UW Math Circle
October 3rd, 2019

Digital Circuits

Recall that the $\boxed{\text{NAND}}$ gate implements the negation of the logical conjunction of its arguments:

A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

Problem 1: Constant circuits

Build a $\boxed{0}$ circuit from $\boxed{\text{NAND}}$ gates, that outputs 0 regardless of its single input:

In	Out
0	0
1	0

And similarly a $\boxed{1}$ circuit that always outputs 1.

Problem 2: Universal-NOR

The $\boxed{\text{NOR}}$ gate is the negation of the logical inclusive-or:

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

Show that $\boxed{\text{NOR}}$ is universal, by using it to implement all the gates of some other universal family.

Problem 3: Universal-AND

Show that $\boxed{\text{AND}}$ and $\boxed{\text{NEG}}$ form a universal family.

Problem 4: SWITCH

The $\boxed{\text{SWITCH}}$ circuit has three inputs, S, A and B, and one output that is equal to A if S is 0 and equal to B if S is 1. Implement $\boxed{\text{SWITCH}}$ using basic circuits.

S	A	B	Out
0	a	b	a
1	a	b	b

Problem 5: Half-Adder

Suppose that one wants to add two one-bit numbers. The two inputs are each either 0 or 1, so the sum can be either 0, 1 or 2, represented in two-digit binary as 00, 01, and 10. Therefore a $\boxed{\text{HALF-ADD}}$ circuit to add must have two outputs, the low digit (S, for *sum*) and the high-digit (C, for *carry*).

A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Implement $\boxed{\text{HALF-ADD}}$. There will be two different gates providing the two outputs, but the circuits need not be completely independent (pieces could be shared).

Problem 6: Full-Adder

To add larger numbers, represented in binary using several bits, carries must be propagated. For example, adding 6 and 11 in binary

Place	2^4	2^3	2^2	2^1	2^0	
Carry	1	1	1	0		
A		0	1	1	0	(6)
B		1	0	1	1	(11)
Sum	1	0	0	0	1	(17)

This means that the addition circuit for each place other than the first must take three inputs, the bits A and B, and the carry coming in from the right, C_{in}, and produce a sum bit S and a carry bit C_{out} that goes out to the next addition on the left. Because there are 3 inputs, the maximum sum is 3.

(a) Write the logic table for the 3-input 2-output $\boxed{\text{ADD}}$ circuit, and implement it using existing gates - you may use $\boxed{\text{HALF-ADD}}$.

(b) Using $\boxed{\text{ADD}}$ and $\boxed{\text{HALF-ADD}}$, construct a circuit that adds two 4-bit numbers to construct a 5-bit output.

(c) What is the *latency* of your circuit - the maximum depth from input to output, in terms of primitive $\boxed{\text{NAND}}$ gates? Can you give a formula for the depth as a function of the number of bits in the input number, and use it to give the latency of a 64-bit adder?