Computer Architecture Homework 4

Spring 2021, April

1 Boolean Algebra

Simplify the following Boolean expressions step by step (as simple as possible).

a. $(A+B)(A+\overline{B})C$

$= (A + A\overline{B} + BA + 0)C$
$= (A + A(B+\overline{B}))C$
$= (A+A)C$
$= AC$

b. $\overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,C + \overline{A}BC + \overline{A}B\overline{C} + ABC + AB\overline{C}$ (Extra terms may help.)

$= \overline{A}\overline{B}(\overline{C}+C) + \overline{A}B(C+\overline{C}) + AB(C+\overline{C})$
$= \overline{A}\overline{B} + \overline{A}B + AB$
$= \overline{A}(\overline{B}+B) + AB$
$= \overline{A} + AB$
$= \overline{\overline{A+AB}} = \overline{A\overline{AB}} = \overline{A(A+\overline{B})} = \overline{A\overline{B}}$
$= \overline{A} + B$

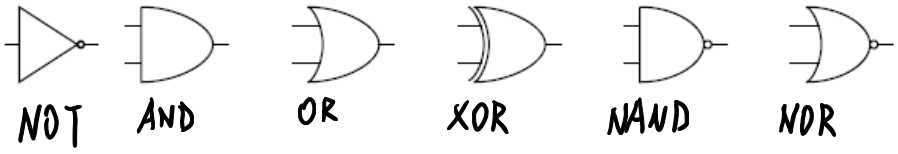
c. $\overline{A}(A+B) + (B+AA)(A+\overline{B})$

$= 0 + \overline{A}B + (B+A)(A+\overline{B})$
$= \overline{A}B + BA + 0 + A + A\overline{B}$
$= \overline{A}B + A + A(B+\overline{B})$
$= A + \overline{A}B$
$= \overline{\overline{A+\overline{A}B}}$
$= \overline{\overline{A}\,\overline{\overline{A}B}}$
$= \overline{\overline{A}(A+\overline{B})}$
$= \overline{\overline{A}\,\overline{B}}$
$= A + B$

2 Logic Gates

2.1 Elementary Logic Gates

a. Label the following logic gates:



b. Convert the following to boolean expressions on input signals A and B:

(1) NAND

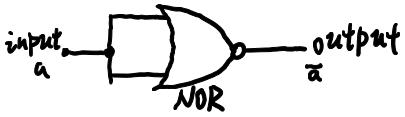
$\overline{AB} = \bar{A} + \bar{B} = \bar{A}\bar{B} + \bar{A}B + A\bar{B}$

(2) XOR

$\bar{A}B + A\bar{B}$

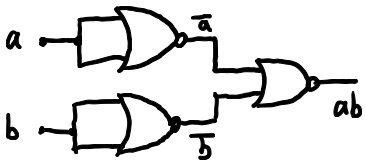
2.2 Design Logic Gates

a. Create a NOT gate using only NOR gates.



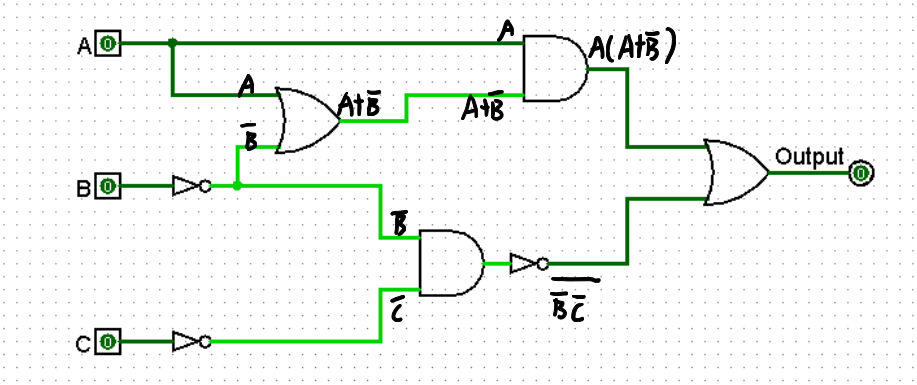
b. Create an AND gate using only NOR gates.

$\bar{a} = \overline{a+a}$, $ab = \overline{\overline{ab}} = \overline{\bar{a}+\bar{b}}$



2.3 Simplification Problem

The circuit shown below can be simplified. Please write the origin boolean expression of this circuit and simplify the expression step by step. Then draw the circuit according to the simplified boolean expression using the minimum number of two-input logic gates.



$A(A+\bar{B})+\overline{\bar{B}\bar{C}}$

$= A+A\bar{B}+B+C$

$= A+C+\overline{\overline{A\bar{B}}+\bar{B}}$

$= A+C+\overline{\bar{B}A\bar{B}}$

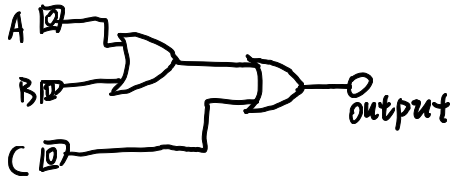
$= A+C+\overline{B(\bar{A}+B)}$

$= A+C+\overline{\bar{B}\bar{A}}$

$= A+C+A+B$

$= A+B+C$

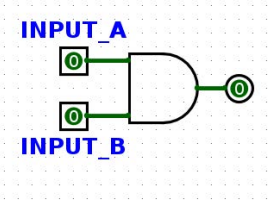
So, only 2 OR gate.



3 SDS and FSM

3.1 Synchronous Digital System

There are two basic types of circuits: combinational logic circuits and state elements. **Combinational logic** circuits simply change based on their inputs after whatever propagation delay is associated with them. For example, if an AND gate (pictured below) has an associated propagation delay of 2ps, its output will change based on its input as follows:

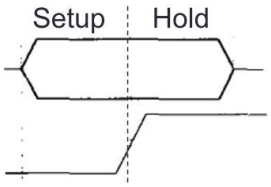


input a 2L 4H 3L 2H 2L 1H 5L 1H 1L 1H 2L
input b 2L 4H 3L 2H 2L 1H 5L 1H 1L 1H 2L
output 2U 2L 4H 3L 2H 2L 1H 5L 1H 1L 1H

Where U, L and H refer to an undefined, low(0), or high(1) signal respectively and the preceding number refers to the number of picosecond(ps). You should notice that the output of this AND gate always changes 2ps after its inputs change.

State elements, on the other hand, can *remember* their inputs ever after the inputs change. State elements change value based on a clock signal. A rising edge-triggered register, for example, samples its input at the rising edge of the clock (when the clock signal goes from 0 to 1).

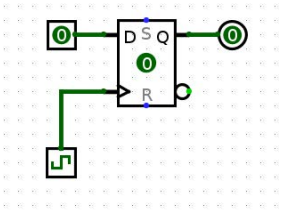
Like logic gates, registers also have a delay associated with them before their output will reflect the input that was sampled. This is called the **clk-to-q delay**. ('Q' often indicates output). This is the time between the rising edge of the clock signal and the time register's output reflects the input change.



The input the register samples has to be stable for a certain amount of time around the rising edge of the clock for the input to be sampled accurately. The

amount of time before the rising edge the input must be stable is called the **setup time**, and the time after the rising edge the input must be stable is called the **hold time**. Hold time is included in clk-to-q delay, so clk-to-q time will always be greater than equal to hold time.

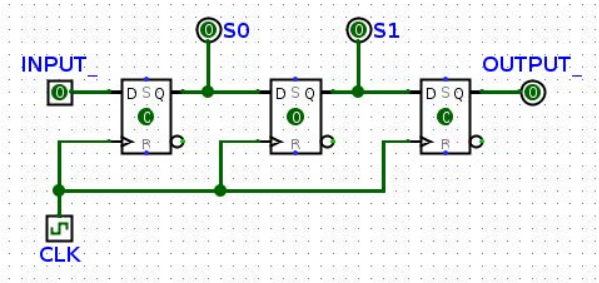
For the following register circuit, assume **setup time** of 2.5ps, **hold time** of 1.5ps, and a **clk-to-q time** of 1.5ps. The clock signal has a period of 13ps.



clock 6.5L 6.5H 6.5L 6.5H
input 1L 2H 1L 5H 3L 2H 2L 1H 5L 1H 3L
output 8U 13H 5L

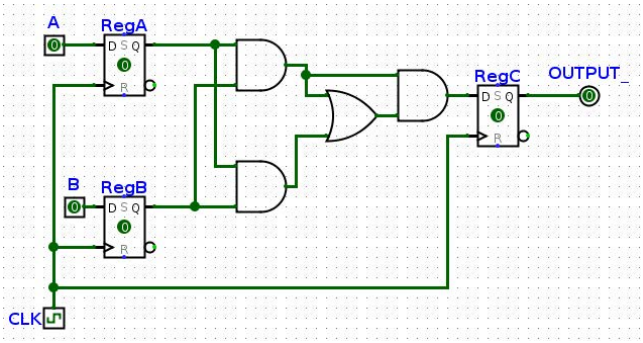
You'll notice that the value of the output in the diagram above doesn't change immediately after the rising edge of the clock. Clock cycle time must be small enough that inputs to registers don't change within the hold time and large enough to account for clk-to-q times, setup times, and combinational logic delays.

a. For the following circuits, fill out the timing diagram. The clock period (rising edge to rising edge) is 8ps. For every register, clk-to-q delay is 2ps, setup time is 4ps, and hold time is 2ps. NOT gates have a 2ps propagation delay.



clk 4L 4H 4L 4H 4L 4H 4L 4H 4L 4H 4L 4H
in 14L 4H 6L 16H 8L
s_0 6U 16L 8U 16H 2L
s_1 14U 16L 8U 10H
out 22U 16L 8U 2H

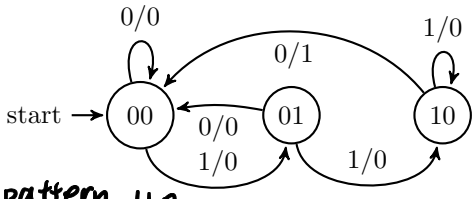
b. In the circuit below, RegA and RegB have setup, hold, and clk-to-q times of 4ns, all logic gates have a delay of 5ns, and RegC has a setup time of 6ns. What is the maximum allowable hold time for RegC? What is the minimum acceptable clock cycle time for this circuit, and clock frequency does it correspond to?



clk-to-q of A/B + Best CL = thold max = 4 + (5+5) = 14 ns
So, the maximum allowable hold time for RegC is 14ns.
min Clk cycle = CLK-to-q + CL_{max} + setup = 4 + (5+5+5) + 6 = 25 ns
So, the minimum acceptable clock cycle time is 25ns
corresponding frequency is $f = \frac{1}{T} = \frac{1}{25 \times 10^{-9}\,s} = 40\,MHz$

3.2 Finite State Machine

a. What pattern in a bitstring does the FSM below detect? What would it output for the input bitstring "011001001110"?



It detects pattern 110
000100000001

b. Fill in the following FSM for outputting a 1 whenever we have two repeating bits as the most recent bits, and a 0 otherwise. You may not need all states.