CENG-232
Logic Design

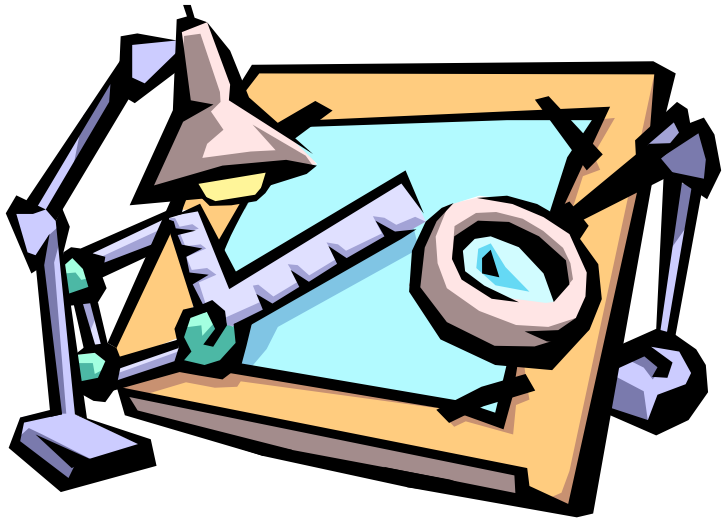
Lecture 6
Sequential Circuit Design

Spring 2015 - Uluç Saranlı
saranli@ceng.metu.edu.tr

Acknowledgment: Most of the following slides are adapted from Prof. Kale's slides at UIUC, USA.

Sequential Circuit Design

- In sequential circuit design, we turn some description into a working circuit.
  - We first make a state table or diagram to express the computation.
  - Then we can turn that table or diagram into a sequential circuit.



Sequence Recognizers

- A sequence recognizer is a special kind of sequential circuit that looks for a special bit pattern in some input.
- The recognizer circuit has only one input, X.
  - One bit of input is supplied on every clock cycle. For example, it would take 20 cycles to scan a 20-bit input.
  - This is an easy way to permit arbitrarily long input sequences.
- There is one output, Z, which is 1 when the desired pattern is found.
- Our example will detect the bit pattern “1001”:

Input:	1 1 1 0 0 1 1 0 1 0 0 1 0 0 1 1 0 …
Output:	0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 0 …

Sequence Recognizers (Cont'd.)

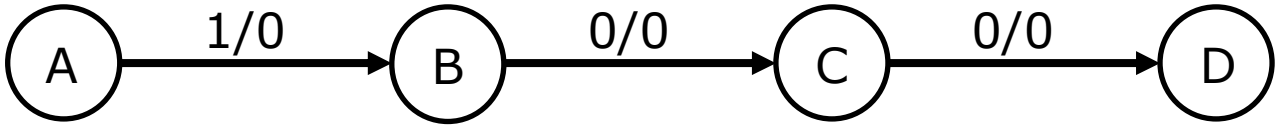
Inputs: 1 1 1 0 0 1 1 0 1 0 0 1 0 0 1 1 0 …

Outputs: 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 0 …

- This requires a sequential circuit because the circuit has to "remember" the inputs from previous clock cycles, in order to determine whether or not a match was found.
- Here, one input and one output bit appear every clock cycle.
- Note that overlapping bit patterns are also detected.

A basic state diagram

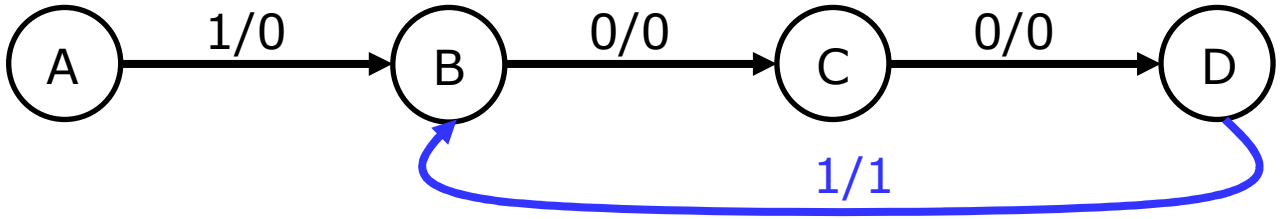
- What state do we need for the sequence recognizer?
  - We have to "remember" inputs from previous clock cycles.
  - For example, if the previous three inputs were 100 and the current input is 1, then the output should be 1.
  - In general, we will have to remember occurrences of parts of the desired pattern - in this case, 1, 10, and 100.
- We'll start with a basic state diagram:



State	Meaning
A	None of the desired pattern (1001) has been input yet.
B	We've already seen the first bit (1) of the desired pattern.
C	We've already seen the first two bits (10) of the desired pattern.
D	We've already seen the first three bits (100) of the desired pattern.

Overlapping pattern occurrences

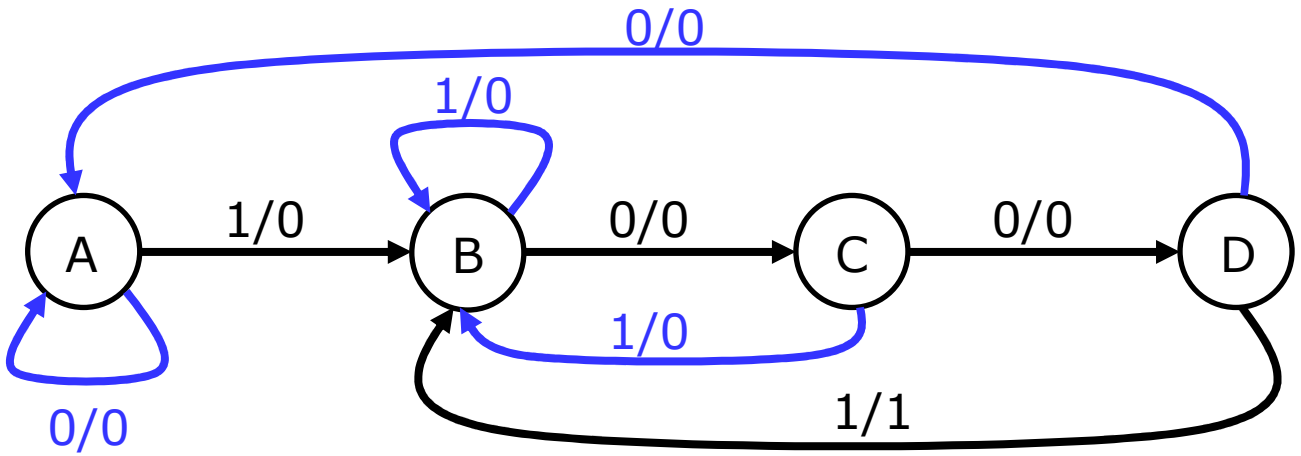
- What happens if we're in state D (the last three inputs were 100), and the current input is 1?
  - The output should be a 1, because we've found the desired pattern.
  - But this last 1 could also be the start of another occurrence of the pattern! For example, 1001001 contains two occurrences of 1001.
  - To detect overlapping occurrences of the pattern, the next state should be B.



State	Meaning
A	None of the desired pattern (1001) has been input yet.
B	We've already seen the first bit (1) of the desired pattern.
C	We've already seen the first two bits (10) of the desired pattern.
D	We've already seen the first three bits (100) of the desired pattern.

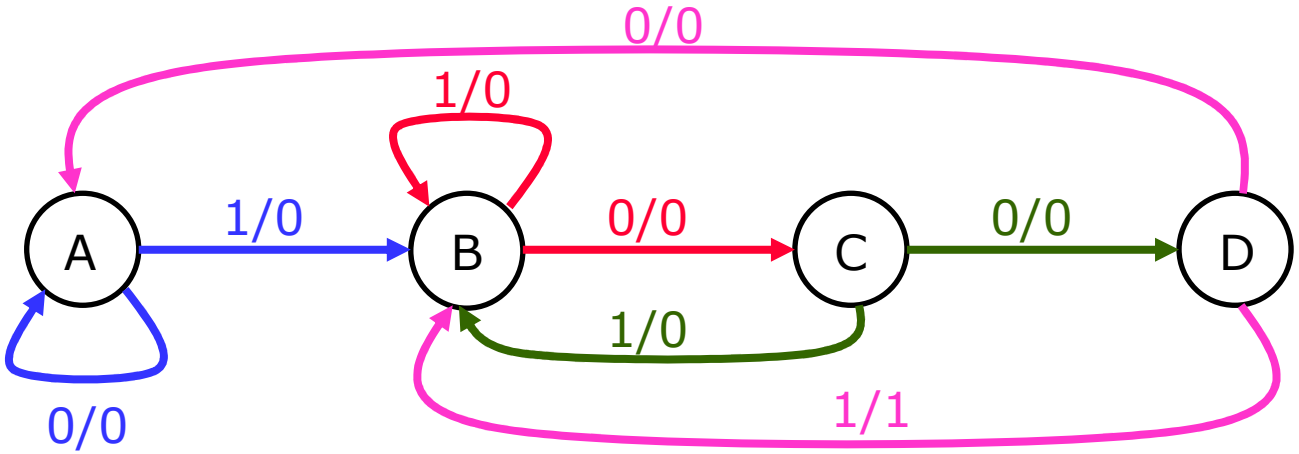
Filling in the other arrows

- Remember that we need two outgoing arrows for each node, to account for the possibilities of X=0 and X=1.
- The remaining arrows are shown in blue. They also allow for the correct detection of overlapping occurrences of 1001.

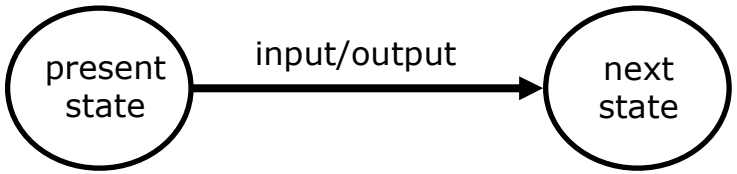


State	Meaning
A	None of the desired pattern (1001) has been input yet.
B	We've already seen the first bit (1) of the desired pattern.
C	We've already seen the first two bits (10) of the desired pattern.
D	We've already seen the first three bits (100) of the desired pattern.

Finally, making the state table



Remember how the state diagram arrows correspond to rows of the state table:



Present State	Input	Next State	Output
A	0	A	0
A	1	B	0
B	0	C	0
B	1	B	0
C	0	D	0
C	1	B	0
D	0	A	0
D	1	B	1

Sequential Circuit Design Procedure

- Step 1:

  Make a state table based on the problem statement. The table should show the present states, inputs, next states and outputs. (It may be easier to find a state diagram first, and then convert that to a table.)

- Step 2:

  Assign binary codes to the states in the state table, if you haven't already. If you have n states, your binary codes will have at least $\lceil \log_2 n \rceil$ digits, and your circuit will have at least $\lceil \log_2 n \rceil$ flip-flops.

Sequential circuit design procedure

- Step 3:

  For each flip-flop and each row of your state table, find the flip-flop input values that are needed to generate the next state from the present state. You can use flip-flop excitation tables here.

- Step 4:

  Find simplified equations for the flip-flop inputs and the outputs.

- Step 5:

  Build the circuit!

Step 2: Assigning binary codes

- We have four states ABCD, so we need at least two flip-flops Q1Q0.
- The easiest thing to do is represent state A with Q1Q0 = 00, B with 01, C with 10, and D with 11 (intuiative).
- The state assignment can have a big impact on circuit complexity, but we won't worry about that too much in this class.

Present State	Input	Next State	Output
A	0	A	0
A	1	B	0
B	0	C	0
B	1	B	0
C	0	D	0
C	1	B	0
D	0	A	0
D	1	B	1

Present State Q_1	Present State Q_0	Input X	Next State Q_1	Next State Q_0	Output Z
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

Step 3: Finding flip-flop inputs

- Next we have to figure out how to actually make the flip-flops change from their present state into the desired next state.
- This depends on what kind of flip-flops you use!
- We'll use two JKs. For each flip-flip Qi, look at its present and next states, and determine what the inputs Ji and Ki should be in order to make that state change.

Present State		Input	Next State		Flip flop inputs				Output
Q_1	Q_0	X	Q_1	Q_0	J_1	K_1	J_0	K_0	Z
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

Finding JK flip-flop input values

- For JK flip-flops, this is a little tricky. Recall the characteristic table:

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

- If the present state of a JK flip-flop is 0 and we want the next state to be 1, then we have two choices for the JK inputs:
  - We can use JK=10, to explicitly set the flip-flop's next state to 1.
  - We can also use JK=11, to complement the current state 0.
- So to change from 0 to 1, we must set J=1, but K could be either 0 or 1.
- Similarly, the other possible state transitions can all be done in two different ways as well.

JK excitation table

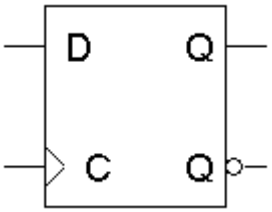
- An excitation table shows what flip-flop inputs are required in order to make a desired state change.

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

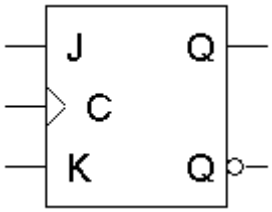
- This is the same information that's given in the characteristic table, but presented "backwards."

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

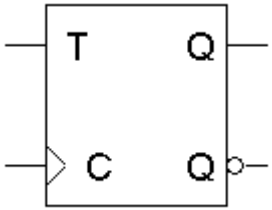
Excitation Tables for all flip-flops



Q(t)	Q(t+1)	D	Operation
0	0	0	Reset
0	1	1	Set
1	0	0	Reset
1	1	1	Set



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



Q(t)	Q(t+1)	T	Operation
0	0	0	No change
0	1	1	Complement
1	0	1	Complement
1	1	0	No change

Back to the Example

- We can now use the JK excitation table on the right to find the correct values for each flip-flop's inputs, based on its present and next states.

Q(t)	Q(t+1)	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Present State		Input	Next State		Flip flop inputs				Output
Q_1	Q_0	X	Q_1	Q_0	J_1	K_1	J_0	K_0	Z
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

Step 4: Find FF in/out equations

- Now you can make K-maps and find equations for each of the four flip-flop inputs, as well as for the output Z.
- These equations are in terms of the present state and the inputs.
- The advantage of using JK flip-flops is that there are many don't care conditions, which can result in simpler equations.

J1 = X' Q0

K1 = X + Q0

J0 = X + Q1

K0 = X'

Z = Q1Q0X

Present State		Input	Next State		Flip flop inputs				Output
Q_1	Q_0	X	Q_1	Q_0	J_1	K_1	J_0	K_0	Z
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

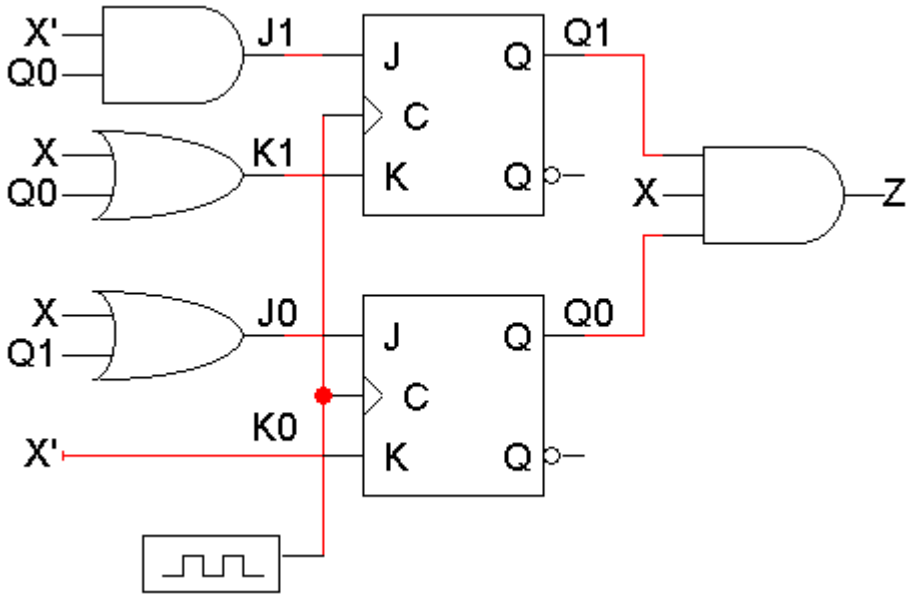
Step 5: Build the Circuit

- Lastly, we use these simplified equations to build the completed circuit.

$J_1 = X' Q_0$
$K_1 = X + Q_0$

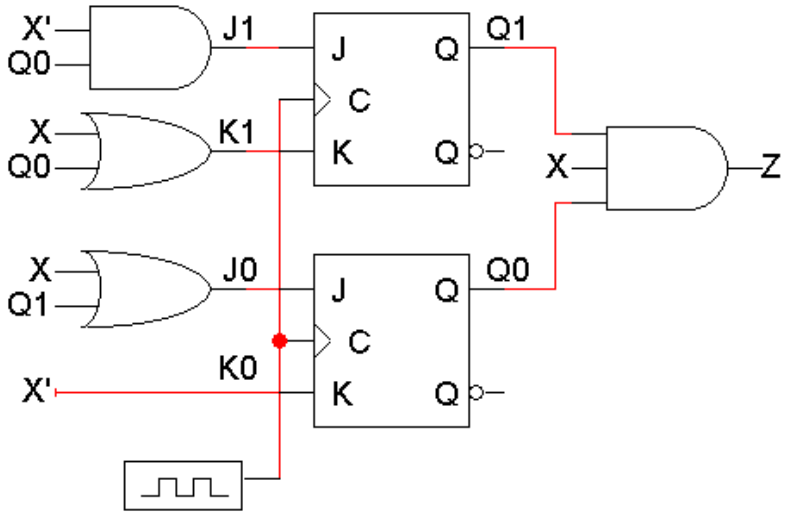
$J_0 = X + Q_1$
$K_0 = X'$

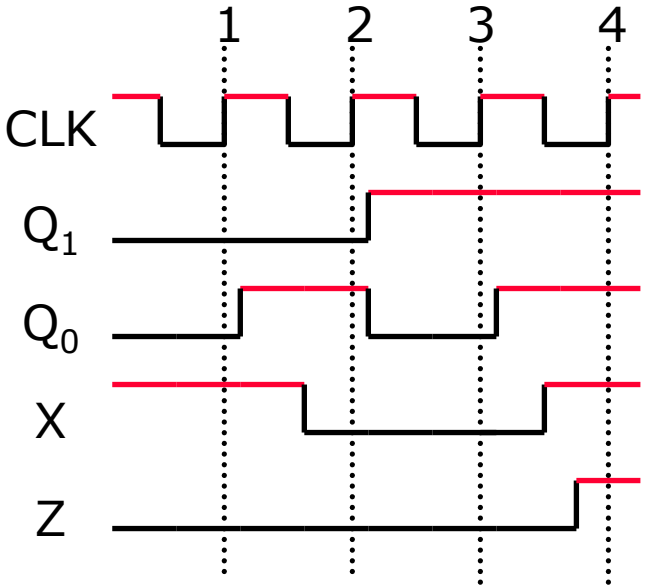
$Z = Q_1 Q_0 X$



Timing Diagram

- Here is one example timing diagram for our sequence detector.
  - The flip-flops Q1Q0 start in the initial state, 00.
  - On the first three positive clock edges, X is 1, 0, and 0. These inputs cause Q1Q0 to change, so after the third edge Q1Q0 = 11.
  - Then when X=1, Z becomes 1 also, meaning that 1001 was found.
- The output Z does not have to change at positive clock edges. Instead, it may change whenever X changes, since Z = Q1Q0X.





Using D flip-flops

- What if you want to build the circuit using D flip-flops instead?
- We already have the state table and state assignments, so we can just start from Step 3, finding the flip-flop input values.
- D flip-flops have only one input, so our table only needs two columns for D1 and D0.

Present State		Input	Next State		Flip-flop inputs		Output
Q_1	Q_0	X	Q_1	Q_0	D_1	D_0	Z
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

D flip-flop input values (Step 3)

- The D excitation table is pretty boring; set the D input to whatever the next state should be.

Q(t)	Q(t+1)	D	Operation
0	0	0	Reset
0	1	1	Set
1	0	0	Reset
1	1	1	Set

- You don't even need to show separate columns for D1 and D0; you can just use the Next State columns.

Present State Q_1	Present State Q_0	Input X	Next State Q_1	Next State Q_0	Flip flop inputs D_1	Flip flop inputs D_0	Output Z
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

The same

Finding Equations (Step 4)

- You can do K-maps again, to find:

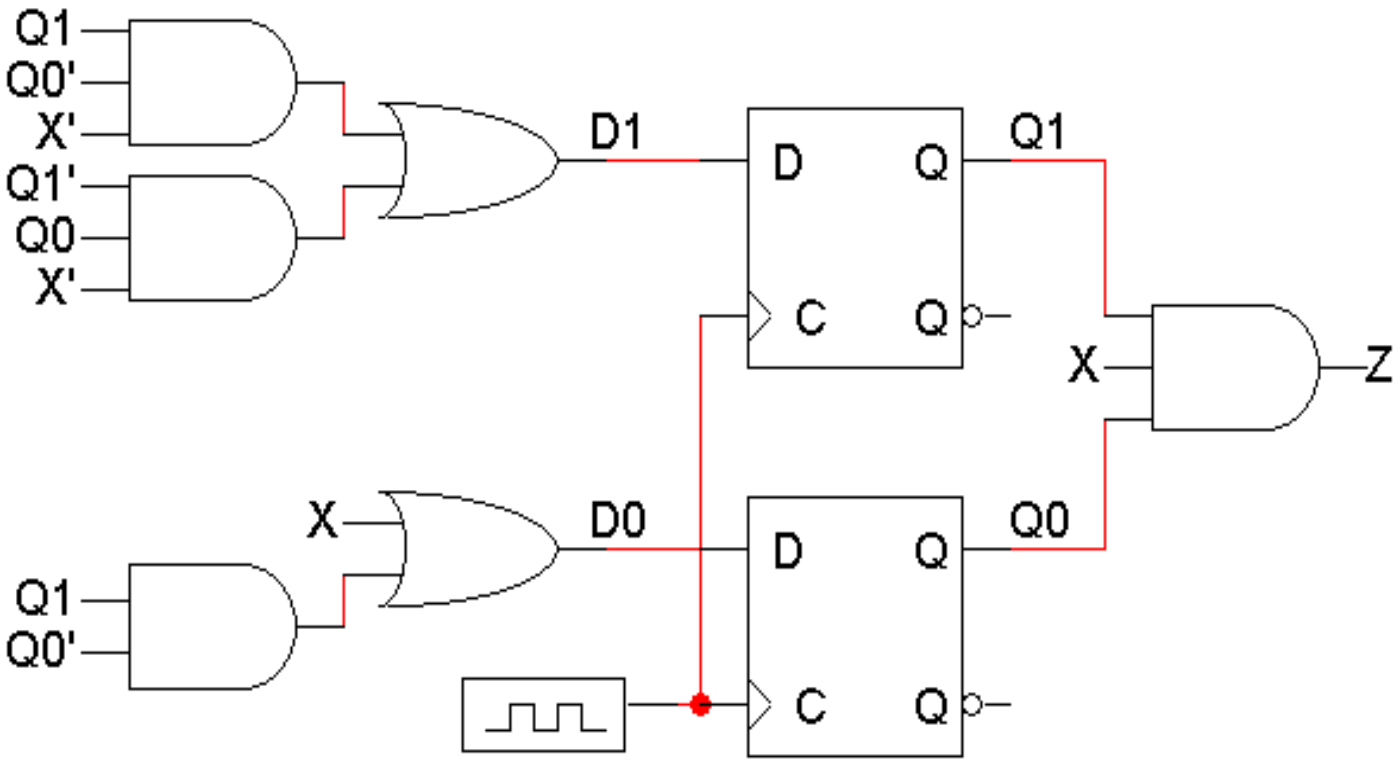
  D1 = Q1 Q0' X' + Q1' Q0 X'

  D0 = X + Q1 Q0'

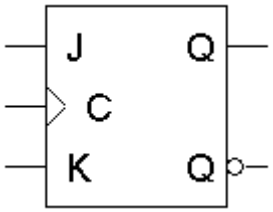
  Z = Q1 Q0 X

Present State		Input	Next State		Flip flop inputs		Output
Q_1	Q_0	X	Q_1	Q_0	D_1	D_0	Z
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

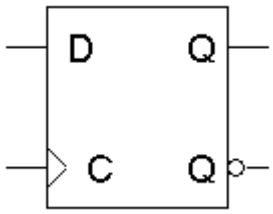
Building the Circuit (Step 5)



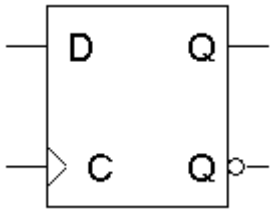
Flip-flop comparison



JK flip-flops are good because there are many don't care values in the flip-flop inputs, which can lead to a simpler circuit.



D flip-flops have the advantage that you don't have to set up flip-flop inputs at all, since Q(t+1) = D. However, the D input equations are usually more complex than JK input equations



In practice, D flip-flops are used more often.

- There is only one input for each flip-flop, not two.
- There are no excitation tables to worry about.
- D flip-flops can be implemented with slightly less hardware than JK flip-flops.

Summary

- The basic sequential circuit design procedure:
  - Make a state table and, if desired, a state diagram. This step is usually the hardest.
  - Assign binary codes to the states if you didn't already.
  - Unused states can be treated as don't care conditions.
  - Use the present states, next states, and flip-flop excitation tables to find the flip-flop input values.
  - Write simplified equations for the flip-flop inputs and outputs and build the circuit.

- Next; how do we minimize the states to be used?
  - Do we need that?

Work …

- Design a sequential circuit that detects 01 patterns coming through an input line X. The circuit's output Z should be set 1 when a 01 pattern occurs and to 0 otherwise.

X	0111001000010
Z	0100000100001 0

- Draw the state diagram of the circuit
- Derive the state table
- Implement using D-FF's
  - Extend the state table for D-FF inputs
  - Derive the expressions for D-FF inputs
  - Draw the full circuit

State Reduction and Assignment

- Goal: Reduce the number of states while keeping the external input-output requirements.
- 2^m states need m flip-flops, so reducing the states may reduce flip-flops.
- If two states are equal, one can be removed but what are “equal states”?
  - State Equivalence

State Reduction Example

As an example consider the input sequence below:

"010101110100" applied and start from state a.

State	a	a	b	c	d	e	f	f	g	f	g	a
Input	0	1	0	1	0	1	1	0	1	0	0	
Output	0	0	0	0	0	1	1	0	1	0	0	

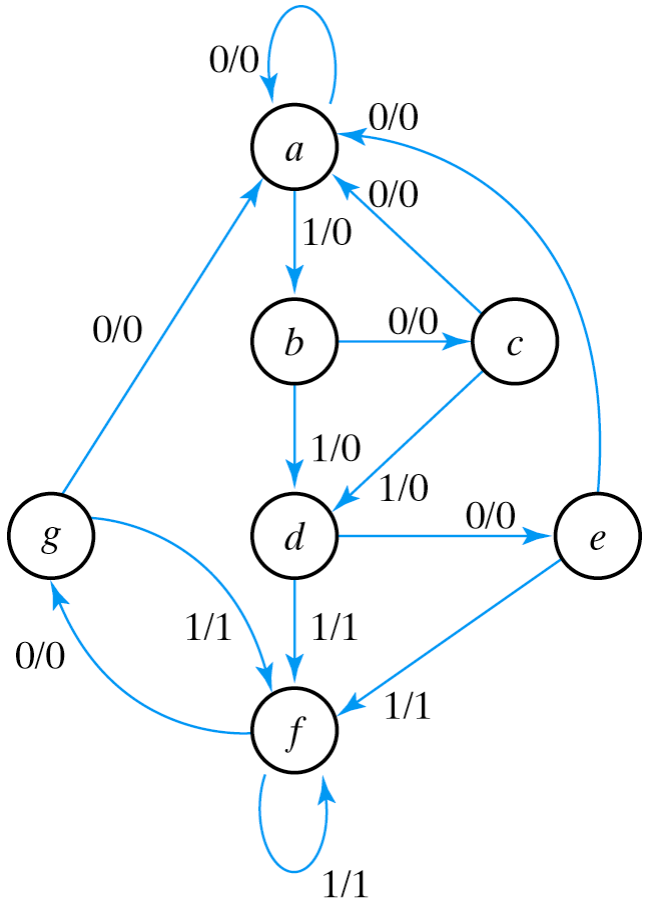


Fig. 5-22 State Diagram

State Reduction Example

Present State	*Next State*		*Output*	
	x=0	x=1	x=0	x=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

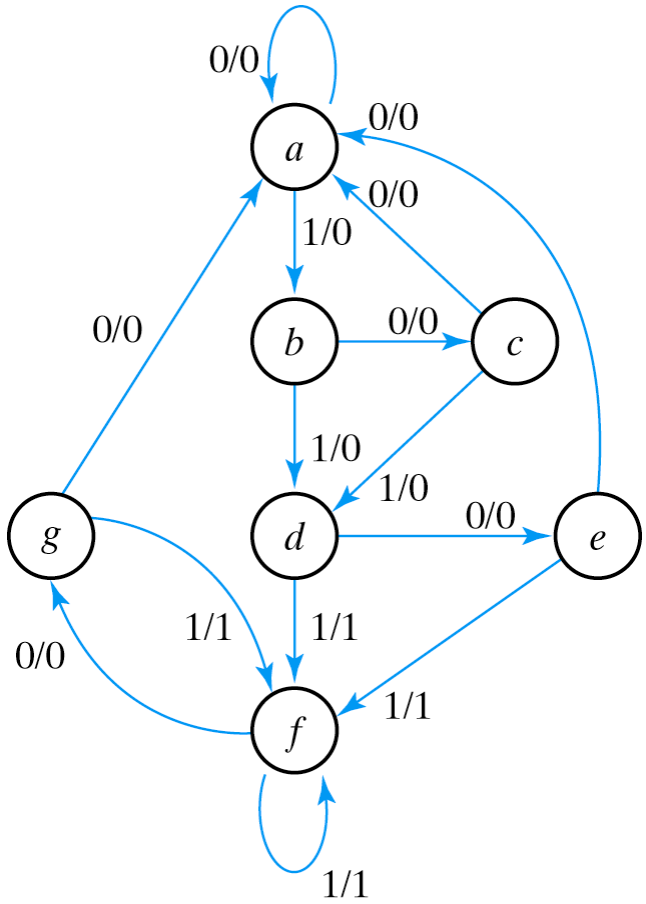


Fig. 5-22 State Diagram

State Reduction Example

Present State	*Next State*		*Output*	
	x=0	x=1	x=0	x=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

States e and g are equal since for each member of the set of inputs, they give the same output and send the circuit either to the same state or an equivalent state.

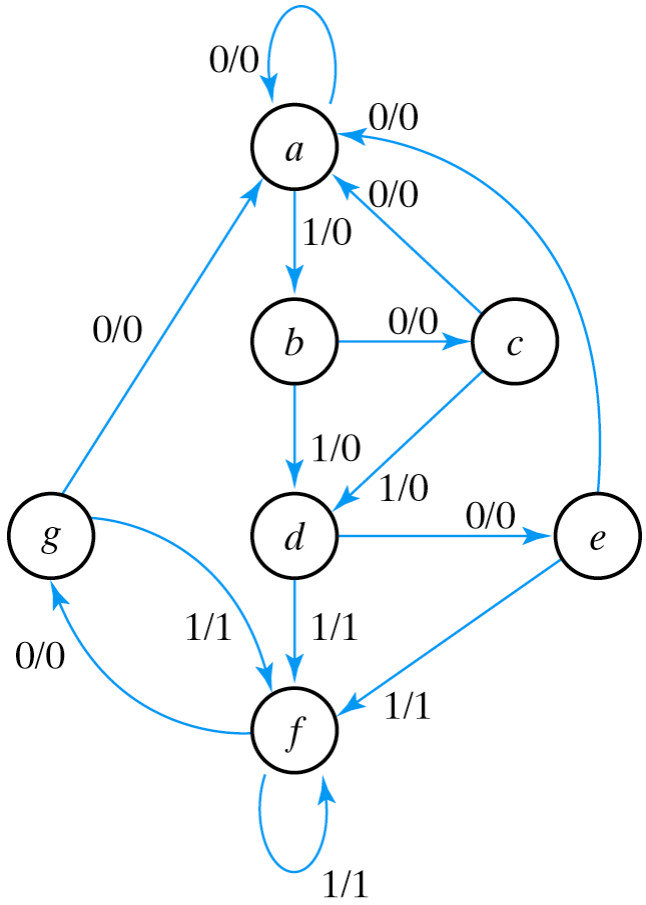


Fig. 5-22 State Diagram

State Reduction Example

Present State	*Next State*		*Output*	
	x=0	x=1	x=0	x=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	~~g~~ e	f	0	1
g	a	f	0	1

Table and state diagram after the first reduction: state g is removed and replaced by state e.

State Reduction Example

Present State	*Next State*		*Output*	
	x=0	x=1	x=0	x=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	e	f	0	1
g	a	f	0	1

Table and state diagram after the first reduction: g is removed and replaced by state e.

However, we now have NEW equal states: d and f

State Reduction Example

Present State	*Next State*		*Output*	
	x=0	x=1	x=0	x=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1
f	e	f	0	1
g	a	f	0	1

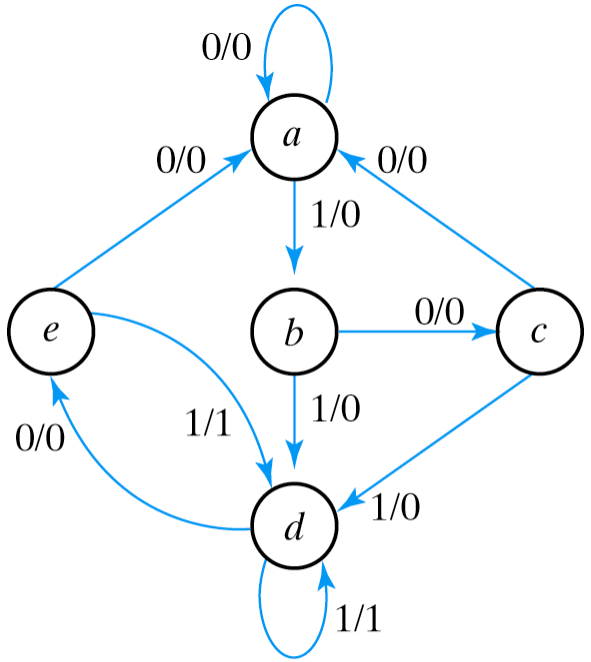


Fig. 5-23 Reduced State Diagram

Table and state diagram after the second reduction: f is removed and replaced by state d.

If we apply the same input sequence:

State	**a**	**a**	**b**	**c**	**d**	**e**	**d**	**d**	**e**	**d**	**e**	**a**
input	0	1	0	1	0	1	1	0	1	0	0	
output	0	0	0	0	0	1	1	0	1	0	0