Verilog Quiz (20 points)
November 7, 2019 Name: _____

No notes, calculators, or other aids are allowed. Read all directions carefully and write your answers in the space provided.

1. (2 points) Why does the binary system consist of 0 and 1? (why not any 2 random numbers like -1 and 1)

   Since the base is 2, one should be able to do modulo 2 arithmetic with the chosen numbers. Any two numbers can be used for modulo 2 arithmetic as long as one of them can be expressed as 0mod2 and the other as 1mod2. 0 and 1 is the simplest choice which satisfies this condition.

2. (2 points) What is an inferred latch and when can it arise? How would you prevent it? (Hint : Conditional statements)

   An inferred latch can arise when not all cases are accounted for in conditional statements (like if else ladders or case). In this case, the hardware latches on to previous states when an unexpected case is encountered. In order to prevent this, one must have valid outputs for each possible case. This is modelled as a multiplexer.

3. State whether the following statements are true or false.
   (a) (1 point) A reg can be updated in multiple always blocks.
   False (Multiple driver issues)
   (b) (1 point) Statements within an always block are executed sequentially.
   True

4. (2 points) always clk = $\sim$ clk;

   Is the above code synthesizable? How will it behave in simulation? Explain your stance.

   The code cannot be synthesized. The signal is constantly being updated without a finite delay and this can't be realized in hardware. It will also hang in simulation because of the absence of a delay.

5. (3 points) Explain what is wrong with the given block of code. Write the corrected code and explain what the output would be.
(The code should be synthesizable)

```
initial begin
a <= '0000';
end

always @(posedge clk or rst)
begin
If (rst)
a <= 4'b`0000';
else
a <= a+1;
end
```

Corrected Code

```
always @(posedge clk or posedge(rst))
begin
If (rst)
a <= 4'b`0000;
else
a <= a+1;
end
```

It is a counter that counts up from 0 to 15 (mod 15 counter).
Main errors:

- Initial blocks are only used in simulation as they are not synthesized in Verilog.
- Sensitivity list cannot be level triggered and edge triggered simultaneously. Should be posedge(clk) or posedge(rst)

6. (9 points) You need to design a special type of 3 bit counter. Initially, all the bits are set to 0 using an asynchronous reset. In every state, the last bit of the previous state is flipped and fed to the first bit. The remaining bits are right shifted by one. (First bit is fed to the second bit and so on)

   (a) (2 points) How would you implement this in hardware. (You do not have to be too technical. Just explain the logic) Also, how many states will the counter count up to before it repeats itself?

   Every clock cycle, use a shift register to shift the bit by one to the right. Use a not gate to flip the last bit of the previous state and feed it to the first bit.The counter will count up to six states 000, 100, 110, 111, 011, 001 before repeating itself.

   (b) (4 points) Write a module in Verilog that implements the above counter.

   module top(clk,rst,state);

   input clk,rst;
   output reg [2:0]state;
   always@(posedge(clk))
   begin
   if (rst)
   state<=3'b000;
   else

   $$state <= \{\sim state[0], state[2:1]\};$$

   end
   endmodule

   (c) (3 points) Draw the waveforms for each bit of the counter along with the clock. Make sure you cover all the states.

   Refer to the six states in part (a). State transition occurs every clock cycle.