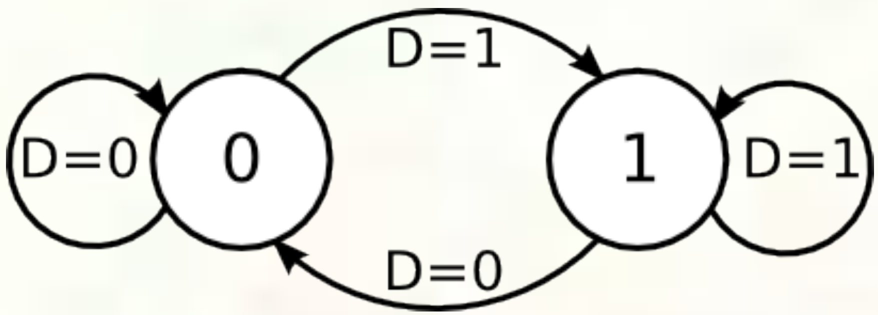
State Machines

Another look at D latch/flip-flop



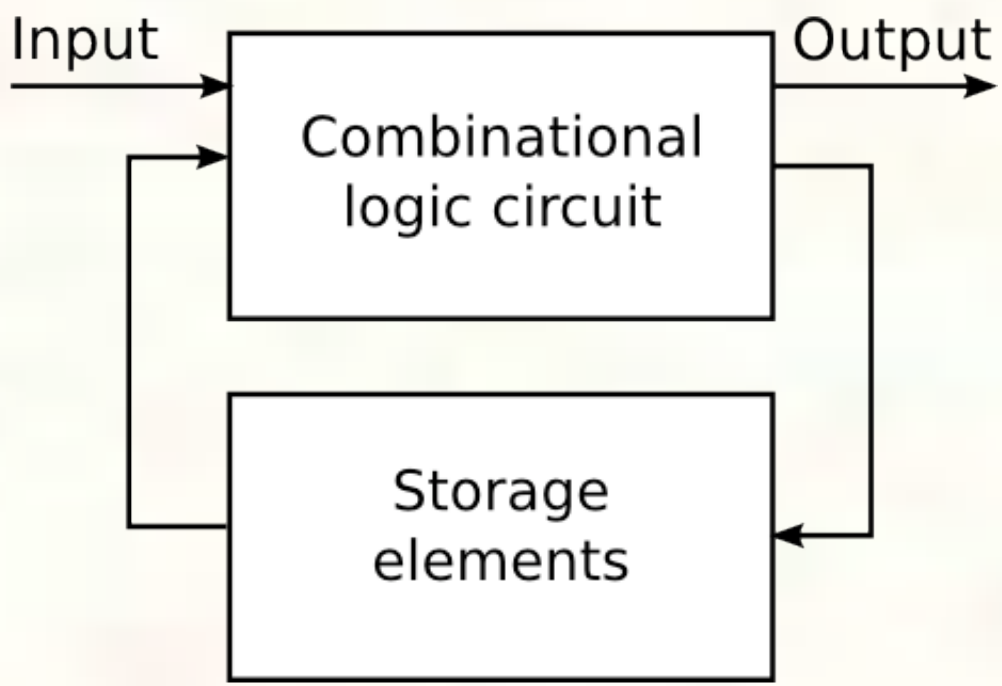
q_{old}	D	q_{new}
0	0	0
0	1	1
1	0	0
1	1	1

This is an example of a **state diagram**
more specifically a **Moore** machine

$q_{new} = D$

Synchronous state machines

If a system can both process and store information, then the values stored in the memory elements depend on both the inputs and the previous values in these elements. This is called a **sequential** system.



Such a system is also called a **finite-state machine (FSM).**

If all changes to memory values happen at the same time as determined by a global system clock, we have a **synchronous FSM**.

FSM definition

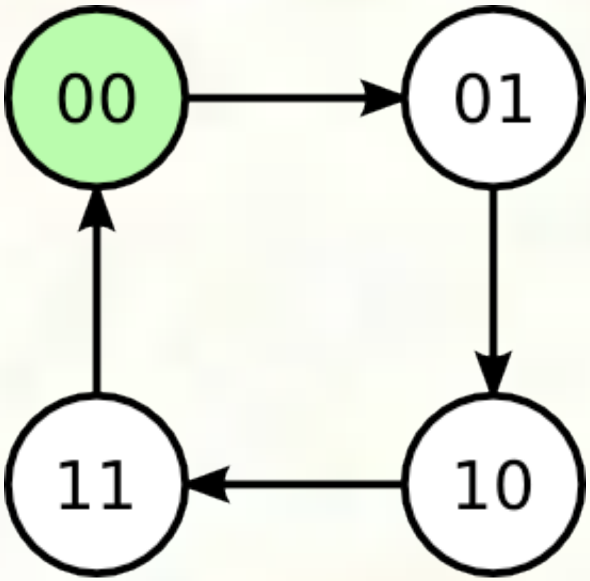
An FSM has the following components:

- A set of states
- A set of inputs
- A set of outputs
- A state-transition function (of the states and inputs)
- An output function (of the states and maybe inputs)
  - Moore machine - function of states only
  - Mealy machine - function of states and inputs

This can be represented by a **state diagram**

- States are circles
- Arcs show the state transition function
- Arcs are labeled with input values
- Outputs are labels on states (Moore) or arcs (Mealy)

Another example - 2-bit counter



Counter starts at 0 (green) and increments each time the clock cycles, until it gets to 3 and then overflows back to 0.

Only input is the clock, we don't show that.

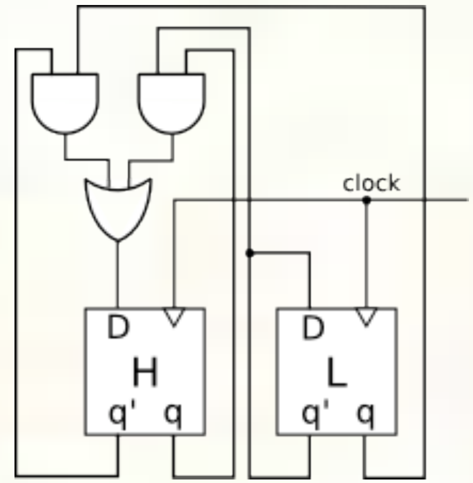
H_{old}	L_{old}	H_{new}	L_{new}
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

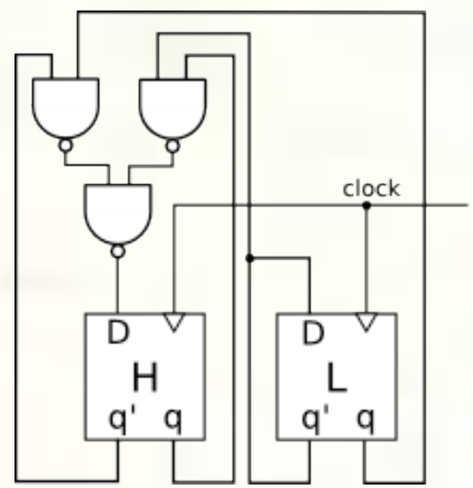
2-bit counter

H_{old}	L_{old}	H_{new}	L_{new}
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

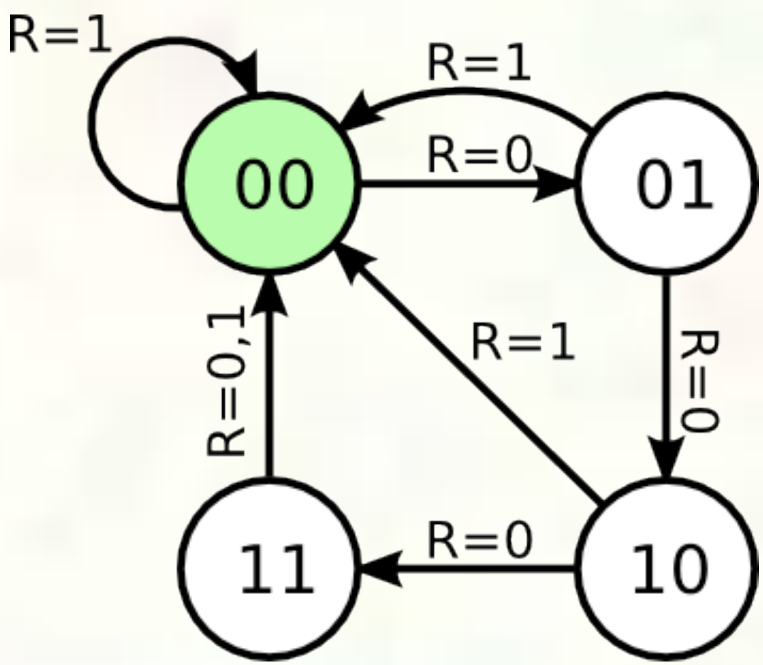
$$L_{new} = H_{old}'L_{old}' + H_{old}L_{old}' = L_{old}'$$

$$H_{new} = H_{old}'L_{old} + H_{old}L_{old}'$$





2-bit counter with reset

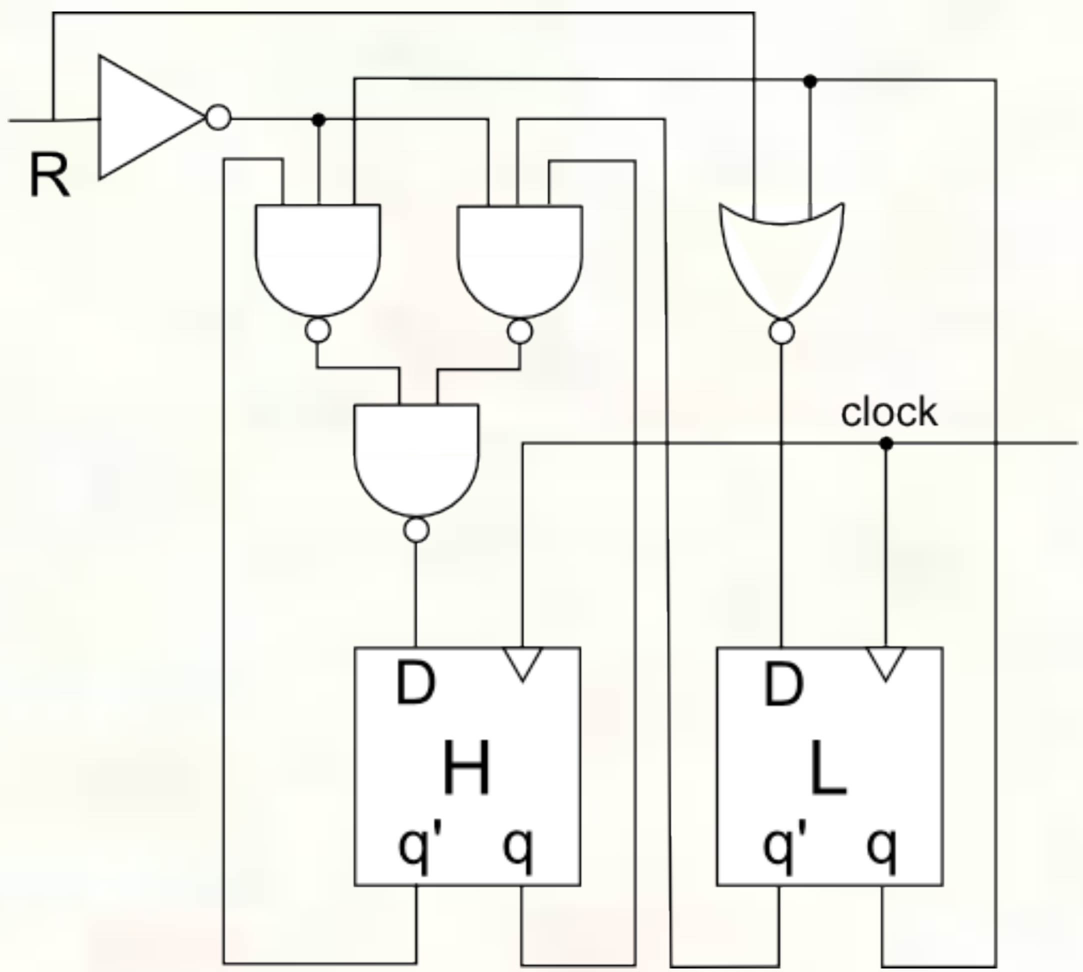


R	H_{old}	L_{old}	H_{new}	L_{new}
0	0	0	0	1
0	0	1	1	0
0	1	0	1	1
0	1	1	0	0
1	x	x	0	0

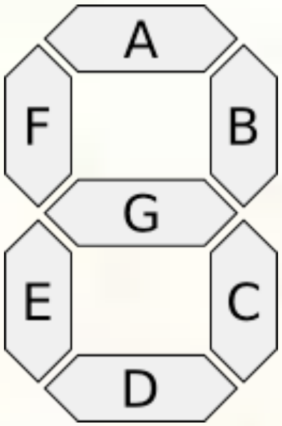
$$L_{new} = R'H_{old}'L_{old}' + R'H_{old}L_{old}' = R'L_{old}' = (R + L_{old})'$$

$$H_{new} = R'H_{old}'L_{old} + R'H_{old}L_{old}' = R'(H_{old}'L_{old} + H_{old}L_{old}')$$

2-bit counter with reset

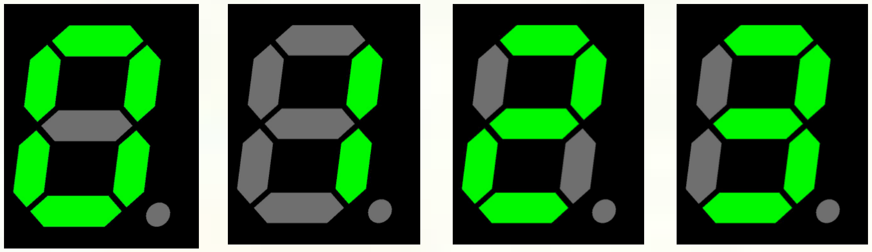


Counter with 7-segment display



Each segment in the display can be lit independently to allow all 10 decimal digits to be displayed (also hex)

2-bit counter will need to display digits 0-3, so will output a 1 for each segment to be lit for a given state



Counter with output functions

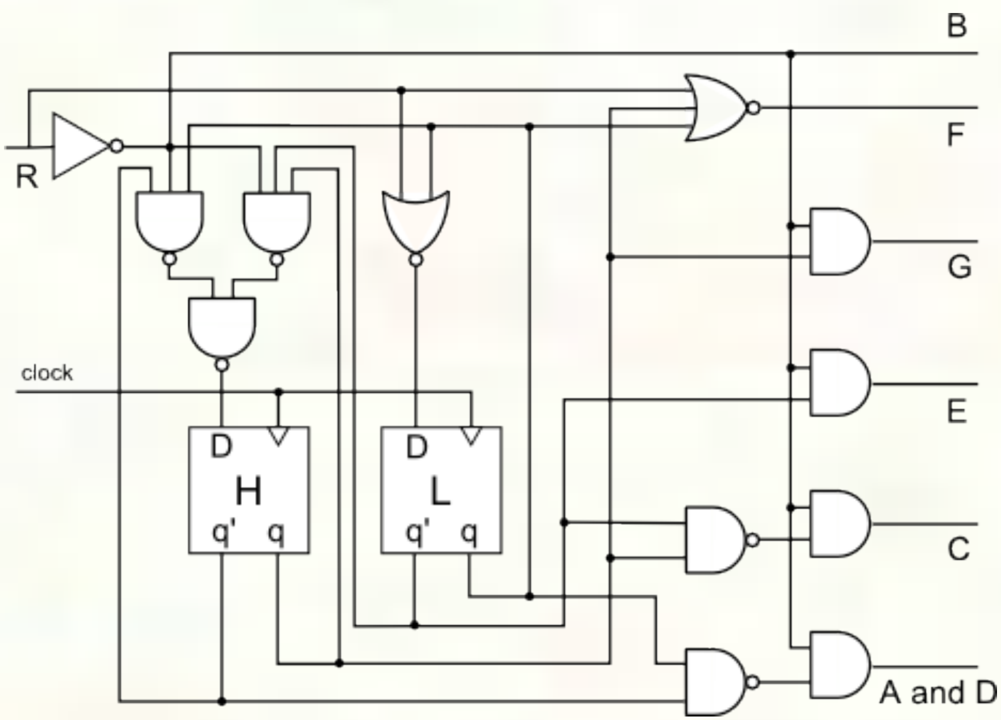
R	H_o	L_o	H_n	L_n	A	B	C	D	E	F	G
0	0	0	0	1	1	1	1	1	1	1	0
0	0	1	1	0	0	1	1	0	0	0	0
0	1	0	1	1	1	1	0	1	1	0	1
0	1	1	0	0	1	1	1	1	0	0	1
1	x	x	0	0	0	0	0	0	0	0	0

$A = D = R'H_o'L_o' + R'H_oL_o' + R'H_oL_o = R'(H_o'L_o)'$

$B = R'$ $\quad C = R'(H_oL_o')'$ $\quad E = R'L_o'$

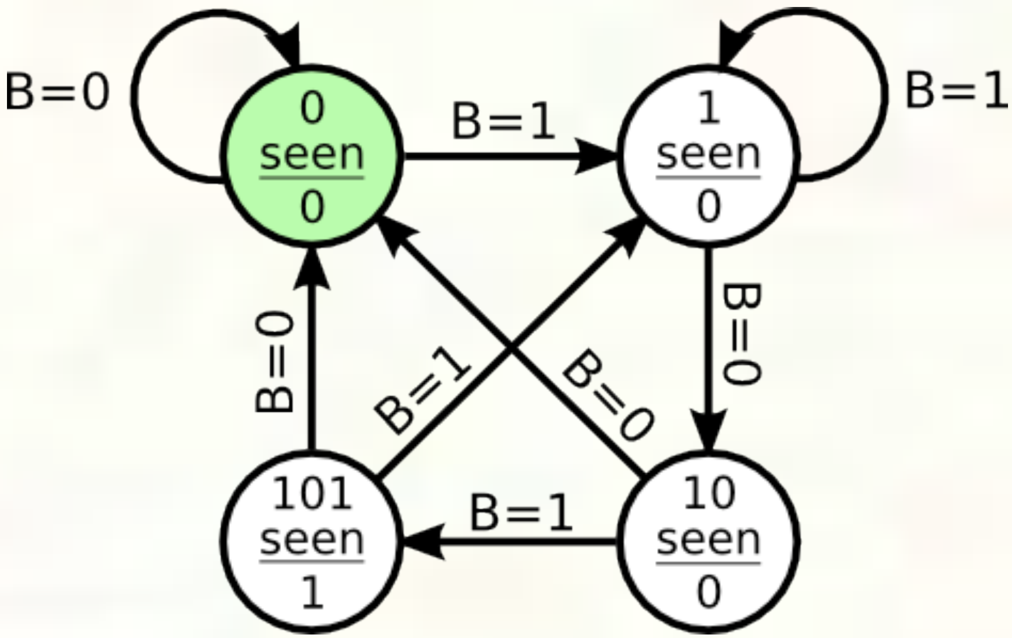
$F = R'H_o'L_o' = (R+H_o+L_o)'$ $\quad G = R'H_o$

7-segment output logic



Example - 101 lock

Combination lock with 101 being the combination



B is input signal to the lock,
X is output signal to unlock

B	H_o	L_o	H_n	L_n	X
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	0	0	0
0	1	1	0	0	1
1	0	0	0	0	0
1	0	1	0	1	0
1	1	0	1	1	1
1	1	1	0	1	0

101 combination lock

$X = H_o L_o$

$H_n = B'H_o'L_o + BH_oL_o'$

$L_n = BH_o'L_o + BH_oL_o' + BH_oL_o$
$= BH_o'L_o + BH_oL_o + BH_oL_o' + BH_oL_o$
$= BL_o + BH_o$

LC-3 datapath