EE 209 Homework 2
Logic Function Synthesis

Name: ______________________________________

Due: See Blackboard **Score: ________**

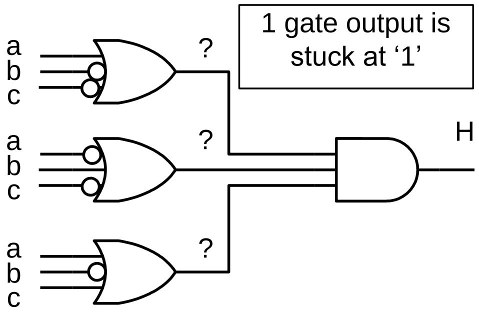
[BB] = Blackboard Submission. For non-BB problems, show work to get full credit.

HW 2 Blackboard Form

1. **[BB]** (12 pts.) When the circuit below was fabricated on a chip, a problem occurred. One of the first level OR gates continually outputs a '1' (a.k.a. a "stuck-at-1" fault) no matter what the inputs to the gate.

   a. Propose and list a minimal number of input combinations that could be used to determine which gate had the fault (i.e. a,b,c = 1,1,0 and a,b,c = 1,1,1)? Describe how you could use these values to determine the gate with the error?

   b. If instead one of the OR gates was stuck at '0' would any set of input combinations be able to determine which gate had the error? Explain?

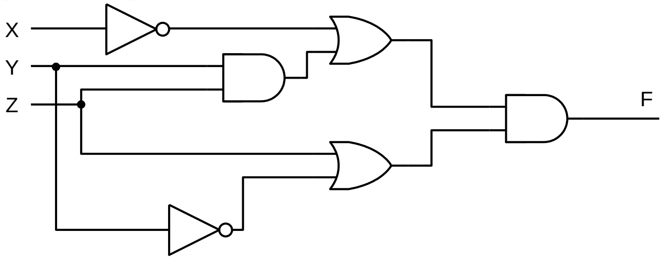


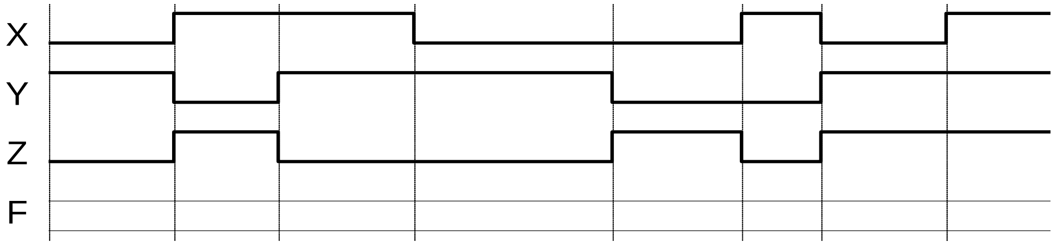
2. **[BB]** (14 pts) Write out the *algebraic* form of both the minterm and maxterm representation for each function (both for each one…meaning convert G to a canonical product as well.)

   $G = \Sigma_{A,B,C,D}(2,3,6,13,15)$

3. **[Practice-Only] (0 pts.)** Given a 4-bit, unsigned input (A[3:0]) design a logic circuit using a sum of minterms approach with a single output, F, which outputs '1' if the input number is a multiple of 3 (i.e. A[3:0] in {0_{10}, 3_{10}, 6_{10}, …}

4. **[BB]** (8 pts.) Complete the waveform for the circuit below.

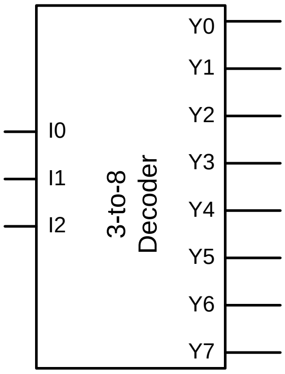




5. **[BB] (30 pts.) Using a Karnaugh map, find both the minimal POS and SOP expression for each of the following functions (both for each).**
   a. $F = \sum_{ABC} (0,2,4,7)$
   b. $G = \prod_{WXYZ} (2,3,7,8,11,12,14,15)$
   c. H(w,x,y,z) such that H = 1 for input combinations divisible by 2 or 3 (w,x,y,z=0000 is divisible by 2 and 3).

6. **[Practice Only] (0 pts.) Draw the specified implementation schematics.**
   a. AND-OR implementation for 3a.
   b. NOR-NOR implementation for 3b.
   c. NAND-NAND implementation for 3c.

7. **[BB] (36 pts.) Design a circuit that takes in a 3-bit signed-magnitude number, A[2:0] ($A_2A_1A_0$), adds two to its decimal equivalent, and produces a 2's complement result, Z[n:0]. (i.e. Z = A + 2.) (Consider the signed magnitude combination representing –0 as an illegal input combination that should never occur.)**
   a. Write out a truth table (consider how many outputs would be necessary for this problem)
   b. Use Karnaugh Maps to produce minimal equations for each output bit
   c. Now implement the same design again use a 3-to-8 decoder and at most 3 OR gates.



I0
I1
I2
3-to-8
Decoder
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7