First Name _____ **Last Name** _____ **First letter of last name** ☐

UW Userid _____

ECE 327 Midterm

2014t2 (Spring)

Instructions and General Information

- 100 marks total
- Time limit: 1 hour and 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 part marks, you must show the formulas you use 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		10	7	3
Q2	The Yellow, the Red, and the Gooaaallllll!!!!		20	10	4
Q3	Area Analysis		15	10	7
Q4	Function Table and Encoding		15	15	8
Q5	State Machine		20	15	11
Q6	Design with Memory		20	15	13
Totals			100	74	

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 (10 Marks) VHDL Semantics

(*estimated time: 7 minutes*)

Is it possible for a simulation round not to contain any delta cycles? **Justify your answer in terms of VHDL simulation semantics.**

Q2 (20 Marks) The Yellow, the Red, and the Gooaaallll!!!!

(*estimated time: 10 minutes*)

For each of the code fragments Q2a–Q2d:

1. Answer whether the code is *legal*
2. If the code is *illegal*: explain why, and proceed to the next code fragment.
3. Answer whether the code is *synthesizable*.
4. If the code is *unsynthesizable*: explain why, and proceed to the next code fragment.
5. Answer whether the code adheres to good coding practices, according to the guidelines for ECE 327.
6. If the code does *not follow good coding practices*: explain why.
7. If the code does *follow good practices*: draw the circuit that would most likely result from synthesizing the code.

NOTES:

1. If the VHDL code includes an implicit state machine: draw the gates, wires, and flops for the datapath. All of the arithmetic and logical operators in the VHDL code (*e.g.*, "+", "−", "<", and "`xor`") are considered part of the datapath.
2. You may draw the control portion of the circuit as a cloud or black-box that drives the appropriate signals in the datapath.
3. The signal declarations are:

```
clk     : std_logic;
a, b    : unsigned( 7 downto 0 );
m, n, p : std_logic_vector( 0 to 3 );
```

Code fragments begin on next page

Q2a

```
for_i : for i in 0 to 3 generate
  if_yes : if i = 0 generate
    m(i) <= p(3);
  end generate;
  if_no : if i /= 0 generate
    m(i) <= p(i-1);
  end generate;
  p(i) <= m(i) xor n(i);
end generate;
```

	Yes	No
Legal	☐	☐
Synthesizable	☐	☐
Good Practice	☐	☐

Explanation or drawing:

Q2b

```
process begin
  a <= (others => '0');
  wait until rising_edge(clk);
  loop
    a <= a + 1;
    wait until rising_edge(clk);
  end loop;
end process;
```

	Yes	No
Legal	☐	☐
Synthesizable	☐	☐
Good Practice	☐	☐

Explanation or drawing:

Q2c

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

	Yes	No
Legal	☐	☐
Synthesizable	☐	☐
Good Practice	☐	☐

Explanation or drawing:

Q2d

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

	Yes	No
Legal	☐	☐
Synthesizable	☐	☐
Good Practice	☐	☐

Explanation or drawing:

Q3 (15 Marks) Area Analysis

(*estimated time: 10 minutes*)

Calculate the minimum number of FPGA cells needed to implement the VHDL code below.

NOTES:

1. The signals `ab_sel` and `cd_sel` are `std_logic`.
2. The signals `a`, `b`, `c`, `d`, `e`, `k`, `m`, `n`, `p`, and `z` are 12-bit `unsigned`.
3. Optimizations are allowed, so long as the externally visible input-to-output behaviour of the system does not change.
4. For full marks, you must justify your answer with a drawing and/or text.

```
k <= a when ab_sel = '1' else b;
m <= c when cd_sel = '1' else d;
process (clk) begin
  if rising_edge(clk) then
    n <= k + m;
    p <= e;
    z <= n + p;
  end if;
end process;
```

Number of FPGA cells: ____

Justification:

Q4 (15 Marks) Function Table and Encoding

(*estimated time: 15 minutes*)

This question will examine a function table and encoding for the pseudocode specifications of `y` and `z` given below.

NOTES:

1. Inputs:
   - The signal `a` is a `std_logic_vector` that is a one-hot encoding of a size; where the size is `small`, `medium`, or `large`.
   - The signal `b` is a `std_logic`.
2. Outputs:
   - The signal `y` is a color, which is one of `red`, `blu`, or `grn`
   - The signal `z` is a 8-bit `unsigned`.
3. Any condition not defined by the specifications below is a don't care.

```
if b then
  y = blu;
elsif a == large then
  y = grn;
else
  y = red;
```

```
if a == small or a == medium then {
  if b then
    z = 3;
  else
    z = 5;
}  # there intentionally is not an else clause
```

Q4a (3 Marks) One-Hot Encoding

Define the encoding for `a`.

NOTES:

1. The table shows 5 bits for `a`, if you do not need all 5 bits, draw an $\times$ through the label of any bits that you do not need (*e.g.*, ~~`a(2)`~~).

	a(4)	a(3)	a(2)	a(1)	a(0)
small					
medium					
large					

Q4b (6 Marks) Function Table

Using your encoding for `a`, draw *one* function table that defines the behaviour of *both* `y` and `z`.

NOTES:

1. Requirement: Each output value shall appear in exactly one cell.

Q4c (6 Marks) Code

Using your encoding for `a`, write if-then-else statements, in either VHDL or pseudocode, for `y` and `z`.

NOTES:

1. Optimization goal: Minimize the total cost of the conditions:
   - Each if-then-else statement has a cost of 1
   - Each AND, OR, and NOT has a cost of 1
   - Each *n*-bit equality test (=) has a cost of *n*
2. If you use VHDL, you may pretend that if-then-else conditions may be `std_logic`.
   That is, you *may* write `if a(0) then ...` and do *not* need to write `if a(0)='1' then ...`
3. You may choose either to combine the code for `y` and `z`, or to use separate if-then-else statements for `y` and `z`.

Q5 (20 Marks) State Machine

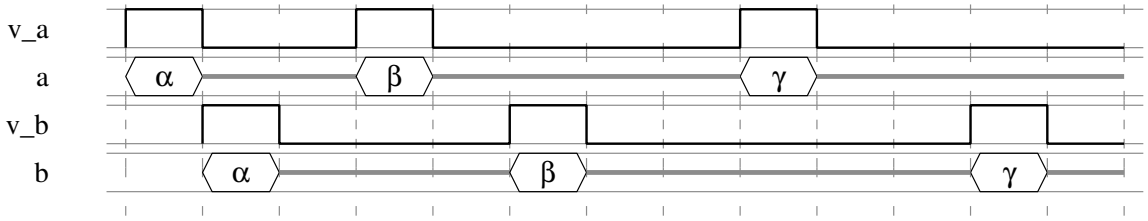
(*estimated time: 15 minutes*)

This question examines a state machine that implements the equation `z = a + b`.

NOTES:

1. Inputs:
   `a, b : unsigned( 7 downto 0 )`
   `a_v, b_v : std_logic`
   Outputs:
   `z : unsigned( 7 downto 0 )`
   `z_v : std_logic`
2. The signals `a` and `b` are both in the same parcel.
3. There is an unpredictable number of clock cycles between when `a` arrives and when `b` arrives, but `b` arrives *at least* one clock cycle later than `a`.
4. The signal `a_v = '1'` when `a` has valid data. The signal `b_v = '1'` when `b` has valid data.
5. The state machine shall assign `z_v = '1'` for exactly one clock cycle when `z` is valid.
6. You may choose the minimum number of bubbles between when `b` arrives and when the next value of `a` arrives. The environment may insert an unpredictable number of extra bubbles beyond the minimum that you specify.
7. Inputs and outputs may be either registered or combinational.

Example waveform for inputs:



Workspace for question is on next page

Q5a (15 Marks) State Machine Design

Draw a state machine that implements the specification.

Q5b (5 Marks) Bubbles and Throughput

What is the minimum number of bubbles that your state machine requires between `b` and the next `a`?

What is the maximum throughput of your state machine?

Q6 (20 Marks) Design with Memory

(*estimated time: 15 minutes*)

This question examines the implementation of the pseudocode specification:

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

NOTES:

1. Inputs shall be *registered*
2. Outputs may be *either combinational or registered*
3. The system shall support an *indeterminate number of bubbles*
4. Memory has registered inputs and combinational outputs (same as in class)
5. The memory may be *either dual-ported or single-ported.*
6. Optimization goals in order of decreasing importance:
   (a) minimize *latency* to `z`
   (b) minimize *clock period*
   (c) minimize *area*
      i. input ports
      ii. adders and subtracters
      iii. registers (*excluding* memory)
      iv. output ports
      v. use single-ported memory instead of dual-ported memory
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, as long as the final values of `z` and `M` are correct.
9. You do *not* need to do allocation.

Work space is on the next page

Q6a (15 Marks) Dataflow Diagram

Draw a dataflow diagram for the system.

Q6b (5 Marks) Memory Ports

How many ports does your memory have: ☐

Briefly justify that your choice of number of memory ports produced the most optimal design.