EECS 151/251A Homework 4

Due Monday, Feb 19th, 2024

Introduction

This homework is meant to test your understanding of the basic principles used to construct finite state machines. If asked to submit a Verilog module, please show the full module. If asked to simulate a Verilog module, create a testbench to run your modules in a simulator. We recommend the following free, online Verilog simulator: `https://www.edaplayground.com`.

Important: Use the register library in `EECS151.v` when sequential logic is needed for all of the HW problems.

Problem 1: Gray Code

In lecture, we discussed Gray codes. As a reminder, Gray codes are an ordering of binary numbers such that each successive value only has a single bit change relative to the previous number. For example, 00 → 01, or 00 → 10 both represent possible Gray codes. Gray codes are often seen in error correction since only one bit can change between two numbers.

1. Write a Gray code for a 4-bit binary number.
2. Create a schematic for an optimal circuit using 2-input logic gates such that when given any 3-bit value as input produces a 3-bit output representing the next number in the Gray code. For example, if 000 is passed as input to the circuit, then the output of the circuit should be 001. (*Hint:* try making a combintional logic circuit per bit)

Problem 2: Boolean Simplification

Boolean algebra can be used to simplify even the most complex Boolean expression. Simplify each expression list below to prove equivalence to the given final expression. For a list of Boolean algebra laws vist: `https://www.asic-world.com/digital/boolean1.html`.

1. $AB + BCD + \overline{BC}D + B\overline{A} + \overline{A}B \Rightarrow B + D$
2. $\overline{D(A0 + D\overline{A})} + B + \overline{BA} \Rightarrow 1$
3. $\overline{C}(A + \overline{A}B) + C(\overline{A}B + A\overline{B}) \Rightarrow \overline{C}A + \overline{A}B + CA\overline{B}$

Problem 3: SOP to POS

Below there are two optimal Boolean functions in their SOP forms. For each function shown, use Boolean algebra to generate the POS. **Work must be shown.** (*Hint*: You can confirm your answer using a K Map).

1. $ABD + AC$
2. $A + B\overline{D} + BC$

Problem 4: NANDs to Other Gates

How many two variable functions can be implemented with a single AND gate along with (any number) of inverters? Which two variable functions cannot be implemented?

Problem 5: Karnaugh Maps

Karnaugh maps are used to quickly generate simplified Boolean expression for a given Boolean function. Below are truth tables for a unknown functions. For truth table each, draw the accompanying Karnaugh map, then write out a simplified Boolean expression in both **POS** and **SOP** forms using the Karnaugh map. The solutions should use the least amount of logic gates possible.

1. *Hint:* the optimal SOP has 3 terms and the optimal POS has 3 terms.

A	B	C	Output
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

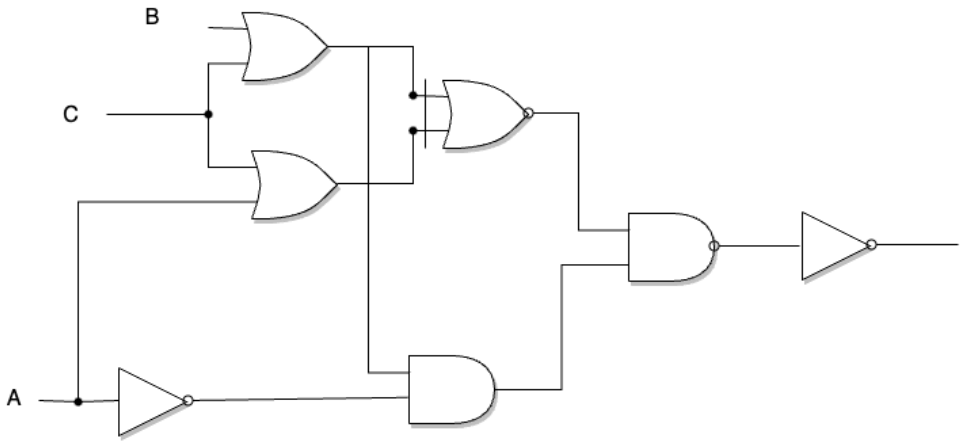
2. *Hint:* the optimal SOP has 6 terms and the optimal POS has 3 terms.

A	B	C	D	Output
0	0	0	0	1
0	0	0	1	-
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	-
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	-
1	1	1	1	1

Problem 6: It's a NAND's World

CMOS is the most modern process for digital logic. In CMOS, it is simple to create NAND gates which is advantageous because all logic can be reduced to NAND gates.

1. How would you wire a NAND gate to construct an inverter for any given input?
2. How would you wire a NAND gate to construct an OR gate for any given input?
3. How would you wire a NAND gate to construct an AND gate for any given input?
4. If the above is true, then why can all logic be represent with NAND gates?
5. Below is a schematic using a various types of 2-input gates that you want to create, but you only have NAND gates! Translate the circuit into a new schematic using only 2-input NAND gates. Your new circuit should use the minimum number of NAND gates.



Problem 7: Driving a Car (FSM)

Let's create a FSM to represent driving a car. To reduce the complexity of such a dynamic problem, we greatly simplify the our conceptual model of driving. The FSM will represent the state of the car. Only two things can change the state of the car: (1) the driver making a decision (to continue driving, or to stop and park the car), or (2) and emergency (flat tire, accident, or a mechanical failure). The only thing we care about as engineers is whether the brake and hazard lights work or not. The brake lights turn on when braking and the driver turns on the hazards light when an emergency has occurred. Below we explain our simplified conceptual model of driving:

a. The car can be: *on*, *off*, *accelerating*, *braking*, *parking*, or in an *emergency* situation (stopped and non-functional)

b. The driver enters the car when it's off, and must decide whether to start driving (they can just sit in the car and think about K-Maps)

c. If the driver is in the car and the car is off, then the driver cannot perform any action except turn the car on

d. In order to start accelerating, the driver must first press the brake pedal to switch gears

e. In order to park the car, the driver must first press the brake pedal to switch gears

f. At any point when accelerating, the driver can slow down by braking

g. In a case of an emergency, regardless if the driver wants to keep driving, they must apply the brakes to stop the car, turn the car off, and then turn on the hazard lights

h. Emergencies can only occur when accelerating in which a light on the dashboard indicates that the driver must stop and turn the car off

Answer the following questions:

1. Draw a Moore state machine diagram. In this diagram you must:
   (a) Have a bit vector to representing the state of the car. (This should be gray coded)
   (b) Have 2 inputs:
      i. *driver* indicating the driver decision to take an action to drive the car, or stop it
      ii. *emergency* indicating an emergency has happened and the dashboard light is on
   (c) Have 1 output: *light* which is asserted if either the brake *or* hazard lights are on
2. Telsa took interest in the simple driving model. They believe that this simplified model can help their engineers understand the decision making process for drivers in the Bay. Engineers collected data from two drivers to help train their auto navigation ML model.
   (a) The state of the car was recorded for the first driver. Telsa engineers want to deduce the driver's decision making process.
   (b) The second driver's decisions were recorded live and the engineers want their ML model to accurate guess the state of the car.

In order to verify the ML model's output, the engineers need to know the correct answers! Use your FSM to fill in the tables below for each driver. (Hint: Aren't Telsa owners suppose to be cautious software engineers. One is a very aggressive driver...).

Time	Car	Driver	Emergency	Light
0	CAR_OFF			
1	CAR_ON			
2	BRAKE			
3	ACCEL			
4	ACCEL			
5	BRAKE			
6	ACCEL			
7	BRAKE			
8	ACCEL			
9	ACCEL			
10	BRAKE			
11	PARKING			
12	CAR_OFF			

Table 1: Telsa driver 1

Time	Car	Driver	Emergency	Light
0		1	0	0
1		1	0	0
2		1	0	1
3		1	0	0
4		0	0	0
5		1	0	1
6		1	0	0
7		1	0	0
8		1	0	0
9		1	0	0
10		1	0	0
11		1	0	0
12		1	1	1
13		1	1	1
14		1	1	0
15		1	1	0
16		1	1	0
17		1	1	0

Table 2: Telsa driver 2