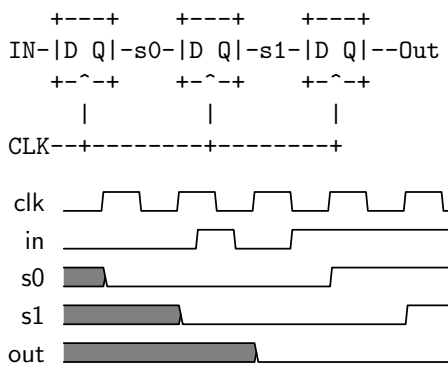
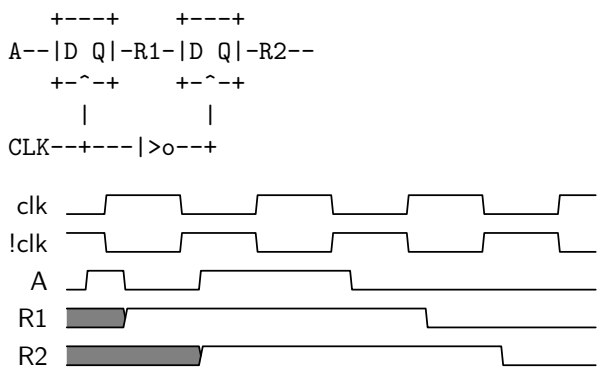
State

1. Fill out the timing diagram for the circuit below:

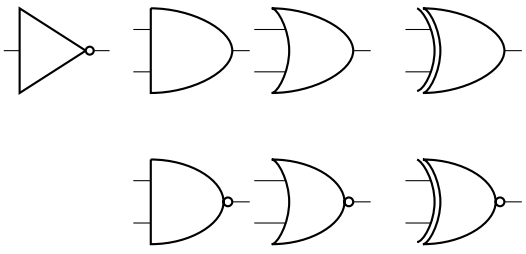


2. Fill out the timing diagram for the circuit below:



Logic Gates

1. Label the following logic gates:



Solution: not, and, or, xor, nand, nor, xnor

2. Convert the following to boolean expressions:
   (a) NAND

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

(b) XOR

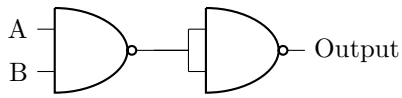
Solution: $\bar{A}B + A\bar{B}$

(c) XNOR

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

3. Create an AND gate using only NAND gates.

Solution:



4. How many different two-input logic gates can there be? How many n-input logic gates?

Solution: A truth table with n inputs has 2^n rows. Each logic gate has a 0 or a 1 at each of these rows. Imagining a function as a 2^n-bit number, we count 2^{2^n} total functions, or 16 in the case of n = 2.

Boolean Logic

$1 + A = 1$	$A + \bar{A} = 1$	$A + AB = A$	$(A + B)(A + C) = A + BC$
$0B = 0$	$B\bar{B} = 0$	$A + \bar{A}B = A + B$	
DeMorgan's Law:	$\overline{AB} = \bar{A} + \bar{B}$	$\overline{A + B} = \bar{A}\bar{B}$	

1. Minimize the following boolean expressions:

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

Solution:

$$(AA + A\bar{B} + AB + B\bar{B})C = (A + A(\bar{B} + B))C = AC \quad (1)$$

(b) Grouping & Extra Terms: $\bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + AB\bar{C} + A\bar{B}\bar{C} + ABC + A\bar{B}C$

Solution:

$$\begin{aligned} \bar{A}\bar{C}(\bar{B} + B) + A\bar{C}(B + \bar{B}) + AC(B + \bar{B}) &= \bar{A}\bar{C} + A\bar{C} + AC && (2)\\ &= \bar{A}\bar{C} + A\bar{C} + A\bar{C} + AC && (3)\\ &= (\bar{A} + A)\bar{C} + A(\bar{C} + C) && (4)\\ &= A + \bar{C} && (5) \end{aligned}$$

(c) DeMorgan's: $\overline{A(\bar{B}\bar{C} + BC)}$

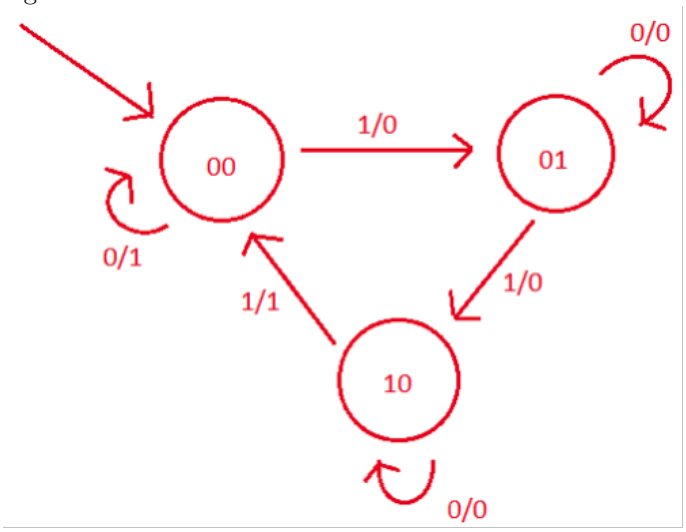
Solution:

$$\begin{aligned}
\overline{A(\bar{B}\bar{C} + BC)} &= \bar{A} + \overline{\bar{B}\bar{C} + BC} && (6)\\
&= \bar{A} + \overline{\bar{B}\bar{C}}\,\overline{BC} && (7)\\
&= \bar{A} + (B + C)(\bar{B} + \bar{C}) && (8)\\
&= A + B\bar{C} + \bar{B}C && (9)
\end{aligned}$$

Finite State Machine

1. 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 (remember to assign each state a binary encoding). Finally, write the Boolean algebra expressions that implement the FSMs truth table.

Solution: 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.
We assign our three states the encodings 00, 01, 10. Note that these encodings are completely arbitrary but other encodings (i.e. 01, 10, 11) will lead to a different truth table and final combinational logic.



Behavior for current state 11 is undefined since we dont expect our machine to ever reach that state. We represent this in the truth table above using an X to stand for dont care. We can use this to our advantage by choosing values (0 or 1 for each X) that will simplify our combinational logic.
For example, we can assume:
$NS[1] = 1$ for both $CS[1] * CS[0]$,
$NS[0] = 1$ for $CS[1] * CS[0] * \bar{I}$,
$NS[0] = 0$ for $CS[1] * CS[0] * I$,
and $Out = 0 for CS[1] * CS[0] * \bar{I}$

and $Out = 1 for CS[1] * CS[0] * I$.

$$NS[1] = CS[0] * I + CS[1] * \bar{I} \tag{10}$$
$$NS[0] = C\bar{S}[1] * C\bar{S}[0] * I + CS[0] * \bar{I} \tag{11}$$
$$Out = C\bar{S}[1] * C\bar{S}[0] * \bar{I} + CS[1] * I \tag{12}$$