



16.07.20 13:57

# CSCI 150

## Introduction to Digital and Computer System Design

### Lecture 4: Sequential Circuit V



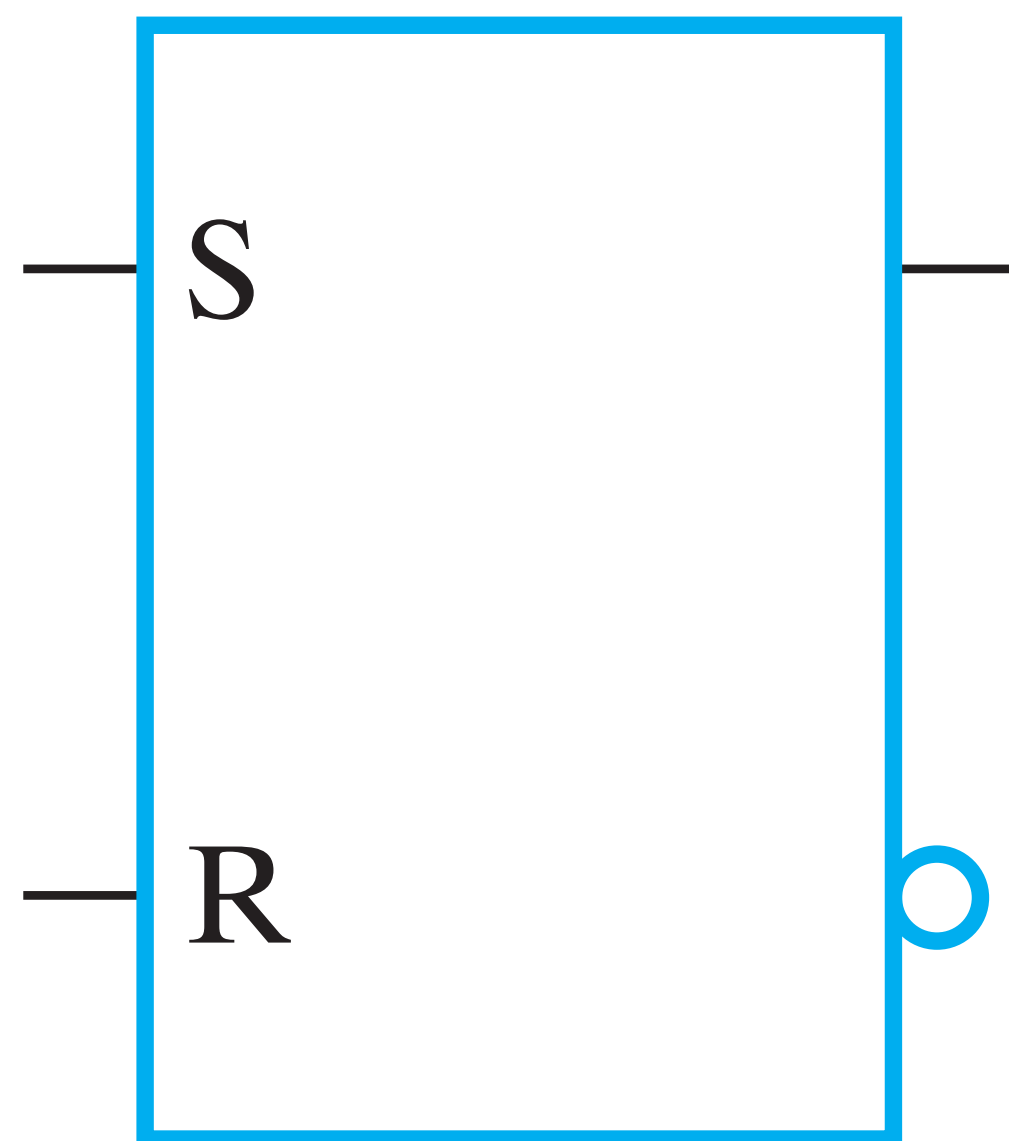
Jetic Gū  
2020 Summer Semester (S2)

# Overview

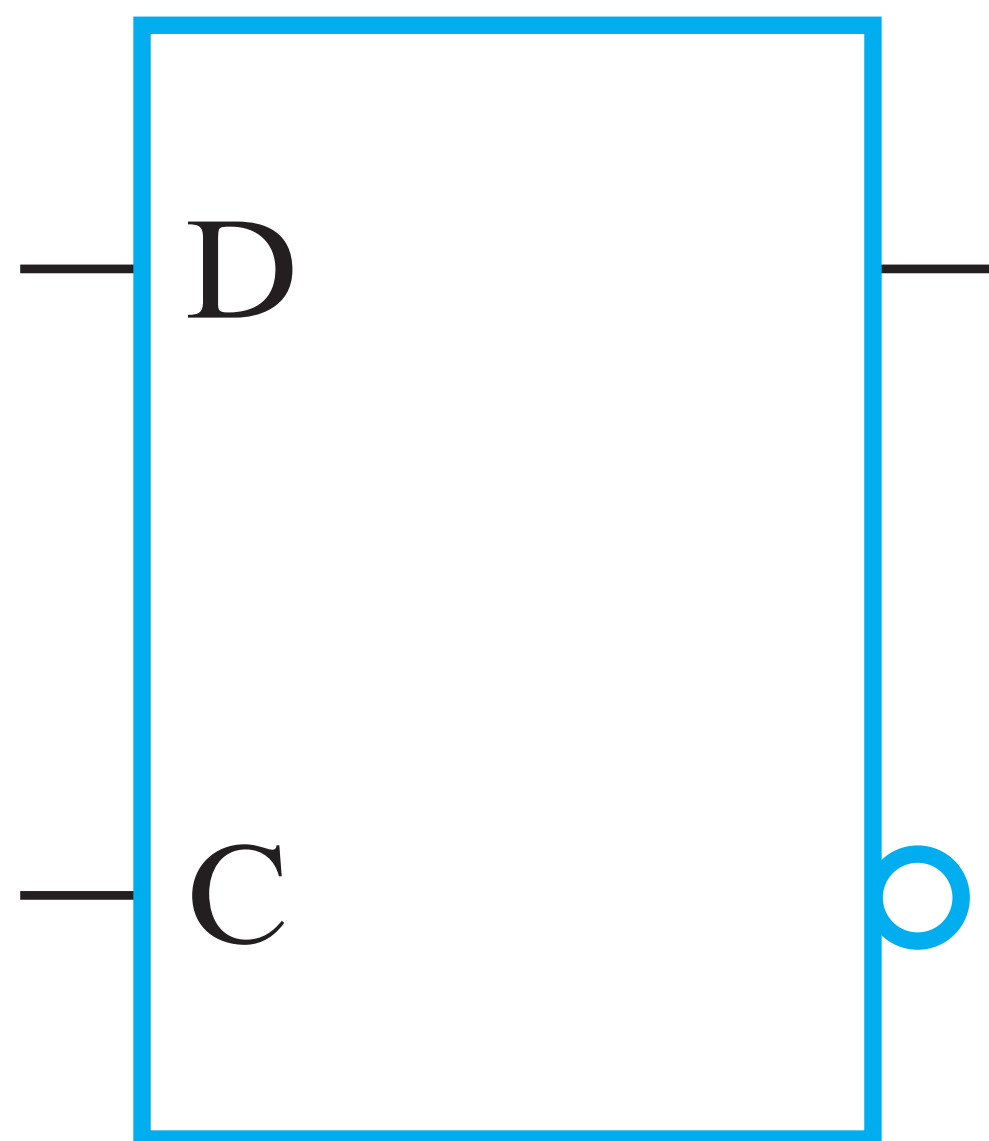
- Focus: Basic Information Retaining Blocks
- Architecture: Sequential Circuit
- Textbook v4: Ch5 5.5, 5.6; v5: Ch4 4.5
- Core Ideas:
  1. Sequential Circuit Design Procedures
  2. Other Flip-Flop Types

# Latches and Flip-Flops

## Latches

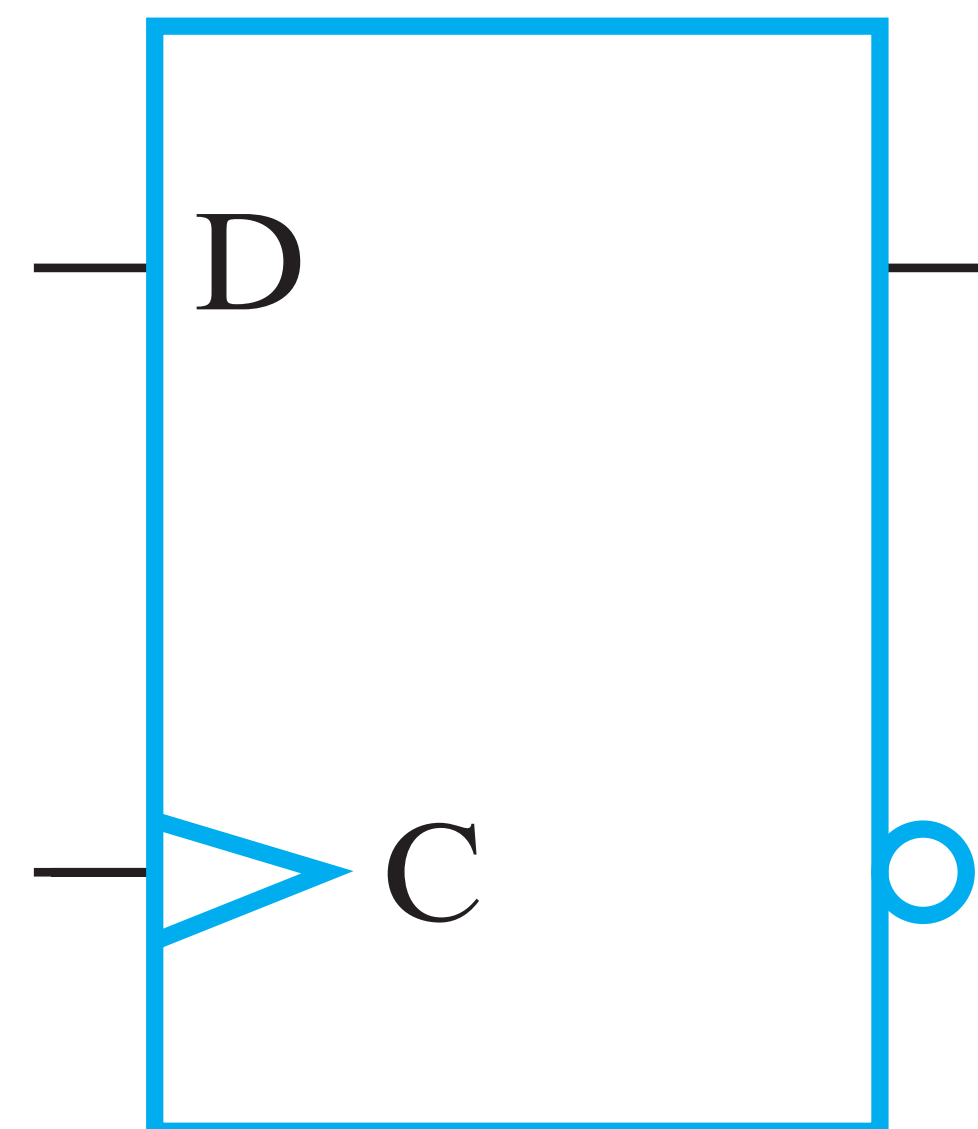


SR



D with 1 Control

## Flip-Flops



Triggered D

# Systematic Design Procedures Sequential Circuits

1. **Specification**
2. **Formulation**  
e.g. using **state table** or **state diagram**
3. **State Assignment**: assign binary codes to states
4. **Flip-Flop Input Equation Determination**: Select flip-flop types, derive input equations from next-state entries
5. **Output Equation Determination**: Derive output equations from the output entries
6. **Optimisation**
7. **Technology Mapping**
8. **Verification**

# Sequential Circuit Design II

State Assignment; Input Equation Determination;  
Output Equation Determination

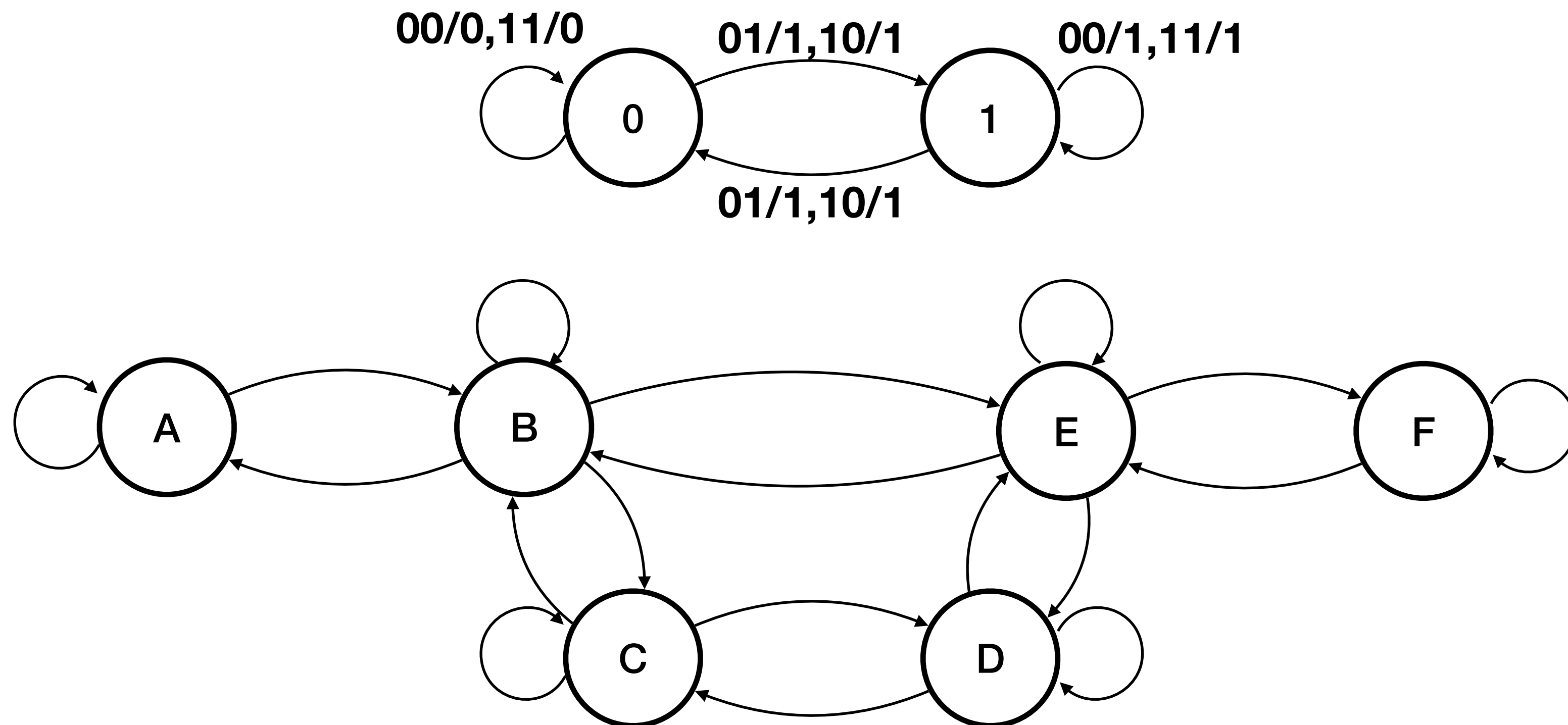
# Systematic Design Procedures

## Sequential Circuits

1. **Specification**
2. **Formulation**  
e.g. using **state table** or **state diagram**
3. **State Assignment**: assign binary codes to states
4. **Flip-Flop Input Equation Determination**: Select flip-flop types, derive input equations from next-state entries
5. **Output Equation Determination**: Derive output equations from the output entries
6. **Optimisation**
7. **Technology Mapping**
8. **Verification**

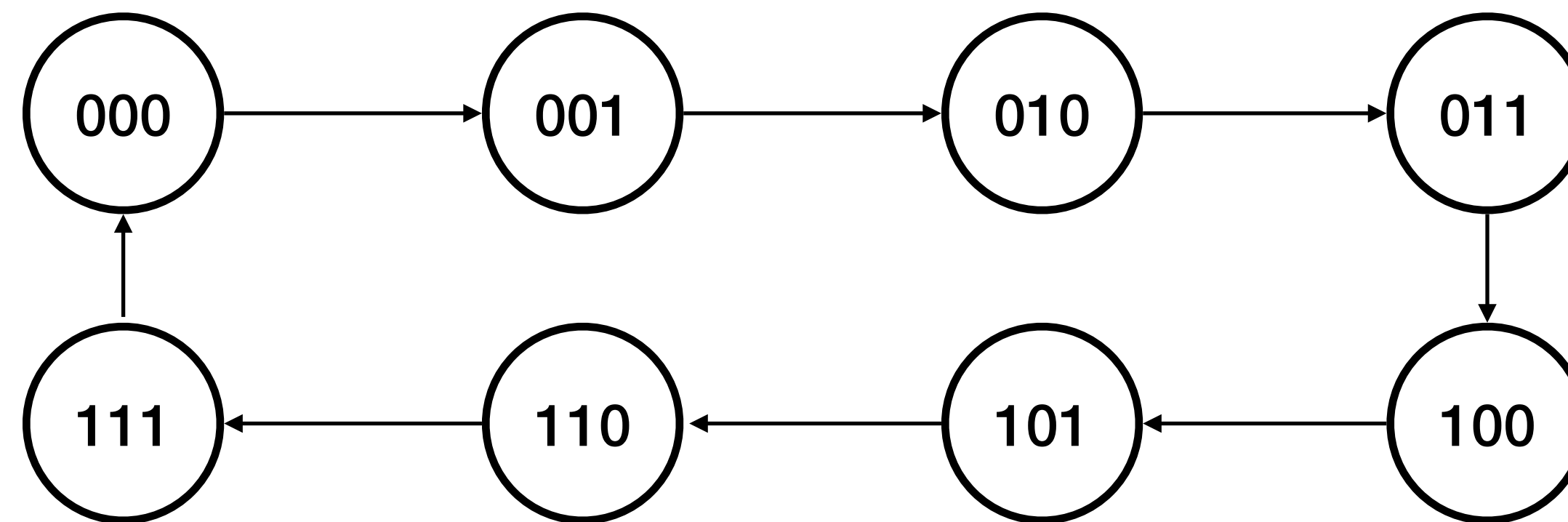
## 2. Formulation

- Sometimes it is more intuitive to describe state transitions then defining the states



## 2. Formulation

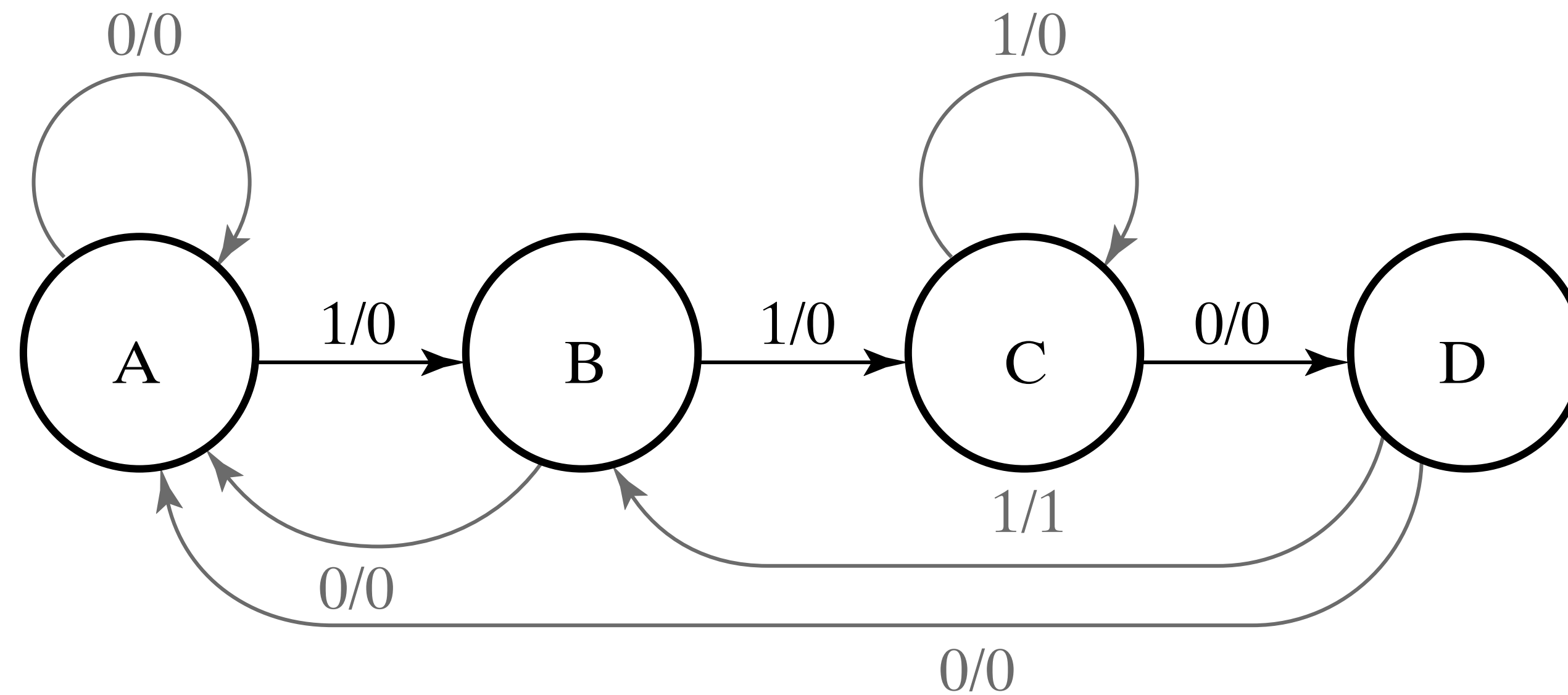
- Incrementer: perform  $+1$  operation every  $\text{CLK}$  on 3-bit





# 3. State Assignment

- Used when states are quite complicated and expressed using variables during **Formulation**
- Define the **binary values** for each state



State Diagram

# 3. State Assignment

- Used when states are quite complicated and expressed using variables during **Formulation**
- Define the **binary values** for each state

Present State	Next State		Output Z	
	X = 0	X = 1	X = 0	X = 1
A	A	B	0	0
B	A	C	0	0
C	D	C	0	0
D	A	B	0	1

State Table

# 3. State Assignment

- Method 1: sequential assignment  
 $A = 0, B = 1, C = 2, D = 3, \dots$

Present State	Next State		Output Z	
	X = 0	X = 1	X = 0	X = 1
A 00	00 A	B 01	0	0
B 01	00 A	C 10	0	0
C 10	11 D	C 10	0	0
D 11	00 A	B 01	0	1

State Table

# 3. State Assignment

- Method 2: one hot

$$A = (0001)_2, B = (0010)_2, C = (0100)_2, D = (1000)_2$$

Present State	Next State		Output Z	
	X = 0	X = 1	X = 0	X = 1
A 0001	0001 A	B 0010	0	0
B 0010	0001 A	C 0100	0	0
C 0100	1000 D	C 0100	0	0
D 1000	0001 A	B 0010	0	1

State Table

# 3. State Assignment

- Are these the only methods?
  - No, there's tons
- Are these methods equivalent?
  - No, they each lead to completely different solutions, with different costs
- For this course, we don't require you to come up with the best state assignment solution

# 3. State Assignment

- Are we using all of the combinations?
  - No. Some states are not designed to be reachable
  - Could also be used in the future for extensions

# 4. Flip-Flop Input Expressions

## 5. Output Expressions

- Express all Flip-Flops using input variables
- Express all outputs using variables and Flip-Flop outputs

$D_1D_0$  for next state  
 $S_1S_0$  for present

Present State $S_1S_0$	Next State $D_1D_0$		Output Z	
	X = 0	X = 1	X = 0	X = 1
A 00	00 A	B 01	0	0
B 01	00 A	C 10	0	0
C 10	11 D	C 10	0	0
D 11	00 A	B 01	0	1

# 4. Flip-Flop Input Expressions

## 5. Output Expressions

- Express all Flip-Flops using input variables
- Express all outputs using variables and Flip-Flop outputs

$D_1D_0$  for next state

$S_1S_0$  for present

$$D_1 = F_1(X, S_1, S_0) = \Sigma m(2, 5, 6)$$

$$D_0 = F_0(X, S_1, S_0) = \Sigma m(2, 4, 7)$$

$$Z = m_7$$

$X$	$S_1S_0$	$D_1D_0$	$Z$
0	00	00	0
0	01	00	0
0	10	11	0
0	11	00	0
1	00	01	0
1	01	10	0
1	10	10	0
1	11	01	1



# 6. Optimisation with Unused States

- Unused states can be implemented as don't care conditions

- In this example

$m_0, m_1, m_{12}, m_{13}, m_{14}, m_{15}$   
are unused, and can all be ***don't care conditions***

$$D_A = \Sigma m(5,7,8,9,11)$$

$$D_B = \Sigma m(3,4)$$

$$D_C = \Sigma m(2,4,6,8,10)$$

$$d = \Sigma m(0,1,12,13,14,15)$$

Present State			Input X	Next State		
A	B	C		A	B	C
0	0	1	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	0	1	1
0	1	0	1	1	0	0
0	1	1	0	0	0	1
0	1	1	1	1	0	0
1	0	0	0	1	0	1
1	0	0	1	1	0	0
1	0	1	0	0	0	1
1	0	1	1	1	0	0

# 6. Optimisation with Unused States

$$D_A = \Sigma m(5,7,8,9,11)$$

$$D_B = \Sigma m(3,4)$$

$$D_C = \Sigma m(2,4,6,8,10)$$

$$d = \Sigma m(0,1,12,13,14,15)$$

		C			
		00	01	11	10
A	00	X	X		
	01		1	1	
	11	X	X	X	X
	10	1	1	1	

Diagram illustrating a 4x4 Karnaugh map for variables A, B, and C. The map shows the following values:

- Row 00: (0,0)=X, (0,1)=X, (0,2)=, (0,3)=
- Row 01: (1,1)=1, (1,2)=1, (1,3)=
- Row 11: (2,0)=X, (2,1)=X, (2,2)=X, (2,3)=X
- Row 10: (3,0)=1, (3,1)=1, (3,2)=1, (3,3)=

Groupings are indicated by boxes:

- Group 1: (0,0), (0,1), (1,1), (1,2) - Labeled 'X' below.
- Group 2: (1,1), (1,2), (2,1), (2,2) - Labeled 'B' to the right.
- Group 3: (2,0), (2,1), (2,2), (2,3) - Labeled 'C' above.
- Group 4: (3,0), (3,1), (3,2), (3,3) - Labeled 'A' to the left.

		C			
		00	01	11	10
A	00	X	X	1	
	01	1			
	11	X	X	X	X
	10				

Diagram illustrating a 4x4 Karnaugh map for variables A, B, and C. The map shows the following values:

- Row 00: (0,0)=X, (0,1)=X, (0,2)=1, (0,3)=
- Row 01: (1,0)=1, (1,1)=, (1,2)=, (1,3)=
- Row 11: (2,0)=X, (2,1)=X, (2,2)=X, (2,3)=X
- Row 10: (3,0)=, (3,1)=, (3,2)=, (3,3)=

Groupings are indicated by boxes:

- Group 1: (0,0), (0,1), (1,0), (1,1) - Labeled 'X' below.
- Group 2: (0,0), (0,1), (1,0), (1,1) - Labeled 'B' to the right.
- Group 3: (2,0), (2,1), (2,2), (2,3) - Labeled 'C' above.
- Group 4: (3,0), (3,1), (3,2), (3,3) - Labeled 'A' to the left.

		C			
		00	01	11	10
A	00	X	X		1
	01	1			1
	11	X	X	X	X
	10	1			1

Diagram illustrating a 4x4 Karnaugh map for variables A, B, and C. The map shows the following values:

- Row 00: (0,0)=X, (0,1)=X, (0,2)=, (0,3)=1
- Row 01: (1,0)=1, (1,1)=, (1,2)=, (1,3)=1
- Row 11: (2,0)=X, (2,1)=X, (2,2)=X, (2,3)=X
- Row 10: (3,0)=1, (3,1)=, (3,2)=, (3,3)=1

Groupings are indicated by boxes:

- Group 1: (0,0), (0,1), (1,0), (1,1) - Labeled 'X' below.
- Group 2: (0,0), (0,1), (1,0), (1,1) - Labeled 'B' to the right.
- Group 3: (2,0), (2,1), (2,2), (2,3) - Labeled 'C' above.
- Group 4: (3,0), (3,1), (3,2), (3,3) - Labeled 'A' to the left.

# Systematic Design Procedures

## Sequential Circuits

1. **Specification**
2. **Formulation**  
e.g. using **state table** or **state diagram**
3. **State Assignment**: assign binary codes to states
4. **Flip-Flop Input Equation Determination**: Select flip-flop types, derive input equations from next-state entries
5. **Output Equation Determination**: Derive output equations from the output entries
6. **Optimisation**
7. **Technology Mapping**
8. **Verification**

# Summary

3. **State Assignment:** assign binary codes to states
4. **Flip-Flop Input Equation Determination:** Select flip-flop types, derive input equations from next-state entries
5. **Output Equation Determination:** Derive output equations from the output entries
6. **Optimisation with unused states**

# Some Other Flip-Flop Types

*JK* Flip-Flop; *T* Flip-Flop

# $T$ Flip-Flop

- Conditional Inverter

<b>T</b>	<b><math>Q(t + 1)</math></b>	<b>Operation</b>
0	$Q(t)$	No change
1	$\overline{Q}(t)$	Complement

# $T$ Flip-Flop

- Follow 8 step design principles
- Write down the boolean expression
- Draw the circuit diagram

<b>T</b>	<b><math>Q(t + 1)</math></b>	<b>Operation</b>
0	$Q(t)$	No change
1	$\bar{Q}(t)$	Complement

# $T$ Flip-Flop

3. State Assignment
4. Flip-Flop Input Equation
5. Output Equation Determination
6. Optimisation
7. Technology Mapping

<b>T</b>	<b><math>Q(t + 1)</math></b>	<b>Operation</b>
0	$Q(t)$	No change
1	$\overline{Q}(t)$	Complement



# $T$ Flip-Flop

3. State Assignment

4. Flip-Flop Input Equation

$$Q(t + 1) = Q \oplus T$$

5. Output Equation Determination

6. Optimisation

7. Technology Mapping

<b>T</b>	<b>Q(t + 1)</b>	<b>Operation</b>
0	$Q(t)$	No change
1	$\overline{Q}(t)$	Complement

# $T$ Flip-Flop

3. State Assignment

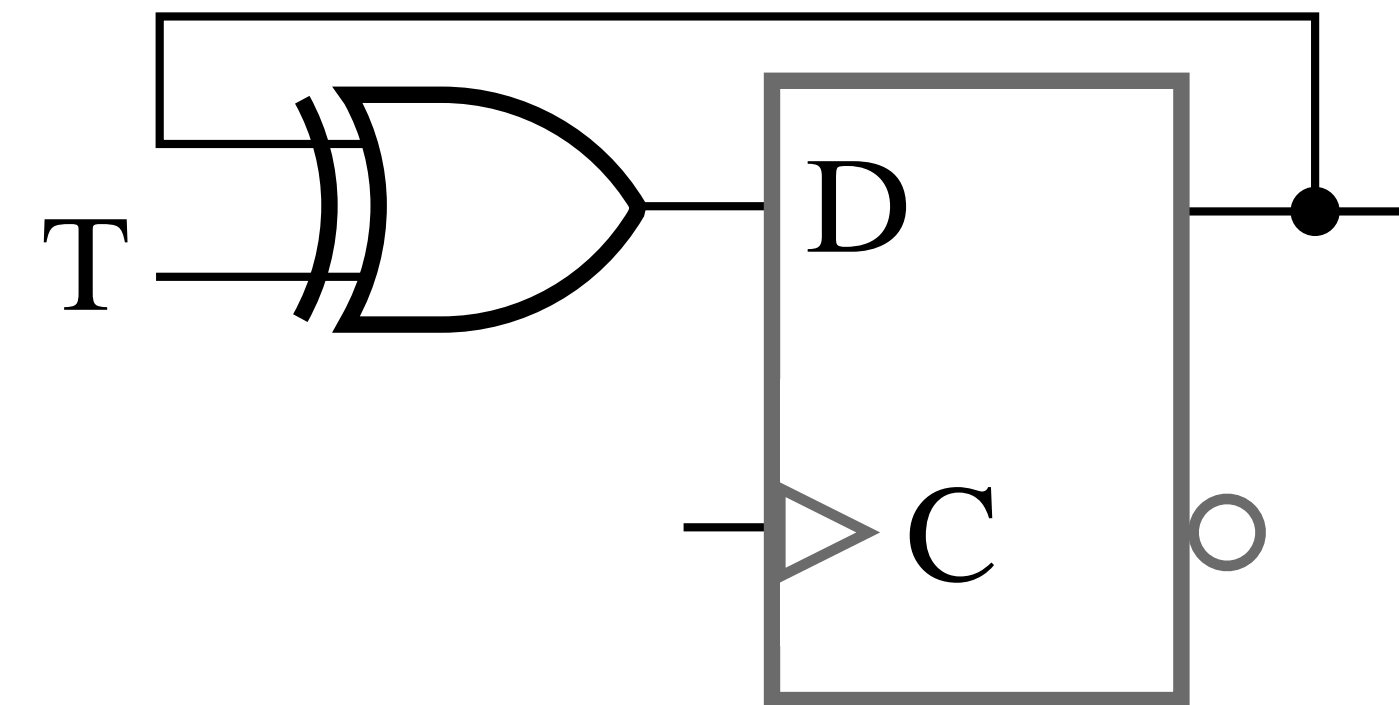
4. Flip-Flop Input Equation

$$Q(t + 1) = Q \oplus T$$

5. Output Equation Determination

6. Optimisation

7. Technology Mapping



<b>T</b>	<b><math>Q(t + 1)</math></b>	<b>Operation</b>
0	$Q(t)$	No change
1	$\overline{Q}(t)$	Complement

# *JK* Flip-Flop

- Similar to *SR* Master-Slave Flip-Flop with 11 input inverting internal value

<b>J</b>	<b>K</b>	<b>Q(t + 1)</b>	<b>Operation</b>
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$\overline{Q}(t)$	Complement

# *JK* Flip-Flop

- Follow 8 step design principles
- Write down the boolean expression
- Draw the circuit diagram

<b>J</b>	<b>K</b>	<b>Q(t + 1)</b>	<b>Operation</b>
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$\overline{Q}(t)$	Complement

# *JK* Flip-Flop

3. State Assignment

4. Flip-Flop Input Equation

5. Output Equation Determination

6. Optimisation

7. Technology Mapping

<b>J</b>	<b>K</b>	<b>Q(t + 1)</b>	<b>Operation</b>
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$\overline{Q}(t)$	Complement

# *JK* Flip-Flop

3. State Assignment

4. Flip-Flop Input Equation

$$Q(t + 1) = J \cdot \overline{Q} + \overline{K} \cdot Q$$

5. Output Equation Determination

6. Optimisation

7. Technology Mapping

<b>J</b>	<b>K</b>	<b>Q(t + 1)</b>	<b>Operation</b>
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$\overline{Q}(t)$	Complement

# *JK* Flip-Flop

3. State Assignment

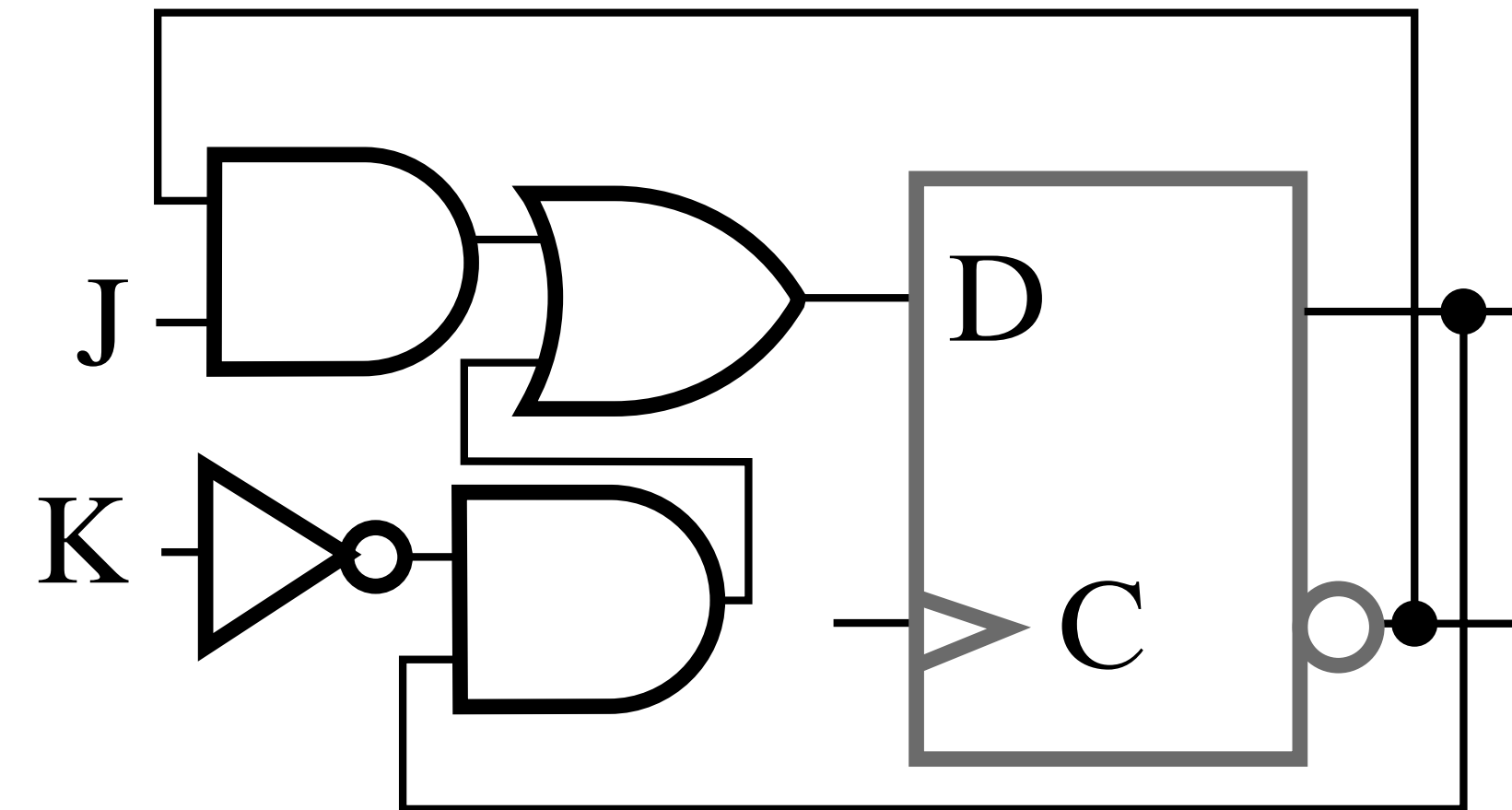
4. Flip-Flop Input Equation

$$Q(t + 1) = J \cdot \bar{Q} + \bar{K} \cdot Q$$

5. Output Equation Determination

6. Optimisation

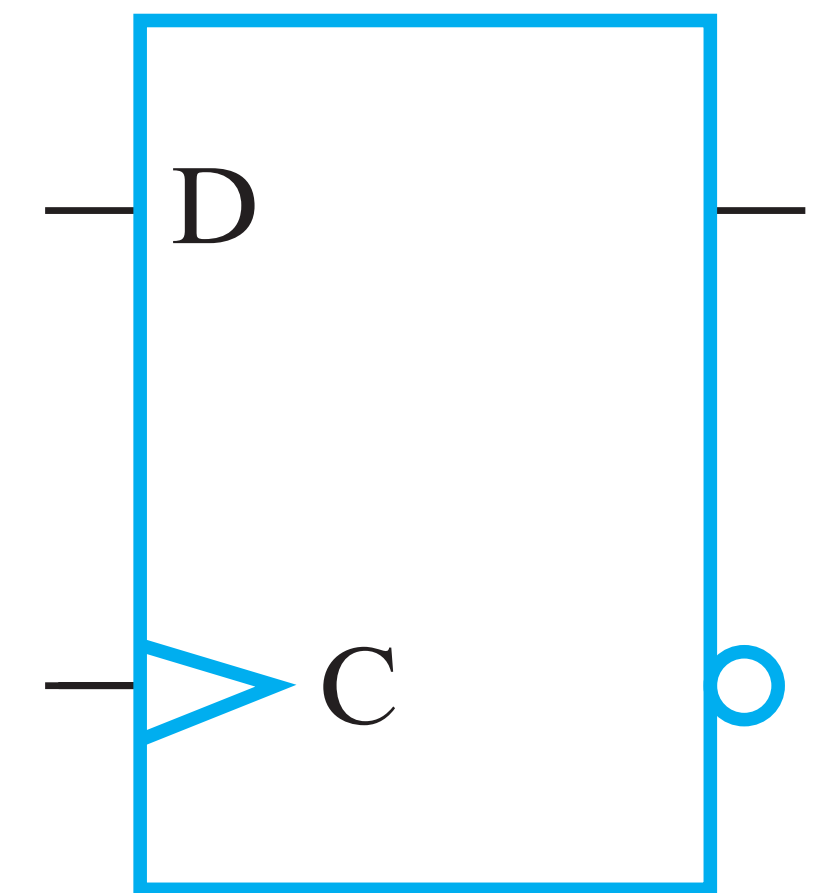
7. Technology Mapping



<b>J</b>	<b>K</b>	<b>Q(t + 1)</b>	<b>Operation</b>
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$\bar{Q}(t)$	Complement

# LogicWorks Exercise

- Implement  $D$  flip flop using  $D$  latch and  $SR$  latch  
Save it as a component in your library
- Implement circuit  $D_S = X \oplus Y \oplus S$ , where  $D_S$  is a  $D$  flip flop
- Implement  $D_A = \bar{X}A + XY$ ,  $D_B = \bar{X}B + XA$ ,  $Z = XB$
- Draw the state table and diagram, and verify your table with LogicWorks





# Implementation

- Implement  $JK$  Flip-Flop
  - Is there any other way to implement? What if you cannot use  $D$  Flip-Flop?
- Implement  $T$  Flip-Flop