First Name _____ **Last Name** _____ **First let of last na** [ ]

UW Userid _____

ECE 327 Midterm

2013t1 (Winter)

Instructions and General Information

- 100 marks total. Time limit: 1 hour, 20 minutes (80 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 concision.**

			Total Marks	Approx. Time	Page
Q0	!!Almost Free!!		2	2	2
Q1	VHDL Semantics		20	15	3
Q2	Dataflow Diagram and Control Table		20	15	4
Q3	The Sleet, the Snow, and the Slush		20	15	6
Q4	State Machines		20	15	9
Q5	Dataflow Diagram with Memory		20	15	11
Totals			100	77	

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?

Q1 (20 Marks) VHDL Semantics

(*estimated time: 15 minutes*)

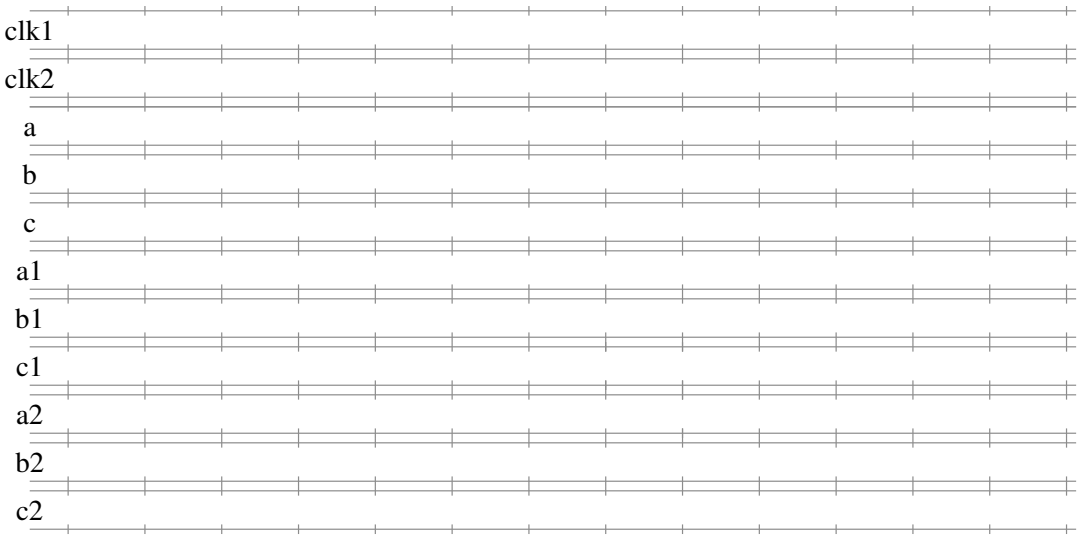
For the VHDL program below, calculate the values for the signals `a1`, `b1`, `c1`, `a2`, `b2`, and `c2` at 15ns.

NOTES:

1. All of the processes are in the *same architecture*.
2. The signals `a1`, `b1`, `c1`, `a2`, `b2`, and `c2` are declared to be `unsigned(3 downto 0)`.
3. **For full marks you must justify your answer, using text and/or the waveform diagram. You may, but are *not* required to, show a delta-cycle simulation.**

```
process begin
  clk1 <= '0';
  wait for 10 ns;
  clk1 <= '1';
  wait for 10 ns;
end process;

process begin
  a <= to_unsigned(  1, 4 );
  wait for 10 ns;
  a <= to_unsigned(  2, 4 );
  wait for 20 ns;
  a <= to_unsigned(  3, 4 );
  wait for 20 ns;
end process;

process begin
  b <= to_unsigned(  7, 4 );
  wait until rising_edge(clk1);
  b <= to_unsigned(  8, 4 );
  wait until rising_edge(clk1);
  b <= to_unsigned(  9, 4 );
  wait until rising_edge(clk1);
end process;
```

```
process (clk1) begin
  clk2 <= clk1;
end process;

process begin
  c <= to_unsigned( 10, 4 );
  wait until rising_edge(clk2);
  c <= to_unsigned( 11, 4 );
  wait until rising_edge(clk2);
  c <= to_unsigned( 12, 4 );
  wait until rising_edge(clk2);
end process;

process begin
  wait until rising_edge(clk1);
  a1 <= a;
  b1 <= b;
  c1 <= c;
end process;

process begin
  wait until rising_edge(clk2);
  a2 <= a;
  b2 <= b;
  c2 <= c;
end process;
```



	value at 15ns
a1	
b1	
c1	
a2	
b2	
c2	

Q2 (20 Marks) Dataflow Diagram and Control Table

(*estimated time: 15 minutes*)

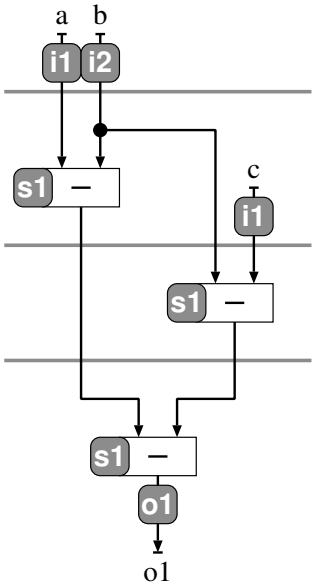
In this question you will do register allocation and draw the control-table for the dataflow diagram shown below.

NOTES:

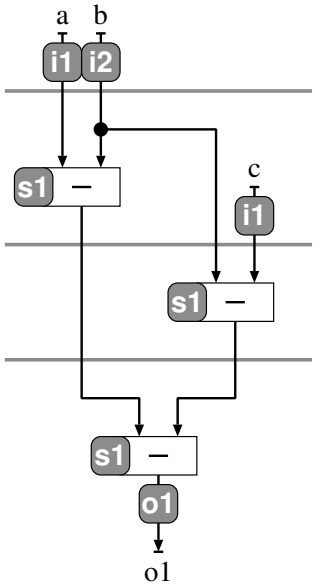
1. The optimization goal is to *minimize the total number of multiplexers and chip-enables*. More precisely, minimize the sum of the number of flip-flops that need a chip enable, and the number of 2:1 multiplexers for both registers and the subtracter. You may combine 2:1 multiplexers to create larger multiplexers. (*e.g.*, two 2:1 muxes can be combined to create a 3:1 multiplexer.)
2. You shall *not* change the allocations that have been done.

Q2a (9 Marks) Register Allocation

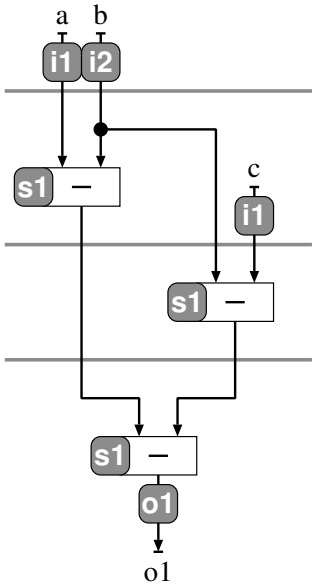
Allocate registers for the dataflow diagram.



Extra copy for scratch work



Extra copy for scratch work



Q2b (9 Marks) Control Table

Draw the control table for the dataflow diagram.

NOTES:

1. Label each row and column of the table clearly.
2. The system shall support an *unpredictable number of bubbles* between valid parcels. If you do not know how to support an unpredictable number of bubbles with this system, you may earn *part marks* by writing a $\surd$ in the box at at the end of this note and answering the question for parcels that *arrive as soon as possible (ASAP)*.

   I am answering for *ASAP parcels*, not an unpredictable number of bubbles. ☐
3. Do *not* do don't-care instantiation.

Q2c (2 Marks) Analysis

For each register that you used and the subtracter, give the number of 2:1 multiplexers that it needs.

For each register that you used, use a $\surd$ in the box for "Yes" or "No" to answer whether it needs a chip-enable signal.

	r1		r2		r3		r4		s1
num 2:1 muxes									
needs chip-enable?	Yes	No	Yes	No	Yes	No	Yes	No	

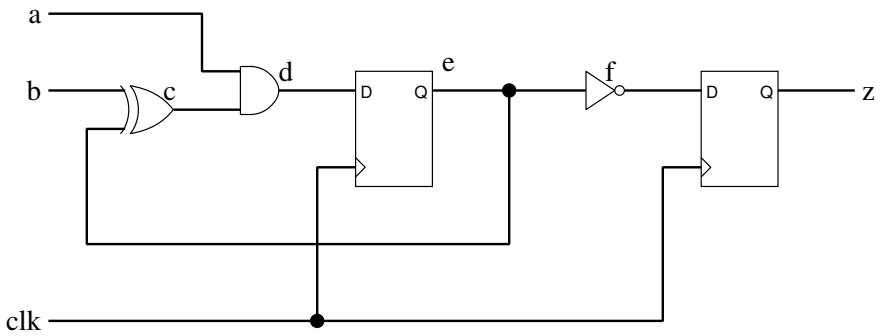
Q3 (20 Marks) The Sleet, the Snow, and the Slush

(*estimated time: 15 minutes*)

Each of the code fragments in Q3a–Q3c is intended to implement the schematic below, such that the behaviour of `z` with respect to the input signals `a`, `b`, and `clk` is the same in both the VHDL code and the schematic. The behaviour of internal signals does not matter.

NOTES:

1. For each of the code fragments Q3a–Q3c:
   (a) Answer whether the code is *legal*
   (b) If the code is *illegal*: explain why, and proceed to the next code fragment.
   (c) Answer whether the output `z` is *correct* with respect to the schematic.
   (d) If the code is *incorrect*: explain why and continue by answering whether the code is synthesizable.
   (e) Answer whether the code is *synthesizable*.
   (f) If the code is *unsynthesizable*: explain why, and proceed to the next code fragment.
   (g) List the signals in the code that are flip-flops.
   (h) Answer whether the code adheres to good coding practices, according to the guidelines for ECE 327.
   (i) If the code does *not follow good coding practices*: explain why.
2. All signals are `std_logic`.
3. When listing the flip-flops, list the signals in *alphabetical order*.



Specification schematic

Q3a

```
process begin
  wait until rising_edge(clk);
  c <= a;
end process;

process (c, d ) begin
  if d = '0' then
    e <= '0';
  else
    e <= c;
  end if;
end process;

d <= b xor e;

process begin
  wait until rising_edge(clk);
  z <= not e;
end process;
```

	Yes	No
Legal	☐	☐
Correct	☐	☐
Synthesizable	☐	☐
Signals that are flip-flops		
Good Practice	☐	☐

Explanation if illegal or incorrect:

Explanation if unsynthesizable or bad practice:

Q3b

```
process begin
  wait until rising_edge(clk);
  c <= a;
  d <= b;
end process;

e <= d xor g;
f <= c and e;

process begin
  wait until rising_edge(clk);
  g <= f;
end process;

z <= not g;
```

	Yes	No
Legal	☐	☐
Correct	☐	☐
Synthesizable	☐	☐
Signals that are flip-flops		
Good Practice	☐	☐

Explanation if illegal or incorrect:

Explanation if unsynthesizable or bad practice:

Q3c

```
process (clk) begin
  if rising_edge(clk) then
    if a = '0' then
      e <= '0';
    else
      e <= a xor e;
    end if;
  end if;
  z <= not e;
end process;
```

	Yes	No
Legal	☐	☐
Correct	☐	☐
Synthesizable	☐	☐
Signals that are flip-flops		
Good Practice	☐	☐

Explanation if illegal or incorrect:

Explanation if unsynthesizable or bad practice:

Q4 (20 Marks) State Machines

(*estimated time: 15 minutes*)

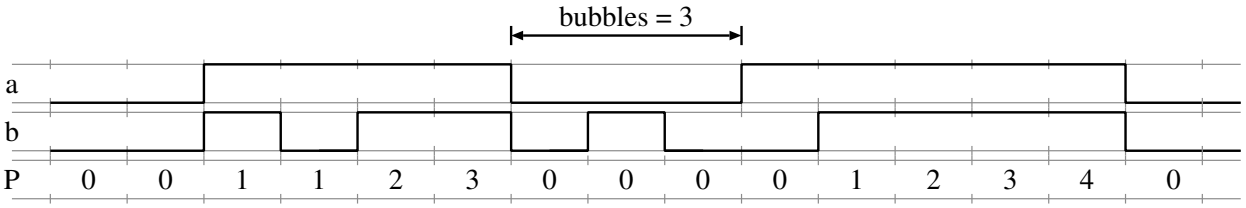
Each of the state machines in Q4a–Q4c is intended to count the number of times that `b='1'` while `a='1'`. The behaviour is described precisely in the table below, which defines the action to perform on `P` in terms of conditions on the inputs `a` and `b`:

Input conditions		
`a`	`b`	**Action**
'0'	–	set `P` to 0
'1'	'0'	no change to `P`
'1'	'1'	increment `P`

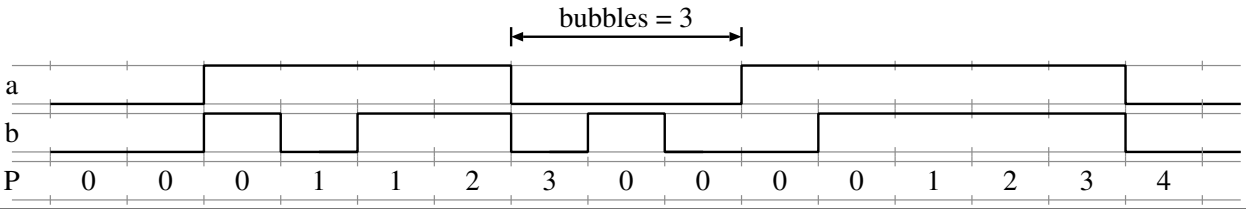
NOTES:

1. The *latency* between the inputs `a` and `b` and the output `P` may be *zero or more* clock cycles. The waveforms below illustrate the behaviour with a latency of 0 and 1.
2. When `a='0'`, the state machine may require multiple clock cycles of `a='0'` before the next clock cycle in which `a='1'`. These clock cycles of `a='0'` are called *bubbles*.
3. For each of the state machines Q4a–Q4c, answer whether the state machine is *correct* with respect to the intended behaviour.
4. If the state machine is *correct*, answer what is the *minimum number of bubbles* that the state machine requires in order to work correctly.
5. 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.
6. The input `a` is guaranteed to be `'0'` for first four clock cycles.

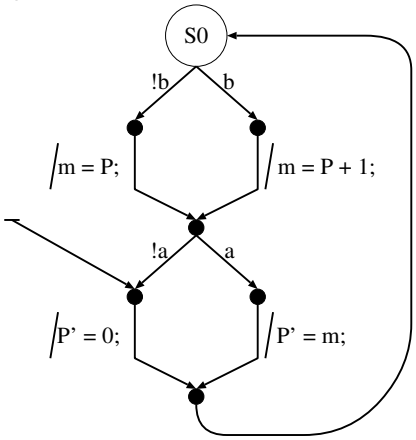
Latency = 0: increment P in same clock cycle as b='1':



Latency = 1: increment P in next clock cycle after b='1':



Q4a



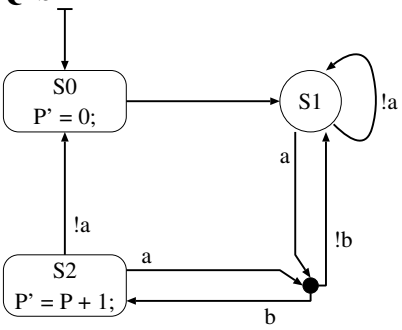
Yes No

Machine is correct ☐ ☐

If correct: number of bubbles ☐

If incorrect, explanation:

Q4b



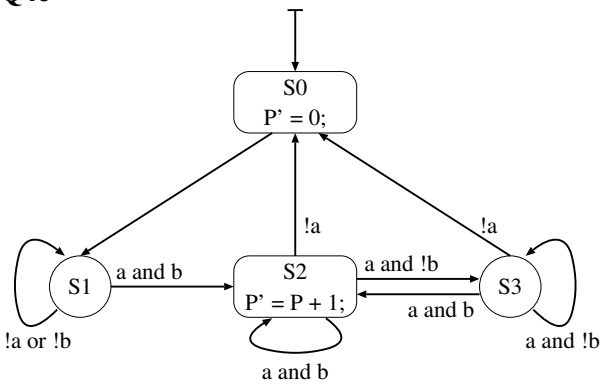
Yes No

Machine is correct ☐ ☐

If correct: number of bubbles ☐

If incorrect, explanation:

Q4c



Yes No

Machine is correct ☐ ☐

If correct: number of bubbles ☐

If incorrect, explanation:

Q5 (20 Marks) Dataflow Diagram with Memory

(*estimated time: 15 minutes*)

Draw a dataflow diagram that implements the specification:

```
M[a] = b;
M[c] = d;
z    = M[c] + M[c+1];
```

NOTES:

1. It is guaranteed that `a` $\neq$ `c`.
2. Inputs shall be *registered*
3. Outputs may be *either combinational or registered*
4. Memory has registered inputs and combinational outputs (same as in class)
5. The memory shall be *dual-ported*. If you do not know how to answer the question with dual-ported memory, you may earn *part marks* by writing a $\surd$ in the box at at the end of this note and answering the question for *single-ported memory*.

   I am answering for *single-ported memory*, not dual-ported memory. ☐
6. Optimization goals in order of decreasing importance:
   (a) minimize *latency*
   (b) minimize *area*
      i. input ports
      ii. registers (*excluding* memory)
      iii. adders
      iv. output ports
   (c) minimize *clock period*
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 final values on `z` and `M` are correct.

Q5a (15 Marks) Dataflow Diagram

Q5b (2 Marks) Bubbles

What is the minimum number of bubbles between valid parcels for your dataflow diagram? ☐

Q5c (3 Marks) Parcel Schedule

Should this system require that parcels arrive as soon as posible (ASAP parcels) or should the system support an unpredictable number of bubbles between parcels? **For full marks, you must justify your answer.**

	ASAP	**Unpredictable num of bubbles**
Parcel schedule	☐	☐

Justification: