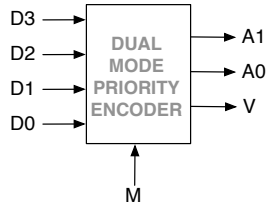
CSEE 3827: Homework 2 — Due: 2/22/10

For each of the following design problems, unless otherwise directed, you may specify your designs in any way other than with truth tables and karnaugh maps. For example you may choose to draw a schematic, write boolean experssions or use a combination of the two – whatever makes the most sense to you. You may use any of the combinational building blocks that we have discussed in class in your designs.

1. In class we have discussed *priority encoders*. The textbook shows one implementation of a 4-bit priority encoder (Inputs: D_3, D_2, D_1, D_0, Outputs: A_1, A_0, V) via k-maps.

   (a) Design an implementation of the same priority encoder without using k-maps.

   (b) Using the 4-bit priority encoder, design a dual-mode priority encoder with the interface below. If the mode bit is not set (M=0), the output indicates the position of the *most significant* input bit that has the value 1. If the mode bit is set (M=1) the output indicates the position of the *least significant* input bit that is equal to 1. In both cases the valid bit of output is true if and only if the input contains a 1.



2. Design a 4-bit comparator. The comparator takes two 4-bit BCD inputs X ($= X_3, X_2, X_1, X_0$) and Y ($= Y_3, Y_2, Y_1, Y_0$), and produces a single output bit, G that is true if and only if $X \geq Y$.

3. Give a boolean expression for the third sum bit of the following binary addition operation:

$$\begin{array}{cccc} & X_2 & X_1 & X_0 \\ + & Y_2 & Y_1 & Y_0 \\ \hline & S_2 & S_1 & S_0 \end{array}$$

   You may assume that the initial carry in is zero. The function should take six input bits, X_0, X_1, X_2 and Y_0, Y_1, Y_2, and produce one output bit, S_2.

4. The diagram below shows a data transmission setup, wherein an 8-bit value, X, is to be transported over a channel that may introduce errors (i.e, probabilistically bits in X may be flipped during transmission). We call the output of the channel, Y, and if there were no errors Y would be equal to X. In order to detect an error[1] we introduce two modules: a *check generator* on the sender side and a *check tester* on the recipient side. The generator inspects the value of X and produces a 4-bit value which indicates in BCD the number of bits in X that have the value 1. This 4-bit check value is then sent along with X. On the recipient side, the check tester inspects the check value and Y, indicating via a single bit of output V which is true if and only if the check value is equal to the number of ones in Y.

   Give implementations of (a) the *check generator* and (b) the *check tester*.

[1] This system will catch some, but not all errors. For example it will not catch cases in which more than one bit is flipped, or situations where bits in both X and the check value are flipped, resulting in $Y \neq X$ but a successful test on the recipient side.