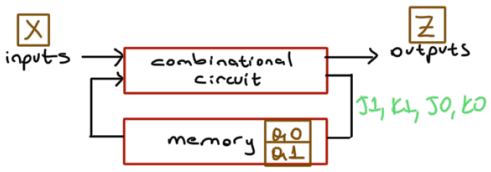
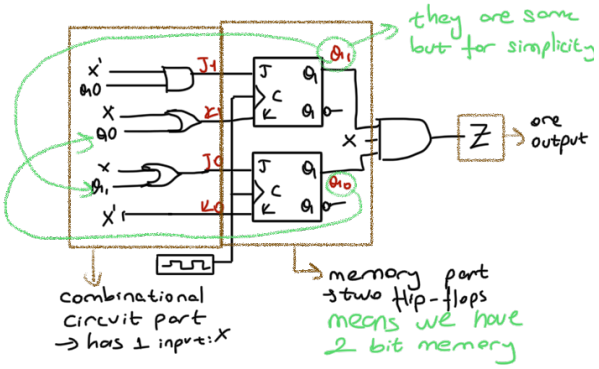
7- analyze and design sequential circuits

analyze

example:



Q1 and Q0: are feedbacks to the system



state table

present state Q_1	present state Q_0	inputs X	flip flop inputs J_1	flip flop inputs K_1	flip flop inputs J_0	flip flop inputs K_0	next state Q_1	next state Q_0	outputs Z
0	0	0	0	0	0	1	0	0	0
0	0	1	0	0	1	0	0	1	0
0	1	0	1	1	0	1	1	0	0
0	1	1	0	1	1	0	0	1	0
1	0	0	0	0	1	1	1	1	0
1	0	1	0	1	1	0	0	1	0
1	1	0	1	1	1	1	0	0	0
1	1	1	0	1	1	0	0	1	1

present state Q_1 a

$Z = Q_1 . X . Q_0$ (in our example)

remember:

J	K	$Q_1(t+1)$	operation
0	0	$Q_1(t)$	no chan
0	1	0	reset
1	0	1	set
1	1	$Q_1'(t)$	comple

filled with this

(in our example we have JK flip flops → it has 2 inputs)

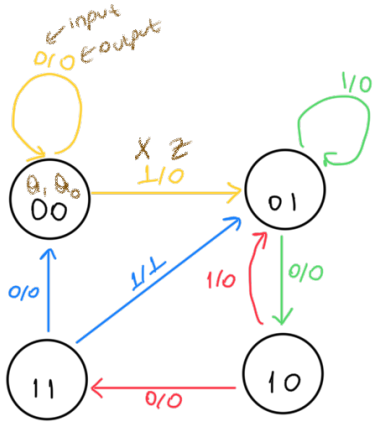
D and T flip flops have one input

flip-flop input equations

$J_1 = X' Q_0$ $\quad J_0 = X + Q_1$

$K_1 = X + Q_0$ $\quad K_0 = X'$

state diagrams



sizes of the state diagrams

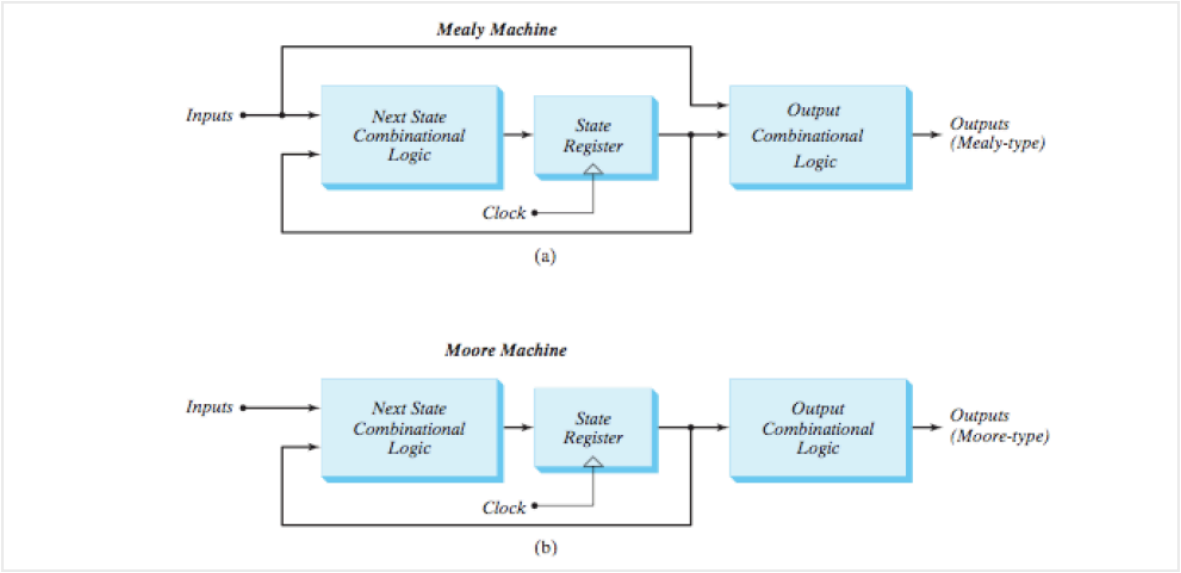
n flip flops → 2^n nodes

m inputs → 2^m each nodes' outgoing arrows

FSM (finite-state machine) models:

mealy machine = the output determined by current state + external input

moore machine = the output determined by only current state



design

example = to detect bit pattern "1001"

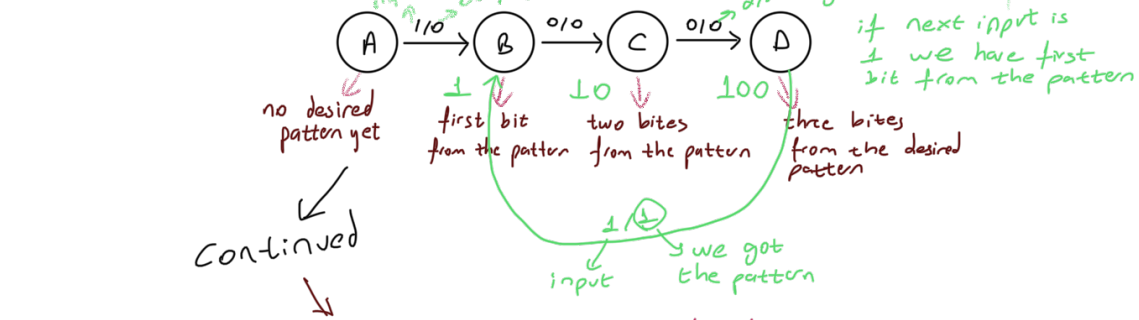
↳ one bit of input supplied on every clock cycle

ex: 1 1 1 0 0 1 1 0 1 0 0 1 0 0 1 1 0 0 1 inputs

0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 outputs

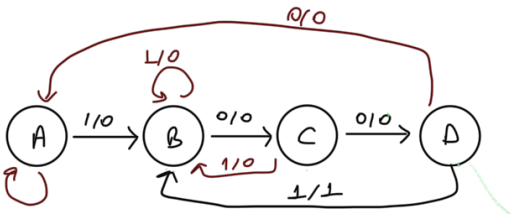
overlapping patterns

what we need to remember inputs from previous clock cycles 1001



continued

we need two outgoing arrows for each node for x=0, x=1.



present state → input/output → next state

present state	input	next state	output
A	0	A	0
A	1	B	0
B	0	B	0
B	1	C	0
C	0	D	0
C	1	B	0
D	0	A	0

0/0

how many flip-flops do we need? $\lceil \log_2 n \rceil$

when 1 FF → 0/1 → 2 states

2 FF → 00/01/10/11 → 4 states

3 FF → 000/001/.../111 → 8 states

we have 4 states

so → 2 flip-flops

assigning binary codes

A → 0 0
B → 0 1
C → 1 0
D → 1 1
$Q_1 Q_0$

Q_1 Q_0 present state	x input	Q_{1+} Q_{0+} next state	z output
0 0	0	0 0	0
0 0	1	0 1	0
0 1	0	1 0	0
0 1	1	0 1	0
1 0	0	1 1	0
1 0	1	0 1	0
1 1	0	0 0	0
1 1	1	0 1	1

J	K	$Q_{1(t+1)}$
0	0	$Q_1(t)$
0	1	0
1	0	1
1	1	$Q_1'(t)$

finding flip-flop inputs

- depends on what kind of flip-flop you use

J-K flip-flop

Q_1 Q_0 present state	x input	Q_{1+} Q_{0+} next state	flip flop inputs J_1	K_1	J_0	K_0	z output
0 0	0	0 0	0	x	0	x	0
0 0	1	0 1	0	x	1	x	0
0 1	0	1 0	1	x	x	1	0
0 1	1	0 1	0	x	x	0	0
1 0	0	1 1	x	0	1	x	0
1 0	1	0 1	x	1	1	x	0
1 1	0	0 0	x	1	x	1	0
1 1	1	0 1	x	1	x	0	1

present Q_1	J_1	K_1	next Q_{1+}
0	0	x	0
0	0	x	0
0	1	x	1
0	0	x	0
1	x	0	1
1	x	1	0
1	x	1	0
1	x	1	0

present Q_0	J_0	K_0	next Q_{0+}
0	0	x	0
0	1	x	1
1	x	1	0
1	x	0	1
0	1	x	1
0	1	x	1
1	x	1	0
1	x	0	1

finding equations for flip-flops

$J_1 = \theta_0 . x'$

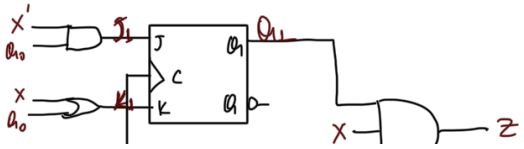
$J_0 = \theta_1 + x$

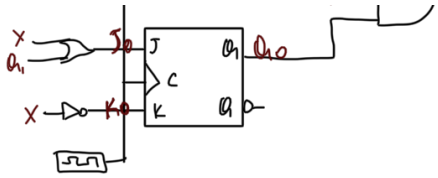
$K_1 = \theta_0 + x$

$K_0 = x'$

also for output → $z = x \theta_1 . \theta_0$

build the circuit

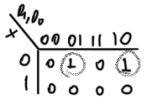




D flip-flop

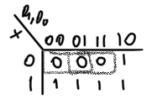
Q_1 present state	Q_0 present state	x input	Q_1 next state	Q_0 next state	D_1	D_0	z output
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0
0	1	0	1	0	1	0	0
0	1	1	0	1	0	1	0
1	0	0	1	1	1	1	0
1	0	1	0	1	0	1	0
1	1	0	0	0	0	0	0
1	1	1	0	1	0	1	1

$D_1 \rightarrow$

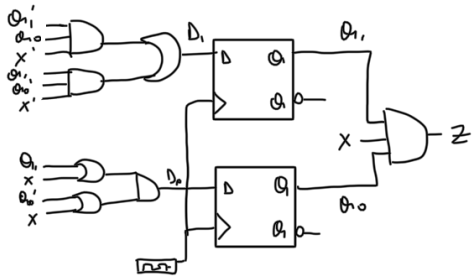


$Q_1' \cdot Q_0 \cdot X' + Q_1 Q_0' X'$

$D_0 \rightarrow$



$(Q_1 + X) \cdot (Q_0' + X)$



☆ JK flip flops are simpler (they are many don't cares); in D flip-flops we don't have to set up flip-flop inputs (since $D = Q(t+1)$), but they are more complex than JK

in practice D flip-flops are used more often
↳ one-input also

(uyarma)
excitation tables

they show what flip flop inputs are required in order to make desired state change
→ backwards of the characteristic tables

J	K	$Q(t+1)$	operation
0	0	$Q(t)$	no change
0	1	0	reset
1	0	1	set
1	1	$Q'(t)$	complement

$Q(t)$	$Q(t+1)$	J	K	Operation
0	0	0	x	no change / reset
0	1	1	x	set / complement
1	0	x	1	reset / complement
1	1	x	0	no change / set

D	$Q(t+1)$	operation
0	0	reset
1	1	set

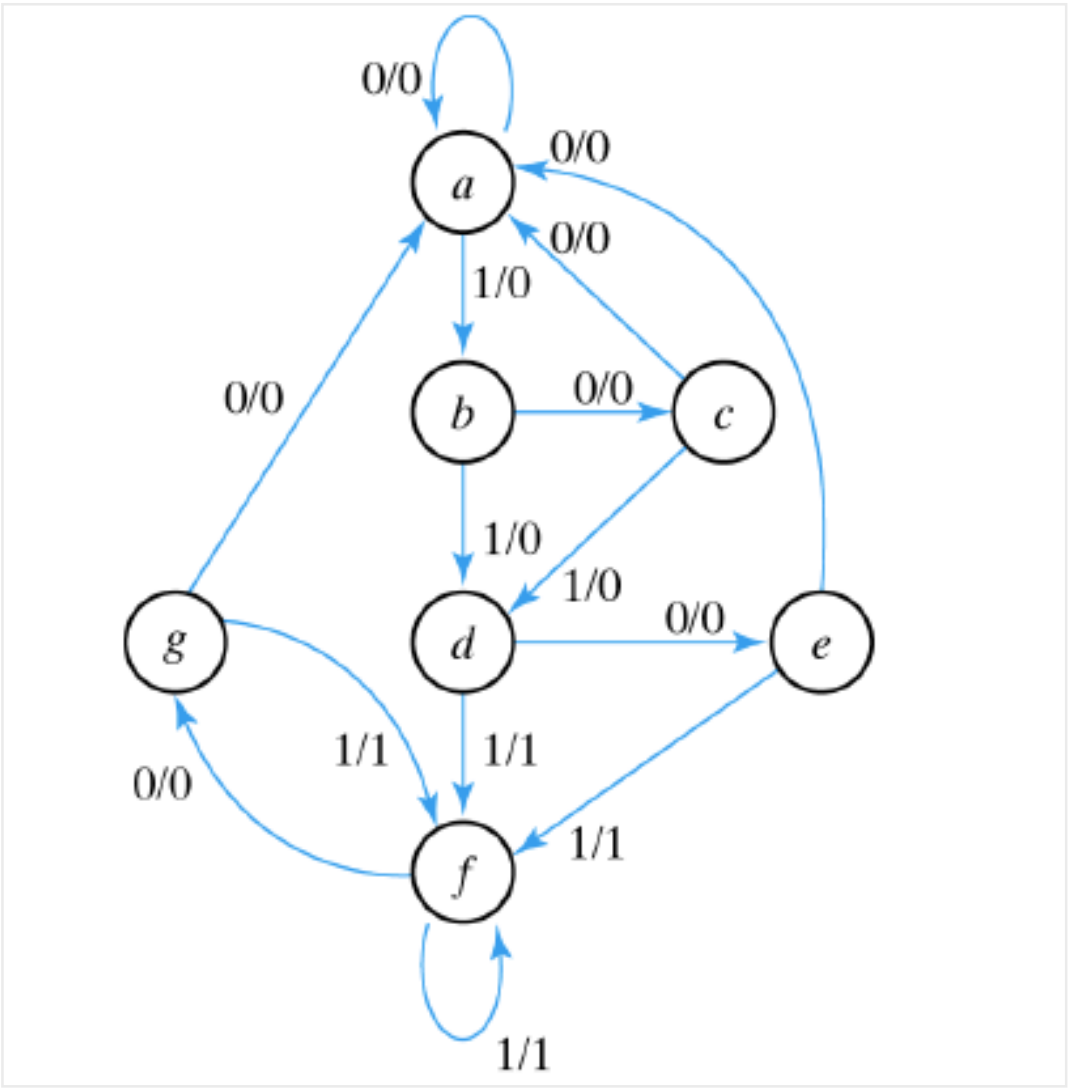
$Q(t)$	$Q(t+1)$	D	Operation
0	0	0	reset
0	1	1	set
1	0	0	reset
1	1	1	set

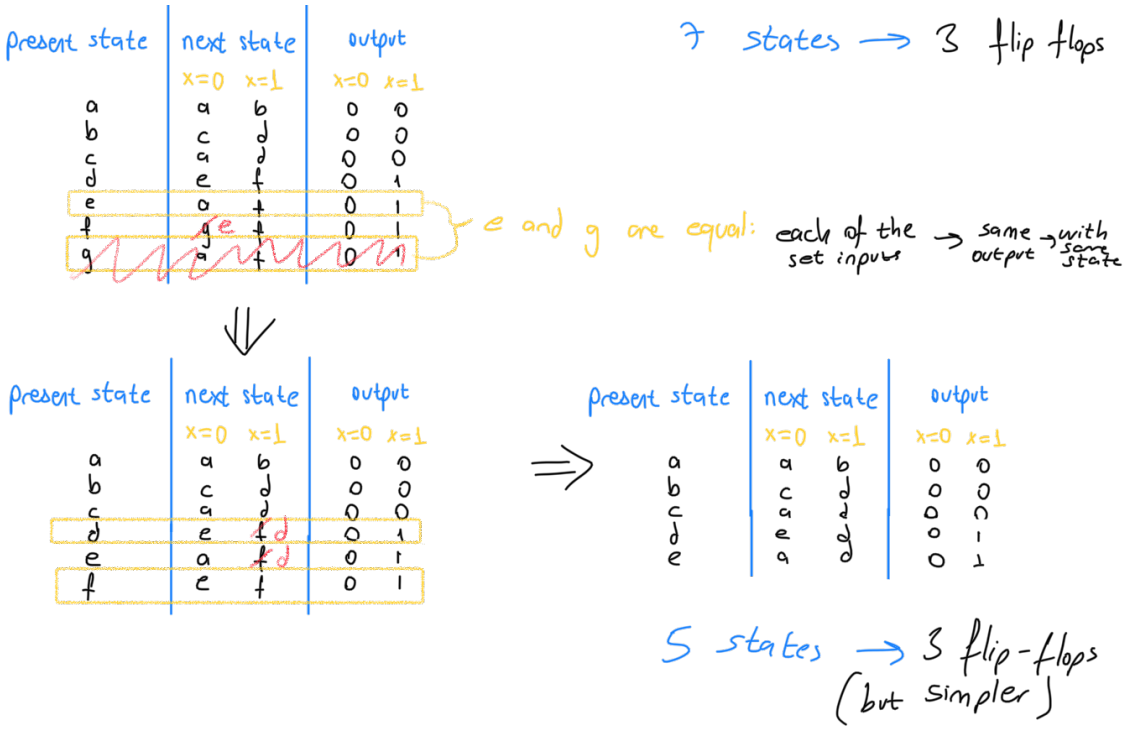
I	$Q_i(t+1)$	operation
0	$Q_i(t)$	no change
1	$Q_i'(t)$	complement

$Q_i(t)$	$Q_i(t+1)$	I	Operation
0	0	0	no change
0	1	1	complement
1	0	1	complement
1	1	0	no change

State reduction

reducing the number of states while keeping the external input-output requirements


0/0
a
0/0
0/0
1/0
0/0
b
0/0
c
1/0
1/0
0/0
g
d
e
1/1
1/1
0/0
1/1
f
1/1



☆ zeros catching → RS latch with 2 NOR gates
→ flip flop made of NAND gates