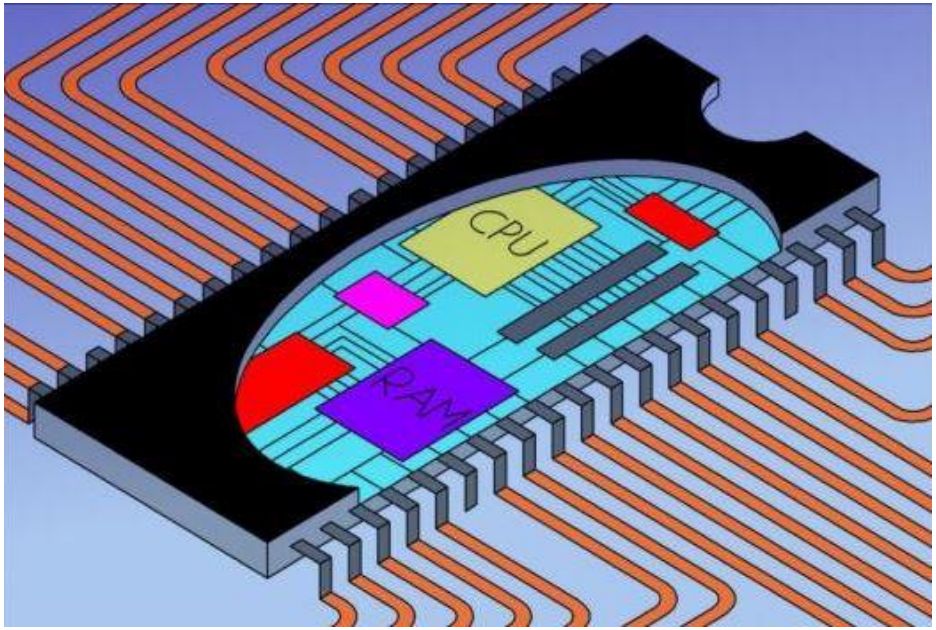
The University of Zambia

EEE 3132 – DIGITAL ELECTRONICS



Jasper HATILIMA

Department of Electrical and Electronic Engineering, School of Engineering

jasper.hatilima@unza.zm
jasperhatilima@yahoo.com

April 2014

Course Outline

Jerry Muwamba

Number Systems and Codes

Logic Fundamentals

Jasper Hatilima

Logic Families and Combinational Logic Circuits:

- Logic Families, TTL, Binary adders and subtractors, Encoders and Decoders, Multiplexers and Demultiplexers.

Sequential Logic Circuits:

- S-R latch, Clock S-R Flip Flop, Level and Edge Triggering, J-K, D-Flip Flops, State Tables, State Diagrams.
- Serial/Parallel In/Out shift registers.
- Timing diagrams, Asynchronous (unclocked) and synchronous systems, Counters: ripple, synchronous, ring counters.

Introduction to Microprocessors and PLDs:

- Three State Registers.
- Memories: ROM, PROM, EPROM, EEPROM, SRAM, DRAM.
- PLAs, PALs, FPGAs.
- Microprocessor Architecture, Instruction Set, Assembly language, Simple-As-Possible Computer.

Finite State Machines

At the end of this component, the student should:

- understand the use of state tables and state diagrams,
- be able to construct state diagrams and state tables and be able to design simple sequential circuits from these tables,
- differentiate between Moore and Mealy state machines and design simple vending machine.

STATE MACHINES

Lets hear some definitions from the class...

SEQUENTIAL CIRCUITS AND STATE MACHINES

✓ As stated earlier, sequential circuits have outputs that depend on the input sequence.

✓ The effect of the input sequence can be memorized as a state of the system.

✓ Sequential circuits are therefore called **STATE MACHINES**.

✓ Memory elements are used to store the state (Usually D flip-flops).

✓ With the flip-flops used to store the state, we can therefore represent states using binary combination of variables.

✓ **n**-bits can represent up to 2^n states.

Consider a system using two flip-flops (X and Y) to store its state.

This means that the system has $2^2 = 4$ states, S_0, S_1, S_2, S_3.

The current state can have the following combinations:

	X	Y
S_0	0	0
S_1	0	1
S_2	1	0
S_4	1	1

A good example of a sequential circuit is a **counter** (as already covered).

Another example is a **sequence detector**.

But in this component, we are going to emphasize the general description (not just examples) of sequential circuits.

This will be done in terms of truth tables (**state tables**) that govern sequential circuits as well as their process diagrams (**state diagrams**).

STATE TRANSITION TABLE

A truth table for a sequential system should be able to give the relationship between the current state and the next state.

This is called the **STATE TRANSITION TABLE** or **STATE TABLE.**

An example of a state transition table is shown below:

Current State		Next State	
X_0	Y_0	X_n	Y_n
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

STATE TRANSITION DIAGRAMS

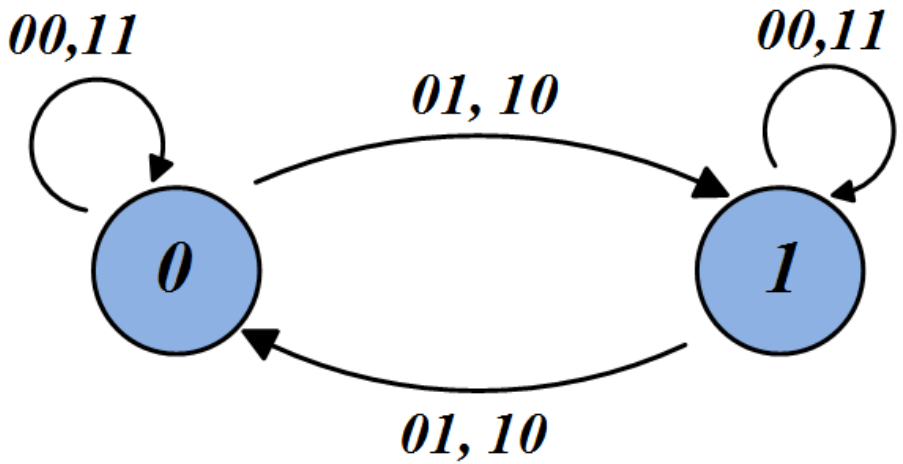
A State Transition Diagram (**also called state diagram**) shows:

- ✓ the state of the circuit,
- ✓ the possible transition between the states,
- ✓ The values of the outputs during transitions.

For a state diagram,

- ➢ A **circle** shows the state,
- ➢ An **arrow** shows the input and the transition from one state to the other,
- ➢ The output is either specified in the state circle or on the transition arrow depending on kind of model (Moore/Mealy)

Example of state transition diagram transitioning between states “1” and state “0” by using two inputs:



KEY TERMS:

- ✓ **State**: Flip-flop output combination.
- ✓ **Present State**: Output of flip-flop before clock pulse.
- ✓ **Next State**: Output of flip-flop after clock pulse.

At the trigger of the clock, the **NEXT STATE** is transferred to the **PRESENT STATE**.

Below is a general sequential circuit showing concepts of state table and state diagram.

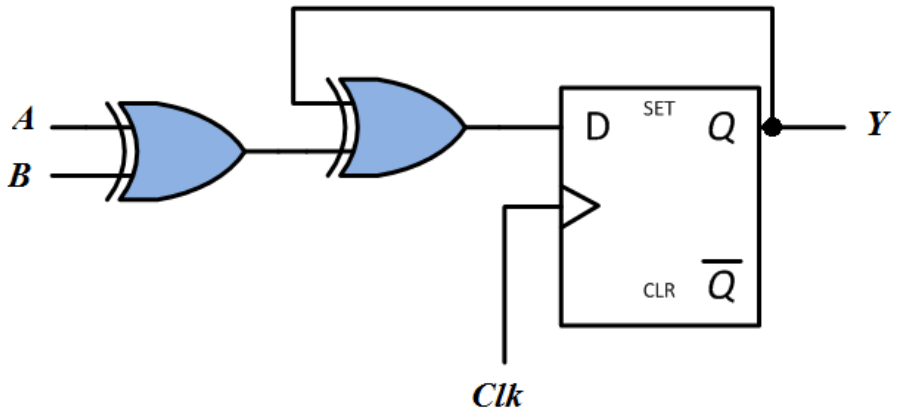
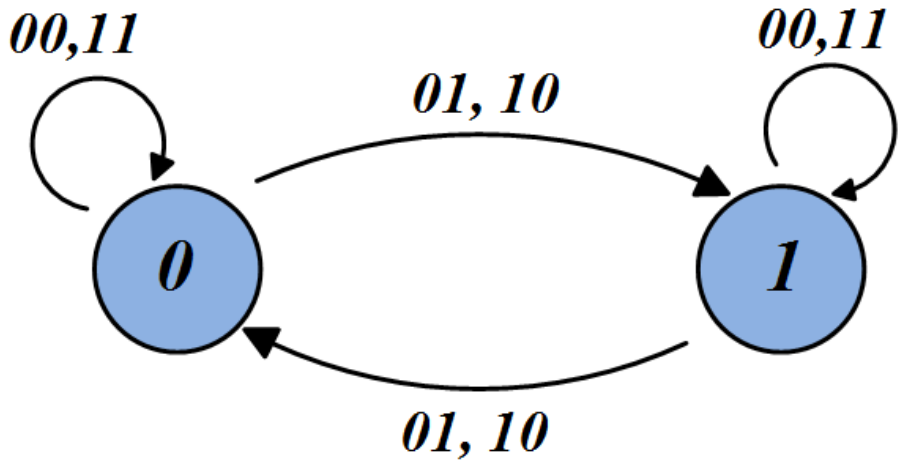


Fig. 2.31Generic Sequential Circuit

HOW MANY STATES IS THIS GOING TO GIVE US?

Present State	Inputs		Next State
$Y_{Present}$	A	B	Y_{Next}
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



NOTE ON STATE DIAGRAMS

It is important to note that 0,1 or 00,11 has a totally different meaning from 0/1 or 00/11.

The comma notation (0,1 or 00,11) is mostly used to indicate inputs that produce the same effect (Refer to diagram above).

The forward slash notation (0/1 or 00/11) is usually used to indicate Input/Output or State/Output relations (Will be clear in a few slides)

Sequential Circuit Models: Categories

Using state tables and state diagrams, finite state machines (FSM) fall under two main models:

1. **Moore Machine**: Output is only function of state.
2. **Mealy Machine**: Output is function of state and inputs.

The specific implementation depends on the logic relations specified in the state table. For instance, a **vending machine**'s output will depend on both state and current input – Mealy. An ant's brain is said to be a Moore Machine ☺

The details of the above two architectures are shown below:

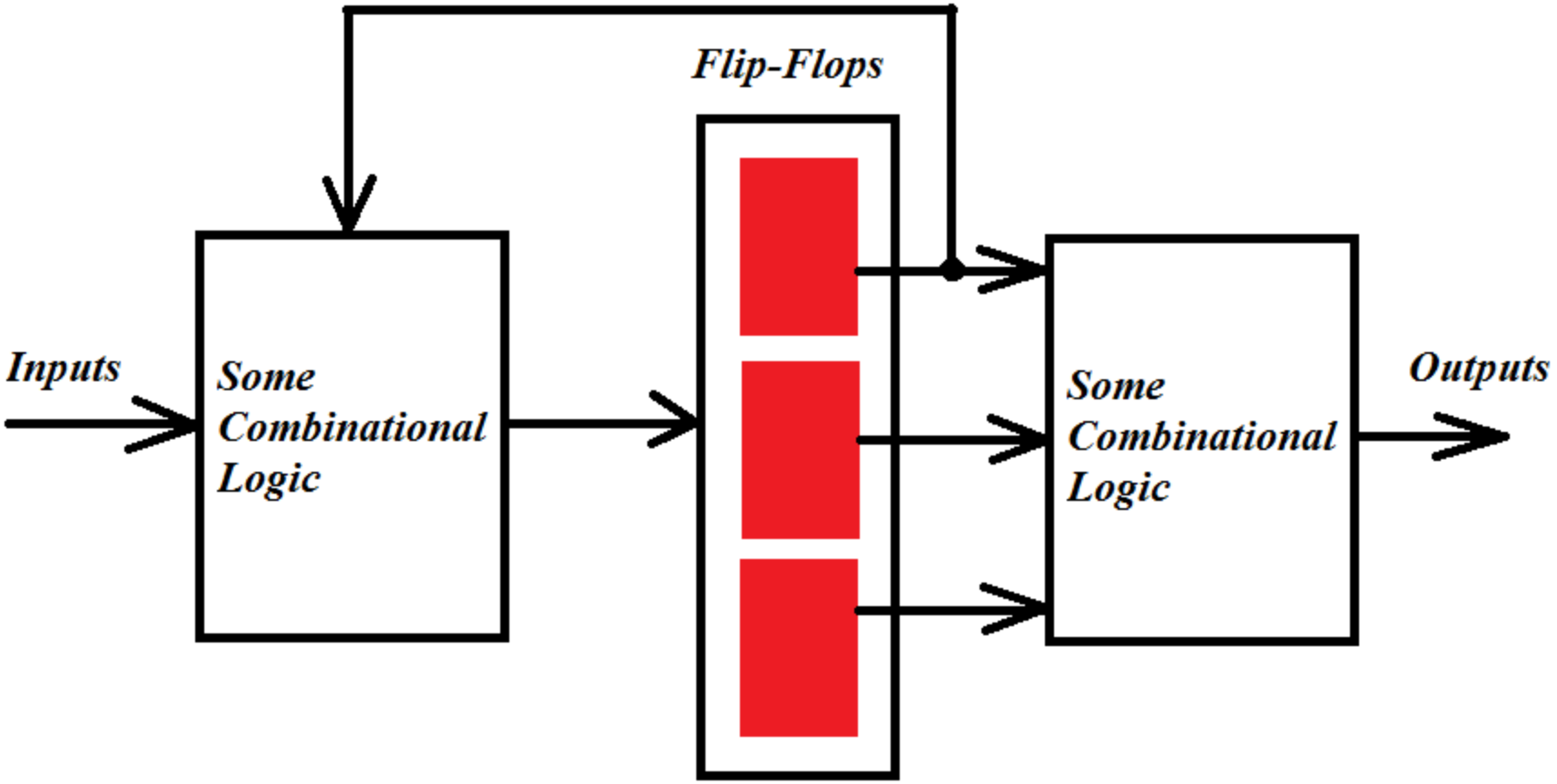


Fig. 2.31 Moore Model of Sequential Circuits

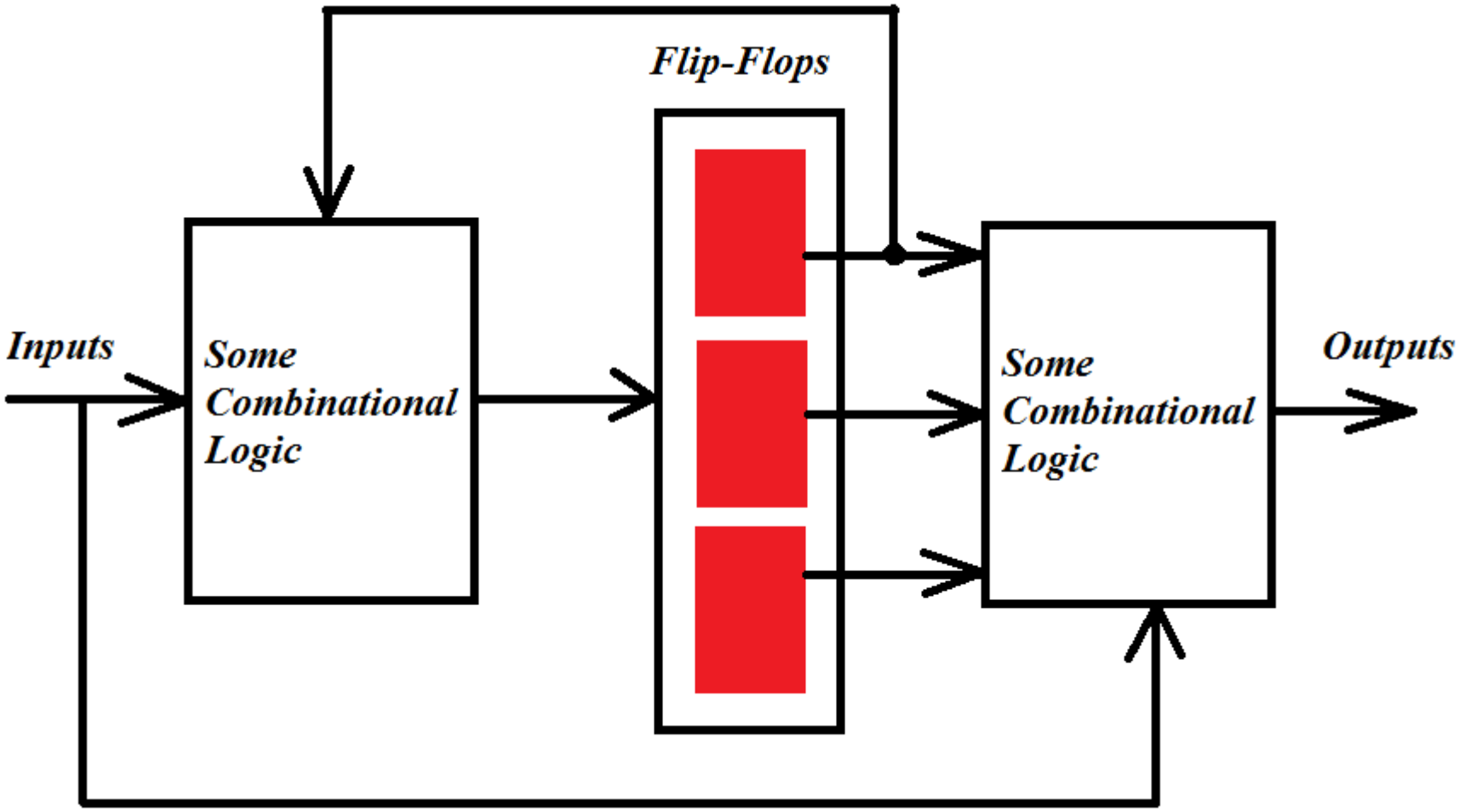


Fig. 2.32 Mealy Model of Sequential Circuits

Moore Model Vs Mealy Model : State Diagrams

In the Moore Model,

- ✓ States are represented by circles and both the state and the output are written within the circles in the form STATE/OUTPUT e.g. 01/1.

- ✓ The arrows between the states only have the inputs written over them.

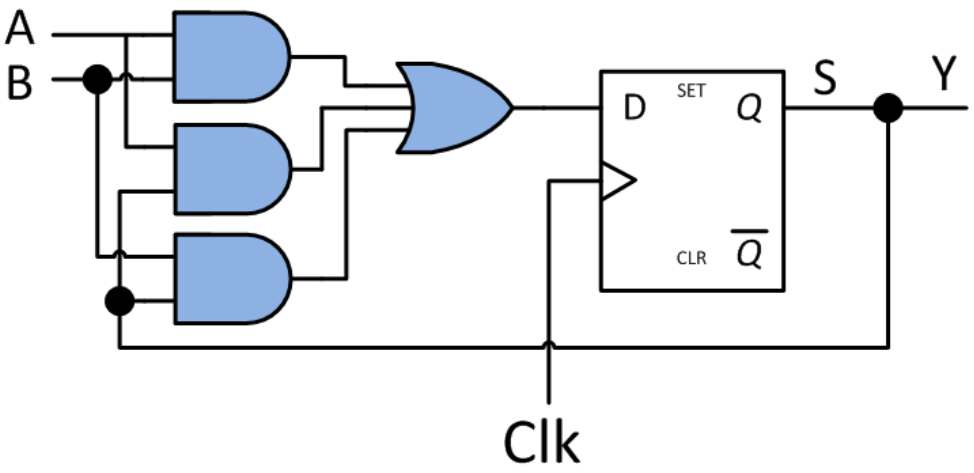
In the Mealy Model,

- ✓ States are represented by circles and only state is written inside the circle.

- ✓ The arrows between the states have the input and the output written over them.

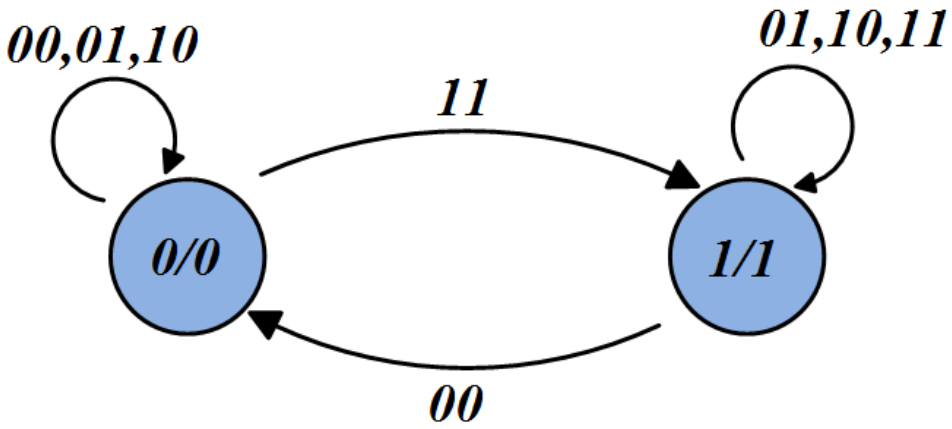
Examples below...

Example of Moore Model



Present State	Inputs		Next State
$S_{Present}$	A	B	S_{Next}
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Fig. 2.33 Example of Moore Model



INPUT EQUATIONS for Moore Model in Figure 2.33

$$S_{NXT} = AB + YS_{PRSNT} + XS_{PRSNT}$$

Next state in terms of input and present state

$$Y = S_{PRSNT}$$

Output in terms of present state

Example of Mealy Model

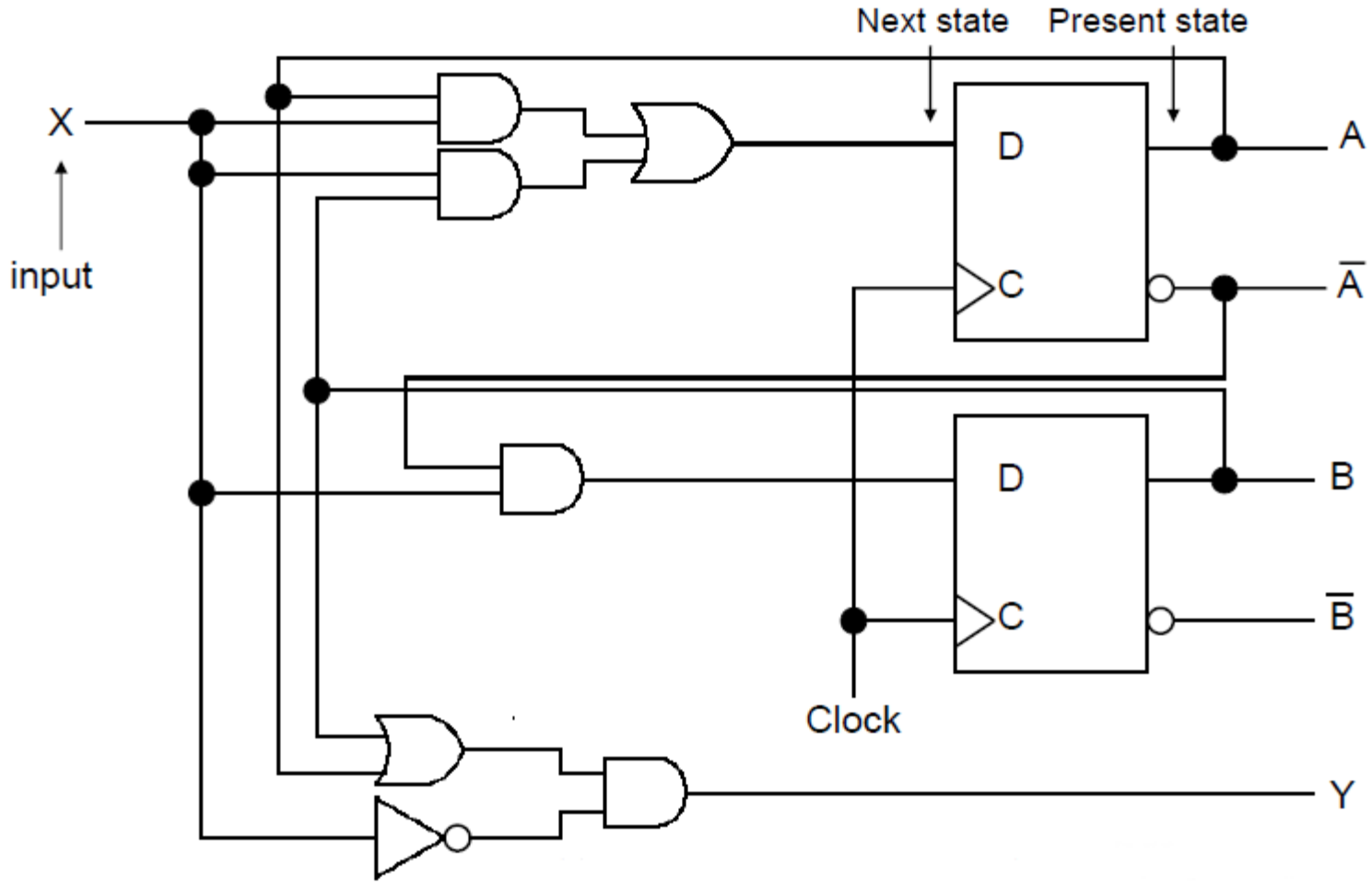


Fig. 2.34 Example of Mealy Model

Input Equations – Mealy

$$A_{next} = A_{present}X + B_{present}X$$

$$B_{next} = A'_{present}X$$

Next state in terms of input and present state

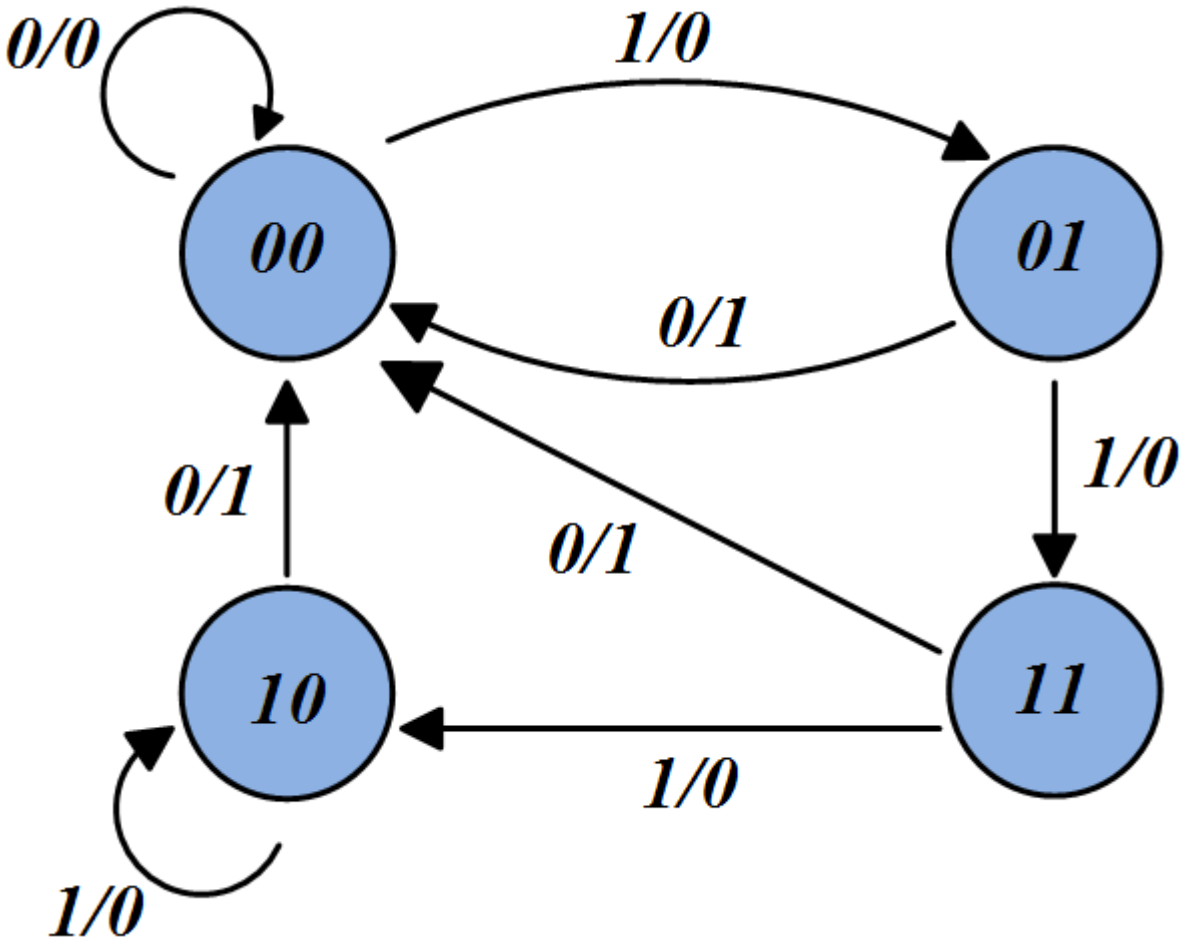
$$Y = (A_{present} + B_{present})X'$$

Output in terms of input and present state

State Table – Mealy

Present State		Input	Next State		Output
A	B	X	A	B	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

State Diagram – Mealy



Moore Vs Mealy : Comparison

1. Output for Moore Machines tends to change at clock edge, but in Mealy machines, output changes as soon as input changes.

✓ This makes interconnection of Mealy machines a big problem.

✓ On the other hand, the above makes Mealy machines faster and preferable in some applications.

What is the general design procedure for Sequential Circuits?

Design Procedure for Sequential Circuits

1. Understand performance specifications,
2. Assign **state number** for each state,
3. Assign One **D flip-flop** for each state bit,
4. Draw a **STATE DIAGRAM**,
5. Draw **STATE TABLE**,
6. Derive **INPUT EQUATIONS**.

We have used sequential circuits in Registers and Counters.

And now with our deeper understanding of FSM and design procedure, can we think of one other common application of Sequential Circuits / Stage Machines?

SEQUENCE DETECTORS

State machines by nature are suited to **track states** and detect specific sequence of events.

We may for example design an algorithmic state machine to track a certain pattern in the input sequence.

For example:

✓ To count the **number of** 1's in a sequence and produce a HIGH output when three 1's are counted.

✓ To produce a HIGH output when a **specific PATTERN** in the sequence is detected (e.g. 011, 0101 etc)

EXAMPLE…

EXAMPLE

Design a sequential circuit that will recognize the input sequence 1101. The output should go HIGH when bits are detected in that order.

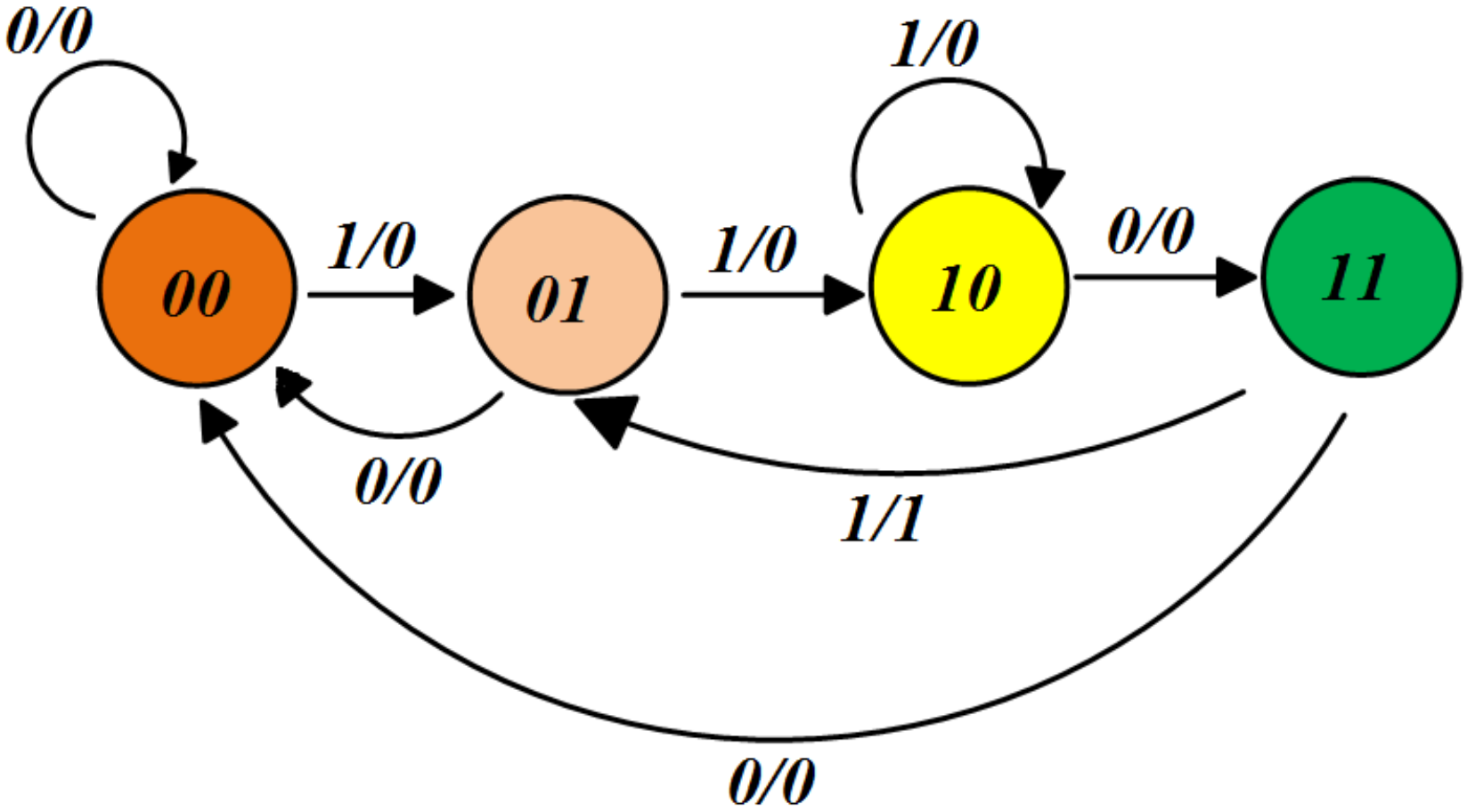
IS IT MOORE OR MEALY SYSTEM?

SOLUTION

Following the design procedure:

1. It is a sequence detector
2. Four states are required => $\log_2 4 = 2$ bits:

   - ✓ Waiting for first detection – **State A (00)**
   - ✓ For detecting the first 1 – **State B (01)**
   - ✓ For detecting the second 1 – **State C (10)**
   - ✓ For detecting the 0 – **State D (11)**
   - ✓ Detecting the last one closes the sequence detection and outputs a HIGH but at the same time may mark the beginning of another sequence and hence goes to **State B.**


0/0
1/0
1/0
1/0
0/0
00
01
10
11
0/0
1/1
0/0

Another Example: MAHEU VENDING MACHINE MEALY or MOORE ?

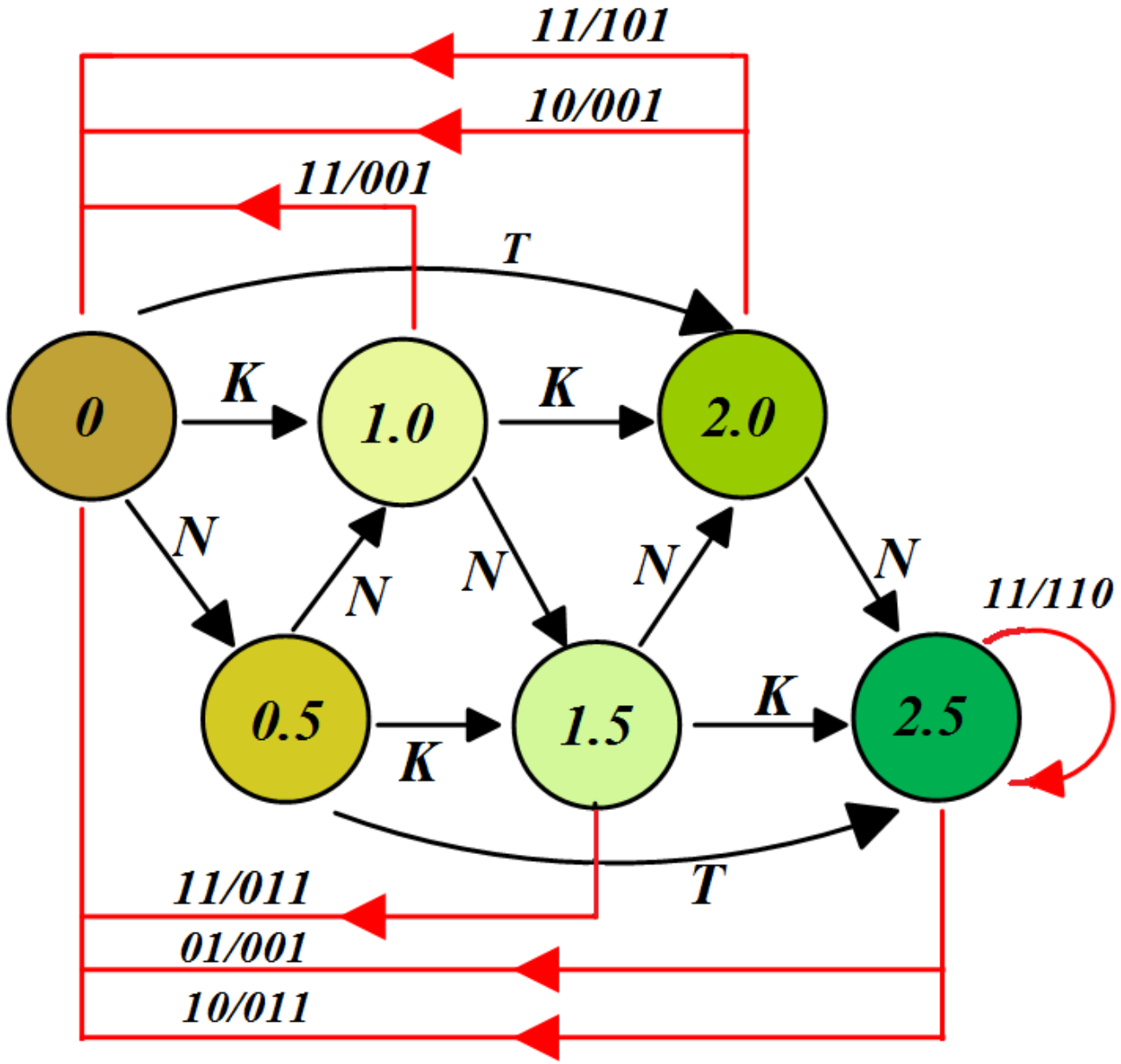
1. Task: Collect money, deliver product and change.
2. The machine may get two kind of inputs, N and K
   - ✓ 50 Ngwee coin = N,
   - ✓ 1 Kwacha coin = K,
   - ✓ 2 Kwacha note = T,
   - ✓ One input at a time,
   - ✓ Maheu cost is ZMW 3
   - ✓ Does not accept more than ZMW 4: Drops the extra coin to Change Collection Tray.
3. How many states?
4. What are the output signals?

With the smallest denomination being 5 Ngwee and considering the Zero Kwacha state as the starting state, it will take us 5 coins to go to ZMW 2.5 before we get our Maheu and reset to Zero Kwacha state.

Therefore a total of 6 states: 3-bits are sufficient (3 Flip-flops)
000 – No Coin
001 – 0.5 Kwacha
010 – 1.0 Kwacha
011 – 1.5 Kwacha
100 – 2.0 Kwacha
101 – 2.5 Kwacha

Our inputs are:
0 Kwacha, Z = 00
50 Ngwee, N = 01
1 Kwacha, K = 10
2 Kwacha, T = 11

Our outputs are:
C_1C_2P
Where C_1C_2 is the Change given back and P is the product.
e.g. 101 gives ZMK 1 change and the Maheu.


11/101
10/001
11/001
T
0
K
1.0
K
2.0
N
N
N
N
N
11/110
0.5
K
1.5
K
2.5
T
11/011
01/001
10/011

In the state diagram above, you can then make the substitutions for the states as:

0 = 000
0.5 = 001
1.0 = 010
1.5 = 011
2.0 = 100
2.5 = 101

And for the inputs as:

Z = 00, N =01, K = 10, T = 11 with their corresponding outputs C_1C_2P

You can then derive the input equations and get the digital implementation using three flip-flops.