ECE 426/516 **Implementation of VLSI Systems with HDL**

HW#3 Verilog Basics and Test bench

Due Date: 11:30pm Feb 12th, 2024

Please upload your homework solution to D2L

Question 1 (Initial Block Statement):
(a) Often in certain applications, we need to apply specified set of values. The best way to apply these is to use 'initial" block of statements as shown in the following example:

```
initial
begin
        test_pattern = 1;
        #20 test_pattern = 0;
        #30 test_pattern = 1;
        #40 test_pattern = 1;
        #50 test_pattern = 0;
        #60 test_pattern = 1;
end
```

Draw a timing diagram for the 'test_pattern" signal.

(b) Another way to apply specified set of values is shown in the following example:

```
initial
begin
        test_pattern = 1;
        test_pattern = #20 0;
        test_pattern = #30 1;
        test_pattern = #40 1;
        test_pattern = #50 0;
        test_pattern = #60 1;
end
```

Draw a timing diagram for the "test_pattern" signal in this case.

(c) Yet, another way to apply specified set of values is shown in following example:

```
initial
```

```
begin
        test_pattern <= 1;
        test_pattern <= #20 0;
        test_pattern <= #50 1;
        test_pattern <= # 90 1;
        test_pattern <= #140 0;
        test_pattern <= #200 1;
end
```

Draw a timing diagram for the "test_pattern" signal in this case.

Question 2 (**Initial and Always Block Statement**):
'initial' block of statements processes only once, whereas 'always' block of statements processes in a repeated manner. The 'always' way to apply specified set of values is shown in the following example:

```
        parameter idle = 300;
initial
        begin
                #1000 $stop;
        End

always
        begin
                test_pattern <= 10;
                test_pattern <= #20 20;
                test_pattern <= #50 30;
                test_pattern <= #90 40;
                test_pattern<= #140 50;
                test_pattern<=#200 60;
                # idle;
        End
```

Draw a timing diagram for the "test_pattern" signal in this case.

Question 3 (**Clock Generation**):
(a) Generate a clock signal with different on-off timings, say, TON=10ns and TOFF=20ns. Use parameter for on-off timings and 'always' block to realize the same. Can you get a 50% duty cycle for this clock? If so, how?

(b) Using a 'repeat' loop, a fixed number of clock pules can be generated. Code such a clock. Parameterize the ON, OFF times and the fixed number of clock pules. Draw the waveform for a 50% duty cycle clock for ten cycles.

(c) Many applications demand two or more clocks with phase delays among them. Generate two clock signals with a phase delay of 5ns and different on-off timings, say, TON=10ns and TOFF=20ns. Use parameter for phase delay, on-off timings and ‘always’ block to realize the same.

Question 4: Provide a short answer to the following questions.
(a) Why is it better to use a case statement over a nested if-else statement?

(b) Suppose we assign A and B as shown in the code below. What are the waveforms of A and B? Assume that the value of A is 0, and the value of B is 1 when the code below is executed.

```
#5 A =1;
#5 B =0;
```

(c) Explain the difference in functionality between the “==” and “===” operators.