First Name

Last Name

First let
of last na

UW Userid

ECE 327 Midterm

2013t2 (Spring)

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	Simulation Semantics		10	7	3
Q2	The Mud, the Rain, and the Pollen		25	20	4
Q3	FPGA Cells		23	15	7
Q4	State Machines		22	15	9
Q5	Datapath Design		20	15	11
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) Simulation Semantics

(*estimated time: 7 minutes*)

Your manager has asked you to evaluate a new register-transfer-level simulator for VHDL named "Unifortl: the uniform RTL simulator". The distinguishing feature of Unifortl is that all processes are simulated using the same rules (*i.e.*, all categories of processes are treated uniformly).

The simulation rules are, for each moment in time:

1. Run the non-combinational processes in *any order*, processes read *old values* of signals (with the exception that clocked processes read the current value of the clock signal).
2. Run the combinational processes in *any order*, processes read *old values* of signals.

Will Unifortl simulate *all* VHDL programs correctly, *some* VHDL programs correctly, or *no* VHDL programs correctly?

NOTES:

1. Unifortl is said "not to simulate a VHDL program correctly" if either Unifortl is unable to simulate the program, or Unifortl simulates the program but produces incorrect results.
2. If Unifortl is able to simulate some VHDL programs correctly, then describe the set of programs that *are simulated correctly*.

Unifortl is able to simulate correctly:

☐ all VHDL programs

☐ some VHDL programs Description of set of programs that are simulated correctly:

_____

_____

☐ no VHDL programs

Justification:

_____

_____

_____

_____

_____

Q2 (25 Marks) The Mud, the Rain, and the Pollen

(*estimated time: 20 minutes*)

NOTES:

1. For each of the code fragments Q2a–Q2d:
   (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 code is *synthesizable*.
   (d) If the code is *unsynthesizable*: explain why, and proceed to the next code fragment.
   (e) Answer whether the code adheres to good coding practices, according to the guidelines for ECE 327.
   (f) If the code does *not follow good coding practices*: explain why.
2. For Q2a–Q2b, if the code is synthesizable: draw the gate-level circuit that would most likely result from synthesizing the code. 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.
3. For Q2c–Q2d, if the code is legal: compute the value of `z` at 30 ns and fill in the value on the waveform diagram.
4. The signals `clk` and `a` are `std_logic`.
5. The signals `b`, `c`, `d`, `e`, and `z` are `unsigned( 7 downto 0 )`.
6. The waveforms for `clk`, `a`, `b`, and `c` are the same for each of Q2a–Q2d. The VHDL code for these signals is:

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

```
process begin
  a <= '0';
  wait for 5 ns;
  a <= '1';
  wait for 20 ns;
  a <= '0';
  wait for 15 ns;
end process;
```

```
process begin
  b <= to_unsigned( 11, 8 );
  c <= to_unsigned( 21, 8 );
  loop
    wait for 5 ns;
    b <= b + 1;
    c <= c + 1;
  end loop;
end process;
```

Q2a

```
a0 : if a =  '0' generate
  z <= b;
end generate;
a1 : if a /= '0' generate
  z <= c;
end generate;
```

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

Explanation if illegal, unsynthesizable or bad practice:

Drawing of hardware, if synthesizable:

Q2b

```
process begin
  wait until rising_edge(clk);
  if a = '0' then
    z <= b;
  else
    wait until rising_edge(clk);
    z <= c;
  end if;
end process;
```

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

Explanation if illegal, unsynthesizable or bad practice:

Drawing of hardware, if synthesizable:

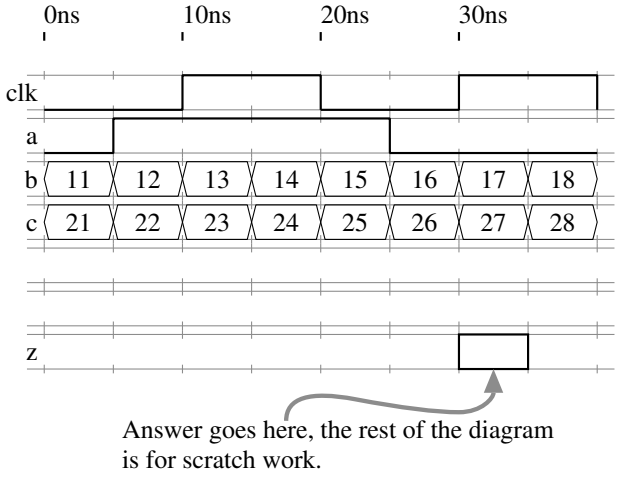
Q2c

```
process begin
  d <= b;
  e <= c;
  wait until rising_edge(clk);
  if a = '0' then
    z <= d;
  else
    z <= e;
  end if;
end process;
```

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

Explanation if illegal, unsynthesizable or bad practice:

Value of z at 30 ns, if legal:



Q2d

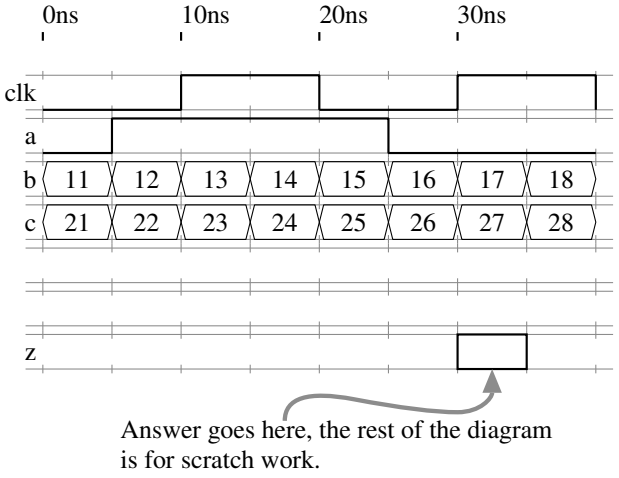
```
process begin
  wait until rising_edge(clk);
  d <= b;
  e <= c;
end process;

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

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

Explanation if illegal, unsynthesizable or bad practice:

Value of z at 30 ns, if legal:



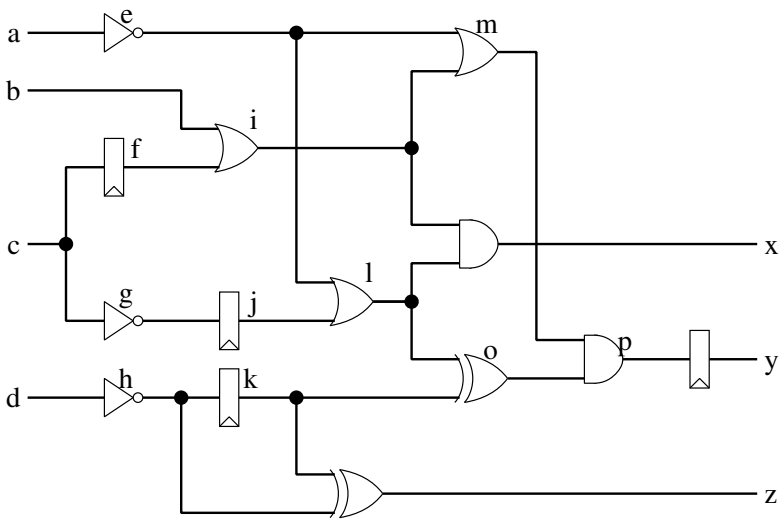
Q3 (23 Marks) FPGA Cells

(*estimated time: 15 minutes*)

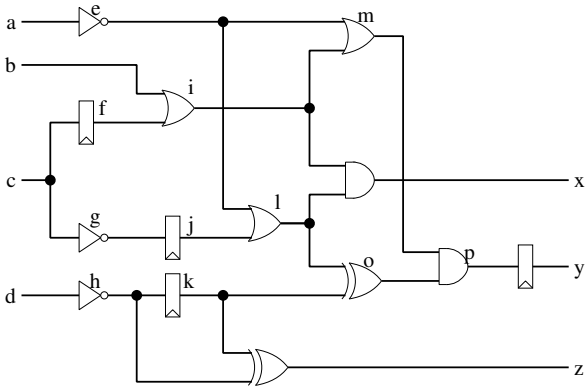
Design an FPGA implementation of the gate-level circuit shown below that uses the minimum number of FPGA cells. Use the FPGA cells on the following page to answer the question.

NOTES:

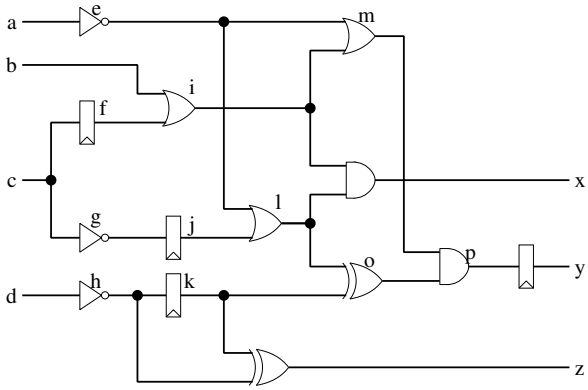
1. The primary inputs of the circuit are: `a`, `b`, `c`, and `d`.
2. The primary outputs of the circuit are: `x`, `y`, and `z`.
3. Do not perform any logic optimizations.
4. For each FPGA cell that you use:
   - Label the input and output ports of the cell using the signal names from the gate-level circuit for ports that you use and **NC** (for no-connect) for ports that you do not use.
   - Show the configuration for the internal multiplexer.

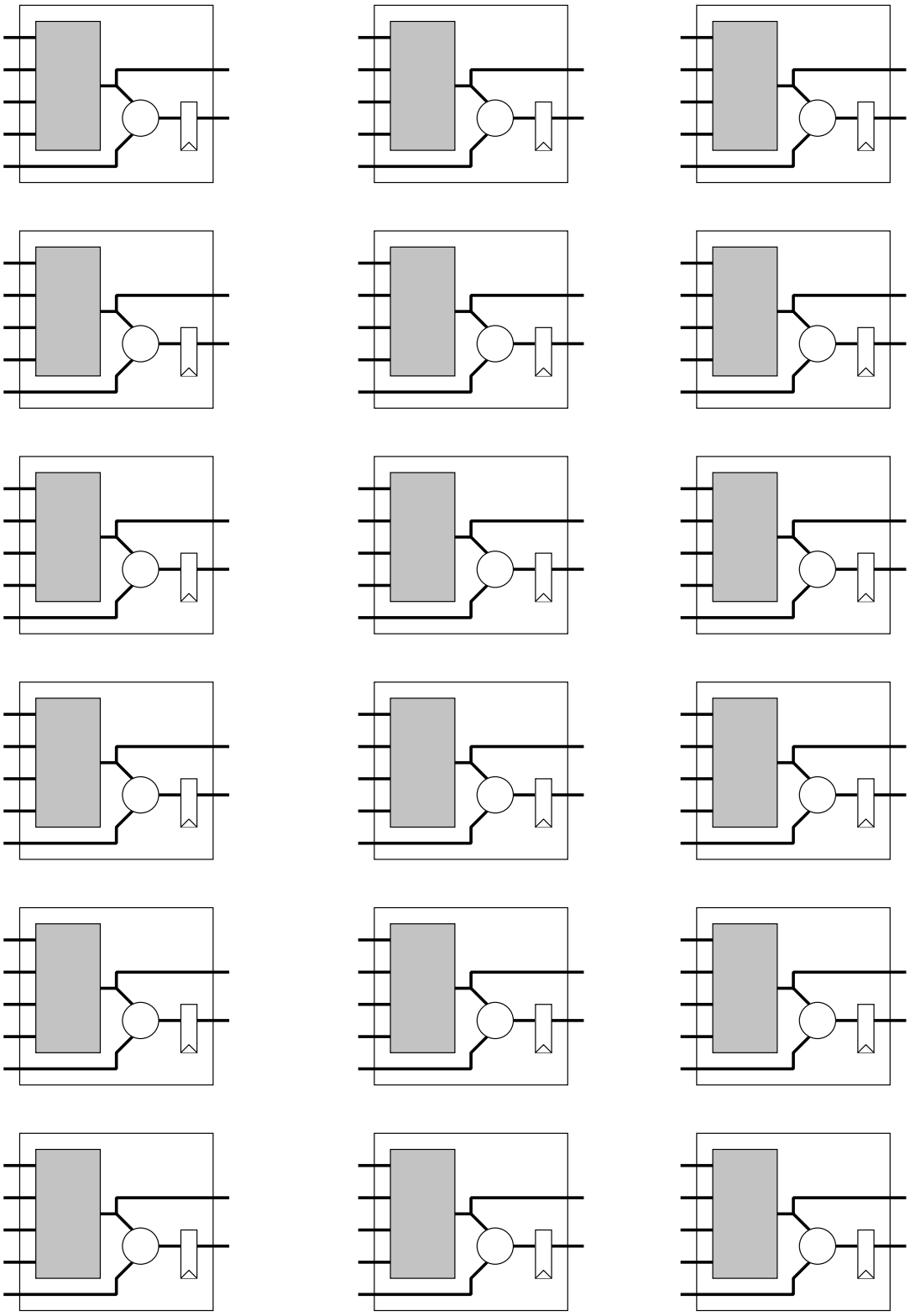


Extra copy for scratch work



Extra copy for scratch work





Q4 (22 Marks) State Machines

(*estimated time: 15 minutes*)

You are part of the design team for "Woggles", a combined smart-watch and ski-goggles. A senior manager is coming for a design review in the afternoon. Your project leader has to go out to lunch with the manager, and has left you to design the last state machine in the system before the design review.

The functional requirements of the state machine are given below, where "`a`" means that `a` is true (in VHDL: equal to `'1'`) and "`!a`" means that `a` is false (in VHDL: not equal to `'1'`):

Functional requirements:

1. The inputs are: `a`, `b`, `j`, `k`, `m`, and `n`.
2. The outputs are: `y` and `z`.
3. The initial state is `S0`.
4. If (the current state is `S0`) and `a`, then in the current clock cycle assign `j` to `y`.
5. If (the current state is `S0`) and `!a`, then in the current clock cycle assign `k` to `y`.
6. If (the current state is `S0`) and `b`, then in the next clock cycle assign `m` to `z`.
7. If (the current state is `S0`) and `!b`, then in the next clock cycle assign `n` to `z`.
8. If (the current state is `S0`) and `b`, then the next state shall be `S1`.
9. If (the current state is `S0`) and `!b`, then the next state shall be `S2`.
10. If the current state is `S1`, then the next state shall be `S2`.
11. If the current state is `S2`, then the next state shall be `S0`.

NOTES:

1. You may choose `y` and `z` to be either registered or combinational. Both may be combinational, both registered, or one combinational and one registered. One of the optimization goals below is to minimize the number of registers.
2. Your design shall satisfy the functional requirements.
3. Your design shall use the state-machine notation and semantics from ECE-327.
4. Your goals, in order of decreasing importance are
   (a) *minimum* number of clocked (*i.e.*, real) states
   (b) *minimum* number of edges
   (c) *minimum* number of transient states
   (d) *minimum* number of registers

Q5 (20 Marks) Datapath Design

(*estimated time: 15 minutes*)

Draw a dataflow diagram that implements the specification:

```
P = P - a + b + c;
Q = Q + d;
z = Q + a - c;
```

NOTES:

1. Inputs shall be *registered*
2. Outputs may be *either registered or combinational*
3. Optimization goals in order of decreasing importance:
   (a) Minimize *adders*
   (b) Minimize *subtracters*
   (c) Minimize *clock period*
   (d) Minimize *latency to* `z` (latency from first input to `z`)
   (e) Minimize *registers*
   (f) Minimize *inputs*
4. Input values may be read in any clock cycle, but each input value shall be read exactly once.
5. Optimizations to the pseudocode are allowed, so long as the final value on `P`, `Q`, and `z` are correct.

Q5a (15 Marks) Dataflow Diagram

Q5b (5 Marks) Analysis

Latency to `z`	
Minimum clock period	
Number of inputs	
Number of registers	
Number of adders	
Number of subtracters	
Best choice for type of state encoding	

below is the buggy version of the question, which was on the original midterm

```
P = P - a + b + c;
Q = Q + d;
z = P + a - c;
```

NOTES:

1. Inputs shall be *registered*
2. Outputs may be *either registered or combinational*
3. Optimization goals in order of decreasing importance:
   (a) Minimize *adders*
   (b) Minimize *subtracters*
   (c) Minimize *clock period*
   (d) Minimize *latency to* `z` (latency from first input to `z`)
   (e) Minimize *registers*
   (f) Minimize *inputs*
4. Input values may be read in any clock cycle, but each input value shall be read exactly once.
5. Optimizations to the pseudocode are allowed, so long as the final value on `P`, `Q`, and `z` are correct.

Q5c (15 Marks) Dataflow Diagram

Q5d (5 Marks) Analysis

Latency to z	
Minimum clock period	
Number of inputs	
Number of registers	
Number of adders	
Number of subtracters	
Best choice for type of state encoding	