Courses
Main
Students
Staff

Students

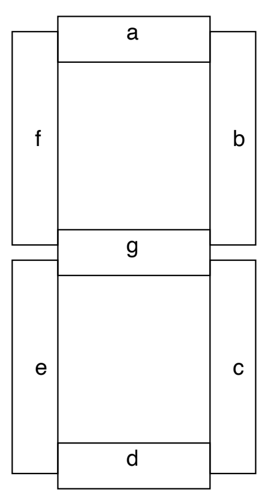
Notices
Give Attendance
Details
Change Password
Enter Name
Assignments
Attendance Log
Section: 1

Single bit ECC with display

You need to be alert to (usually minor) changes that may be made to the assignment statement or to the guidelines after the assignment is first put up. Refresh this frame and re-read the assignment carefully before you make your final submission.

Part 1 (BCD to 7-segment display)

A seven segment display is pictured as follows:



- For the BCD number given as $\langle x_3, x_2, x_1, x_0\rangle$ form the truth tables and the corresponding circuits to light up the LEDs $a..g$ of the 7-segment display to indicate the decimal number properly
- Test that it works by applying appropriate inputs and checking the outputs – test with bit displays in lieu of the LEDs
- Label the terminals to reflect their roles
- Save it as a regular circuit (logic file), reopen and retest
- Save it as a component (cmp file), reopen and retest – only the logic to drive the segments will be effectively encapsulated; when the module is used bit displays will again have to be connected to the outputs

Part 2 (3-bit decoder)

A k-bit decoder has three input lines, an enable line and 2^k output lines indexed 0 to 2^k-1; all output lines take the value 0 in the enable line is 0, otherwise if n represents the binary value of the input bits, output line n takes the value 1 while all other output lines take the value 0

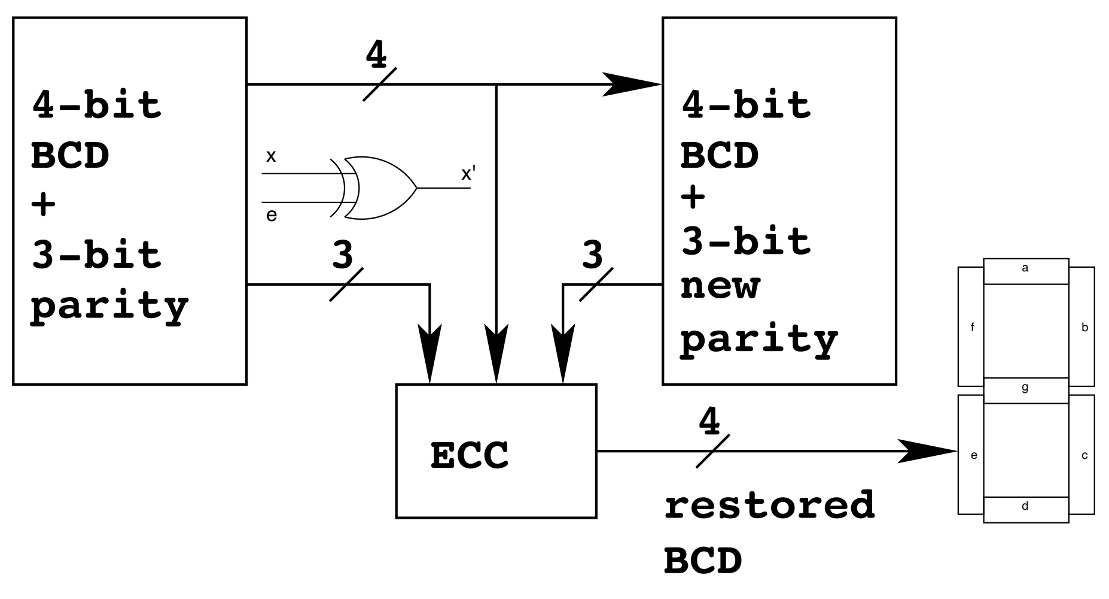
- Design a 3-bit decoder (it can be conveniently designed by connecting a number of 1-bit decoder in a tree structure)
- Test that the decoders work by applying appropriate inputs and checking the outputs
- Label the terminals to reflect their roles
- Save decoders as regular circuits (logic file), reopen and retest
- Save the decoders as components (cmp file), reopen and retest

Part 3 (3-bit ECC parity generator)

- For the BCD number given as $\langle x_3, x_2, x_1, x_0\rangle$ form the truth tables and the corresponding circuits for the three parity bits required for 1-bit Hamming ECC
- Test that it works by applying appropriate inputs and checking the outputs
- Label the terminals to reflect their roles
- Save it as a regular circuit (logic file), reopen and retest
- Save it as a component (cmp file), reopen and retest

Part 4 (1-bit ECC)

- As indicated in the figure construct the ECC module that takes the 4-bit received data $\langle x_3, x_2, x_1, x_0\rangle$, 3-bit received parity $\langle p_2, p_1, p_0\rangle$ and 3-bit recomputed parity $\langle p'_2, p'_1, p'_0\rangle$ to generate the 4-bit restored data
- Test that it works by applying appropriate inputs and checking the outputs
- Label the terminals to reflect their roles
- Save it as a regular circuit (logic file), reopen and retest
- Save it as a component (cmp file), reopen and retest



Part 5 (1-bit ECC test setup)

- Wire up the designed components as indicated in the figure with the provision to introduce a single bit error in any of the seven lines using the EXOR gates in conjunction with the 3-bit decoder, displaying the restored data using the 7-segment display
- Test that it works by applying appropriate inputs and checking the outputs
- Label the terminals to reflect their roles
- Save it as a regular circuit (logic file), reopen and retest
- How would you extend this to allow error detection for more than 1-bit (without correction)? You need not realise this in the tool, only report the paper design in your PDF report

Marking guidelines

Assignment marking is to be done only **after** the deadline expires, as submissions gets blocked after the assignment is marked. Enter the breakup of marks while marking.

BCD to 7-segment display	
Correctly working circuit	**6**
Labels	**2**
Saving and component creation	**2**