E&CE 327 Midterm Solution

2012t2 (Spring)

All requests for re-marks must be submitted in writing to Mark Aagaard before 10:30am on Friday June 29.

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

Q1 (25 Marks) Short Answer

(estimated time: 20 minutes)

Q1a (4 Marks) VHDL Semantics

What are the two defining characteristics of zero-delay simulation, and how does VHDL simulation achieve these characteristics?

Answer:

1. *Events propagate instantaneously through combinational logic. VHDL achieves this with* delta cycles.
2. *Gates operate in parallel. VHDL achieves this with* provisional assignments

.

Marking:

1 mark *each fully correct answer*

Q1b (7 Marks) Idle State

When is an idle state needed in a control table?

Answer:
An idle state is needed if we have interparcel variables *and an* indeterminate number of bubbles.

Marking:

2 marks *interparcel variables*
2 marks *indeterminate number of bubbles*

What could go wrong if you do not include an idle state when one is needed?

Answer:
The data in registers used to hold inter-parcel variables could be corrupted, because without the idle state, the control table might have the chip-enable turned on when there are bubbles in the system.

Marking:

2 marks *data corrupted*
1 mark *corruption done by bubbles or garbage*

Q1c (7 Marks) Unused

When must the "unused" symbol be used in a control table?

Answer:
Whenever the conditions for two rows of the control table might be true at the same time, we must mark each resource as "unused" in one of the two rows.

Marking:

2 marks *two rows of control table*
2 marks *conditions for both rows true at same time*
part marks: 2 marks *parcels at input and output at same time*

What could go wrong if you do not use the "unused" symbol when one is needed?

Answer:
We could have contradictory assignments for a resource, which would lead either to a value of `'X'` *or buggy behaviour.*

Marking:

2 marks *two inputs driving same signal*
1 mark *unknown,* `'X'`*, or buggy output*

Q1d (7 Marks) State Encoding

Which state encoding is probably the best choice for a system that has a *latency of 3 clock cycles* and where there are *at least 2 bubbles* between each pair of valid parcels?

Either write down the name of the encoding, or describe the encoding. If you write the name of the encoding, you do not need to describe it.

Answer:
Valid bits

Marking:

7 marks *valid-bit encoding*
4 marks *one-hot encoding*
2 marks *binary or gray encoding*

Q2 (20 Marks) VHDL Simulation

(estimated time: 15 minutes)

For the VHDL program below, how many *simulation cycles* will occur in the simulation round that begins at 10ns?

Answer the question by using the list on the next to page to give a **brief** description of what happens in each simulation cycle of the simulation round.

```
entity simple is
end entity;

architecture main of simple is
  signal clk, a, b, c, d, e, f, g : std_logic;
begin
  proc1 : process begin
    clk <= '0';
    wait for 10 ns;
    clk <= '1';
    wait for 10 ns;
  end process;
  proc2 : process begin
    a <= '0';
    b <= '1';
    wait for 15 ns;
    a <= '1';
    wait for 20 ns;
  end process;
  proc3 : process begin
    wait until rising_edge(clk);
    c <= a;
  end process;
  proc4 : process (a, c, e) begin
    d <= a xor c;
    f <= not e;
  end process;
  proc5 : process (d) begin
    e <= d;
  end process;
  proc6 : process begin
    wait until rising_edge(clk);
    g <= f;
  end process;
end architecture;
```

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 VHDL simulation semantics without all of the tedious details of the full waveform diagram.

Answer:

1. *time advances*

2. *proc1 : rising edge on clock*
3. *proc3: c: U → 0; proc6: g: U → U*
4. *proc4: d: U → 0; f: U → U*
5. *proc5: e: U → 0*
6. *proc4: d: 0 → 0; f: U → 1*

Marking:

examples of independent mistakes

- *in first simulation cycle, time does not advance*
- *only one process runs in a simultation cycle*
- *new processes become postponed before all currently postponed processes have had a chance to run*
- *time increments at end of simulation round*
- *missing that a process runs*
- *extra running of a process*

-3 marks *one mistake*

-6 marks *two mistakes*

-8 marks *three mistakes*

-9 marks *four mistakes*

-10 marks *five mistakes*

-11 marks *six mistakes*

10 marks max *RTL simulation*

Q3 (20 Marks) The Good, the Bad, and the Unsynthesizable *(estimated time: 15 minutes)*

NOTES:

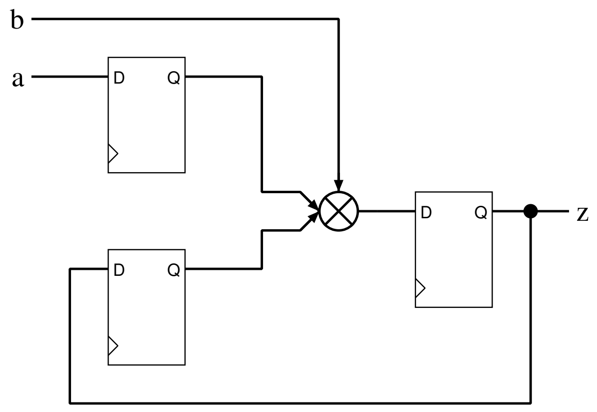
1. For each of the code fragments Q3a–Q3e, answer whether the code is legal VHDL.
2. If the code is legal VHDL, answer whether it is synthesizable.
3. If the code is synthesizable:
   (a) Answer whether it adheres to good coding practices, according to the guidelines for E&CE 327.
   (b) Draw the hardware that would be most likely to result from synthesizing the code.
   (c) If the VHDL code includes an implicit state machine: draw the gates, wires, and flops for the datapath. You may draw the control portion of the circuit as a cloud or black-box that drives the appropriate signals in the datapath. Draw all of the connections between the implicit state machine and the datapath, inputs, and outputs.
4. If the the code is not legal, or is not synthesizable, or does not follow good coding practices, explain why.
5. All signals are `std_logic` and are internal to an architecture.
6. You do not need to draw the signal `clk` if it is used as the clock input to a flip-flop. You *must* draw the signal `clk` if it is used for *something other than a clock input to a flip-flop in the datapath.*
7. If a signal other than `clk` is used as the clock-input to flip-flop in the datapath, then you must draw that signal.

Q3a

```
process (clk) begin
  if rising_edge(clk) then
    d0 <= z;
    d1 <= a;
    if b = '1' then
      z <= d1;
    else
      z <= d0;
    end if;
  end if;
end process;
```

Answer:

Legal, synth, good

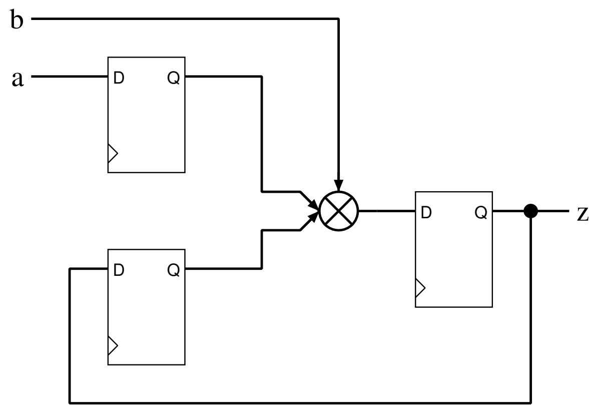


Q3b

```
process (clk) begin
  if rising_edge(clk) then
    d0 <= z;
    d1 <= a;
  end if;
end process;
process begin
  wait until rising_edge(clk);
  if b = '1' then
    z <= d1;
  else
    z <= d0;
  end if;
end process;
```

Answer:

Legal, synth, good

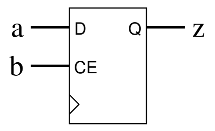


Q3c

```
process (clk) begin
  if rising_edge(clk) then
    if b = '1' then
      z <= a;
    end if;
  end if;
end process;
```

Answer:

Legal, synth, good



Q3d

```
process (clk) begin
  if rising_edge(clk) then
    d0 <= z;
    d1 <= a;
    s  <= b;
  end if;
end process;

if s = '1' generate
  z <= d1;
end generate;
if s = '0' generate
  z <= d0;
end generate;
```

Answer:

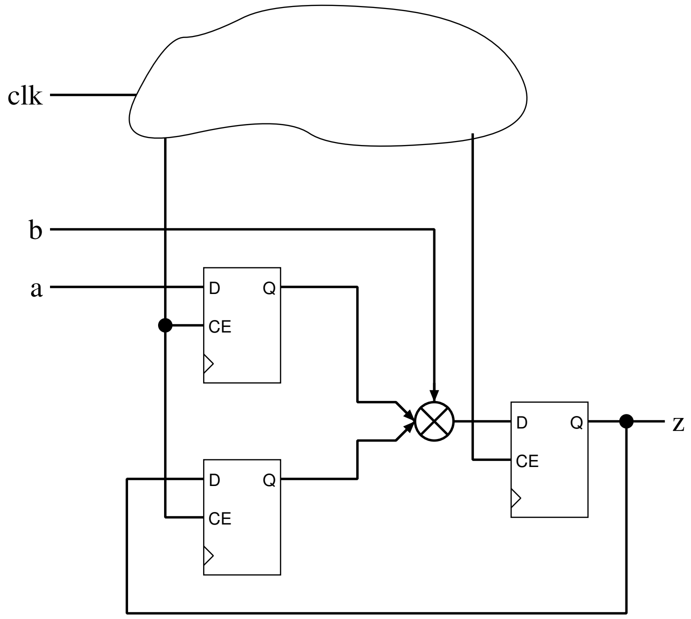
Illegal: conditions for generate statements must be static.

Q3e

```
process begin
  wait until rising_edge(clk);
  d0 <= z;
  d1 <= a;
  wait until rising_edge(clk);
  if b = '1' then
    z <= d1;
  else
    z <= d0;
  end if;
end process;
```

Answer:

Legal, synth, good



Marking:

Overview *The marks are divided into two categories: baseline and explanation/hardware. Baseline is what the student earns for the legal/synth/good portion. The explanation/hardware part is for the quality of the explanation or the correctness of the hardware drawing.*

Q3a,b,c,e *(correct answer = legal,synth,good)*

	Student answer			
	legal Yes synth Yes good Yes	*legal Yes synth Yes good No*	*legal Yes synth No*	*legal No*
base	*1.5*	*1*	*0.5*	*0*
explanation/hw	*2.5*	*2*	*1.5*	*1*
total max possible	*4*	*3*	*2*	*1*

Q3d *(correct answer = illegal)*

	Student answer			
	legal No	*legal Yes synth No*	*legal Yes synth Yes good No*	*legal Yes synth Yes good Yes*
base	*1.5*	*1*	*0.5*	*0*
explanation/hw	*2.5*	*2*	*1.5*	*1*
total max possible	*4*	*3*	*2*	*1*

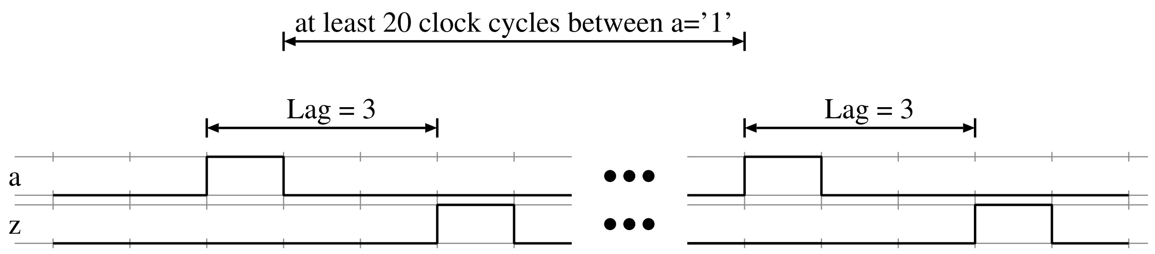
Q4 (20 Marks) State Machines

(estimated time: 15 minutes)

The intended behaviour of the state machines below is that whenever the input `a='1'`, the machine will generate `z='1'` some number of clock cycles later (See waveform below). The number of clock cycles between `a='1'` and `z='1'` is called the "lag" of the system. Each state machine may have a different lag.

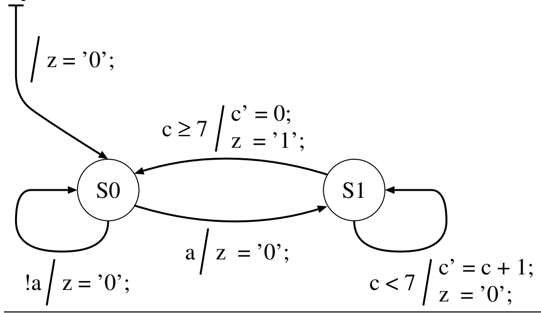
NOTES:

1. For each of the state machines Q4a–Q4c, answer whether the state machine is correct with respect to the intended behaviour.
2. If the state machine is correct, answer what its lag is.
3. If the state machine is incorrect, explain either how the machine's behaviour differs from the intended behaviour or how the state machine could be modified to fix the incorrect behaviour.
4. The state machines are allowed to ignore the value of `a` in the first clock cycle.
5. The input `a` is guaranteed to be `'0'` for at least 20 clock cycles between each pair of clock cycles when `a` is `'1'`.



Example waveforms with a lag of 3 clock cycles

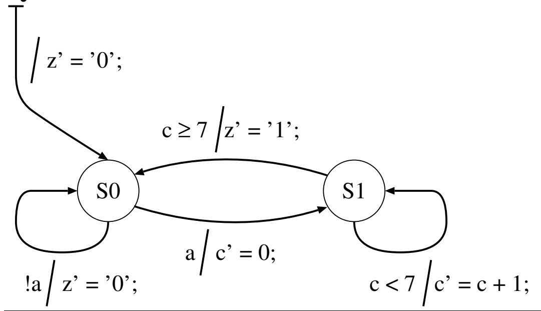
Q4a



Answer:

Incorrect. Missing initialization of c. The assignment `c' = '0'` should be moved to the lower edge (the one from S0 to S1). If said "correct", the answer for lag should be 8

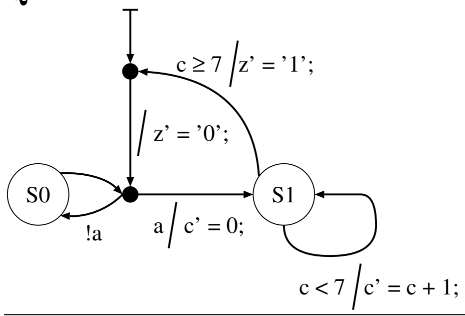
Q4b



Answer:

Correct, lag=9

Q4c



Answer:

Incorrect. The signal z is never '1', because the assignment z' = '1' is followed immediately by z' = '0' in the same clock cycle. If said "correct", the answer for lag should be 9.

Marking:

Q4a,c *(correct answer is "buggy")*

student answered "ok"

4 marks	*correct lag*
3 marks	*lag +1*
2 marks	*lag +2 or -1*

student answered "buggy"

7 marks	*correct and complete*
5 marks	*partially correct and complete*
5 marks	*correct and incomplete*
3 marks	*some correct info*

"partially correct" means that there is some incorrect information in the answer.

Q4b *(correct answer is "ok, lag=9")*

student answered "ok"

6 marks	*lag=9*
5 marks	*lag=8*
4 marks	*lag=7 or 10*
3 marks	*lag=6 or 11*
-1 mark	*gave lag for Q4a or Q4c and lag is inconsistent with Q4a or Q4c*

student answered "buggy"

4–5 marks	*identified one bug*
2–4 marks	*identified two bugs*

Common problems

- *confusion on comb vs reg asn*
- *confusion on transient state vs normal state*
- *confusion on when reg asn are visible*
- *forget that need to write to variable before read*
- *Q4b: add z' = '0' on S0 to S1*
- *Q4b: add c' = 0 on S1 to S0*

Q5 (15 Marks) Dataflow Diagram with Memory

(estimated time: 10 minutes)

Draw a dataflow diagram that implements the specification:

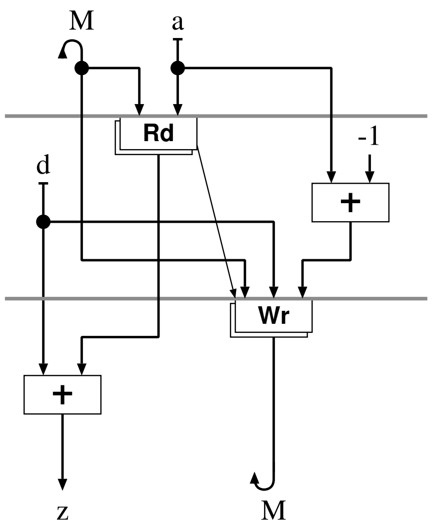
```
M[a-1] = d;
z      = M[a] + M[a-1];
```

NOTES:

1. Inputs shall be *registered*
2. Outputs shall be *combinational*
3. Memory has registered inputs and combinational outputs (same as in class)
4. The memory is *single-ported*
5. There are exactly *five bubbles* between each pair of consecutive valid parcels.
6. Optimization goals in order of decreasing importance:
   (a) minimize *area*
      i. registers (excluding memory)
      ii. total of (number of adders + number of subtracters)
      iii. input ports
      iv. output ports
   (b) minimize *latency*
7. Input values may be read in any clock cycle, but each input value shall be read exactly once.
8. Optimizations to the pseudocode are allowed, so long as the values on z and M are correct.

Answer:

```
M[a-1] = d;
z      = M[a] + M[a-1];
```



Marking:

2 marks	*functional correctness*
2 marks	*DFD syntax is correct*
1 mark	*DFD uses M*
1 mark	*Wr produces M*
1 mark	*anti-dependency arrow*
1 mark	*mem operations on clock cycle boundaries*
1 mark	*at most one mem operation per clock cycle*
1 mark	*registered inputs*
1 mark	*combinational outputs*
1 mark	*2 registers*
1 mark	*1 input*
1 mark	*1 arithmetic unit (convert a-1 to a+(-1))*
1 mark	*optimal latency*