

ECE 464 / ECE 520/ DS 510P
Midterm 2003

Each question is worth 1 point. The correct answer earns 1 point, the incorrect 0. You have up to two hours to take this test. Answer all of the questions. Off-campus students, please return this test completed by the date specified on the web page.

The exam is open book, open notes. **90 minutes are permitted.**

Write your name and student number on the bubble sheets in the specified locations. I would like to display final scores against your student number. If you do NOT wish this for your test results, please give your name to the TAs during the exam.

Question 1

Consider the following code fragment

```
always@(A or B)
    if (B) C = A;
    else C = ~A;
always@(posedge clock)
    if (D) C <= F;
```

There is a potential synthesis problem with this code fragment. Which is the problem and a potential fix?

A. Unintentional latches.

```
always@(A or B)
    if (B) C = A;
    else C = ~A;
always@(posedge clock)
    if (D) C <= F;
    else C <= ~F;
```

B. Unintentional wired-or logic.

```
always@(posedge clock)
    if (B) C <= A;
    else if (D) C <= F;
    else C <= ~A;
```

C. Incorrect use of blocking assignment.

```
always@(A or B)
    if (B) C <= A;
    else C <= ~A;
always@(posedge clock)
    if (D) C <= F;
```

D. Incomplete sensitivity list.

```
always@(A or B)
    if (B) C = A;
    else C = ~A;
always@(posedge clock or F)
    if (D) C <= F;
```

E. None of the above

Question 2

Which of the following is the most important impact Moore's law (i.e. integrated circuit scaling) has on the semiconductor industry:

- A. The cost of chips continues to decline exponentially with time. Asymptotically, electronics has zero cost.
- B. That standard cell ASICs will be eventually completely supplanted by FPGAs.
- C. That every new generation of chips can achieve radically increased levels of performance or incorporate functions up-til-now not possible to include.
- D. The cost of fab plants will get so high that no-one will be able to afford to build chips.
- E. None of the above are an important impact.

Question 3

Which of the following statements are **not** true about FPGAs vs. Standard Cell ASICs?

- A. FPGAs are generally designed with cheaper, easier-to-use CAD tools, and required less sophistication of the design team.
- B. An FPGA chip will have a faster clock speed than the equivalent standard cell design.
- C. The per-unit cost of an FPGA will be higher than that of the equivalent ASIC.
- D. FPGA verification does not have to be as thorough before implementation, as is the case of a standard cell ASIC.
- E. All of these statements are true.

Question 4

It is desirable to minimize clock skew in a design because:

- A. Increased clock skew directly results in an increase in clock period.
- B. The synthesis tools assume a certain amount of clock skew exists but can only cope with it up to a limit set by the fabrication foundry.
- C. In many chips, time is important (e.g. a GPS) and thus you need to track the time on-chip accurately.
- D. The synthesis script becomes too hard to write otherwise.
- E. It is not desirable to minimize clock skew.

Question 5

Clock skew is specified to Synopsys by which mechanism?

- A. It is built into the library.
- B. Using set_driving_cell on the clock buffer.
- C. With the create_clock command.
- D. With set_clock_skew command.

E. None of the above.

Questions 6, 7

These questions refer to the following logic diagram. The minimum and maximum delays between each set of successive gates are marked as #(min: typical: max) ns and are marked on the output node of the driving gate. You also need the following:

- $T_{\text{setup}} = \#(1: 2 : 3)$ ns.
- $T_{\text{hold}} = \#(1: 2 : 3)$ ns.
- $T_{\text{skew}} = \#(1: 1 : 1)$ ns for the clock.
- $T_{\text{clock-Q}} = \#(2 : 3 : 6)$ ns.
- $T_{\text{logic}} = \#(1 : 3 : 5)$ ns for each and every logic gate.

Consider the logic equivalent to the following code (implemented exactly as described):

```
reg [2:0] A, C;
wire [2:0] B;

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

assign B = {&{A[1],C[0]^C[1]}, A[0]^A[1], C[0]^C[1]};
```

Question 6

What is the fastest possible clock period?

- A. 4 ns
- B. 15 ns
- C. 20 ns
- D. 21 ns
- E. None of the above are correct.

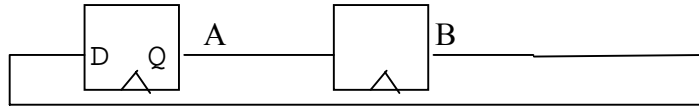
Question 7

Is there a potential hold violation?

- A. Yes, the fastest logic is 2 ns too fast.
- B. Yes, the fastest logic is 1 ns too fast.
- C. No, there is a 1 ns safety margin.
- D. No, there is a 2 ns safety margin.
- E. None of the above are correct.

Question 8

Which code fragment correctly captures the following logic. Notice the use of blocking assignment.



- A.

```
always@(posedge clock)
begin
    A = B;
    B = A;
end
```
- B.

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

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

```
always@(posedge clock)
begin
    A = B = A;
end
```
- E. None of the above

Question 9

Which alternative best describes the behavior of the logic in the following verilog fragment.

```
wire [3:0] A, B;
assign B = {A,A[A[1:0]]};
```

- If A=4'b0111, then
- A. B=5'b01110
 - B. B=5'b01111
 - C. B=4'b1111
 - D. B=4'b1110;
 - E. None of the above

Question 10

Which alternative best describes the behavior of the logic in the following verilog fragment.

```
wire [4:0] A;  
wire [2:0] B;  
assign B = {&A[2:0]; {2{A[4] | A[3]}} };
```

If $A = 5'b10101$, then

- A. $B = 3'b000$;
- B. $B = 3'b011$;
- C. $B = 3'b100$;
- D. $B = 3'b111$;
- E. None of the above.

Question 11

Consider the following code.

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

If it is synthesized with the following constraints

```
Create_clock -period 15 -waveform {0 10} clock  
set_clock_skew -uncertainty 1.0 clock  
  
set_input_delay 4.0 -clock clock all_inputs() - clock
```

then what is the maximum allowed delay for the logic $A+B$ if the setup time for the flip-flops with output C is 1 ns?

- A. 1 ns
- B. 4 ns
- C. 5 ns
- D. 10 ns
- E. None of the above

Question 12

Which alternative best describes the behavior of the logic in the following verilog fragment. Notice the use of blocking assignment.

```
reg [3:0] A, B, C;

always@(posedge clock)
begin
    B = {A[1:0], A[3:2]};
    C = A + B;
end
```

If A =4'b1101, and B=4'b0001 before the positive edge of the clock, then after the positive edge.

- A. B=4'b0011; C=4'b0001;
- B. B=4'b0111; C=4'b1011;
- C. B=4'b0111; C=4'b1110;
- D. B=4'b0111; C=4'b0100;
- E. None of the above.

Question 13

Consider the following code fragment

```
module foobar (clock, D, Q);
input clock, D;
output Q;
always@(posedge clock) Q <= D;
endmodule
```

...

```
module top;

foobar u1 (clock, D, Q);
end
```

Which of the following correctly declares clock, D, and Q, within module “top” based on the information given above.

- A. reg clock, D, Q;
- B. wire clock, D, Q;
- C. tri clock, D, Q;
- D. trireg clock, D, Q;
- E. None of the above.