University of California at Berkeley
Department of Electrical Engineering and Computer Science

EECS150/EE141/EE241A, Fall 2014

Homework 2: Verilog and Boolean Algebra

Due: Friday Sep 19, 5pm

Please include your name, SID and specify either CS150, EE141 or EE241A at the top of your homework handin. Homeworks must be submitted electronically.

1. The following code describes a 3-bit linear-feedback shift register (LFSR), which generates a repeating pattern of pseudo-random numbers.

```
module lfsr(
    input [2:0] R,
    input Load,
    input Clock,
    output reg [2:0] Q
    );

    always@ (posedge Clock)
        if (Load)
            Q <= R;
        else
            Q <= {Q[1], Q[0]^Q[2], Q[2]};
endmodule
```

   (a) Draw the circuit that corresponds to the code.

   (b) If 001 is loaded into the LFSR initially, what is the sequence of numbers generated? List the binary numbers it generates, start from 001.

   (c) At which cycle does the sequence repeat?

2. The code below is similar to the code in the previous, but blocking assignments are used. The example below should illustrate why blocking assignments for synchronous logic should be avoided if possible.

```
module lfsr(R, Load, Clock, Q) ;
    input [2:0] R;
    input Load, Clock;
    output reg [2:0] Q;
```

```
    always@ (posedge Clock)
        if (Load)
            Q <= R;
        else begin
            Q[0] = Q[2];
            Q[1] = Q[0] ^ Q[2] ;
            Q[2] = Q[1];
        end
endmodule
```

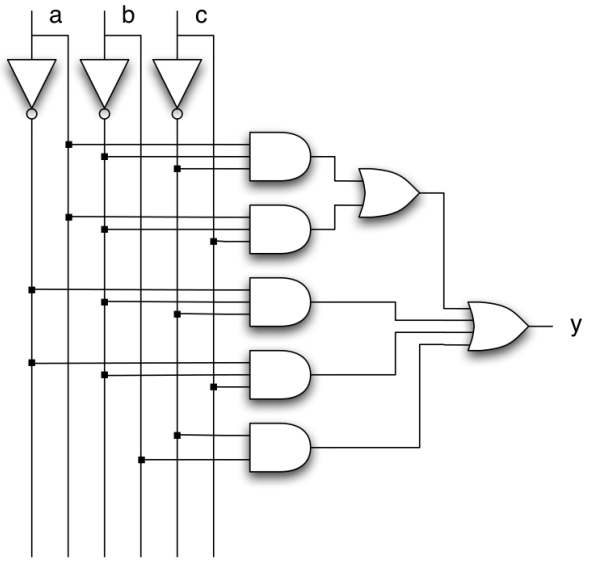
(a) Draw the circuit that corresponds to the code.

(b) If 001 is loaded initially, what is the sequence of numbers generated?

3. Use *behavioral* Verilog to design a 8-bit up/down counter. It should have a `clock` input and a `count` output. It should also have a `reset` input, which resets the counter to zero on the next rising clock edge, and an input control signal `up_down`, which indicates counting up when set to 0, and counting down when set to 1.

4. (a) Write an (unsimplified) boolean expression for the circuit below.

   (b) Using the boolean algebra rules, simplify the circuit as far as possible.



5. A 3-bit majority detector has 3 boolean inputs and 1 boolean output. It outputs logical true if 2 or more of its inputs are true; else it outputs false.

   (a) Draw a truth table for a 3-bit majority detector.

(b) Write the sum-of-products expression for the circuit. Simplify the expression as far as possible, and draw a circuit to implement it using just 2-input gates.

(c) Write the product-of-sums expression for the circuit. You do not need to simplify this.

6. The BCD incrementer was introduced in lecture this week (see Module 3 slides - Boolean logic). Draw the K-maps for this circuit and give the simplified boolean expressions for outputs `w, x, y` and `z`.

7. Gray codes have a useful property in that consecutive numbers differ in only a single bit position. The table below lists a 3-bit Gray code representing the decimal numbers 0 to 7.

Number	Gray Code
0	000
1	001
2	011
3	010
4	110
5	111
6	101
7	100

A 3-bit modulo-8 Gray code counter can be built using a 3-bit register, and a Gray code incrementer connected in feedback between the output and input of the register. The grey-code incrementer has a 3-bit input (call them `a,b,c`) and a 3-bit output (`x,y,z`). The output is simply 1 larger than the input (modulo-8). Note that a modulo-N counter counts from 0 to N-1, and then repeats.

(a) Construct a K-map to represent the Gray code incrementer, and then derive a simplified expression for output bits (x,y,z).

(b) Write a verilog module that implements the complete Gray code counter. It should have a `clk` input and output `cnt[2:0]`. The output should be in Gray code, and should increment every clock cycle. Do not worry about initializing the counter to zero.

8. Write a Verilog module for a 4-bit serial-to-parallel converter. The circuit has the following inputs and outputs:

- `sin`: the input serial data
- `pout[3:0]`: the 4-bit parallel output
- `clk`: the input clock. All state and outputs should only change on rising clock edges.
- `shift_in`: at a rising clock edge, when this input is high, the value of `sin` is valid, and the next bit can be shifted in.

- `load_pout`: when this input is high, it indicates that all 4 serial bits have been shifted in, and that the parallel output should be updated at the next rising edge of the clock.

You can assume that the circuit will be used as follows: on clock cycles 0 to 3, valid serial input data (for bits 0 to 3) will be present at input `sin`, and `shift_in` will be high the entire time. The output during this time is irrelevant. On clock cycle 4, `shift_in` will be low and `load_pout` will be high. Therefore, the parallel output, `pout[3:0]`, should be updated in this clock cycle, with the 4 bits that were received in cycles 0 to 3.

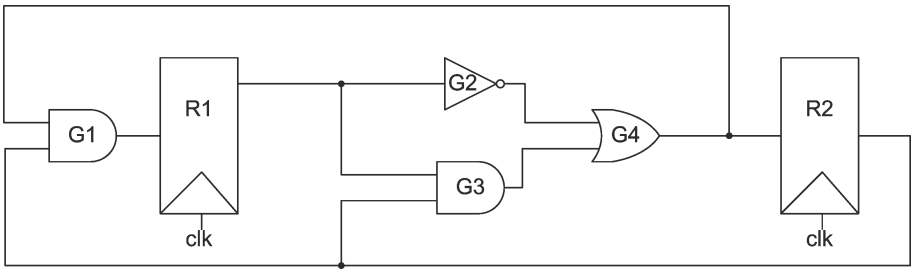
9. The timing parameters for the circuit below are as follows:

Setup time	50 ps
Hold time	45 ps
Clock-to-q delay	40 ps (min), 60 ps (max)
Propagation delay per gate	80 ps (min), 100 ps (max)

(a) Identify the critical path (i.e. the longest timing path) or paths in this circut. You may make use of the register names (R1 and R2) and the gate names (G1, G2, G3, G4) to help you describe the critical path(s).

(b) Use your answer in part (a) to calculate the maximum clock rate at which this circuit can be reliably run.

(c) Can the register hold times ever be violated in this circuit? Why?



10. **For EE241A only:** Prove or disprove the following. A 2-input multiplexor circuit is a universal logic element (in the same sense as a NAND or NOR gate).