

## Combinational Logic

a) How many different two-input logic gates are possible? How many n-input logic gates?  
Hint: Think about the truth table.

For each gate, the truth table has  $2^n$  rows (one for each possible combination of input bits).

Each of these rows can be either 0 or 1, so we have a total of  $2^{(2^n)}$  total n-input logic gates.

b) Create an inverter (NOT gate) using only NAND gates (Hint: look at the truth tables for each).  
Feed the same input into both NAND input terminals.

c) Minimize the following boolean expressions and draw out logic circuits for them:

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

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

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

First:  $(AA + AB + AB' + BB')C = (AA + A + 1)C = AC$

Second:  $A'C'(B'+B) + AB(C'+C) + AB'(C'+C) = A'C' + AB + AB' = A'C' + A = A + C'$

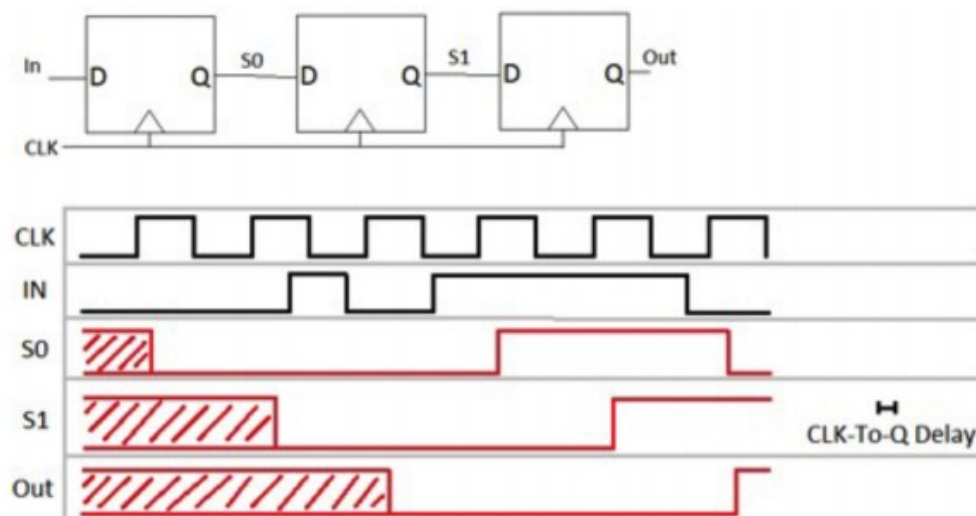
Third: Can't simplify but it's A xor B

## State Elements

State elements provide a means of storing values, and controlling the flow of information in the circuit. One example is a D-type flip-flop shown below connected in series.

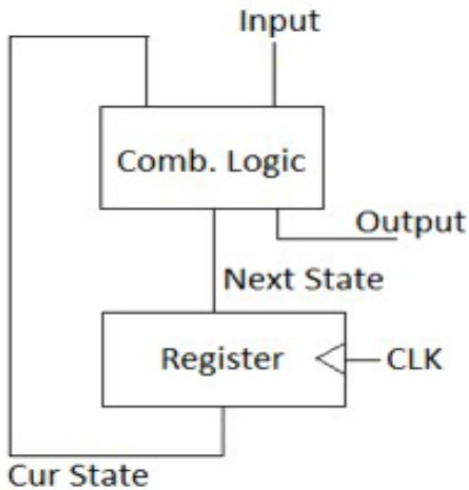
D is a single bit input, Q is a single bit output. On the rising edge of the clock, the value from D is copied to Q, after a small delay, known as the clk-to-q delay. To ensure the input value is read correctly, the input must be stable during the setup time before the edge, and during the hold time after the edge of the clock.

Fill out the timing diagrams for the circuit below.



## Finite State Machines

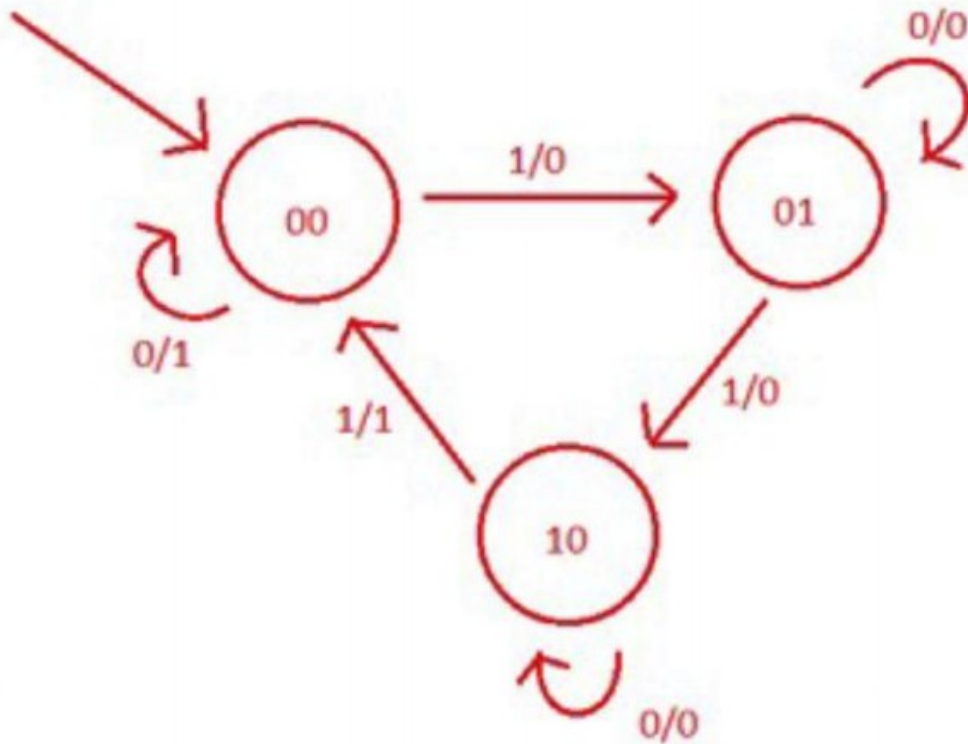
FSMs can be useful computational models for logic circuits (as well as programs). There are a finite number of states the machine can be in and at each point in time, it is at a single state. The next state and output only depends on the current state and input value. Here's how FSMs can be implemented in hardware:



The register holds the current state (encoded as a particular combination of bits), and the combinational logic block is like a function that maps {current state, input} to {next state, output}.

## Exercises

Draw a transition diagram for an FSM that can take in an input sequence one bit at a time, and after each input is received, output whether the number of 1s is divisible by 3. Write out the truth table that the combinational logic block must implement.



The states each correspond to the number of 1s seen so far, mod 3. When this quantity is 0, 1s seen so far is divisible by 3, and we output 1.

cur. state	input	next	output
00	0	00	1
01	0	01	0
10	0	10	0
00	1	01	0
01	1	10	0
10	1	00	1

Behavior for current state 11 is undefined, since we don't expect our machine to ever reach that state

Say you convert the FSM into a circuit like the diagram above. Given a setup time of  $s$ , a hold time of  $h$ , and a clk-to-q delay of  $cq$ , and a clock frequency of  $f$ , what is the most time,  $cl$ , the combinational logic can take?

Clock period can't be less than  $s + cq + cl$  so  $1/f - s - cq$ .