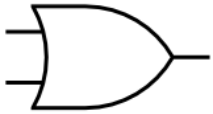
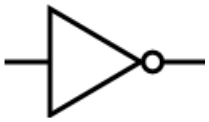
## Logic Gates

Fill in the following truth tables:

| a b | a•b |
|---|---|
| 00 | 0 |
| 01 | 0 |
| 10 | 0 |
| 11 | 1 |

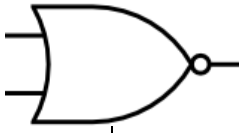


| a b | a+b |
|---|---|
| 00 | 0 |
| 01 | 1 |
| 10 | 1 |
| 11 | 1 |

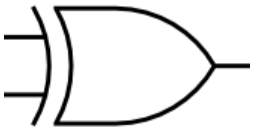


| a | ¬a |
|---|---|
| 0 | 1 |
| 1 | 0 |

| a b | ¬(a•b) |
|---|---|
| 00 | 1 |
| 01 | 1 |
| 10 | 1 |
| 11 | 0 |



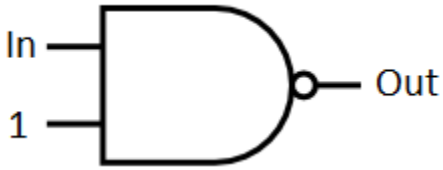
| a b | ¬(a+b) |
|---|---|
| 00 | 1 |
| 01 | 0 |
| 10 | 0 |
| 11 | 0 |



| a b | a⊕b |
|---|---|
| 00 | 0 |
| 01 | 1 |
| 10 | 1 |
| 11 | 0 |

Create an inverter (NOT gate) using only NAND gates (Hint: look at the truth tables for each).

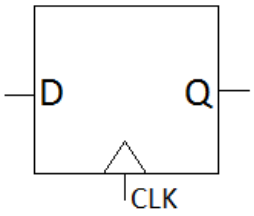

or




with the left solution based on the top and bottom rows of the NAND truth table, and the right one based on the bottom two rows.

## State Elements

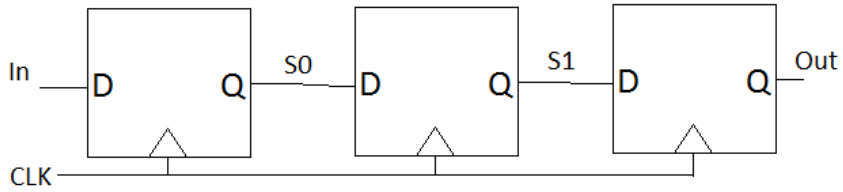
One of the basic building blocks of SDS. State elements provide a means of storing values, and controlling the flow of information in the circuit. The most basic state element (we're concerned with) is a DQ Flip-Flop:

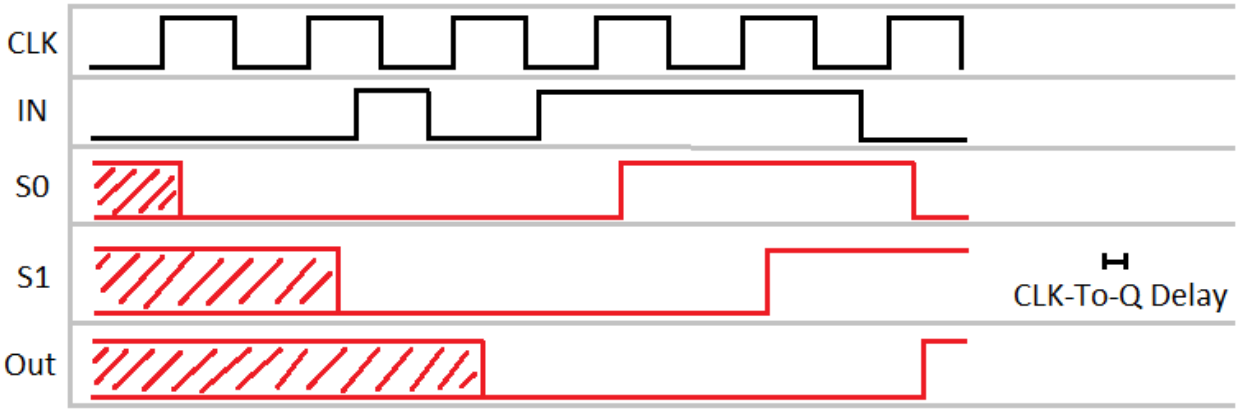


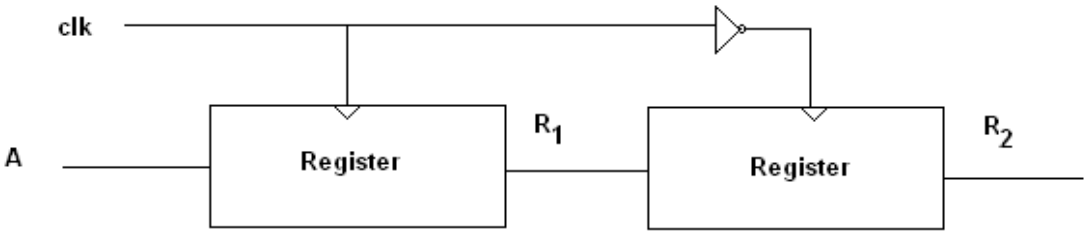

D is a single bit input, Q is a single bit output. On the **rising edge** of the clock, the value from D is copied to Q, after a small delay (known as the Clk-to-Q delay). At all other times, Q presents the value last copied and ignores D.

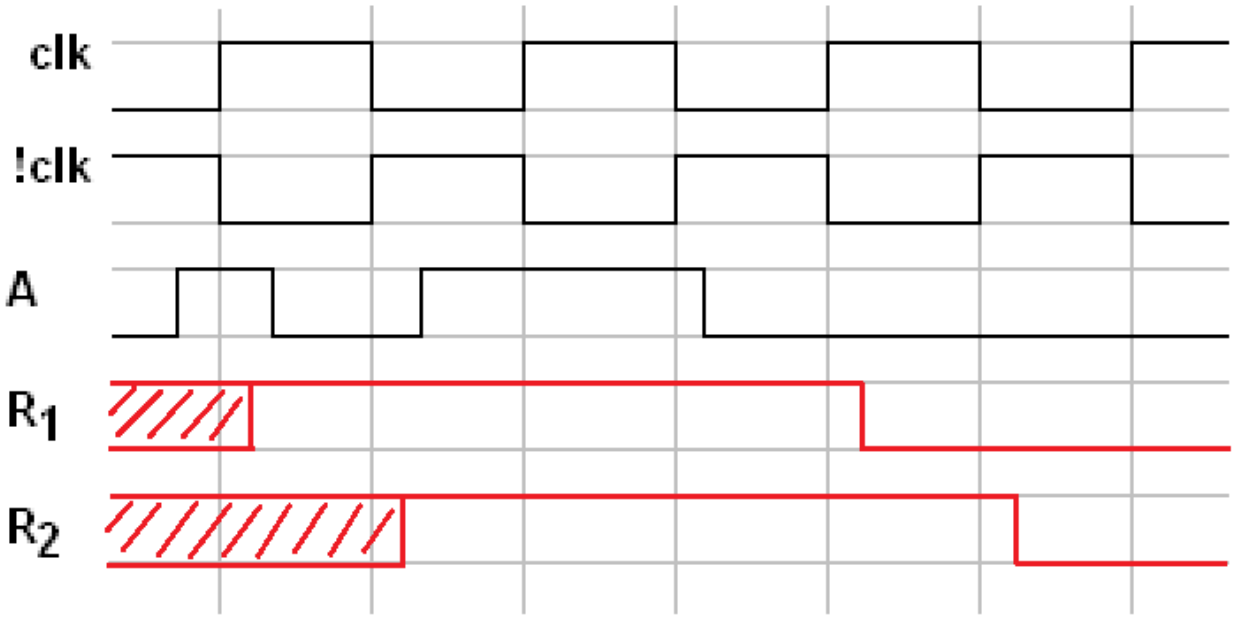
An n-bit register is just n DQ Flip-Flops aligned in parallel, tied to the same clock.

**Timing Diagrams:** Fill out the timing diagrams for the circuits below.
An unknown value is "shaded."



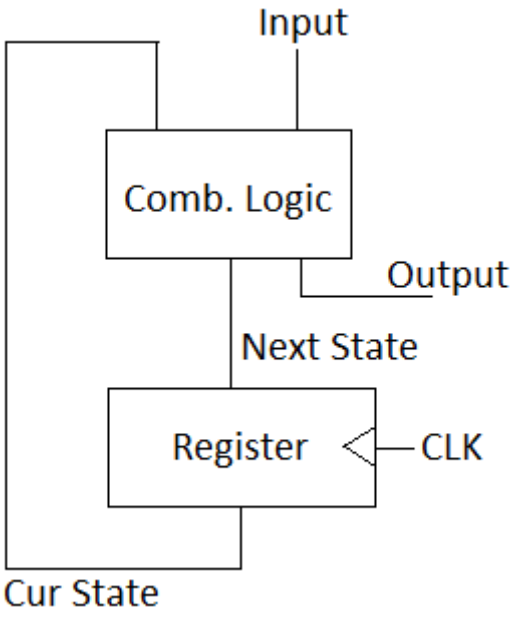






**Finite State Machines**

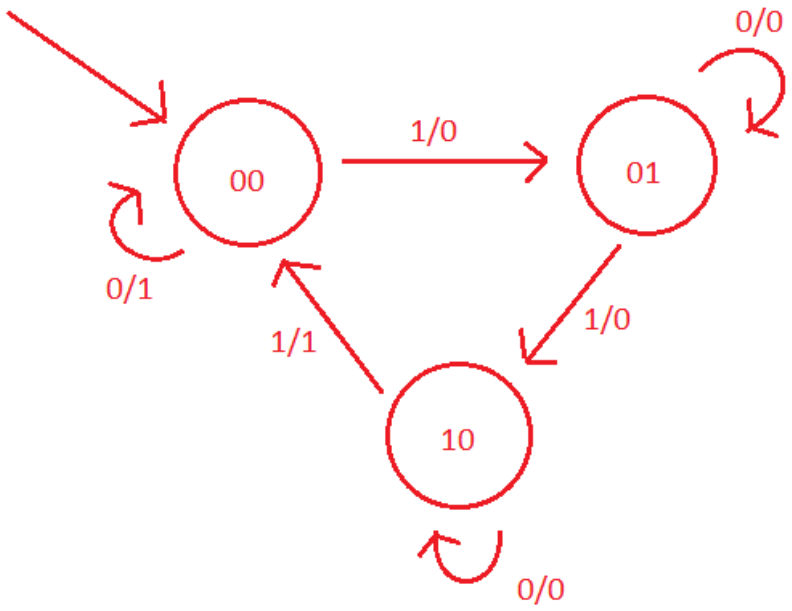
FSMs can be an incredibly useful computational tool. They have a straightforward implementation in hardware:




The register holds the current state (encoded as a particular combination of bits), and the combinational logic block maps from {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 state | output |
|---|---|---|---|
| 00 | 0 | 00 | 1 |
| 00 | 1 | 01 | 0 |
| 01 | 0 | 01 | 0 |
| 01 | 1 | 10 | 0 |
| 10 | 0 | 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.