

Fundamentals of Digital Circuit Design

Cezary Zieliński

Publisher: Wydawnictwo Naukowe PWN, Warsaw, 2003
(Published in Polish; contains 450 pages)

Objectives of the book:

The course is targeted at all computer science, electrical, electronics, telecommunications and mechanical engineers who require the understanding of the functioning and the design of logic circuits at different levels of their complexity. The motif for this lecture is to show what are the major components and how to design digital logic for real time systems (e.g. mechatronics, robotics, computer systems). The lecture starts with combinational logic and ends with the design of large digital systems. The circuits integrated to a higher extent or more complex systems are designed using the already introduced components. For instance, a D-type flip-flop is designed using gates, then in turn functional blocks, such as counters or registers, are designed using those flip-flops, and finally functional blocks are used to construct data processing and control systems. In this way no phase of integration is left out giving the reader a full understanding of the field. As a result the readers should be able to design circuits and systems using elements of any scale of integration according to the needs of a project. Moreover, different approaches to designing the same circuits or systems are explained. All the concepts are explained explicitly showing the design process by simple yet real engineering examples. The mathematical background is kept to the minimum necessary level (rudimentary Boolean algebra is utilised). The examples are chosen in such a way that the inner workings of computers are explained, starting at the gate level, traversing ever more complex modules, and ending with microprogrammed machines.

Table of contents:

Foreword

Notation

1. Introduction

- 1.1. Digital circuits
- 1.2. A short esej on designing

2. Combinational circuits

- 2.1. Basics of combinational circuit design
 - 2.1.1. Natural language description
 - 2.1.2. Formalization of the description
 - 2.1.3. Basic gates
 - 2.1.4. Creation of the circuit diagram
 - 2.1.5. Algebraic minimisation of logical expressions
 - 2.1.6. Cost evaluation
 - 2.1.7. Reduction of the number of gate types used
 - 2.1.8. NAND and NOR gates
 - 2.1.9. Functionally complete systems

- 2.1.10. Specification of logical functions
- 2.1.11. Fundamentals of mnemonic methods of minimising logical functions
- 2.1.12. Grays code
- 2.1.13. Karnaugh maps
- 2.1.14. Minimisation of logical functions using Karnaugh maps
- 2.1.15. Dont care values
- 2.1.16. Sum of Products normal form
- 2.1.17. Factorization
- 2.1.18. Prohibition
- 2.1.19. Factorization and prohibition combined
- 2.1.20. Product of Sums normal form
- 2.1.21. Quine and McCluskeys method of minimisation
- 2.1.22. Static hazard
- 2.1.23. Dynamic hazard
- 2.1.24. XOR gates
- 2.2. Complex combinational circuits
 - 2.2.1. Iterative circuits with one-way flow of information between blocks
 - 2.2.2. Iterative circuits with two-way flow of information between blocks
 - 2.2.3. Cascade circuits
 - 2.2.4. Combinational circuits utilising multiplexers
 - 2.2.5. Combinational circuits utilising decoders
 - 2.2.6. Demultiplexers
 - 2.2.7. Read only memory
 - 2.2.8. Application Specific Integrated Circuits

Design exercises

3. Synchronous circuits

- 3.1. General introduction
 - 3.1.1. Problem formulation
 - 3.1.2. Automaton state
 - 3.1.3. Graphs
 - 3.1.4. State transition and output table
 - 3.1.5. Coding
 - 3.1.6. D type flip-flop
 - 3.1.7. Flip-flop excitation tables
 - 3.1.8. Synchronous automaton realisation
- 3.2. Theoretical foundations
 - 3.2.1. Synchronous automaton structures
 - 3.2.2. Automaton
 - 3.2.3. Equivalence of complete (fully specified) automaton states
 - 3.2.4. Compatibility of incomplete (partially specified) automaton states
 - 3.2.5. Formalisation of the minimisation algorithms
 - 3.2.6. Realisation of a minimal automaton using D type flip-flops

- 3.2.7. JK flip-flop
- 3.2.8. Realisation of a synchronous automaton using JK type flip-flops
- 3.2.9. Partial equivalence of Moore and Mealy automata
- 3.2.10. Automata without inputs

Design exercises

4. Asynchronous circuits

- 4.1. General introduction
 - 4.1.1. Problem formulation
 - 4.1.2. State graph and state transition and output table
 - 4.1.3. Coded state transition and output table
 - 4.1.4. Circuit realization
- 4.2. Basic definitions
 - 4.2.1. Moore and Mealy automata
 - 4.2.2. Stable and unstable states
 - 4.2.3. Design assumptions
- 4.3. Fundamentals of asynchronous circuit design
 - 4.3.1. Time plots
 - 4.3.2. Assigning of states
 - 4.3.3. Primary state transition and output table
 - 4.3.4. Adding the missing information
 - 4.3.5. Minimisation of the number of states
 - 4.3.6. Incorrect coding
 - 4.3.7. Races
 - 4.3.8. Cyclic transitions
 - 4.3.9. State coding using hypercubes
 - 4.3.10. Coded state transition and output table and the realisation of the automaton
- 4.4. Alternative method of minimising asynchronous automata
 - 4.4.1. Phase one search for pseudoequivalent states
 - 4.4.2. Phase two search for pseudocompatible states
 - 4.4.3. An example of the utilization of the alternative method of minimising asynchronous automata
- 4.5. Selected problems of asynchronous circuit design
 - 4.5.1. Assigning of states to sections of the time plots
 - 4.5.2. Elaboration of state transition and output functions
 - 4.5.3. Static RS type flip-flop
 - 4.5.4. Utilisation of static RS type flip-flops in the design of asynchronous automata
 - 4.5.5. Asynchronous Mealy automata
- 4.6. Asynchronous circuit design using state graphs
 - 4.6.1. Problem formulation
 - 4.6.2. Creation of the state graph
 - 4.6.3. Minimization of the number of states and coding

4.6.4. Realization of an automaton

Design exercises

5. Number representation

5.1. Historical notes

5.2. Positional representation of natural numbers

5.3. Positional representation of positive rational numbers

5.4. Radix transformation

5.5. Addition of numbers

5.6. Multiplication of numbers

5.7. Multiplication by radix

5.8. Division by radix

5.9. Representation of negative numbers

5.9.1. Signed magnitude representation of negative numbers

5.9.2. Fixed point representation

5.9.3. Diminished radix complement systems

5.9.4. Radix complement systems

Exercises

6. Functional blocks

6.1. Internal structure of a functional block

6.1.1. Problem formulation

6.1.2. Design phases

6.1.3. Design of the synchronous part

6.1.4. Design of the asynchronous (static) part

6.1.5. Design of the combinational part

6.1.6. Circuit diagram of the resulting functional block

6.2. Types of functional block inputs

6.2.1. Synchronic operations

6.2.2. Asynchronic static operations

6.2.3. Asynchronic dynamic operations

6.3. General survey of functional blocks

6.3.1. Combinational functional blocks

6.3.2. Sequential functional blocks

Design exercises

7. Control and data processing digital systems

7.1. General introduction and formulation of the problem

7.2. System and its environment

7.3. Algorithm

7.3.1. Communication algorithm of the designed system with the external system

- 7.3.2. Multiplication algorithm
- 7.4. Data path subsystem
- 7.5. An improved multiplication algorithm and the resulting data path subsystem
- 7.6. Structure of a digital system
- 7.7. Signals controlling a data path subsystem
- 7.8. Methods of transforming flow charts into state graphs
- 7.9. Moore control automaton
 - 7.9.1. Transformation of the flow chart into a state graph
 - 7.9.2. Realization of a minimal control automaton
 - 7.9.3. Problems with a Moore automaton
- 7.10. Mealy control automaton
 - 7.10.1. Transformation of the flow chart into a state graph
 - 7.10.2. Realization of a minimal control automaton
 - 7.10.3. Problems with a Mealy automaton
- 7.11. Realization of the control subsystem as a sequencer
 - 7.11.1. Transformation of the flow chart into a sequencer
 - 7.11.2. Realization of a Moore control sequencer
 - 7.11.3. Realization of a Mealy control sequencer
- 7.12. Initialization of the control subsystem
- 7.13. Microprogrammed control subsystems
 - 7.13.1. Microprogrammed Moore control automaton
 - 7.13.2. Microprogrammed Mealy control automaton
 - 7.13.3. Microprogrammed machines
- 7.14. An outline of yet another project
 - 7.14.1. Problem formulation
 - 7.14.2. Data path subsystem and the flow chart
 - 7.14.3. Priority selector
 - 7.14.4. Throughput analysis of the pulse generator

Design exercises

8. Concurrent digital systems

- 8.1. Problem formulation
- 8.2. Petri nets
 - 8.2.1. Petri net definition
 - 8.2.2. Marked Petri net
 - 8.2.3. Functioning of a marked Petri net
 - 8.2.4. A few useful terms
 - 8.2.5. Example of Petri net utilization
 - 8.2.6. Reachability tree
 - 8.2.7. Enhanced Petri net
- 8.3. Concurrent system controller

- 8.3.1. Creation of a Petri net
- 8.3.2. Analysis of a Petri net
- 8.3.3. Control automaton state transition graph
- 8.3.4. Control automaton
- 8.3.5. Company development

Design exercise

9. Quo vadis? (Where from here?)

Appendix A. Boolean algebras

Appendix B. Automata and languages

Exercises

Appendix C. Historical notes

References

Index