CSCI 150
Introduction to Digital and Computer System Design
Lecture 4: Sequential Circuit V

Jetic Gū

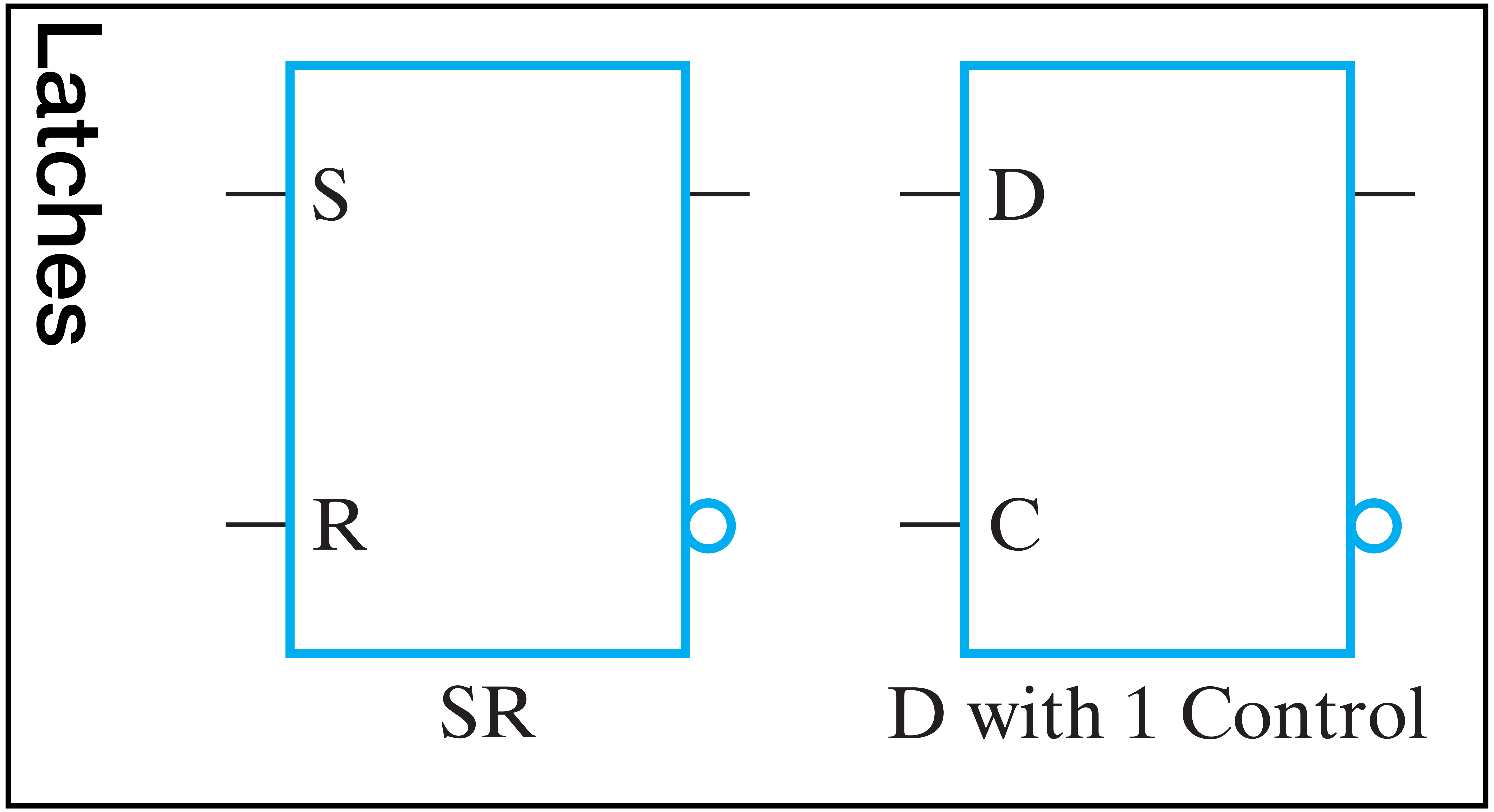
2020 Fall Semester (S3)

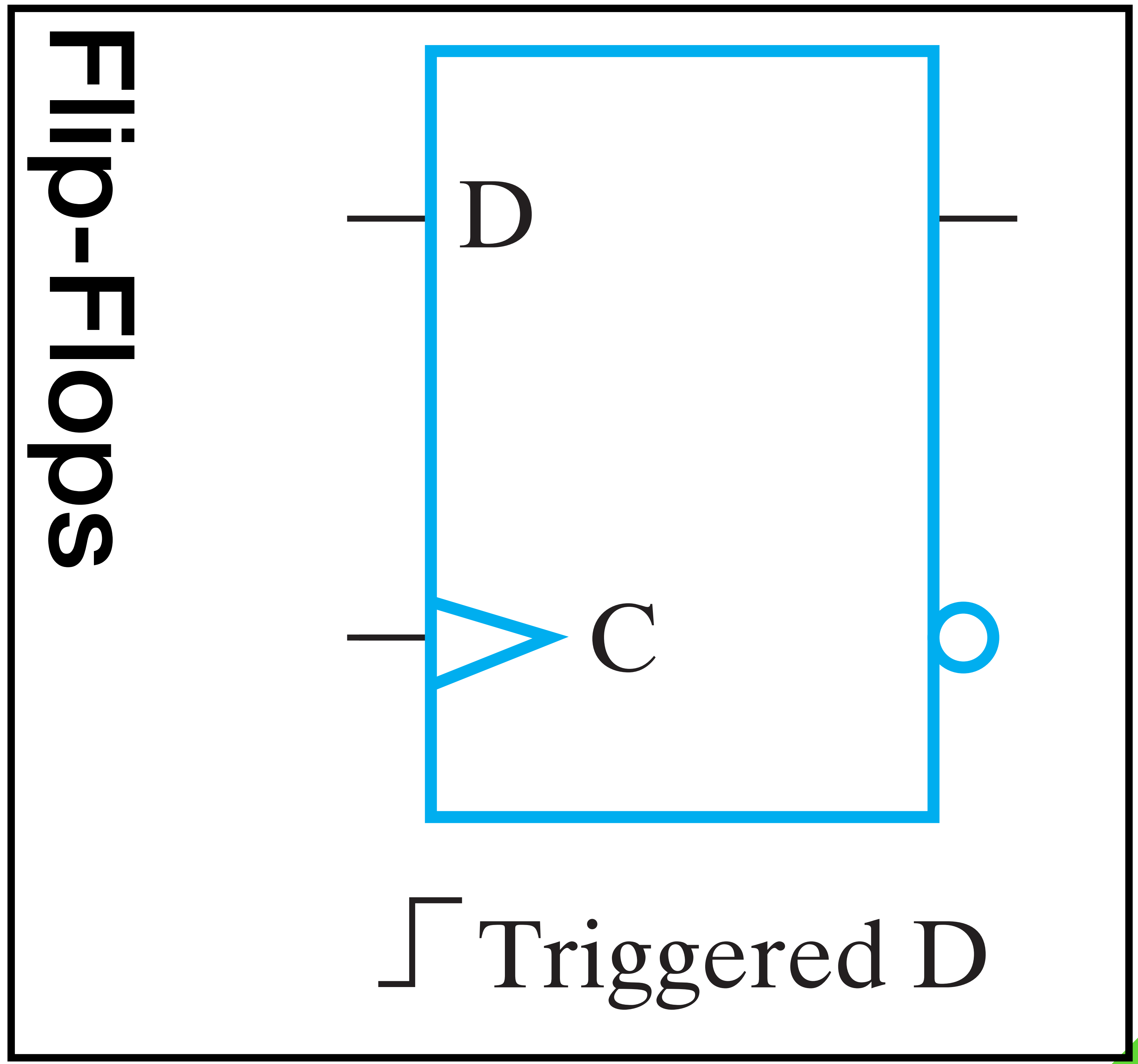
Overview

- 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

P0
Review

Latches and Flip-Flops





Review

P0
Review

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**

Review

Sequential Circuit Design II

State Assignment; Input Equation Determination; Output Equation Determination

P1
Design

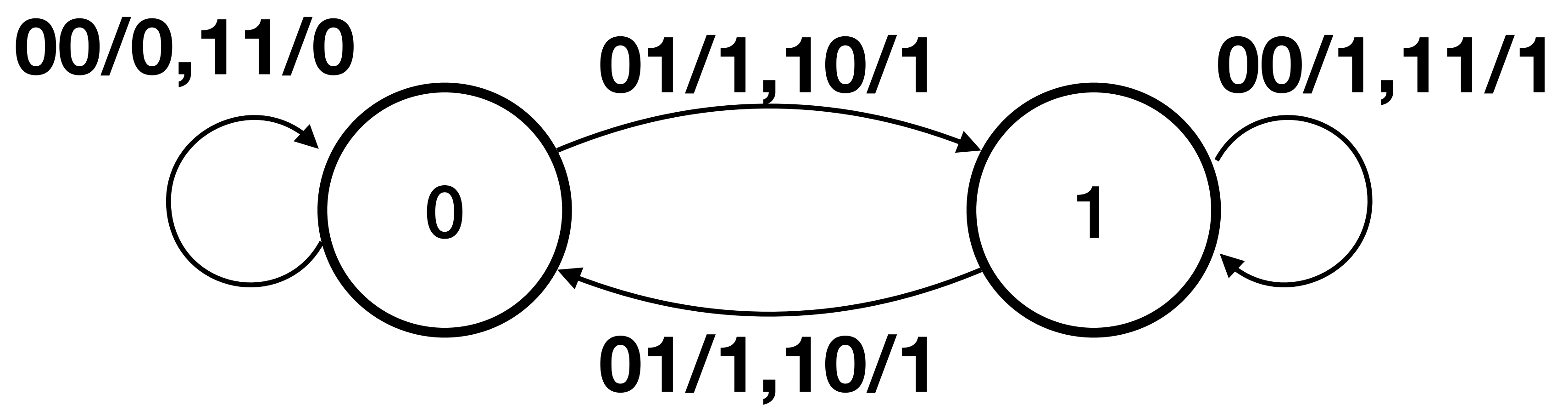
Systematic Design Procedures Sequential Circuits

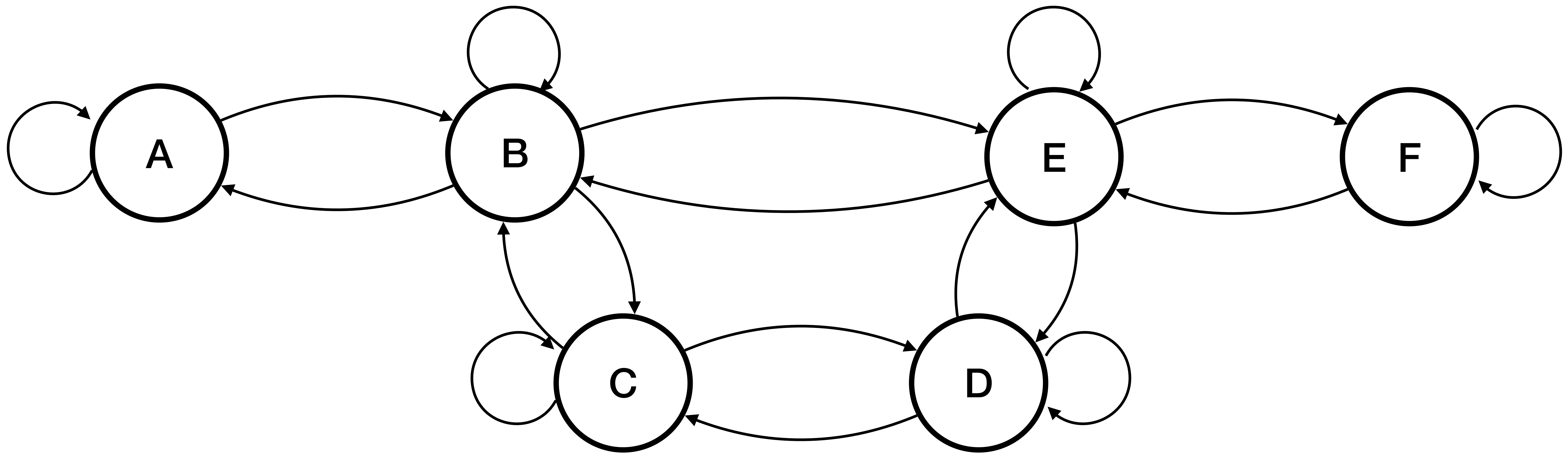
1. **Specification**
2. **Formulation**
   e.g. using state table or state diagram
3. **State Assignment**: assign binary codes to states
4. **Flip-Flop Input Equation Determination**: Select flip-flop types, derive input equations from next-state entries
5. **Output Equation Determination**: Derive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

Review

2. Formulation

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





2. Formulation

- Incrementer: perform +1 operation every `CLK` on 3-bit

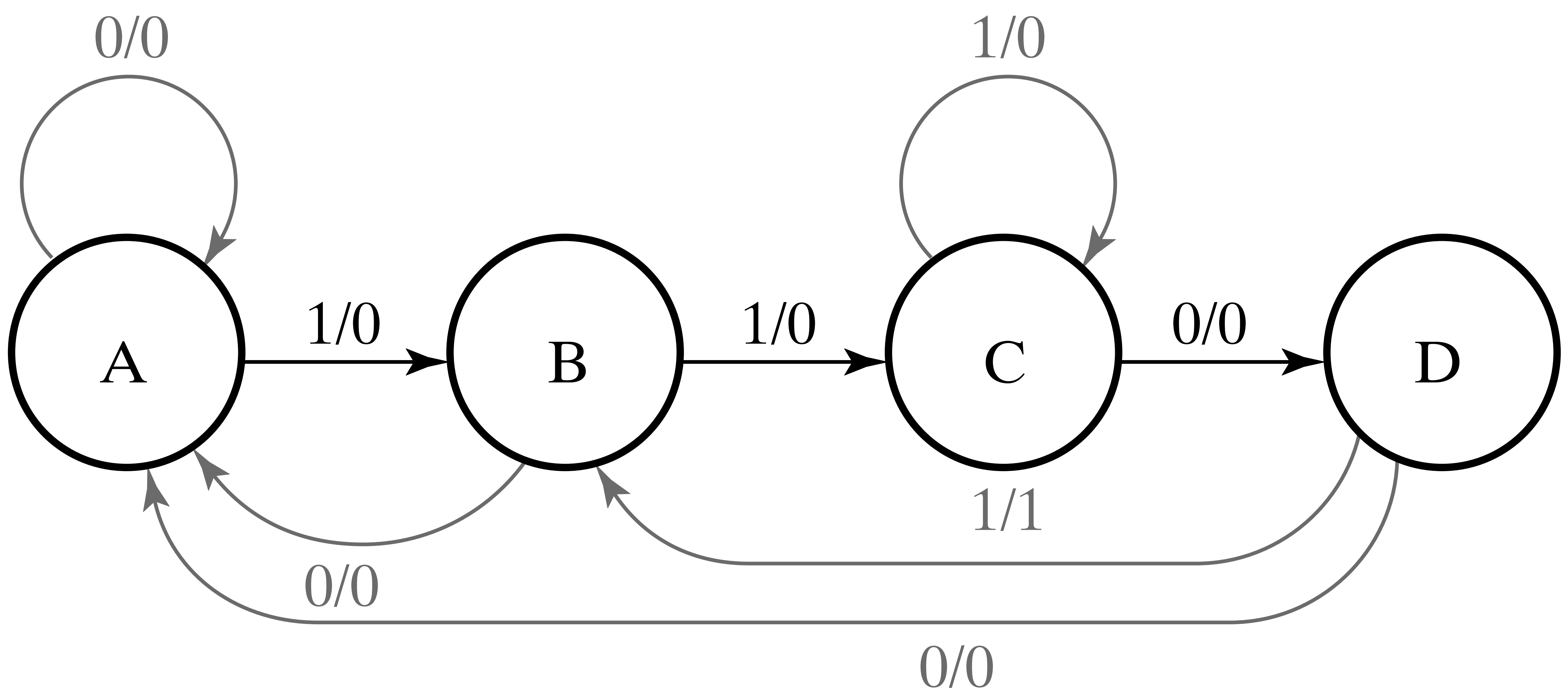
$$\begin{tikzcd}
000 \arrow[r] & 001 \arrow[r] & 010 \arrow[r] & 011 \arrow[d] \\
111 \arrow[u] & 110 \arrow[l] & 101 \arrow[l] & 100 \arrow[l]
\end{tikzcd}$$

Example

P1
Design

3. State Assignment

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



State Diagram

Concept

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

P1
Design

3. State Assignment

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

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

Concept

P1
Design

3. State Assignment

- Method 2: one hot
  $A = (0001)_2, B = (0010)_2, C = (0100)_2, D = (1000)_2$

Present State	Next State X = 0	Next State X = 1	Output Z X = 0	Output Z 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

Concept

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

P1
Design

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

Think

P1
Design

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

Concept

P1
Design

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

Concept

P1
Design

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***

$$D_A = \Sigma m(5,7,8,9,11)$$
$$D_B = \Sigma m(3,4)$$
$$D_C = \Sigma m(2,4,6,8,10)$$
$$d = \Sigma m(0,1,12,13,14,15)$$

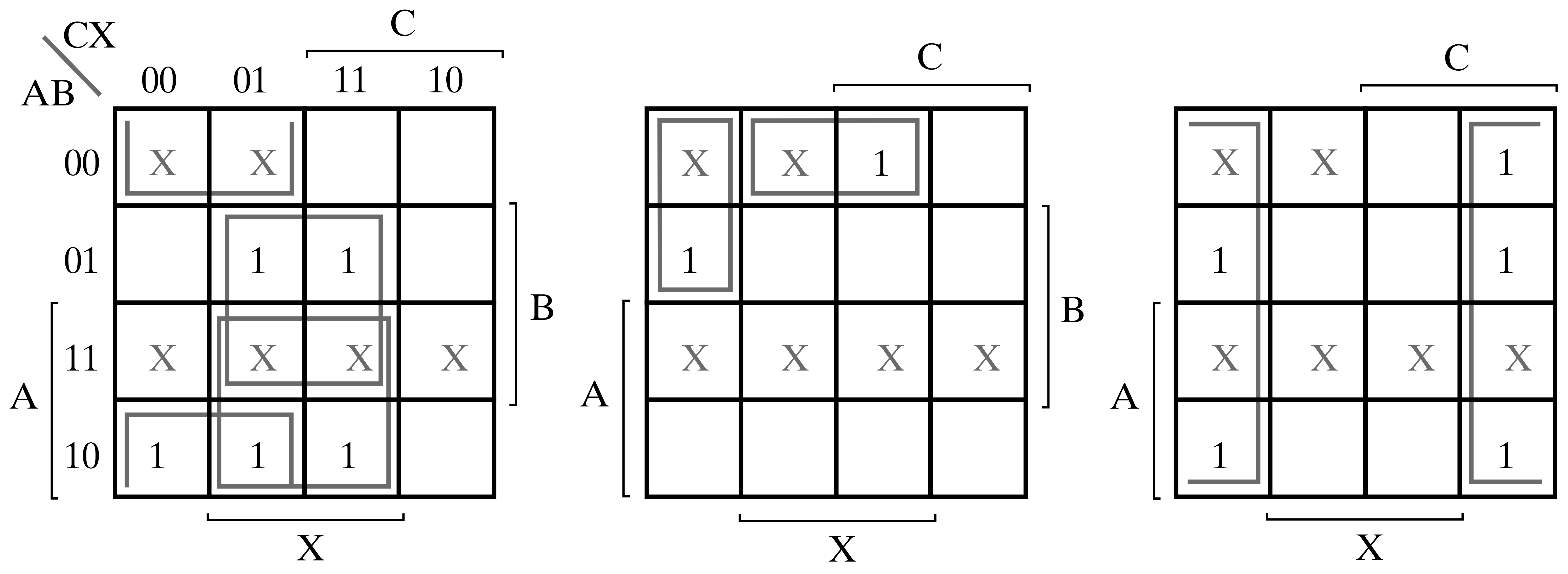
Present State			Input	Next State		
A	B	C	X	A	B	C
0	0	1	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	0	1	1
0	1	0	1	1	0	0
0	1	1	0	0	0	1
0	1	1	1	1	0	0
1	0	0	0	1	0	1
1	0	0	1	1	0	0
1	0	1	0	0	0	1
1	0	1	1	1	0	0

Example

P1
Design

6. Optimisation with Unused States

$$D_A = \Sigma m(5,7,8,9,11)$$
$$D_B = \Sigma m(3,4)$$
$$D_C = \Sigma m(2,4,6,8,10)$$
$$d = \Sigma m(0,1,12,13,14,15)$$



Example

P1
Design

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**

Concept

P1
Design

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**

Review

Some Other Flip-Flop Types

JK Flip-Flop; T Flip-Flop

T Flip-Flop

- Conditional Inverter

T	Q(t + 1)	Operation
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

T	**Q(t+1)**	**Operation**
0	$Q(t)$	No change
1	$\overline{Q}(t)$	Complement

Example

T Flip-Flop

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

T	**Q(t + 1)**	**Operation**
0	$Q(t)$	No change
1	$\overline{Q}(t)$	Complement

Example

T Flip-Flop

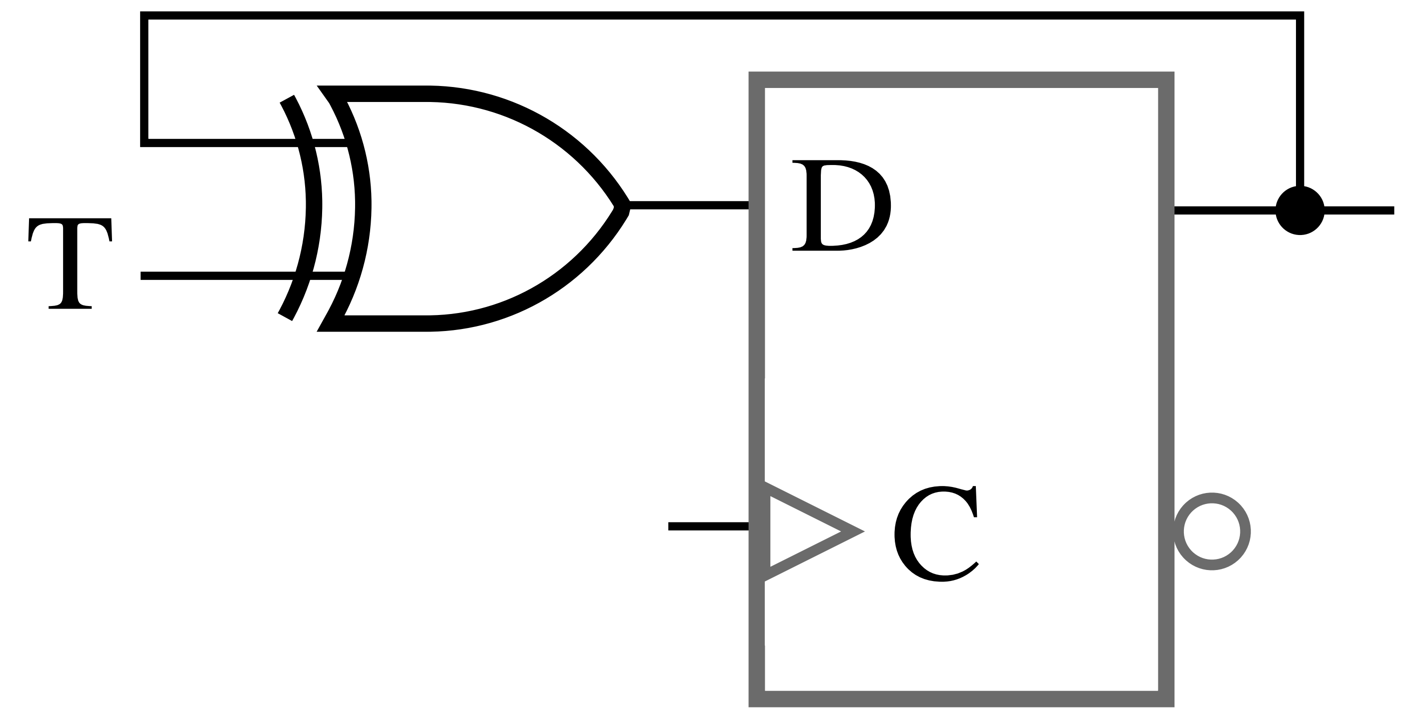
3. State Assignment
4. Flip-Flop Input Equation
   $Q(t+1) = Q \oplus T$
5. Output Equation Determination
6. Optimisation
7. Technology Mapping

T	**Q(t + 1)**	**Operation**
0	$Q(t)$	No change
1	$\overline{Q}(t)$	Complement

Example

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



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

Example

JK Flip-Flop

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

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

Concept

JK Flip-Flop

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

J	**K**	**Q(t + 1)**	**Operation**
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

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

Example

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

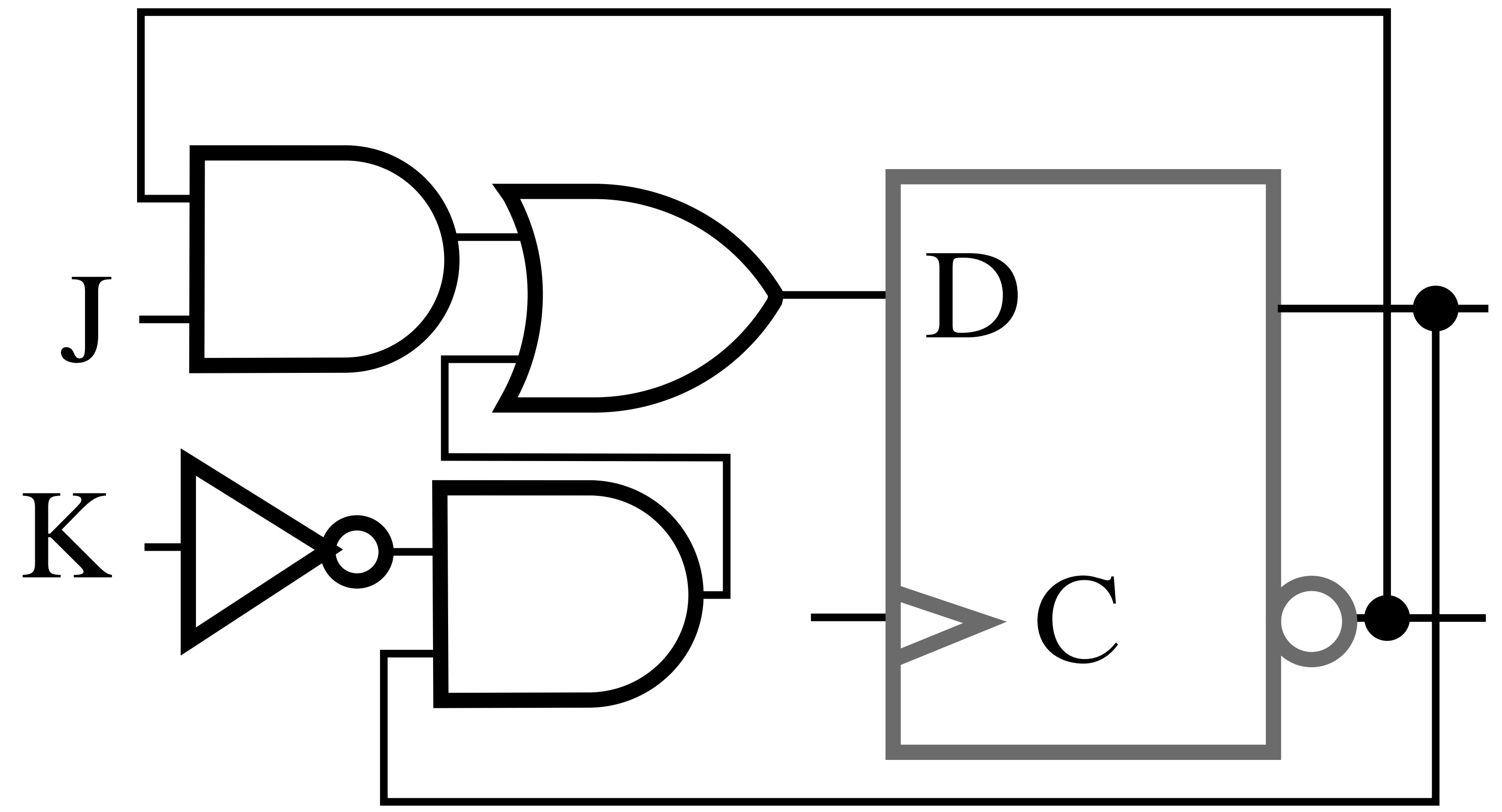
7. Technology Mapping

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

Example

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

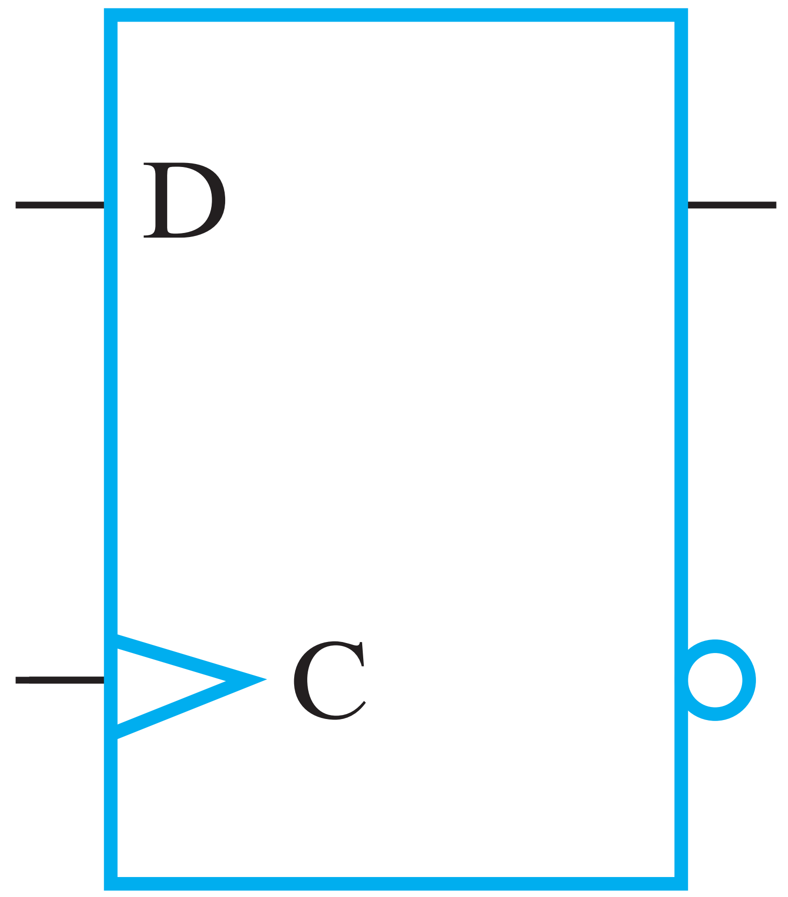


J	K	Q(t + 1)	Operation
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$\overline{Q}(t)$	Complement

Example

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 $D_A = \overline{X}A + XY$, $D_B = \overline{X}B + XA$, $Z = XB$
- Draw the state table and diagram, and verify your table with LogicWorks



Exercise

P2
Other Flip-Flop

Implementation

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

Technical