Recitation 09

Combinational Logic

Today's agenda

- We will discuss in recitation
  - Combinational Logic
    - Logic gates, truth tables, rules
- For homework Tonight
  - R10
    - Two parts
      - One – given an expression, produce the truth table
      - Two – given a truth table, write a logical expression for it

Building Blocks

For combinational logic

Combinational Logic

- There is no memory
  - That is, the outputs are a function ONLY of the **current** inputs, not of anything in the past
- Values are either true or false (but not both, or anything else)
  - We commonly represent true with 1 and false with 0
- There is at least one input and at least one output
  - But there can be more
  - Each input is either true or false
  - Each output should be defined for all possible values for the inputs!

Combinational Logic

- There are three main ways to represent combinational circuits
  1. As a circuit diagram
  2. As a set of equations/expressions
  3. As a truth table

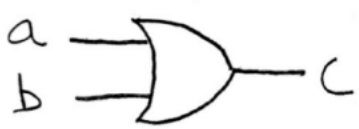
Combinational Logic

- Logical expressions are build from a number of “gates”
- You are already familiar with the most important ones!
  - AND, OR, NOT
    - All boolean functions can be written with these three building blocks!
  - There are others, like XOR and NAND
    - All boolean functions can be written with just NAND!!!
- Letters/words are used to represent the inputs
  - For recitation homework, the variables are a single letter

OR

a	b	a OR b
0	0	0
0	1	1
1	0	1
1	1	1

- Output is 1 if ANY input is 1, 0 otherwise
- Output is 0 if ALL inputs are 0, 1 otherwise
- Output is the first 1 if there is one, last 0 otherwise (from the python world)
- a or b is written as a + b
  - $a \mid b$ or $a \vee b$ are also common
  - Use a + b for recitation



AND

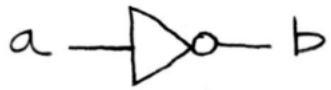
a	b	a AND b
0	0	0
0	1	0
1	0	0
1	1	1

- Output is 0 if ANY input is 0, 1 otherwise
- Output is 1 if ALL inputs are 1, 0 otherwise
- Output is the first 0 if there is one, last 1 otherwise (from the python world)
- a or b is written as ab
  - $a \,\&\, b$ or $a \wedge b$ are also common
  - Use ab for recitation

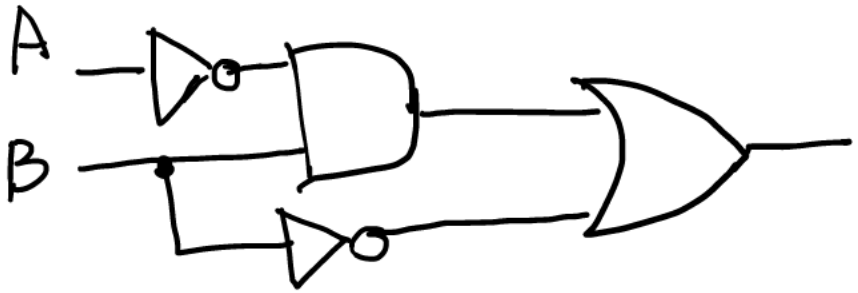
NOT

- Output is 1 if the input is 0, 0 otherwise
- A few ways to represent, both in text and graphically
  - Typically, as a “hat” on the input name, such as $\bar{a}$
  - Can also be written with a prefix operator, such as
    - $\neg a$, $!a$, or $\sim a$
  - For recitation homework, use ~
- Note that when drawing a not, it is common to see just the circle on the input or output of another gate

a	NOT a
0	1
1	0



Circuit Diagram



- I stole this from the lecture, among the other pictures

Expression

- ~ab + ~b
- $\bar{a}b + \bar{b}$

Truth Table

a	b	~ab + ~b
0	0	1
0	1	1
1	0	1
1	1	0

Filling in a truth table

- First, list out all of the possible inputs
  - For N inputs, there are 2^N possibilities
  - It helps to split things in two, that is, have space for the 2^N possibilities, and for the first variable put 0 for the first half and 1 for the second half
    - For the next variable, put 0 for the first half of the previous variable's 0 group and 1 for the second half, then do the same for its 1 group
- Then, evaluate the expression for each possible inputs
  - This is tedious, so there are some shortcuts, especially if the expression is written nicely

Filling in a truth table

- Groups of things ANDed together are typically called a clause
  - That is, clauses are things separated by ORs
- For each clause, see what the variables present are
  - For any variable that is negated, keep in mind that that variable is 0
  - For any other variable in the clause, keep in mind it is 1
  - Then, look through the truth table and wherever you see a row that has all of the variables in the clause with the right value, put a 1 for the output
- Then when you are out of clauses, fill in 0 for any output left

Getting a circuit from a truth table

- For each output, look for where that output is 1 in the truth table
  - Look at the list of inputs – anywhere an input a is 1, write a in the clause
    - Anywhere an input a is 0, write $\bar{a}$ in the clause
  - Say the output equals all of the clauses ORed together
  - Typically you can simplify, but that's a topic for a different day