OF THE
DEPARTMENT OF
INFORMATION ENGINEERING

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Digital Systems

Synthesis of Sequential Circuits

Marta Bagatin, marta.bagatin@unipd.it

Degree Course in Information Engineering

Academic Year 2023-2024

Purpose of the Lesson

- Identify the **steps for the synthesis (= design) of a synchronous sequential circuit**
- See an example of sequential circuit design (sequence recognizer)

Steps for the Synthesis of Sequential Circuits

1) Identify/formulate the **specifications** of the circuit
2) Find the **state diagram**, starting from circuit specifications
3) Determine the **state table** starting from the state diagram (identify possible equivalent states and minimize the number of states)
4) Assign binary codes to the states and obtain an **encoded state table**
5) Select the type of flip-flop to be used (positive-edge-triggered D flip-flops)
6) Determine the **flip-flop input equations** from the next-state entries in the encoded state table
7) Determine the **output equations** from the output entries in the state table
8) **Minimize** the flip-flop input equations and circuit output equations
9) Draw the circuit and check the correct operation with a **time diagram** and the appropriate state transitions

Same steps as combinational circuit design!

How to Find the State Diagram

State Diagram Formulation

- Finding the **state diagram is the hardest (but also <u>creative</u>!) part** of the synthesis
- Most of the subsequent steps are performed automatically by computer-based tools
- Important notes
  - To correctly translate a verbal description of circuit specifications to a state diagram, an abstraction step is required
  - There is no algorithm: practice and attention (but also inventiveness) are needed!
- Intuitive ideas of the concept of **state**
  - The system state summarizes everything that happened in the past and that is relevant to determine the next state of the system
  - The state depends on the history of the inputs (sequence of inputs provided to the system in the past, at triggering clock edges)
  - The state is what the system needs to **remember its own history**

State as an Abstraction

- When formulating the state diagram, it can be very useful to **write down the description of each state in an abstract way**, with the aim of **describing the abstraction represented by the state**

Ex: State as Abstraction of an Incoming Sequence

- Example 1: A system is in state S_1 if '1' was received at the input bit X for 3 consecutive times (i.e. for 3 consecutive clock rising edges)
  - The system is in S_1 if '00111' or '10101111' was received as an input sequence
  - The system is NOT in S_1 if '00011' or '011100' was received as an input sequence

- Example 2: A system is in state S_2 if the combinations 00, 01, 11, 10 were received in this order at the inputs X_1 X_2, with any number of consecutive repetitions of each combination and with '10' as the last combination
  - The system is in S_2 if the sequences 00,00,01,01,01,11,10,10 or 00,01,11,11,11,10 were received as inputs
  - The system is NOT in S_2 if the sequences 00,11,10,10 or 00,00,01,01,11,11 were received as inputs

How Many States Should We Use?

- In formulating the state diagram, new states are added as new transitions are examined
- Potentially, every transition can lead to a new state, but it is important to **reuse existing states if possible**, to limit the number of states as much as possible
  - Example 1- Abstract description of state S_1: the system received the value '1' at input bit X for the last 3 consecutive times. Suppose the system arrived in S_1 having received 00111 sequence and receiving '1' as an input (=> 001111). Can we reuse S_1 as next state? YES! (as there are no constraints on the maximum number of ‘1’ at the input)
- Our purpose is to realize the system with as few states as possible: with a deep, abstract understanding of the states, it is easier to identify the correct number of states and avoid equivalent states

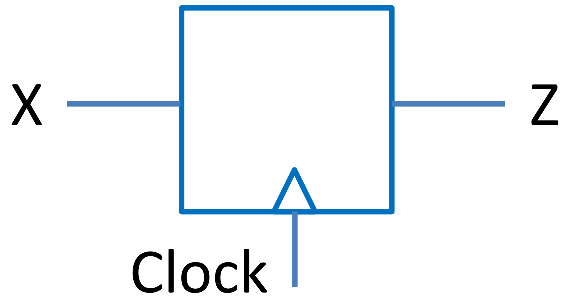
Reset

- Unlike combinational systems, **a sequential system is in an undefined state when the power is turned on** (the state of the flip-flops is not known)
- A sequential circuit needs to have a **known initial state**, before being able to operate correctly and output significant values. A **reset signal** is used to place the circuit in a known, initial state
  - Usually it is applied automatically when the circuit is turned on
  - A mechanism may also be provided to apply the reset, in case the system is in an error state, unknown or undefined

Example 4.3: Sequence Recognizer

- A sequence recognizer is a circuit capable of detecting the occurrence of a particular sequence of bits at the input, regardless of where it occurs in a longer sequence

- **Input (1 bit): X   Output (1 bit): Z   Recognizes the sequence 1101 (overlapping)**
  - Reset brings the system to the state «no symbol of the sequence was recognized»
  - Z = 1 if the previous 3 inputs were 110 and the current input is 1
  - Z = 0 otherwise



Example 4.3: Sequence Recognizer

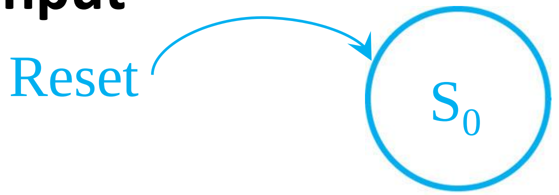
- **Input (1 bit): X   Output (1 bit): Z   Recognizes the sequence 1101 (overlapping)**
  - Reset brings the system to the state «no symbol of the sequence was recognized»
  - Z = 1 if the previous 3 inputs were 110 and the current input is 1
  - Z = 0 otherwise

- **First step: Mealy or Moore model**? The formulation of the problem says «Z = 1 if the previous 3 inputs were 110 and the current input is 1», meaning that the output depends not only on the present state, but also on the input: we need a Mealy model

Example 4.3: Sequence Recognizer

- **Input (1 bit): X** **Output (1 bit): Z** **Recognizes the sequence 1101 (overlapping)**
  - Reset brings the system to the state «no symbol of the sequence was recognized»
  - Z = 1 if the previous 3 inputs were 110 and the current input is 1
  - Z = 0 otherwise

- We start from an **arbitrary state S_0**, where the system arrives **after the reset signal**: we define S_0 as **"No symbol of the sequence was recognized at the input"**

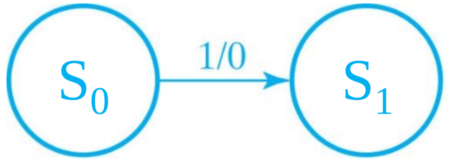


- Now let's **identify the states**, keeping in mind that each state needs to be able to remember the input history (even better, the part of the history we are interested in). We start considering at the input exactly the symbols of the sequence

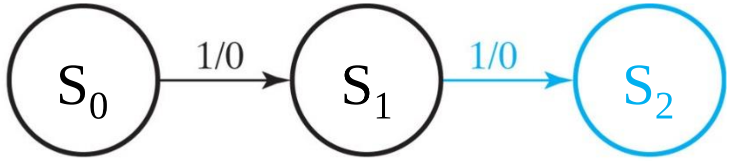
Example 4.3: Sequence Recognizer

- **Input (1 bit): X Output (1 bit): Z Recognizes the sequence 1101 (overlapping)**
  - Reset brings the system to the state «no symbol of the sequence was recognized»
  - Z = 1 if the previous 3 inputs were 110 and the current input is 1
  - Z = 0 otherwise

- If the system is in state S_0 and 1 occurs at the input, **state S_1** is defined as **"The first bit of the sequence (1) has been recognized"**


$$\begin{tikzcd} S_0 \arrow[r, "1/0"] & S_1 \end{tikzcd}$$

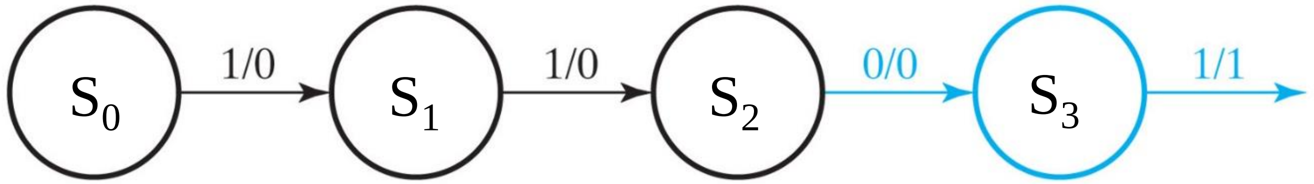
- If the system is in state S_1 and 1 occurs at the input, state **S_2** is defined as **"The first 2 bits of the sequence (11) have been recognized**"


$$\begin{tikzcd} S_0 \arrow[r, "1/0"] & S_1 \arrow[r, "1/0"] & S_2 \end{tikzcd}$$

Example 4.3: Sequence Recognizer

- **Input (1 bit): X   Output (1 bit): Z   Recognizes the sequence 1101 (overlapping)**

- If the system is in state S_2 and 0 occurs at the input, state $\mathbf{S_3}$ is defined as **"The first 3 bits of the sequence 110 have been recognized"**
- Finally, if the system is in state S_3 and 1 occurs at the input, the output becomes 1

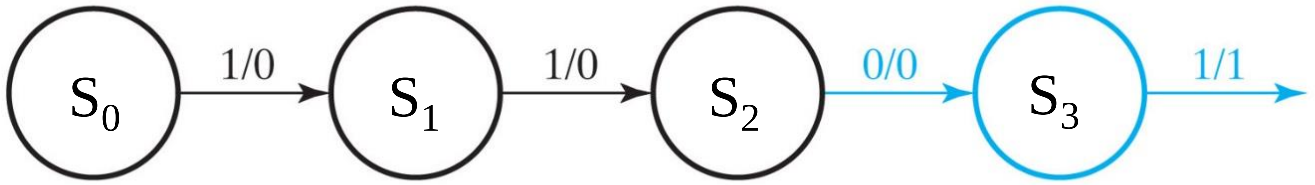


$$\begin{tikzcd} S_0 \arrow[r, "1/0"] & S_1 \arrow[r, "1/0"] & S_2 \arrow[r, "0/0"] & S_3 \arrow[r, "1/1"] & {} \end{tikzcd}$$

Example 4.3: Sequence Recognizer

- **Input (1 bit): X   Output (1 bit): Z   Recognizes the sequence 1101 (overlapping)**

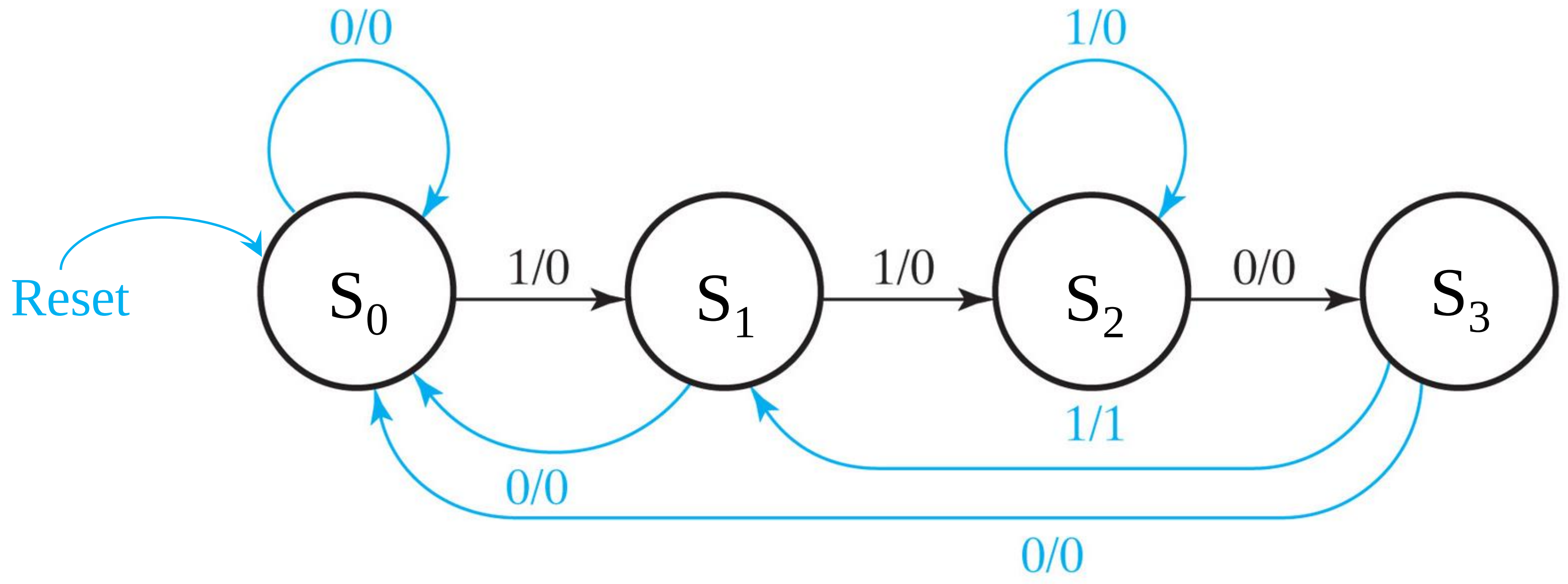
- So far, we have constructed a partial state diagram (only the transitions corresponding to the correct sequence were considered)


$$\begin{tikzcd} S_0 \arrow[r, "1/0"] & S_1 \arrow[r, "1/0"] & S_2 \arrow[r, "0/0"] & S_3 \arrow[r, "1/1"] & {} \end{tikzcd}$$

- Now we need to complete the diagram for the other transitions
  - State S_0, input 0, the system stays in S_0 ("No bits of the sequence were recognized")
  - State S_1, input 0, the system goes back to S_0 ("No bits of the sequence were recognized")
  - State S_2, input 1, the system stays in S_2 ("2 bits of the sequence have been recognized")
  - State S_3, input 0, the system goes back to S_0 ("No bits of the sequence are recognized")
  - State S_3, input 1, the system goes to S_1 ("1 bit of the sequence was recognized")

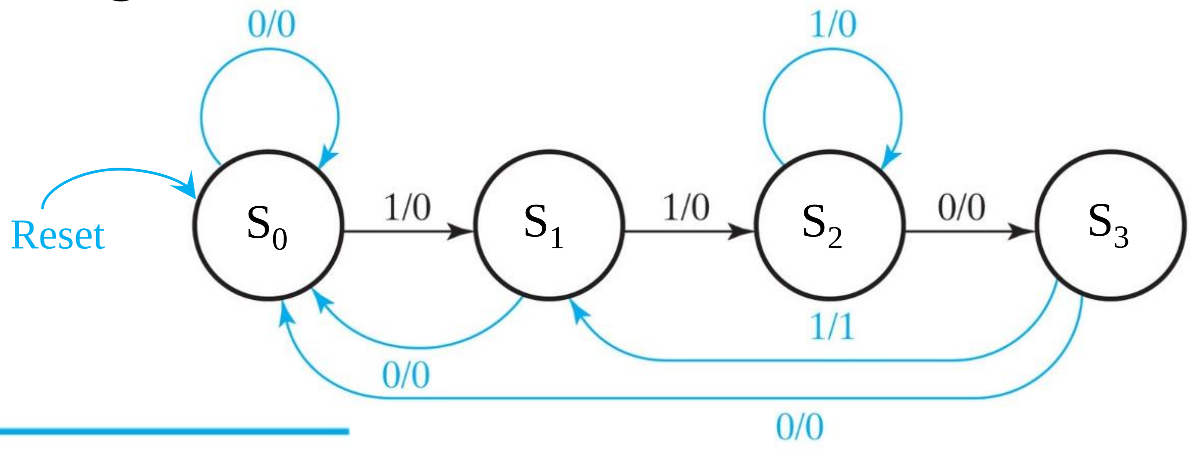
Example 4.3: Sequence Recognizer

- **Input (1 bit): X   Output (1 bit): Z   Recognizes the sequence 1101 (overlapping)**
- The **complete state diagram** is:



Example 4.3: Sequence Recognizer

- **Input (1 bit): X**   **Output (1 bit): Z**   **Recognizes the sequence 1101 (overlapping)**



- Let's derive the state table

Present State	Next State X = 0	Next State X = 1	Output Z X = 0	Output Z X = 1
S_0	S_0	S_1	0	0
S_1	S_0	S_2	0	0
S_2	S_3	S_2	0	0
S_3	S_0	S_1	0	1

State Assignments

State Assignment

- So far, we have indicated the states with symbolic names S_i. Now we will assign a binary code to each state
- We need at least n bits to represent **m states** with $2^n \geq m$ ($n = \lceil \log_2 m \rceil$)
  - Different codes can be chosen and there are **different methods for states assignment**, some of them are quite complex (e.g. to obtain a two-level combinational circuit)
  - Using fewer bits for the state (therefore fewer flip-flops) does not always minimize the cost of the entire circuit (in addition to flip-flops, also the cost of the combinational part must be considered)
  - Warning: the optimization of state encoding is a complex problem and it is often solved with heuristic methods, there is no algorithm that guarantees the best result
- We will see two examples (then we will compare the costs of the two choices) of state assignment for the sequence recognizer:
  1. Gray encoding
  2. 1-hot encoding

State Assigment: 2 Methods Compared

i. **Gray code** assigment
   - It is easier to enter the next state and output into Karnaugh maps
   - The minimum number of bits is used for the representation of states, so the number of flip-flops in the circuit is minimized

S_0= 00
S_1= 01
S_2= 11
S_3= 10

ii. **1-hot code** assigment
   - A single 1 appears in the code of each state, so the logic for entering each state is separate from that for entering the other states
   - A larger number of flip-flops is used, but the combinational logic is generally simpler

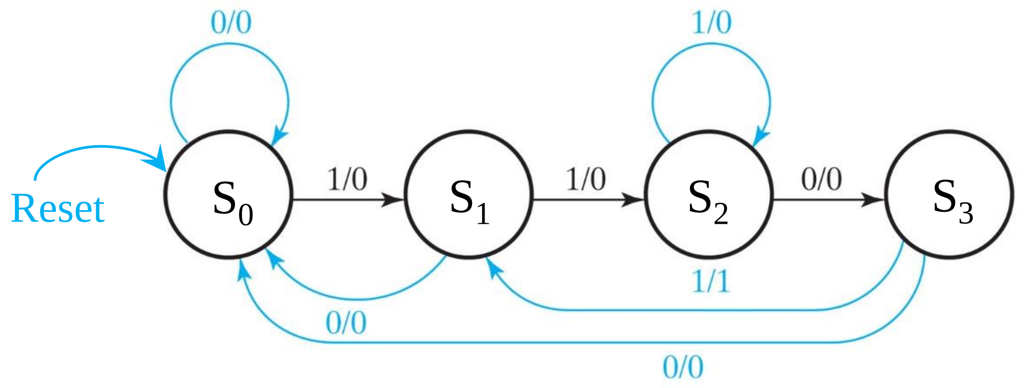
S_0= 1000
S_1= 0100
S_2= 0010
S_3= 0001

Example 4.3: Sequence Recognizer

Input (1 bit): X

Output (1 bit): Z

Recognizes the sequence 1101



i. States with **Gray binary code** (2 bits → A, B), 2 flip-flops are needed:

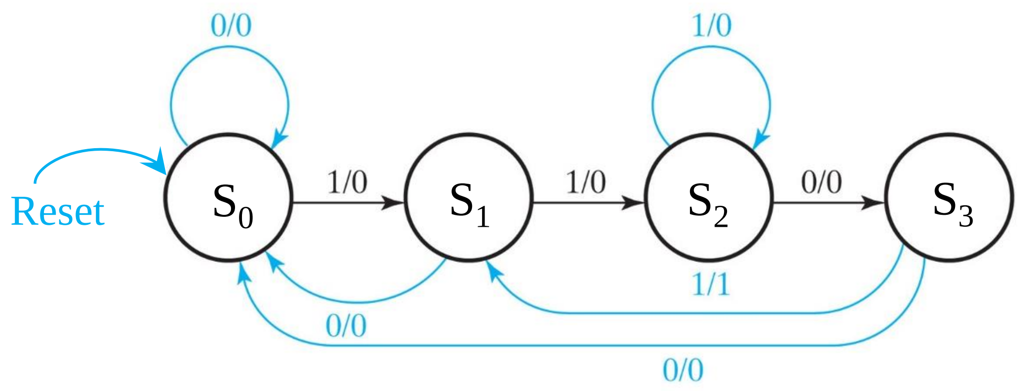
Present State	Next State		Output Z	
AB	X = 0	X = 1	X = 0	X = 1
00	00	01	0	0
01	00	11	0	0
11	10	11	0	0
10	00	01	0	1

Example 4.3: Sequence Recognizer

Input (1 bit): X

Output (1 bit): Z

Recognizes the sequence 1101



ii. States with **1-hot code** (4 bits → A, B, C, D), 1 flip-flop for each state:

Present State	Next State		Output Z	
ABCD	X = 0	X = 1	X = 0	X = 1
1000	1000	0100	0	0
0100	1000	0010	0	0
0010	0001	0010	0	0
0001	1000	0100	0	1

Design of the Circuit with D Flip-flops

Implementation of the Sequential Circuit

- Once we have found the state diagram (and/or state table) and assigned codes to the states, the rest of the procedure consists in designing two **combinational circuits**
  - A combinational circuit to determine the **next state**
  - A combinational circuit to determine the **output**

Example 4.3: Sequence Recognizer

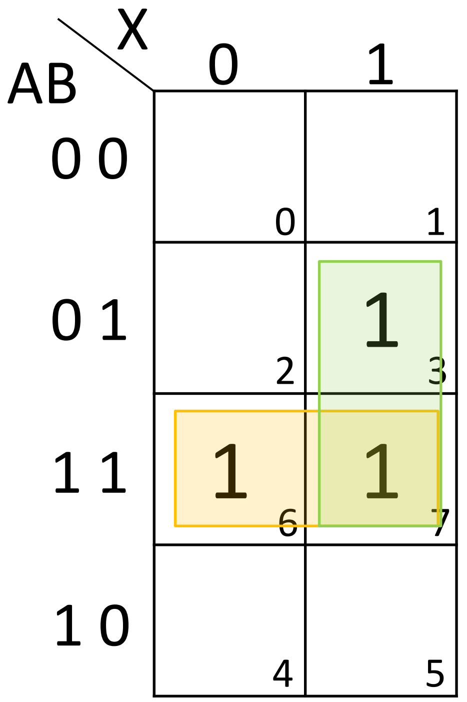
Input (1 bit): X

Output (1 bit): Z

Recognizes the sequence 1101

i. States represented with binary Gray code:

Present State	Next State		Output Z	
AB	X = 0	X = 1	X = 0	X = 1
00	00	01	0	0
01	00	11	0	0
11	10	11	0	0
10	00	01	0	1



Let's find the **next state equations** (2 state bits => 2 flip-flops)

- 1st bit of the state (A):

$$A\,(t+1) = D_A(A,B,X) = \sum m(3,6,7)$$

$$\Rightarrow D_A = AB + BX$$

Example 4.3: Sequence Recognizer

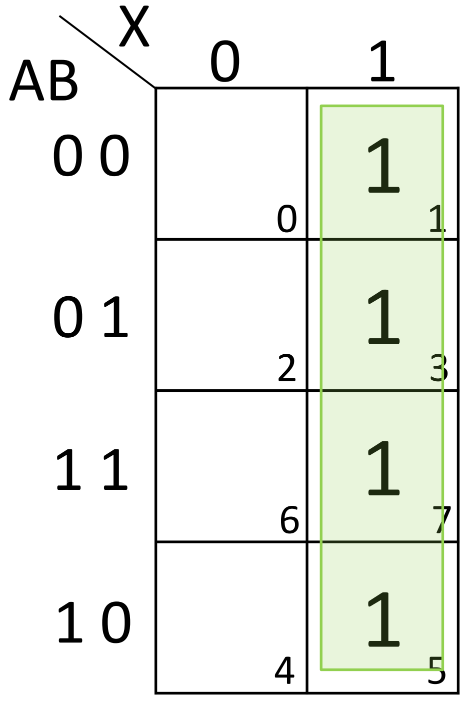
Input (1 bit): X

Output (1 bit): Z

Recognizes the sequence 1101

i. States represented with binary Gray code:

Present State	Next State		Output Z	
AB	**X = 0**	**X = 1**	**X = 0**	**X = 1**
00	00	01	0	0
01	00	11	0	0
11	10	11	0	0
10	00	01	0	1



- 2nd bit of the state (B):

$$B\,(t+1) = D_B(A, B, X) = \sum m(1, 3, 5, 7)$$

$$\Rightarrow D_B = X$$

Example 4.3: Sequence Recognizer

Input (1 bit): X

Output (1 bit): Z

Recognizes the sequence 1101

AB \ X	0	1
0 0	$_0$	$_1$
0 1	$_2$	$_3$
1 1	$_6$	$_7$
1 0	$_4$	1 $_5$

i. States represented with binary Gray code:

Present State	Next State		Output Z	
AB	X = 0	X = 1	X = 0	X = 1
00	00	01	0	0
01	00	11	0	0
11	10	11	0	0
10	00	01	0	1

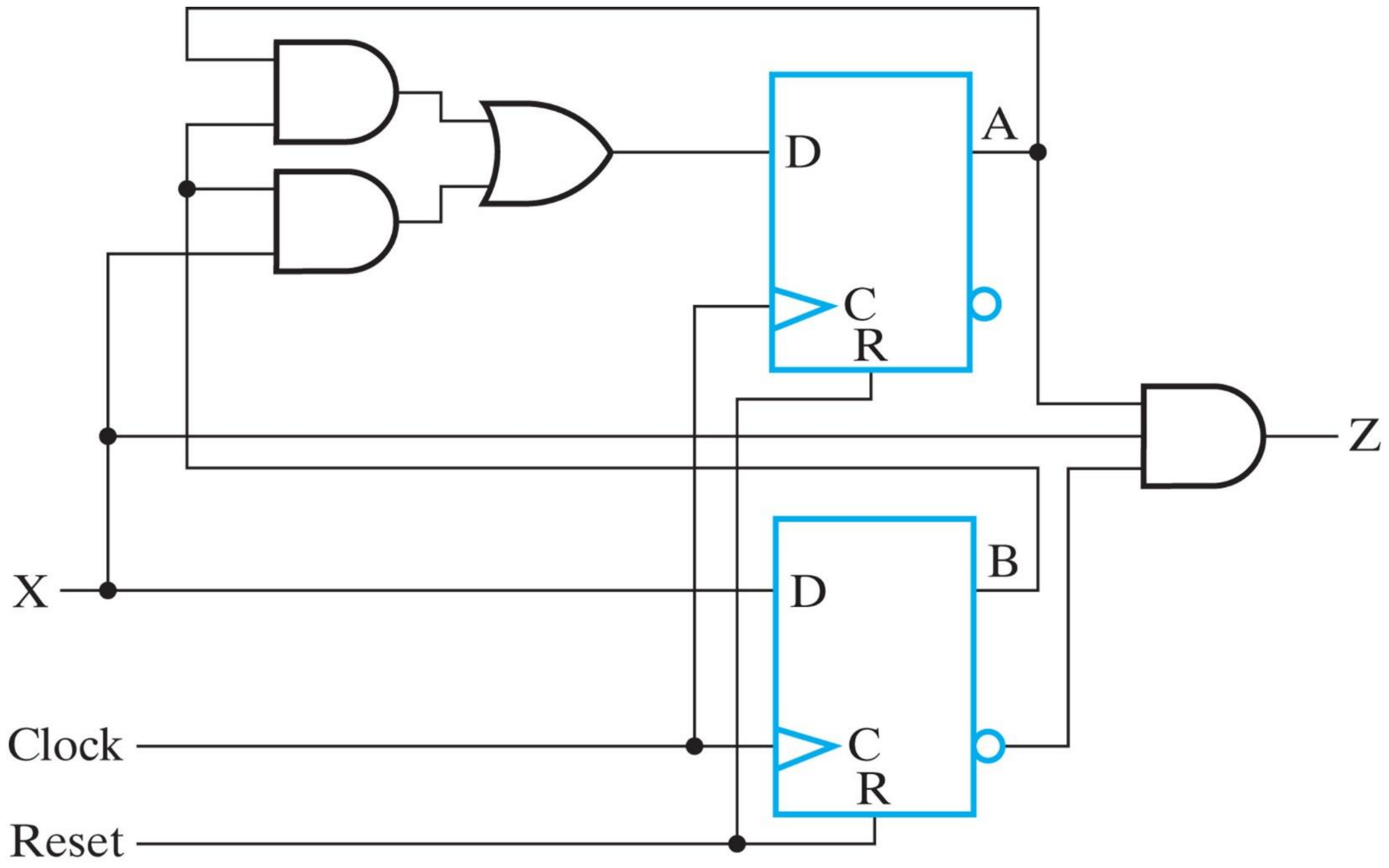
- Let's find the output equation:

$$Z\,(A, B, X) = \sum m(\,5)$$

$$\Rightarrow Z = A\bar{B}X$$

Example 4.3: Sequence Recognizer

Final circuit (states represented with binary Gray code)
2 flip-flops are needed to store the state



$D_A = AB + BX$

$D_B = X$

$Z = A\overline{B}X$

Cost (comb) = 9
Cost (flip-flops) = 14 * 2
Total cost = 37

Example 4.3: Sequence Recognizer

Input (1 bit): X

Output (1 bit): Z

Recognizes the sequence 1101

ii. States represented with 1-hot codes:

Present State	Next State		Output Z	
ABCD	$X = 0$	$X = 1$	$X = 0$	$X = 1$
1000	1000	0100	0	0
0100	1000	0010	0	0
0010	0001	0010	0	0
0001	1000	0100	0	1

Let's find the next state equations (4 state bits => 4 flip-flops):

$$A(t+1) = D_A = A\overline{X} + B\overline{X} + D\overline{X} = (A + B + D)\,\overline{X}$$

$$B(t+1) = D_B = AX + DX = (A + D)\,X$$

$$C(t+1) = D_C = BX + CX = (B + C)\,X$$

$$D(t+1) = D_D = C\overline{X}$$

No need to use K-maps, equations are simple to write: just one bit is enough to identify the present state

Example 4.3: Sequence Recognizer

ii. States represented with 1-hot codes:

Input (1 bit): X

Output (1 bit): Z

Recognizes the sequence 1101

Present State	Next State		Output Z	
ABCD	$X = 0$	$X = 1$	$X = 0$	$X = 1$
1000	1000	0100	0	0
0100	1000	0010	0	0
0010	0001	0010	0	0
0001	1000	0100	0	1

Let's find the output equation:

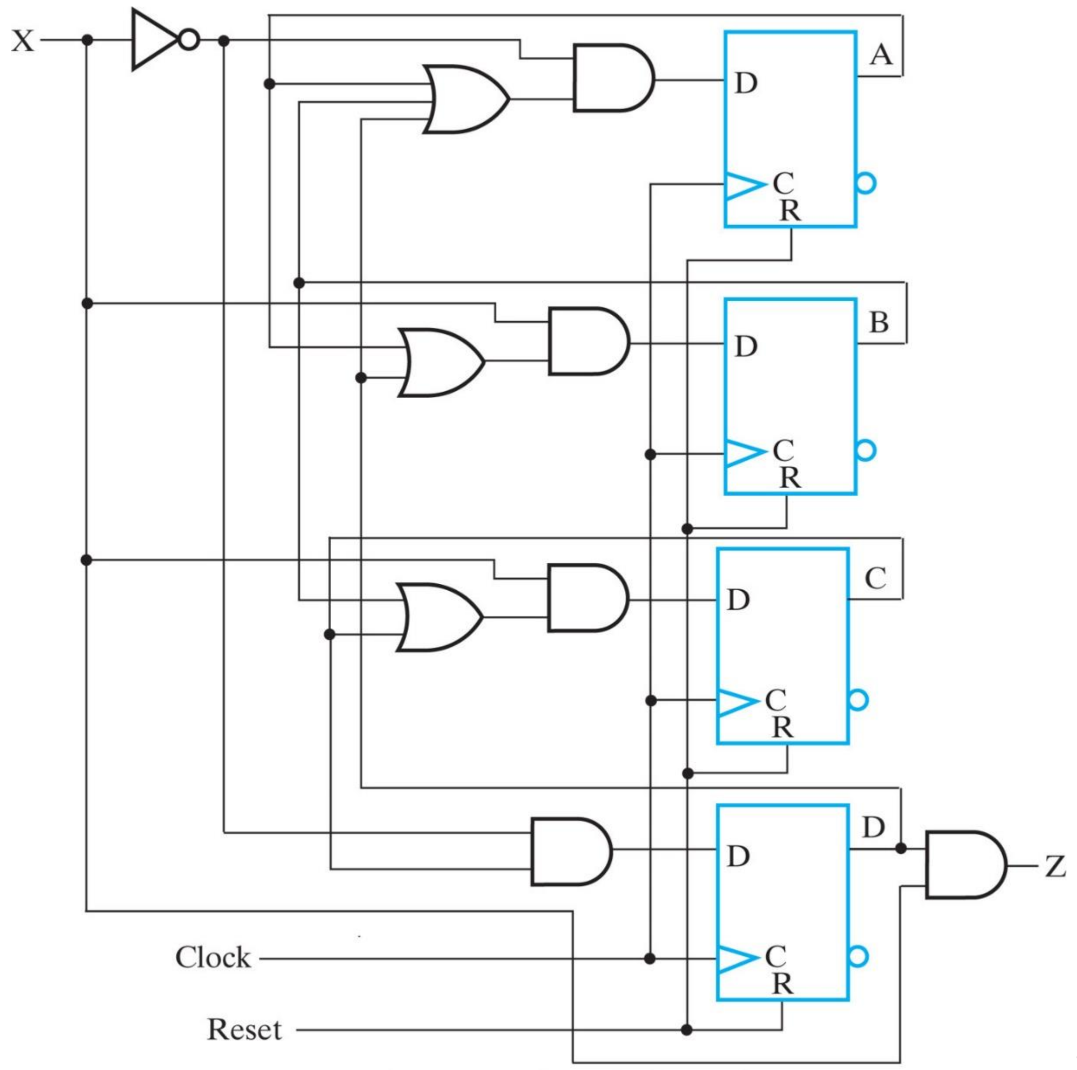
$$Z = DX$$

Example 4.3: Sequence Recognizer

Final circuit (states represented with 1-hot code)
4 flip-flops are needed, 1 FF for each state

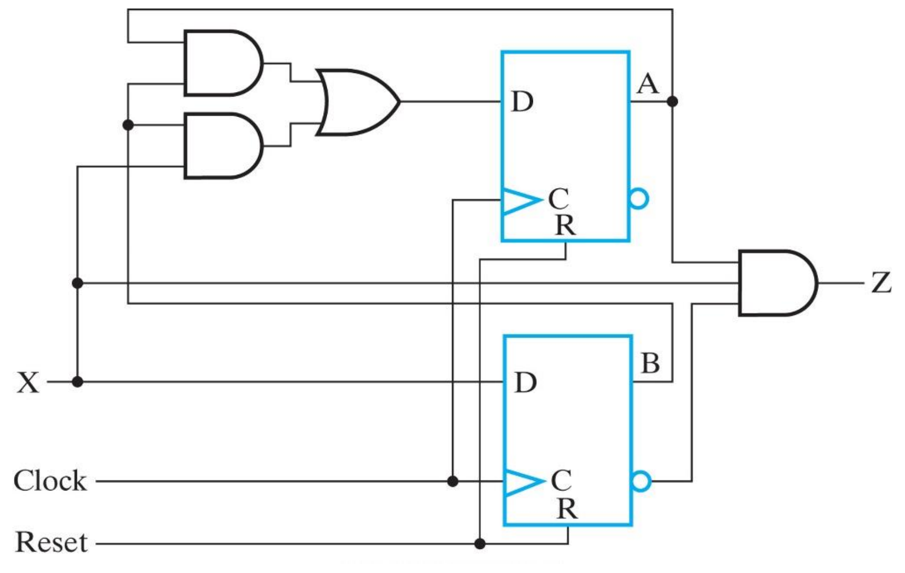
$$D_A = (A + B + D)\,\overline{X}$$
$$D_B = (A + D)\,X$$
$$D_C = (B + C)\,X$$
$$D_D = C\overline{X}$$
$$Z = DX$$

Cost (comb) = 18
Cost (flip-flops) = 14 * 4
Total cost = 74



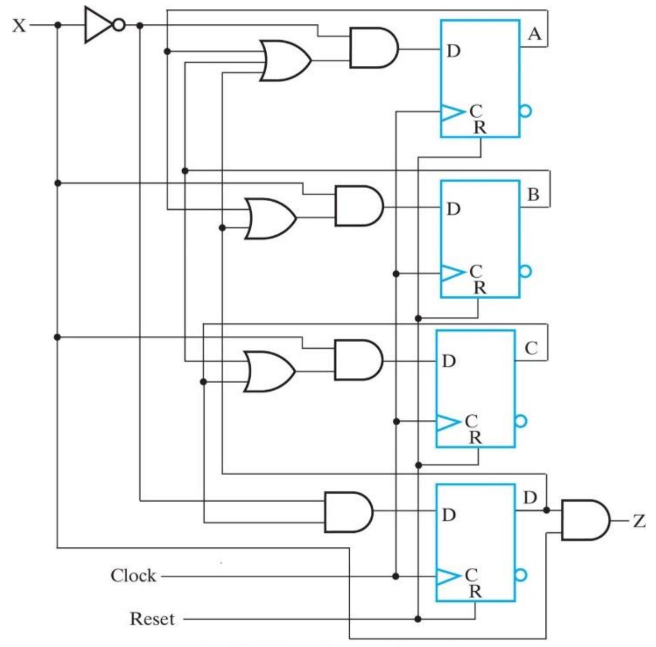
Example 4.3: Sequence Recognizer

States represented with Gray code:



Total cost = 37

States represented with 1-hot code:



Total cost = 74

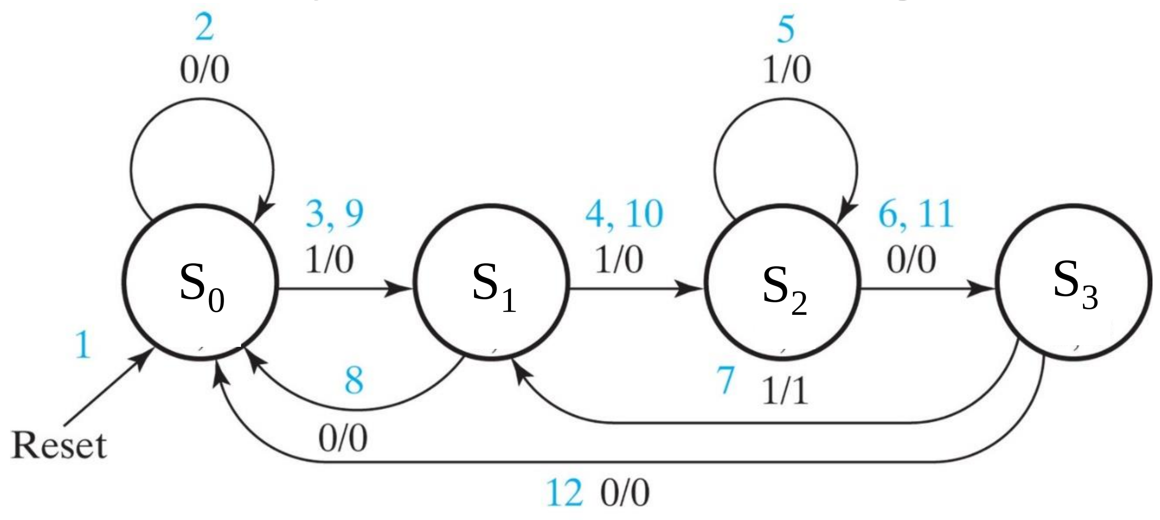
- With Gray code, the cost is almost halved (in terms of gate input number)
- With 1-hot code, there might be advantages in some cases (e.g. easier design, higher reliability, better circuit performance)

Verification

Verification of a Sequential Circuit

- The purpose is to verify that the **circuit behaves as expected, producing the original state diagram** (or state table)
  - Different combinations of the inputs are applied, then the state variables and outputs are observed at the appropriate instant of times (relative to clock changes), to check that the system produces the expected transitions
  - At the beginning of the verification process, a reset is applied to put the circuit in its initial state
- For simple enough circuits, all the present state/input combinations can be applied, in order to **check all possible transitions**
- For more complex circuits, usually only **some significant test vectors** are applied
- In the real world, verification is a very complex activity that makes use of automatic tools and techniques that go beyond the scope of this course

Example 4.3: Sequence Recognizer



Sequence of inputs to apply for circuit verification (to verify all combinations):

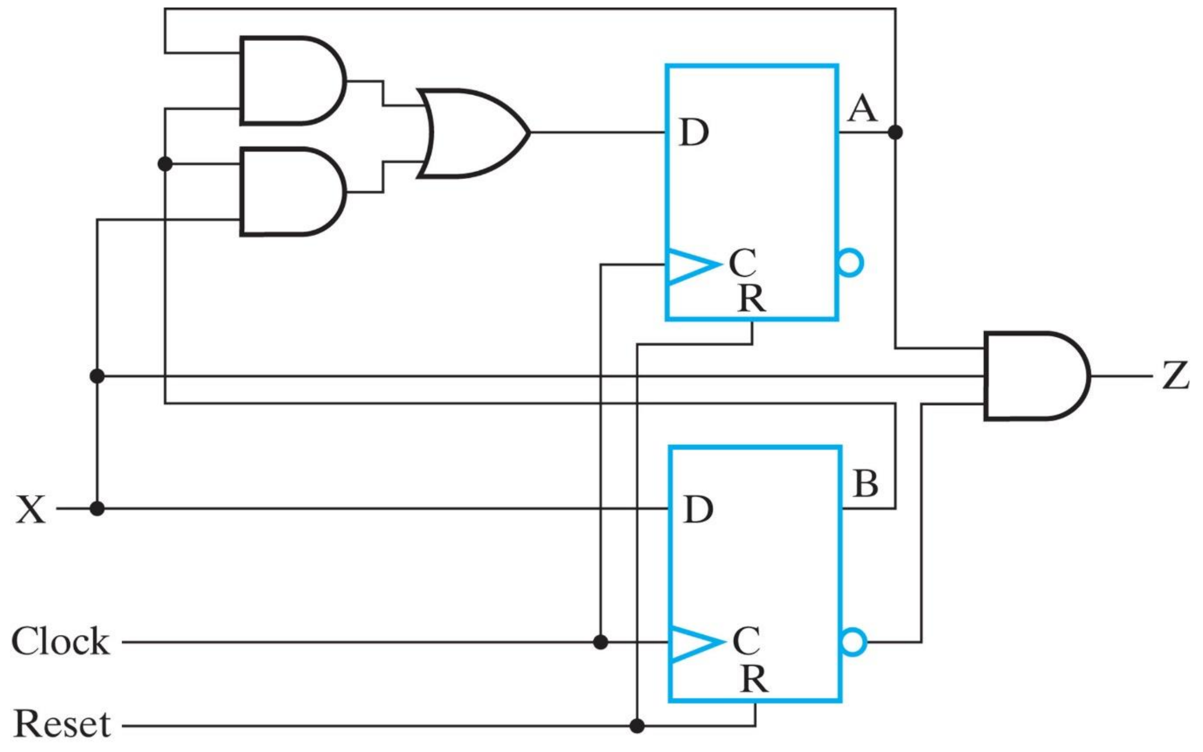
1. Reset
2. State S_0, input 0
3. State S_0, input 1
4. State S_1, input 1
5. State S_2, input 1
6. State S_2, input 0
7. State S_3, input 1
8. State S_1, input 0
9. State S_0, input 1
10. State S_1, input 1
11. State S_2, input 0
12. State S_3, input 0

We draw the shortest path through the state diagram, passing through each state/input combination at least once

NB: To verify all the state/input combinations (8 in this example), we typically need a larger number of transitions (we have to pass through some states multiple times)

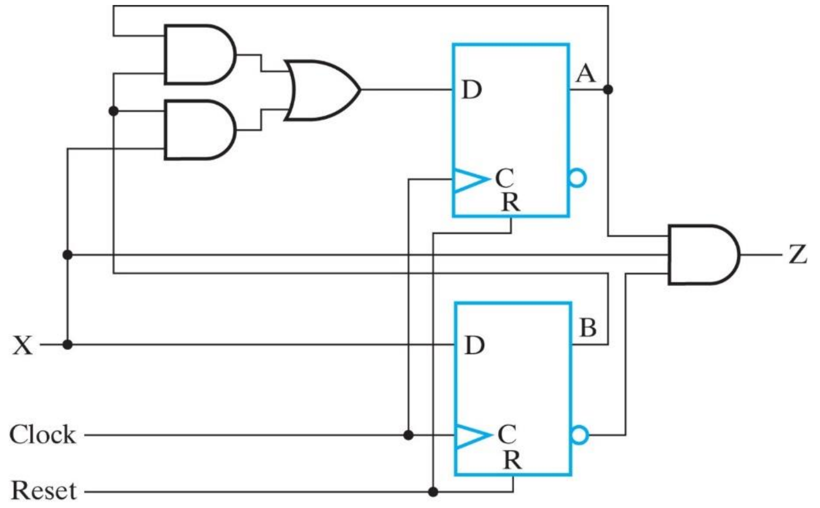
Example 4.3: Sequence Recognizer (Gray Coded States)

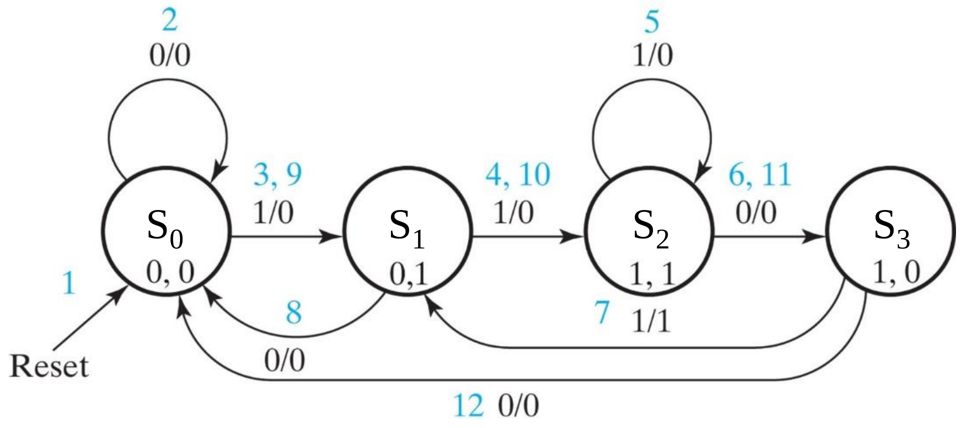
State	Bit A	Bit B
S_0	0	0
S_1	0	1
S_2	1	1
S_3	1	0

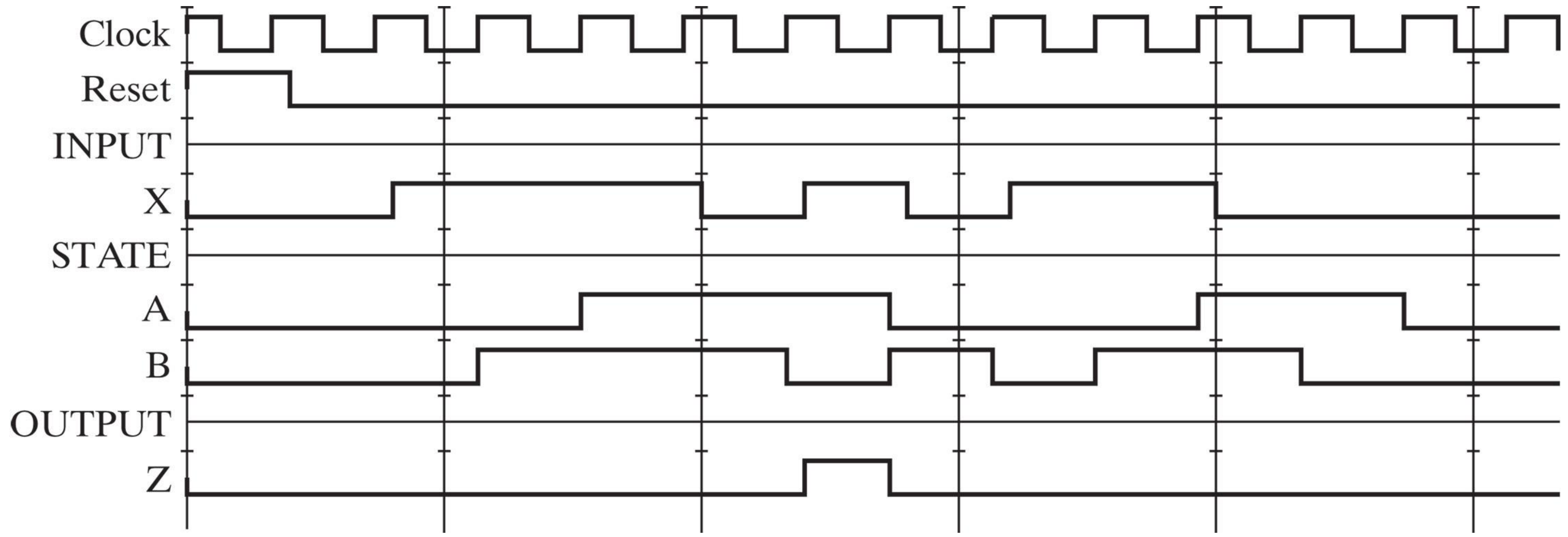


- To verify each transition, we apply the input value at the present state and observe the next state and output
  - In D flip-flops, the next state is equal to the D input right before the clock edge
  - Ex: Present state S_0, input 1 => Next state S_1, output 0

Example 4.3: Sequence Recognizer (Gray Coded States)







Summary

- A synchronous sequential system consists of flip-flops that store the system state and a combinational part to calculate the next state and the output
- The steps for the design of a synchronous sequential circuit are
  - Translate the specifications to a state diagram describing the operation of the system
  - Determine the state table: for each combination of present state and input, the table provides the next state and output
  - Assign a code to the states
  - Find the encoded state table
  - Design the combinational circuit to determine the next state
  - Design the combinational circuit to determine the output
  - Draw the circuit
  - Verify the sequential circuit operation

Disclaimer