

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 | $A(\bar{A}) = 0$ | $A + \bar{A} = 1$ |
| Demorgan's | $\overline{AB} = \bar{A} + \bar{B}$ | $\overline{A + B} = \bar{A}(\bar{B})$ |

1.1 Simplify the following Boolean expressions:

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

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

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

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

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

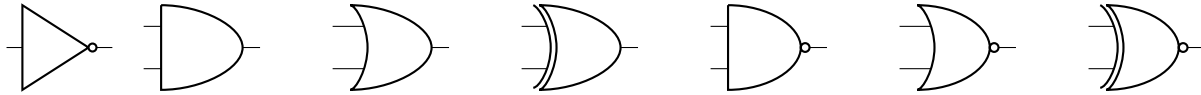
$$\begin{aligned}
 \overline{A(\bar{B}\bar{C} + BC)} &= \bar{A} + \overline{\bar{B}\bar{C} + BC} \\
 &= \bar{A} + \overline{\bar{B}\bar{C}}\overline{BC} \\
 &= \bar{A} + (B + C)(\bar{B} + \bar{C}) \\
 &= \bar{A} + B\bar{C} + \bar{B}C
 \end{aligned}$$

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

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

2 Logic Gates

2.1 Label the following logic gates:



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

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

(a) NAND

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

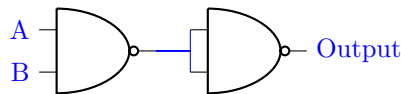
(b) XOR

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

(c) XNOR

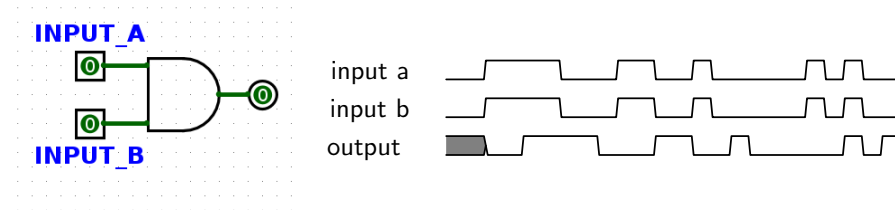
$$\bar{A}\bar{B} + AB$$

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



3 State Intro

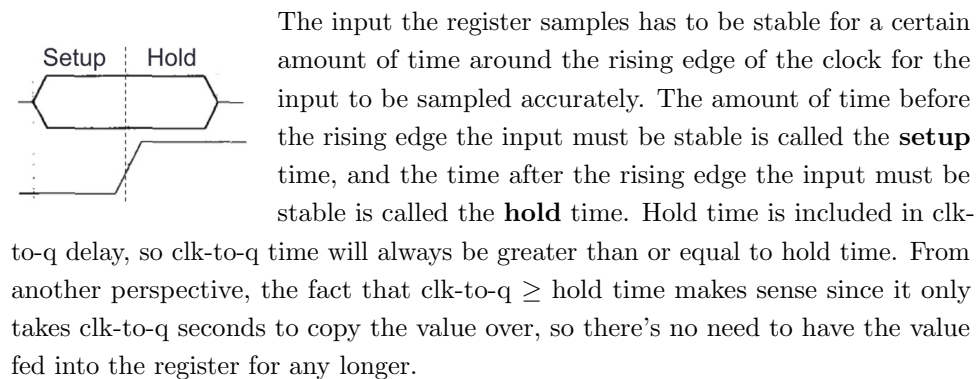
There are two basic types of circuits: combinational logic circuits and state elements. **Combinational logic** circuits simply change based on their inputs after whatever propagation delay is associated with them. For example, if an AND gate (pictured below) has an associated propagation delay of 2ps,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



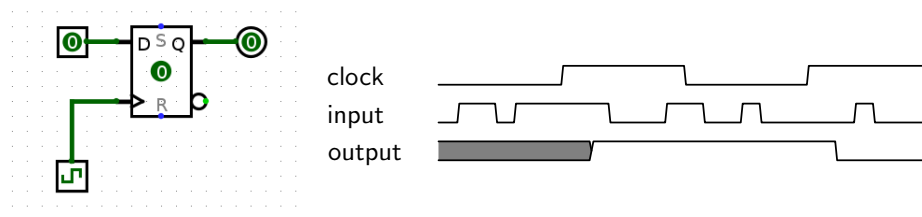
You should notice that the output of this AND gate always changes 2ps after its inputs change.

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.



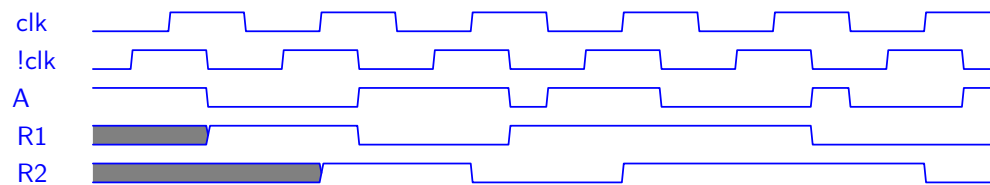
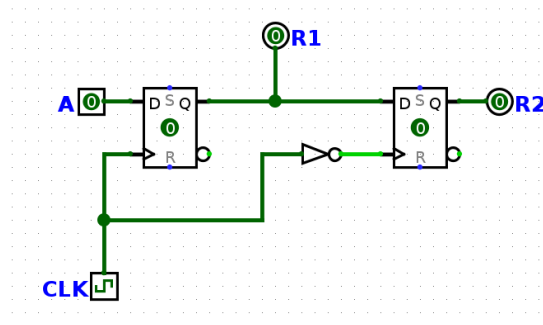
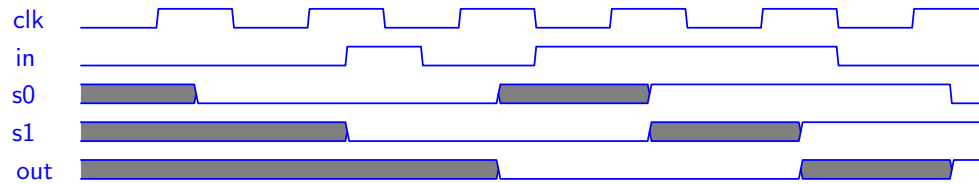
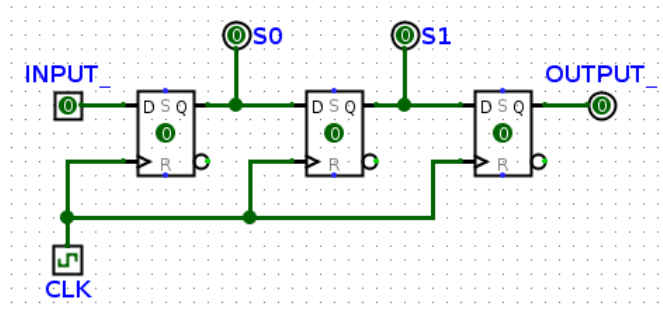
For the following register circuit, assume **setup** of 2.5ps, **hold** time of 1.5ps, and a **clk-to-q** time of 1.5ps. The clock signal has a period of 13ps.



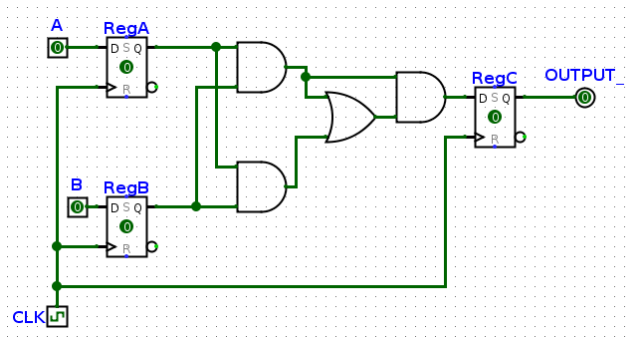
You'll notice that the value of the output in the diagram above doesn't change immediately after the rising edge of the clock. Clock cycle time must be small enough that inputs to registers don't change within the hold time and large enough

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



- 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.

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

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