CS 251 — Lecture 3

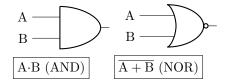*Bartosz Antczak*   *Instructor: Stephen Mann*   *January 10, 2017*

Recall — Gates

Gates can be considered as functions that take in a fixed amount of Boolean values and maps them to a single Boolean value. Examples include AND, OR, and NOT. We can use truth tables to draw out these mappings (some tables were drawn in the last lecture). Some examples are shown:

We interpret these diagrams into Boolean algebra, which can be solved using a truth table.

3.0.1   What's inside a gate?

Gates contain transistors, which are electrically-controlled switches. There are two main transistors:

- NMOS (High resistance when $A = 0$; Low resistance when $A = 1$)

- PMOS (High resistance when $A = 1$; Low resistance when $A = 0$). Will be denoted with a little "bubble" on diagram to distinguish it from NMOS

Figure 3.1: A PMOS transistor will be denoted with a little bubble to distinguish it from NMOS (diagram courtesy from Prof. Mann's lecture slides)

We'll combine both of these transistors to create a CMOS transistor.

3.1   How gates compute Boolean values

Depending on what current state each input is in (either 1 or 0) each respective transistor will either have a high or low resistance. A high resistance restricts any flow, whereas a low resistance allows for it. This means that based on the resistance of transistors in a given circuit, the flow will either have a clean path to power (denoted by 1) or ground (denoted by 0).

Example 3.1.1. *The filled in table from a CMOS diagram from Prof. Mann's slides on slide 2-16:*

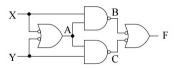| A	B	Q_1	Q_2	Q_3	Q_4	Z
0	0	LOW	LOW	HIGH	HIGH	1
0	1	LOW	HIGH	HIGH	LOW	1
1	0	HIGH	LOW	LOW	HIGH	1
1	1	HIGH	HIGH	LOW	LOW	0

Figure 3.2: Inputs A and B are connected to their respective transistors. The flow from Z can either flow towards power (1) or ground (0) (diagram courtesy from Prof. Mann's lecture slides)

3.1.1 Deriving a Truth Table from a Circuit

Interpreting a circuit is pretty simple:

1. Find a gate whose all of its inputs are known

2. Compute its output

3. Repeat

Example 3.1.2. *Fill in the truth table defining this circuit diagram (diagram courtesy from Prof. Mann's lecture slides)*

Filling in the table yields

X	Y	A	B	C	F
0	0	1	1	1	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	0

This is an XOR gate ($A \oplus B$), which returns 1 if one and only one input is true.

To finish off...

3.1.2 Decoders

This is a useful hardware component. A decoder converts binary to "unary". With n inputs, we'll have 2^n outputs.

3.1.3 Multiplexors (or Multiplexers)

Based on inputs S_0 and S_1, a multiplexor determines which input D_S to process (i.e., + - / *). For instance, if we want to add two values, we set S_0 and S_1 to certain values, and then we select the given output D_S that outputs addition. More on this in the next lecture.