

FOURTH EDITION

# DIGITAL LOGIC DESIGN

BRIAN HOLDSWORTH • CLIVE WOODS



Newnes  
An imprint of Elsevier Science  
Linacre House, Jordan Hill, Oxford OX2 8DP  
225 Wildwood Avenue, Woburn, MA 01801-2041

First published 1982  
Second edition 1987  
Third edition 1993  
Fourth edition 2002

Copyright © Brain Holdsworth and Clive Woods, 1982, 1987, 1993, 2002

The right of Brian Holdsworth and Clive Woods to be identified as the authors of this work has been asserted in accordance with the Copyright, Designs and Patents Act 1988

No part of this publication may be reproduced in any material form (including photocopying or storing in any medium by electronic means and whether or not transiently or incidentally to some other use of this publication) without the written permission of the copyright holder except in accordance with the provisions of the Copyright, Designs and Patents Act 1988 or under the terms of a licence issued by the Copyright Licensing Agency Ltd, 90 Tottenham Court Road, London, England W1T 4LP. Applications for the copyright holder's written permission to reproduce any part of this publication should be addressed to the publishers

**British Library Cataloguing in Publication Data**

A catalogue record for this book is available from the British Library

ISBN 0 7506 45882

Typeset by Integra Software Services Pvt. Ltd, Pondicherry, India  
[www.integra-india.com](http://www.integra-india.com)

Printed and bound in Great Britain

---

# Contents

---

|                                                                |           |
|----------------------------------------------------------------|-----------|
| <i>Preface to the fourth edition</i>                           | xii       |
| <i>Acknowledgments</i>                                         | xiii      |
| <b>1 Number systems and codes</b>                              | <b>1</b>  |
| 1.1 Introduction                                               | 1         |
| 1.2 Number systems                                             | 2         |
| 1.3 Conversion between number systems                          | 3         |
| 1.4 Binary addition and subtraction                            | 5         |
| 1.5 Signed arithmetic                                          | 6         |
| 1.6 Complement arithmetic                                      | 7         |
| 1.7 Complement representation for binary numbers               | 7         |
| 1.8 The validity of 1's and 2's complement arithmetic          | 9         |
| 1.9 Offset binary representation                               | 10        |
| 1.10 Addition and subtraction of 2's complement numbers        | 10        |
| 1.11 Graphical interpretation of 2's complement representation | 12        |
| 1.12 Addition and subtraction of 1's complement numbers        | 13        |
| 1.13 Multiplication of unsigned binary numbers                 | 13        |
| 1.14 Multiplication of signed binary numbers                   | 14        |
| 1.15 Binary division                                           | 15        |
| 1.16 Floating point arithmetic                                 | 16        |
| 1.17 Binary codes for decimal digits                           | 18        |
| 1.18 $n$ -cubes and distance                                   | 19        |
| 1.19 Error detection and correction                            | 20        |
| 1.20 The Hamming code                                          | 21        |
| 1.21 Gray code                                                 | 23        |
| 1.22 The ASCII code                                            | 24        |
| <b>2 Boolean algebra</b>                                       | <b>28</b> |
| 2.1 Introduction                                               | 28        |
| 2.2 Boolean algebra                                            | 28        |
| 2.3 Derived Boolean operations                                 | 29        |
| 2.4 Boolean functions                                          | 29        |
| 2.5 Truth tables                                               | 30        |
| 2.6 The logic of a switch                                      | 30        |
| 2.7 The switch implementation of the AND function              | 31        |
| 2.8 The switch implementation of the OR function               | 32        |
| 2.9 The gating function of the AND and OR gates                | 33        |
| 2.10 The inversion function                                    | 33        |
| 2.11 Gate or switch implementation of a Boolean function       | 34        |
| 2.12 The Boolean theorems                                      | 34        |

---

|          |                                                                           |           |
|----------|---------------------------------------------------------------------------|-----------|
| 2.13     | Complete sets                                                             | 38        |
| 2.14     | The exclusive-OR (XOR) function                                           | 38        |
| 2.15     | The Reed–Muller equation                                                  | 39        |
| 2.16     | Set theory and the Venn diagram                                           | 40        |
| <b>3</b> | <b>Karnaugh maps and function simplification</b>                          | <b>43</b> |
| 3.1      | Introduction                                                              | 43        |
| 3.2      | Minterms and maxterms                                                     | 43        |
| 3.3      | Canonical forms                                                           | 44        |
| 3.4      | Boolean functions of two variables                                        | 45        |
| 3.5      | The Karnaugh map                                                          | 46        |
| 3.6      | Plotting Boolean functions on a Karnaugh map                              | 49        |
| 3.7      | Maxterms on the Karnaugh map                                              | 50        |
| 3.8      | Simplification of Boolean functions                                       | 51        |
| 3.9      | The inverse function                                                      | 54        |
| 3.10     | ‘Don’t care’ terms                                                        | 55        |
| 3.11     | Simplification of products of maxterms                                    | 57        |
| 3.12     | The Quine–McCluskey tabular simplification method                         | 58        |
| 3.13     | Properties of prime implicant tables                                      | 61        |
| 3.14     | Cyclic prime implicant tables                                             | 61        |
| 3.15     | Semi-cyclic prime implicant tables                                        | 63        |
| 3.16     | Quine–McCluskey simplification of functions containing ‘don’t care’ terms | 64        |
| 3.17     | Decimal approach to Quine–McCluskey simplification of Boolean functions   | 65        |
| 3.18     | Multiple output circuits                                                  | 67        |
| 3.19     | Tabular methods for multiple output functions                             | 70        |
| 3.20     | Reduced dimension maps                                                    | 73        |
| 3.21     | Plotting RDMs from truth tables                                           | 75        |
| 3.22     | Reading RDM functions                                                     | 76        |
| 3.23     | Looping rules for RDMs                                                    | 76        |
| 3.24     | Criteria for minimisation                                                 | 77        |
| <b>4</b> | <b>Combinational logic design principles</b>                              | <b>81</b> |
| 4.1      | Introduction                                                              | 81        |
| 4.2      | The NAND function                                                         | 81        |
| 4.3      | NAND logic implementation of AND and OR functions                         | 82        |
| 4.4      | NAND logic implementation of sums-of-products                             | 83        |
| 4.5      | The NOR function                                                          | 85        |
| 4.6      | NOR logic implementation of AND and OR functions                          | 86        |
| 4.7      | NOR logic implementation of products-of-sums                              | 86        |
| 4.8      | NOR logic implementation of sums-of-products                              | 87        |
| 4.9      | Boolean algebraic analysis of NAND and NOR networks                       | 88        |
| 4.10     | Symbolic circuit analysis for NAND and NOR networks                       | 89        |
| 4.11     | Alternative function representations                                      | 90        |
| 4.12     | Gate signal conventions                                                   | 91        |
| 4.13     | Gate expansion                                                            | 91        |
| 4.14     | Miscellaneous gate networks                                               | 92        |
| 4.15     | Exclusive-OR and Exclusive-NOR                                            | 94        |
| 4.16     | Noise margins                                                             | 98        |

---

|          |                                                     |            |
|----------|-----------------------------------------------------|------------|
| 4.17     | Propagation time                                    | 99         |
| 4.18     | Speed-power products                                | 100        |
| 4.19     | Fan-out                                             | 101        |
| <b>5</b> | <b>Combinational logic design with MSI circuits</b> | <b>105</b> |
| 5.1      | Introduction                                        | 105        |
| 5.2      | Multiplexers and data selection                     | 105        |
| 5.3      | Available MSI multiplexers                          | 106        |
| 5.4      | Interconnecting multiplexers                        | 106        |
| 5.5      | The multiplexer as a Boolean function generator     | 107        |
| 5.6      | Multi-level multiplexing                            | 111        |
| 5.7      | Demultiplexers                                      | 114        |
| 5.8      | Multiplexer/demultiplexer data transmission system  | 115        |
| 5.9      | Decoders                                            | 116        |
| 5.10     | Decoder networks                                    | 119        |
| 5.11     | The decoder as a minterm generator                  | 121        |
| 5.12     | Display decoding                                    | 122        |
| 5.13     | Encoder circuit principles                          | 123        |
| 5.14     | Available MSI encoders                              | 125        |
| 5.15     | Encoding networks                                   | 127        |
| 5.16     | Parity generation and checking                      | 129        |
| 5.17     | Digital comparators                                 | 131        |
| 5.18     | Iterative circuits                                  | 135        |
| <b>6</b> | <b>Latches and flip-flops</b>                       | <b>142</b> |
| 6.1      | Introduction                                        | 142        |
| 6.2      | The bistable element                                | 142        |
| 6.3      | The SR latch                                        | 143        |
| 6.4      | The controlled SR latch                             | 146        |
| 6.5      | The controlled D latch                              | 147        |
| 6.6      | Latch timing parameters                             | 148        |
| 6.7      | The JK flip-flop                                    | 149        |
| 6.8      | The master/slave JK flip-flop                       | 151        |
| 6.9      | Asynchronous controls                               | 152        |
| 6.10     | 1's and 0's catching                                | 153        |
| 6.11     | The master/slave SR flip-flop                       | 153        |
| 6.12     | The edge-triggered D flip-flop                      | 154        |
| 6.13     | The edge-triggered JK flip-flop                     | 157        |
| 6.14     | The T flip-flop                                     | 157        |
| 6.15     | Mechanical switch debouncing                        | 158        |
| 6.16     | Registers                                           | 159        |
| <b>7</b> | <b>Counters and registers</b>                       | <b>163</b> |
| 7.1      | Introduction                                        | 163        |
| 7.2      | The clock signal                                    | 163        |
| 7.3      | Basic counter design                                | 164        |
| 7.4      | Series and parallel connection of counters          | 166        |
| 7.5      | Scale-of-five up-counter                            | 167        |

---

|          |                                                                 |            |
|----------|-----------------------------------------------------------------|------------|
| 7.6      | The design steps for a synchronous counter                      | 170        |
| 7.7      | Gray code counters                                              | 170        |
| 7.8      | Design of decade Gray code up-counter                           | 172        |
| 7.9      | Scale-of-16 up/down counter                                     | 172        |
| 7.10     | Asynchronous binary counters                                    | 173        |
| 7.11     | Decoding of asynchronous counters                               | 176        |
| 7.12     | Asynchronous resettable counters                                | 177        |
| 7.13     | Integrated circuit counters                                     | 178        |
| 7.14     | Cascading of IC counter chips                                   | 182        |
| 7.15     | Shift registers                                                 | 183        |
| 7.16     | The 4-bit 7494 shift register                                   | 185        |
| 7.17     | The 4-bit 7495 universal shift register                         | 186        |
| 7.18     | The 74165 parallel loading 8-bit shift register                 | 187        |
| 7.19     | The use of shift registers as counters and sequence generators  | 187        |
| 7.20     | The universal state diagram for shift registers                 | 188        |
| 7.21     | The design of a decade counter                                  | 189        |
| 7.22     | The ring counter                                                | 191        |
| 7.23     | The twisted ring or Johnson counter                             | 192        |
| 7.24     | Series and parallel interconnection of Johnson counters         | 195        |
| 7.25     | Shift registers with XOR feedback                               | 196        |
| 7.26     | Multi-bit rate multipliers                                      | 200        |
| <b>8</b> | <b>Clock-driven sequential circuits</b>                         | <b>207</b> |
| 8.1      | Introduction                                                    | 207        |
| 8.2      | The basic synchronous sequential circuit                        | 207        |
| 8.3      | Analysis of a clocked sequential circuit                        | 207        |
| 8.4      | Design steps for synchronous sequential circuits                | 210        |
| 8.5      | The design of a sequence detector                               | 215        |
| 8.6      | The Moore and Mealy state machines                              | 217        |
| 8.7      | Analysis of a sequential circuit implemented with JK flip-flops | 221        |
| 8.8      | Sequential circuit design using JK flip-flops                   | 223        |
| 8.9      | State reduction                                                 | 225        |
| 8.10     | State assignment                                                | 229        |
| 8.11     | Algorithmic state machine charts                                | 232        |
| 8.12     | Conversion of an ASM chart into hardware                        | 235        |
| 8.13     | The 'one-hot' state assignment                                  | 237        |
| 8.14     | Clock skew                                                      | 238        |
| 8.15     | Clock timing constraints                                        | 239        |
| 8.16     | Asynchronous inputs                                             | 240        |
| 8.17     | The handshake                                                   | 242        |
| <b>9</b> | <b>Event driven circuits</b>                                    | <b>248</b> |
| 9.1      | Introduction                                                    | 248        |
| 9.2      | Design procedure for asynchronous sequential circuits           | 248        |
| 9.3      | Stable and unstable states                                      | 249        |
| 9.4      | Design of a lamp switching circuit                              | 250        |
| 9.5      | Races                                                           | 252        |
| 9.6      | Race free assignments                                           | 254        |

---

|           |                                                            |            |
|-----------|------------------------------------------------------------|------------|
| 9.7       | The pump problem                                           | 256        |
| 9.8       | Design of a sequence detector                              | 258        |
| 9.9       | State reduction for incompletely specified machines        | 264        |
| 9.10      | Compatibility                                              | 264        |
| 9.11      | Determination of compatible pairs                          | 265        |
| 9.12      | The merger diagram                                         | 267        |
| 9.13      | The state reduction procedure                              | 267        |
| 9.14      | Circuit hazards                                            | 268        |
| 9.15      | Gate delays                                                | 268        |
| 9.16      | The generation of spikes                                   | 268        |
| 9.17      | The generation of static hazards in combinational networks | 270        |
| 9.18      | The elimination of static hazards                          | 271        |
| 9.19      | Design of hazard-free combinational networks               | 273        |
| 9.20      | Detection of hazards in an existing network                | 275        |
| 9.21      | Hazard-free asynchronous circuit design                    | 277        |
| 9.22      | Dynamic hazards                                            | 279        |
| 9.23      | Function hazards                                           | 281        |
| 9.24      | Essential hazards                                          | 282        |
| <b>10</b> | <b>Instrumentation and interfacing</b>                     | <b>287</b> |
| 10.1      | Introduction                                               | 287        |
| 10.2      | Schmitt trigger circuits                                   | 287        |
| 10.3      | Schmitt input gates                                        | 289        |
| 10.4      | Digital-to-analogue conversion                             | 292        |
| 10.5      | Analogue-to-digital conversion                             | 295        |
| 10.6      | Flash converters                                           | 296        |
| 10.7      | Integrating A/D converter types                            | 298        |
| 10.8      | A/D converter types using an embedded D/A converter        | 299        |
| 10.9      | Shaft encoders and linear encoders                         | 303        |
| 10.10     | Sensing of motion                                          | 304        |
| 10.11     | Absolute encoders                                          | 306        |
| 10.12     | Conversion from Gray code to base 2                        | 309        |
| 10.13     | Petherick code                                             | 311        |
| 10.14     | Incremental encoders                                       | 312        |
| 10.15     | Open collector and tri-state gates                         | 314        |
| 10.16     | Use of open collector gates                                | 316        |
| 10.17     | Use of tri-state buffers and gates                         | 320        |
| 10.18     | Other interfacing components                               | 322        |
| <b>11</b> | <b>Programmable logic devices</b>                          | <b>326</b> |
| 11.1      | Introduction                                               | 326        |
| 11.2      | Read only memory                                           | 326        |
| 11.3      | ROM timing                                                 | 329        |
| 11.4      | Internal ROM structure                                     | 330        |
| 11.5      | Implementation of Boolean functions using ROMs             | 331        |
| 11.6      | Internal addressing techniques in ROMs                     | 334        |
| 11.7      | Memory addressing                                          | 335        |
| 11.8      | Design of sequential circuits using ROMs                   | 337        |

---

|           |                                                           |            |
|-----------|-----------------------------------------------------------|------------|
| 11.9      | Programmable logic devices (PLDs)                         | 337        |
| 11.10     | Programmable gate arrays (PGAs)                           | 339        |
| 11.11     | Programmable logic arrays (PLAs)                          | 341        |
| 11.12     | Programmable array logic (PAL)                            | 346        |
| 11.13     | Programmable logic sequencers (PLSs)                      | 349        |
| 11.14     | Field programmable gate arrays (FPGAs)                    | 355        |
| 11.15     | Xilinx field programmable gate arrays                     | 357        |
| 11.16     | Actel programmable gate arrays                            | 361        |
| 11.17     | Altera erasable programmable logic devices                | 363        |
| <b>12</b> | <b>Arithmetic circuits</b>                                | <b>367</b> |
| 12.1      | Introduction                                              | 367        |
| 12.2      | The half adder                                            | 367        |
| 12.3      | The full adder                                            | 368        |
| 12.4      | Binary subtraction                                        | 370        |
| 12.5      | The 4-bit binary full adder                               | 371        |
| 12.6      | Carry look-ahead addition                                 | 372        |
| 12.7      | The 74283 4-bit carry look-ahead adder                    | 373        |
| 12.8      | Addition/subtraction circuits using complement arithmetic | 376        |
| 12.9      | Overflow                                                  | 377        |
| 12.10     | Serial addition and subtraction                           | 378        |
| 12.11     | Accumulating adder                                        | 380        |
| 12.12     | Decimal arithmetic with MSI adders                        | 381        |
| 12.13     | Adder/subtractor for decimal arithmetic                   | 383        |
| 12.14     | The 7487 true/complement unit                             | 385        |
| 12.15     | Arithmetic/logic unit design                              | 386        |
| 12.16     | Available MSI arithmetic/logic units                      | 390        |
| 12.17     | Multiplication                                            | 392        |
| 12.18     | Combinational multipliers                                 | 393        |
| 12.19     | ROM implemented multiplier                                | 394        |
| 12.20     | The shift and add multiplier                              | 396        |
| 12.21     | Available multiplier packages                             | 401        |
| 12.22     | Signed arithmetic                                         | 401        |
| 12.23     | Booth's algorithm                                         | 403        |
| 12.24     | Implementation of Booth's algorithm                       | 404        |
| <b>13</b> | <b>Fault diagnosis and testing</b>                        | <b>408</b> |
| 13.1      | Introduction                                              | 408        |
| 13.2      | Fault detection and location                              | 409        |
| 13.3      | Gate sensitivity                                          | 412        |
| 13.4      | A fault test for a 2-input AND gate                       | 413        |
| 13.5      | Path sensitisation                                        | 414        |
| 13.6      | Path sensitisation in networks with fan-out               | 416        |
| 13.7      | Undetectable faults                                       | 419        |
| 13.8      | Bridging faults                                           | 422        |
| 13.9      | The fault detection table                                 | 423        |
| 13.10     | Two-level circuit fault detection in AND/OR circuits      | 428        |
| 13.11     | Two-level circuit fault detection in OR/AND circuits      | 432        |

---

|                 |                                                             |            |
|-----------------|-------------------------------------------------------------|------------|
| 13.12           | Boolean difference                                          | 435        |
| 13.13           | Compact testing techniques                                  | 441        |
| 13.14           | Signature analysis                                          | 442        |
| 13.15           | The scan path testing technique                             | 444        |
| 13.16           | Designing for testability                                   | 447        |
| <b>Appendix</b> | <b>Functional logic symbols</b>                             | <b>451</b> |
| A.1             | Introduction                                                | 451        |
| A.2             | Basic principles of the functional symbol system            | 451        |
| A.3             | Dependency notation                                         | 454        |
| A.4             | Simple examples of G dependency in functional logic symbols | 455        |
| A.5             | Control, Set, and Reset dependency                          | 457        |
| A.6             | Bistable logic elements and C dependency                    | 459        |
| A.7             | Counters, Z and M dependency                                | 461        |
| A.8             | Shift registers                                             | 463        |
| A.9             | Programmable devices and A dependency                       | 464        |
| A.10            | Arithmetic circuits and N dependency                        | 465        |
|                 | <i>Answers to problems</i>                                  | 467        |
|                 | <i>Bibliography</i>                                         | 498        |
|                 | <i>Index</i>                                                | 500        |

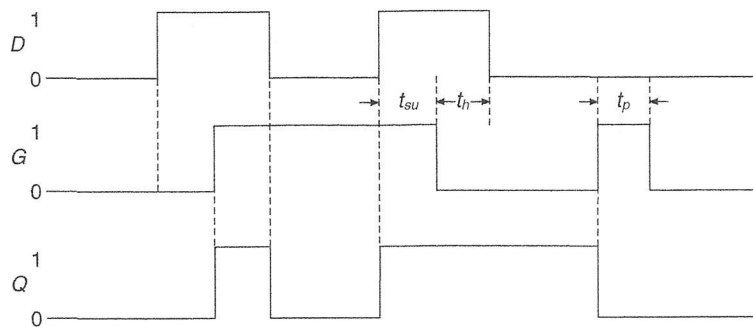
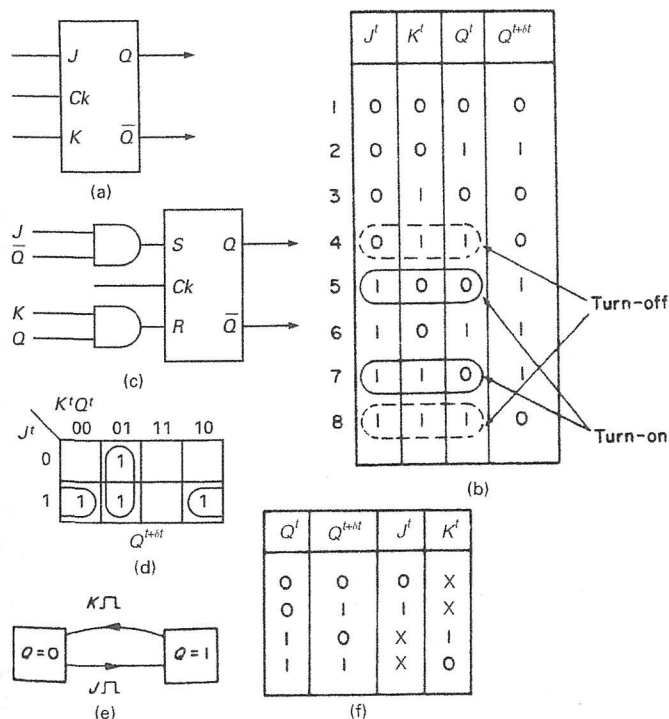


Figure 6.13 Set-up time and hold time

## 6.7 The JK flip-flop

The latch circuits previously described are not suitable for operation in synchronous sequential circuits because of their transparency. For synchronous circuits a clock signal is provided which governs the time at which the outputs of the memory elements are allowed to change state. In a synchronous circuit, flip-flops are used as the basic memory element, a typical example being the JKFF. Unlike latches, they only respond to a transition on a clock input or to a change in an asynchronous input such as Clear.

The symbolic representation of the JKFF is shown in Figure 6.14(a) and the state table describing its logical operation is in Figure 6.14(b). The logical operation of this

Figure 6.14 The JK flip-flop (a) symbolic representation (b) state table (c) representation of JK flip-flop by an SR latch and two AND gates (d) K-map plot of  $Q^{t+dt}$  (e) state diagram (f) steering table

flip-flop differs in one respect from that of the SR latch in that it is allowable for  $J$  and  $K$  to be simultaneously equal to 1. When  $J = K = 1$ , the flip-flop *toggles*, i.e. in row 7 the flip-flop changes state from 0 to 1, while in row 8 the converse action takes place. In rows 4 and 5 normal reset and set operations take place, as described for the SR latch in section 6.3.

An examination of the state table shows that the flip-flop is turned on in rows 5 and 7, while it is turned off in rows 4 and 8. The turn on condition for  $Q$  is

$$\begin{aligned} S &= J\bar{K}\bar{Q} + JK\bar{Q} \\ &= J\bar{Q} \end{aligned}$$

The turn off condition for  $Q$  is

$$\begin{aligned} R &= \bar{J}KQ + JKQ \\ &= KQ \end{aligned}$$

These two equations indicate that a JK flip-flop may be regarded as an SR latch preceded by two AND gates which implement the turn-on and turn-off functions respectively, as illustrated in Figure 6.14(c).

The characteristic equation of the JK flip-flop is obtained by plotting the present state conditions on the K-map shown in Figure 6.14(d). After simplification, the characteristic equation can be written as

$$Q^{t+\delta t} = (J\bar{Q} + \bar{K}Q)$$

The state diagram describing the terminal behaviour of the flip-flop is shown in Figure 6.14(e). Assuming that the flip-flop is clocked and is presently in the state  $Q = 0$  with  $J = 1$  and  $Ck$  changing from 0 to 1, it makes a transition to the state  $Q = 1$ . Similarly, in the state  $Q = 1$  with  $K = 1$  and  $Ck$  changing from 0 to 1 it makes a transition to  $Q = 0$ .

A steering table for the JK flip-flop derived from the state stable is shown in Figure 6.14(f). Comparing the steering table of the SR latch and the JK flip-flop in Figures 6.7 and 6.14(f), it will be noticed that the JK flip-flop has more 'X' or 'don't care' input conditions. In practice, the increased number of 'don't care' terms leads to simpler combinational logic when designing a sequential logic circuit.

A JK flip-flop can be implemented by connecting the output of two AND gates in Figure 6.14(c) to the  $S$  and  $R$  inputs of the controlled latch shown in Figure 6.10(a). The  $Q$  and  $\bar{Q}$  outputs of this latch and its clock connections are fed to the inputs of the two AND gates in conjunction with the  $J$  and  $K$  inputs, as shown in Figure 6.15(a). Notice that the AND gates are formed from two pairs of NAND gates in cascade, namely  $g_5$  and  $g_7$ , and  $g_6$  and  $g_8$ . Clearly, gates  $g_7$  and  $g_1$ , and gates  $g_8$  and  $g_2$ , give a double inversion and are redundant, thus reducing the JKFF to an array of four gates only, as shown in Figure 6.15(b).

As in the case of the controlled latches described earlier in this chapter, the flip-flop is disabled when  $Ck = 0$  and is active when  $Ck = 1$ . Unfortunately, the connection shown in Figure 6.15(b) exhibits instability when  $J = K = 1$  and  $Ck = 1$  due to the feedback of the complementary output signals to the input. The state diagram indicates that under these conditions the  $Q$  output is oscillatory and will remain so until such time as the  $Ck$  makes a  $1 \rightarrow 0$  transition when the clock is disabled.

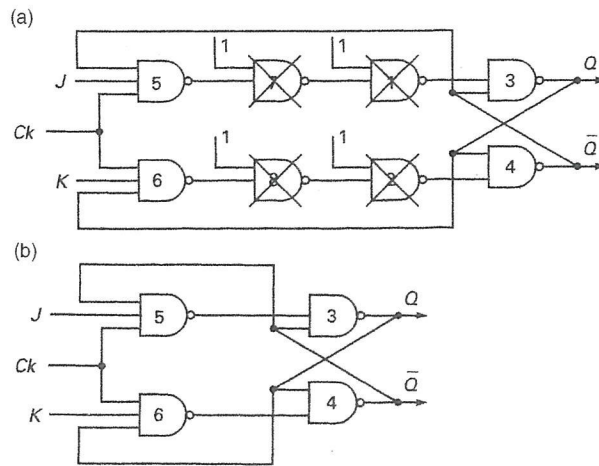


Figure 6.15 (a) NAND implementation of JK flip-flop and (b) its reduced form

### 6.8 The master/slave JK flip-flop

In order to overcome the difficulties described in the previous section, a master/slave JKFF can be used. It consists of two SR latches, the master and the slave, connected in cascade, as shown in Figure 6.16(a). The master is clocked in the normal way, while the clock signal to the slave is inverted. Assuming that changes in the  $J$  and  $K$  signals are only allowed to occur when the clock is low, the master then being disabled, changes in its output will take place on the rising edge of the clock pulse and these changes are transmitted to the input of the slave. However, no change can occur at the output of the slave until the rising edge of the inverted clock pulse, which is the trailing edge of the

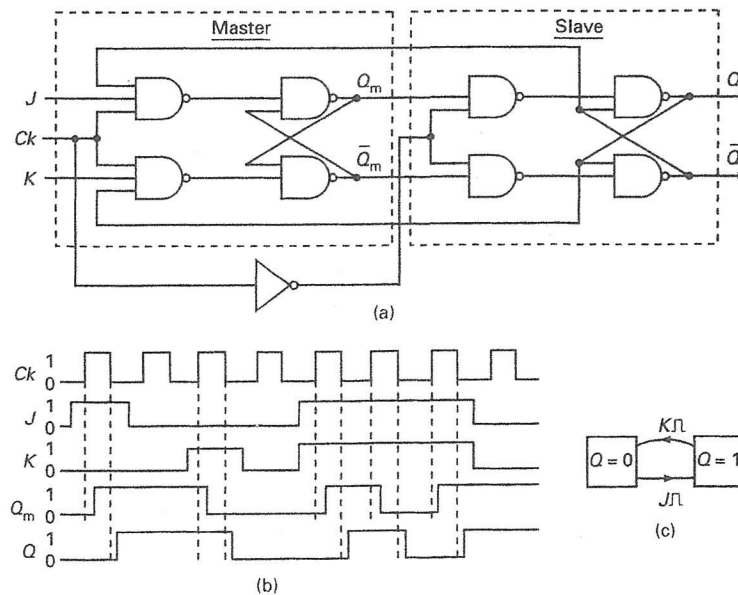


Figure 6.16 (a) The master/slave JK flip-flop (b) The timing diagram (c) The external state diagram

clock pulse. Consequently changes in  $Q$  and  $\bar{Q}$  which are fed back to the input of the master do not take place until the trailing edge of the clock pulse arrives. In terms of the state diagram shown in Figure 6.16(c), when  $J = K = 1$ , the transition of  $Q$  from  $0 \rightarrow 1$  is made on the trailing edge of the clock pulse. The flip-flop will remain in that state until the trailing edge of the next clock pulse when the reverse transition will take place. The flip-flop is then said to be operating in a *toggling* mode which is analogous to the unstable oscillatory condition described earlier. However, the toggling of  $Q$  is now controlled while the condition  $J = K = 1$  is maintained, and the flip-flop will toggle on the trailing edge of each successive clock pulse. Timing diagrams for the master/slave JKFF are given in Figure 6.16(b).

### 6.9 Asynchronous controls (direct preset and clear)

As well as the  $J$ ,  $K$  and  $Ck$  inputs a master/slave JKFF may have one or two additional controls which allow both the master and the slave to assume one of their two states irrespective of whether  $Ck = 0$  or 1. These asynchronous controls are usually called *preset* and *clear*. Most commercially available flip-flops are provided with a clear control whereas the preset control is not quite as common. The operation of these controls is described in the table shown in Figure 6.17(a) and a circuit including these controls appears in Figure 6.17(b).

With both controls at 0 they are inactive, and the flip-flop is under the control of  $J$ ,  $K$  and  $Ck$ . If  $Cl = 1$  and  $Pr = 0$  both master and slave are cleared so that  $Q_m = Q = 0$ . If  $Cl = 0$  and  $Pr = 1$  the flip-flop is preset and  $Q_m = Q = 1$ . Active high signals on the  $Pr$  and  $Cl$  lines will override signals on the  $J$  and  $K$  lines. These signals are normally only used during the asynchronous periods when the clock is low. Typically, the clear control might be used to clear all the flip-flops in an array when the power is first switched on.

|           | $Cl$ | $Pr$ | $Q$ |
|-----------|------|------|-----|
| Forbidden | 1    | 1    |     |
| Clear     | 1    | 0    | 0   |
| Preset    | 0    | 1    | 1   |
| Enable    | 0    | 0    | X   |

(a)

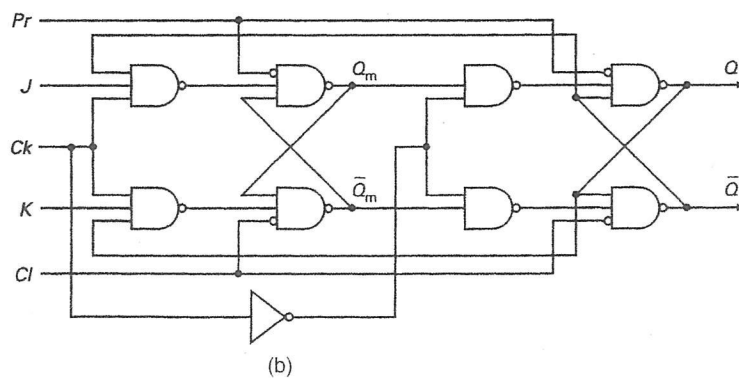


Figure 6.17 (a)  $Pr$  and  $Cl$  control tabulation (b) Master/Slave JK flip-flop with  $Pr$  and  $Cl$  controls

---

## 7 Counters and registers

---

### 7.1 Introduction

A counting circuit composed of memory elements, such as flip-flops and electronic gates, is the simplest form of sequential circuit available. All sequential circuits are of two types, (1) *synchronous* (clock driven) and (2) *asynchronous* (event driven). In synchronous circuits, changes in the circuit state are synchronised to the normally periodic clock pulses, whereas in event driven circuits state changes are governed by events such as, for example, the occurrence of a system fault.

Counting circuits can be in either of the two categories described above. All counter circuits count clock pulses and store the number received in an array of memory elements. In the case of synchronous counters the flip-flops are all clocked at precisely the same instant in time, whereas in an asynchronous circuit only the least significant stage is clocked, and succeeding flip-flops are clocked at later times which depend on the flip-flop propagation times. Design of synchronous counters is generally more complex than that of asynchronous counters.

Counters are fundamental and important components of a digital system and can be used for timing, control or sequencing operations. Alternatively, they can be used for frequency division and in some cases there may be a non-binary count, for example a Gray code counter or a BCD counter. In practice it would be most unusual for the logic designer to design a counter circuit since there are a large number available on MSI chips. Nevertheless, it is important that the reader should be aware of the basic design techniques employed.

### 7.2 The clock signal

An essential feature of a synchronous system is that flip-flops which are part of the system should all change at the same instant in time. This is achieved by the use of a synchronising signal which is formally known as the *clock*. The clock signal is normally periodic, and there must always be a sufficient time period between adjacent clock pulses to ensure that the combinational logic has reached its steady state condition before the next clock pulse in the sequence arrives.

In general, as shown in Figure 7.1, an idealised form of clock signal will approximate to a square wave, and the period in the cycle when the clock is high is termed the active period. Flip-flop transitions, initiated by the synchronising clock, are arranged to take place on either the leading edge or, alternatively, the trailing edge of a clock pulse, and these two types of flip-flop operation should not be used in the same circuit. The active clock edge will initiate a change of state in a synchronous circuit providing there is no

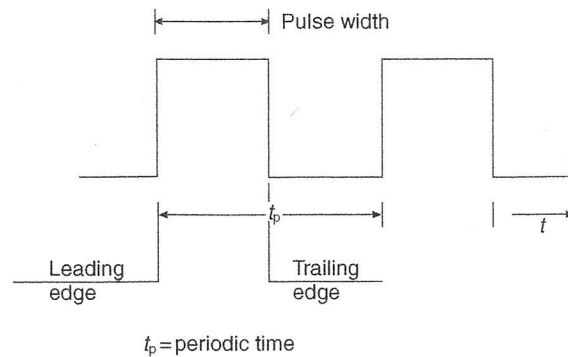


Figure 7.1 The clock pulse

other external input to the circuit. It should be recognised that transitions in clock values from low to high and vice versa are never instantaneous, but providing that flip-flop changes take place during the course of the transition, the conditions for synchronous operation are satisfied.

### 7.3 Basic counter design

The simplest possible counter is the scale-of-two counter which has only two states 0 and 1. Since the output of the flip-flop can only exist in one of these states at any time, the counter can be implemented with a single flip-flop.

One design technique is to draw up a state table in which the first column represents the present state of the counter while the second column gives the next state of the counter after the arrival of a clock pulse, as shown in Figure 7.2(a). The table identifies

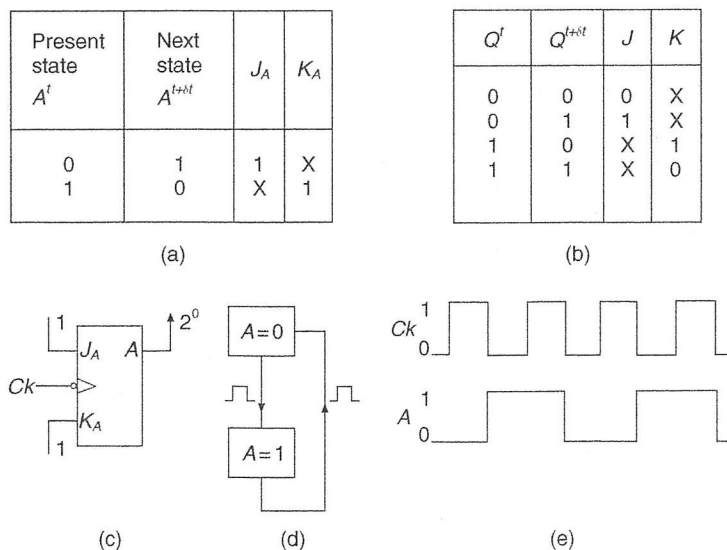


Figure 7.2 Scale-of-two counter (a) State table (b) Steering table of JK flip-flop (c) Implementation (d) State diagram (e) Timing diagram