

ECE 464 / ECE 520
Final 2009

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. Version A.

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.

Name : _____

Student Number : _____

Question	A	B	C	D	E
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					

Question 1

What is wrong with the following code fragment?

```
always@(A or C)
    D = A & C;
always@(D or E or B)
    if (B) A = E ^ D;
    else A = E;
```

- 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?

```
always@(A or D or E or B)
    if (B) 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?

```
wire [7:0] A, B;
wire [2:0] C;
assign A = {B,B} << C;
```

- A. A logical left shift unit.
- B. An arithmetic left shift unit.
- C. A left rotate unit (shift with wrap around)
- D. A right shift unit
- E. None of the above apply.

Question 4

What is the most valid use of a verilog “initial” statement ?

- A. To help implement the reset strategy.
- B. As an interface from the design (where it is used) to the verification environment.
- C. To help implement scan chains used for manufacturing test.
- D. To better optimize resource usage in FPGAs.
- E. None of the above are valid uses

Question 5

Based on trends in mask and design costs for standard cells, vs. FPGA capabilities, do you believe the number of new designs per year executed in standard cells will increase or decrease in the future as compared with a baseline of 2007?

- A. Increase
- B. Decrease

Question 6

Consider the following code fragment (note the use of non-blocking assignment)?

```
always@(posedge clock)
begin
    A <= B;
    B <= C;
end
```

Would the described function change if blocking assignment was used instead?

- A. Yes
- B. No

Question 7

Which of the following statements is **most true** about the differences between designing for FPGAs and standard cells?

- A. Special Verilog templates must be used to specify how the programmable interconnect is programmed in an FPGA.
- B. In an FPGA, more effort must be spent to tailor the design to the specific hardware resources provided.
- C. Assertions are a greater aid to verification in an FPGA than in a standard cell design.
- D. Since power is more of an issue in an FPGA, the designer should consider specifying logic gates in the clock tree, so as to reduce power consumption.

Question 8

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

```
always@(posedge clock) state <= next_state;

always@(state or A)
begin
    case (state)
        2'b00 : if (A) next_state = 2'b00;
                else next_state = 2'b01;
        2'b01 : next_state = 2'b10;
        2'b10 : next_state = 2'b00;
        Default : next_state = 2'b00;
    endcase
    out = &state;
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

In a fragment of System Verilog code, you see a statement group bounded by `property` and `endproperty`. What is it most likely specifying?

- A. An interface.
- B. A constrained random functional verification check.
- C. A timing block.
- D. An assertion.
- E. None of the above.

Question 11

How many scan chains are likely to be implemented in a modern ASIC to make it testable?

- A. None. We rely on constrained random verification instead.
- B. Very few, possibly even just one. I/O pins are a constraint.
- C. Hundreds. We want to shorten the test time.
- D. Tens of Thousands or more, one per flip-flop.
- E. None of these are true.

Question 12

If “enable” is high only 50% 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, F, G;
always@(posedge clock)
  begin
    if (enable) E <= A * B;
  end
```

B.

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

C.

```
wire [31:0] A, B, D, MultIn1, MultIn2, MultOut;
reg [31:0] E, F, G;
always@(posedge clock)
  begin
    AA <= A;
    BB <= B;
    if (enable) E <= A * B;
    else E <= AA * BB;
  end
```

D.

```
wire [31:0] A, B, D;
reg [31:0] E, F, G;
always@(posedge clock)
  begin
    if (enable)
      begin
        AA <= A;
        BB <= B;
      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-0 fault at f 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} = 4'b10100;
- B. {a, b, c, d, e} = 4'b10101;
- C. {a, b, c, d, e} = 4'b11001;
- D. {a, b, c, d, e} = 4'b11000;
- E. None of the above

Question 14

What does the following primitive describe:

```
primitive PlanetX (A, B);
```

```
output F;
```

```
reg F;
```

```
input A, B;
```

```
table
```

```
0 0 : 0
```

```
0 1 : 1
```

```
1 1 : 1
```

```
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

What is the function of the following code fragment:

```
specparam tA0 = 0.30:0.60:1.40;
```

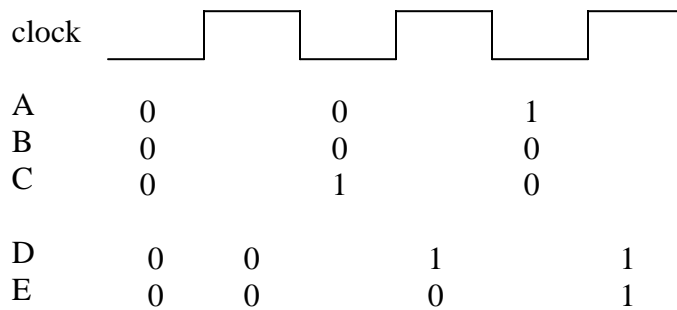
```
specparam tA1 = 0.35:0.65:1.45;
```

```
(A=>out) = (tA0, tA1);
```

- A. Specifying the delay from A to out for rising and falling values of out, respectively, under minimum, typical and maximum delay conditions (in that order).
- B. Specifying the delay from A to out for falling and rising values of out, respectively, under minimum, typical and maximum delay conditions (in that order).
- C. Specifying the delay from A to out for rising and falling values of out, respectively, under typical, minimum, and maximum delay conditions (in that order).
- D. Specifying the delay from A to out for falling and rising values of out, respectively, under typical, minimum, and maximum delay conditions (in that order).
- E. None of the above

Question 16

Which code fragment synthesizes the logic that meets the following timing diagram. Note the answers use **blocking assignment**.



- A.

```
always@(posedge clock)
begin
  E = A ^ B;
  D = E | C;
end
```
- B.

```
always@(posedge clock)
begin
  D = E | C;
  E = A ^ B;
end
```
- C. Both
- D. Neither

Question 17

In a hierarchical design, when you run the first compile command in the script, what exactly is synthesized?

- A. The entire design is synthesized one module at a time.
- B. The design is synthesized one module at a time for the sub-hierarchy specified by `current_design`.

- C. The entire design is flattened and synthesized.
- D. The sub-hierarchy specified by `current_design` is flattened and synthesized.
- E. None of the above

Question 18-20

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

```
always@(posedge clock)
  begin
    A <= C;
    B <= F;
    E <= H;
  end
assign C = A ^ B;
assign F = C | 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 18

The fastest possible viable clock period is:

- A. 3 ns
- B. 5 ns
- C. 10 ns
- D. 17 ns
- E. None of the above

Question 19

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

Question 20

This design used in the last two questions is interfacing to another engineers design. What would you advise her to use as the “input_delay” in her synthesis script. Assume wiring delay is 0.

- A. 1 ns.
- B. 2 ns.
- C. 3 ns
- D. 4 ns
- E. None of the above are correct