Lab 2 - Combinational Logic Circuits (II)

CDA 3201L-003
Jason Keene and Jacob Manfre
Submitted Feb 1st, 2015

Purpose and Objectives

During our last lab we demonstrated the universality of NAND gates. However, we did not implement the XOR gate with NAND gates only. We set out to do that in this lab. We also designed a prototype circuit for a car's signaling electronics. This included left and right turing switches and signals, a brake switch and lights, and an emergency switch that activates both turning lights.

Component list

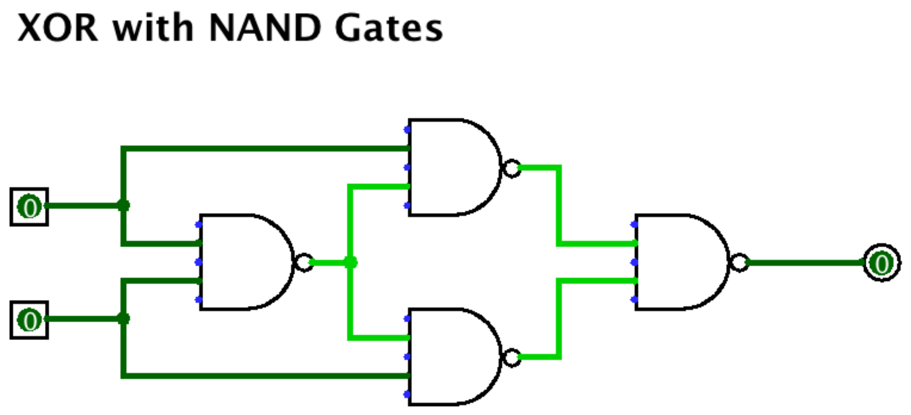
- Breadboard
- Wiring
- 5v power supply
- 1 x Quad 2-Input NAND Gate (74LS00)
- 1 x Quad 2-Input OR Gate (74LS32)
- 5x LEDs
- 4x Resistors (470 Ohms 5%)

Design

We began by using boolean algebra laws to covert the given expression to use only NAND gates:

```
xy' + x'y
0 + xy' + x'y + 0                  # identity law
x'x + xy' + x'y + y'y              # inverse law
x(x' + y') + y(x' + y')            # distributive
[x(x' + y') + y(x' + y')]''        # double complement
{ [x(x' + y')]' [y(x' + y')]' }'   # demorgan's
{ [x(xy)']' [y(xy)']' }'           # demorgan's
```

We then implemented this in logisim:



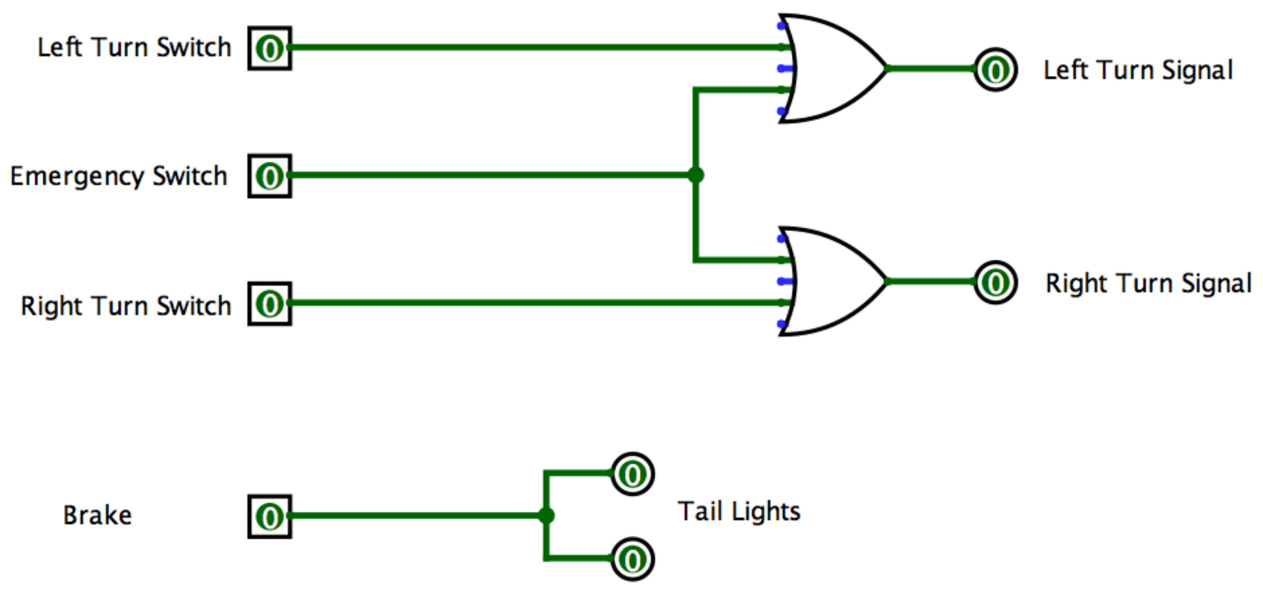
We then implemented the logisim design in hardware.

For the car signals we began with establishing all inputs and outputs and creating a truth table for all possible combinations:

left in	right in	brake	emerg	left out	right out	tail
0	0	0	0	0	0	0
0	0	0	1	1	1	0
0	0	1	0	0	0	1
0	0	1	1	1	1	1
0	1	0	0	0	1	0
0	1	0	1	1	1	0
0	1	1	0	0	1	1
0	1	1	1	1	1	1
1	0	0	0	1	0	0
1	0	0	1	1	1	0
1	0	1	0	1	0	1
1	0	1	1	1	1	1
1	1	0	0	1	1	0
1	1	0	1	1	1	0
1	1	1	0	1	1	1
1	1	1	1	1	1	1

Once we had this truth table it seemed obvious that the tail lights were directly driven by the brake switch and that the emergency switch is simply OR'd with the left and right switches to control the left and right lights.

We implemented this design in logisim and checked the result against the truth table to verify it was correct:



We then implemented the logisim design in hardware.

Test vectors and verification

To verify that we implemented XOR with NAND gates correctly we tested each input against what we know XOR to output. Additionally, we evaluated the result of the original expression for each input to verify it is indeed the XOR operation.

a	b	a ⊕ b	xy' + x'y
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

The outputs of our circuit and the original expression matched the expected behavior of XOR which demonstrates that XOR can be implemented with NAND gates.

For the car signaling circuit we first tested that the brake input directly drove the tail lights. Once we established this was true we tested the combinations of the three inputs (left in, right in and emerg) against the truth table's outputs (left out and right out). We also observed that at no time the brake lights were affected by any of the three inputs (left in, right in and emerg).

.

left in	right in	emerg	left out	right out
0	0	0	0	0
0	0	1	1	1
0	1	0	0	1
0	1	1	1	1
1	0	0	1	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Discussion and conclusion

Through this exercise we learned that XOR can be represented as `a'b + b'a` and that the remaining gate XOR (and thus XNOR) can be implemented with only NAND gates. This fully demonstates the universality of NAND. Additionally, we showed that a single input (`A`) that is OR'd with other inputs (`B`) will override `B` when `A` is set to high but continue to allow `B` to function normally when `A` is low. This might be useful in designing circuits where you want to override multiple inputs, setting them either high or low based on a single input.