ECE 464 / ECE 520
Final 2011

Each question is worth 2 points. The correct answer earns 2 points, the incorrect 0. You have up to three hours to take this test. Answer all of the questions. Off-campus students, please return this test completed as specified in the syllabus.

The exam is open book, open notes. No computers, PDAs, or cell-phones are allowed. Please do NOT have cell phones on your desk.

Note that sometimes the answer is INTENTIONALLY E.

Clearly indicate your answers in the matrix below, except for those questions that are answered on the page. For the multiple choice questions if you feel further explanation is needed, please do so next to the actual question.

Name: ____________________

Student Number: ________________

<table>
<thead>
<tr>
<th>Question</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Question 1
What is wrong with the following code fragment?

```verilog
always@(*)
  begin
    D = A & C;
    if (E) D = D | F;
  end
```

A. Unintentional latches are being built.
B. A timing loop (or arc) is present.
C. D should not be assigned like this. Unbuildable “wired-or” logic is implied.
D. E is missing from the second sensitivity list.
E. Nothing is wrong with this code fragment.

Question 2
What is wrong with the following code fragment?

```verilog
always@(*)
  if (A) A = E ^ D;
  else A = ~A;
```

A. Unintentional latches are being built.
B. A should not be assigned like this. Unbuildable “wired-or” logic is implied.
C. There is combinational logic feedback, i.e. a “timing loop” or “arc”.
D. The sensitivity list is incomplete.
E. Nothing is wrong with this code fragment.

Question 3
What logic function does the following code fragment describe? Sketch the logic.

```verilog
wire [3:0] A;
wire [7:0] B;
wire [2:0] C;
wire [1:0] D;
assign A = {3'b0, B[C]} << D;
```
Question 4
Is “===” (3 “=” signs) synthesizable?
A. No, as ‘z’ and ‘x’ can’t appear in the actual logic.
B. Yes, as ‘z’ or ‘x’ are optimized out.
C. No, as neither ‘==’ nor ‘===’ are synthesizable.
D. Yes, as long as the variables being compared never take on the values of ‘z’ or ‘x’.
E. None of the above are valid

Question 5
What is the main key to achieving fast designs in FPGAs? Why?
A. Try to map the entire design to fit into 18 bit multipliers.
B. Floorplan the design to maximize nearest neighbor data flow.
C. Use tri-state drivers as muxes
D. Use the Verilog templates for the clock to configure it to be fast.
E. None of the above.

Question 6
Are FPGAs commonly used in mobile, battery driven applications? Why?
A. Yes, often the volume in a mobile applications can not support the high NRE cost of an ASIC.
B. Yes, the low per unit cost of FPGAs is appropriate to mobile applications.
C. No, the power consumption of an FPGA is generally too high when compared with the other implementation styles.
D. No, FPGAs do not include RF parts and that is most of the chip area in a typical mobile application.
E. None of the above apply.
**Question 7**
Sketch the logic being described by the following statement group.

```verilog
wire signed [3:0] A, B, C;
wire [1:0] D;
assign C = {A,B} >>> D;
```

**Question 8**
What style of state encoding is used in the following FSM code fragment?

```verilog
always@(posedge clock) state <= next_state;
always@(state or A)
begin
  case (state)
    2'b01 : if (A) next_state = 2'b10;
    2'b10 : next_state = 2'b01;
    default : next_state = 2'b01;
  endcase
  out = state[0];
end
```

A. Minimal sequential  
B. Gray scale  
C. One-hot encoding  
D. No state encoding is specified.  
E. Other.

**Question 9**
The code fragment in Question 8 is likely to implement unintended latches?  
A. True  
B. False
Question 10
Are User Defined Primitives synthesizable?
A. True
B. False

Question 11
Sketch the design described by the following code fragment.

```verilog
task foo (input A, B; output C);
begin
  C = A ^ B;
end

always@(posedge clock) foo (D, E, D);
```

Question 12
If “enable” is high only 1% of the time, and the code below is synthesized, which of the following will have the lowest power consumption.
A. wire [31:0] A, B, D;
   reg [31:0] E;
   always@(posedge clock)
     begin
       if (enable) E <= A * B;
     end

B. wire [31:0] A, B;
   reg [31:0] AA, BB, E;
   always@(posedge clock)
     begin
       AA <= A;
       BB <= B;
       if (enable) E <= AA * BB;
```
C.
wire [31:0] A, B, AA, BB;
reg [31:0] E;
always@(posedge clock)
begin
    E <= AA * BB;
end
assign AA = enable ? A : 0;
assign BB = enable ? B : 0;

D.
wire [31:0] A, B;
reg [31:0] E, AA, BB;
always@(posedge clock)
begin
    if (enable)
        begin
            AA <= A;
            BB <= B;
        end
    else
        begin
            AA <= 0;
            BB <= 0;
        end
    E <= AA * BB;
end

E. These all consume the same power.
Question 13
If the following logic is built exactly as described, which test vector sensitizes a stuck-at-1 fault at g and propagates it to the output h.

wire a, b, c, d, e, f, g, h;

assign f = a ? b : c;
assign g = d | f;
assign h = e & g;

A. \{a, b, c, d, e\} = 5'b10100;
B. \{a, b, c, d, e\} = 5'b10101;
C. \{a, b, c, d, e\} = 5'b11001;
D. \{a, b, c, d, e\} = 5'b11000;
E. None of the above

Question 14
What does the following primitive describe:

primitive PlanetX (A, B, F);
output F;
reg F;
input A, B;
table
0 0 : 0
0 1 : 1
1 1 : 0
1 0 : 1
endtable
endprimitive

A. An XOR gate.
B. An OR gate.
C. An AND gate
D. A D flip-flop
E. None of the above
**Question 15**
Sketch the logic described by the following code fragment:

```verilog
always@(posedge clock)
begin
    C = A | B;
    D = C;
end
```

**Question 16**
If you accidently put glue logic high in the hierarchy what is the best way to fix it (short of rewriting the code from scratch) with the least compromise?

A. Place the glue logic into a module by itself and group it with a neighboring module.
B. Flatten the entire design.
C. Work out the timing for the glue logic and specify it by hand to synthesis without placing it in its own module.
D. Uniquify the glue logic so it can be synthesized.
E. None of the above would fix the problem.

**Question 17**
What does “characterize –constraints” do?

A. Determines the clock frequency for the specified instance.
B. Determines the clock frequency for all modules of the same name as the specified instance.
C. Determines the input and output delays for all instances that match the module whose instance is specified here.
D. Determines the input and output delays for the specified instance.
E. None of the above.
**Question 18**
What best describes the main purpose of assertions?

A. To assert a specific property to Synopsys to ensure correct synthesis.
B. To place self-checking logic into the design.
C. To ensure that a specparam block is properly applied to the design.
D. To check that a specific behavior expected in the logic behaves as expected during simulation.
E. None of the above apply

**Question 19, 20**

Both of these questions apply to the following code implemented exactly as described.

```vhdl
always@(posedge clock)
    begin
        A <= C;
        B <= F;
        E <= H;
    end
assign C = A;
assign F = B | E;
assign G = F & C;
assign H = G | A;
```

Each gate has a delay from its input to output as given as {1 : 2 : 3}, t_{Cp_Q} is {1 : 2 : 3} the clock skew is 1 ns, the flip-flop setup time is 1 ns and the hold time 1 ns. Format above is {min : typical : max}.

**Question 19**
The fastest possible viable clock period is:

A. 3 ns  
B. 5 ns  
C. 14 ns  
D. 17 ns
E. None of the above

**Question 20**
Is there potential for a hold violation in this logic?

A. No, with a margin of 1 ns  
B. No, with no margin  
C. Yes, with a margin of 1 ns  
D. Yes, with a margin of 2 ns  
E. None of the above apply