CS370 Computer Architecture
Lab 2

Total Points: 100

Goal:

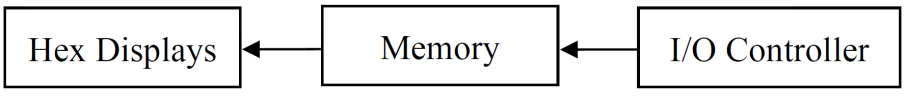
The purpose of this project is to demonstrate the use of Boolean logic and build basic circuits. Note that this lab assignment is **group** assignment with each group having **no more than 2 students**. (You are free to work by yourself.)

Preparation:

Please read the textbook's Reading Supplement "Error Detection and Correction".

Problem Statement:

The predominant storage inside computer systems are on disks drives. SCSI (Small Computer System Interface) disks are the standard disks in most Unix Workstations from Sun, HP, SGI, and other vendors. They are also the standard disks in Macintoshes and Higher-end Intel PC's, especially network servers. Consider the Wide Ultra4 SCSI, which transfers data packets in 16-bit bursts at 160 MHz with a maximum throughput of 320 MB/sec. The data transfers at higher rates can result in random-noise pulse changes from 0 to 1 and 1 to 0. As the speed of processors and electronic communications increases, these parity flips become more prevalent and the inability to detect when these errors occur can be fatal. As a Design Engineer you have been requested to create system for the transmission of these 8-bit packets from an I/O Controller to Memory using Error Correcting Code over 12-bit data bus line. Wide SCSI contains a 68-bit bus; however, for the sake of simplification we are only concerned with the data bits. The other bits in the SCSI bus are for bus arbitration, synchronization, power management, etc. In this project, we will use **even parity**.



- **Transmission Vectored Bit:** A 4-Bit Parity Vector (P_1-P_4) are interlaced with the 8-bit Data Vector (D_1:D_8), and an additional parity bit P_5 is appended to the 12-bit hamming code to ensure that the entire 13-bit vector is **even parity**. The 13-bit vector is as follows:

$$\mathbf{P_1}\ \mathbf{P_2}\ D_1\ \mathbf{P_3}\ D_2\ D_3\ D_4\ \mathbf{P_4}\ D_5\ D_6\ D_7\ D_8\ \mathbf{P_5}$$

- **Steps:**
  - (1) Create an ECC (error correcting code) Generator, at the I/O Controller from the 8-bit Data Vector. The output of the ECC Generator will be the 13-Bit Vector.
  - (2) Construct a 13-bit Data Transmission bus to send the 8-bit binary data and 5 parity bits over to Memory.
  - (3) Construct an ECC (error correcting code) Detector at Main Memory that **corrects single bit errors**. Generally, an interrupt/error handler is used to handle errors from the OS. In this exercise, we will use 3 Hex displays, 2 for data and 1 for an error status, for diagnostic purposes:
    - In the event that no error has occurred, your design must display the data transferred using the 2 Hex data displays and a "0" as an error status.
    - For single-bit transmission errors, your system must correct the error and display the data along with "C" as an error status.
    - For multiple-bit transmission errors (i.e. all other cases except above two cases), your design must display "E" in the error status display (we cannot correct these errors). Your design can display received erroneous data in the 2 Hex data displays.

Submission Instructions:

- Write a `readme.txt` file that includes (1) your group member names and the contribution of each member (who did what); (2) list each file enclosed in your submission package and briefly explain its function and usage.
- In your `.cct` file, please document each section using the Text tool (Ctrl_E) so that we know how it works. Note that **documentation is essential for FULL credit**.
- Zip all your LogicWorks files (including your circuit schematic `.cct` files and your library `.clf` file) and the `readme.txt` file into one document and name it with the first initial of your first name and your last name. For example, student Andrew Jordan's zip file would be `ajordan.zip`.
- Submit your zip file in Canvas. Note that **EVERY student is required to submit** (the submission of two members from the same group should be the same).