Borja Rojo

2/12/15

Reduction Techniques and Multilevel Logic

Lab 2

Abstract

This lab was an exercise in reducing logic circuits to their lowest cost. Reduction with Boolean algebra and Karnaugh maps were the main techniques used. This lab also exhibited the variability of combining different gates to represent other gates logically. NAND gates were often used to simulate NOT and NOR gates, as well as the more complex AND and OR gates.

Assignment

Part 1

This section of the lab was about proving certain logical relationships using Boolean Algebra.

a) Prove A (A + B) = A

A (A + B) ?= A Given

AA + AB ?= A Distributive Property

A + AB ?= A Theorem 7a.

A = A Absorption

b) Prove (A + B)(A + B’) = A

(A + B)(A + B’) ?= A Given

A = A Combining, Theorem 14a.

c) Prove (A + B)(A + C’) = A + BC’

(A + B)(A + C’) ?= A + BC’ Given

A(A + C’) + B(A + C’) ?= A + BC’ Distributive Property

AA + AC’ + BA + BC’ ?= A+BC’ Distributive Property

A + A(B+C’) + BC’ ?= A+BC’ Reverse Distributive

A + BC’ = A + BC’ Absorption

Part 2

This section of the lab was about simplifying functions using Boolean Algebra

a) Simplify F(A,B,C) = BC’ + A’BC + ABC

= BC’ + A’BC + ABC Given

= BC’ + BC Combining

= B Combining

b) Simplify G(A,B,C) = (A + B)(A’ + B + C)(A’ + B + C’)

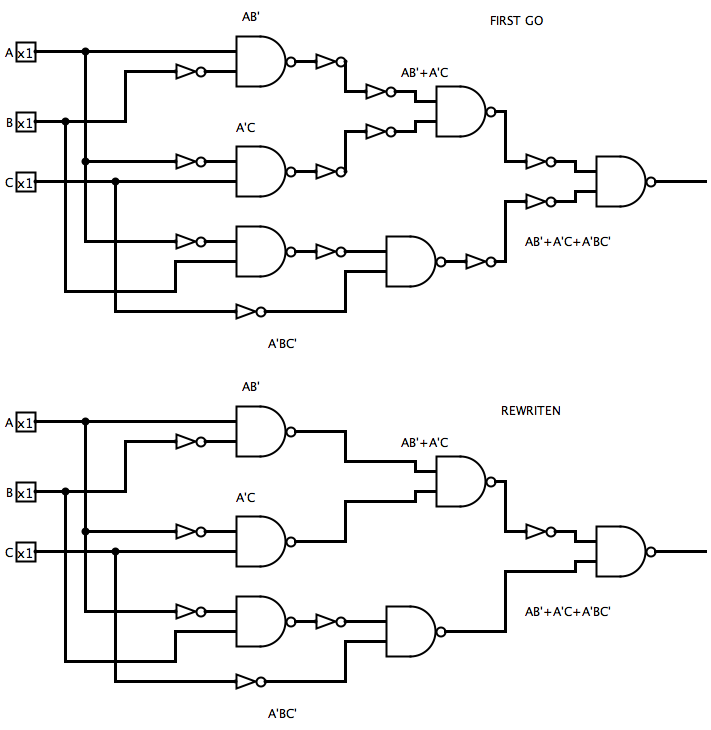
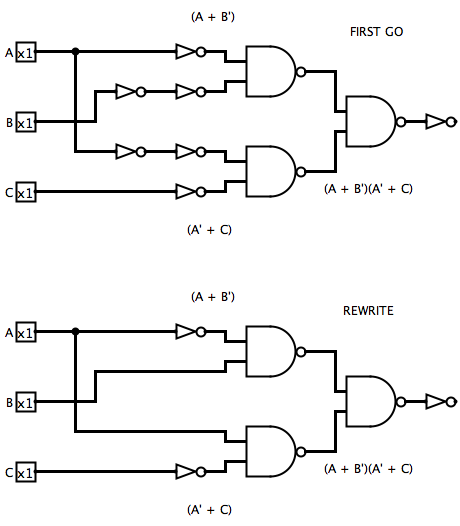
= (A + B)(A’ + B + C)(A’ + B + C’) Given

= (A + B)(A’ + B) Combing; C and C’ cancel, (A’+ B) as one term

= B Combing

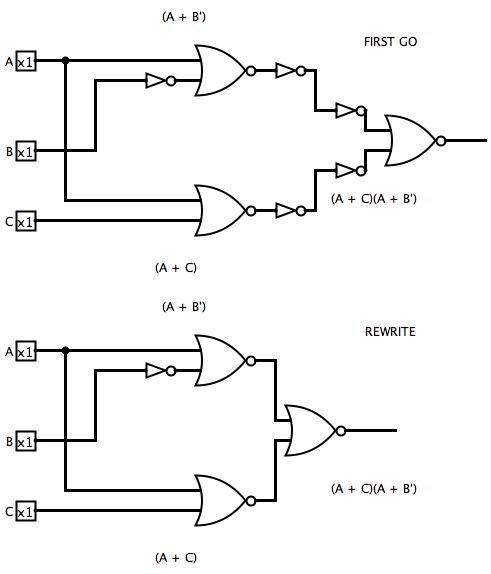
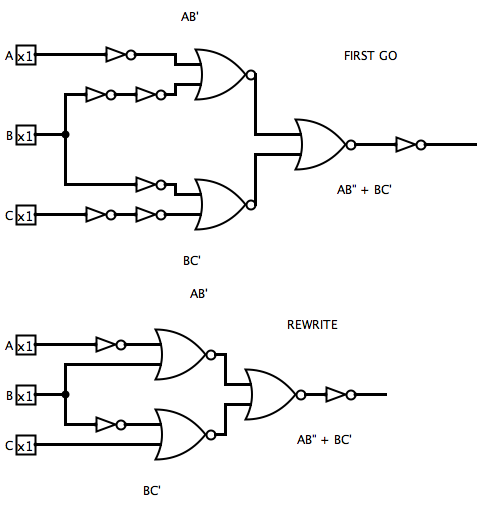
Part 3

This section of the lab required diagrams of the following functions using NAND and NOT gates. The system used was formulating the circuit as shown and then expanding each gate as necessary.

a) F(A,B,C) = AB’ + A’C + A’BC’ b) G(A,B,C) = (A’ + C)(A + B’)

Part 4

This section of the lab required diagrams of the following functions using NOR and NOT gates. The system used was formulating the circuit as shown and then expanding each gate as necessary.

a) F(A,B,C) = (A + C)(A + B’) b) G(A,B,C) = AB’ + BC’

Part 5

This section of the lab was about designing a circuit of 3 inputs that would output TRUE when a prime number was given in binary as the input; numbers 0-7.

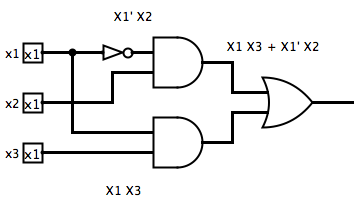
Truth Table Karnaugh Map

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X1\X2X3 | 00 | 01 | 11 | 10 |
| 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | X1 | X2 | X3 | ƒ |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 2 | 0 | 1 | 0 | 1 |
| 3 | 0 | 1 | 1 | 1 |
| 4 | 1 | 0 | 0 | 0 |
| 5 | 1 | 0 | 1 | 1 |
| 6 | 1 | 1 | 0 | 0 |
| 7 | 1 | 1 | 1 | 1 |

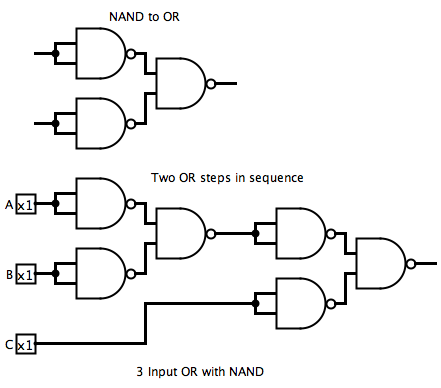
Function

ƒ = X1X3 + X1’X2



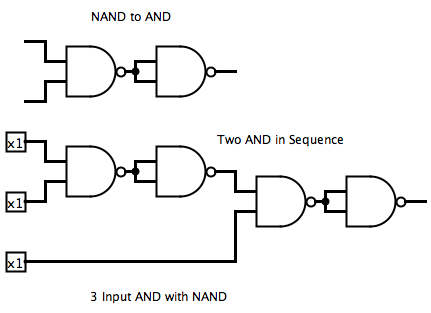
Additional Questions

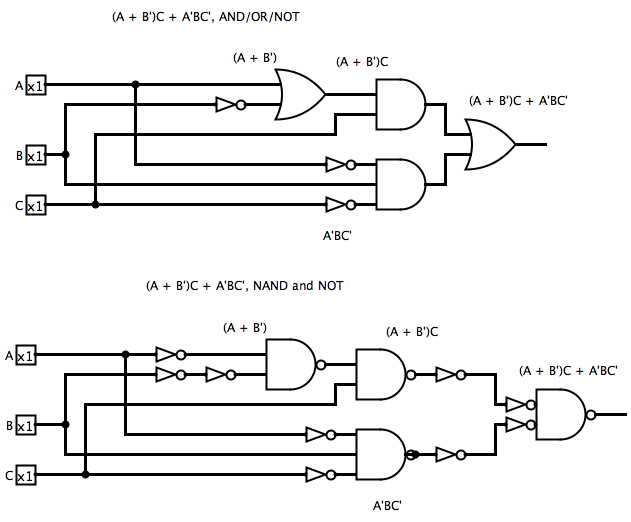
1) Design a 3-input OR gate using only 2-input NAND gates



The approach used to design this circuit was building a base using any gate and then expanding the base using only NAND gates. A three input OR is the same as two OR’s chained together.

2) Design a 3-input AND gate using only two 2-input NAND gates

The approach used to design this circuit was building using any gate and then expanding the base using only NAND gates. A three input AND is the same as two AND’s chained together.

3) Design a circuit for the function F(A,B,C) = (A + B’)C + A’BC’ using

1. only AND, OR, and NOT gates
2. only NAND and NOT gates

Assumed: 3-input gates are available.

4) Design a circuit that will produce a parity bit for three inputs using XOR gates

|  |  |  |  |
| --- | --- | --- | --- |
| X1 | X2 | X3 | ƒ |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

Truth Table

An observation found is that when X1 = 0, the function match an XOR table for the two variables X2 and X3 and represents an XAND table when X1 = 1. Another observation found is that when X2 ≠ X3, X1 ≠ ƒ. This lead to the conclusion that the output of X2 and X3 through an XOR gate can be XOR’d with X1 in order to get ƒ. In other words, ƒ23 = X1’ at all times in order for the parity bit to be correct. Therefore, the logical circuit created for this function is two cascaded XOR gates.

