CS 61C Fall 2019

Logic, SDS

Discussion 6: October 7, 2019

1 Boolean Logic

In digital electronics, it is often important to get certain outputs based on your inputs, as laid out by a truth table. Truth tables map directly to Boolean expressions, and Boolean expressions map directly to logic gates. However, in order to minimize the number of logic gates needed to implement a circuit, it is often useful to simplify long Boolean expressions.

We can simplify expressions using the nine key laws of Boolean algebra:

Name	AND Form	OR form
Commutative	$AB = BA$	$A + B = B + A$
Associative	$AB(C) = A(BC)$	$A + (B + C) = (A + B) + C$
Identity	$1A = A$	$0 + A = A$
Null	$0A = 0$	$1 + A = 1$
Absorption	$A(A + B) = A$	$A + AB = A$
Distributive	$(A + B)(A + C) = A + BC$	$A(B + C) = AB + AC$
Idempotent	$A(A) = A$	$A + A = A$
Inverse	$A(\overline{A}) = 0$	$A + \overline{A} = 1$
Demorgan's	$\overline{AB} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A}(\overline{B})$

1.1 Simplify the following Boolean expressions:

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

$$(A+B)(A+\bar{B})C = (A+B\bar{B})C \quad \text{(distributive property)}$$
$$= AC$$

$(AA + A\bar{B} + AB + B\bar{B})C$

$(A + A(\bar{B} + B) + B\bar{B})C$ ($B\bar{B} = 0$)

$(A + A)C$

AC

(b) $\bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + AB\bar{C} + A\bar{B}\bar{C} + ABC + A\bar{B}C$

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

($\bar{B}+B = 1$, Inverse property; adding this does not change the truth table; $\bar{A}+A = 1$, $\bar{C}+C = 1$)

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

split the line, change the sign

$$\begin{aligned}\overline{A(\bar{B}\bar{C}+BC)} &= \bar{A}+\overline{\bar{B}\bar{C}+BC} \\ &= \bar{A}+\overline{\bar{B}\bar{C}}\,\overline{BC} \\ &= \bar{A}+(B+C)(\bar{B}+\bar{C}) \\ &= \bar{A}+B\bar{C}+\bar{B}C\end{aligned}$$

use demorgans law

$B\bar{B}^{0} + B\bar{C} + C\bar{B} + C\bar{C}^{0}$

(d) $\overline{A}(A+B)+(B+AA)(A+\overline{B})$

$$\begin{aligned}\overline{A}(A+B)+(B+AA)(A+\overline{B} &= (\overline{A}A+\overline{A}B)+(B+AA)(A+\overline{B}) \\ &= \overline{A}B+(B+AA)(A+\overline{B}) \\ &= \overline{A}B+(B+A)(A+\overline{B}) \\ &= \overline{A}B+(BA+AA+B\bar{B}+A\overline{B}) \\ &= \overline{A}B+(BA+A+A\overline{B}) \\ &= \overline{A}B+A \\ &= A+B\end{aligned}$$

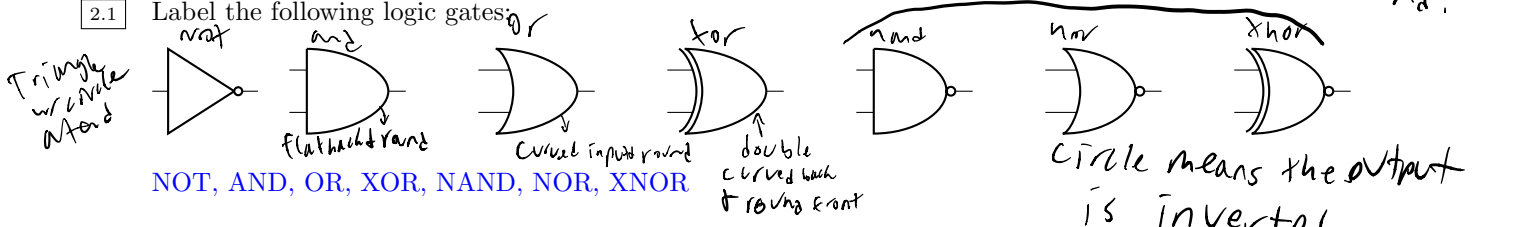
$A + A(B+\bar{B})$

$A+A=A$

If A true, it's true, if it's false, it's what B is thus we can say $A+B$

2 Logic Gates

2.1 Label the following logic gates:



Same shapes as and, or, xor but w/ circle at end.

Triangle w/ circle = not

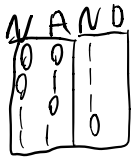
Circle means the output is inverted.

NOT, AND, OR, XOR, NAND, NOR, XNOR

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

(a) NAND

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



(b) XOR

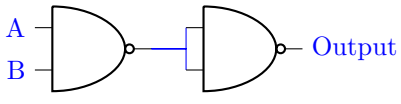
$\bar{A}B+A\bar{B}$

(c) XNOR

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

Nand can flip input if you put it to both terminals.

2.3 Create an AND gate using only NAND gates. (Hint: use 2.2!)



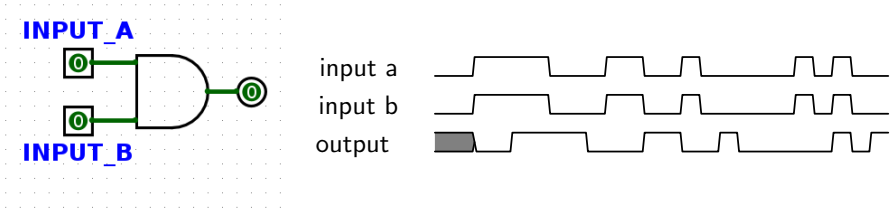
Want: AB from $\bar{A}\bar{B}+\bar{A}B+A\bar{B}$

NAND again

a	b	out
0	0	1 → 0
0	1	1 → 0
1	0	1 → 0
1	1	0 → 1

3 State Intro

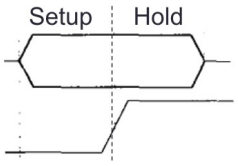
There are two basic types of circuits: combinational logic circuits and state elements. **Combinational logic** circuits simply change based on their inputs after whatever propagation delay is associated with them. For example, if an AND gate (pictured below) has an associated propagation delay of 2ps,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



You should notice that the output of this AND gate always changes 2ps after its inputs change. (aka sequential logic)

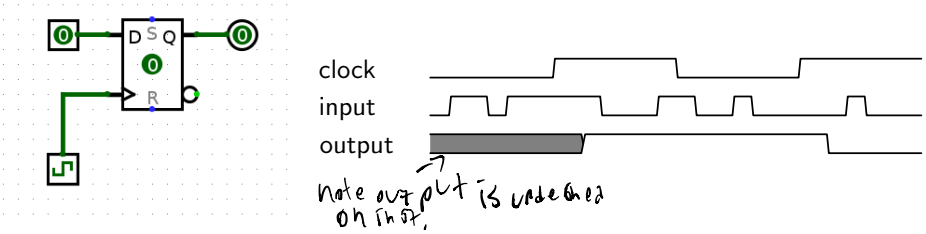
State elements, on the other hand, can *remember* their inputs even after the inputs change. State elements change value based on a clock signal. A rising edge-triggered register, for example, samples its input at the rising edge of the clock (when the clock signal goes from 0 to 1).

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 the 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 or equal to hold time. From another perspective, the fact that clk-to-q $\geq$ hold time makes sense since it only takes clk-to-q seconds to copy the value over, so there's no need to have the value fed into the register for any longer.

For the following register circuit, assume **setup** 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.



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~~ large 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.

3.1 For the following 2 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

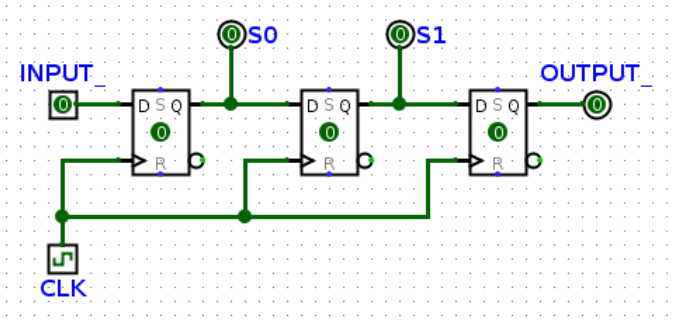
max delay = setup time + clk-to-Q + CL critical path delay

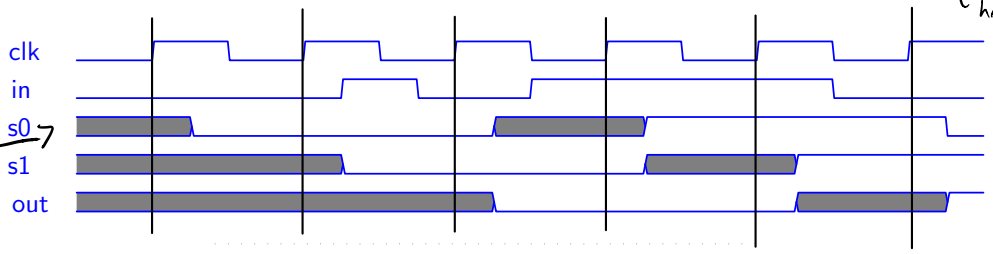
min period = max delay

max freq = $\frac{1}{\text{min period}}$

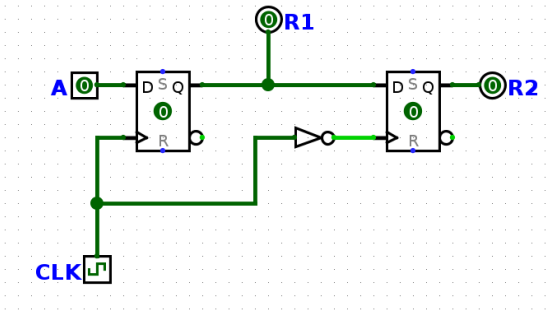
max clock frequency constraints

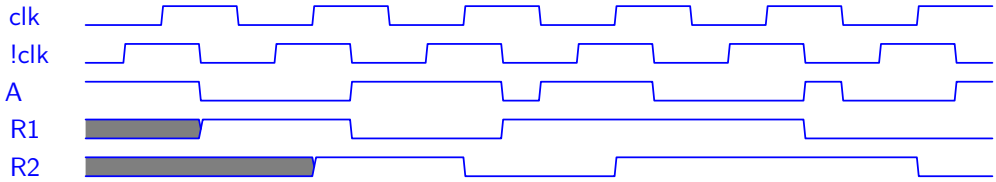
$t_{hold} \leq t_{input} \leq t_{clkperiod} - t_{setup}$



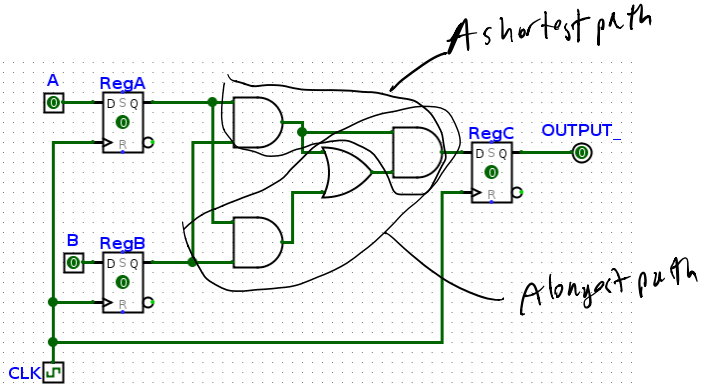


Not initialized until after first clk tick





3.2 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?



The maximum allowable hold time for RegC is how long it takes for RegC's input to change, so (clk-to-q of A or B) + shortest CL time = $4 + (5 + 5) = 14$ ns.

min → clk-to-q ↑ shortest path (and->and)

The minimum acceptable clock cycle time is clk-to-q + longest CL time + setup time = $4 + (5 + 5 + 5) + 6 = 25$ ns.

clk-to-q; longest path (and->or->and); ← setup time of logic

25 ns corresponds to a clock frequency of $(1/(25 * 10^{-9}))s^{-1} = 40MHz$

need to invert period to get frequency.