input data `i_data` is valid only when `i_valid` is asserted.
5. The types of `i_data` and `o_count` must be `std_logic_vector`.
6. If a comment applies to multiple signals or processes, list the comment just once.
7. Keep the comments focused and try to preserve as much of the intern's code as possible. Try to avoid large structural changes to the code, such as "Combine the three processes into a single process."
8. If you cannot find five changes that will improve the code, then give positive comments about the good features in the code.

```
1)  library ieee;
2)  use ieee.std_logic_1164.all;
3)  use ieee.numeric_std.all;
4)
5)  entity count_gt is
6)    port (
7)      reset, clk, i_valid : in  std_logic;
8)      i_data              : in  std_logic_vector( 7 downto 0 );
9)      o_done              : out std_logic;
10)     o_count             : out std_logic_vector( 7 downto 0 )
11)   );
12) end entity;
13)
14) architecture main of count_gt is
15)   signal gt_count, data_count : unsigned( 7 downto 0 );
16)   signal done : std_logic;
17) begin
18)
19)   process (clk) begin
20)     if rising_edge( clk ) then
21)       if reset = '1' then
22)         data_count <= (others => '0');
23)       elsif i_valid = '1' then
24)         data_count <= data_count + 1;
25)       else
26)         data_count <= data_count;
27)       end if;
28)     end if;
29)   end process;
30)
31)   process (clk) begin
32)     if rising_edge( clk ) then
33)       if reset = '1' then
34)         gt_count <= (others => '0');
35)       elsif unsigned(i_data) > 63 then
36)         gt_count <= gt_count + 1;
37)       end if;
38)     end if;
39)   end process;
40)
41)   process begin
42)     wait until rising_edge( clk );
43)     if reset = '1' then
44)       o_done = '0';
45)     elsif data_count >= 256 then
46)       o_done <= '1';
47)     end if;
48)   end process;
49)
50)   o_count <= std_logic_vector(gt_count);
51)
52) end architecture;
```

The table below lists the different categories of comments in their order of importance, from most important to least important.

Each category has a key (B, S, A, C, or P). For each comment, write the key of the comment's category. If a comment fits into more than one category, write down the most important category that the comment fits into.

Key	Category
B	*bug* fixes
S	making the code *synthesizable*
A	decrease the *area*
C	making the *code* simpler and more elegant
P	a *positive* comment

Answer:

5 marks each:

1. **B***:* `gt_count` *should be incremented only when* `i_valid=1`
2. **B***:* `o_count`, `gt_count`, *and* `data_count` *should all be 9 bits wide, so that they can count all the way up to 256. With 8 bits, we can count only up to 255. (3 marks for noticing that 256 can't be represented in 8 bits and not recommending to expand data to 9 bits).*
3. **B***:* `gt_count` *keeps counting after* `o_done` *is true.*
4. **B***:* `o_done = '0'` *should be* `o_done <= '0'`*. (This is a typo on the exam, it contradicts claim that code was compiled).*

4 marks each:

1. **A***:* `data_count <= data_count` *will probably synthesize to mux. The* `else` *clause should be deleted. The code will then synthesize to a register with a chip-enable. (2 marks for saying to delete the assignment without explanation or justification.)*
2. **A***: The condition* `i_data>63` *should be optimized to check just the two most significant bits. Similarly,* `data_count>=256` *should check just the MSB.*
3. **A***:* `o_done` *could be done combinationally*

3 marks each:

1. **C***:* `done` *not used, delete the signal declaration*
2. **C***: The entity and a signal are both named* `gt_count`*, this is legal but confusing. One of the names should be changed.*

2 marks each: particularly insightful positive comment

1 mark each: a comment that demonstrated knowledge of course material

Q5 (23 Marks) Performance

(estimated time: 15 minutes)

You work for Dark Silicon on their Waterluvian filter project, which run on FPGAs. At last year's Wonderful Waterluvian World Conference (W3C), you demonstrated the D1 Waterluvian filter. You are now working on the D2, which will be demonstrated at this year's W3C.

Another company, Fluffville Software, makes a software based Waterluvian filter.

Your project leader has gathered some data and rumours about Fluffville's Waterluvian filter. Assuming that the data and rumours that she has are true, she has asked you to predict the performance that the D2 must achieve to be faster than Fluffville's Waterluvian filter at this year's W3C.

NOTES:

1. At last year's W3C, Fluffville's filter (the F1) was 10% faster than the D1.
2. There's a rumour that Fluffville will demonstrate their new F2 filter a month before this year's W3C and will demonstrate the even faster F3 at W3C.
3. The rumours are that the F2 will be 1.80 times the speed of the F1. The performance of the F3 is still unknown.
4. Assume that Fluffville is able to sustain their rate of performance increase from the F2 to the F3.
5. Your goal is to develop the D2 Waterluvian filter in time for W3C such that the D2 will be faster than the F3.

How much faster than the D1 does the D2 need to be, in order for the D2 to be faster than the F3 at this year's W3C?

Answer:

Use P_{d1} for the performance of the D1 filter and corresponding subscripts for the performance of the other filters.

Basic equation for performance increase over time: $P(t) = P(0) \times k^{t/n}$.

Equations based on given information:

$$\begin{aligned} P_{f1} &= 1.10 \times P_{d1} \\ P_{f2} &= 1.8 \times P_{f1} \\ P_{f3} &= P_{f1} \times 1.8^{12/11} \\ P_{d2} &= P_{f3} \end{aligned}$$

Solve for P_{d2}/P_{d1}

$$\begin{aligned} P_{d2} &= P_{f1} \times 1.8^{12/11} \\ &= 1.10 \times P_{d1} \times 1.8^{12/11} \end{aligned}$$

$$\begin{aligned} P_{d2}/P_{d1} &= 1.10 \times 1.8^{12/11} \\ &= 2.09 \end{aligned}$$

The performance of the D2 must be 2.09 times the performance of the D1. Stated another way, the D2 must be 109% faster than the D1.

Marking:

23 marks *Correct answer*

20 marks *Interpreted "sustain rate of performance increase" to mean* $P_{f3}/P_{f2} = P_{f2}/P_{f1}$

18 marks *Large amount of correct work*

17 marks *Treated performance increase as linear*

14 marks *Linear equations and large amount of correct work*

10 marks *Basic equation for performance increase over time and some correct work.*

8 marks *Some performance equations and calculations*

5 marks *Some correct work*

–1 mark *Simple arithmetic mistake*

–2 marks *No units on answer*