# [CS M51A FALL 14] Assignment 3

Due: 11/21/14

TA: Hyunjin Kim (roykim78@cs.ucla.edu)

---

## Homework Problems (40 points total)

### Problem 1 (20 points)

We would like to design a BCD-to-Gray-Code converter. The Gray Code is a coding scheme where each number sequence differs from its predecessor by exactly one bit. The variation we use for this problem is for encoding a decimal digit. The actual code values used in this problem are shown in the table below.

| Coded value | $z_3$ | $z_2$ | $z_1$ | $z_0$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 1 |
| 3 | 0 | 0 | 1 | 0 |
| 4 | 0 | 1 | 1 | 0 |
| 5 | 1 | 1 | 1 | 0 |
| 6 | 1 | 0 | 1 | 0 |
| 7 | 1 | 0 | 1 | 1 |
| 8 | 1 | 0 | 0 | 1 |
| 9 | 1 | 0 | 0 | 0 |

1. Write the truth table for the converter. How many input bits do we need? Do not forget don't-care cases.
2. Draw the K-maps for each $z$ bit. Find the prime implicants.
3. Write the minimal sum of products for each $z$ bit. How many gates would we need for an AND-OR network implementation?
4. Implement this code converter using a PLA. On each AND array line, write the product term that each line represents.

# LogiSim Design Problem (20 points)

## 1 Introduction

For this assignment you will be designing a 4-bit carry-ripple adder, which calculates the arithmetic sum of two 4-bit inputs. You will be given a **add4bit_skeleton.circ** file and you will be responsible for implementing all required modules. You will first need to design the **fullAdder** module, which calculates the sum and carry for a pair of input bits and one carry-in bit. Once this is done, use the **fullAdder** module as a component inside the **TOP_add4bit** module to build the full 4-bit adder.

## 2 Design Guidelines

### 2.1 Allowed Components

In completing the 4-bit adder, you are only allowed to use the gates NOT, AND, OR, NAND, NOR, XOR, XNOR, and the **Tunnel** and **Constant** components under the Wiring category. Refrain from using long wires to connect everything, instead utilize the tunnel component to improve readability of the circuit. Please do not directly use the components under the arithmetic category for any of the modules.

### 2.2 Bit vectors

All bit vectors used in the design are indexed in little-endian form; the most significant bit has the largest index, all the way down to the least significant bit, which is always index 0. Take note of this fact when deciding the direction of the carryOut and carryIn signals.

### 2.3 Skeleton module

**DO NOT** change the location of the input/output pins in each module or add any new pins to the skeleton module. Moving the positions of the pins can cause the signals to connect to different pins from the original setup. Points will be deducted for any errors caused by this.

## 3 Submission

Print out your design and attach it to your homework submission.