EE116C/CS151B Homework 2

Problem 1

As: CPU time = Sum (instruction counts * CPI) * seconds/cycles

Thus:
Old CPU time = (500m*1+300m*10+100m*3)*CT = 3800m*CT;
New CPU time = (500m*0.75*1+300m*10+100m*3)*1.1CT = 4042.5m*CT;
Thus the new CPU time is larger than the old one. So it is not a good design.

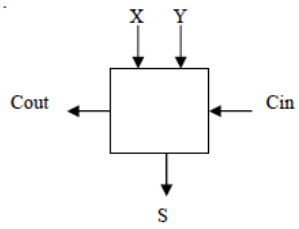
Problem 2

When operating normally, the ALU does subtraction based on the equation:
A – B = A + (–B)
In those cases, the CarryIn is set to 1 as part of the computation of –B. Since the malfunction causes CarryIn to be 0, the result for subtraction will always be one less than it should be. Specifically, instead of computing A – B, the ALU will compute A – B – 1.

A user/programmer will observe incorrect results for all instructions that require the ALU to perform subtraction. Specifically:
A) The sub instruction will compute Rs – Rt – 1 instead of Rs – Rt.
B) The beq instruction will branch whenever Rs = = Rt + 1 instead of whenever Rs = = Rt.
C) The slt instruction will set Rd to 1 whenever Rs < Rt + 1 instead of whenever Rs < Rt.

Problem 3

A modular 1-bit adder:



Truth table for the 1-bit adder:

X	Y	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Karnaugh map for S:

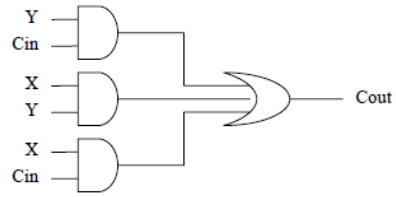
S		YCin 00	01	11	10
X	0	0	1	0	1
	1	1	0	1	0

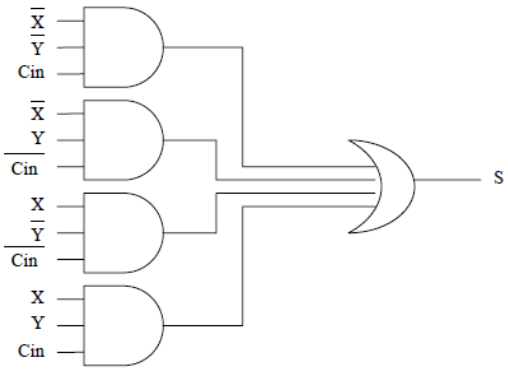
$$S = \overline{X}\overline{Y}Cin + \overline{X}Y\overline{Cin} + X\overline{Y}\overline{Cin} + XYCin$$

Karnaugh map for Cout:

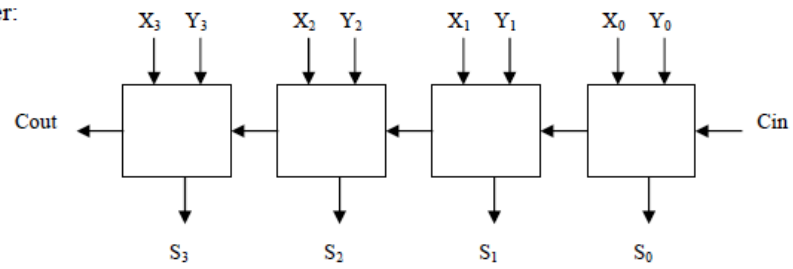
Cout		YCin 00	01	11	10
X	0	0	0	1	0
	1	0	1	1	1

$$Cout = YCin + XY + XCin$$





4-bit adder:



problem 4