CS 61C
Fall 2022

SDS and FSMs
Discussion 6

1 Pre-Check

This section is designed as a conceptual check for you to determine if you conceptually understand and have any misconceptions about this topic. Please answer true/false to the following questions, and include an explanation:

1.1 Simplifying boolean logic expressions will not affect the performance of the hardware implementation.

1.2 The fewer logic gates, the faster the circuit (assuming each gate has the same propagation delays).

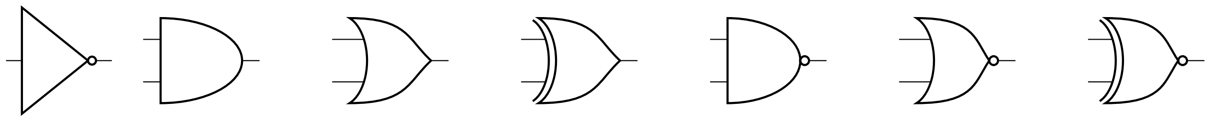
1.3 The time it takes for clock-to-q and register setup can be greater than one clock cycle.

1.4 Every possible combinational logic circuit can be expressed by some combination of NOR gates.

1.5 The shortest combinational logic path between two state elements is useful in determining circuit frequency and minimum clock cycle.

2 Logic Gates

2.1 Label the following logic gates:



2.2 Convert the following to simplified boolean expressions on input signals A and B. Remember that simplified boolean expressions should only have NOT, AND, and OR primitives ($\overline{A}$, $\times$, and $+$ respectively):

(a) NAND

(b) XOR

(c) XNOR

2.3 Create an AND gate using only NAND gates.

3 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	$\mathrm{AB} = \mathrm{BA}$	$\mathrm{A} + \mathrm{B} = \mathrm{B} + \mathrm{A}$
Associative	$\mathrm{AB(C)} = \mathrm{A(BC)}$	$\mathrm{A} + (\mathrm{B} + \mathrm{C}) = (\mathrm{A} + \mathrm{B}) + \mathrm{C}$
Identity	$1\mathrm{A} = \mathrm{A}$	$0 + \mathrm{A} = \mathrm{A}$
Null	$0\mathrm{A} = 0$	$1 + \mathrm{A} = 1$
Absorption	$\mathrm{A}(\mathrm{A} + \mathrm{B}) = \mathrm{A}$	$\mathrm{A} + \mathrm{AB} = \mathrm{A}$
Distributive	$(\mathrm{A} + \mathrm{B})(\mathrm{A} + \mathrm{C}) = \mathrm{A} + \mathrm{BC}$	$\mathrm{A}(\mathrm{B} + \mathrm{C}) = \mathrm{AB} + \mathrm{AC}$
Idempotent	$\mathrm{A(A)} = \mathrm{A}$	$\mathrm{A} + \mathrm{A} = \mathrm{A}$
Inverse	$\mathrm{A}(\overline{\mathrm{A}}) = 0$	$\mathrm{A} + \overline{\mathrm{A}} = 1$
De Morgan's	$\overline{\mathrm{AB}} = \overline{\mathrm{A}} + \overline{\mathrm{B}}$	$\overline{\mathrm{A} + \mathrm{B}} = \overline{\mathrm{A}}(\overline{\mathrm{B}})$

$\boxed{3.1}$ Use multiple iterations of De Morgan's laws to prove the identity $\bar{A} + AB = \bar{A} + B$.

$\boxed{3.2}$ Prove that De Morgan's law can be generalized for the complement of any number of terms.

$\boxed{3.3}$ Simplify the following Boolean expressions:

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

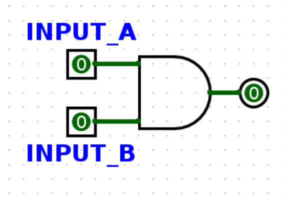
(b) $\bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + AB\bar{C} + A\bar{B}\bar{C} + ABC + A\bar{B}C$

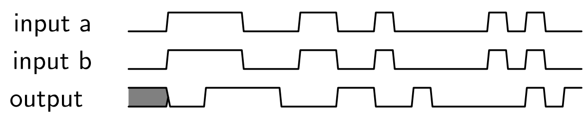
(c) $\overline{A(\bar{B}\bar{C} + BC)}$

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

4 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:

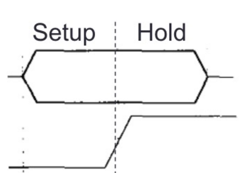




You should notice that the output of this AND gate *always* changes 2ps after its inputs change.

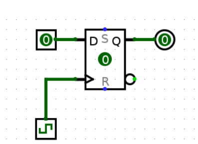
State elements, on the other hand, can *remember* their inputs even after the inputs change. State elements change value based on a clock signal. A rising edge-triggered register, for example, samples its input at the rising edge of the clock (when the clock signal goes from 0 to 1).

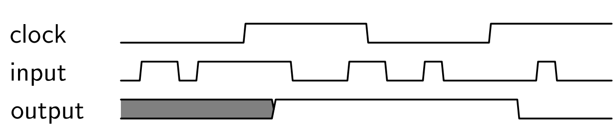
Like logic gates, registers also have a delay associated with them before their output will reflect the input that was sampled. This is called the **clk-to-q** delay. ("Q" often indicates output). This is the time between the rising edge of the clock signal and the time the register's output reflects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



The input to 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 generally included in clk-to-q delay, so clk-to-q time will usually be greater than or equal to hold time. Logically, 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.

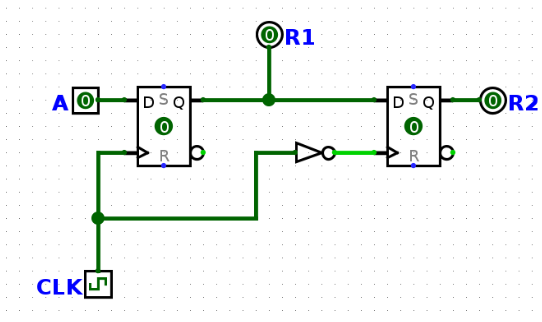
For the following register circuit, assume **setup** of 2.5ps, **hold** time of 1.5ps, and a **clk-to-q** time of 1.5ps. The clock signal has a period of 13ps.

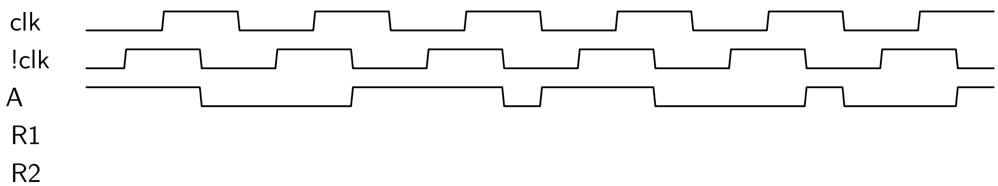


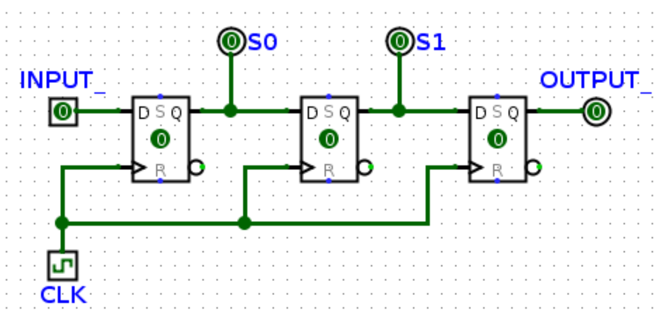


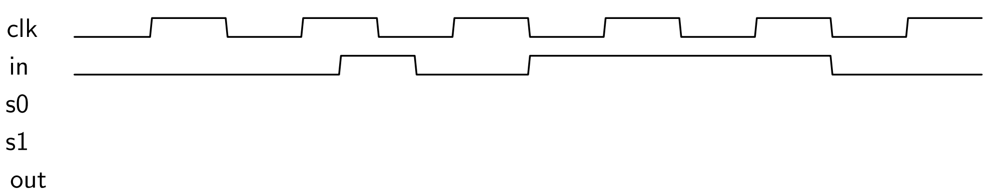
You'll notice that the value of the output in the diagram above doesn't change immediately after the rising edge of the clock. Until enough time has passed for the output to reflect the input, the value held by the output is garbage; this is represented by the shaded gray part of the output graph. Clock 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.

4.1 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, which is already accounted for in the !clk signal given.

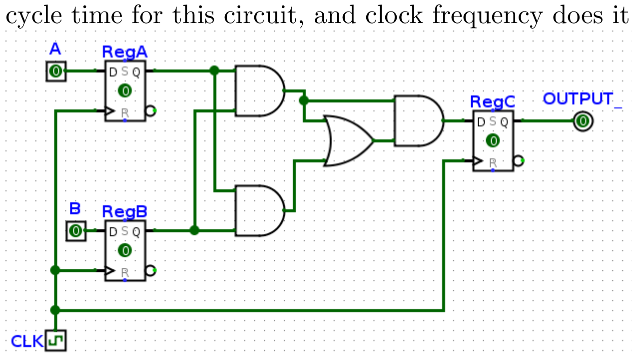








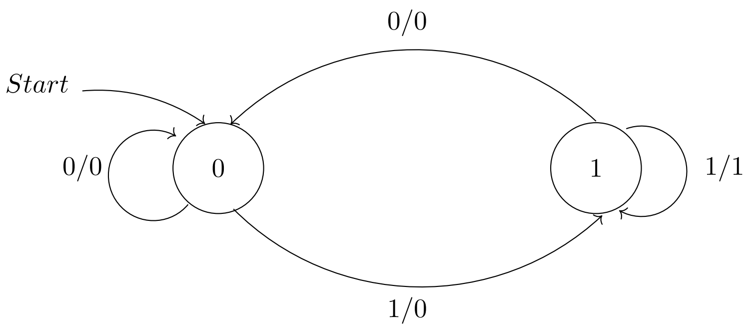
4.2 In the circuit below, RegA and RegB have setup, hold, and clk-to-q times of 4ns, all logic gates have a delay of 5ns, and RegC has a setup time of 6ns. What is the maximum allowable hold time for RegC? What is the minimum acceptable clock cycle time for this circuit, and clock frequency does it correspond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



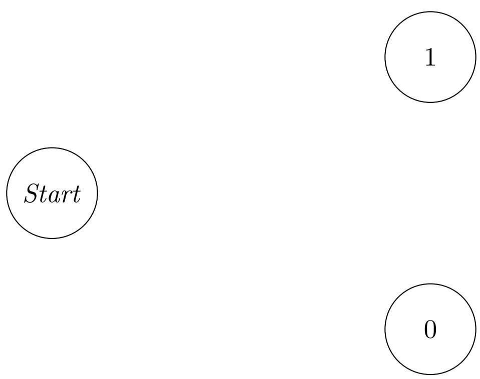
5 Finite State Machines

Automatons are machines that receive input and use various states to produce output. A finite state machine is a type of simple automaton where the next state and output depend only on the current state and input. Each state is represented by a circle, and every proper finite state machine has a starting state, signified either with the label "Start" or a single arrow leading into it. Each transition between states is labeled [input]/[output].

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



5.2 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.



5.3 Draw an FSM that will output a 1 if it recognizes the regex pattern {10+1}. (That is, if the input forms a pattern of a 1, followed by one or more 0s, followed by a 1.)