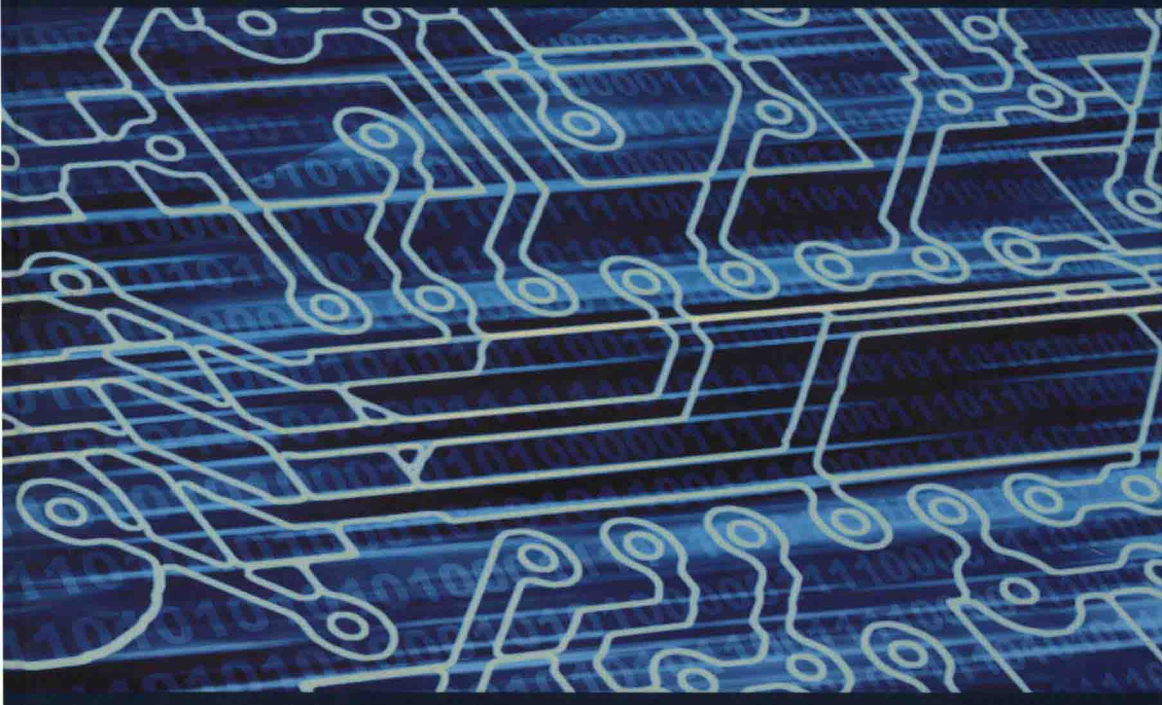
ELECTRONICS ENGINEERING SERIES

Digital Electronics 3

Finite-state Machines

Tertulien Ndjountche



ISTE WILEY

Series Editor
Robert Baptist

Digital Electronics 3

Finite-state Machines

Tertulien Ndjountche

ISTE

WILEY

First published 2016 in Great Britain and the United States by ISTE Ltd and John Wiley & Sons, Inc.

ISTE Ltd
27-37 St George's Road
London SW19 4EU
UK

www.iste.co.uk

John Wiley & Sons, Inc.
111 River Street
Hoboken, NJ 07030
USA

www.wiley.com

Library of Congress Control Number: 2016950312

British Library Cataloguing-in-Publication Data
A CIP record for this book is available from the British Library
ISBN 978-1-84821-986-1

Digital Electronics 3

Preface

The omnipresence of electronic devices in everyday life is accompanied by the size reduction and the ever-increasing complexity of digital circuits. This comprehensive and easy-to-understand book deals with the basic principles of digital electronics and allows the reader to grasp the subtleties of digital circuits, from logic gates to finite state machines. It presents all the aspects related to combinational logic and sequential logic. It introduces techniques to establish logic equations in a simple and concise manner, as well as methods for the analysis and design of digital circuits. Emphasis has been especially laid on design approaches that can be used to ensure a reliable operation of finite state machines. Various programmable logic circuit structures by practical examples and well-designed exercises with worked solutions.

This series of books discusses all the different aspects of digital electronics, following a descriptive approach combined with a gradual, detailed and comprehensive presentation of basic concepts. The principles of combinational and sequential logic are presented, as well as the underlying techniques for the analysis and design of digital circuits. The analysis and design of digital circuits with increasing complexity is facilitated by the use of abstractions at the circuit and architecture levels. The series is divided into three volumes devoted to the following subjects:

1) combinational logic circuits;

2) sequential and arithmetic logic circuits;

3) finite state machines.

A progressive approach has been chosen and the chapters are relatively independent of each other. To help master the subject matter and put the different concepts and techniques into practice, the book is complemented by a selection of exercises with solutions.

Summary

This volume deals with finite state machines. These machines are characterized by a behavior that is determined by a limited and defined number of states, and the holding conditions for each state and the branching conditions from one state to another. They only allow one transition at a time and can be divided into two components: a combinational logic circuit and a sequential logic circuit. This third volume contains the following three chapters.

1) Synchronous Finite State Machines;

2) Algorithmic State Machines;

3) Asynchronous Finite State Machines.

The reader

This book is an indispensable tool for all engineering students in a bachelors or masters course who wish to acquire detailed and practical knowledge of digital electronics. It is detailed enough to serve as a reference for electronic, automation engineers and computer engineers.

Tertulien NDJOUNTCHE
August 2016

Contents

1

Synchronous Finite State Machines

1.1. Introduction

Digital circuits composed of combinational and sequential logic sections are generally described as finite state machines.

A machine is synchronous when the state transitions are controlled or synchronized by a clock signal.

A machine whose operation is not dependent on a clock signal is said to be asynchronous.

The present state (PS) of a state machine is determined by the variables stored in the flip-flops of the sequential section. The next state (NS) of the state machine is defined by the circuit of the combinational logic section.

Among finite state machines, we can differentiate between the *Moore* model and the *Mealy* model:

– Moore state machine: the state machine output depends entirely on the PS;

– Mealy state machine: the state machine output depends on the inputs and PS.

It must be noted that there are also hybrid machines with some outputs being of Moore type and others of Mealy type.

A machine always has a finite number of states. For N variables, the machine must have between 2 and 2^N states.

A machine is defined by specifying the number of inputs and outputs, the initial state, the PS and the NS.

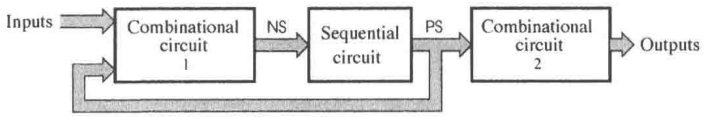


Figure 1.1. *Finite state machine: Moore model (NS: next state; PS: present state)*

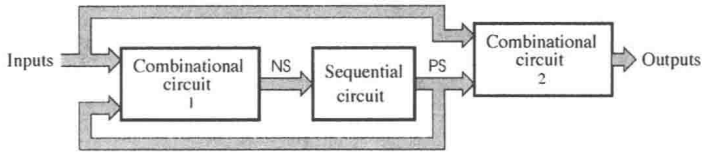


Figure 1.2. *Finite state machine: Mealy model (NS: next state; PS: present state)*

1.2. State diagram

Consider the state diagram for the Moore state machine shown in Figure 1.3. Starting from the initial state S_0, the machine goes to the state S_1 regardless of the logic state of the input X. Assuming that the PS corresponds to S_2 and that the output is set to 1, the NS will be either S_1, with the output remaining at 1 if the logic level of the input X becomes 0, or S_3, with the output being set to 0 if the input X takes the logic level 1.

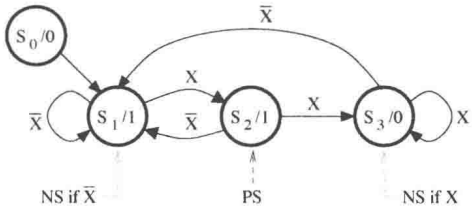


Figure 1.3. *Moore state machine: state diagram with present state and next state*

Figure 1.4(a) shows a section of the state diagram for a Mealy state machine. The states whose binary codes are 000, 010, 001 and 011 are denoted by A, B, C and D, respectively, and the outputs are S_1 and S_2.

We assume that B is the PS. The holding condition in state B is XY and the outputs S_1 and S_2 take the logic state 1. The input condition $\overline{X}$ causes the machine

to enter the state D and the output, S_2, is set to 0. Once in this state, the $\overline{X}$ condition allows the machine to remain in this state. When the logic condition $X\ \overline{Y}$ is true, the machine goes to the state C, where there is no holding condition and the output S_1 is set to 0.

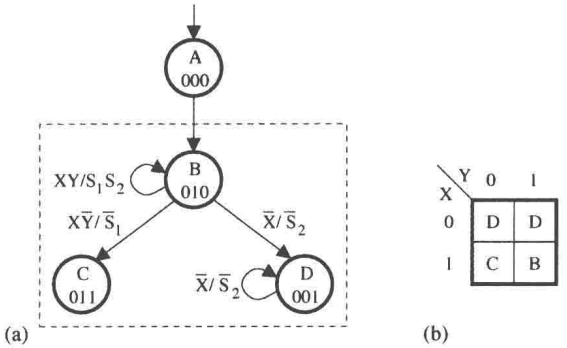


Figure 1.4. *Mealy state machine: a) state diagram; b) map showing input/next state from state B*

Figure 1.4(b) shows what state the machine may move to once in the state B based on the logic levels of inputs X and Y.

A state diagram is constructed according to certain rules. For a section of the state diagram, such as the one illustrated in Figure 1.5, where the conditions that cause the machine to remain in state S_j and to move from S_j to S_k $(k = 1, 2, \cdots, n-1)$ are represented by F_0 and F_k, respectively, the following logic equations must be verified:

– *Sum rule*: the Boolean sum of all conditions under which a transition from a given state occurs must be equal to 1:

$$F_0 + F_1 + \cdots + F_{n-1} = 1 \qquad [1.1]$$

– *Mutual-exclusion requirement*: each condition under which a transition from a given state occurs cannot be associated with more than one transition path:

$$F_0 = \overline{F_1 + F_2 + \cdots + F_{n-1}} \qquad [1.2]$$

$$F_1 = \overline{F_0 + F_2 + \cdots + F_{n-1}} \qquad [1.3]$$

$$\vdots$$

$$F_{n-1} = \overline{F_0 + F_1 + \cdots + F_{n-2}} \qquad [1.4]$$

As a result, the Boolean product of both state transition conditions, $F_l \cdot F_k$ ($l, k = 0, 1, 2, \cdots, n-1$ and $l \neq k$), is equal to 0.

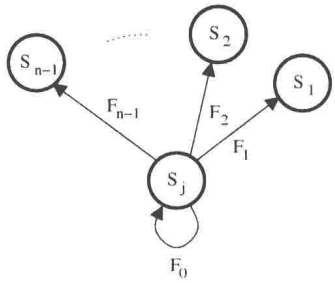


Figure 1.5. *Section of a state diagram*

However, these relationships need not be verified for applications where certain conditions will never happen or are not allowed (don't-care conditions).

EXAMPLE 1.1.– Let us consider the section of the state diagram illustrated in Figure 1.6(a). Using the Boolean transformation, we have:

$$\overline{X} + X \cdot Y + X \cdot \overline{Y} = \overline{X} + X(Y + \overline{Y}) = \overline{X} + X = 1 \qquad [1.5]$$

and

$$\overline{X} = \overline{X \cdot Y + X \cdot \overline{Y}} = (\overline{X} + \overline{Y})(\overline{X} + Y) = \overline{X}(1 + Y + \overline{Y}) = \overline{X} \qquad [1.6]$$

$$X \cdot Y = \overline{\overline{X} + X \cdot \overline{Y}} = X(\overline{X} + Y) = X \cdot Y \qquad [1.7]$$

$$X \cdot \overline{Y} = \overline{\overline{X} + X \cdot Y} = X(\overline{X} + \overline{Y}) = X \cdot \overline{Y} \qquad [1.8]$$

Thus, the sum rule and the mutual-exclusion requirement are both satisfied. Figure 1.6(b) depicts the map showing the input/NS from state A.

EXAMPLE 1.2.– Analyzing the state diagram shown in Figure 1.6(c), we can see that the sum rule is verified while the mutual-exclusion requirement is not fulfilled because the product of the terms X and $X \cdot Y$ is not equal to 0. Figure 1.6(d) shows the map for the input/NS from state A. For the branching condition $XY = 11$, the NS can be either B or C, while only one transition at a time can be carried out from a given state. Thus, when the mutual-exclusion requirement is satisfied for a given state, no cell in the input/NS map should contain more than one state symbol.

EXAMPLE 1.3.– A section of the state diagram of a finite state machine is depicted in Figure 1.6(e). We can verify that the sum rule is satisfied, but not the mutual-exclusion

requirement. This is because the product of the terms $\overline{X}$ and $\overline{Y}$ is not equal to 0. As shown in Figure 1.6(f), that illustrates the input/NS starting from the state A, the $\overline{X} \cdot \overline{Y}$ condition causes the state machine either to remain in state A or to advance to state C. However, this ambiguity can be ignored if it is assumed that the condition $\overline{X} \cdot \overline{Y}$ will never occur.

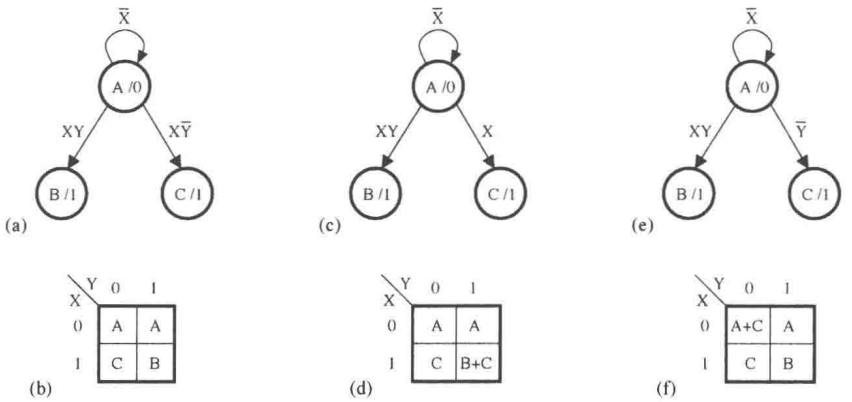


Figure 1.6. *Examples of state diagram sections and maps showing the input/next state from state A*

In a state diagram, we can differentiate between conditional and unconditional transitions:

– conditional transitions are only carried out on the edge of a clock signal when a certain condition, relating to the inputs, is verified. There are always at least two conditional transitions from the same state;

– unconditional transitions are automatically carried out on the occurrence of a clock signal edge. Only one unconditional transition is possible from a given state.

Let us consider an example: the state diagram for an incompletely specified Moore state machine shown in Figure 1.7. There are two inputs and the output can take either 0 or 1 or a don't-care state, represented by (–). The only unconditional transition exists between the states S_3 and S_0.

The operation of this machine can also be described based on the state table shown in Table 1.1. Starting from the state S_3, where the output is in the don't-care state, the machine goes to S_0 regardless of the logic combination of the inputs.

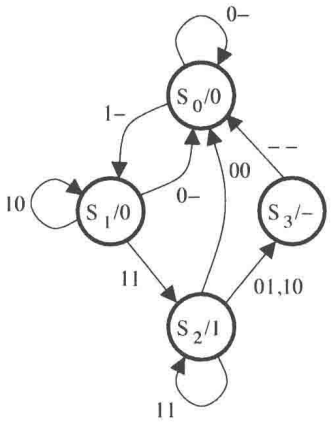


Figure 1.7. *State diagram for an incompletely specified state machine based on Moore model*

PS	NS				Output
	$XY = 00$	01	10	11	
S_0	S_0	S_0	S_1	S_1	0
S_1	S_0	S_0	S_1	S_2	0
S_2	S_0	S_3	S_3	S_2	1
S_3	S_0	S_0	S_0	S_0	–

Table 1.1. *State table of an incompletely specified state machine based on Moore model*

1.3. Design of synchronous finite state machines

The procedure for designing synchronous finite state machines may include the following steps:

1) derive the state diagram;

2) draw up the state table;

3) assign bit combinations to the variables in order to represent the different states (encoding the different states) and draw up the corresponding state table;

4) choose the flip-flop type;

5) derive the input equations based on Karnaugh maps;

6) represent the resulting logic circuit.