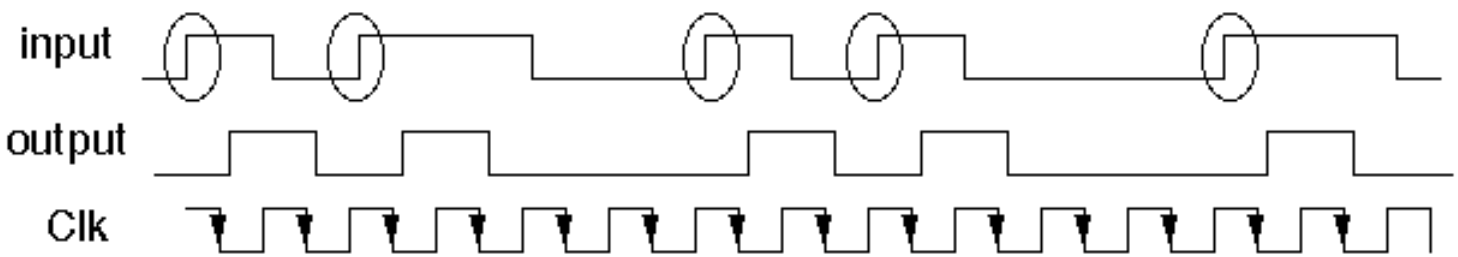
Some Simple Examples

1. We already did the counter. It's a FSM whose outputs and state bits are one and the same!

2. The Edge Finder:

   We'd like to build a circuit that outputs a "1" each time it sees a 0-to-1 transition on its input



Interpretation of the clock input "Clk":

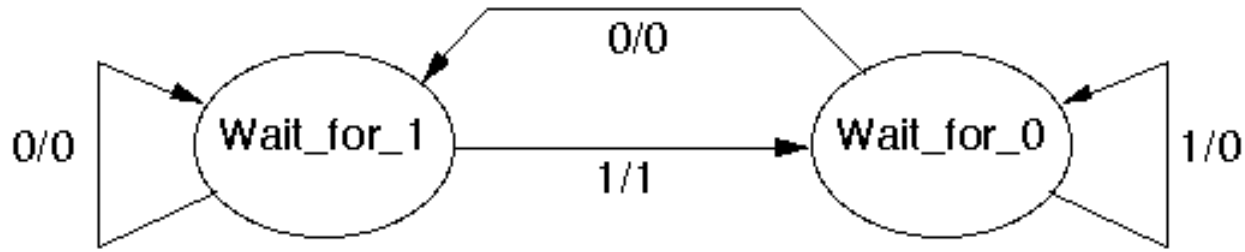
On each 'active' (falling) edge, the circuit examines "input" and figures out what to put on "output".

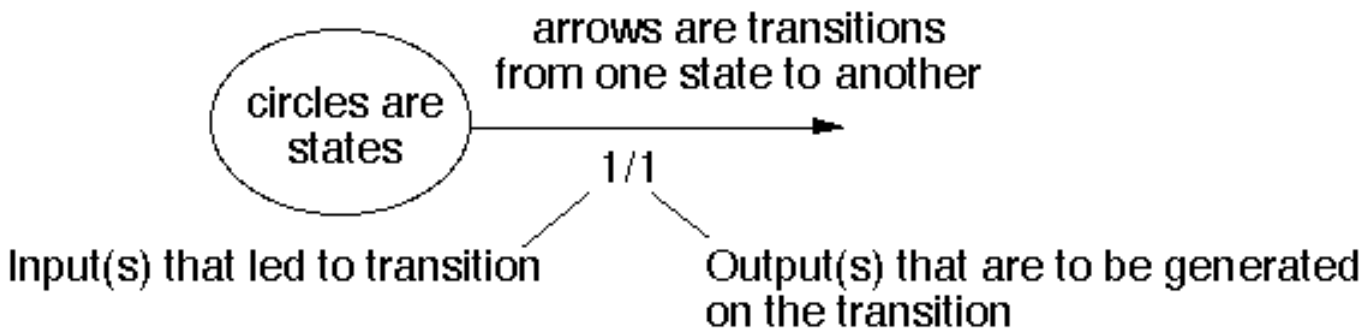
State Transition Graph

English description of operation:

a. If input = 0 on previous cycle, and it's still 0, output=0.
b. If input = 0 on previous cycle and now it's a 1, output=1.
c. If previous input =1 and it's still = 1, output = 0.
d. If previous input = 1 and now it's = 0, output = 0.

Can be economically captured in graphical form:

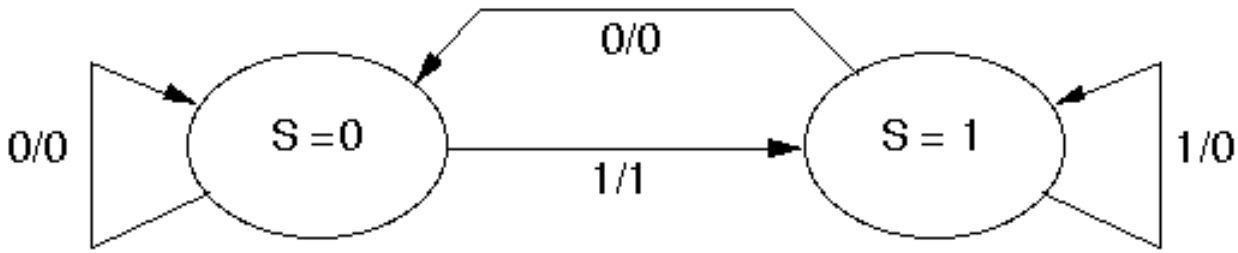




Assignment of State Bits

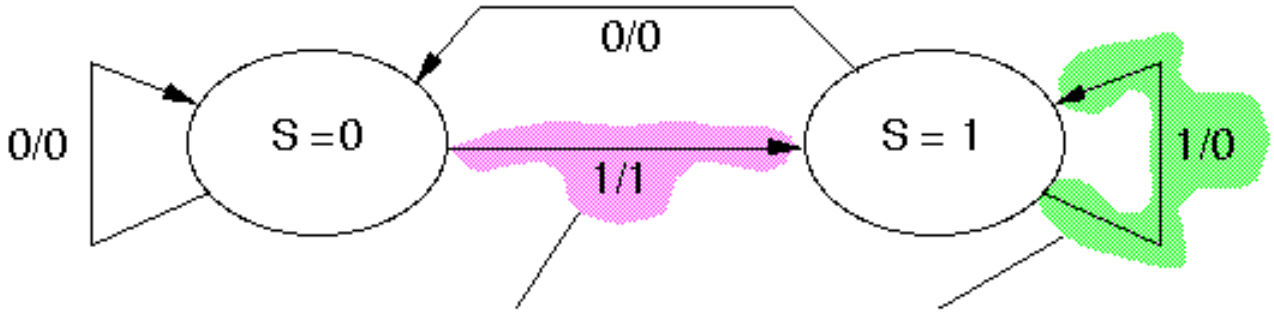
Only 2 state, so only 1 state bit is needed. State assignment is trivial:

State:	State bit S:
Wait_for_1	0
Wait_for_0	1



If there are more than 2 states, things get a bit more interesting; the encoding of the state can have a large effect on the amount of logic circuitry needed to implement the FSM!

Logic Design



From the transition graph, we can write logic equations for the "1s" for S and Output:

Next_S = !S•Input + S•Input

For each arrow leading to the state where S=1, find the previous state and input that got us there!

Output = !S•Input

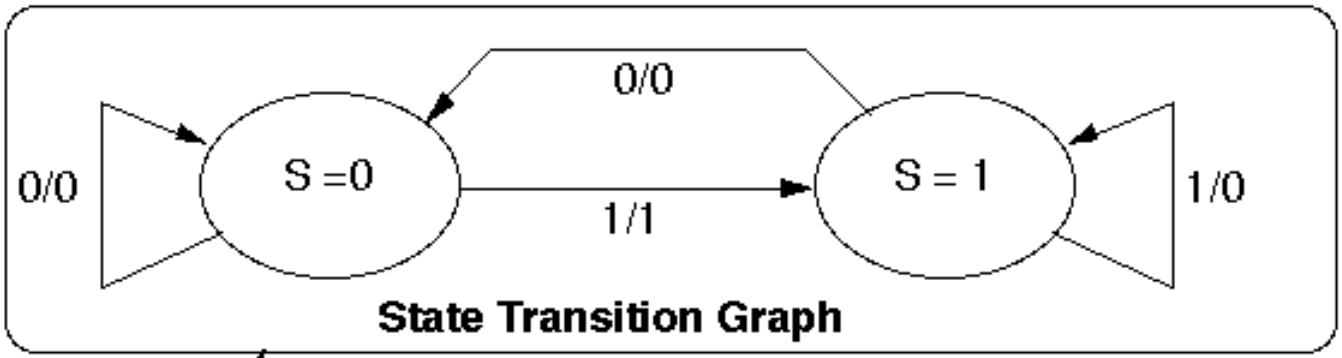
For each arrow in which Output = 1, find the previous state and input that produced that arrow (transition)

Next, perform logic simplification (by inspection, Karnaugh maps, or other techniques):

Next_S = !S•Input + S•Input = Input

Output = !S•Input

Logic Equations to Gates

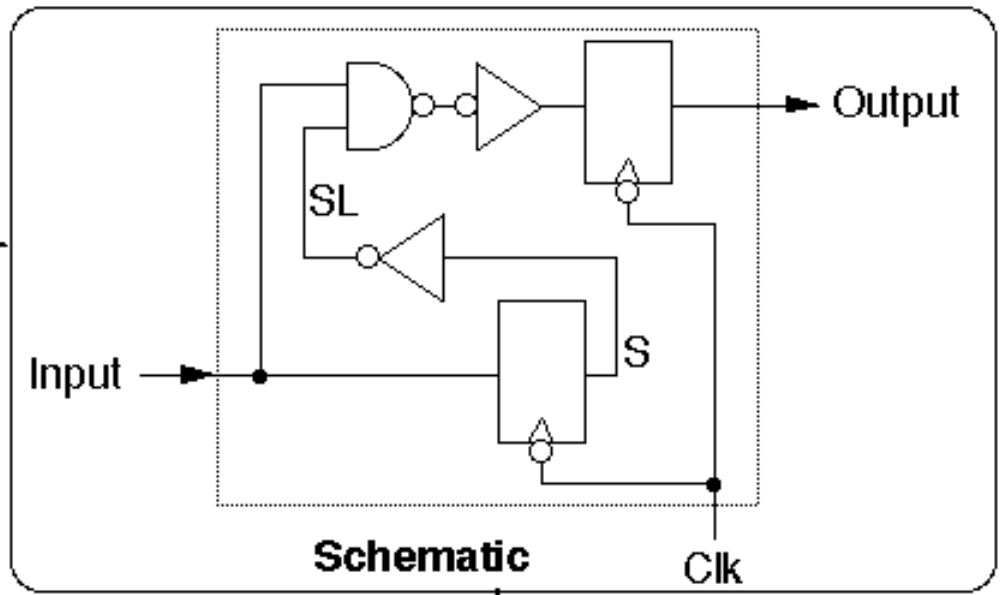


State Transition Graph

Next_S = Input

Output = !S•Input

Logic Equations
(e.g., reduced SOP form)



Schematic

Layout

Notation for Logical Inverse:

$!S = \overline{S} = SL$